



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

**UIT-T**

SECTEUR DE LA NORMALISATION  
DES TÉLÉCOMMUNICATIONS  
DE L'UIT

**G.728**

**Appendice I**  
**Outils de vérification**  
(07/95)

SÉRIE G: SYSTÈMES ET SUPPORTS DE  
TRANSMISSION, SYSTÈMES ET RÉSEAUX  
NUMÉRIQUES

Systèmes de transmission numériques – Equipements  
terminaux – Codage des signaux analogiques par des  
méthodes autres que la MIC

---

**Programme et séquences de test pour la  
vérification des implémentations de l'algorithme  
du vocodeur LD-CELP G.728 à 16 kbit/s**

Recommandation UIT-T G.728 – Appendice I

(Antérieurement Recommandation du CCITT)

---

## AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

La Conférence mondiale de normalisation des télécommunications (CMNT), qui se réunit tous les quatre ans, détermine les thèmes d'études à traiter par les Commissions d'études de l'UIT-T lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution n° 1 de la CMNT.

Dans certains secteurs de la technologie de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

## NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

## DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT avait été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

© UIT 1998

Droits de reproduction réservés. Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'UIT.

# **PROGRAMME ET SEQUENCES DE TEST POUR LA VERIFICATION DES IMPLEMENTATIONS DE L'ALGORITHME DU VOCODEUR LD-CELP G.728 A 16 kbit/s**

## **1 Généralités**

Le présent document décrit les séquences numériques de test et le logiciel de mesure à utiliser pour la vérification des implémentations de la Recommandation G.728. Les essais sont prévus pour les implémentations aussi bien en virgule flottante (sur la base du corps de la Recommandation G.728) qu'en virgule fixe en calculs exacts (sur la base de l'Annexe G/G.728).

### **1.1 Principe de vérification des implémentations en virgule fixe**

Le texte principal spécifiant l'algorithme LD-CELP n'est pas formulé sous forme de calculs exacts, de façon à l'implémenter de manière simple sur différents types de matériel. Cela implique que dans cette procédure, on ne peut pas faire l'hypothèse d'une égalité exacte entre l'implémentation sous test et une implémentation de référence quelconque. Des mesures objectives sont donc nécessaires afin de déterminer le degré de divergence entre le test et la référence. Si la divergence mesurée est suffisamment faible, l'implémentation sous test sera supposée être interopérable avec toute autre implémentation satisfaisant au test. Comme aucun essai de durée finie n'est en mesure de vérifier la totalité des caractéristiques d'une implémentation, on ne pourra jamais garantir une certitude totale du bon fonctionnement d'une implémentation. Cependant, la procédure de test décrite excite toutes les parties principales de l'algorithme LD-CELP, et devrait donc constituer un outil précieux pour le réalisateur.

Les procédures de vérification en virgule flottante décrites dans le présent document ont été conçues en fonction d'implémentations en virgule flottante sur 32 bits. Bien que les conditions de test soient applicables à toute implémentation du LD-CELP, le format 32 bits à virgule flottante sera sans doute requis pour y satisfaire.

### **1.2 Principes de vérification des implémentations en virgule fixe**

L'Annexe G/G.728 décrit au bit près l'algorithme LD-CELP en virgule fixe. Cela implique que, lorsque deux implémentations de codeur ou de décodeur sont lancées à partir de leur état initial avec des signaux d'entrée identiques, toutes les variables d'état doivent être exactement identiques à des instants équivalents au cours du traitement de ces signaux d'entrée. Par conséquent, on peut utiliser à l'entrée les séquences de test à virgule flottante, qui doivent donner des séquences de sortie exactes en calculs exacts.

Un court segment de signal vocal d'entrée a été retenu comme signal d'entrée de test additionnel. Toutes les variables d'état interne qui sont associées au cours du traitement de ce signal d'entrée sont également indiquées. Elles permettront au réalisateur de vérifier que tout le traitement effectué dans l'implémentation correspond exactement au traitement décrit dans la spécification. Pour qu'une implémentation puisse être considérée comme totalement conforme à l'Annexe G/G.728, il faut qu'elle soit en correspondance exacte avec les valeurs de sortie prescrites pour les signaux d'entrée pour le test en virgule flottante ainsi qu'avec les variables d'état pour le court segment de signal vocal.

Comme dans le cas de la procédure de vérification d'une implémentation à virgule flottante, on ne peut jamais garantir avec une certitude de 100% qu'une implémentation est correcte, parce que aucun test de longueur finie n'est en mesure de contrôler chaque aspect d'une implémentation. La procédure

de test décrite met cependant à contribution les principaux modules de l'algorithme LD-CELP, ce qui devrait en faire un outil intéressant pour les réalisateurs.

