TTC標準
Standard

JT-G711.0

G. 711パルス符号変調向け ロスレス符号化

Lossless compression of G.711 pulse code modulation

第 1 版

2011 年 2 月 23 日制定

社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

目　　次

＜参考＞

０．本標準の概要

本標準は、ＩＴＵ－Ｔ勧告Ｇ．７１１ビット列のロスレス符号化を行うＪＴ－Ｇ７１１．０符号化アルゴリズムを記述したものである。

符号化器は、40、80、160、240、320 サンプルのフレーム長で処理を行い、最大アルゴリズム遅延はそのフレーム長となる。また、ワーストケースの計算量は、符号化器と復号器を合わせて 1.7 WMOPS（weighted million operations per second）以下である。

１． 国際勧告等との関連

本標準は、２００９年９月に承認されたＩＴＵ－Ｔ勧告Ｇ．７１１．０に準拠したものである。

２．上記国際勧告等に対する追加項目等

２．１ オプション選択項目

な し

２．２ ナショナルマター決定項目

な し

２．３ 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。

２．４ 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。

３．改版の履歴

版 数	制 定 日	改 版 内 容
第 1 版	2011 年 2 月 23 日	制定

４．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。

５．その他

(1) 参照している勧告、標準等

ＩＴＵ－Ｔ勧告： Ｇ．１９１、Ｇ．７１１

(2) ＴＴＣ標準ＪＴ－Ｇ７１１．０は、ＩＴＵ－Ｔ勧告Ｇ．７１１．０に準拠しており、本標準中で言及しているＣコードおよびテストシーケンスとは、ＩＴＵ－Ｔ勧告Ｇ．７１１．０のものをさし、ＩＴＵ－ＴのＷｅｂ

サイトから入手可能である。

(3) 本標準においては、上記ＩＴＵ－Ｔ勧告Ｇ．７１１．０に記載されているＡ則符号化方式に対する事項も便宜上、記載されているが、我が国ではμ則符号化方式のみが採用されており、Ａ則符号化方式は使用されていないことに留意されたい。またＧ．７１１．０にＡ則符号化方式に対する記載もあることから、本標準でベースとする符号化標準として、ＩＴＵ－Ｔ勧告であるＧ．７１１を参照している。なお、Ｇ．７１１のμ則符号化方式に関しては、ＴＴＣ標準ＪＴ－Ｇ７１１で規定され、その記述がある。

６．標準作成部門

メディア符号化専門委員会

1　本標準の規定範囲

本標準は、ＩＴＵ－Ｔ Ｇ．７１１ビット列向けロスレス符号化の概要について記載している。

本標準の構成は次の通りである。本標準を通して使用されている文献、定義、略語、頭字語、表記法については、第２章、第３章、第４章、第５章のそれぞれで定義している。第６章ではＪＴ－Ｇ７１１．０アルゴリズムの概略について述べている。ＪＴ－Ｇ７１１．０符号化器及び復号器の原理については第７章、第８章のそれぞれで述べている。第９章では、16-32 ビット固定小数点演算で本符号化アルゴリズムを定義したソフトウェアについて記載している。

本標準の固定小数点演算による参考実装例、及び前記参照実装例向けの非網羅的なテスト信号は、ＩＴＵ－ＴのＷｅｂサイトから入手可能である。

2　参照文献

下記のＩＴＵ－Ｔ勧告は、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。出版された時点でその版が適用され、全ての標準および他の参照文献は、改定に従うものとする。従って、本標準のユーザには、以下の標準やその他の参照すべき文献について、最新の版の適用の可能性を調査するよう奨励される。現在有効なＩＴＵ－Ｔ勧告のリストは定期的に出版されている。

本標準内での文書の参照は、単独の文書としては、それを標準の扱いとはしない。

－　ＩＴＵ－Ｔ勧告Ｇ．１９１（２００５）
　Software tools for speech and audio coding standardization

－　ＩＴＵ－Ｔ勧告Ｇ．７１１（１９８８）
　Pulse code modulation (PCM) of voice frequencies

3　定義

本標準では新たな用語の定義を行っていない。

4　略語と頭字語

本標準では、下記の略語、及び頭字語を使用している。

IP	Internet Protocol	インターネットプロトコル
LLC	LossLess Compression	ロスレス符号化
LP	Linear Prediction	線形予測
LPC	Linear Predictive Coding	線形予測符号化
LSB	Least Significant Bit	最下位ビット
LTP	Long-Term Prediction	長期予測
MSB	Most Significant Bit	最上位ビット
PARCOR	PARtial autoCORrelation	偏自己相関
PCM	Pulse Code Modulation	パルス符号変調
VoIP	Voice over IP	ボイスオーバーＩＰ
WMOPS	Weighted Million Operations Per Second	重み付けＭＯＰＳ

5　表記法

表記法の詳細は次の通りである。

(1) 時間領域の信号は、そのシンボルと丸括弧で括られたサンプル番号で記述する(例　*s(n)*)。変数*n*は、サンプル番号である。

(2) 記号 ^ は量子化されたパラメータ値（例　$\hat{g}_c$）、あるいは予測されたサンプル値（例　$\hat{x}(n)$）である。

(3) パラメータの範囲は、角括弧で括られた値で記述する。この値は境界値を含む(例　［0.6，0.9］)。境界の片側が丸括弧を用いて表記されている時は、境界の終端部は範囲には含まれない（例　［0.6，0.9)）。

(4) 極性関数*sgn(x)*は、以下のように引数の極性を示す。

$$sgn(x) = \begin{cases} 1 & if\ x \geq 0 \\ -1 & if\ x < 0 \end{cases}$$

(5) 演算子 $max\{x(n) | n = 0,...,N-1\}$ は*x(n)*の最大値を示す。

(6) 演算子 $min\{x(n) | n = 0,...,N-1\}$ は*x(n)*の最小値を示す。

(7) 整数演算子$\lceil x \rceil$は正の方向への丸めを示すceiling関数である(例　$\lceil x \rceil = min\{n \in \{...,-2,-1,0,1,2,...\} | x \leq n\}$)。

(8) 整数演算子$\lfloor x \rfloor$は負の方向への丸めを示すfloor関数である(例　$\lfloor x \rfloor = max\{n \in \{...,-2,-1,0,1,2,...\} | x \geq n\}$)。

(9) ビット演算子⊗及び⊕は、それぞれANDビット演算子、XORビット演算子を示す。

(10) 接頭値0x付きの定数は16進数で表記された値である。

(11) 変数*x*のNビット右シフト演算は、2の－N乗のフロア値（$\lfloor 2^{-N} x \rfloor$）として示す

(12) 16ビット固定小数点ＡＮＳＩ Cにおいては、浮動小数点値はそれぞれ16/32ビット表現に丸めた値とする。

Table 5-1／JT-G711.0 は、本標準で用いられている主な記号のリストである。

本標準においては、正零、及び負零（それぞれ 0^+、0^-と表記される）は、Ｇ．７１１復号器にて最小値として復号される正／負の値に対するＧ．７１１符号化値に、それぞれ対応する。

Table 5-1／JT-G711.0 – Glossary of most relevant symbols

(ITU-T G.711.0)

Type	Name	Description
Signals	$I_A(n)$	Input ITU-T G.711 A-law samples to be encoded
	$I_\mu(n)$	Input ITU-T G.711μ-law samples to be encoded
	$x_A(n)$	ITU-T G.711 A-law samples
	$x_\mu(n)$	ITU-T G.711 μ-law samples
	$x_{int8}(n)$	Samples in int8 domain ($-128 \leq x_{int8}(n) \leq 127$)
	$\hat{x}_{int8}(n)$	Predicted samples in int8 domain
	$x_{PCM}(n)$	Samples in uniform (linear) PCM domain
	$\hat{x}_{PCM}(n)$	Predicted samples in uniform (linear) PCM domain
	$\tilde{x}_{PCM}(n)$	Windowed samples of $x_{PCM}(n)$
	$\bar{x}_{PCM}(n)$	Down-sampled PCM signal
	$\bar{x}_{LTP}(n)$	LTP contribution
	$\hat{x}_{LTP}(n)$	LTP predicted PCM signal
	$r(n)$	Linear prediction residual samples in *int8* domain
	$r_{PCM}(n)$	Short-term prediction residual samples in uniform PCM domain
	$r_{LTP}(n)$	Long-term prediction residual samples in *int8* domain

Table 5-1／JT-G711.0 – Glossary of most relevant symbols

(ITU-T G.711.0)

Type	Name	Description
Coefficients	$c(i)$	Autocorrelation coefficients of the windowed signal $\tilde{x}_{\mathrm{PCM}}(n)$
	$c'(i)$	Bandwidth expanded autocorrelation coefficients
	k_i	PARCOR coefficients
	$\hat{k}_i$	Quantized PARCOR coefficients
	$a_i^{[j]}$	Linear prediction coefficients for *j*-th prediction
Parameters	n	Index of sample
	N	Number of samples in a frame
	R	Data range
	X_{MIN}	The minimum sample value of $x_{\mathrm{int8}}(n)$
	X_{MAX}	The maximum sample value of $x_{\mathrm{int8}}(n)$
	X_{ANCHOR}	An anchor value
	P	Prediction order
	$\hat{P}$	Quantized prediction order
	P_{MAX}	Maximum possible prediction order
	$p(j)$	Candidate values of prediction order
	N_{sfr}	Number of sub-frames in LTP analysis
	N'_{sfr}	Number of sub-frames in prediction residual coding
Special values	0^+	Denotes plus-zero value ("zero" decoder output for positive PCM input value: 0xd5 for A-law and 0xff for μ-law)
	0^-	Denotes minus-zero value ("zero" G.711 decoder output for negative PCM input value: 0x55 for A-law and 0x7f for μ-law)
	0_m	More zero: The value of 0^+ or 0^- which has more occurrences in the frame
	0_l	Less zero: The value of 0^+ or 0^- which has less occurrences in the frame
Other	*N / A*	Not applicable

6　JT-G711.0 コーデック概略

本標準のＪＴ－Ｇ７１１ロスレスコーデックは、ＩＴＵ－Ｔソフトウェアツールライブラリ ＳＴＬ２００５ ｖ２．２（ＩＴＵ－Ｔ勧告Ｇ．１９１）の基本演算子を用いて固定小数点演算において実装されている。本標準は詳細なアルゴリズム記述を提供するものである。

本標準において使用されているロスレス圧縮技術は、Ｇ．７１１符号が音声やオーディオのような平均値ゼロの音源のディジタル表現である場合に、最も効率的となる。この場合、全体的な圧縮が期待される。しかし、ＪＴ－Ｇ７１１．０は、正規化の仮定とは独立に、Ｇ．７１１に規定されているいかなる符号の入力に対しても符号化するように設計されている。実装上の処理演算量を制限するために、使用される符号化技術も制限されている。

ＪＴ－Ｇ７１１．０コーデックは、また、直前の状態に依存しないように設計されている。つまり、Ｇ．７１１記号の入力フレームの符号化処理は、直前の、あるいは先読みのＧ．７１１フレームとは独立している。

この特性により、誤り伝播がないことが保証される。過去の情報を使うことにより高い圧縮率を得ることは出来るが、直前の状態に依存しないという制約の設計により、本標準はパケットベースのアプリケーションに対して非常に適したものとなっている。

6.1　符号化器

符号化器のハイレベルブロック図を Figure 6-1／JT-G711.0 に示す。

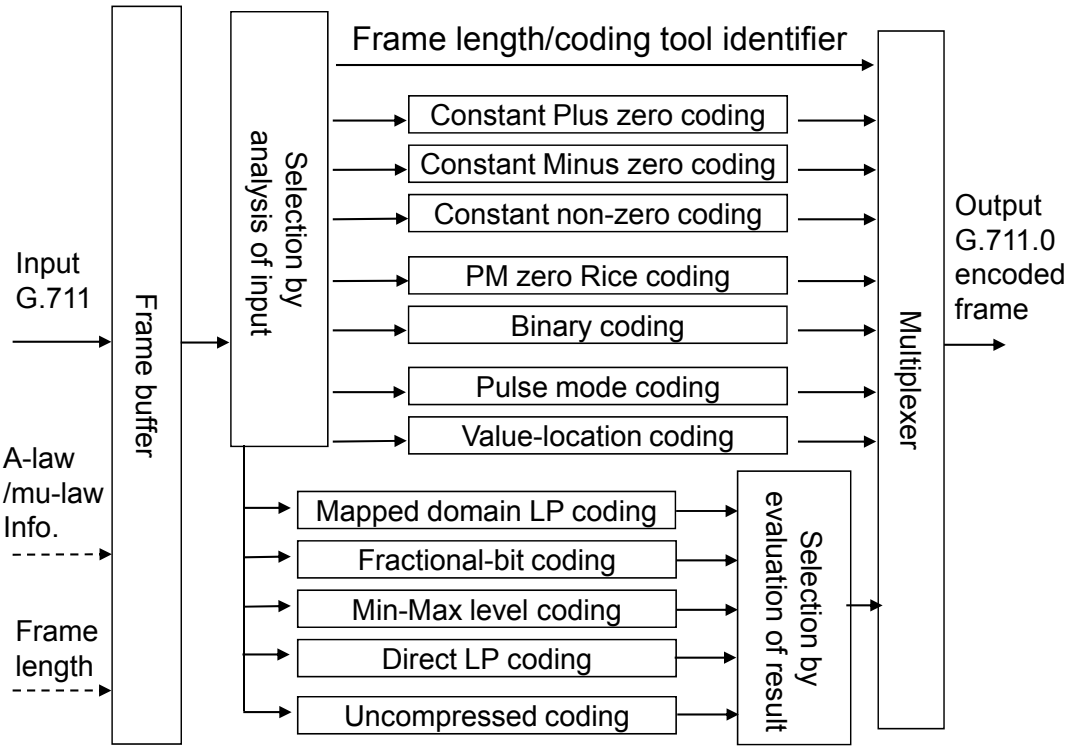

Figure 6-1 / JT-G711.0 – High-level encoder block diagram

(ITU-T G.711.0)

本標準は、ＪＴ－Ｇ７１１．０符号化出力フレームを生成するために、Figure 6-1／JT-G711.0 に示した符号化方法の１つを選択する。この選択に先立ち、１つ以上の符号化方法が実行、あるいは部分的に実行される。決定処理は７．３節にて詳細に述べられる。方法選択の後に、接頭符号が選択された符号化情報の前に置かれ、ＪＴ－Ｇ７１１．０符号化出力フレームの一部として送信される。接頭符号には、フレーム長、選択された符号化方法、及び選択された符号化方法に指定されるその他の副情報を定義する情報が含まれる。接頭符号の定義は、フレーム長に関しては７．１節に、符号化方法に関しては７．２節にそれぞれ述べられる。

本標準では、以後、特定の符号化方法を用いて符号化／復号するために必要な計算／演算機構を示すために“ツール（tool)”という用語を用いる。例えば、パルスモード符号化は、パルスモード符号化ツールにより実現される。

6.2　復号器

復号器のハイレベルブロック図を Figure 6-2／JT-G711.0 に示す。

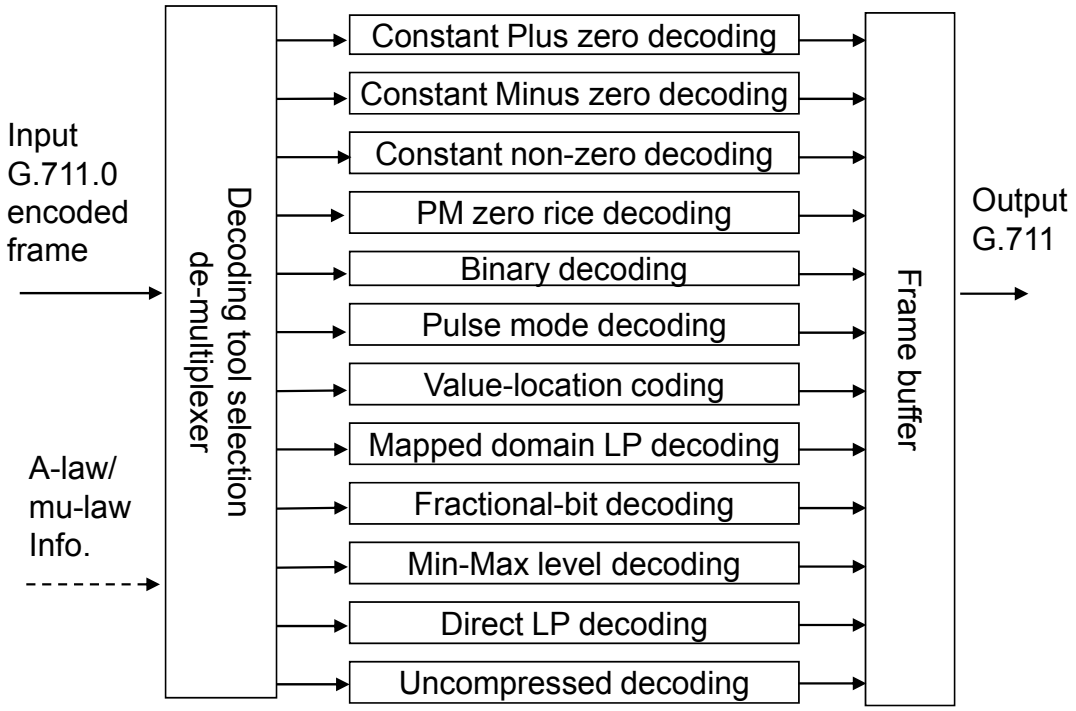

Figure 6-2 / JT-G711.0 – High-level decoder block diagram

(ITU-T G.711.0)

ＪＴ－Ｇ７１１．０復号器は受信したＪＴ－Ｇ７１１．０符号フレーム中の接頭符号を読み、必要な情報を適切な復号ツールに渡す。復号ツールは、受信したＪＴ－Ｇ７１１．０符号化フレームを使ってＧ．７１１サンプルを再生成する。

6.3　サポートされるフレーム長

ＪＴ－Ｇ７１１．０符号器は、1 フレームあたり 40、80、160、240、320 サンプル長の５つのフレーム長をサポートする。フレーム間の依存性が無いため、フレーム長は毎フレーム変更可能である。フレーム長の符号化については７．１節で述べられる。復号器は、フレーム長に関する帯域外情報を用いずにフレームを復号することができる。従って、ＪＴ－Ｇ７１１．０符号化フレームは、それが示すサンプルの数の観点から“自己記述的”である。

6.4　ビットレート

ＪＴ－Ｇ７１１．０コーデックは、可変ビットレートコーデックである。生成されるＪＴ－Ｇ７１１．０ビット列のサイズは入力信号の特性に依る。符号化フレームの最小サイズは１オクテット（８ビット）である。また、符号化フレームの最大サイズは入力フレームデータのサイズに１オクテット加算した値であり、入力フレームがどの符号化ツールを用いても圧縮不可能である場合に起こり得る。符号化フレームのサイズはオクテット単位で表現される。

6.5　アルゴリズム遅延

ＪＴ－Ｇ７１１．０符号化器のアルゴリズム遅延は、8000Ｈｚサンプリングにおけるフレーム長 40、80、160、240、320 サンプルに対応して、それぞれ 5、10、20、30、40ms である。このフレームバッファリング以外の追加のアルゴリズム遅延はない。

6.6　計算量、所要記憶容量

ＪＴ－Ｇ７１１．０符号化器／復号器のＩＴＵ－Ｔソフトウェアツールライブラリ ＳＴＬ２００５ｖ２．２（ＩＴＵ－Ｔ勧告Ｇ．１９１）の基本演算子に基づいた最大計算量は 1.667WMOPSである。μ則、及び A 則を通じて最大の計算量を Table 6-1／JT-G711.0 に詳述する。ＪＴ－Ｇ７１１．０符号化器／復号器の所要記憶容量を Table 6-2／JT-G711.0 にオクテット単位で示す。なお、RAM の値は、配列及び単一変数を含むほぼ全ての変数に基づくものとする。また、‘ｋ’は 1000 単位とする。

Table 6-1／JT-G711.0 – Worst computational complexity of JT-G711.0 coder (WMOPS)

(ITU-T G.711.0)

Encoder	Decoder
1.0785	0.5887

Table 6-2／JT-G711.0 – Memory requirements of JT-G711.0 coder

(ITU-T G.711.0)

	Encoder	Decoder
Static RAM (k octets)	0.01	0
Scratch RAM (k octets)	3.59	1.37
Data ROM (k octets)	5.48	
Program ROM (number of basic ops)	3554	

6.7　コーデックの記述

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算術演算で記述されている。第９章で示されるＡＮＳＩ Ｃコードは、本標準の主要な部分を構成するものであり、このビットイグザクトな固定小数点での記述を反映している。コーデックの算術的な記述は、他の方法でも実装し得るが、本標準に準拠しないコーデックを実装することになってしまう可能性がある。したがって、不一致が生じた場合には、算術的な記述よりも、第９章のＡＮＳＩ Ｃコードによるアルゴリズム記述の方が優先される。ＡＮＳＩ Ｃコードと共に用いられるテスト信号のセットは、ＩＴＵ－ＴのＷｅｂサイトから入手可能である。

6.8　写像関数

本節では、ＪＴ－Ｇ７１１．０符号化器及び復号器の異なる符号化ツールで必要とされる様々なフォーマットに、Ｇ．７１１サンプルを写像する関数について記述する。

6.8.1　A則／μ則から線形PCMへの変換

6.8.1.1　A則から線形PCMへの変換

変換関数 $f_{A\to PCM}$ は、下記に従って、A則サンプル $x_A(n)$を、線形ＰＣＭサンプル $x_{PCM}(n)$に変換する。

$$x_{PCM}(n) = f_{A\to PCM}\left(x_A(n)\right) \tag{6-1}$$

$$
\begin{aligned}
&s_A = x_A(n) \otimes 0x80 \\
&y_A = (x_A(n) \oplus 0x55) \otimes 0x7F \\
&e_A = \left\lfloor \frac{y_A}{2^4} \right\rfloor \\
&m_A = y_A(n) \otimes 0x0F \\
&if\ e_A > 0 \\
&\qquad x_{PCM}(n) = \begin{cases} \left\{2^{e_A - 1} \cdot \left(2^4 m_A + 8 + 256\right)\right\} / 8 & if\ s_A = 0x80 \\ -\left\{2^{e_A - 1} \cdot \left(2^4 m_A + 8 + 256\right)\right\} / 8 & if\ s_A = 0 \end{cases} \\
&else \\
&\qquad x_{PCM}(n) = \begin{cases} \left\{2^4 m_A + 8\right\} / 8 & if\ s_A = 0x80 \\ -\left\{2^4 m_A + 8\right\} / 8 & if\ s_A = 0 \end{cases}
\end{aligned}
\tag{6-2}
$$

ここで、s_A、 e_A、および m_A は、それぞれ $x_A(n)$の極性、指数部、仮数部である。

6.8.1.2 μ則から線形PCMへの変換

変換関数 $f_{\mu \to PCM}$ は、下記に従って、μ則サンプル $x_\mu(n)$を、線形ＰＣＭサンプル $x_{PCM}(n)$に変換する。

$$
x_{PCM}(n) = f_{\mu \to PCM}\left(x_\mu(n)\right)
\tag{6-3}
$$

$$
\begin{aligned}
&s_\mu = x_\mu(n) \otimes 0x80 \\
&y_\mu = (x_\mu(n) \oplus 0x7F) \otimes 0x7F \\
&e_\mu = \left\lfloor \frac{y_\mu}{2^4} \right\rfloor \\
&m_\mu = y_\mu(n) \otimes 0x0F \\
&x_{PCM}(n) = \begin{cases} \left\{2^e \cdot \left(2^3 m_\mu + 128 + 4\right) - 132\right\} / 4 & if\ s_\mu = 0x80 \\ -\left\{2^e \cdot \left(2^3 m_\mu + 128 + 4\right) - 132\right\} / 4 & if\ s_\mu = 0 \end{cases}
\end{aligned}
\tag{6-4}
$$

ここで、s_μ、e_μ、および m_μは、それぞれ $x_\mu(n)$の極性、指数部、仮数部である。

6.8.2 線形PCMからA則／μ則への変換

6.8.2.1 線形PCMからA則への変換

変換関数 $f_{PCM \to A}$ は、下記に従って、線形ＰＣＭサンプル $x_{PCM}(n)$を、Ａ則サンプル $x_A(n)$に変換する。

$$
x_A(n) = f_{PCM \to A}\left(x_{PCM}(n)\right)
\tag{6-5}
$$

$$
\begin{aligned}
&m_A = \begin{cases} min\{x_{PCM}(n), 4095\} & if\ x_{PCM}(n) \geq 0 \\ -max\{x_{PCM}(n), -4096\} - 1 & if\ x_{PCM}(n) < 0 \end{cases} \\
&e_A = \left\lceil log_2\left(\frac{m_A}{2^6} + 1\right) \right\rceil \\
&y_A = \left(2^4 e_A + \left\lfloor \frac{m_A}{2^{e_A + 1}} \right\rfloor\right) \oplus 0x55 \\
&x_A(n) = \begin{cases} y_A \oplus 0x80 & if\ x_{PCM}(n) \geq 0 \\ y_A & if\ x_{PCM}(n) < 0 \end{cases}
\end{aligned}
\tag{6-6}
$$

