UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

G.728

Anexo J

(09/99)

SERIE G: SISTEMAS Y MEDIOS DE TRANSMISIÓN, SISTEMAS Y REDES DIGITALES

Sistemas de transmisión digital – Equipos terminales – Codificación de señales analógicas mediante métodos diferentes de la MIC

Codificación de señales vocales a 16 kbit/s utilizando predicción lineal con excitación por código de bajo retardo

Anexo J: Funcionamiento a velocidad binaria variable de la predicción lineal con excitación por código de bajo retardo principalmente para aplicaciones de datos en la banda vocal en equipos de multiplicación de circuitos digitales

Recomendación UIT-T G.728 – Anexo J

(Anteriormente Recomendación del CCITT)

RECOMENDACIONES UIT-T DE LA SERIE G

SISTEMAS Y MEDIOS DE TRANSMISIÓN, SISTEMAS Y REDES DIGITALES

CONEXIONES Y CIRCUITOS TELEFÓNICOS INTERNACIONALES	G.100–G.199
SISTEMAS INTERNACIONALES ANALÓGICOS DE PORTADORAS	
CARACTERÍSTICAS GENERALES COMUNES A TODOS LOS SISTEMAS ANALÓGICOS DE PORTADORAS	G.200–G.299
CARACTERÍSTICAS INDIVIDUALES DE LOS SISTEMAS TELEFÓNICOS INTERNACIONALES DE PORTADORAS EN LÍNEAS METÁLICAS	G.300–G.399
CARACTERÍSTICAS GENERALES DE LOS SISTEMAS TELEFÓNICOS INTERNACIONALES EN RADIOENLACES O POR SATÉLITE E INTERCONEXIÓN CON LOS SISTEMAS EN LÍNEAS METÁLICAS	G.400–G.449
COORDINACIÓN DE LA RADIOTELEFONÍA Y LA TELEFONÍA EN LÍNEA	G.450–G.499
EQUIPOS DE PRUEBAS	
CARACTERÍSTICAS DE LOS MEDIOS DE TRANSMISIÓN	
SISTEMAS DE TRANSMISIÓN DIGITAL	
EQUIPOS TERMINALES	G.700–G.799
Generalidades	G.700–G.709
Codificación de señales analógicas mediante modulación por impulsos codificados (MIC)	G.710–G.719
Codificación de señales analógicas mediante métodos diferentes de la MIC	**G.720–G.729**
Características principales de los equipos múltiplex primarios	G.730–G.739
Características principales de los equipos múltiplex de segundo orden	G.740–G.749
Características principales de los equipos múltiplex de orden superior	G.750–G.759
Características principales de los transcodificadores y de los equipos de multiplicación de circuitos digitales	G.760–G.769
Características de operación, administración y mantenimiento de los equipos de transmisión	G.770–G.779
Características principales de los equipos múltiplex de la jerarquía digital síncrona	G.780–G.789
Otros equipos terminales	G.790–G.799
REDES DIGITALES	G.800–G.899
SECCIONES DIGITALES Y SISTEMAS DIGITALES DE LÍNEA	G.900–G.999

Para más información, véase la Lista de Recomendaciones del UIT-T.

RECOMENDACIÓN UIT-T G.728

CODIFICACIÓN DE SEÑALES VOCALES A 16 kbit/s UTILIZANDO PREDICCIÓN LINEAL CON EXCITACIÓN POR CÓDIGO DE BAJO RETARDO

ANEXO J

Funcionamiento a velocidad binaria variable de la predicción lineal con excitación por código de bajo retardo principalmente para aplicaciones de datos en la banda vocal en equipos de multiplicación de circuitos digitales

Resumen

El anexo J a la Recomendación G.728 define una ampliación a 40 kbit/s, optimizada para señales de datos en la banda vocal, de la especificación de la aritmética de coma fija a 16 kbit/s del actual anexo G/G.728. La principal diferencia entre el códec aquí descrito y el códec descrito en el anexo G/G.728 consiste en que, para la búsqueda en el libro de códigos, se aplica un método de cuantificación con codificación reticular (TCQ, *trellis-coded quantization*). El método TCQ reemplaza al método de búsqueda de análisis por síntesis empleado para la búsqueda en el libro de códigos, en la Recomendación G.728, únicamente en el modo datos en la banda vocal (VBD, *voiceband data*).

La adaptación hacia atrás del predictor conseguida en el modo VBD es casi igual a la adaptación hacia atrás conseguida en el modo conversación (Recomendación G.728). Además, se utiliza el mismo ciclo de adaptación para el modo conversación (Recomendación G.728) y para el modo VBD. En el modo conversación, la velocidad de 40 kbit/s pasa a ser la de la predicción lineal con excitación por código de bajo retardo de la Recomendación G.728.

Este anexo incluye un fichero electrónico con vectores de prueba para verificación de la implementación del anexo J/G.728.

Orígenes

El anexo J a la Recomendación G.728 ha sido preparado por la Comisión de Estudio 16 (1997-2000) del UIT-T y fue aprobado por el procedimiento de la Resolución N.° 1 de la CMNT el 30 de septiembre de 1999.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución N.° 1 de la CMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión *empresa de explotación reconocida (EER)* designa a toda persona, compañía, empresa u organización gubernamental que explote un servicio de correspondencia pública. Los términos *Administración, EER* y *correspondencia pública* están definidos en la *Constitución de la UIT (Ginebra, 1992)*.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

© UIT 2000

Es propiedad. Ninguna parte de esta publicación puede reproducirse o utilizarse, de ninguna forma o por ningún medio, sea éste electrónico o mecánico, de fotocopia o de microfilm, sin previa autorización escrita por parte de la UIT.

ÍNDICE

		Página
J.1	Alcance	1
J.2	Referencias normativas	1
J.3	Visión de conjunto	2
J.4	Descripción del algoritmo	2
J.4.1	Estructura del códec	2
J.4.2	Estructura del decodificador	6
J.4.3	Descripción detallada del codificador	7
J.4.4	Descripción detallada del decodificador	21
J.4.5	Descripción detallada de los módulos de conmutación de modo	23
J.4.6	Cuadros de transiciones reticulares	26
J.4.7	Coeficientes del polinomio utilizado en el calculador logarítmico	29
J.4.8	Coeficientes de expansión de la anchura de banda para el modo datos	29
J.4.9	Bloques internos	30
J.4.10	Variables y constantes utilizadas en el procesamiento interno	30
J.4.11	Valores iniciales	34
J.5	Bibliografía	35

Fichero electrónico

– Vectores de prueba

Recomendación G.728

CODIFICACIÓN DE SEÑALES VOCALES A 16 kbit/s UTILIZANDO PREDICCIÓN LINEAL CON EXCITACIÓN POR CÓDIGO DE BAJO RETARDO

ANEXO J

Funcionamiento a velocidad binaria variable de la predicción lineal con excitación por código de bajo retardo principalmente para aplicaciones de datos en la banda vocal en equipos de multiplicación de circuitos digitales[1]

(Ginebra, 1999)

J.1 Alcance

Este anexo proporciona información para la implementación del códec, y las modificaciones requeridas en el anexo G/G.728 para permitir una conmutación de modo en un dispositivo que emplea aritmética de coma fija.

El anexo J especifica el funcionamiento a una velocidad de transmisión de 40 kbit/s. El retardo algorítmico tiene una duración correspondiente a la longitud de 5 muestras (0,625 ms), lo que es exactamente igual a la del algoritmo del anexo G y a las de todos los algoritmos de predicción lineal con excitación por código de bajo retardo (LD-CELP, *low-delay code excited linear prediction*) especificados en la Recomendación G.728. El algoritmo de datos en la banda vocal (VBD, *voiceband data*) a 40 kbit/s del anexo J está previsto para la transmisión con compresión de señales VBD, en aplicaciones tales como los equipos de multiplicación de circuitos digitales (DCME, *digital circuit multiplication equipment*). El algoritmo permite una transición blanda hacia y desde la LD-CELP (Recomendación G.728) y ha sido también concebido para mantener señales vocales con la calidad propia de las comunicaciones de larga distancia. Se tiene previsto que el anexo J reemplace al modo MIC diferencial adaptativa a 40 kbit/s (Recomendación G.726) en sistemas DCME que incorporan LD-CELP (Recomendación G.728).

J.2 Referencias normativas

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes.

[1] Recomendación CCITT G.728 (1992), *Codificación de señales vocales a 16 kbit/s utilizando predicción lineal con excitación por código de bajo retardo*.

[2] Recomendación UIT-T G.728 anexo G (1994), *Especificación de coma fija a 16 kbit/s*.

[3] Recomendación UIT-T G.763 (1998), *Equipo de multiplicación de circuitos digitales que emplea modulación por impulsos codificados diferencial adaptativa (Recomendación G.726) e interpolación digital de la palabra*.

[1] Este anexo incluye un fichero electrónico con vectores de prueba para verificación de la implementación.

J.3 Visión de conjunto

El códec utiliza una velocidad de transmisión de 40 kbit/s. El retardo logarítmico tiene una longitud igual a la de 5 muestras, o sea 0,625 ms. El códec puede efectuar una conmutación de modo cada "ciclo de adaptación" (2,5 ms).

La principal diferencia entre el códec aquí descrito y el códec descrito en el anexo G/G.728 consiste en que, para la búsqueda en el libro de códigos, se utiliza un método de cuantificación con codificación reticular (TCQ, *trellis-coded quantification*). El método TCQ reemplaza al método de análisis por síntesis para la búsqueda en el libro de códigos de la Recomendación G.728, únicamente en el modo VBD.

La adaptación hacia atrás del predictor conseguida en el modo VBD es casi idéntica a la adaptación hacia atrás conseguida en el modo conversación (Recomendación G.728). Además, se utiliza el mismo ciclo de adaptación para el modo conversación (Recomendación G.728) y para el modo VBD. En el modo conversación, la velocidad de 40 kbit/s pasa a ser la de la predicción lineal con excitación por código de bajo retardo de la Recomendación G.728.

La cláusula J.4, Descripción del algoritmo, proporciona la descripción completa del funcionamiento del códec. En J.4.3 – Codificador, comienza una descripción detallada de la implementación; en J.4.4 se presentan detalles adicionales relativos al decodificador, y en J.4.5 se presentan detalles de la conmutación de modo.