## 2 Configurations de test

Le présent paragraphe décrit la façon dont les différentes séquences de test et les différents programmes de mesurage devront être utilisés ensemble pour exécuter les tests de vérification. La procédure est fondée sur des tests du type "boîte noire" aux interfaces SU et ICHAN du codeur de test, ainsi qu'aux interfaces ICHAN et SPF du décodeur de test. Les signaux SU et SPF sont représentés en précision de 16 bits et virgule fixe, comme décrit au sous-paragraphe 4.2. Il y a lieu de prévoir, dans l'implémentation du décodeur en test, la possibilité de désactiver le postfiltre adaptatif. Il convient de commencer le traitement de toutes les séquences de test lorsque l'application en test est à l'état de réinitialisation première, comme défini dans la Recommandation G.728. Trois programmes de mesure – CWCAMP, SNR et WSNR – sont nécessaires pour effectuer les évaluations à la sortie des séquences de test. Ces programmes seront décrits en plus amples détails au paragraphe 3. Les sous-paragraphe suivants (2.1 à 2.6) décrivent les différentes configurations de test à utiliser.

### 2.1 Test du codeur

Le fonctionnement de base du codeur sera contrôlé par la configuration représentée à la Figure 1. Une séquence de signaux de test, IN, est appliquée à l'entrée du codeur en test. Les mots-codes de sortie sont comparés directement aux mots-codes de référence, INCW ou INCW\*G, au moyen du programme CWCAMP.

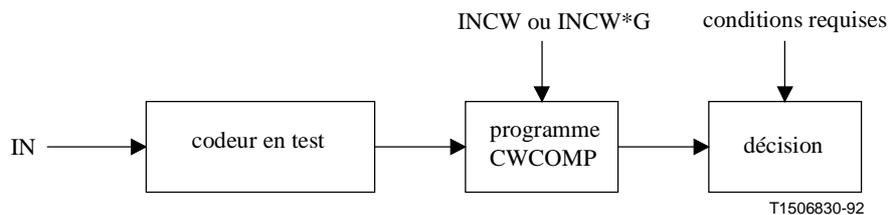


Figure 1 – Configuration de test n° 1 – Test du codeur

### 2.2 Test du décodeur

Le fonctionnement de base du décodeur sera contrôlé par la configuration représentée à la Figure 2. Une séquence de mots-codes de test, CW, est appliquée au décodeur en test après désactivation du postfiltre adaptatif. Le signal de sortie est alors comparé au signal de sortie de référence, OUTA, par le programme SNR.

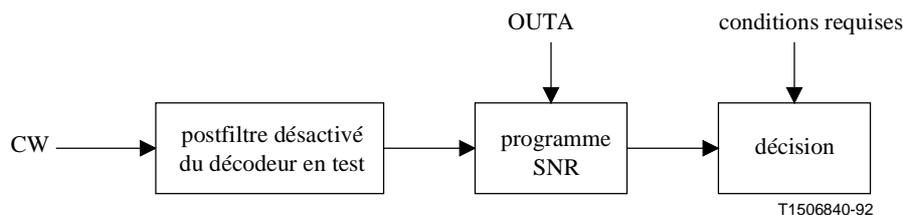


Figure 2 – Configuration de test n° 2 – Test du décodeur

### 2.3 Test du filtre de pondération perceptible

Le filtre de pondération perceptible du codeur sera contrôlé par la configuration de la Figure 3. Une séquence de signaux de test d'entrée, IN5, est injectée dans le codeur en test et on mesure la qualité des mots-codes de sortie au moyen du programme WSNR, qui lui-même a besoin de la séquence d'entrée pour calculer la mesure de distance correcte.

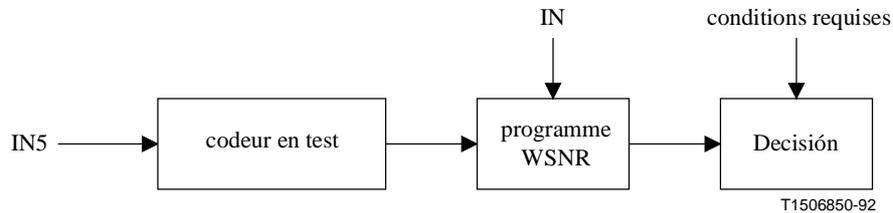


Figure 3 – Configuration de test n° 3 – Test du codeur

### 2.4 Test du postfiltre

Le postfiltre adaptatif du décodeur sera contrôlé par la configuration de la Figure 4. Une séquence de mots-codes test, CW, est appliquée au décodeur en test après activation du postfiltre adaptatif. Le signal de sortie est ensuite comparé au signal de sortie de référence, OUTB, par le programme SNR.