6.8.2.2 線形PCMからμ則への変換

変換関数 $f_{PCM \to \mu}$ は、下記に従って、線形ＰＣＭサンプル $x_{PCM}(n)$を、μ則サンプル $x_\mu(n)$に変換する。

$$
x_\mu(n) = f_{PCM \to \mu}\left(x_{PCM}(n)\right)
\tag{6-7}
$$

$$
\begin{aligned}
m_\mu &= \begin{cases} min\{x_{PCM}(n)+33,8191\} & if\ x_{PCM}(n)\geq 0 \\ -max\{x_{PCM}(n)-33,-8192\}-1 & if\ x_{PCM}(n)<0 \end{cases} \\
e_\mu &= \left\lceil log_2\left(\frac{m_\mu}{2^6}+1\right)\right\rceil \\
y_\mu &= \left(2^4(e_\mu-1)+\left\lfloor\frac{m_\mu}{2^{e_A+1}}\right\rfloor\right)\oplus 0x7F \\
x_\mu(n) &= \begin{cases} y_\mu \oplus 0x80 & if\ x_{PCM}(n)\geq 0 \\ y_\mu & if\ x_{PCM}(n)<0 \end{cases}
\end{aligned}
\qquad (6\text{-}8)
$$

6.8.3 A則／μ則から*int8*への変換

８ビット整数フレームデータは、極性付き８ビット整数値（*int8*）に変換されたＧ．７１１のA則 $x_A(n)$、またμ則 $x_\mu(n)$記号から生成され、Ｇ．７１１符号値の正の最大値が 127 に写像され、負の最大値が-128 に写像される。

6.8.3.1 A則から*int8*への変換

変換関数 $f_{A\to int8}$ は、下記に従って、A則サンプル $x_A(n)$を、*int8* サンプル $x_{int8}(n)$に変換する。

$$x_{int8}(n) = f_{A\to int8}(x_A(n)) \qquad (6\text{-}9)$$

$$
\begin{aligned}
s_A &= x_A(n)\otimes 0x80 \\
y_A &= (x_A(n)\oplus 0x55)\otimes 0x7F \\
x_{int8}(n) &= \begin{cases} y_A & if\ s_A = 0x80 \\ -y_A-1 & if\ s_A = 0 \end{cases}
\end{aligned}
\qquad (6\text{-}10)
$$

6.8.3.2 μ則から*int8*への変換

変換関数 $f_{\mu\to int8}$ は、下記に従って、μ則サンプル $x_\mu(n)$を、*int8* サンプル $x_{int8}(n)$に変換する。

$$x_{int8}(n) = f_{\mu\to int8}(x_\mu(n)) \qquad (6\text{-}11)$$

$$
\begin{aligned}
s_\mu &= x_\mu(n)\otimes 0x80 \\
y_\mu &= (x_\mu(n)\oplus 0x7F)\otimes 0x7F \\
x_{int8}(n) &= \begin{cases} y_\mu & if\ s_\mu = 0x80 \\ -y_\mu-1 & if\ s_\mu = 0 \end{cases}
\end{aligned}
\qquad (6\text{-}12)
$$

6.8.4 *int8*からA則／μ則への変換

6.8.4.1 *int8*からA則への変換

変換関数 $f_{int8\to A}$ は、下記に従って、*int8* サンプル $x_{int8}(n)$を、A則サンプル $x_A(n)$に変換する。

$$x_A(n) = f_{int8\to A}(x_{int8}(n)) \qquad (6\text{-}13)$$

$$x_A(n) = f_{A\to int8}(x_{int8}(n)+128)+128 \qquad (6\text{-}14)$$

6.8.4.2 *int8*からμ則への変換

変換関数 $f_{A\to int8}$ は、下記に従って、*int8* サンプル $x_{int8}(n)$を、μ則サンプル $x_\mu(n)$に変換する。

$$x_\mu(n) = f_{int8\to\mu}(x_{int8}(n)) \qquad (6\text{-}15)$$

$$x_\mu(n) = f_{\mu\to int8}(x_{int8}(n)+128)+128 \qquad (6\text{-}16)$$

6.9　可変長符号化法

本節では、ＪＴ－Ｇ７１１．０符号化器、及び復号器において用いられる可変長符号化法について記述する。

6.9.1　ユナリ（１進）符号化

本標準では、整数値 v ($v \geq 0$)のユナリ（１進）符号は、バイナリ（２進）表現において v ビットのゼロと、それに後続する数値 1 として定義される。このユナリ（１進）符号により、整数値 v の符号長は v+1 となる。

ユナリ（１進）符号化ビット列を復号するために、数値１が後続する連続するゼロが探索され、連続するゼロの数 v が復号値となる。

*"Unary(v)"*は、整数値 v ($v \geq 0$)のユナリ（１進）符号化ビット列を示すものとする。

6.9.2　ライス符号化

整数値 v ($v \geq 0$)のライス符号（ゴロムーライス符号）は、ライスパラメータを S ($S \geq 0$)とした場合、以下のように定義される。

(1)　v の最下位 S ビットを切り離し、j に設定する。j は S ビットで表現される剰余値である。
　　これらのビットはライス符号の最下位ビットとなる。

(2)　商である $k = \left\lfloor \frac{v}{2^S} \right\rfloor$ を算出し、商 k を k+1 ビットでユナリ（１進）符号化する（６．９．１節参照)。
　　これらのビットはライス符号の最上位ビットとなる。

ライス符号化ビット列は、ライスパラメータが S ($S \geq 0$)である場合、以下のようにして復号される。

(1)　1 が後続する、連続する 0 の数を探索することにより k の値を得る。

(2)　後続する S ビットを読み込み、これを j に設定する。

(3)　$v = k \times 2^S + j$ を算出する。

*"Rice(S,v)"*は、ライスパラメータが S で与えられた際の、整数値 v ($v \geq 0$)のライス符号化ビット列を示すものとする。関数の第１引数はライスパラメータであり、第２引数は符号化すべき値である。

7　符号化器の機能記述

7.1　フレーム長に対する接頭符号

６．３節で述べたように、ＪＴ－Ｇ７１１．０コーデックは、フレームあたり 40、80、160、240、320 サンプルのフレーム長をサポートしている。Table 7-1／JT-G711.0 に、サポートされる各フレーム長に対応する接頭符号を示す。この識別子はビット列の最初、つまり符号化ビット列の最初の 8 ビットに置かれる。さらに、符号 0x00 は特例としてフレーム長が 0 であることを示すものとする。ＪＴ－Ｇ７１１．０符号化フレームの最初の符号が 0x00 であった場合は、復号処理が行われないことを意味する。従って、符号 0x00 は、１つ以上のＪＴ－Ｇ７１１．０符号化フレームの間、あるいは後に、符号化システムにより要求されるパディングオクテットとして使用される（ワード単位の伝送システムに、圧縮されたペイロードを適用する場合に、ワード境界を揃えるためにパディングオクテットを挿入することができる)。

Table 7-1／JT-G711.0 – List of supported frame lengths and corresponding prefix codes

(ITU-T G.711.0)

Frame length (*N* samples)	**Prefix code (in binary representation)**
0	0000 0000
40	01-- ----
80	10-- ----
160	11-- ----
240	0010 ----
320	0011 ----

7.2 各符号化ツールに関連する接頭符号

Table 7-2／JT-G711.0 に、符号化ツールのリストとそれらに対応する接頭符号を示す。この接頭符号は、符号化ビット列の最初のオクテットとして（幾つかの符号化ツールに対しては２番目のオクテットまで)、フレーム長の接頭符号の直後に置かれる。個別の符号化ツールによって生成される符号化ビット列は、接頭符号の直後に置かれる。

Table 7-2／JT-G711.0 – List of encoding tools and corresponding tool prefix codes

(ITU-T G.711.0)

Tool type		**Prefix code for ITU-T G.711.0 encoded frame (initial bits, in binary representation)**		**Note**
		***N*=40, 80, 160**	***N*=240, 320**	
Uncompressed coding		--00 0000	---- 0000	See clause 7.4
Constant coding tools	Constant plus zero coding	--00 0001	---- 0001	See clause 7.5
	Constant minus zero coding	--00 0010	---- 0010	
	Constant non-zero coding	--00 0011	---- 0011	
Plus Minus (PM) zero only	Binary coding	--00 0100	---- 0100	See clause 7.7
	PM zero Rice coding (Minus ≤ Plus)	--01 0ss	---- 100s s	Following two bits after the prefix code ss != 00 See clause 7.6
	PM zero Rice coding (Minus > Plus)	--01 1ss	---- 101s s	
Plus Minus zero only except one sample	Pulse mode coding (Minus < Plus)	--01 000	---- 1000 0	See clause 7.8
	Pulse mode coding (Minus > Plus)	--01 100	---- 1010 0	
Value-location coding		--00 0110	---- 0101	See clause 7.9
Mapped domain LP coding	LTP: enabled	1100 1	---- 011	Only for *N*≥160 See clause 7.10
	LTP: disabled	--1	---- 11	See clause 7.10
Fractional bit coding		0000 0010 to 0001 1111		See clause 7.11
Min-Max level coding		0100 0101	N/A	Only for *N*=40 See clause 7.12
Direct LP coding		0100 1	N/A	Only for *N*=40 See clause 7.13
NOTE – The first bits "--" (for *N*=40, 80, 160) or "----" (for *N*=240, 320) are the prefix codes representing the frame length, given in Table 4／JT-G711.0.				

7.3　符号化ツール選択

Table 7-3／JT-G711.0 に、各フレーム長に対して使用可能な符号化ツールを示す。“x”で示された項目は指定フレーム長に対して符号化ツールが使用可能であることを意味する。“N/A”で示された項目は指定フレーム長に対してその符号化ツールが使用不可であることを意味する。

Table 7-3／JT-G711.0 – Possible tools for each frame length

(ITU-T G.711.0)

Tool type	Frame length (*N* samples)				
	40	80	160	240	320
Uncompressed coding	x	x	x	x	x
Constant coding tools	x	x	x	x	x
Plus Minus (PM) zero only	x	x	x	x	x
Plus Minus zero only except one sample	x	x	x	x	x
Value-location coding	N/A	x	x	x	x
Mapped domain LP coding	x	x	x	x	x
Fractional bit coding	x	x	x	x	x
Min-Max level coding	x	N/A	N/A	N/A	N/A
Direct LP coding	x	N/A	N/A	N/A	N/A

Figure 7-1／JT-G711.0 に、符号化ツールの選択方法を示す。

以下は、Figure 7-1／JT-G711.0 において用いられる用語の定義である。

Perfome Y encoding は、符号化フレームサイズを決定するために、符号化器が実際に符号化ツール Y の全符号化処理を完了することを示す。

Calculate L(Y) は、符号化フレームサイズを決定するために、符号化器が実際に符号化ツール Y の全符号化処理を完了することなく、推定を行うことを示す。

Encode using Y は、符号化器が実際に符号化ツール Y を実行し、もしこれまでに符号化ツール Y による符号化処理を実行していない場合にはフレームの符号化を行い、既に符号化ツール Y を用いた符号化を実行済みである場合には単に符号化ビット列を使用する、ということを示す。

L(Best) は、符号化ビット列のサイズが既に計算されている符号化ツールに対して、フレームの符号化ビット列の最小サイズを示す。

Best:Y は、フレームの符号化ビット列のサイズが最小である符号化ツール Y が最良符号化ツールに決定されることを示す。

フレーム長がゼロである場合には、接頭符号 0x00 が生成され、符号化処理はスキップされる。他のフレーム長の場合、符号化器はまず入力フレームの全サンプルが単一値であるかどうかを判定する。上記に該当する場合には、固定値符号化ツールの内の１つが適用される（７．５節参照）。全ての値が正零 0^{+}である場合、固定正零値符号化ツール（*Constant Plus Zero coding tool*）が用いられる。また、全ての値が負零 0^{-}である場合、固定負零値符号化ツール（*Constant Minus Zero coding tool*）が用いられる。上記に該当しない場合、つまり他の固定値の場合、固定非零値符号化ツール（*Constant Non-zero coding tool*）が用いられる。

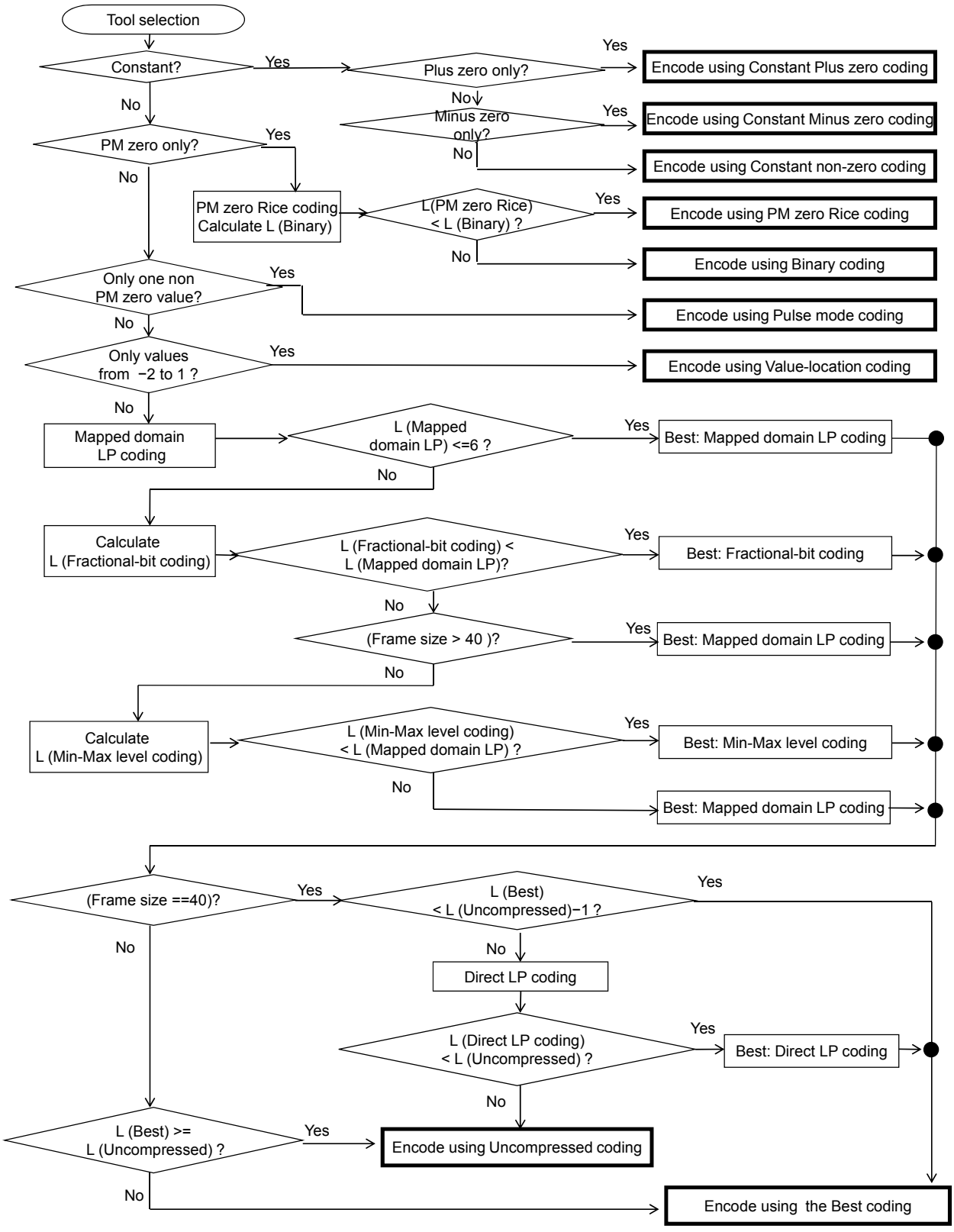

Figure 7-1 / JT-G711.0 – Flow chart for encoding tool selection

(ITU-T G.711.0)

入力フレームが正零または負零のみから構成されている場合、正負零値ライス符号化ツール（*Plus-Minus (PM) zero Rice coding tool*）（７．６節参照）、またはバイナリ（２進）符号化ツール（*Binary coding tool*）（７．７節参照）が適用される（いずれか圧縮率が高い方によって）。バイナリ（２進）符号化ツールは常に $(N/8+1)$ オクテットを生成するため、２つの符号化ツールのどちらを用いるべきかの決定に際しては、正負零値ライス符号化ツールのみが適用される。

入力フレームが１つのサンプルを除いて正零 0^+ と負零 0^- から構成されている場合、パルスモード符号化ツール（*Pulse mode coding tool*）（７．８節参照）が用いられる。

上記に該当しない場合、値位置符号化ツール（*Value-Location coding tool*）（７．９節参照）が判定される。これに該当しない場合、写像領域線形予測符号化ツール（*Mapped domain LP coding tool*）（７．１０節参照）が実行される。写像領域線形予測符号化ツールによる符号化データサイズが６オクテット以上である場合、さらに、端数ビット符号化ツール（*Fractional-bit coding tool*）（もし適用可能であれば）（７．１１節参照）、及び最小フレーム長（40 サンプル）に対して最小最大レベル符号化ツール（*Min-Max level coding tool*）（７．１２節参照）という２つの符号化ツールが試される。端数ビット符号化ツールの圧縮サイズは、圧縮アルゴリズムを実際に実行せずとも、フレーム長とフレーム内の値とから算出することができる。同様に、最小最大レベル符号化ツールについても、フレーム内の値、及び他の符号化ツールの符号化結果に応じて試される。40 サンプルのフレーム長に対して、上記のアルゴリズムにより選択された符号化ツールの符号化データサイズが入力Ｇ．７１１フレームのデータサイズよりも小さくない場合、直接線形予測符号化ツール（*Direct LP coding tool*）（７．１３節参照）が、それまでに選択された符号化ツールに代わって実行される。

全フレーム長に対しての最終ステップとして、符号化データサイズがフレーム長よりも大きくなることがわかった場合、非圧縮符号化ツール（*Uncompressed coding tool*）（７．４節参照）が選択される。

7.4　非圧縮符号化ツール（Uncompressed coding tool）

全ての符号化ツールが圧縮に失敗した場合、符号化器は非圧縮符号化ツール用の接頭符号を生成し（例えば、バイナリ（２進）表現では、40、80、160 サンプル長のフレームに対しては、“--00 0000”であり、240、320 サンプル長のフレームに対しては“---- 0000”である）、その接頭符号の後に単にオリジナルのＧ．７１１ビット列を再生成する。この場合、符号化データサイズは、入力データサイズに１オクテット加えたものとなる。Table 7-4／JT-G711.0 に、非圧縮符号化ツールのビットパッキングフォーマットを示す。

Table 7-4／JT-G711.0 – Bit packing format of the uncompressed coding tool

(ITU-T G.711.0)

Frame length (*N*)	**40, 80, 160**	**240, 320**
Frame length prefix	2 bits	4 bits
Tool prefix	6 bits	4 bits
G.711 symbols	$N \times 8$ bits	

NOTE – The frame length prefix and the tool prefix are given in Tables 7-1／JT-G711.0 and 7-2／JT-G711.0, respectively.

7.5　固定値符号化ツール（Constant value coding tool）

入力フレーム内の全てのサンプルの値が同じである場合、固定値符号化ツールの内の１つが適用される。Table 7-2／JT-G711.0 に示すように、A則に対する 0xd5、μ 則に対する 0xff のような固定正零値が、“--00 0001”、“---- 0001”のバイナリ（２進）表現によって示される。A則に対する 0x55、μ 則に対する 0x7f のような固定

負零値は、“--00 0001”、“---- 0001”というオクテットによって示される。これらの場合には、この後にはオクテットは続かない。

上記以外の固定値は、“--00 0011”、または“---- 0011”という接頭符号により示され、接頭符号のオクテットの後に 8 ビット表現（１オクテット）のＧ．７１１Ａ則、μ 則記号の実際の固定値が続く。Table 7-5／JT-G711.0 に固定値符号化ツールのビットパッキングフォーマットを示す。

Table 7-5／JT-G711.0 – Bit packing format of the constant value coding tools

(ITU-T G.711.0)

	Constant Plus zero coding and Constant Minus zero coding		**Constant non-zero coding**	
Frame length (*N*)	**40, 80, 160**	**240, 320**	**40, 80, 160**	**240, 320**
Frame length prefix	2 bits	4 bits	2 bits	4 bits
Tool prefix	6 bits	4 bits	6 bits	4 bits
Constant G.711 symbol value	–		8 bits	

7.6　正負零値ライス符号化ツール（Plus-Minus zero Rice coding tool）

入力フレーム内の全てのサンプルの値が正零 0^+か負零 0^-である場合、符号化器は正負零値ライス符号化ツールの判定を行う。

まず始めに、この符号化ツールは、フレーム中に存在する正零 0^+、及び負零 0^-の数をカウントし、どちらの頻度が高いかを判定する。正零値または負零値のうち、より多く出現する方の値を 0_m と呼ぶ。正零 0^+と負零 0^-の頻度が同じ場合、0_m は正零 0^+に設定される。符号化ツールの接頭符号は、$0_m = 0^+$である場合は“--01 0”、あるいは“---- 100”が用いられ、$0_m = 0^-$である場合は“--01 1”、あるいは“---- 101”が用いられる（Table 5 参照）。

次に、符号化ツールは、以下のステップに従って、入力サンプルを、0_l が後続する、0_m の連続数の値 $v_i \ (0 \le i \le N_v - 1)$に変換する。

(1)　$i = 0$ に初期化する。

(2)　第 i 番目の連続する 0_m を取得するために、0_lが後続する、連続する 0_m の数 v_i をカウントする。
連続する 0_m がフレームの最終サンプルで終了する場合は、仮想的な 0_l が最終サンプル直後に存在するものとして連続する 0_m の数をカウントする。

(3)　i を 1 増やし、入力フレームの全てのサンプルが処理されるまでステップ 2)を繰り返す。

(4)　$N_v = i$ に設定する。

全ての取りうる S 値に対して圧縮後の符号長を推定し、v_i に対して符号長が最小となる最良ライスパラメータ値を選択する。

ライスパラメータ S（S は 0 より大きい）は、Table 7-6／JT-G711.0 に示すハフマン符号により符号化される。S の範囲は入力フレーム長に依存する。

最後に、符号化ツールは、６．９．２節で述べたようにして、最良ライスパラメータ値 S を用いて、ライス符号化によって数値列 v_i を符号化する。

Table 7-6／JT-G711.0 – Huffman codes of the Rice parameters for PM zero Rice coding

(ITU-T G.711.0)

S	Huffman encodings of *S* for each frame length			Note
	N=40	*N*=80	*N*=160, 240, 320	
1	01	01	01	Note that the value of S=0 is not allowed for the PM zero coding tool and code 00 is reserved for the prefix of pulse mode coding.
2	10	10	10	
3	110	1100	1100	
4	1110	1101	1101	
5	1111	1110	11100	
6	–	1111	11101	
7	–	–	11110	
8	–	–	111110	
9	–	–	111111	

Table 7-7／JT-G711.0 に、正負零値ライス符号化ツールのビットパッキングフォーマットを示す。

Table 7-7／JT-G711.0 – Bit packing format of the Plus-Minus zero Rice coding tool

(ITU-T G.711.0)

Frame length (*N*)	40, 80	160	240, 320
Frame length prefix	2 bits		4 bits
Tool prefix	3 bits		3 bits
Huffman code for the Rice parameter *S*	2 to 4 bits	2 to 6 bits	
Run length codes	Variable (Rice codes for runs of more zeros)		
Octet boundary alignment	0 to 7 bits of '0's		

7.7 バイナリ（２進）（binary）符号化ツール（Binary coding tool）

入力フレーム内の全てのサンプルの値が、正零 0^+か負零 0^-のどちらかである場合、かつ、正負零値ライス符号化ツールが符号化データサイズを減少させない場合、バイナリ（２進）符号化ツールが適用される。このツールは、バイナリ（２進）表現の“--00 0100”、または“---- 0100”という 1 オクテットの接頭符号を生成し、それに 1 サンプルあたり 1 ビットに変換された入力サンプル値が後続する。生成された符号において、0 は正零 0^+を意味し、また 1 は負零 0^-を意味する。Table 7-8／JT-G711.0 に、バイナリ（２進）符号化ツールのビットパッキングフォーマットを示す。

Table 7-8／JT-G711.0 – Bit packing format of the binary coding tool

(ITU-T G.711.0)

Frame length (*N*)	40, 80, 160	240, 320
Frame length prefix	2 bits	4 bits
Tool prefix	6 bits	4 bits
1 bit per sample	*N* bits	

7.8　パルスモード符号化ツール（Pulse mode coding tool）

入力フレーム内の全てのサンプルの値が、1 サンプルを除いて、正零 0^+か負零 0^-のどちらかである場合、符号化器は、入力に対してパルスモード符号化ツールを適用する。パルスモード符号化ツールに対する接頭符号はフレーム長識別子の接頭符号の後に生成される。

まず始めに、この符号化ツールは、フレーム中に存在する正零 0^+、及び負零 0^-の数をカウントし、どちらの頻度が高いかを判定し、また正零 0^+でもなく負零 0^-でもない値を持つサンプルの位置を探索する。正零 0^+でもなく、負零 0^-でもないサンプルをパルスと呼ぶ。また、正零 0^+、及び負零 0^-のサンプルを非パルスと呼ぶ。正零値または負零値のうち、より多く出現する方の値を 0_m と呼び、もう一方の値を 0_l と呼ぶ。符号化ツールの接頭符号は、$0_m = 0^+$である場合は“--01 000”、あるいは“---- 1000 0”が用いられ、$0_m = 0^-$である場合は“--01 100”、あるいは“---- 1010 0”が用いられる。