J.4 Descripción del algoritmo

J.4.1 Estructura del códec

El algoritmo se basa en una estructura de codificación predictiva lineal con excitación por residuo (RELP, *residual excited linear predictive coding*) y se divide en los siguientes bloques (figura J.1):

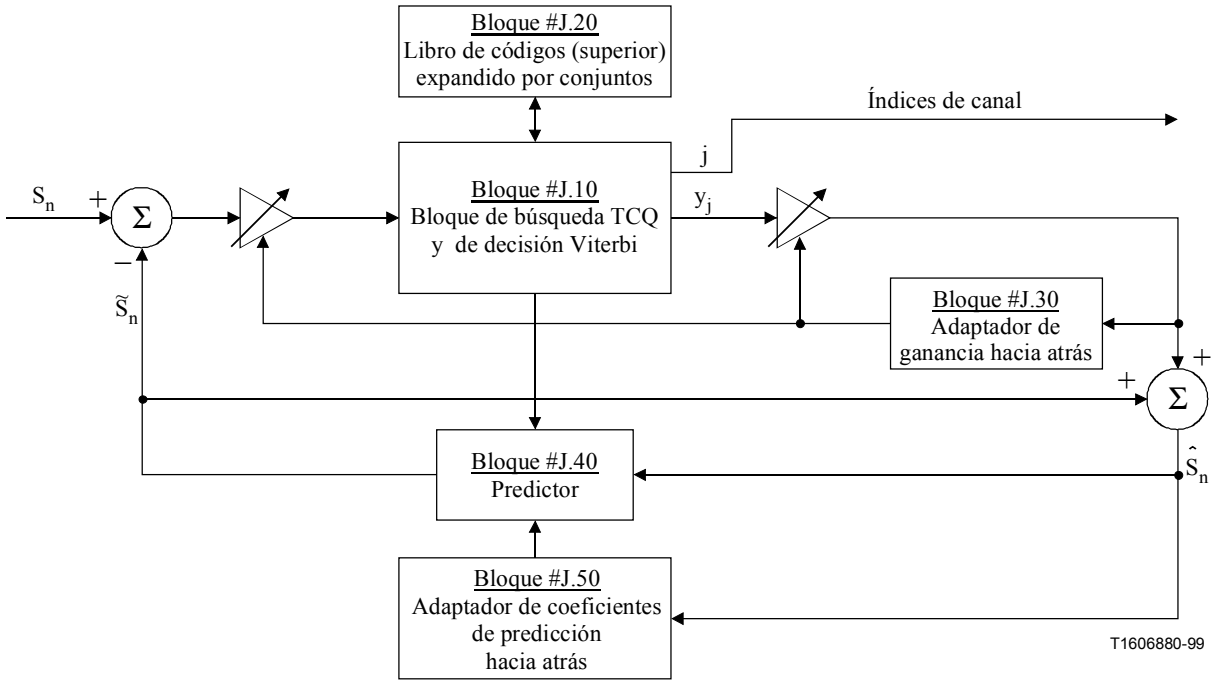

Figura J.1/G.728 – Estructura del codificador

J.4.1.1 Bloque #J.10 – Bloque de búsqueda TCQ y de decisión Viterbi

Este bloque realiza todas las operaciones requeridas para el algoritmo de cuantificación con codificación reticular (TCQ, *trellis-coded quantization*), como la gestión de los supervivientes reticulares y de los valores de reproducción especificados, cálculo y comparación de matrices, y determinación de las decisiones Viterbi. El codificador libera cinco símbolos de canal, seleccionados por el algoritmo Viterbi, a partir del mejor superviviente para las cinco muestras de origen.

La figura J.2 representa la máquina de estados que genera el diagrama reticular y la figura J.3 muestra el diagrama reticular. Los cuadros en J.4.6.1, J.4.6.2, J.4.6.3, y J.4.6.4 se han derivado de las figuras antes mencionadas, como se describe más adelante.

La subcláusula J.4.6.1 proporciona el trayecto permitido hacia los nodos **precedentes,** a través de la retícula, para cada nodo. Por ejemplo, los nodos precedentes permitidos para el primer nodo (s[0]) son el nodo 0 bajo la rama 0 (b[0]) y el nodo 2 bajo la rama 1 (b[1]).

La subcláusula J.4.6.2 proporciona el trayecto permitido hacia los nodos **siguientes,** a través de la retícula, para cada nodo. Por ejemplo, los nodos siguientes permitidos para el primer nodo (s[0]) son el nodo 0 bajo la rama 0 (b[0]) y el nodo 2 bajo la rama 1 (b[1]).

La subcláusula J.4.6.3 proporciona el subconjunto de cuantificación {D0, D1, D2, D3} que se asocia con cada trayecto reticular. Por ejemplo, la transición de s[0] a s[0] está asociada con el subconjunto D0. La transición de s[0] a s[1] está asociada con el subconjunto D2, y las transiciones a s[2] y s[3] no están permitidas y por tanto están marcadas con X.

La subcláusula J.4.6.4 proporciona el bit de índice que etiqueta cada transición e identifica las dos ramas que emanan de cada nodo. Por ejemplo, la transición de s[0] a s[0] está asociada con 0. La transición de s[0] a s[1] está asociada con 1 (obsérvese que se utiliza el bit 5 y que 0x10 es 10h en el lenguaje de programación C), y que las transiciones a s[2] y s[3] no están permitidas, por lo están marcadas con X.

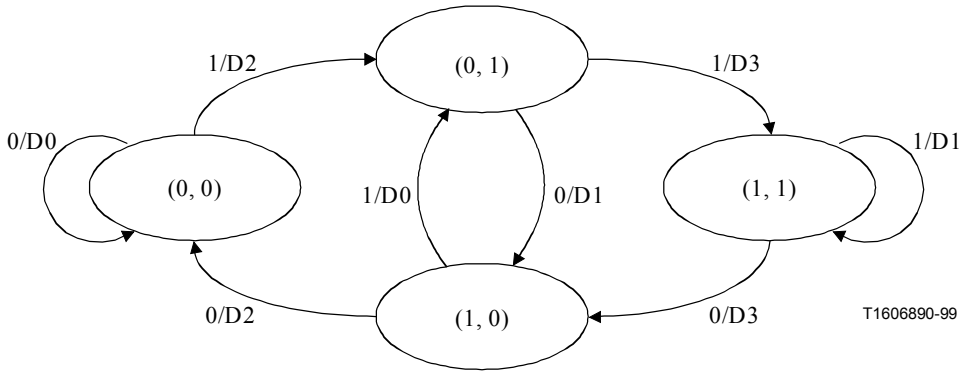

Figura J.2/G.728 – Máquina de estados TCQ

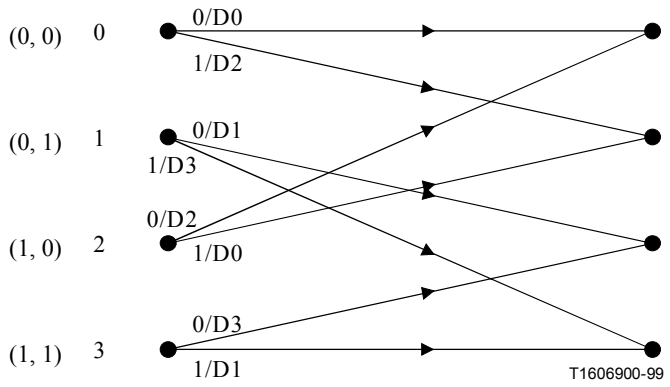

Figura J.3/G.728 – Diagrama reticular TCQ

J.4.1.2 Bloque #J.20 – Libro de códigos (superior) expandido por conjuntos

El libro de códigos (superior) (*super codebook*) es un cuantificador Lloyd-Max escalar expandido en conjuntos. Los 64 niveles de salida están divididos en cuatro subconjuntos, que comienzan por el punto más negativo y avanzan hacia el punto más positivo, etiquetando puntos consecutivos como {D0, D1, D2, D3,... , D0, D1, D2, D3}. Los niveles de cuantificación se indican en J.4.6.6 y los límites de los intervalos se indican en J.4.6.5. Los niveles pertenecientes al subconjunto D0 se muestran en la columna señalada por s[0]. Los niveles D1 se muestran debajo de s[1], los niveles D2 se muestran debajo de s[2] y los niveles D3 se muestran debajo de s[3].

J.4.1.3 Bloque #J.30 – Adaptador de ganancia hacia atrás

El esquema de adaptación de ganancia es casi igual para datos en la banda vocal (VBD) y para señales vocales de la Recomendación G.728. Las principales diferencias son las siguientes:

1) En el modo VBD, la media cuadrática (valor RMS) de los valores de salida del libro de códigos se calcula para una secuencia de niveles de salida (residuos cuantificados) que son especificados por el trayecto superviviente. La media cuadrática se calcula para una secuencia de ocho muestras. Sin embargo, a diferencia de la situación considerada en el anexo G, donde cuadros previamente calculados contienen el logaritmo de la media cuadrática (log RMS), en el modo VBD es necesario calcular el valor logarítmico de la media cuadrática. La ecuación J.4-1 da la aproximación logarítmica. Los coeficientes d_0,d_1,d_2,d_3,d_4 se proporcionan en J.4.7 y la descripción detallada del calculador logarítmico se proporciona en J.4.3.17.

$$2*\log_{10}(x) = d_0*(x-1) + d_1*(x-1)^2 + d_2*(x-1)^3 + d_3*(x-1)^4 + d_4*(x-1)^5 \qquad \text{(J.4-1)}$$

para $1 \le x < 2$

El valor log RMS reemplaza la salida del libro de códigos de forma y ganancia, bloques #G.93 y #G.94 de los cuadros de ganancia logarítmica (los dos últimos términos de la ecuación G-14).

2) Se ha introducido un filtro alisador en el bucle de ganancia logarítmica para reducir la oscilación en estado estacionario de señales con varianza estacionaria, como las formas de onda de datos en la banda vocal. Para superar las señales vocales y las señales de datos, un cuantificador de enganche dinámico (DLQ, *dynamic locking quantizer*) genera una adaptación de velocidad variable. El DLQ es similar al bloque DLQ de la Recomendación G.726.

La entrada al DLQ es la ganancia logarítmica con el desplazamiento suprimido, d(n). Esta entrada es promediada por el filtro de ponderación (J.4.3.18, bloque #J.14) para producir la ganancia enganchada G_L (*locked gain*)

El cuantificador está en el estado puramente enganchado si $a_l = 0$, y en el estado puramente desenganchado si $a_l = 1$. a_l se calcula comparando la energía a largo plazo y la energía a corto plazo de los residuos cuantificados ET(n) (J.4.3.10, bloque #J.12). La comparación da una medida de la constancia de la varianza de los residuos cuantificados.

$$G = G_U * \alpha_1 + G_L * (1 - \alpha_1) \qquad \text{(J.4-2)}$$

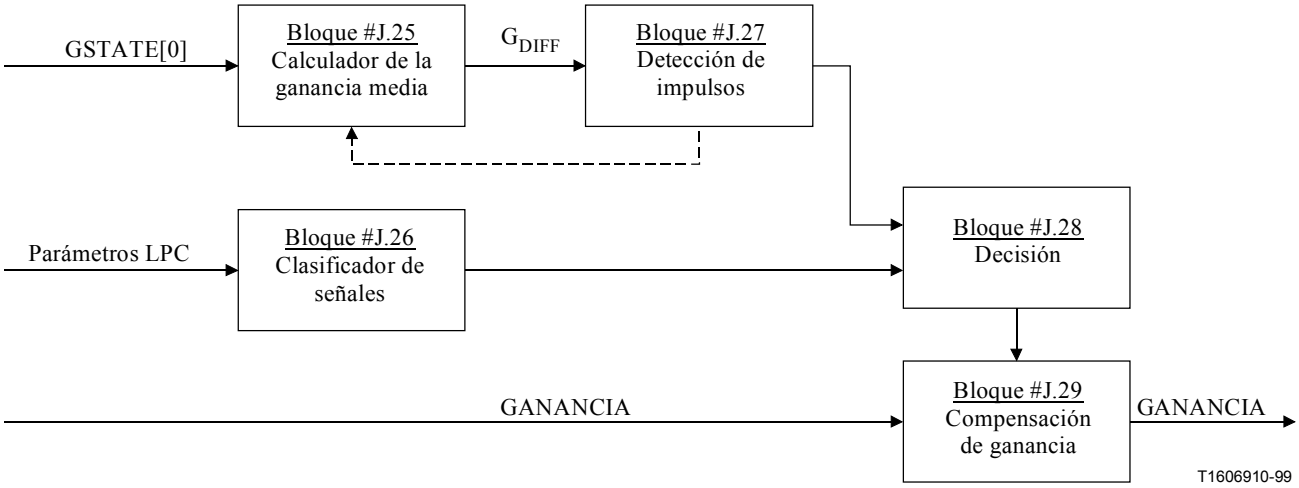

Figura J.4/G.728 – Compensación de ganancia

3) Los impulsos de error de predicción pudieran provocar la saturación del cuantificador. Para evitar esto, un grupo adicional de cinco bloques (véase la figura J.4) produce un cambio temporal en la ganancia de cuantificación. Estos bloques son:

Bloque #J.25 – Calculador de la ganancia media

Un filtro alisador calcula la media de la estimación de ganancia, G_{ave}, utilizando el valor de ganancia vectorial más reciente, GSTATE[0] (J.4.3.12, bloque #J.25 y ecuación J.4-3). La diferencia entre GSTATE[0] y G_{ave}, (G_{diff}), se calcula y se pasa al bloque detección de impulsos.