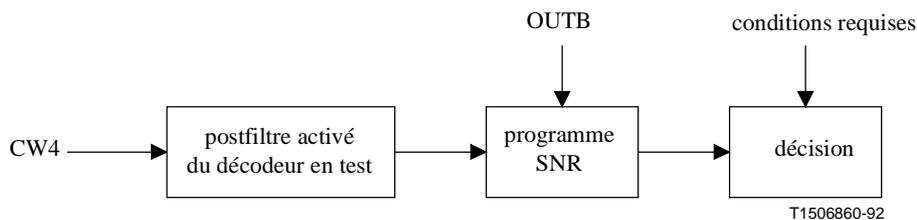


Figure 4 – Configuration de test n° 4 – Test du décodeur

### 2.5 Test du décodeur en virgule fixe

Le fonctionnement de base du décodeur de l'Annexe G est contrôlé par la configuration représentée sur la Figure 5. Une séquence de mots-codes, CW, est appliquée au décodeur en test après coupure du postfiltre adaptatif. Le signal de sortie est ensuite comparé au signal de sortie de référence, OUTA\*G, au moyen d'un programme *diff* de type Unix ® ou d'un programme FC de type MS-DOS ®. Aucune différence ne doit être constatée.

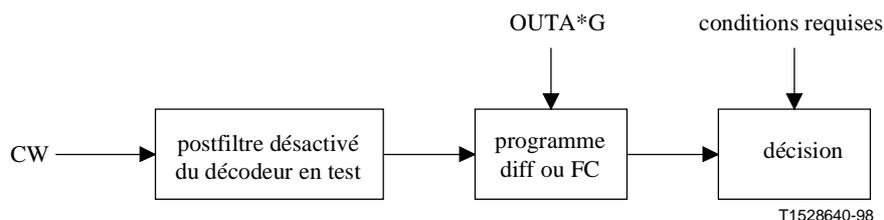


Figure 5 – Configuration de test n° 5 – Test du décodeur en virgule fixe

## 2.6 Test du postfiltre en virgule fixe

Le fonctionnement du postfiltre adaptatif du décodeur en virgule fixe est contrôlé par la configuration représentée sur la Figure 6. Une séquence de mots-codes, CW, est appliquée au décodeur en test après activation du postfiltre adaptatif. Le signal de sortie est ensuite comparé au signal de sortie de référence, OUTB\*G, au moyen d'un programme *diff* de type UNIX<sup>1</sup> ® ou d'un programme FC de type MS-DOS ®<sup>1</sup>. Aucune différence ne doit être constatée.

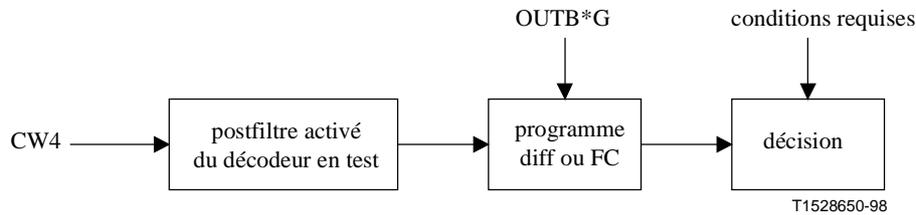


Figure 6 – Configuration de test n° 6 – Test du postfiltre en virgule fixe

## 2.7 Variables d'état interne en virgule fixe

Pour les implémentations en virgule fixe, un court segment de signal vocal d'entrée a été retenu comme signal d'entrée de test additionnel. Toutes les variables d'état interne qui sont associées au cours du traitement de ce signal d'entrée (aussi bien dans le codeur que dans le décodeur) sont également indiquées. Ce signal doit être utilisé par le réalisateur pour vérifier que tout le traitement effectué dans l'implémentation correspond exactement au traitement décrit dans la spécification. Pour être considérée comme totalement conforme à l'Annexe G, il faut qu'une implémentation soit en correspondance exacte avec les valeurs de sortie prescrites pour les signaux d'entrée dans le test en virgule flottante décrit aux sous-paragraphes 2.1, 2.5 et 2.6 ci-dessus, ainsi qu'avec toutes les variables d'état interne pour le court segment de signal vocal.

## 3 Programmes de vérification

Le présent paragraphe décrit les programmes CWCOMP, SNR et WSNR qui sont cités ci-dessus à propos de la configuration de test, ainsi que le programme LDCDEC qui est fourni en tant qu'outil de mise au point pour les réalisateurs.

Le logiciel de vérification est écrit en Fortran et respecte autant que possible la norme Fortran 77 de l'ANSI. Une résolution de virgule flottante en double précision est utilisée le plus souvent afin de minimiser les erreurs numériques dans les modules LD-CELP de référence. Les programmes ont été compilés au moyen d'un compilateur Fortran disponible sur le marché afin d'établir des versions exécutables pour PC à base de processeurs 386/87. Le fichier READ.ME du jeu de distribution décrit la façon de créer des programmes exécutables sur d'autres ordinateurs.

### 3.1 CWCOMP

Le programme CWCOMP est un outil simple qui permet de comparer le contenu de deux fichiers de mots-codes. L'utilisateur est invité à indiquer deux noms de fichier de mots-codes: celui du fichier de sortie du codeur de référence (nom de fichier figurant dans la dernière colonne du Tableau 1) et celui

<sup>1</sup> Unix ® est une marque commerciale des Laboratoires Unix Systems et MS-DOS est une marque commerciale de Microsoft Corporation.

du fichier de sortie du codeur en test. Le programme compare chaque mot-code contenu dans les deux fichiers d'entrée et envoie au terminal le résultat de la comparaison. La condition requise pour la configuration 1 est qu'il n'y ait pas de mots-codes différents.