パルスの位置と値を記憶した後、パルスの値はより多く出現する零値 0_m として考慮される。次に、符号化ツールは、７．６節で述べたようにして、入力サンプル値を、0_l が後続する、0_m の連続数の値 v_i $(0 \le i \le N_v - 1)$ に変換する。

ライスパラメータ S は、Table 7-9／JT-G711.0 に示すハフマン符号を用いて最初に符号化される。次に、Table 7-10／JT-G711.0 に示すようにして、パルス位置インデックスが入力フレーム長に応じて直接符号化される。符号化効率を改善するために、符号化ツールはパルス値と、0_m、あるいは 0_l の値との差分を比較し、差分がより小さい方が選択される。0_m あるいは 0_l との差分符号化を切り替えるために 1 ビットのフラグが用いられる。差分に１を加えた値が６．９．２節で述べたようにして、ライスパラメータ 0 を使って、ライス符号化にて符号化される。

最後に、符号化ツールは、パルスサンプルの符号化ビットの後に、６．９．２節で述べたライス符号化を使って連続する数値を符号化する。

Table 7-11／JT-G711.0 に、パルス符号化ツールのビットパッキングフォーマットを示す。

Table 7-9／JT-G711.0 – Huffman codes of the Rice parameters for the pulse mode coding

(ITU-T G.711.0)

S	Huffman encodings of *S* for each frame length		
	N=40	N=80	N=160, 240, 320
0	00	00	00
1	01	01	01
2	10	10	10
3	110	1100	1100
4	1110	1101	1101
5	1111	1110	11100
6	–	1111	11101
7	–	–	11110
8	–	–	111110
9	–	–	111111

Table 7-10／JT-G711.0 – Number of bits for pulse position index

(ITU-T G.711.0)

Frame length (*N* samples)	**Numbers of bits for pulse position index**
40	6 bits
80	7 bits
160	8 bits
240	8 bits
320	9 bits

Table 7-11／JT-G711.0 – Bit packing format of the pulse mode coding tool

(ITU-T G.711.0)

Frame length (*N*)	**40**	**80**	**160**	**240**	**320**
Frame length prefix	2 bits			4 bits	
Tool prefix	5 bits			5 bits	
Huffman code for the Rice parameter *S*	2 to 4 bits		2 to 6 bits		
Pulse position index	6 bits	7 bits	8 bits		9 bits
Flag for the differential base (0^+ or 0^-)	1 bit				
Rice codes for the differential pulse value	Variable (Rice codes for the pulse value differential from 0^+ or 0^-)				
Rice codes	Variable (Rice codes for runs of more zeros)				
Octet boundary alignment	0 to 7 bits of '0's				

7.9 値位置符号化ツール（Value-location coding tool）

7.9.1 値位置符号化器の概要

値位置符号化器は、フレーム内データの位置を符号化するために *int8* フレームデータ $x_{int8}(n)$に対して動作する。入力であるＧ．７１１のシンボル $I_A(n)$, n=0,...,N–1 もしくは $I_\mu(n)$, n=0,...,N–1 は、6.8.3 項にて説明されている A 則／μ 則から *int8* への変換を用いて、$x_{int8}(n)$に変換される。ここで、N はフレームあたりのサンプル数である。

Table 7-12／JT-G711.0 は、値位置符号化ツールのビットパッキングフォーマットを示す。

Table 7-12／JT-G.711.0 – Bit packing format of the value-location coding tool

(ITU-T G711.0)

Frame length (*N*)	**40, 80, 160**	**240, 320**	**Note**
Frame length prefix	2 bits	4 bits	
Tool prefix	6 bits	4 bits	
Data range case identifier	2 bits		See clause 7.9.2
Condition flag	1 bit		See clause 7.9.3
Value-location method identifier	3 bits		See clause 7.9.4
Binary or Rice codes	Variable		See clause 7.9.5
Octet boundary alignment	0 to 7 bits of '0's		

7.9.2 値位置符号化器の適用決定

値位置符号化器の適用決定は *int8* 領域にて実行され、フレーム長、最大及び最小サンプル値、最頻出サンプル値の情報に基づく。

データ幅 R は、最大値と最小値との差に基づき以下のように決定される。

$$R = X_{\max} - X_{\min} + 1 \tag{7-1}$$

ここで、$X_{\max} = \max\{x_{\text{int8}}(n) \mid n = 0,...,N-1\}$ および $X_{\min} = \min\{x_{\text{int8}}(n) \mid n = 0,...,N-1\}$ である。

$R \le 4$ のときは、最小値及び最大値、$x_{\text{int8}}(n)$フレームのフレーム長は、Table 7-13／JT-G711.0 に規定されるデータ幅のいずれかに適合するか検査される。

Table 7-13／JT-G711.0 – Data-range case identifier in value-location coding tool

(ITU-T G.711.0)

R	$X_{\min}$	$X_{\max}$	Frame length	Data-range case	Bitstream sequence
4	−2	1	240, 320	0	00
3	−1	1	All	1	01
3	−2	0	All	2	10
2	0	1	All	3	11

フレームデータ $x_{\text{int8}}(n)$, n=0,...,N–1 が Table 7-13／JT-G711.0 のデータ幅に適合する場合は、値位置符号化器は、参照値 v_0 をフレーム内で最も頻繁に発生するサンプル値として決定する。v_0 = 0 の場合は、値位置符号化が適用され、Table 7-13／JT-G711.0 に規定されるビット列に続く出力ビットストリームに、関連する接頭符号オクテットが書かれる。$v_0 \neq 0$ のような、どの条件にも適合しない場合は値位置符号化ツールは使われない。

7.9.3 値写像

符号化されるフレームの中に生じる特有の値の個数は R 個であり、参照値 $v_0 = 0$ もその個数の中に含まれる。フレーム中の残りの R–1 個の値は、v_k に写像される。ここで、k = 1,...,R–1 である。データ幅事例、N_L と標記される v_0 より大きい値の発生数、N_S と標記される v_0 より小さい値の発生数により写像が決定される。写像は、Table 7-14／JT-G711.0 に示される。

Table 7-14／JT-G711.0 – Mapping values to v_k and condition flag

(ITU-T G.711.0)

Condition	Data range case	v_0	v_1	v_2	v_3	Bitstream sequence
$N_L \ge N_S$	0	0	1	−1	−2	1
	1	0	1	−1	–	
	3	0	1	–	–	
$N_L < N_S$	0	0	−1	1	−2	0

$N_L \ge N_S$ の状態か $N_L < N_S$ の状態かを表す 1 ビットは、Table 7-14／JT-G711.0 に規定される状態フラグとしてビットストリームに書かれる。

7.9.4　逐次的値位置計算

v_kが発生する位置は、k = 1,...,R–1 に対して順に符号化される。一旦 v_kの位置が符号化されると、それらは、それに続く v_i ($i > k$)の位置符号化の際には考慮される必要はない。

まず $s_1(n) = x_{int8}(n)$、n = 0,...,N–1 のように定義される。そして、フレームデータ $x_{int8}(n)$から v_1,...,v_{k-1} の値が取り除かれた後に、k =2,...,R–1 に対して、$s_k(j)$をベクトルとして表現する。$x_{int8}(n)$における v_kの発生数は N_kと表記され、$s_k(j)$の要素数は D_kと表記される。ここで、ベクトル $s_k(j)$の要素数は、k の増加と共に減少する。

$$\begin{aligned} D_1 &= N \\ D_k &= N - \sum_{i=1}^{k-1} N_i \qquad k = 2,...,R-1 \end{aligned} \tag{7-2}$$

k = 2,...,R–1 に対する $s_k(j)$の要素は、ベクトル $s_{k-1}(j)$内の全ての v_{k-1} の発生を除くことにより、以下のように計算される。

$$\begin{gathered} \text{set } i = 0 \\ \text{for each } j = 0,...,D_{k-1}-1 \\ \text{if } s_{k-1}(j) \neq v_{k-1} \\ \text{then } s_k(i) = s_{k-1}(j) \text{ and } i = i + 1 \end{gathered} \tag{7-3}$$

なお、v_0 の発生位置は符号化される必要はない。なぜならば、他の全ての v_k に対する位置が一意に定まるからである。

7.9.5　符号化方法

$s_k(j)$シーケンス内の v_k の位置は、$l_k(m)$、m = 0,...,N_k – 1 のように表記される。位置 $l_k(m)$は、各 k に対して、４つの中から最も圧縮が得られる方法を選択することにより符号化される。

４つの方法は以下の通りである。

- バイナリ（２進）符号化（7.9.5.1 節参照）
- ライス符号化（7.9.5.2 節参照）
- 値の発生がないことの明確な符号化（符号化するべき位置がない）
- 明示的な位置符号化（7.9.5.3 節参照）

位置 $l_k(m)$の符号化は、位置の順番に従って以下のように実施される。

$$l_k(m_1) < l_k(m_2) \text{ whenever } m_1 < m_2 \tag{7-4}$$

各符号化方法に対する値位置符号化識別子のビットシーケンスを Table 7-15／JT-G711.0 に示す。

Table 7-15／JT-G711.0 – Bit sequence of value-location encoding identifier

(ITU-T G.711.0)

Encoding	Bitstream sequence
Binary encoding	000
Run length Rice encoding (S = 1)	001
Run length Rice encoding (S = 2)	010
Run length Rice encoding (S = 3)	011

Encoding	Bitstream sequence
Run length Rice encoding (S = 4)	100
Value not occurring	101
Unused	110
Explicit location encoding	111

7.9.5.1　バイナリ（２進）符号化

v_kの位置をランレングス符号化するのに必要なビット数が D_k（$s_k(j)$の要素数）より大きい場合、バイナリ（２進）符号化が用いられる。符号化器は、バイナリ（２進）符号化を用いることを示すビットシーケンス"000"（Table 7-15／JT-G711.0 参照）をビットストリームに書く。これに続いて D_k ビットがビットシーケンスに書かれ、２進数１及び０は各々$s_k(j) = v_k$及び$s_k(j) \neq v_k$、（$j = 0,...,D_k - 1$）に対応する。ここで、$s_k(j) = v_k$に対する位置は$j = l_k(m)$、$m = 0,...,N_k - 1$ で与えられる。

7.9.5.2　ライス符号化

各 k に対して、$s_k(j)$シーケンス内の v_k の発生間隔のセグメント長を符号化するために必要となるビット数を決定するために、符号化器は 6.9.2 節に記述されるライス符号化ツールを用いる（これは、実際のところは v_k 値の全ての存在位置を規定する）。符号化器は、最適なライスパラメータ S を１から４の範囲で探索し、最適な圧縮を与えるものを選ぶ。

ライス符号化が最適な圧縮を与える場合、選択されたライスパラメータ S が３ビットの２進シーケンス（Table 7-15／JT-G711.0 参照）としてビットストリームに書かれ、符号化されたセグメント長がそれに続く。

7.9.5.3　明示的な位置符号化

バイナリ（２進）符号化が用いられず、かつ $1 \le N_k \le 4$ であり、かつ v_k の位置をランレングス符号化するのに必要なビット数が $N_k\lceil \log_2(D_k) \rceil + 2$ よりも大きい場合、明確な位置符号化が用いられる。符号化器は、$m = 0,...,N_k - 1$ に対して $s_k\left(l_k(m)\right) = v_k$ となるような N_k 個の位置 $l_k(m)$を決定する。位置符号化が用いられることを示すために、ビットシーケンス"111"が出力ビットストリームに書かれ（Table 7-15／JT-G711.0 参照）、２ビットの２進値 $N_k - 1$ がそれに続く。そして、N_k 個のインデックス $l_k(m)$が、$\lceil \log_2(D_k) \rceil$ ビットと共に出力ビットストリームに書かれる。

7.10　写像領域線形予測符号化ツール（Mapped domain LP coding tool）

7.10.1　写像領域線形予測符号化器の概要

各フレーム長に対する写像領域線形予測符号化で用いられる補助ツールを Table 7-16／JT-G711.0 に示す。"x"が印された部分は、補助ツールが該当のフレーム長で利用可能なことを示す。"N/A"が印された部分は、該当のフレーム長でそのツールが使われないことを示す。

Table 7-16／JT-G711.0 – Possible sub-tools for each frame length

(ITU-T G.711.0)

Sub-tool	Frame length (N samples)				
	40	80	160	240	320
Long term prediction (LTP)	N/A	N/A	x	x	x
PM zero mapping (only for μ-law input)	N/A	x	x	x	x
Bandwidth expansion	N/A	N/A	x	x	x
Rice coding of residual	x	N/A	N/A	N/A	N/A

Sub-tool	Frame length (*N* samples)				
	40	80	160	240	320
E-Huffman coding for residual	N/A	x	x	x	x

Figure 7-2／JT-G711.0 は、写像領域線形予測符号化器のブロック図を示す。

写像領域線形予測符号化ツールは、*N* 個のＧ．７１１のＡ則シンボル $I_A(n)$ 、$n = 0,...,N-1$ もしくはＧ．７１１の μ 則シンボル $I_\mu(n)$ 、$n = 0,...,N-1$ を受け取る。

最初に、6.8.1 節に記述されているＡ則／μ 則から均一ＰＣＭへの変換を用いて、これら *N* 個のＧ．７１１シンボルが均一（線形）ＰＣＭ領域の $x_{PCM}(n)$、$n = 0,...,N-1$ に変換される。

そして、線形予測分析を用いて、均一（線形）ＰＣＭ領域において $x_{PCM}(n)$に対する短期予測が実行される（7.10.3 節参照）。しかし、予測値は *int8* 領域の目標値から減ぜられるため、予測残差信号は、[–255, 255]の範囲内で得られる。線形予測符号化パラメータの表現は、偏自己相関係数が用いられる（7.10.3.6 節参照）。残差信号の振幅が計算され（7.10.5.5 節参照）、ライス符号化もしくはＥ－ハフマン符号化を用いて符号化される（7.10.5.5 節及び 7.10.5.6 節参照）。

80 よりも大きいフレームサイズに対しては、短期予測に加えて、線形ＰＣＭ領域において $x_{PCM}(n)$に対する長期予測も実行される（7.10.4 節参照）。短期予測は常に適用されるのに対して、長期相関の利用は、現フレームの長期相関が取り除かれるべきか否かを決定する長期相関分析に基づいてＯＮ／ＯＦＦが切替えられる。予測値は *int8* 領域の目標値から減ぜられるため、得られた長期予測残差信号もまた[–255, 255]の範囲に入る。

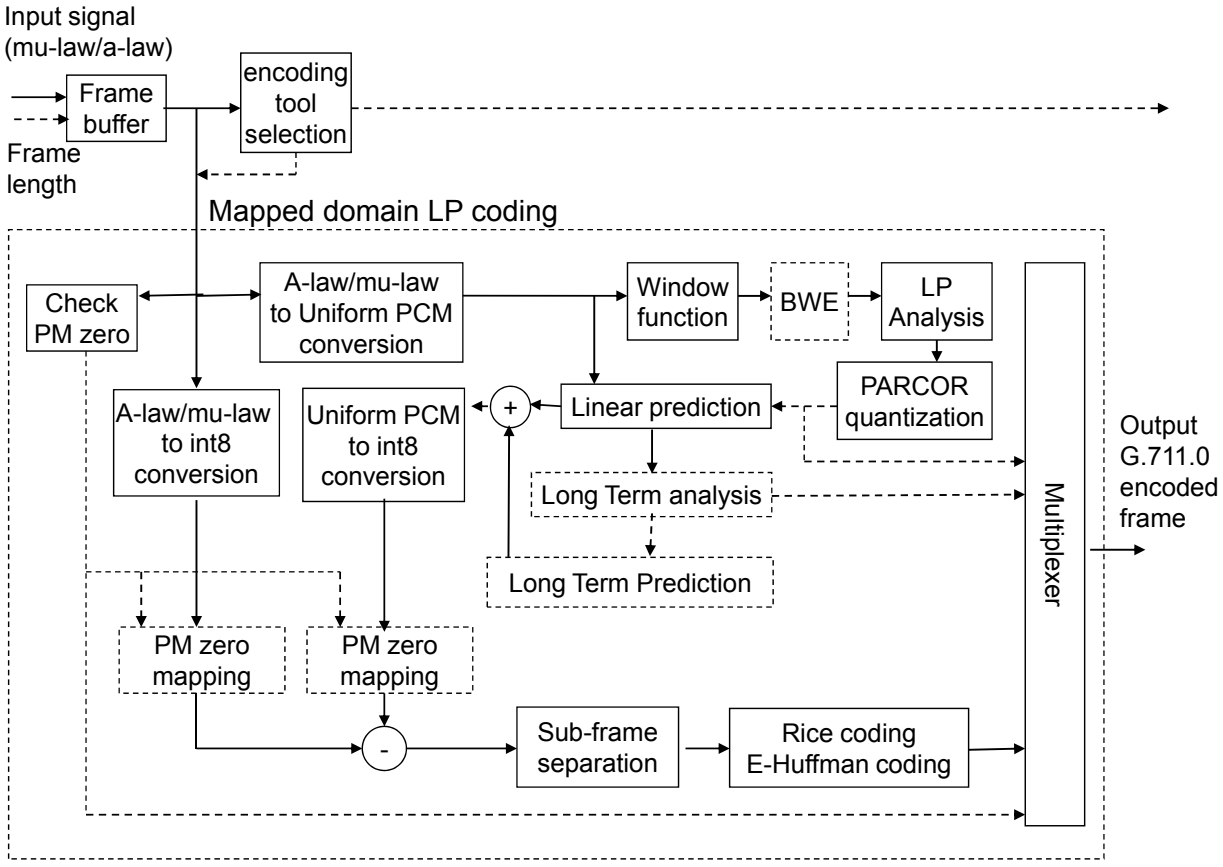

Figure 7-2 / JT-G711.0 – block diagram of the mapped domain LP encoder

(ITU-T G.711.0)

7.10.2　写像領域線形予測符号化ツールのビットパッキング

Table 7-17／JT-G711.0 に写像領域線形予測符号化ツールのビットパッキングフォーマットを示す。

Table 7-17／JT-G711.0 – Bit-packing format of mapped domain LP coding tool

(ITU-T G.711.0)

Frame length (*N*)		**40**	**80**	**160**	**240**	**320**	**Note**
Frame length prefix		2 bits			4 bits		
Tool prefix	LTP disabled	1 bit		1 bit	2 bits		
	LTP enabled	N/A		3 bits	3 bits		
PM zero mapping flag		N/A	0 to 3 bits				See clause 7.10.5.2
Index code for LPC order $\hat{P}$		2 bits					See clause 7.10.3.5
PARCOR		Variable					See clause 7.10.3.6
Separation parameters		Variable					See clause 7.10.5.3
Progressive LPC		Variable					See clauses 7.10.3.8 and 7.10.5.4
Entropy coding of residual		Variable					See clauses 7.10.5.5 and 7.10.5.6
Octet boundary alignment		0 to 7 bits of '0's					

7.10.3　線形予測

7.10.3.1　窓掛け

線形予測分析を実行するために、線形ＰＣＭ領域の入力信号 $x_{\mathrm{PCM}}(n)$は、まず以下のように窓関数 $W_{\mathrm{PCM},N}(n)$ が乗ぜられる。

$$\tilde{x}_{\mathrm{PCM}}(n) = W_{\mathrm{PCM},N}(n) \cdot x_{\mathrm{PCM}}(n)\,, \tag{7-5}$$

ここで、窓関数 $W_{\mathrm{PCM},N}(n)$は、入力フレーム長及び入力信号に依存する。
窓掛け後の信号 $\tilde{x}_{\mathrm{PCM}}(n)$ は、線形予測のための線形予測係数を計算するために用いられる。

7.10.3.1.1　40 サンプルフレームに対する窓関数

40 サンプルフレームに対して、フレーム内の最初のサンプルと最後のサンプルに依存した２つの窓関数セットが用いられる。最初のサンプル $|x_{\mathrm{PCM}}(0)|$ の絶対値が閾値 *win_thr*1 = 128 より小さい場合、窓の最初の４つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},40}(n) = 0.26 + 0.74 \cdot \cos(2 \cdot \pi \cdot (31 - 8 \cdot n)/127) \qquad n = 0,1,2,3 \tag{7-6}$$

そうでない場合は、窓の最初の４つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},40}(n) = 0.23 + 0.77 \cdot \cos(2 \cdot \pi \cdot (31 - 8 \cdot n)/127) \qquad n = 0,1,2,3 \tag{7-7}$$

窓の５番目から 36 番目までの係数値は１に設定される。

$$W_{\mathrm{PCM},40}(n) = 1 \qquad n = 4,...,35 \tag{7-8}$$

最後のサンプル$\left|x_{\mathrm{PCM}}(39)\right|$の絶対値が閾値 *win_thr*1 = 128 より小さい場合、窓の最後の４つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},40}(n) = 0.26 + 0.74 \cdot \cos(2 \cdot \pi \cdot (8 \cdot n - 281)/127) \qquad n = 36,37,38,39 \tag{7-9}$$

そうでない場合は、窓の最後の４つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},40}(n) = 0.23 + 0.77 \cdot \cos(2 \cdot \pi \cdot (8 \cdot n - 281)/127) \qquad n = 36,37,38,39 \tag{7-10}$$

Table 9-1／JT-G711.0 のテーブル（84）及び（85）は、$W_{\mathrm{PCM},40}(n)$の実際の値を与える。

7.10.3.1.2　80 サンプルフレームに対する窓関数

80 サンプルフレームに対して、フレーム内の最初のサンプルと最後のサンプルに依存して、２つの窓関数セットが用いられる。最初のサンプル$\left|x_{\mathrm{PCM}}(0)\right|$の絶対値が閾値 *win_thr*1 = 128 より小さい場合、窓の最初の８つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},80}(n) = 0.26 + 0.74 \cdot \cos(2 \cdot \pi \cdot (31 - 4 \cdot n)/127) \qquad n = 0,1,2,...,7 \tag{7-11}$$

そうでない場合は、窓の最初の８つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},80}(n) = 0.16 + 0.84 \cdot \cos(2 \cdot \pi \cdot (31 - 4 \cdot n)/127) \qquad n = 0,1,2,...,7 \tag{7-12}$$

窓の９番目から 72 番目までの係数値は１に設定される。

$$W_{\mathrm{PCM},80}(n) = 1 \qquad n = 8,...,71 \tag{7-13}$$

最後のサンプル$\left|x_{\mathrm{PCM}}(79)\right|$の絶対値が閾値 *win_thr*1 = 128 より小さい場合、窓の最後の８つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},80}(n) = 0.26 + 0.74 \cdot \cos(2 \cdot \pi \cdot (4 \cdot n - 285)/127) \qquad n = 72,73,74,...,79 \tag{7-14}$$

そうでない場合は、窓の最後の８つの係数値は以下のように設定される。

$$W_{\mathrm{PCM},80}(n) = 0.16 + 0.84 \cdot \cos(2 \cdot \pi \cdot (4 \cdot n - 285)/127) \qquad n = 72,73,74,...,79 \tag{7-15}$$

Table 9-1／JT-G711.0 のテーブル（86）及び（87）は、$W_{\mathrm{PCM},80}(n)$の実際の値を与える。

7.10.3.1.3　160 サンプルフレーム、240 サンプルフレーム及び 320 サンプルフレームに対する窓関数

160 サンプルフレーム、240 サンプルフレーム及び 320 サンプルフレームに対しては、窓は以下のように与えられる。

$$W_{\mathrm{PCM}}(n)=\begin{cases}0.5-0.5\cdot\cos(10\pi\dfrac{n}{N-1}) & 0<n<\left\lceil\dfrac{N}{10}+0.5\right\rceil\\ 1.0 & \left\lceil\dfrac{N}{10}+0.5\right\rceil<n<\left\lceil\dfrac{9N}{10}-0.5\right\rceil\\ 0.5-0.5\cdot\cos(10\pi\dfrac{N-n-1}{N-1}) & \left\lceil\dfrac{9N}{10}-0.5\right\rceil<n<N-1\end{cases} \tag{7-16}$$

Table 9-1／JT-G711.0 のテーブル（82）及び（83）は、$W_{\mathrm{PCM}}(n)$の実際の値を与える。

7.10.3.2 自己相関関数

窓掛け後の信号 $\tilde{x}_{\mathrm{PCM}}(n)$ は、自己相関係数 $c(i)$を計算するために用いられる。

$$c(i)=\sum_{n=i}^{N-1}\tilde{x}_{\mathrm{PCM}}(n)\cdot\tilde{x}_{\mathrm{PCM}}(n-i)\qquad i=0,\ldots,P_{\max} \tag{7-17}$$

ここで、$P_{\max}$ は各フレーム長に対して定義された最大予測次数である（Table 7-18／JT-G711.0 参照）。

Table 7-18／JT-G711.0 – Maximum prediction order

(ITU-T G.711.0)

Frame length (*N*)	$\boldsymbol{P_{\max}}$
40	4
80	8
160, 240	10
320	12

7.10.3.3 帯域幅拡張

線形予測フィルタのスペクトルピークを広げるため、160、240、320 サンプルフレームに対して帯域幅拡張が施される。以下のようにラグ窓 $w_{lag}(i)$を自己相関係数に乗じることにより、34Hz の帯域幅拡張が適用される。

$$w_{lag}(i)=\exp\left[-\frac{1}{2}\left(\frac{2\pi f_0\cdot i}{f_s}\right)^2\right]\qquad i=1,\ldots,P_{\max} \tag{7-18}$$