Bloque #J.27 – Detección de impulsos

Este bloque detecta los cambios bruscos de la ganancia tras un periodo predeterminado de ganancia estable (véase J.4.3.14, bloque #J.27). G_{diff} se compara con un umbral fijo. Si G_{diff} es menor que el umbral durante un periodo más largo que un lapso de tiempo predefinido, se considera que la señal es estable. Se detecta un impulso de error si G_{diff} es mayor que el umbral y la señal precedente había sido estable.

Bloque #J.26 – Clasificador de señales

En el curso de ciertas transmisiones de VBD hay mayores probabilidades de que se produzcan impulsos de error. Por tanto, tras su detección, la compensación de ganancia se maximiza. El bloque clasificador de señales detecta estas transmisiones utilizando el coeficiente LP (véase J.4.3.13, bloque #J.26).

Bloque #J.28 – Decisión

El bloque decisión acepta la salida del bloque clasificador de señales y la salida del bloque detección de impulsos, y activa el bloque compensación de ganancia (véase J.4.3.15, bloque #J.28)

Bloque #J.29 – Compensación de ganancia

Este bloque incrementa el factor de ganancia durante un determinado periodo de tiempo (a menos que se alcance cierto valor de cresta del factor de ganancia, en cuyo caso se prolonga el periodo).

$$G_{ave} = G_{const} \times G_{ave} + (1 - G_{const}) \times GSTATE[0] \qquad \text{(J.4-3)}$$

(Véase J.4.3.16, bloque #J.29.)

J.4.1.4 Bloque #J.40 – Predictor

El predictor es una versión abreviada del filtro de síntesis G.728 (bloque #G.22). El orden de LPC es 10 derivaciones, en vez de las usuales 50 derivaciones utilizadas en el filtro de síntesis. La predicción se basa en el trayecto superviviente (véase J.4.3.4, bloque #J.7) y se efectúa de la manera siguiente: En el instante n se forma una predicción de la muestra actual para cada nodo (véase J.4.3.5, bloque #J.8), utilizando la secuencia de reproducciones especificada por el superviviente seleccionado en el instante n – 1. Cuando se utiliza este método se efectúa una predicción escalar de un solo paso, y no es necesario extender la predicción a un futuro muy lejano. Esto hace que la predicción sea más "localizada" que en muchos otros esquemas VQ predictivos.

J.4.1.5 Bloque #J.50 – Adaptador de coeficientes de predicción hacia atrás

El adaptador de coeficientes de predicción hacia atrás es casi igual al adaptador de filtro de síntesis hacia atrás (bloque #G.23). Las principales diferencias son las siguientes:

- Sólo se calculan 10 parámetros LPC. El módulo de ventanización mixta (bloque #G.49) calcula constantemente 51 coeficientes de autocorrelación, con lo que se mejora la calidad de las transiciones de datos a voz.
- El factor expansión de la anchura de banda del filtro de síntesis es ahora 240/256. Los coeficientes de expansión de la anchura de banda se indican en J.4.8.

J.4.2 Estructura del decodificador

La figura J.5 muestra el diagrama de bloques del decodificador. La figura J.6 muestra el módulo de correspondencia de señales (bloque #J.60). La secuencia de 5 bits por muestra recibida del decodificador está formada por una combinación de dos tipos de secuencias de bits. El módulo de correspondencia de señales (bloque #J.60) separa la secuencia combinada formando dos secuencias j_{4n}. Una secuencia de 1 bit por muestra se expande a 2 bits mediante el empleo de un codificador convolucional. La salida de este codificador especifica el trayecto reticular y selecciona el subconjunto apropiado. La selección del subconjunto se efectúa de acuerdo con los cuadros de J.4.6.2 y J.4.6.3. La selección se explica detalladamente en J.4.4.2. Los restantes 4 bits por muestra $j_{0n} \ldots j_{3n}$ se utilizan para seleccionar el nivel de salida en el subconjunto seleccionado. El nivel de salida es sometido a escala por el adaptador de ganancia hacia atrás. El nivel de salida de la ganancia sometido a escala se introduce entonces en el predictor (bloque #J.40) y da los valores de reproducción (véase la figura J.5).

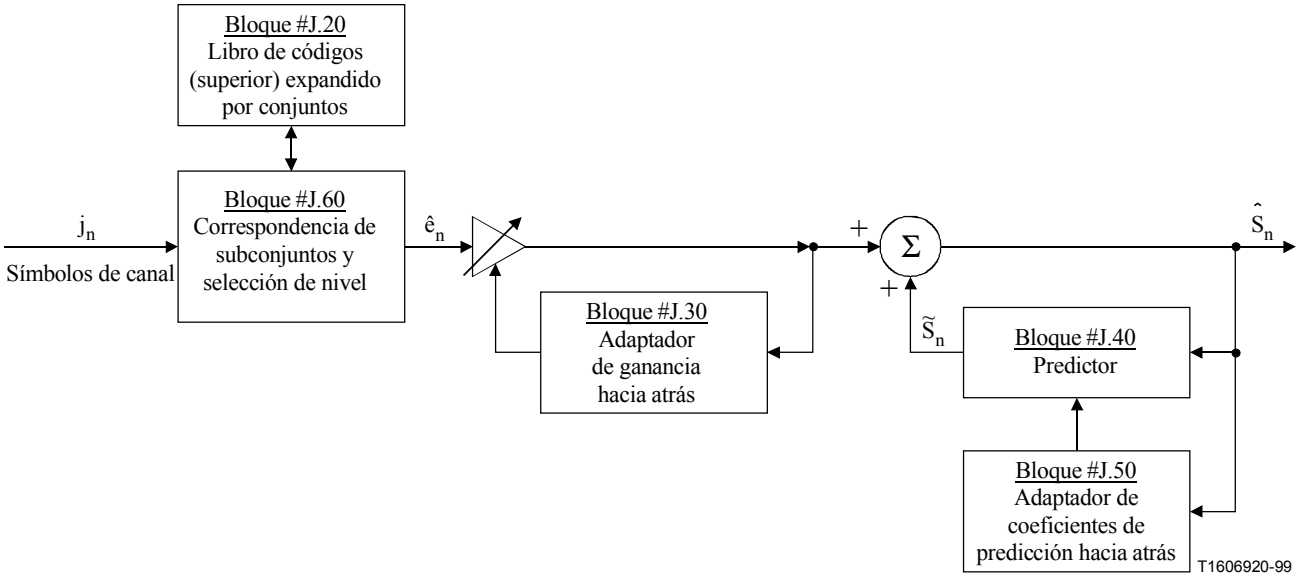

Figura J.5/G.728 – Estructura del decodificador

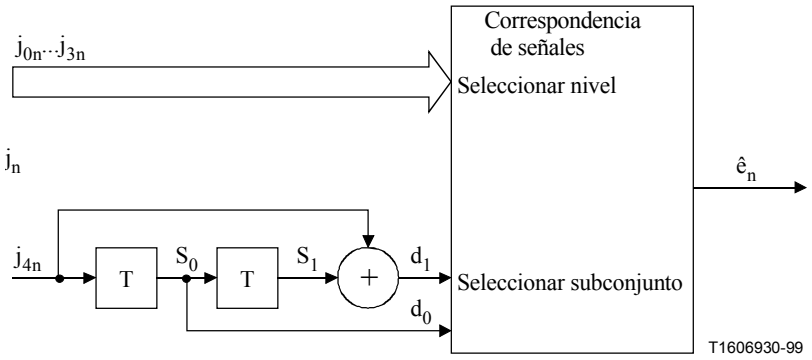

Figura J.6/G.728 – Módulo de correspondencia (bloque #J.60)

J.4.3 Descripción detallada del codificador

En esta subcláusula se da una descripción detallada de los bloques internos del codificador. La descripción se presenta como un diseño del tipo "de arriba hacia abajo"; comienza por la figura J.7 que proporciona una descripción esquemática de la operación de G.728 a diversas tasas de codificación. Los dos bloques – bloque #J.1 y bloque #J.2 – se han derivado de la figura 2/G.728 [1]. Los bloques designados por anexo H y anexo G representan los dos anexos para la Recomendación G.728 (Especificación de coma fija y modo de funcionamiento a velocidad binaria variable). El bloque #J.1 representa la conmutación de modo de conversación a datos y de datos a conversación. El bloque #J.2 representa el modo datos en la banda vocal.

En la descripción se utilizan varios operadores del lenguaje de programación C (por ejemplo: &, *, >>, <<). Se parte del supuesto de que el lector está familiarizado con las definiciones de estos operadores (Obsérvese que la palabra "**OR**" reemplaza al operador OR bit por bit del lenguaje de programación C "|").

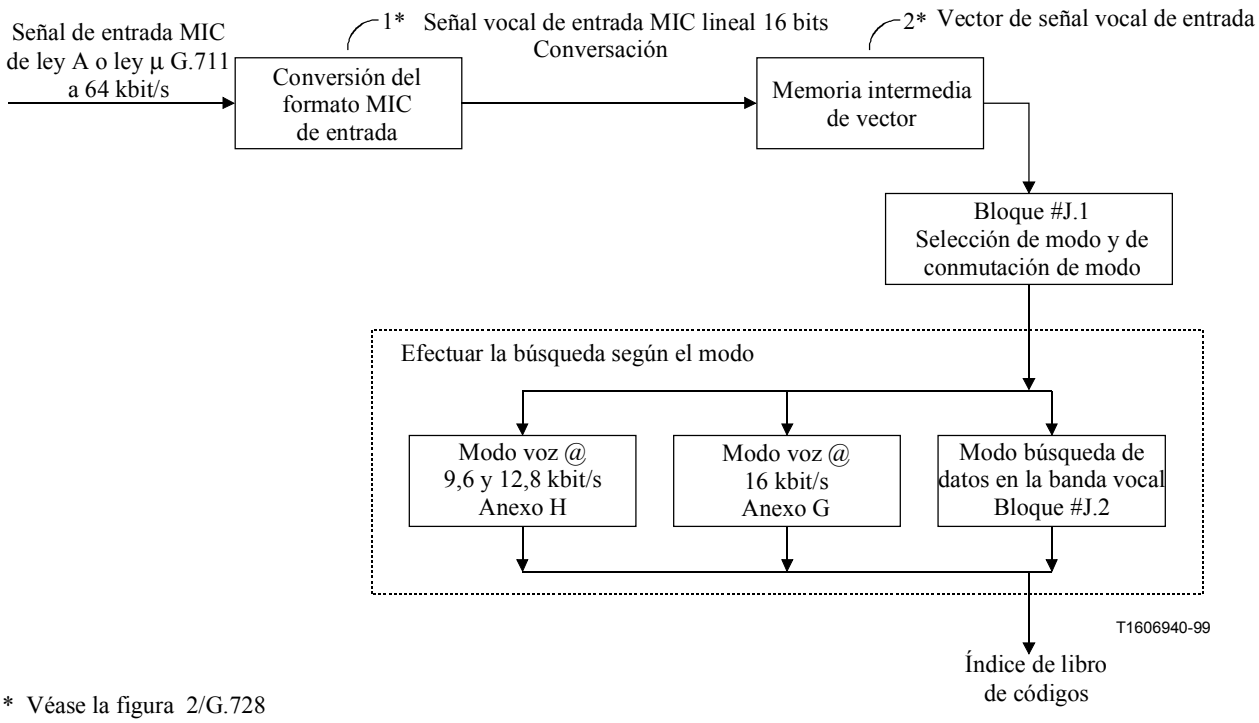

Figura J.7/G.728 – Diagrama de principio de los diversos modos de la Recomendación G.728