### 3.2 SNR

Le programme SNR implémente une mesure du rapport signal sur bruit entre deux fichiers de signaux. Le premier est un fichier de référence fourni par le programme du codeur de référence et le deuxième est le fichier de sortie du décodeur de test. Un rapport SNR global, GLOB, est calculé comme rapport signal sur bruit total du fichier. Un rapport SNR "segmentaire", SEG256, est calculé en tant que rapport signal sur bruit moyen de tous les segments de 256 échantillons dont le signal de référence a une puissance supérieure à un seuil donné. Des rapports segmentaires minimaux seront déterminés pour des segments ayant une longueur de 256, 128, 64, 32, 16, 8 et 4 échantillons et ayant une puissance supérieure au seuil indiqué.

Pour exécuter le programme SNR, l'utilisateur doit indiquer les noms de deux fichiers d'entrée. Le premier sera le fichier de sortie du décodeur de référence tel que décrit dans la dernière colonne du Tableau 4, le deuxième sera le fichier de sortie décodé qui est issu du décodeur en test. Après traitement de ces fichiers, le programme envoie au terminal les différents rapports SNR. Les valeurs requises pour les configurations de test 2 et 4 seront indiquées en termes de ces rapports.

### 3.3 WSNR

L'algorithme WSNR est fondé sur une implémentation du décodeur de référence et de la distance de mesure pour calculer la valeur moyenne de distorsion (pondérée par la perception) d'une séquence de mots-codes. Un rapport signal logarithmique sur distorsion est calculé pour chaque vecteur de signal de 5 échantillons et les rapports ainsi obtenus sont réduits à leur moyenne pour tous les vecteurs de signaux dont l'énergie est supérieure à un seuil donné.

Pour exécuter le programme WSNR, l'utilisateur doit indiquer les noms de deux fichiers d'entrée. Le premier étant le fichier de signaux d'entrée du codeur (première colonne du Tableau 1) et le deuxième le fichier de mots-codes de sortie du codeur. Après traitement de la séquence, WSNR envoie au terminal la valeur WSNR ainsi obtenue. La valeur requise pour la configuration de test 3 sera indiquée en termes de ce WSNR.

### 3.4 LDCDEC

En plus des trois programmes de mesurage précédents, les disquettes de distribution comportent un programme de démonstration du décodeur de référence: LDCDEC. Ce programme est fondé sur le même sous-programme de décodage que WSNR et pourra être modifié afin de contrôler, pour la mise au point, les variables du décodeur. L'utilisateur sera invité à indiquer le fichier de mots-codes d'entrée, le fichier de signaux de sortie et à choisir s'il y a lieu d'inclure ou non le postfiltre adaptatif.

### 3.5 Commande *diff* ou FC

En plus des logiciels distribués par les disquettes, l'on part du principe que les réalisateurs disposent d'un système d'exploitation Unix ® ou MS-DOS ®. Les commandes *diff* et FC comparent deux fichiers et indiquent à l'utilisateur s'ils sont identiques ou différents. Pour comparer des fichiers binaires en DOS, la commande appropriée est: FC /B FILE1 FILE2. Pour comparer deux fichiers en Unix, la commande est: diff file1 file2.

## 4 Séquences de test

Les paragraphes suivants décrivent les séquences de test à appliquer. Cette description précise les exigences particulières de chaque séquence.

### 4.1 Conventions de dénomination

Les séquences de test sont numérotées consécutivement, avec un préfixe indiquant le type de signal comme suit:

IN	signal d'entrée du codeur ( <i>encoder input signal</i> )
INCW	mots-codes de sortie du codeur en virgule flottante ( <i>floating point encoder output codewords</i> )
INCW*G	mots-codes de sortie du codeur en virgule fixe ( <i>fixed point encoder output codewords</i> )
CW	mots-codes d'entrée du codeur ( <i>decoder input codewords</i> )
OUTA	signal de sortie du codeur sans postfiltre ( <i>decoder output signal without postfilter</i> )
OUTA*G	signal de sortie du décodeur en virgule fixe sans postfiltre ( <i>fixed point decoder output signal without postfilter</i> )
OUTB	signal de sortie du codeur avec postfiltre ( <i>decoder output signal with postfilter</i> )
OUTB*G	signal de sortie du décodeur en virgule fixe avec postfiltre ( <i>fixed point decoder output signal with postfilter</i> )

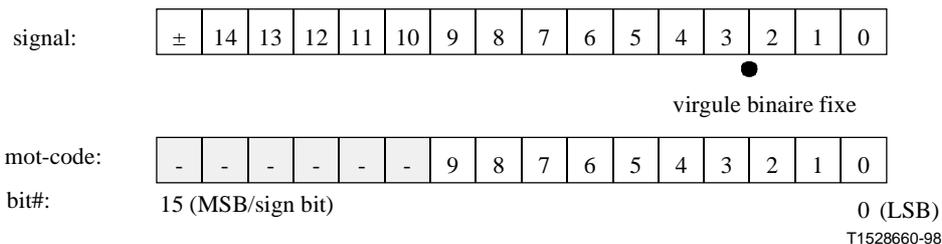
Toutes les séquences de test ont l'extension \*.BIN.