ここで、f_0 = 34Hz は拡張帯域幅、f_s = 8000Hz はサンプリング周波数、$P_{\max}$ は最大予測次数である。１番目の自己相関係数 $c(0)$に対しては、フレームごとに以下のように白色雑音補正係数が計算される。

$$w_{lag}(0)=\begin{cases}1.0018-2^{-14}(E_{norm}+E_{thr}) & \text{if } E_{norm}<E_{thr}\\ 1.0028-2^{-14}(E_{norm}+E_{thr}) & \text{otherwise}\end{cases} \tag{7-19}$$

ここで、$E_{norm}=30-\lfloor\log_2[c(0)]\rfloor$ と E_{thr}はフレーム長に依存する適応閾値であり、以下のように定義される。

$$E_{thr}=\begin{cases}13 & N=160\\ 12 & N=240,320\end{cases} \tag{7-20}$$

帯域幅拡張された自己相関係数は以下のように与えられる。

$$c'(i) = w_{lag}(i) \cdot c(i) \qquad i = 0,...,P_{\max} \tag{7-21}$$

40 及び 80 サンプルフレームに対しては、帯域幅拡張は適用されない。

$$c'(i) = c(i) \qquad i = 0,...,P_{\max} \tag{7-22}$$

Table 9-1／JT-G711.0 のテーブル（104）は、$w_{lag}(i)$の実際の値を与える。

7.10.3.4　線形予測分析

下記の方程式を解くことにより、変形された自己相関係数 $c'(i)$、$i = 0,...,P_{\max}$ を用いて線形予測フィルタ係数 $a_i^{[P]}$、$i = 1,...,P$ が得られる。ここで、P はそのフレームにおける予測次数である（$P \le P_{\max}$）。

$$\sum_{i=1}^{P} a_i^{[P]} c'(|i-j|) = -c'(j) \qquad j = 1,...,P \tag{7-23}$$

方程式（7-23）は、レビンソンダービンアルゴリズムを用いて解かれる。予測次数 P、（$P \le P_{\max}$）及び偏自己相関係数として知られる反射係数 k_i は、レビンソンダービンアルゴリズムの副産物として得られる。
このアルゴリズムは、以下のような再帰的ステップにより実行される。

1)　$P = P_{\max}$ 及び $a_0^{[0]}$ =1.0 に初期化し、繰り返し数を $i = 1$ に設定する。
2)　$k_{i-1} = -\dfrac{1}{E^{[i-1]}}\left(\sum_{j=0}^{i-1} a_j^{[i-1]} c'(i-j)\right)$ を計算する
3)　k_{i-1} を検査し、$i > 1$ かつ $\left|2^{15} k_{i-1}\right| \ge 32750$ である場合には、$k_{i-1} = 0$ とする。
4)　k_{i-1} を検査し、$i > 1$ かつ $k_{i-1} = 0$ である場合には、$P = i$ とする。
5)　$a_i^{[i]} = k_{i-1}$ とする。
6)　$j = 1,...,i-1$ に対して、$a_j^{[i]} = a_j^{[i-1]} + k_{i-1} \cdot a_{i-j}^{[i-1]}$ を計算する。
7)　$E^{[i]} = (1 - k_{i-1}^2) E^{[i-1]}$ を計算する。
8)　i が最大予測次数 $P_{\max}$ に達するまで、i を１増加させてステップ２に戻る。

最終的な解として、P と k_j（$j = 0,...,P-1$）が得られる。

7.10.3.5　線形予測次数の量子化

Table 7-19／JT-G711.0 を用いて P を量子化することにより、量子化された予測次数 $\hat{P}$ が得られる。量子化された線形予測次数 $\hat{P}$ に対する２ビットのインデックス符号 j が復号器に送られる。
実際の線形予測次数 $\hat{P}$ は、フレーム長に依存して４つの候補 $p(j)$、（j=0,1,2,3）から選択される。各フレーム長に対応する４つの予測次数 $p(j)$は、Table 7-19／JT-G711.0 に示される。7.10.3.4 節に記述されたレビンソンダービンアルゴリズムによって計算された予測次数 P が最大予測次数 $P_{\max}$ に満たない場合は、$\hat{P}$ は予測次数 P 以下となるようにする。ただし、$P_{\max}$ はフレーム長に依存して定義され、$\hat{P} = p(j)$ はインデックス j により規定される。そして、P は復号器には伝送されない。

Table 7-19／JT-G711.0 – Index code for the quantized prediction order

(ITU-T G.711.0)

Frame length (*N*)	P_{max}	***p*(*j*)**			
		***p*(0)**	***p*(1)**	***p*(2)**	***p*(3)**
40	4	1	2	3	4
80	8	1	2	6	8
160, 240	10	1	5	8	10
320	12	1	5	10	12
Code		00	01	10	11

実際の次数 $\hat{P}$ は、偏自己相関係数に対して必要な符号長と残差信号に対して必要な符号長の和である推定合計符号長を最小にするように探索される。残差信号の符号長は、分散 $E^{[p(j)]}$ と対数的に比例させることで近似でき、分散は 7.10.3.4 節に記述されているように次式に比例する。

$$\prod_{i=0}^{p(j)-1}\left(1-k_i^2\right)$$

更に、対数関数はマクローリン級数により近似される。結果として、最小となる合計符号長を見つけることは、C_j の最小値を与える予測次数の最適インデックスを見つけることで近似できる。

$$C_j = \sum_{i=0}^{p(j)-1}\left(\gamma_i - \alpha \cdot k_i^2\right) \tag{7-24}$$

ここで、*p*(*j*)は Table 7-19／JT-G711.0 に示されるようにインデックス *j*（*j*=0,1,2,3）に対応する予測次数を表し、γ_i は Table 9-1／JT-G711.0 のテーブル（16）、（17）、（18）、（19）、（20）、（21）、（47）、（48）、（50）、（52）、（54）、（56）及び（58）に示されるように偏自己相関係数を表現するのに必要な符号長であり、αは Table 9-1／JT-G711.0 のテーブル（99）に記述されているようにフレーム長に対応する重み係数である。最小の合計符号長を与える最適予測次数のインデックス *j* が選択され、*p*(*j*)が $\hat{P}$ に割り付けられる。

7.10.3.6　偏自己相関係数の量子化

7.10.3.6.1　40 もしくは 80 サンプルフレームの場合

この場合、全ての係数は非線形写像により量子化される。

1)　2^{15} が乗じられた偏自己相関係数 k_j、（*j*=0,…, $\hat{P}-1$）の絶対値は、k_j^+、（*j*=0,…, $\hat{P}-1$）として定義される。

$$k_j^+ = 2^{15}\left|k_j\right| \qquad (j = 0,..., \hat{P}-1) \tag{7-25}$$

2)　k_j^+ の整数値は量子化に用いられ、（16-U_j）ビットだけ算術右シフトされる。ここで、U_j は、Table 9-1／JT-G711.0 のテーブル（33）、（34）、（35）及び（36）で定義される量子化の際の精度を表すビット数であり、*j* 及び量子化された予測次数 $\hat{P}$ に依存する。

$$V_j = \left\lfloor 2^{-(16-U_j)} k_j^+ \right\rfloor \qquad (j = 1,..., \hat{P}-1) \tag{7-26}$$

3)　偏自己相関係数 k_j、（*j*=0,…, $\hat{P}-1$）が負の場合、Table 9-1／JT-G711.0 のテーブル（66）で与えられるバイアス W_j が V_j、（*j*=0,…, $\hat{P}-1$）に加算される。

$$V_j = \begin{cases} V_j & \text{if } k_j \geq 0 \\ V_j + W_j & \text{if } k_j < 0 \end{cases} \qquad (j = 0,...,\hat{P}-1) \tag{7-27}$$

4) 偏自己相関係数の量子化インデックス X_j、(j=0,…, $\hat{P}-1$) は、値 V_j、(j=0,…, $\hat{P}-1$) 及び Table 9-1／JT-G711.0 のテーブル（25)、(26)、(27）から得られる。量子化値 $\hat{k}_j$ 、(j=0,…, $\hat{P}-1$) に対応するこれらのインデックスは、Table 9-1／JT-G711.0 のテーブル（29)、(30）及び（31）で与えられる。

5) 各インデックス値 X_j、(j=0,…, $\hat{P}-1$) は並べ替えられ値 Y_j、(j=0,…, $\hat{P}-1$) に写像される。そして、Y_j はライス符号により符号化される。これらの値の写像テーブルは、Table 9-1／JT-G711.0 のテーブル（47)、(48)、(50)、(52)、(54)、(56）及び（58）に与えられる。U_j が３もしくは４の場合はライスパラメータは０であり、U_j が５の場合はライスパラメータは１である。

7.10.3.6.2　160、240、320 サンプルフレームの場合

この場合、最初の２つの偏自己相関係数は、以下のように非線形量子化される。

1) 最初の２つの偏自己相関係数 k_j、(j=0,1）の量子化インデックス X_j、(j=0,1）は、$2^{15}k_j$ 、(j=0,1）の整数値から得られる。量子化の際の精度を表すビット数 U_j、(j=0, 1）は、Table 9-1／JT-G711.0 のテーブル（13)、(14）及び（15）に与えられる。

$$X_0 = \begin{cases} \left\lfloor 2^{-(12-U_j)} \left\{ \left\lfloor 2^{-15}\left(-23648 \times 2^{15}k_0 + 16384\right) \right\rfloor + 19552 \right\} \right\rfloor & \text{if } 30801 < 2^{15}k_0 \\ \left\lfloor 2^{-(14-U_j)} \left\{ \left\lfloor 2^{-15}\left(-31103 \times 2^{15}k_0 + 16384\right) \right\rfloor + 18529 \right\} \right\rfloor & \text{if } 24576 < 2^{15}k_0 \leq 30801 \\ \left\lfloor 2^{-(15-U_j)} \left\{ \left\lfloor 2^{-15}\left(-24209 \times 2^{15}k_0 + 16384\right) \right\rfloor + 8559 \right\} \right\rfloor & \text{otherwise} \end{cases} \tag{7-28}$$

$$X_1 = \begin{cases} \left\lfloor 2^{-(12-U_j)} \left\{ \left\lfloor 2^{-15}\left(-23648 \times (-2^{15}k_1) + 16384\right) \right\rfloor + 19552 \right\} \right\rfloor & \text{if } 30801 < -2^{15}k_1 \\ \left\lfloor 2^{-(14-U_j)} \left\{ \left\lfloor 2^{-15}\left(-31103 \times (-2^{15}k_1) + 16384\right) \right\rfloor + 18529 \right\} \right\rfloor & \text{if } 24576 < -2^{15}k_1 \leq 30801 \\ \left\lfloor 2^{-(15-U_j)} \left\{ \left\lfloor 2^{-15}\left(-24209 \times (-2^{15}k_1) + 16384\right) \right\rfloor + 8559 \right\} \right\rfloor & \text{otherwise} \end{cases} \tag{7-29}$$

2) Table 9-1／JT-G711.0 のテーブル（12）にあるように、量子化された偏自己相関係数 $\hat{k}_j$ 、(j=0,1）は、偏自己相関係数量子化インデックス X_j、(j=0,1）から得られる V_j、(j=0, 1）の値に対応する。ただし、１番目の係数 $\hat{k}_0$ の負号は、基に戻すことが必要である（$\hat{k}_0 = -\hat{k}_0$）。

$$V_j = \left(\left\lfloor 2^{(6-U_j)} X_j \right\rfloor + \left\lfloor \left(2^{(6-U_j)} - 1\right)/2 \right\rfloor + 64 \right) \qquad j = 0,1 \tag{7-30}$$

3) インデックス X_j、(j=0,1）はハフマン符号により符号化され、そのテーブルは Table 9-1／JT-G711.0 のテーブル（16)、(17)、(18)、(19)、(20）及び（21）に与えられる。

高次の線形予測次数に対しては（$\hat{P}$ ＞2)、偏自己相関係数は以下のように線形量子化される。

1) $2^{15}k_j$、(j=2,…, $\hat{P}-1$) の整数値は、(15-U_j）ビットだけ算術右シフトされる。ここで、U_j は、Table 9-1／JT-G711.0 のテーブル（13)、(14）及び（15）で定義され、予測次数 j 及び量子化された予測次数 $\hat{P}$ に依存する。Q0 形式のこれらのシフト値は、偏自己相関係数の量子化インデックス X_j、(j=2,…, $\hat{P}-1$) となる。

$$X_j = \left\lfloor 2^{-(15-U_j)} 2^{15} k_j \right\rfloor = \left\lfloor 2^{U_j} k_j \right\rfloor \qquad j = 2,...,\hat{P}-1 \tag{7-31}$$

2) 量子化された偏自己相関係数 $\hat{k}_j$ 、(j=2, …, P–1）は、以下のように計算される。

$$\hat{k}_j = 2^{-15}\left(\left\lfloor 2^{(15-U_j)} X_j \right\rfloor + \left\lfloor 2^{(14-U_j)} \right\rfloor\right) \qquad j = 2,...,\hat{P}-1 \tag{7-32}$$

量子化インデックス X_j、(j=0,…, $\hat{P}-1$) はハフマン符号により符号化され、そのテーブルは Table 9-1／JT-G711.0 のテーブル（16)、(17)、(18)、(19)、(20）及び（21）に与えられる。

7.10.3.7　線形予測係数

線形予測係数は、量子化された偏自己相関係数 $\hat{k}_i$ 、（$i = 0,...,\hat{P}-1$）から得られる。

1)　繰り返し回数を $i = 1$ に設定し、 $a_0^{[0]} = 1.0$ とする。

2)　$a_i^{[i]} = \hat{k}_{i-1}$ とする。

3)　$j = 1,...,i–1$ に対して $a_j^{[i]} = a_j^{[i-1]} + \hat{k}_{i-1} \cdot a_{i-j}^{[i-1]}$ を計算する。

4)　i が量子化された予測次数 $\hat{P}$ に達するまで、i を 1 増加させてステップ２に戻る。

7.10.3.8　漸増線形予測

量子化された予測次数 $\hat{P}$ よりも前の位置のサンプルに対しては、漸増的な線形予測が用いられる。
フレームの最初のサンプルに対しては、予測値はない。

$$\hat{x}_{\mathrm{PCM}}(0) = 0 \tag{7-33}$$

$1 \le n < \hat{P}$ であるインデックス n のサンプルに対しては、予測値 $\hat{x}_{\mathrm{PCM}}(n)$ は、n 次予測係数 $a_i^{[n]}(1 \le i \le n)$ を用いた n 次予測により以下のように得られる。

$$\hat{x}_{\mathrm{PCM}}(n) = \sum_{i=1}^{n} a_i^{[n]} x_{\mathrm{PCM}}(n-i) \qquad n = 1,...,\hat{P}-1 \tag{7-34}$$

結果として、係数の数は、最初のサンプル位置から量子化された予測次数 $\hat{P}$ のサンプル位置まで、サンプル位置と共に増加する。
サンプルインデックス n が $\hat{P} \le n \le N-1$ の場合は、予測されたサンプル値 $\hat{x}_{\mathrm{PCM}}(n)$ は $a_i^{[\hat{P}]}(1 \le i \le \hat{P})$ による $\hat{P}$ 次予測を用いて得られる。

$$\hat{x}_{\mathrm{PCM}}(n) = \sum_{i=1}^{\hat{P}} a_i^{[\hat{P}]} x_{\mathrm{PCM}}(n-i), \qquad n = \hat{P},...,N-1 \tag{7-35}$$

ここで、最後の係数（$\hat{P}$ 次予測の $\hat{P}$ 番目の係数）$a_{\hat{P}}^{[\hat{P}]}$ は Q15 形式で表現され、それ以外の係数は Q12 形式である。

7.10.4　長期予測（LTP）

7.10.4.1　長期予測分析

入力信号の周期的冗長性を取り除くため、160、240 及び 320 サンプルフレームに対して長期予測（ＬＴＰ）符号化モードが利用できる。

7.10.4.1.1　長期予測事前処理

長期予測事前処理は、予め決められたいくつかの閾値を用いて、現フレームが長期予測ツールからの効果を得られるかどうか確認するために、長期予測演算の前に入力信号の特性を分析する。長期予測モジュールは、有効でない場合は適用されない。長期予測モジュールが適用されるか否かに関わらず、7.10.3 節に記述されているように、全ての場合において線形予測（短期予測）が適用される。Figure 7-3／JT-G.711.0 に、決定がどの

ようになされるかについてのフローチャートを示す。

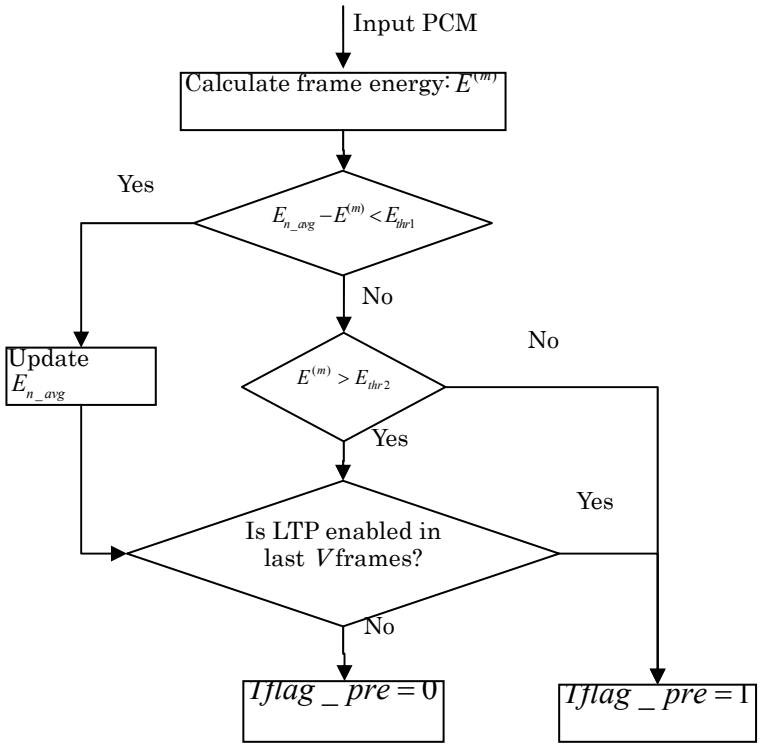

Figure 7-3 / JT-G.711.0 – Diagram of the LTP pre-processor

(ITU-T G.711.0)

現フレーム番号が m のとき、フレームエネルギ $E^{(m)}$は以下のように計算される。

$$E^{(m)} = 30 - \lfloor \log_2(c^{(m)}(0)) \rfloor \tag{7-36}$$

ここで、$c^{(m)}(0)$は、式（7-17）から得られる、現フレームの１番目の自己相関係数である。E_{nbuff}は、背景雑音フレームの場合に $E_{nbuff} = E_{nbuff} + E^{(m)}$のように更新される平均雑音パワを表し、$E_{n\_avg} - E^{(m)} < E_{thr1}$ の条件を満たす。積算された背景雑音フレーム数が４になると、エネルギ閾値 E_{n_avg} は $E_{n\_avg} = E_{nbuff}/4$ のように更新され、E_{nbuff} は０にリセットされる。最初の Q フレームにおける初期平均背景雑音 E_{n_avg} エネルギは以下のように計算される。

$$E_{n\_avg} = \frac{1}{Q}\sum_{m=0}^{Q-1} E^{(m)} \quad 0 \le m < Q \tag{7-37}$$

Q はフレーム長に依存し、160 サンプルフレームに対しては８、240 及び 320 サンプルフレームに対しては４である。

なお、Q はフレーム長が変化したときに更新される。長期予測事前処理フラグ *Tflag_pre* は、長期予測が有効であると判断された場合に１に設定され、そうでない場合は０に設定される。$0 \le m < Q$ のフレームに対しては、長期予測事前処理フラグ *Tflag_pre* は１に初期化される。２つの定数エネルギ閾値が、長期予測事前処理に対して定義される。E_{thr1} は２であり、E_{thr2} はフレーム長 160、240 及び 320 に対して各々15、14 及び 13 である。また、長期予測事前処理は、直近の V フレームの長期予測フラグを必要とする。V はフレーム長に依存し、フレ

ーム長 160、240 及び 320 に対して各々９、５、４である。
全体の平均処理量を低減するために、長期予測事前処理は、無音部分と非周期的背景雑音部分に対して長期予測ツールは使用しない。

7.10.4.1.2　ＰＣＭ信号領域における長期予測ピッチ探索

長期予測ピッチ探索は、線形ＰＣＭ信号領域において、ピッチラグ推定値 T を得るために最初に実行される。その後、ピッチ推定は精細化される（7.10.4.1.3 節参照）。このピッチ精細化は線形予測残差信号領域にて実行され、現フレームは可変長のサブフレームに分割される。
長期予測探索の処理量を低減するため、入力信号 $x_{\mathrm{PCM}}(n)$を 2:1 にダウンサンプリングされたものが用いられる。ＰＣＭ信号の最初の 160 サンプルをダウンサンプリングするために、低域通過フィルタが用いられる。$\bar{x}_{\mathrm{PCM}}(n)$ は、以下により得られるダウンサンプリングされたＰＣＭ信号である。

$$\bar{x}_{\mathrm{PCM}}(n)=\begin{cases}x_{\mathrm{PCM}}(0) & n=0\\ 0.25\bar{x}_{\mathrm{PCM}}(n-1)+0.25x_{\mathrm{PCM}}(2n-1)+0.5x_{\mathrm{PCM}}(2n) & n=1,2,...,79\end{cases} \tag{7-38}$$

ピッチ範囲は、８kHz サンプリング領域において$[T_{\min}, T_{\max}]$に制限され、$T_{\min}=20$ 及び $T_{\max}=84$ である。
最初に、以下を満たすような $\left|\bar{x}_{\mathrm{PCM}}(n)\right|$ の最大値を与えるサンプル位置 $n_{\max\_\mathrm{val}}$ が、ダウンサンプルされた領域の範囲 $n\in\left[\frac{T_{\max}}{2},79\right]$ において探索される。

$$\left|\bar{x}_{\mathrm{PCM}}(n_{\max\_val})\right|\geq\left|\bar{x}_{\mathrm{PCM}}(n)\right| \tag{7-39}$$

探索範囲において最大値となるサンプル数が１より多い場合、$n_{\max\_\mathrm{val}}$ は最大値となる最後のサンプルの位置に設定される。
その後、$n_{\max\_\mathrm{val}}$ 近傍の区間$[w_{\min}, w_{\max}]$における $\bar{x}_{\mathrm{PCM}}(n)$ に対して、ピッチラグ T が探索される。上限及び下限の境界は以下のように計算される。

$$\begin{cases}w_{\min}=\max\left\{\frac{T_{\max}}{2},n_{\max\_\mathrm{val}}-15\right\}\\ w_{\max}=\min\left\{80,n_{\max\_\mathrm{val}}+15\right\}\end{cases} \tag{7-40}$$

ダウンサンプリングされたＰＣＭ信号 $\bar{x}_{\mathrm{PCM}}(n)$ と、各ピッチ候補 k、$k\in\left[\frac{T_{\min}}{2},\frac{T_{\max}}{2}\right)$ に対して固定のピッチ利得 0.745 を用いたそれと対応する長期予測信号 $\bar{x}_{\mathrm{PCM}}(n-k)$ との間の差分として、ダウンサンプリングされた領域の長期予測残差信号 $x_k(n)$が計算される。

$$x_k(n)=\bar{x}_{\mathrm{PCM}}(n)-0.745\bar{x}_{\mathrm{PCM}}(n-k),\ n=w_{\min},...,w_{\max}-1 \tag{7-41}$$

ただし、ピッチ予測利得を符号化するためのビットは必要としない。
ピッチ候補に k 対する $x_k(n)$のエネルギは、以下のように計算される。

$$E_k=\sum_{n=w_{\min}}^{w_{\max}-1}x_k(n)\cdot x_k(n) \tag{7-42}$$

処理量を低減するために、$\frac{T_{\min}}{2}<k<\frac{T_{\max}}{2}$ に対する $\mathrm{sgn}\left(\bar{x}_{\mathrm{PCM}}(n_{\max_\mathrm{val}})\right)=\mathrm{sgn}\left(\bar{x}_{\mathrm{PCM}}(n_{\max_\mathrm{val}}-k)\right)$の条件が満たされる場合のみ、$E_k$が計算される。

ダウンサンプリング領域におけるピッチラグ T は、最大エネルギ E_k（$E_T = \min\{E_k\}$）を与えるものである。ダウンサンプリング領域におけるピッチラグ T が 20 よりも小さい場合、$x_{\mathrm{PCM}}(n)$に対応する最適なピッチラグ T は、$2T$ に更新される。そうでない場合は、２重ピッチの影響を避けるため、以下の手法が用いられる。最適なピッチ候補 T は、最大 R_{norm} を持つものである。

$$T = \begin{cases} T & \text{if } R_{norm}(T) \geq R_{norm}(2T) \\ 2T & \text{otherwise} \end{cases} \tag{7-43}$$

ここで、

$$R_{norm}(j) = \frac{\sum_{n=j}^{N-1} x_{\mathrm{PCM}}(n) \cdot x_{\mathrm{PCM}}(n-j)}{\sum_{n=j}^{N-1} x_{\mathrm{PCM}}(n-j) \cdot x_{\mathrm{PCM}}(n-j)} \qquad j = T \text{ or } 2T \tag{7-44}$$