J.4.3.1 Bloque #J.2 – Modo búsqueda de datos en la banda vocal

Entrada: S(n)

Salida: TCQ_channel_symbols

Operación: Codificación de vector de entrada.

```
| For every Vector (5 samples) perform the following blocks.

CALL BLOCK #J.3  | Trellis search per vector module.
CALL BLOCK #J.4  | Select Best Survivor module.
CALL BLOCK #J.5  | Adaptation module.

CALL BLOCK #J.17 | Initialize next search module.
```

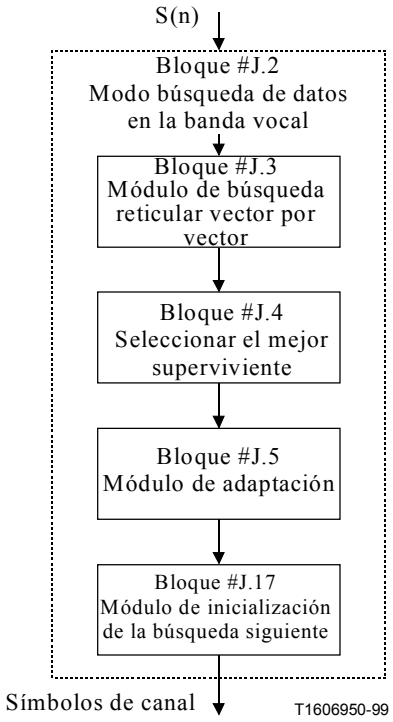

Figura J.8/G.728 – Bloque #J.2 Modo datos en la banda vocal

J.4.3.2 Bloque #J.3 – Módulo de búsqueda reticular vector por vector

Entrada: block_depth, next_stage

Salida: Véase la salida de los bloques #J.8, #J.9, #J.10

Operación: Un recorrido muestra por muestra del diagrama reticular.

```
FOR I = 1, 2, ..., IDIM | Do the next line
    CALL BLOCK #J.6 | TCQ transition.

block_depth = next_stage
next_stage = next_stage+1
IF (BLOCK_LEN <= next_stage)
    next_stage = 0
```

J.4.3.3 Bloque #J.6 – TCQ_transition

Entrada: S(n), block_depth, next_stage

Salida: Véase la salida de los bloques #J.8, #J.9, #J.10

Operación: Un solo paso a través del diagrama reticular.

```
| For every sample perform the following blocks.

CALL BLOCK #J.7 | Select Predictor state.
CALL BLOCK #J.8 | Calculate residuals.
CALL BLOCK #J.9 | Select "new" survivor.
CALL BLOCK #J.10 | Calculate reconstructed Signal.
```

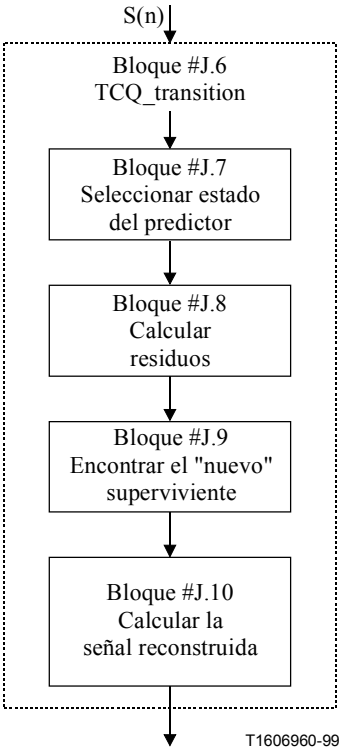

Figura J.9/G.728 – Bloque #J.6 TCQ_transition

J.4.3.4 Bloque #J.7 – Seleccionar estado del predictor

Entrada: block_depth, TCQ_reconstructed_q2

Salida: TCQ_predictor_state_q2

Operación: Para cada nodo 0, 1, 2, 3 (de la figura J.3), formar el estado del predictor rastreando hacia atrás a través del trayecto reticular con niveles reconstruidos

NOTA – El estado del filtro de predicción comprende dos secuencias:

Secuencia 1: Los niveles reconstruido anteriormente seleccionados (a partir del índice: block_depth+1, block_depth+2, ..., hasta 10), que fueron seleccionados en el anterior ciclo de búsqueda, son válidos para todos los estados reticulares (nodos) y se calculan una vez para cada muestra.

Secuencia 2: Cada nodo se asocia con una secuencia de niveles reconstruidos (a partir del índice: 1, 2, ..., hasta block_depth), que se está formando en ese momento en la retícula con niveles reconstruidos.

Este módulo forma la secuencia 2 de cada estado de predictor del nodo.

```
FOR d = 0, 1, ..., (N_STATES - 1) | Do next 6 lines
    IF (0 < block_depth)
        I=0
        src = d
        FOR k = block_depth, block_depth - 1, ..., 1 | Do the next 3 lines
            TCQ_predictor_state_q2[d][i] = TCQ_reconstructed_q2[k][src]
            src = prev_node [k] [src]
            i=i+1
```

J.4.3.5 Bloque #J.8 – Calcular residuos

Entrada: block_depth, A, TCQ_STTMP_q2, DLQ_inv_GAIN, DLQ_inv_nls_m, sample

Salida: TCQ_predict_sample_q2, TCQ_resid

Operación: Para cada modo 0, 1, 2, 3 (de la figura J.3), calcular el valor predicho y los residuos.

NOTA – Cada nodo de estado actual se asocia con una sola predicción, mientras que los siguientes nodos de estado se asocian con dos niveles de predicción (uno para cada rama entrante). Por tanto, es preferible asociar las predicciones con el nodo de estado actual (menor tiempo de procesamiento) y referenciarlas por medio de block_depth.

```
FOR d = 0, 1, ..., (N_STATES - 1) | Do lines till END #J.8

| Calculate the part of the predicted value that is valid (common)
| for all trellis states (nodes). See note of J.4.3.4.

    IF (d == 0) | Do next FOR. Calculate only once, during 1st node.
       TCQ_common_part_pred_q2 = 0
       FOR i = 0, 1, ... (PREDICTOR_ORDER - block_depth-1)
          TCQ_common_part_pred_q2=TCQ_common_part_pred_q2-TCQ_STTMP_q2
[(NFRSZ - 1) - i] * A[i+1+block_depth]
    | End of IF (d == 0).
| FOR each trellis state (node), predict the current sample.
AA0 = TCQ_common_part_pred_q2
FOR i = 0, 1, .., (block_depth - 1) | Do the next line
    AA0 = AA0 - TCQ_predictor_state_q2[d][i] * A[i+1]

AA0 = rnd_int (AA0<<2) | Q2 representation.

TCQ_predict_sample_q2[d] = AA0  | save predicted value.
TCQ_resid[d] = sample - AA0

| Normalize the residuals.

AA0 = TCQ_resid[d] * DLQ_inv_GAIN
AA1 = -1*(DLQ_inv_nls_m-1)
IF (0 <= AA1)
    AA0 = AA0 >> AA1
ELSE
    AA1 = -1*AA1
    AA0 = AA0 << AA1

| Force saturation.

IF (32767L < AA0)
    AA0 = 32767L

IF (AA0 < -32768L)
    AA0 = -32768L
TCQ_resid [d] = AA0 | Save the normalized residual.

END#J.8:
```

J.4.3.6 Bloque #J.9 – Encontrar el nuevo ('new') superviviente

Entrada: block_depth, next_stage, TCQ_resid

Salida: distortion_metric, Q_resid, prev_node, TCQ_channel_indices

Operación: Para cada estado reticular (nodo) 0, 1, 2, 3 (de la figura J.3) seleccionar una de las dos ramas entrantes y marcarla como superviviente 'new' para ese estado reticular (nodo).

NOTA – Cada nodo de estado siguiente se asocia con un solo superviviente (una de las dos ramas entrantes). Por tanto, es preferible asociar el superviviente 'new' con un nodo de estado siguiente y referenciarlo por medio del puntero next_stage.

El consumo, en complejidad (millones de instrucciones por segundo), de este bloque es considerable.

```
| For each trellis state, quantize the associated residuals, using the
| subset that labels the two emanating branches (of Figure J.3).

| Perform 8 Quantizations. CALL BLOCK #J.11 8 times.
TCQ_quantize_resid ( TCQ_resid [0], 0, &ch_index[0], &qerror[0] )
TCQ_quantize_resid ( TCQ_resid [0], 2, &ch_index[1], &qerror[1] )
TCQ_quantize_resid ( TCQ_resid [1], 1, &ch_index[2], &qerror[2] )
TCQ_quantize_resid ( TCQ_resid [1], 3, &ch_index[3], &qerror[3] )
TCQ_quantize_resid ( TCQ_resid [2], 2, &ch_index[4], &qerror[4] )
TCQ_quantize_resid ( TCQ_resid [2], 0, &ch_index[5], &qerror[5] )
TCQ_quantize_resid ( TCQ_resid [3], 3, &ch_index[6], &qerror[6] )
TCQ_quantize_resid ( TCQ_resid [3], 1, &ch_index[7], &qerror[7] )

| Calculate the distortion metric associated with each node.

AA0 = (TCQ_resid[0] - qerror[0]) * (TCQ_resid[0] - qerror[0])
AA0 = AA0 >> 2
temp_metric[0] = distortion_metric[0] + AA0

AA0 = (TCQ_resid[0] - qerror[1]) * (TCQ_resid [0] - qerror[1])
AA0 = AA0 >> 2
temp_metric[1] = distortion_metric[0] + AA0

AA0 = (TCQ_resid[1] - qerror[2]) * (TCQ_resid[1] - qerror[2])
AA0 = AA0 >> 2
temp_metric[2] = distortion_metric[1] + AA0

AA0 = (TCQ_resid[1] - qerror[3]) * (TCQ_resid[1] - qerror[3])
AA0 = AA0 >> 2
temp_metric[3] = distortion_metric[1] + AA0

AA0 = (TCQ_resid[2] - qerror[4]) * (TCQ_resid[2] - qerror[4])
AA0 = AA0 >> 2
temp_metric[4] = distortion_metric[2] + AA0

AA0 = (TCQ_resid[2] - qerror[5]) * (TCQ_resid[2] - qerror[5])
AA0 = AA0 >> 2
temp_metric[5] = distortion_metric[2] + AA0

AA0 = (TCQ_resid[3] - qerror[6]) * (TCQ_resid[3] - qerror[6])
AA0 = AA0 >> 2
temp_metric[6] = distortion_metric[3] + AA0

AA0 = (TCQ_resid[3] - qerror[7]) * (TCQ_resid[3] - qerror[7])
AA0 = AA0 >> 2
temp_metric[7] = distortion_metric[3] + AA0

| Select the new survivor for each node.

| node 0

IF (temp_metric[0] < temp_metric[4])
  distortion_metric[0] = temp_metric[0]
  Q_resid[0] = qerror[0]
  prev_node [next_stage][0] = 0
  TCQ_channel_indices [next_stage] [0] = ch_index[0]
```

```
ELSE
  distortion_metric[0] = temp_metric[4]
  Q_resid[0] = qerror[4]
  prev_node [next_stage][0] = 2
  TCQ_channel_indices [next_stage] [0] = ch_index[4]

| node 1

IF (temp_metric[1] < temp_metric[5])
  distortion_metric[1] = temp_metric[1]
  Q_resid[1] = qerror[1]
  prev_node [next_stage][1] = 0
  TCQ_channel_indices [next_stage] [1] = ch_index[1] OR 0x10
ELSE
  distortion_metric[1] = temp_metric[5]
  Q_resid[1] = qerror[5]
  prev_node [next_stage][1] = 2
  TCQ_channel_indices [next_stage] [1] = ch_index[5] OR 0x10

| node 2

IF (temp_metric[2] < temp_metric[6])
  distortion_metric[2] = temp_metric[2]
  Q_resid[2] = qerror[2]
  prev_node [next_stage][2] = 1
  TCQ_channel_indices [next_stage] [2] = ch_index[2]
ELSE
  distortion_metric[2] = temp_metric[6]
  Q_resid[2] = qerror[6]
  prev_node [next_stage][2] = 3
  TCQ_channel_indices [next_stage] [2] = ch_index[6]

| node 3

IF (temp_metric[3] < temp_metric[7])
  distortion_metric[3] = temp_metric[3]
  Q_resid[3] = qerror[3]
  prev_node [next_stage][3] = 1
  TCQ_channel_indices [next_stage] [3] = ch_index[3] OR 0x10
ELSE
  distortion_metric[3] = temp_metric[7]
  Q_resid[3] = qerror[7]
  prev_node [next_stage][3] = 3
  TCQ_channel_indices [next_stage] [3] = ch_index[7] OR 0x10
```

J.4.3.7 Bloque #J.11 – TCQ_quantize_resid

Entrada: resid, state

Salida: index, qerror

Operación: Cuantificación de un solo residuo utilizando un solo subconjunto.