### 4.2 Format des fichiers

Les fichiers de signaux conformes aux interfaces LD-CELP SU et SPF (préfixes de fichier IN, OUTA et OUTB) sont tous en format binaire de 16 bits en complément à 2; il y a lieu de les considérer comme ayant une virgule binaire fixe entre les éléments binaires #2 et #3, comme indiqué dans la Figure 5. Noter que les 16 éléments binaires disponibles doivent tous être utilisés pour obtenir une précision maximale lors des mesurages de test.

Les fichiers de mots-codes (ICHAN de signaux LD-CELP, préfixe de fichier CW ou INCW) sont enregistrés dans le même format binaire à 16 bits que les fichiers de signaux. Les 10 éléments binaires de poids le plus faible de chaque mot de 16 bits représentent le mot-code de 10 bits, comme indiqué dans la Figure 7. Les autres éléments binaires (#12 à #15) sont mis à zéro.

Les fichiers de signaux comme les fichiers de mot-codes sont enregistrés dans le format mots à octets faibles en premier, habituel dans les ordinateurs de type IBM/DOS et VAX/VMS. Pour les applications sur d'autres plates-formes, comme sur la plupart des machines UNIX, cet ordre devra sans doute être modifié par une opération de permutation d'octets.



**Figure 7 – Format de fichiers binaires de signaux et de mots-codes**

### 4.3 Séquences et conditions de test

Les tableaux du présent sous-paragraphe décrivent la séquence complète des tests à effectuer pour vérifier qu'une implémentation en virgule flottante de l'algorithme LD-CELP est conforme à la spécification et est interopérable avec d'autres implémentations correctes. Le Tableau 1 donne un résumé des séquences de tests du codeur. Les conditions correspondantes sont indiquées dans les Tableaux 2 et 3. Les Tableaux 4, 5 et 6 contiennent le résumé et les conditions des séquences de test du décodeur.

**Tableau 1 – Tests du codeur**

Signal d'entrée	Longueur des vecteurs	Description de test	Configuration de test	Signal sortie
IN1	1 536	vérifier que les 1024 mots-codes possibles sont correctement implémentés	1	INCW1, INCW1G
IN2	1 536	mettre à contribution la dynamique de la fonction d'autocorrélation des gains logarithmiques	1	INCW2, INCW2G
IN3	1 024	mettre à contribution la dynamique de la fonction d'autocorrélation des signaux décodés	1	INCW3, INCW3G
IN4	10 240	balayage de fréquence dans l'étendue tonale normale de la parole	1	INCW4, INCW4G
IN5	84 480	signal vocal réel avec différents niveaux d'entrée et différents microphones	3	– INCW5G
IN6	256	limiteurs du codeur en test	1	INCW6, INCW6G

**Tableau 2 – Conditions de test du codeur en virgule flottante**

Signal d'entrée	Signal de sortie	Condition requise
IN1	INCW1	aucun mot-code différent détecté par CWCOMP
IN2	INCW2	aucun mot-code différent détecté par CWCOMP
IN3	INCW3	aucun mot-code différent détecté par CWCOMP
IN4	INCW4	aucun mot-code différent détecté par CWCOMP
IN5	–	WSNR > 20,55 dB
IN6	INCW6	aucun mot-code différent détecté par CWCOMP

**Tableau 3 – Conditions de test du codeur en virgule flottante**

Signal d'entrée	Signal de sortie	Condition requise
IN1	INCW1G	aucun mot-code différent détecté par CWCOMP
IN2	INCW2G	aucun mot-code différent détecté par CWCOMP
IN3	INCW3G	aucun mot-code différent détecté par CWCOMP
IN4	INCW4G	aucun mot-code différent détecté par CWCOMP
IN5	INCW5G	aucun mot-code différent détecté par CWCOMP
IN6	INCW6G	aucun mot-code différent détecté par CWCOMP

**Tableau 4 – Tests du décodeur**

Signal d'entrée	Longueur des vecteurs	Description de test	Configuration de test	Signal sortie
CW1	1 536	vérifier que les 1024 mots-codes possibles sont correctement implémentés	2,5	OUTA1, OUTA1G
CW2	1 792	mettre à contribution la dynamique de la fonction d'autocorrélation des gains logarithmiques	2,5	OUTA2, OUTA2G
CW3	1 280	mettre à contribution la dynamique de la fonction d'autocorrélation des signaux décodés	2,5	OUTA3, OUTA3G
CW4	10 240	contrôler le décodeur par un balayage des fréquences de l'étendue tonale normale de la parole	2,5	OUTA4, OUTA4G
CW4	10 240	contrôler le postfiltre par un balayage des fréquences de l'étendue tonale permise	4,6	OUTB4, OUTB4G
CW5	84 480	signal vocal réel avec différents niveaux d'entrée et différents microphones	2,5	OUTA5, OUTA5G
CW6	256	limiters du décodeur en test	2,5	OUTA6, OUTA6G