である。

7.10.4.1.3　適応フレーム分割によるＰＣＭ残差信号領域におけるピッチ精細化

7.10.4.1.2 節で見つけられた最初のピッチ推定値 T は、線形予測残差信号領域にて精細化される。
進歩的な線形予測の影響を避けるため、最初の $\hat{P}$ サンプルはこの精細化に使われないことに注意が必要である。１つのピッチ周期におけるそれに続く τ_0 サンプル（$\tau_0 = \min\{T_{\max} - 1, T + 3\}$ として初期化される）もまた使われない。
残りの（$N - \hat{P} - \tau_0$）フレームサンプルは、長さ l_{sfr} の N_{sfr} 個のサブフレームに適応的に分割される。ここで、160 サンプルフレームに対しては $N_{sfr} = 1$、240 サンプルフレームに対しては N_{sfr}、320 サンプルフレームに対しては $N_{sfr} = 3$ である。初期サブフレーム長 l_{sfr} は最初のピッチ推定値 T に依存し、以下のように設定される。

$$l_{sfr} = \left\lfloor \left(N - \hat{P} - \tau_0 \right) / N_{sfr} \right\rfloor \tag{7-45}$$

最初のサブフレームに対して、ピッチ精細化探索に対する T 周辺の制限された範囲の下限及び上限境界である $\tau_{\min}$ 及び $\tau_{\max}$ は、以下のように定義される。

$$\begin{cases} \tau_{\min} = \max\{T_{\min}, T - 3\} \\ \tau_{\max} = \tau_0 \end{cases} \tag{7-46}$$

その後、正規化相関 $\widetilde{R}_{norm}(j)$、$j \in [\tau_{\min}, \tau_{\max}]$ の最大値を与える最適なピッチラグ τ_1 を得るために、ＰＣＭ領域の線形予測残差信号に対して、範囲 $[\tau_{\min}, \tau_{\max}]$ におけるピッチ探索が実行される。

$$\widetilde{R}_{norm}(j) = \frac{\sum_{n=\hat{P}}^{\hat{P}+l_{sfr}-1} r_{\mathrm{PCM}}(n+j) \cdot r_{\mathrm{PCM}}(n)}{\sum_{n=\hat{P}}^{\hat{P}+l_{sfr}-1} r_{\mathrm{PCM}}(n+j) \cdot r_{\mathrm{PCM}}(n+j)} \tag{7-47}$$

ここで、

$$r_{\mathrm{PCM}}(n) = x_{\mathrm{PCM}}(n) - \hat{x}_{\mathrm{PCM}}(n) \tag{7-48}$$

である。その後 τ_0 は τ_1 で更新される。
次のサブフレーム i、($2 \le i \le N_{sfr}$) に対しては、探索は、直近のサブフレームの最適なピッチラグ τ_{i-1} の周辺の狭い領域に制限される。

$$\tau_i \in [\tau_{i-1} - 2, \tau_{i-1} + 2] \qquad 2 \le i \le N_{sfr} \tag{7-49}$$

そして、最大正規化相関 $\widetilde{R}_i$ が、$r_{\mathrm{PCM}}(n)$、$\hat{P} + \tau_0 + (i-1)\cdot l_{sfr} \le n < \hat{P} + \tau_0 + i \cdot l_{sfr}$ の l_{sfr} サンプルで評価される。ここで、

$$\widetilde{R}_i(j) = \sum_{n=\hat{P}+\tau_0+(i-1)\cdot l_{sfr}}^{\hat{P}+\tau_0+i\cdot l_{sfr}-1} r_{PCM}(n)\cdot r_{PCM}(n-j), j \in [\tau_{i-1} - 2, \tau_{i-1} + 2], 2 \le i \le N_{sfr} \tag{7-50}$$

である。最初のサブフレームピッチ τ_1 は 6 ビットで符号化される。他のサブフレームのピッチラグは、6.9.2 節で記述されているライス符号化で差分が符号化される。このとき、ライスパラメータは 0 である。符号化されたピッチラグの差分は、$\Delta\tau_i = \tau_{i-1} - \tau_i, 2 \le i \le N_{sfr}$ として計算される。$\Delta\tau_i \ge 0$ のときは、値 $2\times\Delta\tau_i$ が $Rice(0,2\Delta\tau_i)$ により符号化され、$\Delta\tau_i < 0$ のときは、値 $-2\times\Delta\tau_i - 1$ が $Rice(0,-2\Delta\tau_i - 1)$ により符号化される。

7.10.4.1.4　長期予測モード決定

以下の演算は、長期予測符号化が最大圧縮比を生じさせるかどうかを決定する。長期予測残差信号エネルギが重み付け線形予測残差信号のエネルギよりも小さい場合、長期予測が有効となり、*Tflag* が 1 に設定される。そうでない場合、長期予測は無効となり、*Tflag* が 0 に設定される。
7.10.4.1.3 節で実行される N_{sfr} サブフレームへの適応フレーム分割は、以下のように修正される。
サブフレーム長 l_{sfr} は、式（7-45）を用いて更新される。このとき、$\tau_0 = \tau_1$ である。
i 番目のサブフレームの開始位置である下限境界 b_i は、以下のように計算される。
$b_i = b_0 + (i-1)\cdot l_{sfr}$、$0 < i \le N_{sfr}$、ここで、$b_0 = \hat{P} + \tau_0$ 及び $b_{N_{sfr}+1} = N$ である。
長期予測寄与 $\bar{x}_{\mathrm{LTP}}(n)$ は、以下のように記述される。

$$\bar{x}_{\mathrm{LTP}}(n) = \begin{cases} 0 & n = 0,\ldots,b_0 - 1 \\ g_i \cdot r_{\mathrm{PCM}}(n-\tau_i) & b_i \le n < b_{i+1}, i = 1,\ldots,N_{sfr} \end{cases} \tag{7-51}$$

ここで、$g_i = \begin{cases} 0.745, \tau_i < 40 \\ 0.705, \tau_i \ge 40 \end{cases}$ は、サブフレームピッチに対応する i 番目のサブフレームのピッチ利得である。

線形予測残差信号のエネルギは E_{LPC}、以下のように計算される。

$$E_{\mathrm{LPC}} = \sum_{n=n_0}^{N-1} r_{\mathrm{PCM}}(n)\cdot r_{\mathrm{PCM}}(n) \tag{7-52}$$

そして、長期予測残差信号のエネルギ E_{LTP} は、以下のように計算される。

$$E_{\mathrm{LTP}} = \sum_{n=n_0}^{N-1} r_{\mathrm{LTP}}(n)\cdot r_{\mathrm{LTP}}(n) \tag{7-53}$$

ここで、$r_{\mathrm{LTP}}(n) = r_{\mathrm{PCM}}(n) - \breve{x}_{\mathrm{LTP}}(n)$ である。
最後に、*Tflag* が以下のように設定される。

$$Tflag = \begin{cases} 1 & \text{if } E_{\mathrm{LTP}} < 0.875 \cdot E_{\mathrm{LPC}} \\ 0 & \text{if } E_{\mathrm{LTP}} \geq 0.875 \cdot E_{\mathrm{LPC}} \end{cases} \tag{7-54}$$

Tflag に基づき、長期予測されたＰＣＭ信号 $\hat{x}_{\mathrm{LTP}}(n)$ は、以下のようになる。

$$\hat{x}_{\mathrm{LTP}}(n) = \begin{cases} 0 & Tflag = 0 \\ \breve{x}_{\mathrm{LTP}}(n) & Tflag = 1 \end{cases} \tag{7-55}$$

7.10.5　予測残差の符号化

7.10.5.1　残差計算

予測残差 *r(n)*は、*int8* 領域において以下のように計算される。

$$r(n) = x_{\mathrm{int8}}(n) - \hat{x}_{\mathrm{int8}}(n) \tag{7-56}$$

ここで、A 則入力に対する $\hat{x}_{\mathrm{int8}}(n)$ は、以下のように得られる。

$$\begin{aligned} x_{\mathrm{int8}}(n) &= f_{A\to\mathrm{int8}}(I_A(n)) \\ \hat{x}_{\mathrm{int8}}(n) &= f_{A\to\mathrm{int8}}(f_{\mathrm{PCM}\to A}(\hat{x}_{\mathrm{PCM}}(n) + \hat{x}_{\mathrm{LTP}}(n))) \end{aligned} \tag{7-57}$$

ここで、μ 則入力に対する $\hat{x}_{\mathrm{int8}}(n)$ は、以下のように得られる。

$$\begin{aligned} x_{\mathrm{int8}}(n) &= f_{\mu\to\mathrm{int8}}(I_\mu(n)) \\ \hat{x}_{\mathrm{int8}}(n) &= f_{\mu\to\mathrm{int8}}(f_{\mathrm{PCM}\to\mu}(\hat{x}_{\mathrm{PCM}}(n) + \hat{x}_{\mathrm{LTP}}(n))) \end{aligned} \tag{7-58}$$

あるいは、

$$\begin{aligned} r(n) &= x_{\mathrm{int8}(*,*)}(n) - \hat{x}_{\mathrm{int8}(*,*)}(n) \\ &= f_{\mathrm{int8}\to\mathrm{int8}(*,*)}(f_{\mu\to\mathrm{int8}}(I_\mu(n))) - f_{\mathrm{int8}\to\mathrm{int8}(*,*)}(f_{\mu\to\mathrm{int8}}(f_{\mathrm{PCM}\to\mu}(\hat{x}_{\mathrm{PCM}}(n) + \hat{x}_{\mathrm{LTP}}(n)))) \end{aligned} \tag{7-59}$$

となる。ここで、写像関数 $f_{\mathrm{int8}\to\mathrm{int8}(*,*)}$ は、7.10.5.2 節で詳述する。また、$\hat{x}_{\mathrm{PCM}}(n)$ は、7.10.3.8 節で詳述した予測サンプルであり、$\hat{x}_{\mathrm{LTP}}(n)$ は、7.10.4.1.4 節の(7-55)式から得られた長期予測サンプルである。

7.10.5.2　正負零値写像

40 サンプル/フレームより大きい μ 則フレームにおいて、正のゼロ値 0^+と負のゼロ値 0^-の両方、あるいは、どちらか一方が得られなかった場合、線形予測残差計算に対して零値写像処理ツールが適用される。Table 7-20／JT-G.711.0 に、４つの条件に対する写像と正負零値フラグ符号を示す。

$$x_{\mathrm{int8}(*,*)}(n) = \begin{cases} f_{\mathrm{int8}\to\mathrm{int8}(Y,Y)}(f_{\mu\to\mathrm{int8}}(I_\mu(n))) & \text{when there are both } 0^+ \text{ and } 0^- \text{ in a frame} \\ f_{\mathrm{int8}\to\mathrm{int8}(Y,N)}(f_{\mu\to\mathrm{int8}}(I_\mu(n))) & \text{when there is no } 0^- \text{ in a frame} \\ f_{\mathrm{int8}\to\mathrm{int8}(N,Y)}(f_{\mu\to\mathrm{int8}}(I_\mu(n))) & \text{when there is no } 0^+ \text{ in a frame} \\ f_{\mathrm{int8}\to\mathrm{int8}(N,N)}(f_{\mu\to\mathrm{int8}}(I_\mu(n))) & \text{when there is no } 0^+ \text{ nor } 0^- \text{ in a frame} \end{cases} \tag{7-60}$$

ここで、写像 $f_{\mathrm{int8}\to\mathrm{int8}(Y,N)}$ と $f_{\mathrm{int8}\to\mathrm{int8}(N,Y)}$ は、正負零値フラグ符号を除き同一である。

Table 7-20／JT-G.711.0　PM zero mapping table for *int8*

(ITU-T G. 711.0)

μ-law value	**μ-law to int8 mapping**	**Both 0^+ and 0^- are observed**	**No 0^+ nor 0^- are observed**	**No 0^- is observed in a frame**	**No 0^+ is observed in a frame**
$I_\mu(n)$	$x_{\text{int8}}(n)$	$x_{\text{int8}(Y,Y)}(n)$	$x_{\text{int8}(N,N)}(n)$	$x_{\text{int8}(Y,N)}(n)$	$x_{\text{int8}(N,Y)}(n)$
0x80	+127	+127	+126	+127	
0x81	+126	+126	+125	+126	
⋮	⋮	⋮	⋮	⋮	
0xFE	+1	+1	0	+1	
0xFF	0	0	0	0	
0x7F	−1	−1	0	0	
0x7E	−2	−2	−1	−1	
⋮	⋮	⋮	⋮	⋮	
0x01	−127	−127	−126	−126	
0x00	−128	−128	−127	−127	
PM zero flag		0	100	101	110

7.10.5.3　サブフレーム分割

160、240、320 サンプル長のフレームに対し、予測残差信号のエントロピー符号化は、全フレーム、あるいは、80 サンプルのサブフレームのどちらかで動作する。どちらで動作するかの判定は、以下のステップに基づき決定される。

1）　残差信号 *r(n)*の *i* 番目サブフレーム（$i=0,...,N'_{sfr}-1$、ここで N'_{sfr} は、160、240、320 サンプルのフレームに対して、それぞれ２、３、４のサブフレーム数を表す）の平均絶対値 $\bar{r}_i$ と、フレーム全体の平均絶対値 $\bar{r}_g$ は、以下のように計算する。

$$\bar{r}_i = \begin{cases} \dfrac{1}{\left(80-\min\left\{2,\hat{P}\right\}\right)} \displaystyle\sum_{n=\min\left\{2,\hat{P}\right\}}^{79} |r(n)| & (i=0) \\ \dfrac{1}{80}\displaystyle\sum_{n=0}^{79}|r(n+80\times i)| & (1\le i\le N'_{sfr}-1) \end{cases} \tag{7-61}$$

$$\bar{r}_g = \sum_{i=0}^{m-1} \bar{r}_i / N'_{sfr} \tag{7-62}$$

フレームにおける最初の $\min\left\{2,\hat{P}\right\}$ サンプルは、通常２サンプルで、$\bar{r}_0$ の計算に対してスキップされる（したがって、$\bar{r}_g$ の計算には使用されない）が、そのフレームに対する予測次数 $\hat{P}$ が１の場合は、１サンプルのみスキップされる。

2）　*i* 番目サブフレームに対する平均絶対値 $\bar{r}_i$ から、それぞれのサブフレームに対する分割パラメータ *Si* を計算する。

$$S_i = \lfloor \log_2(2\ln(2)\bar{r}_i) + \lambda \rfloor \tag{7-63}$$

また、フレーム全体の平均絶対値 $\bar{r}_g$ から、全体分割パラメータ *Sg* を計算する。

$$S_g = \lfloor \log_2(2\ln(2)\bar{r}_g) + \lambda \rfloor \tag{7-64}$$

ここで、λは 0.3 のバイアスである。

これは、すなわち $S_g = \lfloor \log_2((2^{(1+\lambda)}\ln(2))\bar{r}_g) \rfloor$ となる。ここで、$\bar{r}_g$ は Q7 で、定数部分 $(2^{(1+\lambda)}\ln(2))$ は、Q14 で、その積は、Q6 フォーマットで保持される。$\lfloor \log_2() \rfloor$ 演算は、$\max(8 - \text{norm\_s}(\text{product in Q6}), 0)$ の演算によって処理される。ここで、norm_s()は、16 ビット変数を正規化するのに必要な左シフト数を与える ITU-T G.191 ソフトウェアツールライブラリ STL2005 v2.2 の基本演算関数である。

3） サブフレーム間の分割パラメータ差分 ΔS_i を計算する。

$$\Delta S_i = S_{i+1} - S_i \qquad (i = 0, ..., N'_{sfr} - 2) \tag{7-65}$$

各サブフレームの $\bar{r}_i$ と平均値 $\bar{r}_g$ との偏差が、下記に示す閾値未満かどうかを調べる。

$$\left|\bar{r}_i - \bar{r}_g\right| \le (N'_{sfr}/16)\bar{r}_g \qquad (i = 0, ..., N'_{sfr} - 1) \tag{7-66}$$

4） 全てのサブフレームに対して、ΔS_i が全て 0 か、上記条件を満たしている場合は、全フレームに対して全体分割パラメータ S_g を適用する。その他の場合は、各サブフレームに対して *Si* を適用する。Table 7-21／JT-G.711.0 に、フレーム長とそのサブフレーム分割数に対する符号化法を示す。１番目の分割パラメータ S_0 、あるいは、 S_g は、Table 7-22／JT-G.711.0 に示すように、ライスパラメータ“１”の符号なしライス符号で符号化される。２番目から最後のサブフレームまでの分割パラメータは、差分が符号化される。差分値 ΔS_i は、Table 7-23／JT-G.711.0 に示すように、ライスパラメータ“０”の符号付きライス符号で符号化される。フレーム長が 160 サンプルの場合、サブフレーム分割の判定は、ΔS_0 (= $S_1 - S_0$)のライス符号結果によって示される。ΔS_0 のライス符号が１の場合、ΔS_0 は０を意味し、そのフレームの全サンプルに対して、S_0 が共通的に適用されるため、サブフレーム分割されない。その他のフレーム長に対しては、サブフレーム分割に対して、明示的なフラグが適用される。

5） 40、80 サンプルのフレーム長に対しては、サブフレーム分割が適用されない。フレームにおける全体平均絶対値 $\bar{r}_g$ は、以下のように計算される。

$$\bar{r}_g = \frac{1}{\left(N - \min\left(2, \hat{P}\right)\right)} \sum_{n=\min\{2,\hat{P}\}}^{N-1} |r(n)| \tag{7-67}$$

全体分割パラメータ S_g は、以下のように計算される。

$$S_g = \lfloor \log_2(2\ln(2)\bar{r}_g) + \lambda \rfloor \tag{7-68}$$

ここで、λ は、40 サンプルフレームに対して 0.5 の固定バイアス、80 サンプルフレームに対して 0.3 の固定バイアスである。

Table 7-21／JT-G.711.0 – Sub-frames and corresponding separation parameters
(ITU-T G. 711.0)

Frame length (*N*)	**Sub division**	**Flag**	**Number of sub-frames**	**Separation parameter codes for sub-frames**			
				1st	**2nd**	**3rd**	**4th**
40	No	–	1	S_g	–	–	–
80	No	–	1	S_g	–	–	–
160	No	–	1	S_g	$\Delta S_0 = 0$	–	–
	Yes	–	2	S_g	ΔS_0	–	–
240	No	0	1	S_g	–	–	–
	Yes	1	3	S_g	ΔS_0	ΔS_1	–
320	No	0	1	S_g	–	–	–
	Yes	1	4	S_g	ΔS_0	ΔS_1	ΔS_2

Table 7-22 ／JT-G.711.0– Rice code for S_0 or S_g (unsigned Rice code with separation parameter 1)

(ITU-T G. 711.0)

	LTP condition	**Value of S_0 or S_g**							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Code	No	11	011	10	010	0010	0011	00010	00011
	Yes	0011	011	11	10	010	0010	00010	00011

Table 7-23／JT-G.711.0 – Rice code for ΔS_i (Rice code with separation parameter 0)

(ITU-T G. 711.0)

ΔS_i	..	−3	−2	−1	0	1	2	..
Code	…	000001	0001	01	1	001	00001	…

7.10.5.4 最初のサンプルおよび、それ以降のサンプルに対する分割パラメータ

算出した分割パラメータ S_g 、あるいは S_i, $(0 \le i \le N'_{sfr} - 1)$ は、フレームの最初の $\min\{2, \hat{P}\}$ サンプルを除き、残差サンプルのエントロピー符号化に用いられる。残差信号が、7.10.5.5 節で述べたライス符号で符号化された場合、そのライスパラメータは、分割パラメータに等しいのに対して、残差信号が、7.10.5.6 節で述べるエスケイプハフマン（E-Huffman）によって符号化される場合は、バイアスされる。

スキップされる予測残差信号の最初の１、２サンプルは、Table 7-24／JT-G.711.0 に示す修正分割パラメータによって符号化される。修正分割パラメータは、１次、２次の量子化偏自己相関係数 $\hat{k}_0$ 、$\hat{k}_1$ に依存し、また、長期予測が適用されたかどうかにも依存する。$(1 < \hat{P})$ 、$(2^{15}\hat{k}_1 < -28000)$ と（長期予測が不適用）が同時に満たされる場合、１番目、２番目サンプルに対する分割パラメータは、$(30800 < 2^{15}\hat{k}_0)$ と（長期予測が適用）の条件に対するパラメータと同じになるよう修正される。但し、$\hat{k}_0$ の他条件と Table 7-24／JT-G.711.0 の長期予測を考慮しない。

Table 7-24／JT-G.711.0 – Modified Rice parameter for first and second samples of the frame

(ITU-T G. 711.0)

Position	**LTP enabled**	**PARCOR**	***S*=0**	***S*=1**	***S*=2**	***S*=3**	***S*=4**	***S*=5**	***S*=6**	***S*=7**
0	No/Yes	$2^{15}\hat{k}_0 \le 26500$	1	2	3	4	5	6	7	7
0	No	$26500 < 2^{15}\hat{k}_0 \le 30800$	2	3	4	5	6	6	7	7
0	Yes	$26500 < 2^{15}\hat{k}_0 \le 30800$	5	5	6	6	6	6	7	7
0	No	$30800 < 2^{15}\hat{k}_0$	4	5	6	6	6	6	7	7
0	Yes	$30800 < 2^{15}\hat{k}_0$	6	6	6	6	6	6	7	7
1	No/Yes	$2^{15}\hat{k}_0 \le 26500$	1	2	3	4	5	6	7	7
1	No	$26500 < 2^{15}\hat{k}_0 \le 30800$	1	2	3	4	5	6	7	7
1	Yes	$26500 < 2^{15}\hat{k}_0 \le 30800$	3	3	4	4	5	6	7	7
1	No	$30800 < 2^{15}\hat{k}_0$	2	3	4	4	5	6	7	7

1	Yes	$30800 < 2^{15}\hat{k}_0$	4	4	5	5	5	6	7	7

7.10.5.5 予測残差のライス符号化

短フレーム長、例えば 40 サンプルフレームに対して、6.9.2 節で述べたライス符号ツールが、予測残差を符号化するのに適用される。このツールは、１進符号の連続に対して“０”を、終止に対して“１”を用いている。ここで、全体分割パラメータ $S(=S_g)$が、ライスパラメータとして用いられ、$j(n)$ と $k(n)$ が、 $r(n)$ の余りと商を表す。これらは、以下に示すように、S から算出される。

S=0 かつ $k(n)$0 後の最初の１が現れた場合

$$k(n)=\begin{cases}2\times r(n) & \text{if } r(n)\geq 0\\ -2\times r(n)-1 & \text{if } r(n)<0\end{cases} \tag{7-69}$$

その他の場合、例えば、S>0 かつ $k(n)$0 後の最初の１が現れ、そして、S ビット表現の $j(n)$が以下の場合

$$k(n)=\begin{cases}\left\lfloor 2^{-(S-1)}r(n)\right\rfloor & \text{if } r(n)\geq 0\\ \left\lfloor 2^{-(S-1)}\{-r(n)-1\}\right\rfloor & \text{if } r(n)<0\end{cases} \tag{7-70}$$

$$j(n)=\begin{cases}r(n)\otimes\left\{2^{(S-1)}-1\right\}+2^{(S-1)} & \text{if } r(n)\geq 0\\ \{-r(n)-1\}\otimes\left\{2^{(S-1)}-1\right\} & \text{if } r(n)<0\end{cases} \tag{7-71}$$

7.10.5.6 予測残差のエスケイプハフマン（E-Huffman）符号化

40 サンプル以上のフレーム長に対しては、エスケイプハフマン符号化が予測残差に適用される。

最初に、フレーム、あるいは、サブフレームにおける残差信号は、式（7-69）、（7-70）、（7-71）と 7.10.5.3 節、7.10.5.4 節で述べた分割パラメータ S ($=S_g$ or = $S_i, (0\leq i\leq N'_{sfr}-1)$)を用い、商と余りの２つに分解される。

商 $k(n)$は、Table 7-25／JT-G.711.0 に示すエスケイプ符号を含むハフマン符号を用いて符号化される。４つの E-Huffman テーブル（E-Huffman table 0, 1, 2, 3）が存在し、それぞれのテーブルには、商の値が *maxCode* に等しいかそれ以上であることを示すエスケイプ値が存在する。Table 7-25／JT-G.711.0 に示すインデックス符号は、フレームあるいは、サブフレームに対して選択された E-Huffman テーブルを指定する符号ビットである。$ext(k(n)-maxCode)$は、エスケイプ符号に続く拡張符号である。分離パラメータの Sg か Si の値によって、ext(k(n)--maxCode)に使用される符号化則は、異なります。

余り $j(n)$は、サンプル毎の S_g 、あるいは、S_i を用いて符号化される。S_g 、あるいは、S_i が 0 の場合は、k の値が E-Huffman テーブルの最大符号値 *maxCode* 以上となる k の値を符号化する拡張符号化則として、6.9.1 節で述べたユナリ符号（１進符号）が適応される。例えば、$k(n)-maxCode$ のユナリ符号（１進符号）は、E-Huffman table 1 のエスケイプ符号’0011’の後に続く。S_g 、あるいは、S_i が 0 より大きい場合、ユナリ符号（１進符号）の代わりに、これら k を符号化するために拡張符号化則として 6.9.2 節で述べたライスパラメータ“１”のライス符号が適用される。例えば、ライス符号（1,$k(n)-maxCode$）、これはライスパラメータ“１”の $k(n)-maxCode$ のライス符号を表す、は E-Huffman table 1 のエスケイプ符号’0011’の後に続く。