NOTA – Esta rutina se efectúa 8 veces por muestra; cada cuantificación utiliza 4 bits y tiene 16 niveles de cuantificación.

```
index = 0
while ( (Xk[state][index] < resid) && (index < Q_CELLS) )| Do the next line
   index = index+1
*qerror = Yk[state][index] | return quantized residual.
```

J.4.3.8 Bloque #J.10 – Calcular la señal reconstruida

Entrada: next_stage DLQ_GAIN, DLQ_nls_p_cb_q_m_18, Q_resid, prev_node

Salida: TCQ_reconstructed

Operación: Para cada nodo 0, 1, 2, 3 (de la figura J.3), calcular el nivel reconstruido.

NOTA – El nivel reconstruido es referenciado por next_stage. Véase la nota a J.4.3.6.

Variables internas (definición según el lenguaje de programación C)

- int src;
- long int AA0;
- short int nls;

```
       FOR d = 0, 1, ..., (N_STATES - 1)
| For each trellis state (node) presented by the variable d.
| Do the next blocks till END #J.10

src = prev_node [next_stage] [d] | find the previous node.

TCQ_ET [ next_stage ] [ d ] = Q_resid [ d ]

AA0 = Q_resid[d] * DLQ_GAIN

IF (0 <= DLQ_nls_p_cb_q_m_18 )
       AA0 <<= DLQ_nls_p_cb_q_m_18
    ELSE
       AA0 >>= abs (DLQ_nls_p_cb_q_m_18)
    AA0 = AA0 + (TCQ_predict_sample_q2 [src] << 16L)
    TCQ_reconstructed_q2 [next_stage] [d] = AA0 >> 16L

END #J.10:
```

J.4.3.9 Bloque #J.4 – Seleccionar el mejor superviviente

Entrada: distortion_metric, prev_node, TCQ_ET, TCQ_reconstructed_q2, TCQ_channel_indices

Salida: TCQ_channel_symbols, ET, TCQ_STTMP_q2

Operación: Seleccionar el nodo superviviente y la secuencia asociada de símbolos de canal.

NOTA – Este módulo se efectúa después de haberse efectuado cinco pasos completos a través del trayecto reticular. Se deberá seleccionar (como el nodo superviviente) una secuencia de cinco símbolos de canal (5 bits por muestra).

Variables internas (definición según el lenguaje de programación C)

- int src;
- int survivor_node;
- long int min_dist;

```
| shift TCQ_STTMP_q2 Kr samples backward.
    FOR i = TCQ_Kr.. (NFRSZ-1)
       TCQ_STTMP_q2 [i - TCQ_Kr] = TCQ_STTMP_q2[i]
| shift ET backward.
FOR i = TCQ_Kr..(RMS_BUF_LEN-1)
    ET [i - TCQ_Kr] = ET [i]

| Select the node with a minimum quantization distortion, and the
| associated sequence of reconstructed samples.
```

```
min_dist = MAX_NUMBER
FOR d= 0..(N_STATES-1)
    IF (distortion_metric[d] <= min_dist) | Do the next 2 lines
       min_dist = distortion_metric[d]
       survivor_node= d

| Copy the last Kr samples from the
| survivor sequence into TCQ_STTMP_q2

k = 0 | TCQ_Kr == TCQ_Kd as in Kd/Kr=5/5.
src = survivor_node

FOR i = TCQ_DEPTH..(TCQ_Kd - TCQ_Kr)  | Do the next lines
| i is decremented.
| Arrange reproductions in ascending order.

    TCQ_STTMP_q2 [i + (NFRSZ-(TCQ_DEPTH+1))]=TCQ_reconstructed_q2 [k] [src]
    ET[ i - (TCQ_Kd - TCQ_Kr) + (RMS_BUF_LEN - TCQ_Kr) ] = TCQ_ET [k] [src]
    TCQ_channel_symbols [ i - (TCQ_Kd - TCQ_Kr) ] =TCQ_channel_indices [ k ] [
src ]
    src = prev_node [k] [src]

    k = k -1
    IF (k < 0)
       k = TCQ_DEPTH
| End of FOR.

| Estimate the gain for the next 5 samples.

CALL BLOCK #J.12 | TCQ Backward Gain Adapter.
```

J.4.3.10 Bloque #J.12 – Adaptador de ganancia hacia atrás TCQ

Entrada: next_stage

Salida: DLQ_GAIN, DLQ_NLSGAIN

Operación: Un cálculo, efectuado una vez por vector, de la ganancia de escalaje s(n).

NOTA – Este bloque sustituye el bloque #20 (del anexo G/G.728) en el modo datos.

Variables internas (definición según el lenguaje de programación C)

- int src;
- int survivor_node;
- long int min_dist;

```
| Calculate the log gain RMS value for the last selected path segment.

CALL BLOCK #J.13 | vbd_log_calc_and_lim97

| Calculate the dms and dml DLQ decision parameters.

dms = (dms << 5) - dms
dml = (dml << 7) - dml
dms += RMS_Q11
dml += RMS_Q11
dms >>= 5
dml >>= 7
diff = labs(dms-dml)
```

```
IF (diff >= (dml >> 3L))
   | Average Power is variable, non stationary signals such
   | as speech. ap -> 2, and unlock the quantizer.

   ap_q11 = (ap_q11 * 15 ) >> 4 | ap = ap * 15.0 /16.0
   ap_q11 += (1 Q11 ) >> 3
ELSE
   IF (DLQ_GAIN >> DLQ_NLSGAIN < 10)
      | Idle Channel Conditions, ap -> 2, and unlock the quantizer.

      ap_q11 = (ap_q11 * 15 ) >> 4
      ap_q11 += (1 Q11 ) >> 3
   ELSE
      | Average Power is constant, stationary signals, such as VBD
      | ap -> 2, and lock the quantizer.

      ap_q11 = (ap_q11 * 15 ) >> 4

IF (1 Q11 < ap_q11)
   al_q11 = 1 Q11
ELSE
   al_q11 = ap_q11

FOR i = 0..NUPDATE | save GSTATE as in Annex G/G.728.
   GTMP[i] = GSTATE[i]
| Predict the gain.

| BLOCK #46, P. 39 of Annex G of Recommendation G.728 is the
| LOG-GAIN LINEAR PREDICTOR.

CALL BLOCK #G.46 | Log-gain linear prediction.
CALL BLOCK #G.98 | log gain limiter 98, part of blocks 46 AND 47, P. 39.
CALL BLOCK #J.14 | log_gain_weighting

GAIN = after_limiter_98 | GAIN is the averaged log-gain.

CALL BLOCK #G.99 | add_log_offset, part of blocks 46 AND 47, P. 39.
CALL BLOCK #G.48 | Inverse logarithmic Calculator.

DLQ_GAIN = GAIN
DLQ_NLSGAIN = NLSGAIN
CALL BLOCK #J.15 | GAIN_inverse
```

J.4.3.11 Bloque #J.13 – vbd_log_calc_and_lim97

Entrada: ET

Salida: GSTATE[0], RMS_Q11

Operación: Calcular el valor cuadrático medio (RMS) de la ganancia logarítmica.

NOTA – Este bloque sustituye a los bloques 67, 39, y 40 (retardo de 1 vector, calculador RMS y calculador logarítmico) en la versión de coma flotante, y a los bloques 93, 94, 96, 97 del anexo G/G.728 en la versión de coma fija.

Variables internas (definición según el lenguaje de programación C)

- long int AA0;
- long int AA1;

```
AA0 = 0
FOR i=(RMS_BUF_LEN - 1), (RMS_BUF_LEN - 2),..., 1
   AA0 += ET [i] * ET [i]
AA0 += ET[0] * ET[0]
```

```
| Divide by RMS_BUF_LEN, and scale AA0 from Q22 to Q11.
AA0 = (AA0 + 1 Q13) >> 14

RMS_Q11 = AA0 | 32 bit operation.

AA0 = calc_10log10(AA0) | BLOCK #J.16 is implemented as a function.

AA0 = AA0 + (after_limiter_98 << 2) | 32 bit operation.

AA0 = AA0 >> 2 | scale to Q9 format.

| block 97 limit GSTATE to -32dB.

IF (AA0 < -16384)
   AA0 = -16384

| If immediately after transition, than use the GSTATE of the speech mode.
IF (speech_to_vbd_transition == 1)
   speech_to_vbd_transition = 0
ELSE
   GSTATE[0]=AA0
```

J.4.3.12 Bloque #J.25 – Calculador de la ganancia media

Entrada: GSTATE[0], UNSTEADY

Salida: GDIFF, G_CNT

Operación: Cálculo de un valor casi medio de la ganancia.

Variables internas (definición según el lenguaje de programación C)

- long int G_AVE

```
GDIFF=GSTATE[0]-G_AVE;
IF UNSTEADY=1           | Do the next 3 lines
    G_AVE=GSTATE[0]
    G_CNT=0
    UNSTEADY=0
ELSE                    | Do the next 3 lines
    IF GDIFF<G_TRS | Do the next 2 lines
       G_AVE=((G_AVE<<G_CONST-G_AVE+GSTATE[0])>>G_CONST
       G_CNT++
```

J.4.3.13 Bloque #J.26 – Clasificador de señales

Entradas: ATMP

Salidas: GC_SC_FLAG

Operación: Detección de señal de banda estrecha.