**Tableau 5 – Conditions de test du décodeur en virgule flottante**

Fichier de sortie	Conditions requises (valeurs minimales, en décibels, du SNR)								
	SEG256	GLOB	MIN256	MIN128	MIN64	MIN32	MIN16	MIN8	MIN4
OUTA1	75,00	74,00	68,00	68,00	67,00	64,00	55,00	50,00	41,00
OUTA2	94,00	85,00	67,00	58,00	55,00	50,00	48,00	44,00	41,00
OUTA3	79,00	76,00	70,00	28,00	29,00	31,00	37,00	29,00	26,00
OUTA4	60,00	58,00	51,00	51,00	49,00	46,00	40,00	35,00	28,00
OUTB4	59,00	57,00	50,00	50,00	49,00	46,00	40,00	34,00	26,00
OUTA5	59,00	61,00	41,00	39,00	39,00	34,00	35,00	30,00	26,00
OUTA6	69,00	67,00	66,00	64,00	63,00	63,00	62,00	61,00	60,00

## Conditions de test du décodeur en virgule fixe

Il n'y a pas de différences entre une sortie de vecteur de test, OUTA\*G ou OUTB4G et une sortie réelle pour un vecteur de test d'entrée quelconque, CW\*, mesurées par la commande *diff* ou FC ou par un programme de comparaison de fichiers équivalent. Par ailleurs, afin d'être considérée comme tout à fait conforme à l'Annexe G/G.728, une implémentation doit suivre exactement toutes les variables d'état interne pour le court segment de signal vocal.

## 5 Distribution des outils de vérification

Un fichier READ.ME est enregistré sur la disquette n° 1. Il décrit le contenu de chaque fichier et les procédures nécessaires pour compiler et concaténer les programmes. Des extensions servent à distinguer les différents types de fichier. Les fichiers \*.FOR sont des codes sources pour les programmes en Fortran; les fichiers \*.EXE sont des exécutable sur 386/87 et les \*.BIN sont les fichiers de séquences de test binaires. Le contenu de chaque disquette est énuméré dans les Tableaux 6, 7, 8 et 9.

**Tableau 6 – Disquette n° 1 – Répertoire de distribution**

Disquette	Nom du fichier	Nombre d'octets
disquette n° 1  capacité totale: 1 289 859 octets	READ.ME	10 430
	CWCOMP.FOR	2 642
	CWCOMP.EXE	25 153
	SNR.FOR	5 536
	SNR.EXE	36 524
	WSNR.FOR	3 554
	WSNR.EXE	103 892
	LDCDEC.FOR	3 016
	LDCDEC.EXE	101 080
	LDCSUB.FOR	37 932
	FILSUB.FOR	1 740
	DSTRUCT.FOR	2 968
	IN1.BIN	15 360
	IN2.BIN	15 360
	IN3.BIN	10 240
	IN5.BIN	844 800
	IN6.BIN	2 560
	INCW1.BIN	3 072
	INCW2.BIN	3 072
	INCW3.BIN	2 048
INCW6.BIN	512	
CW1.BIN	3 072	
CW2.BIN	3 584	
CW3.BIN	2 560	
CW6.BIN	512	

**Tableau 6 – Disquette n° 1 – Répertoire de distribution (fin)**

Disquette	Nom du fichier	Nombre d'octets
	OUTA1.BIN	15 360
	OUTA2.BIN	17 920
	OUTA3.BIN	12 800
	OUTA6.BIN	2 560

**Tableau 7 – Disquette n° 2 – Répertoire de distribution**

Disquette	Nom du fichier	Nombre d'octets
disquette n° 2  capacité totale: 1 361 920 octets	IN4.BIN	102 400
	INCW4.BIN	20 480
	CW4.BIN	20 480
	CW5.BIN	168 960
	OUTA4.BIN	102 400
	OUTB4.BIN	102 400
	OUTA5.BIN	844 800

**Tableau 8 – Disquette n° 3 – Répertoire de distribution**

Disquette	Nom du fichier	Nombre d'octets
disquette n° 3  capacité totale: 1 297 280 octets	INCW1G.BIN	3 072
	INCW2G.BIN	3 072
	INCW3G.BIN	2 048
	INCW4G.BIN	20 480
	INCW5G.BIN	168 960
	INCW6G.BIN	512
	OUTA1G.BIN	15 360
	OUTA2G.BIN	17 920
	OUTA3G.BIN	12 800
	OUTA4G.BIN	102 400
	OUTB4G.BIN	102 400
	OUTA5G.BIN	844 800
	OUTA6G.BIN	2 560
	READ.ME	896

**Disquette n° 4 – Variables d'état internes**

La spécification du codeur en virgule fixe figurant dans l'Annexe G/G.728 est en chiffres exacts. Un court segment de signal vocal est utilisé comme entrée de test pour comparer les représentations internes de toutes les variables d'état de deux implémentations différentes. Tous les vecteurs de sortie doivent correspondre. Afin d'éviter toute confusion, toutes ces variables sont enregistrées en ASCII sur la disquette n° 4.

**Tableau 9 – Disquette n° 4 – Répertoire de distribution**

Capacité	Nom du fichier	Remarques
36 100	a.q14	de a(2) à a(51) dans Q14. Un "-" signifie qu'il n'y a pas de mise à jour pour ce vecteur.
7 700	ap.q14	ap() dans Q14
8 400	apf.bf	apf() est la sortie intermédiaire du bloc 50, à formater ensuite par Q
7 700	apf.q13	variable finale apf() dans Q13 (convertie à partir de apf.bf)
36 900	atmp.bf	de atmp(2) à atmp(51), puis formater par IAQ Q. (sortie du bloc 50)
7 700	awp.q14	de awp(2) à awp(11) dans Q14. Un "-" signifie qu'il n'y a pas de mise à jour pour ce vecteur.
7 700	awz.q14	de awz(2) à awz(11) dans Q14. Un "-" signifie qu'il n'y a pas de mise à jour pour ce vecteur.
8 400	awztmp.bf	de awztmp(2) à awztmp(11) dans Q13, Q14, ou Q15, suivies par le format Q dans la dernière colonne (sortie du bloc 37)
7 700	az.q14	az() dans Q14
1 200	b.q16	b dans Q16 (coefficient du postfiltre à long terme, calculé dans le bloc 84)
14 400	d.q1	plus récent vecteur de la table des d() dans Q1
4 200	dec.q1	dec(21:25) dans Q1 (nouveau résidu de codage prédictif linéaire, sous-échantillonné pour la trame courante)
15 600	et.bf	et() en virgule flottante dans le bloc; 5 mantisses et nlset à chaque ligne.
3 600	gain.sf	gain linéaire utilisé pour normaliser le vecteur-code (mantisse, puis nlsgain)
4 000	gaininv.sf	inverse du gain utilisé pour normaliser le vecteur cible (mantisse, puis NLS)
1 400	gl.q14	gl dans Q14
1 400	glb.q14	glb dans Q14
7 700	gp.q14	gp(2) a gp(11) dans Q14. Un "-" signifie qu'il n'y a pas de mise à jour pour ce vecteur.
8 400	gptmp.bf	de gptmp(2) à gptmp(11), puis format Q des gptmp (à la sortie du bloc 44)
2 800	gstate.q9	gstate(1) dans Q9. (Les 9 autres vecteurs gstate() se trouvent dans les lignes précédentes.)
3 500	gtmp.q9	gtmp() dans Q9. Noter que le premier vecteur gtmp() possède trois valeurs -16384.
4 200	h.q13	vecteur h() dans Q13. Un "-" signifie qu'il n'y a pas de mise à jour pour ce vecteur.
2 000	ichan.q0	index du canal de sortie du codeur "ichant" (un par ligne)
14 400	input.q3	vecteur d'entrée à codage MIC linéaire sur 16 bits (Q3 fixe, un vecteur par ligne)
2 400	isig.q0	index de forme "is" suivi de l'index de gain "ig" à chaque ligne
1 400	kp.q0	période tonale kp dans Q0
2 800	loggain.q9	gain logarithmique avant conversion en gain linéaire (entrée du bloc 48)
14 800	lpfiir.q1	les 20 éléments de lpfiir() correspondant à la trame actuelle
14 400	output.q3	vecteur de sortie du décodeur (avec postfiltre) en codage MIC linéaire sur 16 bits

**Tableau 9 – Disquette n° 4 – Répertoire de distribution (fin)**

Capacité	Nom du fichier	Remarques
14 400	pn.q7	pn() dans Q7 (sortie du bloc 13)
1 200	ptap.q14	ptap dans Q14 (sortie du bloc 83)
8 400	r_b36.bf	r(1) a r(11) à la sortie du bloc 36
8 400	r_b43.bf	de r(1) a r(11) à la sortie du bloc 43
1 400	rc1.q15	variable rc1 du bloc 50 dans Q15 (utilisée pour calculer l'écart z)
5 821	readme	ce fichier décrit le contenu de la disquette n° 4
37 600	rexp.bf	de rexp(1) à rexp(51), puis nlsrexp. (sortie du bloc 49)
9 200	rexp1g.bf	de rexp1g(1) à rexp1g(11), puis nlsrexp1g. (sortie du bloc 43)
9 200	rexp1w.bf	de rexp1w(1) à rexp1w(11), puis nlsrexp1w. (sortie du bloc 36)
36 900	rtmp.bf	de rtmp(1) à rtmp(51) à la sortie du bloc 49
14 400	s.q2	vecteur d'entrée s() vector après conversion de l'entrée q3 en Q2 et arrondissement
3 600	scale.sf	normalisation par calcul scalaire à virgule flottante (sortie du bloc 75)
14 400	scalefil.q14	variable scalefil dans Q14 (sortie du bloc 76)
14 400	sst.q0	Q0 sst(-4:0) après décalage de registre SST() (c'est-à-dire sst(1:5) >> 2)
14 400	sst.q2	sst(1:5) dans Q2
14 400	st.bf	st() en format de virgule flottante par bloc
15 600	statelpc.sbf	les 5 éléments les plus récents des variables statelpc() et nlsstate(10) pour le vecteur courant
15 600	stmp.q2	stmp() dans Q2. Son contenu va jusqu'au vecteur 2 de la trame courante.
14 800	stpffir.q2	stpffir(1:5) après postfiltrage du vecteur courant
14 400	stpfir.q2	stpfir(1:5) après postfiltrage du vecteur courant
14 400	sttmp.sbf	sttmp(), 20 mantisses suivies de 4 exposants (nlssttmp()).
17 700	sw.q2	sw() dans Q2 (sortie du bloc 4)
3 200	sumfil.q2	sumfil dans Q2 à la sortie du bloc 74 (AA1 en pseudo-code)
3 200	sumunfil.q2	sumunfil dans Q2 à la sortie du bloc 73 (AA0 en pseudo-code)
14 400	target.q2	vecteur cible non normalisé dans Q2 (sortie du bloc 11)
14 400	targetn.bf	vecteur cible normalisé par le gain en virgule flottante de bloc
15 600	temp_b72.q2	temp() à la sortie du bloc 72, dans Q2
14 400	tiltz.q14	écart z dans Q14
1 400	wiir.q2	les 5 éléments les plus récents de wiir() dans Q2 après filtrage de pondération
14 400	y2.q5	table des y2() dans Q5. Un "-" signifie qu'il n'y a pas de mise à jour pour ce vecteur.
78 700	zir.q2	vecteur zir() dans Q2
14 400	zirwfir.q2	les 5 éléments les plus récents de zirwfir() après mise à jour en mémoire du bloc 10
14 400	zirwiir.q2	les 5 éléments les plus récents de zirwiir()après mise à jour en mémoire du bloc 10



## LOGICIELS UIT-T

G.191 (11.96)	Bibliothèque logicielle 96 (STL-96) et manuel STL-96 (en anglais seulement)
G.722 Appendice II (03.87)	Séquences numériques de test pour la vérification du codec à 7 kHz G.722 SB-MICDA à 64 kbit/s
G.723.1 Annexe A (11.96)	Programme source C et signaux de test pour le vocodeur bi-débit en virgule fixe à 5,3 et 6,3 kbit/s et pour le schéma de compression des silences, version 5.1
G.723.1 Annexe B (11.96)	Programme source C et signaux de test pour le vocodeur bi-débit en virgule flottante à 5,3 et 6,3 kbit/s, version 5.1F
G.723.1 Annexe C (11.96)	Programme source C et signaux de test pour le codec de voie échelonné, version 3.1
G.726 Appendice II (03.91)	Séquences numériques de test pour la vérification de l'algorithme MICDA G.726 à 40, 32, 24 et 16 kbit/s
G.727 Appendice I (03.91)	Séquences numériques de test pour la vérification de l'algorithme MICDA imbriqué G.727 à 5, 4, 3 et 2 bit/échantillon
G.728 Appendice I (07.95)	Programme et séquences de test pour la vérification des implémentations de l'algorithme du vocodeur LD-CELP G.728 à 16 kbit/s
G.729 (03.96)	Code source C et vecteurs de test pour la vérification des implémentations du vocodeur CS-ACELP G.729 à 8 kbit/s
G.729 Annexe A (11.96)	Code source C et vecteurs de test pour la vérification des implémentations du vocodeur CS-ACELP G.729 à 8 kbit/s simplifié
G.729 Annexe B (10.96)	Code source C et vecteurs de test pour la vérification des implémentations de l'algorithme G.729 de compression des silences
P.501 (08.96)	Signaux de test pour la téléphonométrie
P.861 (08.96)	Programme C de référence de la mesure de qualité vocale perçue (PSQM)
Q.921 <i>bis</i> (03.93)	Suite de test abstraite pour les essais de conformité des procédures LAPD – Partie I: débit de base côté utilisateur
Q.931 <i>bis</i> (02.95)	Formulaires PICS et suite de test abstraite pour le test de conformité de la commande d'appel de base en mode circuit dans le DSS 1 couche 3 du RNIS
Q.933 <i>bis</i> (10.95)	Formulaires PICS et suite de test abstraite pour le test de conformité de la commande d'appel de base en mode relais de trames des PVC – Section I: côtés utilisateur et réseau de l'interface utilisateur/réseau
T.24 (11.94)	Ensemble normalisé de mires numérisées
T.83 (11.94)	Données de test de conformité pour le codeur et le décodeur génériques de compression et de codage numérique des images fixes à modelé continu