４つのテーブル値（E-Huffman table 0, 1, 2, 3）から適切なハフマンテーブルを選択するために、累積相対符号長 U_iが算出される。各フレームに対し、符号化単位（フレームあるいは、サブフレーム）において、全てのサンプルをスキャンすることで、k の値のヒストグラムが生成される。E-Huffman table 3 を除く各 E-Huffman テーブルに対し、当該ヒストグラムの頻度値が、E-Huffman table 3 の符号の長さと比較した当該 E-Huffman テーブルの差分符号の長さと乗算され、その積の値の全ての k の値に対して累算される。その結果は、i 番目の E-Huffman テーブルに対する累積相対符号長 $U_i(i=0,1,2)$を与える。U_iの計算には、選択された E-Huffman テーブルを示すインデックスに必要な符号の相対符号長も考慮される。全ての U_i が正の場合、E-Huffman table 3 が適用され、それ以外の場合は、U_i の最小値を与えるテーブルが適用される。

Table 7-25／JT-G.711.0 – E-Huffman codes for residual coding

(ITU-T G. 711.0)

***k*-values**	**E-Huffman table 0**	**E-Huffman table 1**	**E-Huffman table 2**	**E-Huffman table 3**
	maxCode=7	***maxCode=7***	***maxCode=7***	***maxCode=6***
	Index: 01	**Index: 000**	**Index: 001**	**Index: 1**
0	01	1	1	1
1	10	01	01	01
2	11	0001	001	001
3	001	0010	00001	0001
4	0001	00001	00010	00001
5	00001	000000	000000	000000
6	000000	000001	000001	000001 + *ext*(*k*(*n*) – 6)
7	000001 + *ext*(*k*(*n*) – 7)	0011 + *ext*(*k*(*n*) – 7)	00011 + *ext*(*k*(*n*) – 7)	-

7.11 端数ビット符号化ツール（Fractional-bit coding tool）

7.11.1 端数ビット符号化器概要

端数ビット符号化は、$x_{\text{int8}}(n)$すなわち *int8* フレームデータで動作する。Table 7-26／JT-G.711.0 に端数ビット符号化ツールのビットパッキングフォーマットを示す。

Table 7-26／JT-G.711.0 – Bit packing format of the fractional-bit coding tool

(ITU-T G. 711.0)

Frame length (*N*)	**40, 80, 160, 240, 320**	**Note**
Prefix code for fractional-bit coding tool (for each frame length conjunction with the ranges *R* and *R′*)	8 bits	See clauses 7.2 and 7.11.4
Codebits for V_ks	Variable	See clause 7.11.3

7.11.2 データ領域削減

式（7-1）を用いて計算されるデータ領域は、フレーム内で異なった値の数を表すために修正される。

$$R' = R - \sum_{k=0}^{R-1} \delta_{v_k} \tag{7-72}$$

ここで、v_kは、$X_{\min} < v_k < X_{\max}$（$X_{\max}$と$X_{\min}$は$X_{\max} = \max\{x_{\text{int8}}(n) \mid n = 0,...,N-1\}$と$X_{\min} = \min\{x_{\text{int8}}(n) \mid n = 0,...,N-1\}$である）となるデータ領域における整数である。領域削減フラグδ_{v_k}は、以下のように得られる。

$$\delta_{v_k} = \begin{cases} 1 & \text{if value } v_k \text{ does not exist within a frame} \\ 0 & \text{otherwise} \end{cases} \tag{7-73}$$

Table 7-28／JT-G.711.0 に示す代表要素値に対して、データ領域削減は、フレーム内に$v_k = -1$が存在しない場合が考慮されている。この場合、

$$R' = R - \delta_{-1} \tag{7-74}$$

ここで、

$$\delta_{-1} = \begin{cases} 1 & \text{if value } v_k = -1 \text{ does not exist within a frame} \\ 0 & \text{otherwise} \end{cases} \tag{7-75}$$

である。

データ領域 R と R' は、ほとんど等しいのに対して、フレーム数については、R' の方が R より少ない。

7.11.3 サンプル毎の端数ビット符号化器

サンプル毎の端数ビット符号化器は、ブロック分割動作である。フレーム長 N、ブロック長 M、データ領域 R の場合、M の数のサンプルブロックが、多項式 V_k を計算するのに使用される。

$$V_k = l_{kM} + l_{kM+1}R + \ldots + l_{kM+M-1}R^{M-1} \qquad k = 0,\ldots,\frac{N}{M}-1 \tag{7-76}$$

ここで、$l_n = x_{\text{int8}}(n) - X_{\min}$, ($n$=0,...,$N$–1) は、０ から $R-1$ の範囲おける、正規化されたデータである。

各多項式 V_k は、$\lceil M \log_2(R) \rceil$ ビットを用いて、ビットストリーム中に配置される。

端数ビット符号化ツールは、Table 7-28／JT-G.711.0 に示すように、代表する要素値をもつ選択フレームについて符号化するサンプル毎の端数ビットを結合したデータ範囲削減を利用する。データ要素は、全ての値が０から $R'-1$ の範囲に入るよう、はじめに正規化される。

$$l_n = \begin{cases} x_{\text{int8}}(n) - X_{\min} & \text{for } x_{\text{int8}}(n) < -1 \\ x_{\text{int8}}(n) - X_{\min} - \delta_{-1} & \text{otherwise} \end{cases} \qquad \text{n} = 0,\ldots,\text{N}-1 \tag{7-77}$$

ここで、$X_{\min} = \min\{x_{\text{int8}}(n) \mid n = 0,\ldots,N-1\}$ と $X_{\max} = \max\{x_{\text{int8}}(n) \mid n = 0,\ldots,N-1\}$ である。

R' が、２のべき乗の場合、l_n 値は、ビット列に直接書き込まれる。一方、多項式値 V_k は、計算され、その後にビット列に書き込まれる。

M 値は、全てのフレーム長に対し、５が設定され、M の大きさのサンプルブロックの正数値は、フレームに合致し、フレームの終わりで、不完全なブロックが残らない。各多項式値 V_k は、以下のように計算される。

$$V_k = l_{5k} + l_{5k+1}(R') + l_{5k+2}(R')^2 + l_{5k+3}(R')^3 + l_{5k+4}(R')^4 \qquad k = 0,\ldots,\frac{N}{5}-1 \tag{7-78}$$

40、80、160、240、320 のフレーム長に対し、フレーム毎の５サンプルブロック数は、それぞれ、８、16、32、48、64 である。データ領域 R' に応じ、各多項式値 V_k 及び各フレーム長（接頭符号を含まず）対する、ビット列に書き込まれるビット数を Table 7-27／JT-G.711.0 に示す。

Table 7-27／JT-G.711.0 – Number of output bits

(ITU-T G. 711.0)

R'	**Number of output bits for each V_k**	**Number of output bits per frame**				
		40	**80**	**160**	**240**	**320**
2	–	40	80	160	240	320
3	8	64	128	256	384	512
4	–	80	160	320	480	640
5	12	96	–	–	–	–
6	13	104	–	–	–	–

ここで、データ領域 *R'=2* と *R'=4* に対して、そのビット列多項式値 V_k を計算する必要が無い。これは、端数

ビット手法が、各データ値を１ビットと２ビットでそれぞれ独立に符号化するのと等価であるからである。また、*R'=5* と *R'=6* に対してのみ、40 サンプルフレーム長が適用される。

7.11.4　データ領域削減に対する接頭符号と端数ビット符号化

種々のフレーム長に対するデータ領域削減とサンプル毎端数ビット符号化の方法は、30 種の適用例となる。これらは、接頭符号にて記述される。Table 7-28／JT-G.711.0 にこれら適用例と、対応する接頭符号を示す。

Table 7-28／JT-G.711.0 – Prefix codes for fractional-bit coding

(ITU-T G. 711.0)

R	R'	**Distinctive values within frame**	**Prefix code for each frame length**				
			40	**80**	**160**	**240**	**320**
2	2	{0, 1}	2	14	20	24	28
3	2	{−2, 0}	4	16	22	26	30
3	3	{−2, −1, 0}	6	18	–	–	–
3	3	{−1, 0, 1}	7	19	–	–	–
4	3	{−2, 0, 1}	5	17	23	27	31
4	4	{−2, −1, 0, 1}	3	15	21	25	29
5	4	{−3, −2, 0, 1}	8	–	–	–	–
5	4	{−2, 0, 1, 2}	10	–	–	–	–
5	5	{−2, −1, 0, 1, 2}	9	–	–	–	–
6	5	{−3, −2, 0, 1, 2}	12	–	–	–	–
6	6	{−3, −2, −1, 0, 1, 2}	11	–	–	–	–
7	6	{−4, −3, −2, 0, 1, 2}	13	–	–	–	–

7.12　最小最大レベル符号化ツール（Min-Max level coding tool）

本ツールは、40 サンプルフレームに対してのみ適用される。

7.12.1　最小最大符号化処理の概要

最小最大レベル符号化は、単純に入力されたフレームの最大値最小値で表される区間を２進表現するのに必要な最小のビット数を求め、各サンプルの値をその最小限のビットを用いて符号化する

量子化レベル数 *R* を計数算するのに必要な最小のビット数を *B* とする。

$$B = \lceil \log_2(R) \rceil \tag{7-79}$$

ここで、*R* は（7-1）式より得られる。データ領域の最大値 $X_{\max}$ 最小値 $X_{\min}$ は、以下のように定義される。

$$\begin{cases} X_{\max} = \max\{x_{\text{int8}}(n) \mid n = 0,...,39\} \\ X_{\min} = \min\{x_{\text{int8}}(n) \mid n = 0,...,39\} \end{cases} \tag{7-80}$$

最小最大レベル符号化器は、各サンプル値 $x_{\text{int8}}(n)$ とアンカー値 X_{ANCHOR} との差分を２進表現し、*B* ビットで正確に表現することで符号化する。このアンカー位置は、１あるいは２オーバーヘッドオクテットの補助情報として最初に指定される。サンプル $x_{\text{int8}}(n)$は、以下のように圧縮された最小最大レベルフレームにおいて $z(n)$ として表せる。

$$z(n) = xint8(n) - XANCHOR \quad n = 0,...,39 \tag{7-81}$$

このようにして、全てのサンプルは、最小最大レベル圧縮ペイロードの候補として、$N \times B$ ビットで正確に符号化される。

本ツールの起動条件は、１次の量子化偏自己相関係数 $\hat{k}_0$（7.10 節を参照）とデータ領域 R に、以下のように依存する。

if $(2^{15} \cdot \hat{k}_0 > 22000)$ {

if ($R \le 4$) check if *Min-Max level coding* is better

}

else {

if ($R \le 4$) check if *Min-Max level coding* is better

else if ($(3/4)(2^B) < R < 2^B$) check if *Min-Max level coding* is better

}

最小最大レベル符号化フレームのオクテット数は、必要となる付加オクテット数に依存し、(($N \times B$/8) + 1)と(($N \times B$/8) + 2)のどちらかである。($N \times B$ + 2)が、あらかじめ計算した“写像領域線形予測符号化フレーム”のオクテット数よりも大きい場合、最小最大レベルツールは、適用されない。

7.12.2 最小最大レベルツールのフォーマット

*最初の*オーバーヘッドオクテットのフォーマットは、LSB が A_1（*uint8* フォーマット）である{L_3,L_2,L_1,A_5,A_4,A_3,A_2,A_1}フォーマットにおいて、５つのＡビットの前の３つの“L ビット”である。Ｌビットは、符号化において必要とされるサンプル毎のビット数を決定し、Ａビットは、アンカー位置あるいは使用されるアンカー種類を決定する。L ビットは、符号化器においてサンプル毎に必要なビット数に対応する１から７の値を表す３ビット、すなわち（MSB 始まりの *uint8* フォーマット）: L={L_3,L_2,L_1}において、決定される。

５つの“A ビット”は、32（2^5）アンカー位置の指定が可能であるが、本目的のためには 30 しか使用されない。その 30 アンカー位置を Table 7-29／JT-G.711.0 に示す。

Table 7-29／JT-G.711.0 – A bit definition for Min-Max level encoding

(ITU-T G. 711.0)

Anchor location (in x_{int8} format)	**Anchor codepoint {A_5,A_4,A_3,A_2,A_1}**
1	00000
0	00001
−1	00010
−2	00011
−3	00100
−4	00101
−5	00110
−6	00111
−7	01000
−9	01001
−11	01010
−13	01011
−15	01100
−17	01101
−20	01110
−23	01111

Table 7-29／JT-G.711.0 – A bit definition for Min-Max level encoding

(ITU-T G. 711.0)

Anchor location (in x_{int8} format)	**Anchor codepoint {A_5,A_4,A_3,A_2,A_1}**
−26	10000
−29	10001
−32	10010
−36	10011
−40	10100
−44	10101
−48	10110
−53	10111
−58	11000
−63	11001
−68	11010
−74	11011
−80	11100
−87	11101
Unused	11110
Explicit Anchor in additional (second) overhead octet	11111

上記の一つのコードポイントは、G．７１１AD 変換の内在符号化バイアス偏差を補正するためのゼロ x_{int8} レベルである。また、$x_{int8}(n) = 0$ 小は、G．７１１符号化における 0^+のアナログレベルを表す。

その他のアンカーコードポイントは、“アナログゼロ”から負の最大量子化レベルまでの間を、概ね対数的に分割するよう設計されている。

ここで、いくつかの *1 番目*のオーバーヘッドオクテット組合せが許容されていない（すなわち、A bits = {11110}の場合）。また、その他の組合せは、使用されない（例えば、B の値を持つ L ビットの組合せは、本ツールでは使われない）。

平均値がゼロであるオーディオ信号として圧縮されたG．７１１入力フレームの通常ケースに対しては、オーバーヘッドは、上記で定義されたアンカーポイントに対する“アンカー位置”の指定する１オクテットとなる。しかしながら、それらのアンカー値のいずれを選択しても、X_{max} とアンカー値の差分の値の２進表現に B ビット以上必要な場合、この符号化手法は、上記で定義されたアンカー値を使用しない。その代わりに、X_{min} の値そのものを、アンカー値として使用する。このアンカー値は明示的アンカー値と呼ばれ、*2 番目*のオーバーヘッドオクテットに明示される。明示的アンカー値は、負の最大レベル（$x_{int8} = -128$）からの（符号無し）差分として、符号化される。このカウント値は、“E ビット”として明示される。E ビットは、E_1 から E_8 までにラベル付けられる。E_1 が LSB である（*uint8* 形式）。明示的アンカー値オクテットのフォーマットは、{E_8,E_7,E_6,E_5,E_4,E_3,E_2,E_1}である。明示的アンカーが必要な場合、A ビットの符号値{11111} = 31 が適用される。

オーバーヘッドオクテットの後に符号化されたフレームのオクテットは、Z ビットを含んでいる。フレーム中の全てのサンプルに対して、適切な Z ビット数（すなわち B ビット）で連結され、各サンプルが符号化される。フレーム中の最初のサンプルをサンプル０、最後をサンプル（N-1）としてラベル付けする。例えば、サンプル毎３ビットのオーディオフレーム（すなわち $B = 3$）は、以下のように符号化される。

$$\{Z_{3,0}\, Z_{2,0}\, Z_{1,0}\, Z_{3,1}\, Z_{2,1}\, Z_{1,1}\, Z_{3,2}\, Z_{2,2}\, Z_{1,2}\, \ldots.\, Z_{3,(N-1)}\, Z_{2,(N-1)}\, Z_{1,(N-1)}\}$$

これら Z ビットをオクテットにパッキングする。フレーム長 N が 40 なので、$Z_{1,(N-1)}$は、最終オクテットの最終ビットである。

7.12.3　最小最大レベル符号化アルゴリズムの詳細

フォこのアルゴリズムは、以下４ステップから構成されている。

１）　Ｇ．７１１入力フレームに対して、Ｇ．７１１A 則あるいは μ 則サンプルを内部表現形式 x_{int8} のＧ．７１１ *int8* 変換サンプルへと写像する。本標準におけるこのステップは、最小最大符号化ツール処理の前に実行される。

２）　入力フレームに対して必要なサンプル毎ビット数を計算する。必要なビット数は B（式 7-79）である。B の計算は、*最小最大レベル符号化ツール*処理の前に実行される。このビット数を表す値に M ビットを設定する。

３）　アンカー符号値 X_{ANCHOR} を探索する。X_{min} が、最小アンカー符号値（-87）より小さい場合、ステップ 3A に進む。X_{min} が、アンカー符号値位置に等しい場合、ステップ 3B に進む。X_{min} が、アンカー符号値位置と等しくなく、取り得る最小アンカー符号値値よりも大きい場合、ステップ 3C に進む。

ステップ 3A：陽のアンカーは、X_{min} が使用されなけらばならず、$X_{ANCHOR} = X_{min}$ と設定する。陽のアンカーが要求されたことを示し、7.12.2 節で示される陽のアンカーを符号化するために、A ビットを設定する。

ステップ 3B：$X_{ANCHOR} = X_{min}$ と設定し、このアンカー位置を示す A ビットを設定する。

ステップ 3C：X_{ANCHOR} を次に負に大きいアンカー符号値位置に設定する。$B' = \lceil \log_2 X'_{max} \rceil$ とする。ここで、X'_{max} は、X_{max} から X_{ANCHOR} までを含む隣接するコードワード値である。$B' = B$ の場合、X_{ANCHOR} は、このフレームに対するアンカー位置として使用される。A ビットを適切に設定し、ステップ４に進む。$B' > B$ の場合は、陽のアンカー設定が必要であり、ステップ 3A に進む。

４）　7.12.2 節で示されるように、各サンプル n に対して、式（7-81）により z(n)を圧縮フレームのサンプル部分にパッキングする。

7.12.4　最小最大レベル符号化ツールのビットパッキングフォーマット

Table 7-30／JT-G.711.0 に、最小最大レベル符号化ツールのビットパッキングフォーマットを示す。

Table 7-30 – Bit packing format of the Min-Max level coding tool

(ITU-T G. 711.0)

Frame length (*N*)	**40**	**80, 160, 240, 320**
Frame length prefix	2 bits	N/A
Tool prefix	6 bits	N/A
First overhead octet (always present)	8 bits $\{L_3,L_2,L_1,A_5,A_4,A_3,A_2,A_1\}$	
An explicit anchor octet (only if all A bits are equal to 1)	8 bits $\{E_8,E_7,E_6,E_5,E_4,E_3,E_2,E_1\}$	
The Z bits packed into octets	$N \times B$ bits $\{Z_{B,0}$ …… $Z_{1,(N-1)}\}$	

１番目のオーバーヘッドオクテットは、サンプル毎の必要なビット数と、２番目のオーバーヘッドオクテットが存在するかどうかを規定していることに注意。最初オーバーヘッドオクテットと圧縮されたフレームのサイズとを分析することによって、Z ビットオクテットが完全に定義される。このようにして、サンプル毎ビット数と Z オクテット数から、圧縮前の元の入力フレーム中のサンプル数が分かる。この手法により、最小最大フレームは、自己記述的である。

7.13　直接線形予測符号化ツール（Direct LP coding tool）

最小最大レベルツールと同様に、本ツールは、40 サンプルフレームに対してのみ適用される。本ツールは、線形予測符号化を実行するが、（線形 $x_{PCM}(n)$領域ではなく）圧縮された $x_{int8}(n)$領域から直接予測する。そして、ここまでに記述した全ての符号化ツールが機能しなかった場合に、有効である。すなわち、

$$\begin{cases}\text{Direct LP coding mode is temporally on , } \textit{if coded\_bytes} > N-1 \\ \text{Direct LP coding mode is off, otherwise}\end{cases} \tag{7-82}$$

直接線形予測符号化モードが適用されるとき、本ツールは、以下のステップを実行する。

1） *int8* 領域において平均値０の信号を得るため、式（6-9）から（6-12）を用いて、A 則／μ 則を *int8* へ変換する。

2） 符号情報 $z_{sign}(n)$から、絶対値信号 $z(n)$を計算する。

$$z(n) = \left| x_{\text{int 8}}(n) \right|$$
$$z_{sign}(n) = \begin{cases} 0 & x_{\text{int8}}(n) \ge 0 \\ 1 & x_{\text{int8}}(n) < 0 \end{cases} \tag{7-83}$$

3） 線形予測残差を得るため、固定係数 LPC 予測を適用する。

$$\begin{cases} e(n) = z(n) \quad n < 4 \\ e(n) = 0.25(z(n-1) + z(n-2) + z(n-3) + z(n-4)) - z(n) \quad 4 \le n < N \end{cases} \tag{7-84}$$

4） 残差 $e(n)$および符号情報 $z_{sign}(n)$を符号化する。最初の予測していない４サンプルに対しては、$e(n)$と $z_{sign}(n)$はサンプル毎８ビットのビット列中に一緒に符号化される。その他のサンプルに対しては、$e(n)$は 6.9.2 節のライス符号化を用いて符号化される。この時のライスパラメータは５である。また $z_{sign}(n)$は、サンプル毎１ビットで直接符号化される。直接線形予測符号化ツールの接頭符号は、ビット列に追加される。

直接線形予測符号化で得られるビット列の大きさが、元のＧ．７１１ A 則／μ 則ビット列よりも大きくない場合、残差と符号情報は直接線形予測符号化ツール接頭符号としてビット列へ挿入する。

Table 7-31／JT-G.711.0 に、直接線形予測符号化ツールのビットパッキングフォーマットを示す。

Table 7-31 – Bit packing format of the direct LP coding tool

(ITU-T G. 711.0)

Frame length (*N*)	**40**	**80, 160, 240, 320**
Frame length prefix	2 bits	N/A
Tool prefix	3 bits	
Codes of $e(n)$ and $z_{sign}(n)$ for the first 4 non-prediction samples	4×8 bits	
Rice code of $e(n)$ and 1-bit of $z_{sign}(n)$ for the other samples	Variable	
Octet boundary alignment	0 to 7 bits of '0's	

8　復号器の機能記述

8.1　接頭符号からのフレーム長と復号ツールの取得

フレーム長 *N* は、符号化ビット列の１番目のオクテット中最初の数ビットから Table 7-1／JT-G.711.0 を用いて、復号される。符号化ビット列中の１番目オクテットが 0x00 の場合、復号フレーム長はゼロ（そのフレームにおけるサンプル数が０）である。0x00 符号は、復号処理しないこと意味し、パディングオクテットとして使用される。

8.2 非圧縮復号ツール（Uncompressed decoding tool）

接頭符号が、非圧縮符号化ツールの適用を示す場合、復号器は接頭符号に続くオクテットから、そのまま出力サンプルを再生する。

8.3 定数復号ツール（Constant value decoding tool）

ツール接頭符号が、定数符号化ツールの適用を示す場合、復号器はそのフレームの全てのサンプルを、定数として再生する。Table 7-1、7-2、7-3／JT-G.711.0 に、接頭符号とビットパッキングフォーマットを示す。

8.4 正負零値ライス復号ツール（PM zero Rice decoding tool）

接頭符号が、正負（*PM:Plus Minus*）零値ライス符号化ツールの適用を示す場合、復号器は、各値のランレングスライス復号結果から、正零値あるいは負零値のどちらかを生成する。

1） 本ツールの接頭符号が、正負零値ライス符号（Minus <= Plus）を指定している場合は、0_m を 0^+に、0_l を 0^-に設定し、正負零値ライス符号（Minus > Plus）を指定している場合は、0_m を 0^-に、0_l を 0^+に設定する。

2） 0_m と 0_l のランレングス復号のためのライスパラメータ S をハフマン復号する。

3） 数列 0_m のライス復号（6.9.2 節を参照）し、0_m と 0_l を出力バッファに格納する。

ステップ２において、ライスパラメータ S は、入力フレーム長に対する Table 7-6／JT-G.711.0 のハフマン符号化テーブルを用いて復号される。ツール接頭符号に続く２ビットが零の場合、パルスモード復号は、8.5 章で記述するように処理される。

ステップ３において、N サンプルが再生されればライス復号を停止する。0_m 列の最終ランレングス値を復号するために、仮想的な終止サンプル 0_l は出力バッファに格納しない。

8.5 バイナリ（２進）復号ツール（Binary decoding tool）

接頭符号が、２進符号化モードの適用を示す場合、復号器は、２進符号化列を変換し、正零ないしは負零を再生する。ここで、0 は正零 0^+を、1 は負零 0^-を表す。

8.6 パルスモード復号ツール（Pulse mode decoding tool）

接頭符号が、パルスモード符号化（Minus < Plus）あるいはパルスモード符号化（Minus > Plus）の適用を示す場合、パルスモード復号が適用される。

1） 本ツールの接頭符号が、パルスモード符号化（Minus < Plus）を指定している場合は、0_m を 0^+に、0_l を 0^-に設定し、パルスモード符号化（Minus > Plus）を指定している場合は、0_m を 0^-に、0_l を 0^+に設定する。

2） 0_m と 0_l のランレングス復号のためのライスパラメータ S をハフマン復号する。

3） 入力フレーム長に対するパルス位置インデックスを復号する。

4） 差分符号化による 0_m あるいは 0_l を識別するために、１ビットフラグ f_{diff_PM} を復号する。

5） パルス値と 0_m あるいは 0_l 値との間の差分 d_{pulse_PM} を復号する。

6） 下式を用い、パルス値を算出する。

$$pulse\_value = \begin{cases} d_{pulse\_PM} - 1 + 0_m & f_{diff\_PM} = 0 \\ d_{pulse\_PM} - 1 + 0_l & f_{diff\_PM} = 1 \end{cases}$$

7） 数列 0_m のライス復号（6.9.2 節を参照）し、0_m と 0_l を出力バッファに格納する。

8） パルス位置を正しいパルス位置に置き換える。

ステップ５において、ライスパラメータ S は、入力フレーム長に対する Table 7-6／JT-G.711.0 のハフマン符号化テーブルを用いて復号される。

ステップ１０において、N サンプルが再生されればライス復号を停止する。0_m 列の最終ランレングス値を復号するために、仮想的な終止サンプル 0_l は出力バッファに格納しない。

8.7　値位置復号ツール（Value-location decoding tool）

復号された接頭符号が、値位置符号化ツールを示す場合、値位置復号が実行される。フレーム内のサンプル数は、復号された接頭符号から決定される。

8.7.1　フレームパラメータ復号と値の代入

復号器は、入力ビット列から、Table 7-13／JT-G711.0 に示されるようなデータ範囲を指定する２ビット列を読み出す。復号された場合に基づいて、$X_{\min,}\ n = 0,...,N-1$ と R には、Table 7-13／JT-G711.0 に記載された値が、設定される。復号器は、Table 7-14／JT-G711.0 に示されるように、復号される値 v_k の次数を決定する１ビットを読み出す。

8.7.2　連続な値位置の復号

発生数 N_k と、符号化された値 v_k（7.9.3、7.9.4 節を参照）の位置 $l_k(m)$が、$k = 1,...,R-1$ について、連続して復号される。各 k について、v_k の発生数 N_k が、復号された出力フレーム $x_{\text{int8}}(n)$に設定される。適切な位置に値を設定する為、各位置 $l_k(m)$を、対応する $x_{\text{int8}}(n)$の位置に、写像する変換を行う必要がある。また、値が設定されない $x_{\text{int8}}(n)$の要素インデックスとして、$a_k(j)$ を定義する。$a_k(j)$ の要素数は、D_k として定義される。K が増加して、出力フレーム $x_{\text{int8}}(n)$に、より多くの値を設定されると、$a_k(j)$ の要素数は減少する。まず、出力フレーム $x_{\text{int8}}(n)$における値が設定される以前に、7.9.3 節に示すような、同一の長さ条件をもった値が設定される。

$$a_1(j) = j \qquad j = 0,\ldots,N-1 \tag{8-1}$$

$$\begin{aligned} D_1 &= N \\ D_k &= N - \sum_{i=1}^{k-1} N_i \qquad k = 2,\ldots,R \end{aligned} \tag{8-2}$$

N_k 位置において値 v_k を設定した後、以下に示すように、$x_{\text{int8}}(n)$において、位置インデックスは、フレーム要素のインデックスベクトルから除外される。

$$\begin{aligned} &set\ m = 0,\ j = 0 \\ &for\ each\ n = 0,\ldots,D_k - 1 \\ &\qquad if\ n \neq l_k(m) \\ &\qquad then\ a_{k+1}(j) = a_k(n)\ and\ j = j+1 \\ &\qquad else\ if\ m < N_k - 1\ then\ m = m+1 \end{aligned} \tag{8-3}$$

8.7.3　復号方法

Table 7-15／JT-G711.0 に示されるように、各 k について、v_k に使用された符号化タイプを示す３ビットが読み出される。発生数 N_k と位置 $l_k(m)$は、使用される符号化タイプに基づいて復号される。（7.9.4 節を参照）

8.7.3.1　値の発生が無い場合

符号化タイプが、特定の値 v_k がフレーム内において発生しないことを示す場合、復号される位置はない為、復号器は、次の値 k について処理を進める。

8.7.3.2　ライス復号

符号化タイプが、ライスパラメータ(Table 7-15／JT-G711.0 に示されるように)を指定した場合、 6.9.2 節に記載されるようなライス復号が実行される。復号により、v_k 間のセグメント長を決定する。復号されたセグメント長は、位置 $l_k(m)$と、発生数 N_k を決定する為に使用される。

8.7.3.3 バイナリ（２進）復号

符号化タイプが、バイナリ（２進）符号化が使用されたことを示す場合、$i = 0,...,D_k - 1$ のビット数 D_k 個の $b(i)$ が、ビット列から読み出される。v_k 発生の位置は、バイナリ（２進）の１の位置によって定義される。

$$\begin{aligned} &set\ m = 0 \\ &for\ each\ i = 0,...,D_k - 1 \\ &\quad if\ b(i) = 1 \\ &\quad then\ l_k(m) = i\ and\ m = m + 1 \\ &end \\ &N_k = m \end{aligned} \tag{8-4}$$

ここで、$b(i)$列の１の個数は、フレーム(N_k)の v_k 発生数と等しい。

8.7.3.4 明示的な復号

明示的な位置符号化が使用されていたことを、符号化タイプが示す場合、フレーム N_k 内の v_k の明示的な位置の個数を決める為、ビット列から２ビットを読み出す。N_k 個のバイナリ符号化位置 $l_k(m)$は、各位置に対して $\lceil \log_2(D_k) \rceil$ ビットを読み出すことのよって、明示的に取得される。

8.7.4 出力フレームベクトルの生成

一度、値位置 $l_k(m)$が決められると、以下に示すように、出力フレームベクトル $x_{\text{int8}}(n)$ 中の N_k 個の v_k の位置を設定するために使用される。

$$x_{int8}\left(a_k\left(l_k(m)\right)\right) = v_k \qquad m = 0,...,N_k - 1 \tag{8-5}$$

ここで、全ての M 個の v_k 値が復号されて $x_{\text{int8}}(n)$に設定された後の、$a_{M+1}(j)$ 列の残りのインデックスは、$v_0 = 0$ の参照値位置を示す。$x_{\text{int8}}(n)$ のこれらの位置に、以下に示すような設定が行われる。

$$x_{int8}\left(a_{M+1}(j)\right) = 0 \qquad j = 0,...,D_{M+1} - 1 \tag{8-6}$$

8.7.5 *int8* のA則／μ 則への変換

オリジナル対数 PCM フレームを再構築する為に、テーブル参照を使用して、各サンプルについて、変換が実行される。

$$I_\mu(n) = f_{int8\to\mu}(x_{int8}(n)) \tag{8-7}$$

$$I_A(n) = f_{int8\to A}(x_{int8}(n)) \tag{8-8}$$

8.8 写像領域線形予測復号ツール（Mapped domain LP decoding tool）

8.8.1 写像領域線形予測復号ツールの概要

接頭符号が、写像領域線形予測符号化ツール(長期予測オン) または、写像領域線形予測符号ツール(長期予測オフ)を示す場合、復号器は、写像領域線形予測復号処理を行う。Figure 8-1／JT-G711.0 に、写像領域線形予測復号器のブロック図を示す。目的の符号則(A 則／μ 則)が、帯域外情報として、復号器に示される。圧縮ビット列には、偏自己相関係数、サブフレーム分割、及び長期予測に関連するパラメータ、分離パラメータと、予測残差信号の可変長符号が含まれる。復号された残差信号は、Ｇ．７１１符号列を生成する為、長期予測フィルタと短期線形予測フィルタ、写像処理に供給される。パックされたビット列は、以下の手順で復号される。

1) 接頭符号ツールから長期予測モードを取得する。
2) 目的の符号規則が μ 則、かつ、フレーム長が $N \geq 80$ の場合、正負零値写像フラグを取得する。
3) 量子化線形予測次数 $\hat{P}$ が、予測次数の次数インデックスから決定される。
4) 偏自己相関係数を復号、再構成する。
5) 残差信号の分割パラメータを復号する。
6) 残差信号を復号する。

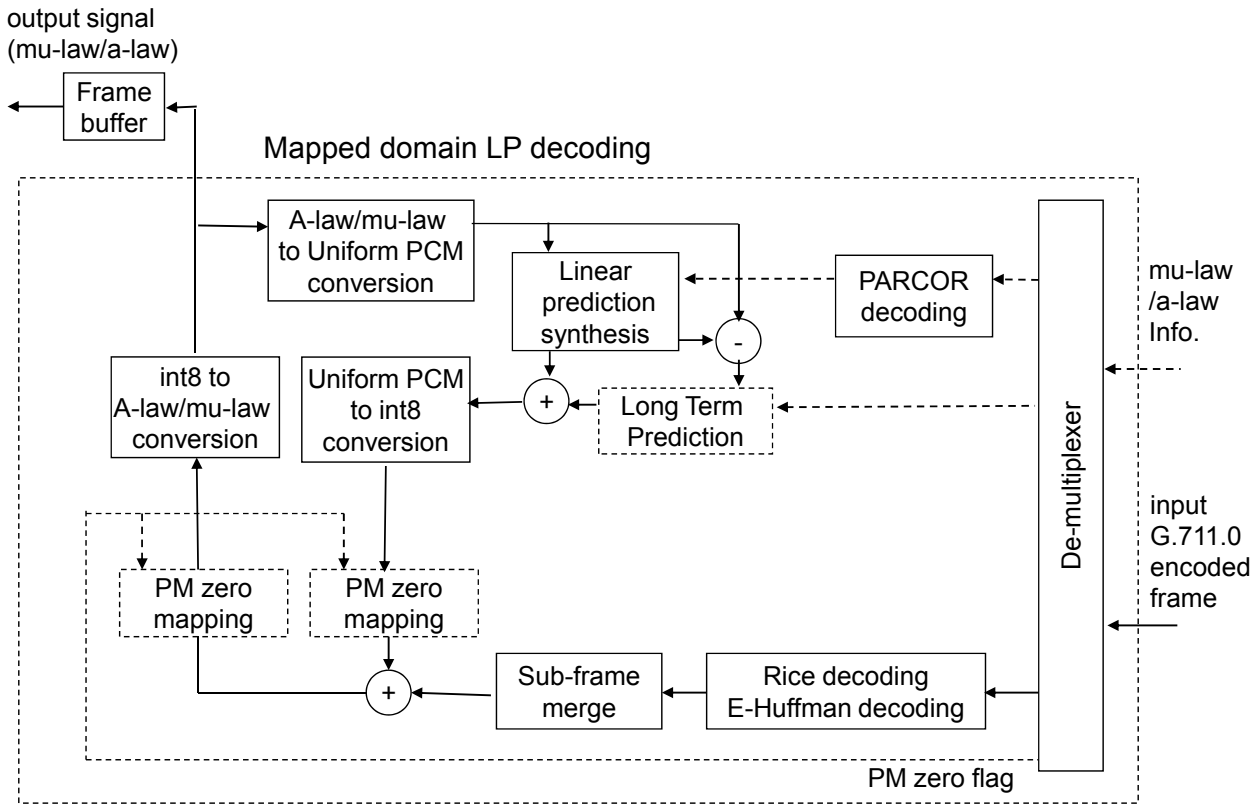

Figure 8-1/JT-G711.0 – Block diagram of the Mapped domain LP decoder

(ITU-T G.711.0)

8.8.2 線形予測パラメータ復号

8.8.2.1 偏自己相関係数の取得

予測次数と量子化偏自己相関係数のインデックスを取得することは、符号化器の逆の手順となる。量子化偏自己相関係数値の取得は、7.10.3.6.節に記載されている。予測次数 $\hat{P}$ は、Table 7-19／JT-G711.0 に記載されるように、復号された間引き予測次数のインデックスによって示される。40 と 80 サンプルの場合について、偏自己相関係数は、以下のように再構成される。

1) ライス符号を復号することにより、Y_j, (j=0,…, $\hat{P}-1$) を取得する。

2) Table 9-1／JT-G711.0 の(47)、(49)、(51)、(53)、(55)、(57)、(59)において、Y_j, (j=0,…, $\hat{P}-1$)から写像テーブルにより、インデックス X_j, (j=0,…, $\hat{P}-1$)を取得する。

3) その結果、量子化偏自己相関値 $\hat{k}_j$, (j=0,…, $\hat{P}-1$) は、Table 9-1／JT-G711.0 の(29)、(30)、(31)に示される。

160、240、320 サンプルフレームの場合、偏自己相関係数のインデックスは、Table 9-1／JT-G711.0 の(16)、(17)、(18)、(19)、(20)、(21)に記載されるハフマン符号を復号することにより取得される。これらのインデックスを利用して、量子化偏自己相関値 $\hat{k}_j$ (j=0,…, $\hat{P}-1$)は、符号化器の場合と同じ方法で算出される。

8.8.3 残差信号の復号

8.8.3.1 分離パラメータ

160、240、320 サンプルのフレーム長において、残差信号は、全フレーム、または、２、３、４サブフレームのいずれかによりそれぞれ符号化される。サブフレーム分割と分離パラメータへの伝達は、符号化部の Table 7-21／JT-G711.0 に示される。各フレームの初期分離パラメータ S_0、または、グローバル分離パラメータ S_g は、Table 8-1／JT-G711.0 に示すように、ライスパラメータ１、符号なしライス符号化列から復号される。ライス符

号と分離パラメータ間の写像は、接頭符号により通知された長期予測の条件に依存する。

Table 8-1/JT-G711.0 – Rice code for S_0 or S_g

(unsigned Rice code with separation parameter 1)

(ITU-T G.711.0)

	LTP condition	Value of S_0 or S_g							
		0	1	2	3	4	5	6	7
Code	No	11	011	10	010	0010	0011	00010	00011
	Yes	0011	011	11	10	010	0010	00010	00011

復号されたフラグが、サブフレーム分割を示す場合、分離パラメータ S_i, $(i=1,...,N'_{sfr}-1)$ は、以下のように復号される。

$$S_i = S_{i-1} + \Delta S_{i-1} \qquad (i=1,...,N'_{sfr}-1) \tag{8-9}$$

ΔS_i は、ライスパラメータ０をもつライス符号列から復号することができる。ライス符号に相当する ΔS_i の値は、Table 7-23／JT-G711.0 に示される。

復号された分離パラメータは、フレームの第一サンプル $min\{2,\hat{P}\}$ を除き、残差信号を復号する為に使用される。フレームの最初のサンプル $min\{2,\hat{P}\}$ を復号する為、残差信号分離パラメータは、最初の偏自己相関係数 $\hat{k}_0$ と長期予測条件に関連する S_0、または、S_g から変更される。Table 7-24／JT-G711.0 に関係を記載する。

8.8.3.2 ライス復号

40 サンプルフレームの場合、各残差信号 $r(n), n=0,...,N-1$ は、8.8.3.1 節において取得される分離パラメータ $S=S_g$ とともに、6.9.2 節に記載されるライス符号から復号される。

S=0 の場合、$r(n)$は、次に示すように復号される。

$$r(n)=\begin{cases} k(n)/2 & if\ \ k(n)\otimes 1=0 \\ -(k(n)+1)/2 & otherwise \end{cases} \tag{8-10}$$

ここで、商 $k(n)$は、１進符号の連続に対するビット”０”と、終止に対するビット”１”を探索して復号される。

S=1 の場合、$r(n)$は、次に示すように復号される。

$$r(n)=\begin{cases} k(n) & if\ \ j(n)=1 \\ -k(n)-1 & otherwise \end{cases} \tag{8-11}$$

ここで、k(n)は、１進符号の連続に対するビット”０”と、終止に対するビット”１”を探索することにより、再復号される。j(n)は、k(n)の直後に続く１ビットである。

S>1 の場合、r(n)は、次に示すように復号される。

$$r(n)=\begin{cases} 2^{(S-1)}k(n)+\left(j(n)\otimes\left(2^{(S-1)}-1\right)\right) & if\ \ (j(n)\otimes 2^{(S-1)})\neq 0 \\ -\left\{2^{(S-1)}k(n)+j(n)\right\}-1 & otherwise \end{cases} \tag{8-12}$$

ここで、$j(n)$は、$k(n)$に後続する符号なしバイナリ（２進）表現で S ビットである。

8.8.3.3 エスケイプハフマン復号

40 サンプル以上のフレーム長の場合、各残差信号 $r(n), n=0,...,N-1$ は、8.8.3.1 節において取得される、分離パラメータ S (=S_g or =S_i, $(i=0,...,N'_{sfr}-1)$)をもつエスケイプハフマン符号から復号される。エスケイプハフマンテーブルとテーブルインデックス符号は、Table 7-25／JT-G711.0 に記載される。エスケイプハフマンテーブルインデックスは、各サブフレームについて復号され、４つのテーブルから指定されるエスケイプハフマンテーブルが、サブフレームにおいて、残差信号を復号する為に使用される。

S=0 の場合、式(8-10)を使用して、$r(n)$が再構成される。ここで、S は S_g または S_i, $(i=0,...,N'_{sfr}-1)$ である。

また、*k*(*n*)は、エスケイプハフマン符号を復号することにより取得される。

S=1 の場合、式(8-11)を使用して、*r*(*n*)が再構成される。ここで、S は S_g または S_i, $(i=0,...,N'_{sfr}-1)$ である。また、*k*(*n*)は、エスケイプハフマン符号を復号することにより再取得される。*j(n)*は、符号なしバイナリ（２進）表現において、1 ビットであり、*k*(*n*)に後続する。

S>1 の場合、式(8-12)を使用して、*r*(*n*)が再構成される。ここで、S は S_g または S_i, $(i=0,...,N'_{sfr}-1)$ である。また、*j(n)*は、符号なしバイナリ（２進）表現において、1 ビットであり、*k*(*n*)に後続する。

エスケイプハフマン符号の復号された値が、*maxCode* 値と等しい場合、符号は、エスケイプ符号となり、エスケイプ符号後に続く拡張符号が、復号される。

S=0 の場合、拡張符号は、ユナリ（１進）(Unary(*k*(*n*) – *maxCode*)）である。

S>0 の場合、7.10.5.6 節に記載されるように、拡張符号は、ライスパラメータ１をもったライス符号（Rice(l,*k*(*n*) – *maxCode*)）である。

8.8.4 線形予測

8.8.4.1 予測係数

予測係数は、以下に示す処理によって、量子化偏自己相関係数から取得される。

1) 繰り返し数 *i*、および $a_0^{[0]}$ を次のように設定する。 $i=1,\ a_0^{[0]}=1.0$

2) $a_i^{[i]}=k_{i-1}$ と設定する。

3) $a_j^{[i]}=a_j^{[i-1]}+k_{i-1}\cdot a_{i-j}^{[i-1]} \quad j=1,...,i-1$ と算出する。

4) i を１ずつ増加し、i が量子化予測次数 $\hat{P}$ に到達するまで、手順２に戻る。

8.8.4.2 線形予測合成

線形予測合成処理は、以下の２つの手順から構成される。

1) 線形領域における予測値の生成

2) オリジナル対数 PCM 列の生成

最初の手順は、7.10.3.8 節において記載した符号化処理と同様である。予測値は、式(7-33)、式(7-34) 、式(7-35)を使用して、線形領域において再帰的に算出される。２番目の手順は、予測サンプルを *int8* 領域に変換することで構成される。

$$\hat{x}_{int8}(n)=f_{PCM\to int8}(\hat{x}_{PCM}(n)) \tag{8-13}$$

目的の符号則が、A 則、または、μ 則であり、$N=40$ の場合、$x_{\mathrm{int8}}(n)$は、以下のようにして、上記の予測値、および復号した残差値 *r*(*n*)を使用して取得される。

$$x_{int8}(n)=r(n)+\hat{x}_{int8}(n) \tag{8-14}$$

目的の符号則が、μ 則であり、$N>40$ の場合、$x_{\mathrm{int8}}(n)$は、以下のようにして、上記の予測値、および復号した残差値 *r*(*n*)を使用して取得される。

$$x_{int8}(n)=f_{int8(*,*)\to int8}\left(r(n)+f_{int8\to int8(*,*)}\left(\hat{x}_{int8}(n)\right)\right) \tag{8-15}$$

ここで、$f_{\mathrm{int8}\to\mathrm{int8}(*,*)}$ は、7.10.5.2 節の式(7-60)に記載される、正負零値写像であり、$f_{\mathrm{int8}(*,*)\to\mathrm{int8}}$ は、$f_{\mathrm{int8}\to\mathrm{int8}(*,*)}$ の逆写像である。これらの処理は、まさに、符号化器の逆処理である。*int8* 領域で再構成されたサンプルは、以下のように、線形領域に変換される。

$$x_{PCM}(n)=f_{int8\to PCM}(x_{int8}(n)) \tag{8-16}$$

この値は、直ぐに、次のサンプルで、再起的な合成フィルタに使用される。予測次数よりも少ないインデックスをもつサンプルについては、漸進性線形予測が使用される。フレームの第一サンプルについては、予測は適用されない。

$$\hat{x}_{PCM}(0)=0 \tag{8-17}$$

インデックス $n\,(1 \le n < \hat{P})$のサンプルに対しては、予測された値 $\hat{x}_{\mathrm{PCM}}(n)$ は、以下に示すような n 次予測係数 $a_i^{[n]}(1 \le i \le n)$ 設定を使用して、n 次の予測により示される。

$$\hat{x}_{PCM}(n) = \sum_{i=1}^{n} a_i^{[n]} x_{PCM}(n-i) \quad n = 1,\ldots,\hat{P}-1 \tag{8-18}$$

その結果、開始から予測次数 $\hat{P}$ の位置までのサンプル位置に対応して、係数の次数が増加する。サンプルインデックス n が、$\hat{P} \le n < N$ の場合、予測サンプル $\hat{x}_{\mathrm{PCM}}(n)$ は、$\hat{P}$ 次の予測 $a_i^{[\hat{P}]}(1 \le i \le \hat{P})$ によって与えられる。

$$\hat{x}_{PCM}(n) \sum_{i=1}^{\hat{P}} a_i^{[\hat{P}]} x_{PCM}(n-i) \qquad n = \hat{P},\ldots,N-1 \tag{8-19}$$

ここで、最終係数（$\hat{P}$ 次予測の第 $\hat{P}$ 次係数）$a_{\hat{P}}^{[\hat{P}]}$ は、Q15 フォーマットで表現され、残り全ての係数は、Q12 フォーマットで表現される。

8.8.4.3　長期予測合成

長期予測モードが使用されることを、復号した接頭符号のフラグが示す場合、長期予測合成が実行される。サブフレーム数 N_{sfr} は、160、240、320 サンプルフレームに対して、それぞれ、１、２、３となる。最初に i 番目のフレームのピッチ値$\tau_i\,(1 \le i \le N_{sfr})$が復号される。第１サブフレーム$\tau_1$のピッチは、６ビットで復号され、ピッチラグ差$\Delta\tau_i$ 、$2 \le i \le N_{sfr}$ は、フレームの残りのピッチτ_i を再構築する為、ライスパラメータ０を伴って、ライス復号される。τ_0は、τ_1に設定され、7.10.4.1.3 節に記載される適応フレーム分割手順が実行される。サブフレーム長 l_{sfr}は、以下のように設定される。

$$l_{sfr} = \left\lfloor \left(N - \hat{P} - \tau_0\right) / N_{sfr} \right\rfloor \tag{8-20}$$

i 番目のサブフレームの開始位置である下限境界 b_i は 、以下のように算出される。

$$b_i = b_0 + (i-1)\cdot l_{sfr} \qquad 0 < i \le N_{sfr},\ \text{with } b_0 = \hat{P} + \tau_0 \tag{8-21}$$

ここで $\hat{P}$ は、量子化線形予測次数である。

$r(n)$は、まず、エスケイプハフマン復号モジュールによって復号される。PCM 領域 長期予測寄与分は、式(7-51)を使用して取得される。一方、PCM 領域線形予測残差 $r_{\mathrm{PCM}}(n)$が算出される。短期予測サンプル $\hat{x}_{\mathrm{PCM}}(n)$ は、式(8-18)、式(8-19)から取得され、長期予測サンプル $\hat{x}_{\mathrm{LTP}}(n)$ は、式(7-55)により取得される。再構築された *int8* 領域の信号は、以下の式より取得される。

$$x_{int8}(n) = \hat{x}_{int8}(n) + r(n) \tag{8-22}$$

ここで、$\hat{x}_{\mathrm{int8}}(n)$ は、$\hat{x}_{\mathrm{PCM}}(n)$ と $\hat{x}_{\mathrm{LTP}}(n)$ を伴う、式(7-57)、または、式(7-58)から取得される。目的の符号則が、μ 則、かつ、N>40 の場合、 $x_{\mathrm{int8}}(n)$ は、式(8-15)を使用して取得される。最初の $\tau_0 + \hat{P}$ サンプルは、線形予測合成によって復号される。残りのサンプルは、長期予測合成と線形予測合成によって復号される。

8.8.5　*int8* のA則／μ 則への変換

オリジナル対数 PCM フレーム値の再構築を行う為、*int8* から A 則、または、μ 則への変換が、式(6-13) (6-14)、または、式(6-15) (6-16)を用いて、それぞれ、各サンプルに対して実行される。

8.9　端数ビット復号ツール（Fractional bit decoding tool）

復号された接頭符号は、Table 7-28／JT-G711.0 において指定された符号の１つであり、端数ビットの復号が実行される。

復号器は、範囲削減フラグ δ_{-1} の値、データ範囲 R'、最小データ値 $X_{\min}$、Table 7-28／JT-G711.0 において示される接頭符号からのフレームサンプル N を決定する。 $x_{\mathrm{int8}}(n)$ 値の復号方法は、以下に示すようにデータ範囲 R' に依存する。

8.9.1 *R'* = 2 に対する復号

R' = 2 の場合、復号器は、入力ビット列から、*N* ビットの $b(n), n = 0,...,N-1$ を読み出し、$x_{\text{int}8}(n)$ 値を、以下に示すように復号する。

$$x_{int8}(n) = \begin{cases} X_{min} & if\ b(n) = 0 \\ X_{min} + (1 + \delta_{-1}) & if\ b(n) = 1 \end{cases} \tag{8-23}$$

8.9.2 *R'* = 4 に対する復号

R' = 4 の場合、復号器は、入力ビット列から、N 2 ビット整数 $d(n), n = 0,...,N-1$,を読み出し、$x_{\text{int}8}(n)$ 値を、以下のように復号する。

$$x_{int8}(n) = d(n) + X_{min} \tag{8-24}$$

上式は、δ_{-1} 範囲調整によって、以下となる。

$$\begin{aligned} &if\ \ x_{int8}(n) \geq -1 \\ &then\ \ x_{int8}(n) = x_{int8}(n) + \delta_{-1} \end{aligned} \tag{8-25}$$

8.9.3 他のR値に対する復号

R' = 3、 R' = 5、R' = 6、について、7.11.2 節で定義されるように、データブロック多項式の値 $V_k,\ k = 0,...,\frac{N}{5} - 1$ が、復号される。復号器は、V_k を決める為、入力ビット列から、B ビットブロックを読み出し、対応する多項式の係数 $l_{5k},...,l_{5k+4}$ を以下のようにして復号する。

$$\begin{aligned} &w_k(4) = V_k \\ &for\ m = 4,...,1 \\ &\qquad l_{5k+m} = \left\lfloor w_k(m) / (R')^m \right\rfloor \\ &\qquad w_k(m-1) = w_k(m) - l_{5k+m}(R')^m \\ &end \\ &l_{5k} = w_k(0) \end{aligned} \tag{8-26}$$

$x_{\text{int}8}(n)$ 値は、以下に示すように、*l* 係数から復号される。

$$x_{int8}(5k + m) = l_{5k+m} + X_{min} \qquad k = 0,...,\frac{N}{5} - 1 \qquad m = 0,...,4 \tag{8-27}$$

上式は、δ_{-1} の範囲調整によって、以下となる（8.9.2 節と同様）。

$$\begin{aligned} &if\ \ x_{int8}(n) \geq -1 \\ &then\ \ x_{int8}(n) = x_{int8}(n) + \delta_{-1} \end{aligned} \tag{8-28}$$

8.9.4 *int8* のA則／μ 則への変換

オリジナルの対数PCMフレーム値の再構築を行う為、*int8* から A則、または、μ 則への変換が、式(6-13) (6-14)、または、式(6-16)を用いて、それぞれ、各サンプルに対して実行される。

8.10 最小最大レベル復号ツール（Min-Max level decoding tool）

最小最大レベル符号化ビット列の復号は、7.12 節における最小最大レベル符号化処理の逆処理となる。8.10.1 節において、詳細な復号の手順を記載する。

8.10.1 最小最大レベル復号アルゴリズムの詳細

復号の手順を、以下に示す。

先頭のオーバーヘッドオクテットから、符号化において使用されたビット数を決定する、また、第２のオーバーヘッドオクテットを含む場合、明示的なアンカーが必要となる。明示的アンカーが必要な場合、明示的なアンカーは、負の最大量子化レベル（すなわち、$x_{\text{int}8}(n) = -128$）からの、８ビット符号なしバイナリ（２進）のカウントとして符号化される。また、そのカウント値は、Ｅビットとして表現される。明示的なアンカーが必要でない場合、アンカー符号値は、このフレームのアンカーとして使用される。X_{ANCHOR} をこのフレームの適切なアンカー値として設定する。

Mビットを復号することにより、サンプル毎のビット数 *B* を決定する。*z(i)*（*N* サンプルが存在するはずであ

る）を取得する為、符号化フレーム(サンプルにつき B ビットを使用して)において、各サンプルに対して、Z ビットを復号する。アンカー符号値の量子化値(X_{ANCHOR})を各サンプルに加算して、上記の式(7-81)について最小最大レベル符号化器に代表される、Ｇ．７１１の *int8* の変換サンプル n ($x_{\mathrm{int}8}(n)$)を取得する。

8.10.2 *int8* のA則／μ則への変換

オリジナルの対数 PCM フレーム値の再構築を行う為、*int8* から A 則、または、μ則への変換が、式(6-13) (6-14)、または、式(6-16)を用いて、各サンプルに対してそれぞれ実行される。

8.11 直接線形予測復号ツール

復号器が、直接線形予測復号モードになる場合、極性ビットと、最初の予測無し４サンプルの残差 $\hat{e}(n)$が、最初に復号される。他のサンプルの残差は、ライスパラメータ５をもつライス復号器によって復号され、極性ビットも同様に復号される。極性ビットは、$z_{sign}(n)$として復号される。復号された信号 $z(n)$の絶対値には、４次の固定小数点係数の線形予測合成フィルタが適用される。

$$\begin{cases} \hat{e}(n) = z(n) & n < 4 \\ \hat{e}(n) = 0.25(z(n-1) + z(n-2) + z(n-3) + z(n-4)) - z(n) & 4 \le n < N \end{cases} \tag{8-29}$$

絶対値 $z(n)$と極性 $z_{sign}(n)$から、以下のように再構成される。

$$x_{int\,8}(n) = \begin{cases} z(n) & if\ \ z_{sign}(n) = 0 \\ -z(n) & otherwise \end{cases} \tag{8-30}$$

最終的には、*int8* 領域の信号 $x_{\mathrm{int}8}(n)$ は、式(6-13)、(6-14)、(6-15)、 (6-16)を使用して、A 則／μ則領域に写像される。

9 標準JT-G711.0 コーデックのビットイグザクトな記述

16 ビット固定小数点において標準ＪＴ－Ｇ７１１．０コーデックをシミュレートした ANSI C コードは、ＩＴＵ－ＴのＷｅｂサイトから入手可能である。以下の節は、このシミュレーションコードの使用方法と、ソフトウェア構成方法について、要約する。

9.1 シミュレーションソフトウェアの使用

C コードは、ITU-T G.711.0 コーデック実装(source/g711llc フォルダ)、幾つかの補助ソースコードと、符号化器と復号器の両方をシミュレーションすることが可能なコマンドラインアプリケーションのソースコード(source/g711llc.c)から構成される。、符号化のコマンドラインは、以下のようである。

g711llc [-v] [-e|-enc] [-u|-a] [-n#] <infile> [<outfile>]

ここで

<infile>	符号化する入力ビット列ファイルの名前。ファイルの拡張子は、μ則の場合は、ul8、mu8、mu、A 則の場合は、al8、al、または、その他の拡張子(この場合、符号則タイプを、手動で指定する)である。
[<outfile>]	出力ビット列ファイルの名前。優先される拡張子は、μ則の場合は、” lcm”、A 則の場合は、”lca” である。出力ファイル名が指定されない場合、入力ファイルの拡張子が優先される拡張子に置き換えられて生成される。
[-v]	詳細情報出力モード。このモードが、指定された場合、符号化器は、各フレームの符号化後に、標準出力に対する拡張情報を出力する。
[-e|-enc]	符号化モード指定。通常、どちらのフラグも指定する必要はなく、符号化モードは、入力ファイル名から推定される。

[-u|-a]　　符号則の指定。通常、どちらのフラグも指定する必要はなく、符号則は、入力ファイル名から推定される。

[-n#]　　フレーム長の指定。#は、40、80、160、240 サンプルの何れか 1 つを指定する。指定されない場合、160 サンプルが指定される。

復号のコマンドラインは、以下のようである。

g711llc [-v] [-u|-a] <infile> [<outfile>]

ここで、

<infile>　　復号する入力ビット列ファイルの名前。ファイルの拡張子は、μ則の場合は、lcm、A 則の場合は、lca、または、その他の拡張子(この場合、符号則タイプを、手動で指定する)である。

[<outfile>]　　復号された出力ファイルの名前。優先される拡張子は、μ則の場合は、”muo”、A 則の場合は、”alo” である。出力ファイル名が指定されない場合、入力ファイルの拡張子が優先される拡張子に置き換えられて生成される。

[-v]　　詳細情報出力モード。このモードが、指定された場合、復号器は、各フレームの復号後に、標準出力に対する拡張情報を出力する。

[-u|-a]　　符号則の指定。通常、どちらのフラグも指定する必要はなく、符号則は、入力ファイル名から推定される。

上記の記載の中で、角括弧は、取り囲まれたコマンドラインオプションが省略できることを意味しており、縦線は、縦線で分割されたオプションから選択することを意味している。符号化器入力と復号器出力のファイルは、G．７１１PCM 信号を含んだ抽出データファイルである。符号化器出力と、復号器入力ファイルは、連続した ITU-T G.711.0 フレーム（本標準において定義されている）から構成される。

9.2　シミュレーションソフトウェアの構成

Tables 9-1／JT-G711.0 から Tables 9-4／JT-G711.0 にシミュレーションソフトウェアの構成を記載する。

Table 9-1/JT-G711.0　– Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
max_prediction_orders	P_{max}	5	Q0	Maximum prediction orders for each frame length	(1)
tbl_ulaw_to_pcm		256	Q0	μ-law to PCM conversion table	(2)
tbl_alaw_to_pcm		256	Q0	A-law to PCM conversion table	(3)
hIndex_len		4	Q0	Huffman code lengths for the E-Huffman table indices	(4)
hIndex_value		4	Q0	Huffman code words for the E-Huffman table indices	(5)
dIndex		8	Q0	Huffman decode lookup table for the E-Huffman table indices	(6)
Huffman_maxCodeValue		4	Q0	The smallest quotient value that requires escape code for each E-Huffman table	(7)
Huffman_table_len		4×8	Q0	Huffman code lengths for the E-Huffman coding	(8)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
Huffman_table_value		4×8	Q0	Huffman code words for the E-Huffman coding	(9)
diff_length		3×8	Q0	Table for optimal E-Huffman table index estimation	(10)
Huffman_index		4×64	Q0	E-Huffman decode lookup tables	(11)
pc_expand01_largeframe	$2^{15}\hat{k}_j$	128	Q15	Reconstruction table for the first two PARCOR coefficients	(12)
pc_bits_largeframe_o1	U_j	1	Q0	PARCOR quantization precision when LPC order = 1	(13)
pc_bits_largeframe_o5	U_j	5	Q0	PARCOR quantization precision when LPC order = 5	(14)
pc_bits_largeframe_others	U_j	12	Q0	PARCOR quantization precision when LPC order = 8, 10, 12	(15)
pc_codes_largeframe_o1		8	Q0	Huffman code words for the PARCOR coefficients when LPC order = 1	(16)
pc_code_lengths_largeframe_o1		8	Q0	Huffman code lengths for the PARCOR coefficients when LPC order = 1	(17)
pc_codes_largeframe_o5		80	Q0	Huffman code words for the PARCOR coefficients when LPC order = 5	(18)
pc_code_lengths_largeframe_o5		80	Q0	Huffman code lengths for the PARCOR coefficients when LPC order = 5	(19)
pc_codes_largeframe_others		144	Q0	Huffman code words for the PARCOR coefficients when LPC order = 8, 10, 12	(20)
pc_code_lengths_largeframe_ others		144	Q0	Huffman code lengths for the PARCOR coefficients when LPC order = 8, 10, 12	(21)
pc_bits_largeframe		13	PTR	Pointer table for the above pc_bits_largeframe_... tables	(22)
pc_codes_largeframe		13	PTR	Pointer table for the above pc_codes_largeframe_... tables	(23)
pc_code_lengths_largeframe		13	PTR	Pointer table for the above pc_code_lengths_largeframe_... tables	(24)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
pc_ind_smallframe_3	X_j	8	Q0	PARCOR quantization mapping table when bits = 3	(25)
pc_ind_smallframe_4	X_j	16	Q0	PARCOR quantization mapping table when bits = 4	(26)
pc_ind_smallframe_5	X_j	32	Q0	PARCOR quantization mapping table when bits = 5	(27)
pc_ind_smallframe		6	PTR	Pointer table for the above pc_ind_smallframe_... tables	(28)
pc_val_smallframe_3	$2^{15}\hat{k}_j$	6	Q15	PARCOR reconstruction tables when bits = 3	(29)
pc_val_smallframe_4	$2^{15}\hat{k}_j$	12	Q15	PARCOR reconstruction tables when bits = 4	(30)
pc_val_smallframe_5	$2^{15}\hat{k}_j$	24	Q15	PARCOR reconstruction tables when bits = 5	(31)
pc_val_smallframe		6	PTR	Pointer table for the above pc_val_smallframe_... tables	(32)
pc_bits_smallframe_n40_80o1	U_j	1	Q0	PARCOR quantization precision when frame length = 40, 80, LPC order = 1	(33)
pc_bits_smallframe_n40o2_3	U_j	3	Q0	PARCOR quantization precision when frame length = 40, LPC order = 2, 1	(34)
pc_bits_smallframe_n40o4	U_j	4	Q0	PARCOR quantization precision when frame length = 40, LPC order = 4	(35)
pc_bits_smallframe_n80oge2	U_j	8	Q0	PARCOR quantization precision when frame length = 80, LPC order = 2, 6, 8	(36)
pc_bits_smallframe_n40		5	PTR	Pointer table for the above pc_bits_smallframe_* tables when frame length = 40	(37)
pc_bits_smallframe_n80		9	PTR	Pointer table for the above pc_bits_smallframe_* tables when frame length = 80	(38)
pc_bits_smallframe		2	PTR	Pointer table for the above two pointer tables	(39)
pc_num_smallframe_n40_80o1		1	Q0	PARCOR quantization ranges when frame length = 40, 80, LPC order = 1	(40)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
pc_num_smallframe_n40o2_3		3	Q0	PARCOR quantization ranges when frame length = 40, LPC order = 2, 3	(41)
pc_num_smallframe_n40o4		4	Q0	PARCOR quantization ranges when frame length = 40, LPC order = 4	(42)
pc_num_smallframe_n80oge2		8	Q0	PARCOR quantization ranges when frame length = 80, LPC order = 2, 6, 8	(43)
pc_num_smallframe_n40		5	PTR	Pointer table for the above pc_num_smallframe_... tables when frame length = 40	(44)
pc_num_smallframe_n80		9	PTR	Pointer table for the above pc_num_smallframe_... tables when frame length = 80	(45)
pc_num_smallframe		2	PTR	Pointer table for the above two pointer tables	(46)
pc_rice_smallframe_n40_80_o1_enc_dec	X_j, Y_j	6	Q0	Rice values for the quantized PARCOR coefficients (and vice versa) when frame length = 40, 80, LPC order = 1	(47)
pc_rice_smallframe_n40_o2_enc	Y_j	18	Q0	Rice values for the PARCOR indices when frame length = 40, LPC order = 2	(48)
pc_rice_smallframe_n40_o2_dec	X_j	18	Q0	PARCOR indices for the Rice values when frame length = 40, LPC order = 2	(49)
pc_rice_smallframe_n40_o3_enc	Y_j	24	Q0	Rice values for the PARCOR indices when frame length = 40, LPC order = 3	(50)
pc_rice_smallframe_n40_o3_dec	X_j	24	Q0	PARCOR indices for the Rice values when frame length = 40, LPC order = 3	(51)
pc_rice_smallframe_n40_o4_enc	Y_j	42	Q0	Rice values for the PARCOR indices when frame length = 40, LPC order = 4	(52)
pc_rice_smallframe_n40_o4_dec	X_j	42	Q0	PARCOR indices for the Rice values when frame length = 40, LPC order = 4	(53)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
pc_rice_smallframe_n80_o2_enc	Y_j	36	Q0	Rice values for the PARCOR indices when frame length = 80, LPC order = 2	(54)
pc_rice_smallframe_n80_o2_dec	X_j	36	Q0	PARCOR indices for the Rice values when frame length = 80, LPC order = 2	(55)
pc_rice_smallframe_n80_o6_enc	Y_j	84	Q0	Rice values for the PARCOR indices when frame length = 80, LPC order = 6	(56)
pc_rice_smallframe_n80_o6_dec	X_j	84	Q0	PARCOR indices for the Rice values when frame length = 80, LPC order = 6	(57)
pc_rice_smallframe_n80_o8_enc	Y_j	102	Q0	Rice values for the PARCOR indices when frame length = 80, LPC order = 8	(58)
pc_rice_smallframe_n80_o8_dec	X_j	102	Q0	PARCOR indices for the Rice values when frame length = 80, LPC order = 8	(59)
pc_riceval_smallframe_enc_n40		5	PTR	Pointer table for the above pc_rice_smallframe_..._enc tables when frame length = 40	(60)
pc_riceval_smallframe_dec_n40		5	PTR	Pointer table for the above pc_rice_smallframe_..._dec tables when frame length = 40	(61)
pc_riceval_smallframe_enc_n80		9	PTR	Pointer table for the above pc_rice_smallframe_..._enc tables when frame length = 80	(62)
pc_riceval_smallframe_dec_n80		9	PTR	Pointer table for the above pc_rice_smallframe_..._dec tables when frame length = 80	(63)
pc_riceval_smallframe_enc		2	PTR	Pointer table for the above pc_riceval_smallframe_enc_... pointer tables	(64)
pc_riceval_smallframe_dec		2	PTR	Pointer table for the above pc_riceval_smallframe_dec_... pointer tables	(65)
pc_quantize_smallframe_nonzero	W_j	6	Q0	Negative PARCOR index mapping helper table	(66)
pc_ricepara_smallframe		6	Q0	Rice parameters for each quantization precision (3, 4, 5)	(67)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
box_l4	$p(j)$	4	Q0	Possible prediction orders when frame length = 40	(68)
box_8	$p(j)$	4	Q0	Possible prediction orders when frame length = 80	(69)
box_10	$p(j)$	4	Q0	Possible prediction orders when frame length = 160, 240	(70)
box_12	$p(j)$	4	Q0	Possible prediction orders when frame length = 320	(71)
clz_lut		256	Q0	Lookup table for 8-bit Count Leading Zeros operation	(72)
from_linear_ulaw		256	Q0	μ-law to *int8* and vice versa conversion helper table	(73)
from_linear_alaw		256	Q0	A-law to *int8* and vice versa conversion helper table	(74)
y_anchor_table		30	Q0	Min-Max level decoder anchor locations	(75)
G711Z_mask		9	Q0	Min-Max level decoder bit reader masking table	(76)
tab_anchor		39	Q0	Min-Max level encoder anchor table for the [–87, –49] range	(77)
tab_codepoint		39	Q0	Min-Max level encoder codepoint table for the [–87, –49] range	(78)
tab_anchor1		12	Q0	Min-Max level encoder anchor table for the [–29, –18] range	(79)
tab_codepoint1		12	Q0	Min-Max level encoder codepoint table for the [–29, –18] range	(80)
tab_codepoint2		5	Q0	Min-Max level encoder table for calculating the number of required bits per sample after anchoring	(81)
cosrect_win32		32	Q15	Window for LPC analysis when frame length = 160, 320	(82)
cosrect_win24		24	Q15	Window for LPC analysis when frame length = 240	(83)
win40_1	$W_{\mathrm{PCM},40}(n)$	4	Q15	Window for LPC analysis when frame length = 40	(84)
win40_2	$W_{\mathrm{PCM},40}(n)$	4	Q15	Window for LPC analysis when frame length = 40	(85)
win80_1	$W_{\mathrm{PCM},80}(n)$	8	Q15	Window for LPC analysis when frame length = 80	(86)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
win80_2	$W_{PCM,80}(n)$	8	Q15	Window for LPC analysis when frame length = 80	(87)
pm_max_rice_params		5	Q0	Maximal Rice parameters for the PM zero Rice coding and Pulse mode coding tools for each frame length	(88)
const_pm_zero_rice_value40		6	Q0	Code words for the Rice parameters of the PM zero Rice coding and Pulse mode coding tools when frame length = 40	(89)
const_pm_zero_rice_len40		6	Q0	Code lengths for the Rice parameters of the PM zero Rice coding and Pulse mode coding tools when frame length = 40	(90)
const_pm_zero_rice_value80		7	Q0	Code words for the Rice parameters of the PM zero Rice coding and Pulse mode coding tools when frame length = 80	(91)
const_pm_zero_rice_len80		7	Q0	Code lengths for the Rice parameters of the PM zero Rice coding and Pulse mode coding tools when frame length = 80	(92)
const_pm_zero_rice_value320		10	Q0	Code words for the Rice parameters of the PM zero Rice coding and Pulse mode coding tools when frame length = 160, 240, 320	(93)
const_pm_zero_rice_len320		10	Q0	Code lengths for the Rice parameters of the PM zero Rice coding and Pulse mode coding tools when frame length = 160, 240, 320	(94)
const_pm_zero_rice_values		5	PTR	Pointer table for the above const_pm_zero_rice_value_... tables	(95)
const_pm_zero_rice_lengths		5	PTR	Pointer table for the above const_pm_zero_rice_len_... tables	(96)
pulse_pos_lengths		5	Q0	Number of signalling bits of the pulse position for each frame length	(97)

Table 9-1/JT-G711.0 – Tables in C-code

(ITU-T G.711.0)

Table name	Symbol	Size	Format	Description	No.
p_order		13	Q0	Helper table for indexing a triangular matrix	(98)
ipar_multipliers	α	5	Q15	Weighting factors of the PARCOR coefficients for cost estimation	(99)
ave_multiplier		3	Q15	Helper table for division by 2, 3, and 4	(100)
qave_multiplier		3	Q15	Helper table for division by 16/2, 16/3, 16/4	(101)
num_range		9	Q0	Table used for estimating the code length of the Min-max level coding tool	(102)
diff_bit_num		5	Q0	Number of signalling bits for the LTP sub-frame pitch differences for each frame length	(103)
autocorr_lag_tab	$w_{lag}(i)$	13	Q15	Lag window coefficients for bandwidth expansion	(104)
Rice_map		16	Q0	Mapping table for the separation parameter	(105)
Rice_map_inv		16	Q0	Inverse mapping table for the separation parameter	(106)
map_ss0		5×8	Q0	Modified Rice parameter table for first sample of the residual	(107)
map_ss1		5×8	Q0	Modified Rice parameter table for second sample of the residual	(108)
autocorr_pre_Tflag_count		3	Q0	Tflag_pre counter thresholds for frame length = 160, 240, 320	(109)
bits_per_block		5	Q0	Optimal number of bits per block for levels 2-6 for the fractional bit coding tool	(110)
samp_per_block		5	Q0	Optimal number of samples per block for levels 2-6 for the fractional bit coding tool	(111)
bytes_per_frame		5×5	Q0	Number of octets per coded frame for the fractional bit coding tool, given number of levels and frame length index	(112)
fb_type_offsets		5	Q0	Helper table to generate the first signalling octet for the Fractional bit coding tool	(113)

Table 9-2 JT-G711.0 – Summary of encoder specific routines

(ITU-T G.711.0)

Filename	Description
autocorr.c	Functions for calculating the autocorrelation of signals. Bandwidth expansion.
g711llc_encoder.c	ITU-T G.711.0 encoder main control logic and encoder functions.
G711Zencode_function.c	Min-Max level encoder functions.
output_bit_stream.c	Bitstream writer and entropy coder functions.
window.c	Window functions.
g711llc_encode_file.c	Auxiliary function for ITU-T G.711.0 encoding of files.

Table 9-3 JT-G711.0 – Summary of decoder specific routines

(ITU-T G.711.0)

Filename	Description
g711llc_decoder.c	ITU-T G.711.0 decoder main control logic and decoder functions.
G711Zdecode_function.c	Min-Max level decoder function.
input_bit_stream.c	Bitstream reader and entropy decoder functions.
g711llc_decode_file.c	Auxiliary function for ITU-T G.711.0 decoding of files.

Table 9-4 JT-G711.0 – Summary of common routines

(ITU-T G.711.0)

Filename	Description
fract_bits.c	Coding of number of levels by blocked fractional-bit representation.
g711_itu.c	Linear PCM to ITU-T G.711 PCM conversion routines.
mapping.c	ITU-T G.711 PCM and Linear PCM subtraction functions.
multirun.c	Multi-level run-length coder functions.
parcor.c	Functions related to the PARCOR coefficients.
tables.c	Definitions of read-only tables.
wmops_timer.c	Auxiliary functions for measuring WMOPS complexity.
stack_profile.c	Auxiliary functions for measuring the worst-case stack use.

付録

（標準ＪＴ－Ｇ７１１．０に対する）

用語対照表

英　　語	ＴＴＣ標準用語
additional code	拡張符号
binary	バイナリ（２進）
bit-packing format	ビットパッキングフォーマット
code point	符号値
data-range	データ範囲
E-Huffman	エスケイプハフマン
floor function	floor 関数
functional discription	機能記述
log PCM	対数 PCM
long-term analysis	長期相関分析
LPC	線形予測
LTP contribution	長期予測寄与分
most negative level	負の最大レベル
octet	オクテット
overhead octet	オーバーヘッドオクテット
padding octet	パディングオクテット
PARCOR	偏自己相関
prefix code	接頭符号
progressive	漸増
self-discribing	自己記述的
sub tool	補助ツール
table signaling	テーブルインデックス符号
unary	ユナリ（１進）
verbose mode	詳細情報出力モード
VoIP	ボイスオーバーIP
WMOPS	重み付け MOPS