Variables internas (definición según el lenguaje de programación C)

- int GC_ATMP_SUM
- int GC_ATMP1

```
GC_ATMP_SUM=0
GC_ATMP1=ATMP[1]
FOR I=2,3,...LPC+1,           |Do the next line
    GC_ATMP_SUM=GC_ATMP_SUM+ABS(ATMP[I])

IF ((GC_ATMP_SUM*ATMP_CONST)>>3)<ABS(GC_ATMP1)
     GC_SC_FLAG=1
ELSE
     GC_SC_FLAG=0
```

J.4.3.14 Bloque #J.27 – Detección de impulso

Entradas: GDIFF, GC_LEN, GC_SC_FLAG

Salidas: GC_ID_FLAG

Operación: Búsqueda de un aumento brusco de la ganancia después de un periodo de ganancia estable.

```
IF GDIFF > G_TRS              | Do the next 4 lines
    IF G_CNT>GC_LEN
       GC_ID_FLAG=1
    ELSE
       GC_ID_FLAG=0
```

J.4.3.15 Bloque #J.28 – Decisión de compensación de ganancia

Entradas: GC_ID_FLAG, GC_SC_FLAG

Salidas: GC_FLAG, GC_CNT, UNSTEADY, GC_NLS_LIMIT, GC_COMPENSATION

Operación: Lógica de decisión para el bloque de compensación de ganancia.

```
GC_LEN=0
GC_NLS_LIMIT=16383
GC_COMPENSATION=0
IF GC_SC_FLAG=1         | Do the next 3 lines
    GC_LEN=GC_CNT_INIT
    GC_NLS_LIMIT=GC_NLS_LIMIT_INIT
    GC_COMPENSATION=GC_COMPENSATION_INIT
IF GC_ID_FLAG=1         | Do the next 3 lines
    UNSTEADY=1
    GC_FLAG=1           | GAIN COMPENSATION FLAG
    GC_CNT=GC_LEN
```

J.4.3.16 Bloque #J.29 – Compensación de ganancia

Entradas: GC_FLAG , DLQ_NLSGAIN, GG_CNT, GC_COMPENSATION, GC_NLS_LIMIT

Salidas: GC_FLAG

Operación: Decrementar DLQ_NLSGAIN por un valor fijo durante un periodo de tiempo predefinido.

```
IF GC_FLAG=1            | Do the next 7 lines
    IF DLQ_NLSGAIN>GC_NLS_LIMIT      | Do the next 6 lines
       GC_CNT=GC_CNT-1
       DLQ_NLSGAIN=DLQ_NLSGAIN-GC_COMPENSATION
       IF DLQ_NLSGAIN< GC_NLS_LIMIT
          DLQ_NLSGAIN= GC_NLS_LIMIT
       IF GC_CNT=0
          GC_FLAG=0
```

J.4.3.17 Bloque #J.16 – Calculador logarítmico

Entrada: Formato AA0. Q11. (Definición en lenguaje C: long int AA0)

Salida: Valor logarítmico

Operación: Aproximación de la función logarítmica.

NOTA – La aproximación es válida para $1 \leq x < 2$.

Variables internas (definición en lenguaje C)

- long int T;
- short int exp;

```
IF (AA0 < 1)
AA0 = 0 | illegal input number clipping to 0dB.
ELSE
| Scale the input number to the range 1Q14..2Q14
exp = 3 | Q3 is the difference Q14 - Q11
WHILE ( AA0 < 1 Q14 ) | Do the next two lines
AA0 = (AA0 << 1)
exp -= 1

WHILE (2 Q14 <= AA0) | Do the next two lines
AA0 = (AA0 >> 1L)
exp += 1

T = AA0 - 1 Q14
T <<= 1 | Translate T from Q14 to Q15
AA0 = log_pol[LOG_POL_ORDER-1]

FOR i = (LOG_POL_ORDER-1), (LOG_POL_ORDER-2).., 1
AA0 = ((T * AA0) * 2 + (log_pol[i-1] << 16) + 1 Q15) >> 16

AA0 = ( (AA0 * T) * 2 + 1 Q15) >> 16

AA0 = AA0 >> 4 | Translate AA0 to Q11.

AA0 = AA0 * 5 | Divide AA0 by 2 and multiply by 10 to get dB units.

| Add the mantisa, and calculate the 10 log dB value of the input

AA0 = AA0 + (log_2 * exp) >> 2

return(AA0)
```

J.4.3.18 Block #J.14 – log_gain_weighting

Entrada: after_limiter_98, GAIN_state, al_q11

Salida: after_limiter_98

Operación: Hallar el valor medio de la ganancia logarítmica y producir la velocidad de adaptación variable.

Variables internas (definición en lenguaje C)

- long int AA0, AA1 | 32 bit accumulators.
- short int locked, unlocked;

```
AA0 = (long)GAIN_state
AA1 = AA0 << 6L | 63/64 iir.
AA1 = AA1 - AA0

| Add AA1 (Q9 format) to after_limiter_98 and round the result to 16 bit Q9.

AA0 = after_limiter_98
AA1 = AA1 + AA0

AA0 = AA1 >> 6L | 63/64 iir.

locked = AA0
unlocked = after_limiter_98
```

```
AA0 = locked * ( 1 Q11 - al_q11)
AA0 = AA0 + unlocked * al_q11
AA0 >>= 11

after_limiter_98 = AA0
GAIN_state = AA0
```

J.4.3.19 Bloque #J.15 – GAIN_inverse

Entrada: DLQ_GAIN, DLQ_NLSGAIN

Salida: DLQ_inv_GAIN, DLQ_nls_p_cb_q_m_18, DLQ_inv_nls_m

Operación: Inversión de la ganancia.

Variables internas (definición en lenguaje C)

- long int NUM | constant 16384: 1 in the format Q14
- long int NUMNLS | constant 14

```
DLQ_nls_p_cb_q_m_18 = 18 - CODEBOOK_Q - DLQ_NLSGAIN |

| Initialize the numerator for inversion.

NUM  = 16384
NUMNLS = 14

divide ( NUM, NUMNLS, DLQ_GAIN, DLQ_NLSGAIN, &DLQ_inv_GAIN, &inv_nls) | The
divide is function Annex G/G.728.

DLQ_inv_nls_m = - 2 - inv_nls + CODEBOOK_Q + 1
```

J.4.3.20 Block #J.5 – Módulo de adaptación

Operación: Efectuar el ciclo de adaptación.

NOTA – El mismo ciclo de adaptación se utiliza para datos y conversación y, antes de cada fase de adaptación, sólo es necesario traducir algunas variables al formato de adaptación de entrada del anexo G/G.728.

Variables internas (definición en lenguaje C)

- long int nls
- short int min_nls;

```
ICOUNT = ICOUNT + 1
IF (ICOUNT > NUPDATE)
    ICOUNT = 1

IF (ICOUNT == 4)
   FOR k = 0, 1, .., (NUPDATE - 1)
      | Prepare STTMP & NLSSTTMP for autocore routine,-HW_s.

      VSCALE( &TCQ_STTMP_q2[k * IDIM], IDIM,
               IDIM, 12, &STTMP[k * IDIM], &NLSSTTMP[k])
      NLSSTTMP[k] = NLSSTTMP[k] + 2 | Q2 correction.

   CALL BLOCK #G.49 | HW_s

IF (ICOUNT == 2)
   IF (ILLCOND == 0)
      durbin (RTMP, ATMP, 10) | BLOCK #G.50
```

```
      IF (DurbinFaultFlag == 0)
         CALL BLOCK #J.26 | signal classifier
         CALL BLOCK #J.51 | bandexpand51_vbd()
      ELSE
         DurbinFaultFlag = 0
         FOR I=1, 2,..LPC
            ATMP[i]=A[i]

IF (ICOUNT == 1)
   CALL BLOCK #G.43 | HW_gain
   IF ( ILLCOND == 0 )
      durbin(R, GPTMP, LPCLG) | BLOCK #G.44
   IF (DurbinFaultFlag == 0)
      CALL BLOCK #G.45 | bandexpand45
   ELSE
      DurbinFaultFlag = 0

IF (ICOUNT == 3)
   FOR i = 1, 2.., PREDICTOR_ORDER
      A[i] = ATMP[i]
```

J.4.3.21 Bloque #J.51 – Módulo de expansión de la anchura de banda

Entrada: Véase el bloque #G.51

Salida: Véase el bloque #G.51

Operación: Véase el bloque #G.51.

NOTA – Este bloque es casi igual al bloque #G.51, del que sólo se diferencia en que la matriz FACV_vbd reemplaza a FACV, que tiene 10 elementos diferentes de cero.

J.4.3.22 Bloque #J.17 – Módulo de inicialización de la búsqueda siguiente

Entrada:survivor_node

Salida:distortion_metric

Operación: Establecer la métrica para una nueva búsqueda.

```
FOR i = 0, 1, ..,(N_STATES-1)
    distortion_metric[i] = (MAX_NUMBER >> 2)

| Force the next selected path to pass through the survivor node.
distortion_metric[survivor_node] = 0L

block_depth = 0
next_stage = 1
```

J.4.4 Descripción detallada del decodificador

En esta sección se presenta una descripción detallada del decodificador.

J.4.4.1 Bloque #J.20 – Módulo del decodificador

Entrada: channel_symbol

Salida: reconstructed_sample_q2

Operación: Realizar las operaciones de decodificación vector por vector.

```
For I = 1, 2, ..., IDIM | Do the next line
    CALL BLOCK #J.21 | Decoder transition module.

CALL BLOCK #J.12 | TCQ Backward Gain Adapter
CALL BLOCK #J.5 | Adaptation Module.
```

J.4.4.2 Block #J.21 – Módulo de transición del decodificador

Entrada: channel_symbol

Salida: reconstructed_sample_q2

Operación: Efectuar las operaciones de decodificador muestra por muestra.

NOTA – La predicción se efectúa sobre el valor predicho por el codificador para el nodo 0.

Variables internas (definición en lenguaje C)

- short int next_state_label;
- short int level_selector;
- short int subset;
- short int next_state;
- short int codebook_level;
- long int AA0;

```
TCQ_predict_sample_q2[0] = 0 | Use the Encoder's node 0 space.

FOR i = 0, 1,..,(PREDICTOR_ORDER-1)
    TCQ_predict_sample_q2[0] -= TCQ_STTMP_q2 [(NFRSZ - 1) - i] * A[i+1]

FOR i = 0, 1,..,(NFRSZ-1)
    TCQ_STTMP_q2[i] = TCQ_STTMP_q2[i + 1]

TCQ_predict_sample_q2[0] =
    rnd_int ( TCQ_predict_sample_q2[0] << 2)

| Separate the index into 2 sets the next state label and
| the level selector.

next_state_label = (channel_symbol >> BITS_PER_LEVEL)
level_selector = (channel_symbol & LEVEL_MASK)
next_state = TCQ_next_state [TCQ_decoder_state] [next_state_label]
subset = TCQ_trans_from_src_to_dst [TCQ_decoder_state] [next_state]
TCQ_decoder_state = next_state

codebook_level = Yk[subset][level_selector] | Get the codebook level

FOR i = 1, 2, ..,(RMS_BUF_LEN - 1)
    ET[i-1] = ET[i]

ET[RMS_BUF_LEN - 1] = codebook_level

AA0 = codebook_level * DLQ_GAIN

IF (0 <= DLQ_nls_p_cb_q_m_18)
    AA0 <<= DLQ_nls_p_cb_q_m_18
ELSE
    AA0 >>= abs (DLQ_nls_p_cb_q_m_18)

AA0 += (TCQ_predict_sample_q2[0] << 16L)
TCQ_STTMP_q2 [(NFRSZ-1)] = AA0 >> 16L
reconstructed_sample_q2 = AA0 >> 16L
```

J.4.5 Descripción detallada de los módulos de conmutación de modo

En esta sección se presenta una descripción detallada de los módulos de conmutación de modo.

J.4.5.1 Bloque #J.18 – Módulo de transición de conversación a datos

Entrada: NLSSTATE, STATELPC, GAIN, NLSGAIN, IAQ_for_VBD, ATMP_for_VBD

Salida: TCQ_STTMP_q2, DLQ_GAIN, DLQ_NLSGAIN, IAQ_for_VBD, ATMP_for_VBD

Operación: Efectuar la transición de conversación a datos.

```
| Translate 20 elements in SBFL format of STATELPC and NLSSTATE
| to the Q2 format of TCQ_STTMP_q2.
| Note that the opposite ordering of STATELPC, NLSSTATE and
| TCQ_STTMP_q2.

I = 0
FOR J = 0, 1, 2, 3
    FOR L = 0, 1,..,4
       k = NLSSTATE [9] - 2
       IF (k<0)
          k = -k
          TCQ_STTMP_q2[NFRSZ - 1 - I] = STATELPC [I] << k
       ELSE
          TCQ_STTMP_q2[NFRSZ - 1 - I] = STATELPC [I] >> k
       I = I + 1

| GAIN VARIABLES.

speech_to_vbd_transition = 1

GAIN_state = after_limiter_98

dms = 0
dml = 0
ap_q11 = 0

DLQ_GAIN = GAIN
DLQ_NLSGAIN = NLSGAIN

CALL BLOCK #J.15 | GAIN_inverse()

| Use the previously saved 10 LPC parameters (saved during last
| adaptation cycle of speech-mode).

FOR I = 1,2, .., PREDICTOR_ORDER
    ATMP[i] = ATMP_for_VBD[i]

IAQ = IAQ_for_VBD
CALL BLOCK #J.51 | bandexpand51_vbd

FOR I = 1, 2,.., PREDICTOR_ORDER
    A[i] = ATMP[i]

GC_FLAG=0;    | Gain Compensation
G_CNT=0;      | Gain Compensation
```

J.4.5.2 Bloque #J.19 – Módulo de transición de datos a conversación

Entrada: Véase la descripción más adelante

Salida: Véase la descripción más adelante

Operación: Efectuar la transición de datos a conversación.

Variables internas (definición en lenguaje C)

- long int temp[NFRSZ]

```
| Reinitialize the perceptual weighting internal variables.
| (Only in the Encoder).

AWP[0]=16384
FOR I=1, 2, ..,LPCW
    AWP[i]=0
AWZ[0]=16384
FOR I = 1, 2, ..,LPCW
    AWZ[i]=0
FOR I = 0, 1, 2, ..(NFRSZ-1)
    STMP[i]=0
FOR I = 0, 1,..,(N3weight-1)
    SBW[i]=0

FOR I = 0, 1, ..,LPCW
    REXPW[i]=0
NLSREXPW= 31

| Reinitialize the post filter internal variables.
| (Only in the decoder).

ILLCONDP = 1
AP[0] = 16384
FOR I =1, 2, ..,10
    AP[i]=0
AZ[0] = 16384
FOR I =1, 2,..,10
    AZ[i]=0
FOR I = 0,1,..,59
    DEC[i]=0
FOR I = 0, 1,..,239
    D[i]=0
FOR I = 0, 1,..,9
    STLPCI[i]=0
FOR I = 0, 1, ..,9
    STPFFIR[i]=STPFIIR[i]=0
FOR I =0, 1, 2, 3
    LPFFIR[i]=0
    LPFIIR[i]=0
FOR I = 0, 1, .., 244
    SST[i]=0
SCALEFIL= 16384
B=0
GL=16384
GLB = 0
TILTZ=0
APF[0] = 16384
FOR I = 1, 2, ..,10
    APF[i]=0
PF_delay = 100
KP=KP1=50
```

```
FOR I =11, 12, .., LPC
    A[i]=0
    ATMP[i]=0

FOR I = 0, 1, ..,(NFRSZ-1)
    temp[i] = TCQ_STTMP_q2[(NFRSZ-1) - i]

FOR I = 0, 1, 2, 3
    VSCALE( &temp[i * 5], 5, 5, 12,&STATELPC[i * 5], &NLSSTATE[9-i] )
    NLSSTATE[9-i] += 2 | Q2 Correction.

FOR I = NFRSZ, NFRSZ+1, .., (LPC - 1)
    STATELPC [i] = 0
FOR I = 0, 1, .., 5
    NLSSTATE [i] = 16

| Update several LD-CELP internal blocks.
| These blocks are usually updated during the third adaptation phase
| ( ICOUNT == 3), but they are needed from phase 1.

CALL BLOCK #G.12 | Impulse response vector calculation.
CALL BLOCK #G.14 | Shape codevector convolution and
                   | energy calculation.

| GAIN VARIABLES.

CALL BLOCK #J.13 | vbd_log_calc_and_lim97()
vbd_to_speech_transition = 1
```

J.4.5.3 Block #J.22 – Modificaciones del adaptador de ganancia vectorial hacia atrás del anexo G/G.728

Operación: Las siguientes líneas sustituyen los bloques #G.93, #G.94, #G.96 y #G.97

NOTA – Durante la transición de datos a conversación, los valores de retardo de un solo índice del índice del libro de códigos de ganancia y de forma no están disponibles. Por tanto, GSTATE[0] se calcula en el curso de la transición.

```
IF ( vbd_to_speech_transition == 1)
    vbd_to_speech_transition = 0
ELSE
    | default operation of Annex G/G.728.
    | Perform blocks #G.93, #G.94, #G.96 and #G.97.
```

J.4.5.4 Bloque #J.23 – Modificaciones relativas a la aplicación del posfiltro del anexo G/G.728

Operación: Contornear el posfiltro durante las primeras 500 muestras (62,5 ms) después de la transición, y utilizar la señal reconstruida como la salida del decodificador durante ese periodo. Transcurridos 62,5 ms, cuando el posfiltro está listo para funcionar, su salida se utiliza como la salida del decodificador.

```
IF (PF_delay == 0)
  FOR I =0, 1, .., (IDIM-1)
    ST[i]=SPF[i]<<1
ELSE
  PF_delay--
  FOR I = 0, 1, ..,(IDIM-1)
    ST[i]=ST[i]<<(16-NLSST+3)
    ST[i]=rnd_int(ST[i])
```

J.4.5.5 Bloque #J.24 – Modificaciones que deben introducirse en el anexo G/G.728 para la salvaguarda de parámetros LPC provisionales

Operación: Salvaguardar los 10 parámetros LPC provisionales. Los parámetros LPC se necesitan para la transición de conversación a datos.

NOTA – Los 10 parámetros LPC provisionales deben ser salvaguardados durante la ejecución del bloque #G.23. Se deberá detener la ejecución del bloque durbin (como se describe en el anexo G/G.728 para los coeficientes del posfiltro, APF). Los coeficientes deben ser salvaguardados, después de lo cual se deberá reanudar el bloque durbin.

```
IF (DurbinFaultFlag == 0)
    FOR I =1, 2,.., 10
       ATMP_for_VBD[I]=ATMP[I];
    IAQ_for_VBD = IAQ;
```

J.4.6 Cuadros de transiciones reticulares

En esta sección se presentan los cuadros de transiciones reticulares. Estos cuadros describen una realización del codificador convolucional mínimo exento de retroalimentación.

J.4.6.1 TCQ_prev_state

Este cuadro proporciona el estado reticular (nodo) anterior (fuente), por rama reticular, para cada estado reticular (nodo), como se muestran en las figuras J.2 y J.3.

Definición en lenguaje C: int TCQ_prev_state[N_STATES][N_BRANCHES]

TCQ_prev_state		
Estado reticular (nodo)	**prev_state**	
	b[0]	b[1]
s[0]	0	2
s[1]	2	0
s[2]	1	3
s[3]	3	1

J.4.6.2 TCQ_next_state

Este cuadro proporciona el estado reticular (nodo) siguiente (deseado), por rama reticular, para cada estado reticular (nodo), como se muestran en las figuras J.2 y J.3.

Definición en lenguaje C: int TCQ_next_state[N_STATES][N_BRANCHES];

TCQ_next_state		
Estado reticular (nodo)	**next_state**	
	b[0]	b[1]
s[0]	0	1
s[1]	2	3
s[2]	0	1
s[3]	2	3

J.4.6.3 TCQ_trans_from_src_to_dst

Este cuadro proporciona el subconjunto del libro de códigos asociado con una transición, como se presentan en las figuras J.2 y J.3.

Definición en lenguaje C: int TCQ_trans_from_src_to_dst [N_STATES][N_STATES];

TCQ_trans_from_src_to_dst				
Estado reticular (nodo)	**Etiqueta de transición**			
	s[0]	s[1]	s[2]	s[3]
s[0]	0	2	X	X
s[1]	X	X	1	3
s[2]	2	0	X	X
s[3]	X	X	3	1

J.4.6.4 TCQ_index_from_src_to_dst

Este cuadro proporciona el bit de índice de canal que identifica la transición, como se presentan en las figuras J.2 y J.3, al decodificador.

Definición en lenguaje C: int TCQ_index_from_src_to_dst [N_STATES][N_STATES];

TCQ_index_from_src_to_dst				
Estado reticular (nodo)	**bit de índice de canal**			
	s[0]	s[1]	s[2]	s[3]
s[0]	0	0x10	X	X
s[1]	X	X	0	0x10
s[2]	0	0x10	X	X
s[3]	X	X	0	0x10

J.4.6.5 Xk – Límites del cuantificador

Este cuadro proporciona los límites de intervalo del libro de códigos (superior) en formato Q11. La primera columna presenta el índice de célula; las columnas siguientes presentan los niveles para cada subconjunto.

Definición en lenguaje C: int Xk [N_STATES][Q_CELLS];

Xk				
Índice	**Límites del libro de códigos**			
	s[0]	s[1]	s[2]	s[3]
0	–9 547	–8 509	–7 779	–7 191
1	–6 690	–6 246	–5 845	–5 477
2	–5 134	–4 812	–4 507	–4 217
3	–3 939	–3 672	–3 414	–3 164
4	–2 921	–2 684	–2 453	–2 226
5	–2 003	–1 783	–1 567	–1 353
6	–1 141	–931	–723	–515

Xk				
Índice	**Límites del libro de códigos**			
7	–309	–103	103	309
8	515	723	931	1 141
9	1 353	1 567	1 783	2 003
10	2 226	2 453	2 684	2 921
11	3 164	3 414	3 672	3 939
12	4 217	4 507	4 812	5 134
13	5 477	5 845	6 246	6 690
14	7 191	7 779	8 509	9 547
15	32 767	32 767	32 767	32 767

J.4.6.6 Yk – Niveles del cuantificador

Este cuadro proporciona los niveles de cuantificación del libro de códigos (superior) en formato Q11. La primera columna presenta el índice de célula; las columnas siguientes presentan los niveles para cada subconjunto.

Definición en lenguaje C: int Yk [N_STATES][Q_CELLS];

Yk				
Índice	**Niveles del libro de códigos**			
	s[0]	s[1]	s[2]	s[3]
0	–11 502	–9 955	–8 962	–8 209
1	–7 592	–7 062	–6 595	–6 174
2	–5 788	–5 430	–5 095	–4 780
3	–4 480	–4 194	–3 919	–3 655
4	–3 399	–3 151	–2 910	–2 674
5	–2 444	–2 218	–1 996	–1 777
6	–1 562	–1 348	–1 138	–928
7	–721	–514	–308	–103
8	103	308	514	721
9	928	1 138	1 348	1 562
10	1 777	1 996	2 218	2 444
11	2 674	2 910	3 151	3 399
12	3 655	3 919	4 194	4 480
13	4 780	5 095	5 430	5 788
14	6 174	6 595	7 062	7 592
15	8 209	8 962	9 955	11 502

J.4.7 Coeficientes del polinomio utilizado en el calculador logarítmico

Este cuadro presenta los coeficientes del polinomio utilizado en el calculador logarítmico.

Definición en lenguaje C: int log_poly[LOG_POL_ORDER];

log_poly		
Índice	**Presentación en coma flotante**	**Presentación en coma fija**
0	0,8678284	28 437
1	–0,4255677	–13 945
2	0,2481384	8 131
3	–0,1155701	–3 787
4	0,0272522	893

J.4.8 Coeficientes de expansión de la anchura de banda para el modo datos

Este cuadro proporciona los coeficientes de expansión de la anchura de banda para el modo datos.

NOTA – Sólo 10 elementos son diferentes de cero.

Definición en lenguaje C: int FACV_vbd[LPC];

Cuadro J.1/G.728 – Coeficientes de expansión de anchura de banda

FACV_vbd	
Índice	**Coeficiente**
1	15 360
2	14 400
3	13 500
4	12 656
5	11 865
6	11 124
7	10 428
8	9 777
9	9 166
10	8 593
11-50	0

J.4.9 Bloques internos

El cuadro J.2 contiene una lista de los bloques internos.

Cuadro J.2/G.728 – Bloques internos

ID de bloque	Nombre de bloque
Bloque #J.1	Selección de modo y de conmutación de modo
Bloque #J.2	Modo búsqueda de datos en la banda vocal
Bloque #J.3	Módulo de búsqueda reticular vector por vector
Bloque #J.4	Seleccionar el mejor superviviente
Bloque #J.5	Módulo de adaptación
Bloque #J.6	TCQ_transition
Bloque #J.7	Seleccionar estado del predictor
Bloque #J.8	Calcular residuos
Bloque #J.9	Encontrar el nuevo ('new') superviviente
Bloque #J.10	Calcular la señal reconstruida
Bloque #J.11	TCQ_quantize_resid
Bloque #J.12	Adaptador de ganancia hacia atrás TCQ
Bloque #J.13	vbd_log_calc_and_lim97
Bloque #J.14	log_gain_weighting
Bloque #J.15	GAIN_inverse
Bloque #J.16	Calculador logarítmico
Bloque #J.17	Módulo de inicialización de la búsqueda siguiente
Bloque #J.18	Módulo de transición de conversación a datos
Bloque #J.19	Módulo de transición de datos a conversación
Bloque #J.20	Módulo del decodificador
Bloque #J.21	Módulo de transición del decodificador
Bloque #J.22	Modificaciones del adaptador de ganancia vectorial hacia atrás del anexo G/G.728
Bloque #J.23	Modificaciones relativas a la aplicación del posfiltro del anexo G/G.728
Bloque #J.24	Modificaciones que deben introducirse en el anexo G/G.728 para la salvaguarda de parámetros LPC provisionales
Bloque #J.25	Calculador de la ganancia media para el módulo de compensación de ganancia
Bloque #J.26	Clasificador de señales para el módulo de compensación de ganancia
Bloque #J.27	Detección de impulso para el módulo de compensación de ganancia
Bloque #J.28	Bloque de decisión para el módulo de compensación de ganancia
Bloque #J.29	Compensación de ganancia para el módulo de compensación de ganancia
Bloque #J.51	Módulo de expansión de la anchura de banda

J.4.10 Variables y constantes utilizadas en el procesamiento interno

Los cuadros J.3 y J.4 proporcionan una lista y una descripción de las variables y constantes utilizadas en el procesamiento interno.

Cuadro J.3/G.728 – Variables utilizadas en el procesamiento interno

Nombre	Gama de los índices de array	Formato de coma fija	Descripción
after_limiter_98			Entrada Q9 de bloque sumador #G.99
al_q11	1	Q11	Factor de enganche del cuantificador
ap_q11	1	Q11	Factor de enganche del cuantificador
ATMP_for_VBD	1..11	Q13/Q14/Q15	Memoria intermedia para parámetros LPC
block_depth	1	Q0	Puntos en el paso reticular actual (tiempo n)
ch_index	0..7	Q0	Memoria temporal para el índice seleccionado
distortion_metric	[0..(N_STATES−1)]	32 bit Q20	Métrica de la distorsión acumulada por cada estado reticular (nodo)
DLQ_GAIN	1	SFL	Ganancia de la excitación lineal, equivalente a GAIN en el anexo G
DLQ_inv_GAIN	1	SBFL	GAIN invertida para VBD
DLQ_inv_nls_m	1	SBFL	NLS para la VBD GAIN invertida
DLQ_nls_p_cb_q_m_18	1	Q0	NLS para ganancia lineal (+ desplazamiento para corrección del formato Q) , equivalente a GAIN en el anexo G
dml	1	Q11	Energía a largo plazo de los residuos cuantificados
dms	1	Q11	Energía a corto plazo de los residuos cuantificados
ET	[0..(RMS_BUF_LEN−1)]		Memoria de residuos cuantificados
FACV_vbd	---	---	Véase FACV del anexo G
GAIN_state	1	Q9	Memoria de filtro de ponderación
IAQ_for_VBD	1	Q0	Bandera de la precisión de la recursión Durbin para ATMP
next_stage	1	Q0	Puntos en el paso reticular siguiente (tiempo n+1).
prev_node	[0..(BLOCK_LEN−1)]*[0..(N_STATES−1)]	Q0	Memoria reticular de estados (nodos) precedentes
Q_resid	[0..(N_STATES−1)]	Q11	Memoria temporal de los residuos cuantificados
qerror	1	Q11	Memoria temporal de residuos cuantificados
RMS_Q11	1	Q11	RMS de ET

Cuadro J.3/G.728 – Variables utilizadas en el procesamiento interno *(fin)*

Nombre	Gama de los índices de array	Formato de coma fija	Descripción
sample	1	Q2	Muestra de entrada, elemento individual de S
speech_to_vbd_transition	1	Q0	Bandera de transición de conversación a VBD
survivor_node	1	Q0	Índice de nodo superviviente
TCQ_channel_indices	[0..(BLOCK_LEN−1)] * [0..(N_STATES−1)]	Q0	Memoria reticular de índices de canal
TCQ_channel_symbols	[0..(IDIM−1)]	Q0	Memoria de señal comprimida
TCQ_common_part_pred_q2	1	Q2	Parte común del valor predicho
TCQ_ET	[0..(BLOCK_LEN−1)] [0..(N_STATES−1)]		Memoria reticular de residuos
TCQ_predict_sample_q2	[0..(N_STATES−1)	Q2	Memoria de valor predicho para cada estado reticular (nodo)
TCQ_predictor_state_q2	[0..(N_STATES−1)][0..(PREDICTOR_ORDER−1)]	Q2	Memoria reticular del estado del predictor
TCQ_reconstructed_q2	[0..(BLOCK_LEN−1)] [0..(N_STATES−1)]	Q2	Memoria reticular de niveles reconstruidos
TCQ_reconstructed_q2	[0..(BLOCK_LEN−1)]*[0..(N_STATES−1)]	Q2	Memoria reticular de muestras reconstruidas
TCQ_resid	[0..(N_STATES−1)]	Q11	Memoria de residuo (estado reticular) de nodos
TCQ_STTMP_q2	[0..NFRSZ]	Q2	Memoria intermedia para la ventana mixta del filtro de síntesis
temp_metric	[0..7]	32 bit Q20	Memoria temporal de las métricas de distorsión
vbd_to_speech_transition	1	Q0	Bandera de transición de VBD a conversación
Xk	[0..(N_STATES−1)] [0..(Q_CELLS−1)]	Q11	Límites de intervalos de cuantificación
Yk	[0..(N_STATES−1)] [0..(Q_CELLS−1)]	Q11	Niveles de cuantificación
GDIFF	1	32bit Q9	Diferencia entre la ganancia actual y su media
G_AVE	1	32bit Q9	Ganancia media
GC_ATMP_SUM	1		Suma de parámetros LPC absolutos del bloque #G.50
GC_ATMP1	1		Segundo parámetro LPC del bloque #G.50
GC_NLS_LIMIT	1	Q0	Umbral NLS de ganancia para uso en el bloque de compensación de ganancia
GC_COMPENSATION	1	Q0	Factor de compensación

Cuadro J.4/G.728 – Constantes utilizadas en el procedimiento interno

Nombre	Valor	Símbolo	Descripción
LOG_POL_ORDER	5		Orden del polinomio de aproximación logarítmica
BITS_PER_LEVEL	4		Número de bits que identifican niveles de cuantificación
LEVEL_MASK	0x0F		Máscara para bits de nivel
PREDICTOR_ORDER	10		Orden del predictor del modo datos
CODEBOOK_Q	1 Q11		Presentación Q del libro de códigos
Log_2	24 660		10*log10(2) en Q13
RMS_BUF_LEN	8		Longitud del cálculo de RMS
BLOCK_LEN	5	K_d	Longitud del bloque reticular
MAX_NUMBER	0x7fffffff		Número positivo máximo
N_BRANCHES	2		Número de ramas que parten de cada estado reticular o que llegan a cada estado reticular
N_STATES	4		Número de estados reticulares
Q_CELLS	16		Número de niveles de cuantificación en cada subconjunto
TCQ_Kd	5	K_d	Longitud del retardo reticular
TCQ_Kr	5	K_r	Rol de liberación reticular
TCQ_DEPTH	BLOCK_LEN−1		Longitud del bloque reticular – 1
G_CONST	5		Se utiliza para el cálculo de G_{ave} en el módulo de compensación de ganancia
ATMP_CONST	3		Umbral para la detección de señal de banda estrecha en el clasificador de señales
G_TRS	1 800		Umbral G_{diff} del compensador de ganancia
GC_NLS_LIMIT_INIT	7		Limitador del compensador de ganancia
GC_COMPENSATION_INIT	3		Valor sustraído de NLS de ganancia cuando se produce una compensación de ganancia
GC_CNT_INIT	11		Periodo de tiempo durante el cual está activa la compensación de ganancia

J.4.11 Valores iniciales

Cuadro J.5/G.728 – Valores iniciales

NOMBRE	Valor inicial
ATMP_for_VBD	16 384, 0, ..., 0
IAQ_for_VBD	14
vbd_to_speech_transition	0
block_depth	0
next_stage	1
TCQ_decoder_state	0
ET	0..0
GAIN	512
NLSGAIN	0
DLQ_GAIN	16 384
DLQ_NLSGAIN	14
DLQ_nls_p_cb_q_m_18	–7
DLQ_inv_GAIN	16 384
DLQ_inv_nls_m	–4
GAIN_state_fx	0L
after_limiter_98	–16 384
TCQ_STTMP_q2	0..0
TCQ_reconstructed_q2	0, ..., 0
distortion_metric	0, (MAX_NUMBER>>2), …, (MAX_NUMBER>>2)
dms	0
dml	0
RMS_Q11	0
ap_q11	0
al_q11	0
GAVE	0
UNSTEADY	1
G_CNT	0
GC_SC_FLAG	0
GC_ID_FLAG	0
GC_CNT	0
GC_FLAG	0

J.5 Bibliografía

(1) GERSHO (A.) y GRAY (R.M.), Vector quantization and signal compression.

(2) MARCELLIN (M.W.) y FISCHER (T.R.), Trellis-Coded Quantization of Memoryless and Gauss-Markov Sources , IEEE transactions on communications vol. 38, N.° 1, enero de 1990.

(3) TRUSHKIN (A.V.), On the design of an optimal Quantizer, IEEE transactions on information theory vol. 39, N.° 4, julio de 1993.

(4) JAYANT (N.S.) y NOLL (P.), Digital coding of waveforms.

(5) PETR (D.W.), 32 Kbps ADPCM-DLQ coding for network applications, IEEE 1982.

(6) MOC/Israel, Test results for non-voiced performance assessment of LD-CELP algorithm operating at 40 kbit/s mainly for DCME applications, SG 16 COM-D.278, Santiago, Chile, 18-27 de mayo de 1999.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	**Sistemas y medios de transmisión, sistemas y redes digitales**
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Transmisiones de señales radiofónicas, de televisión y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos y comunicación entre sistemas abiertos
Serie Y	Infraestructura mundial de la información y aspectos protocolo Internet
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación