INTERNATIONAL TELECOMMUNICATION UNION

# ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

# G.728

## Appendix I
## Verification tools

(07/95)

SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

Digital transmission systems – Terminal equipments – Coding of analogue signals by methods other than PCM

---

# Programs and test sequences for implementation verification of the algorithm of the G.728 16 kbit/s LD-CELP speech coder

ITU-T Recommendation G.728 – Appendix I

(Previously CCITT Recommendation)

---

## FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

## NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

## INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

# PROGRAMS AND TEST SEQUENCES FOR IMPLEMENTATION VERIFICATION OF THE ALGORITHM OF THE G.728 16 kbit/S LD-CELP SPEECH CODER

## 1 General

This document describes the digital test sequences and the measurement software to be used for implementation verification of Recommendation G.728. Provision is included for both floating point implementations, based on the main body of Recommendation G.728, and bit exact fixed point implementations, based on Annex G/G.728.

### 1.1 Verification principles for floating point implementations

The main body of the LD-CELP algorithm specification is formulated in a non-bit exact manner to allow for simple implementation on different kinds of hardware. This implies that the verification procedure cannot assume the implementation under test to be exactly equal to any reference implementation. Hence, objective measurements are needed to establish the degree of deviation between test and reference. If this measured deviation is found to be sufficiently small, the test implementation is assumed to be interoperable with any other implementation passing the test. Since no finite length test is capable of testing every aspect of an implementation, 100% certainty that an implementation is correct can never be guaranteed. However, the test procedure described exercises all main parts of the LD-CELP algorithm and should be a valuable tool for the implementor.

The floating point verification procedures described in this document have been designed with 32-bit floating point implementations in mind. Although they could be applied to any LD-CELP implementation, 32-bit floating point format will probably be needed to fulfil the test requirements.

### 1.2 Verification principles for fixed point implementations

Annex G/G.728 describes the fixed point LD-CELP algorithm in a bit exact manner. This implies that when two encoder or decoder implementations are started from their initial states with identical input signals, all state variables should be exactly identical at equivalent instants throughout the processing of the inputs. Consequently, the floating point input test sequences may be used and bit exact output sequences should result.

A short segment of input speech has been provided as an additional test input signal. All of the associated internal state variables during the course of processing this input are also provided. This is a means for the implementor to verify that all processing in the implementation exactly matches the processing described in the specification. To be considered fully compliant with Annex G/G.728, an implementation should exactly match both the prescribed outputs for the floating point test input signals and all of the internal state variables for the short speech segment.

As in the case of the floating point implementation verification procedure, since no finite length test is capable of testing every aspect of an implementation, 100% certainty that an implementation is correct can never be guaranteed. However, the test procedure described exercises all main parts of the LD-CELP algorithm and should be a valuable tool for the implementor.

## 2 Test configurations

This clause describes how the different test sequences and measurement programs should be used together to perform the verification tests. The procedure is based on black-box testing at the interfaces SU and ICHAN of the test encoder and ICHAN and SPF of the test decoder. The signals SU and SPF are represented in 16-bit fixed point precision as described in subclause 4.2. A possibility to turn off the adaptive postfilter should be provided in the tested decoder

implementation. All test sequences processing should be started with the test implementation in the initial reset state, as defined by Recommendation G.728. Three measurement programs, CWCOMP, SNR and WSNR, are needed to perform the test output sequence evaluations. These programs are further described in clause 3. Descriptions of the different test configurations to be used are found in the following subclauses (2.1-2.6).

## 2.1 Encoder test

The basic operation of the encoder is tested with the configuration shown in Figure 1. An input test sequence, IN, is applied to the encoder under test. The output codewords are compared directly to the reference codewords, INCW or INCW*G, using the CWCOMP program.

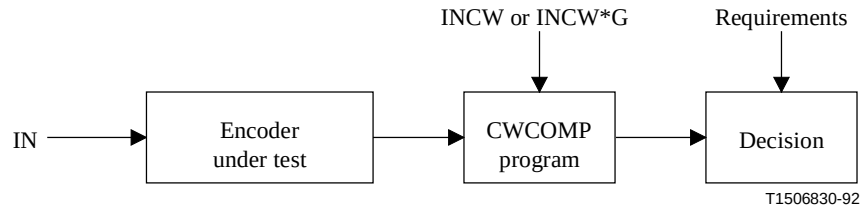


**Figure 1 – Test configuration No. 1 – Encoder test**

## 2.2 Decoder test

The basic operation of the decoder is tested with the configuration shown in Figure 2. A codeword test sequence, CW, is applied to the decoder under test with the adaptive postfilter turned off. The output signal is then compared to the reference output signal, OUTA, with the SNR program.

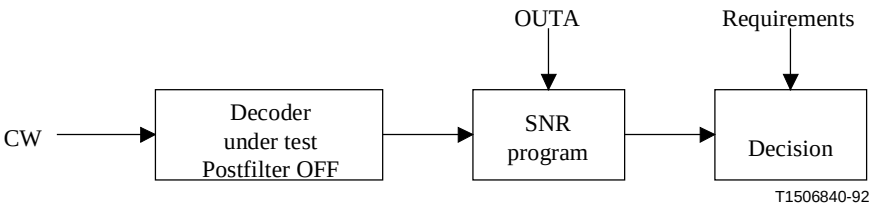


**Figure 2 – Test configuration No. 2 – Decoder test**

## 2.3 Perceptual weighting filter test

The encoder perceptual weighting filter is tested with the configuration in Figure 3. An input signal test sequence, IN5, is passed through the encoder under test, and the quality of the output codewords are measured with the WSNR program. The WSNR program also needs the input sequence to compute the correct distance measure.

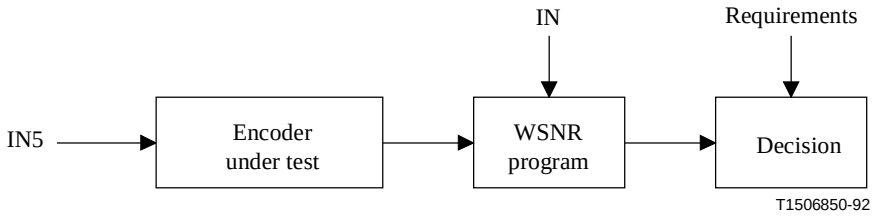


**Figure 3 – Test configuration No. 3 – Encoder test**

## 2.4 Postfilter test

The decoder adaptive postfilter is tested with the configuration in Figure 4. A codeword test sequence, CW, is applied to the decoder under test with the adaptive postfilter turned on. The output signal is then compared to the reference output signal, OUTB, with the SNR program.

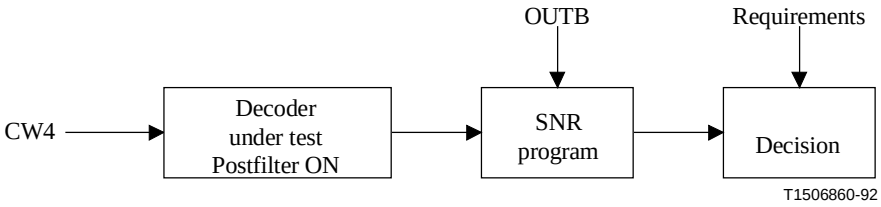


**Figure 4 – Test configuration No. 4 – Decoder test**

## 2.5 Fixed point decoder test

The basic operation of the Annex G decoder is tested with the configuration shown in Figure 5. A codeword sequence, CW, is applied to the decoder under test with the adaptive postfilter turned off. The output signal is then compared to the reference output signal, OUTA*G, with a *diff* program in Unix ® or a FC program in MS-DOS ®. No differences should be encountered.

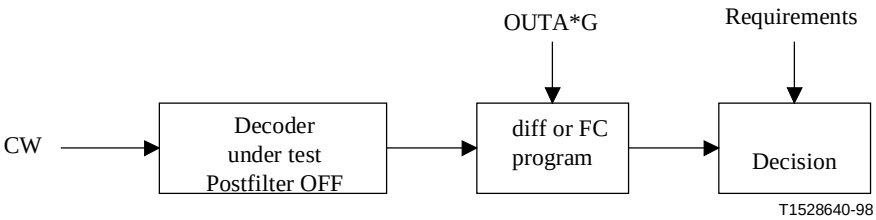


**Figure 5 – Test configuration No. 5 – Fixed point decoder test**

## 2.6 Fixed point postfilter test

The fixed point decoder adaptive postfilter is tested with the configuration in Figure 6. A codeword test sequence, CW, is applied to the decoder under test with the adaptive postfilter turned on. The output signal is then compared to the reference output signal, OUTB*G, with a *diff* program in Unix ®[1] or a FC program in MS-DOS ®[1]. No differences should be encountered.

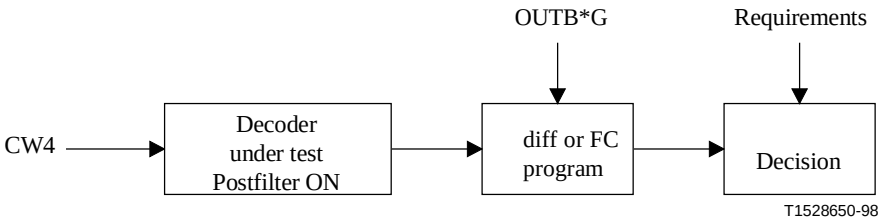


**Figure 6 – Test configuration No. 6 – Fixed point postfilter test**

## 2.7 Fixed point internal state variables

For fixed point implementations, a short segment of input speech has been provided as an additional test input signal. All associated internal state variables required for processing this input (both in the encoder and decoder) are also provided. This signal is to be used by the implementor to verify that all processing in the implementation exactly matches the processing described in the specification. To be considered fully compliant with Annex G, an implementation should exactly match both the prescribed outputs for the floating point test input signals, as described in subclauses 2.1, 2.5 and 2.6 above, and all internal state variables for the short speech segment.

# 3 Verification programs

This clause describes the programs CWCOMP, SNR and WSNR, referred to in the test configuration section, as well as the program LDCDEC provided as an implementors debugging tool.

The verification software is written in Fortran and is kept as close to the ANSI Fortran 77 standard as possible. Double precision floating point resolution is used extensively to minimize numerical error in the reference LD-CELP modules. The programs have been compiled with a commercially available Fortran compiler to produce executable versions for 386/87-based PCs. The READ.ME file in the distribution describes how to create executable programs on other computers.

## 3.1 CWCOMP

The CWCOMP program is a simple tool to compare the content of two codeword files. The user is prompted for two codeword file names, the reference encoder output (filename in last column of Table 1) and the test encoder output. The program compares each codeword in these files and writes the comparison result to terminal. The requirement for test configuration 1 is that no different codewords should exist.

---

[1] ® Unix is a trademark of Unix Systems Laboratories and MS-DOS is a trademark of Microsoft Corporation.

### 3.2 SNR

The SNR program implements a signal-to-noise ratio measurement between two signal files. The first is a reference file provided by the reference decoder program and the second is the test decoder output file. A global SNR, GLOB, is computed as the total file signal-to-noise ratio. A segmental SNR, SEG256, is computed as the average signal-to-noise ratio of all 256-sample segments with reference power above a certain threshold. Minimum segment SNRs are found for segments of lengths 256, 128, 64, 32, 16, 8 and 4 with power above the same threshold.

To run the SNR program, the user needs to enter names of two input files. The first is the reference decoder output file as described in the last column of Table 4. The second is the decoded output file produced by the decoder under test. After processing the files, the program outputs the different SNRs to terminal. Requirement values for the test configurations 2 and 4 are given in terms of these SNR numbers.

### 3.3 WSNR

The WSNR algorithm is based on a reference decoder and distance measure implementation to compute the mean perceptually weighted distortion of a codeword sequence. A logarithmic signal-to-distortion ratio is computed for every 5-sample signal vector and the ratios are averaged over all signal vectors with energy above a certain threshold.

To run the WSNR program, the user needs to enter names of two input files. The first is the encoder input signal file (first column of Table 1) and the second is the encoder output codeword file. After processing the sequence, WSNR writes the output WSNR value to terminal. The requirement value for test configuration 3 is given in terms of this WSNR number.

### 3.4 LDCDEC

In addition to the three measurement programs, the distribution also includes a reference decoder demonstration program, LDCDEC. This program is based on the same decoder subroutine as WSNR and could be modified to monitor variables in the decoder for debugging purposes. The user is prompted for the input codeword file, the output signal file and whether to include the adaptive postfilter or not.

### 3.5 *diff* or FC

In addition to the software distributed with the diskettes, it is assumed that the implementor has available a Unix ® or MS-DOS ® operating system. The commands *diff* and FC compare two files and tell the user whether they are the same or different. For comparing binary files in DOS, the proper command is FC /B FILE1 FILE2. For comparing two files under Unix, the command is diff file1 file2.

## 4 Test sequences

The following is a description of the test sequences to be applied. The description includes the specific requirements for each sequence.

### 4.1 Naming conventions

The test sequences are numbered sequentially, with a prefix that identifies the type of signal:

| | |
|---|---|
| IN | Encoder input signal |
| INCW | Floating point encoder output codewords |
| INCW*G | Fixed point encoder output codewords |

CW Decoder input codewords

OUTA Decoder output signal without postfilter

OUTA*G Fixed point decoder output signal without postfilter

OUTB Decoder output signal with postfilter

OUTB*G Fixed point decoder output signal with postfilter

All test sequence files have the extension *.BIN.

## 4.2 File formats

The signal files, according to the LD-CELP interfaces SU and SPF (file prefix IN, OUTA and OUTB) are all 2's complement 16-bit binary format and should be interpreted to have a fixed binary point between bits #2 and #3, as shown in Figure 5. Note that all the 16 available bits must be used to achieve maximum precision in the test measurements.

The codeword files (LD-CELP signal ICHAN, file prefix CW or INCW) are stored in the same 16-bit binary format as the signal files. The least significant 10 bits of each 16-bit codeword represent the 10-bit codeword, as shown in Figure 7. The other bits (#12-#15) are set to zero.

Both signal and codeword files are stored in the low-byte first word storage format that is usual on IBM/DOS and VAX/VMS computers. For use on other platforms, such as most UNIX machines, the ordering may have to be changed by a byteswap operation.

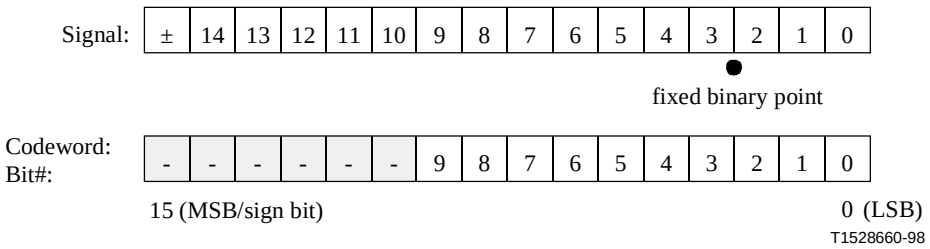


**Figure 7 – Signal and codeword binary file format**

## 4.3 Test sequences and requirements

The tables in this subclause describe the complete set of tests to be performed to verify that a floating point implementation of LD-CELP follows the specification and is interoperable with other correct implementations. Table 1 is a summary of the encoder test sequences. The corresponding requirements are expressed in Tables 2 and 3. Tables 4, 5 and 6 contain the decoder test sequence summary and requirements.

**Table 1 – Encoder tests**

| Input signal | Length, vectors | Description of test | Test config. | Output signal |
|---|---|---|---|---|
| IN1 | 1 536 | Test that all 1024 possible codewords are properly implemented | 1 | INCW1, INCW1G |
| IN2 | 1 536 | Exercise dynamic range of log-gain autocorrelation function | 1 | INCW2, INCW2G |
| IN3 | 1 024 | Exercise dynamic range of decoded signals autocorrelation function | 1 | INCW3, INCW3G |
| IN4 | 10 240 | Frequency sweep through typical speech pitch range | 1 | INCW4, INCW4G |
| IN5 | 84 480 | Real speech signal with different input levels and microphones | 3 | – INCW5G |
| IN6 | 256 | Test encoder limiters | 1 | INCW6, INCW6G |

**Table 2 – Floating point encoder test requirements**

| Input signal | Output signal | Requirement |
|---|---|---|
| IN1 | INCW1 | 0 different codewords detected by CWCOMP |
| IN2 | INCW2 | 0 different codewords detected by CWCOMP |
| IN3 | INCW3 | 0 different codewords detected by CWCOMP |
| IN4 | INCW4 | 0 different codewords detected by CWCOMP |
| IN5 | – | WSNR > 20.55 dB |
| IN6 | INCW6 | 0 different codewords detected by CWCOMP |

**Table 3 – Fixed point encoder test requirements**

| Input signal | Output signal | Requirement |
|---|---|---|
| IN1 | INCW1G | 0 different codewords detected by CWCOMP |
| IN2 | INCW2G | 0 different codewords detected by CWCOMP |
| IN3 | INCW3G | 0 different codewords detected by CWCOMP |
| IN4 | INCW4G | 0 different codewords detected by CWCOMP |
| IN5 | INCW5G | 0 different codewords detected by CWCOMP |
| IN6 | INCW6G | 0 different codewords detected by CWCOMP |

**Table 4 – Decoder tests**

| Input signal | Length, vectors | Description of test | Test config. | Output signal |
|---|---|---|---|---|
| CW1 | 1 536 | Test that all 1024 possible codewords are properly implemented | 2, 5 | OUTA1, OUTA1G |
| CW2 | 1 792 | Exercise dynamic range of log-gain autocorrelation function | 2, 5 | OUTA2, OUTA2G |
| CW3 | 1 280 | Exercise dynamic range of decoded signals autocorrelation function | 2, 5 | OUTA3, OUTA3G |
| CW4 | 10 240 | Test decoder with frequency sweep through typical speech pitch range | 2, 5 | OUTA4, OUTA4G |
| CW4 | 10 240 | Test postfilter with frequency sweep through typical speech pitch range | 4, 6 | OUTB4, OUTB4G |
| CW5 | 84 480 | Real speech signal with different input levels and microphones | 2, 5 | OUTA5, OUTA5G |
| CW6 | 256 | Test decoder limiters | 2, 5 | OUTA6, OUTA6G |

**Table 5 – Floating point decoder test requirements**

| Output | Requirements (minimum values for SNR, in dB) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| file name | 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 |

### Fixed point decoder test requirements

No differences between output test vector, OUTA*G or OUTB4G, and actual output for any input test vector, CW*, as measured by *diff* or FC or an equivalent file comparison program. In addition, to be considered fully compliant with Annex G/G.728, an implementation should follow exactly all of the internal state variables for the short speech segment.

## 5 Verification tools distribution

A READ.ME file is included in diskette #1 to describe the contents of each file and the procedures necessary to compile and link the programs. Extensions are used to separate different file types. *.FOR files are source code for the Fortran programs, *.EXE files are 386/87 executables and *.BIN are binary test sequence files. The content of each diskette is listed in Tables 6, 7, 8 and 9.

**Table 6 – Distribution directory disk #1**

| Diskette | Filename | Number of bytes |
|---|---|---|

| | | |
|---|---|---|
| Diskette #1 | READ.ME | 10 430 |
| | CWCOMP.FOR | 2 642 |
| Total size: | CWCOMP.EXE | 25 153 |
| 1 289 859 bytes | 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 |
| | OUTA1.BIN | 15 360 |
| | OUTA2.BIN | 17 920 |
| | OUTA3.BIN | 12 800 |
| | OUTA6.BIN | 2 560 |

**Table 7 – Distribution directory disk #2**

| Diskette | Filename | Number of bytes |
|---|---|---|
| Diskette #2 | IN4.BIN | 102 400 |
| | INCW4.BIN | 20 480 |
| Total size: | CW4.BIN | 20 480 |
| 1 361 920 bytes | CW5.BIN | 168 960 |
| | OUTA4.BIN | 102 400 |
| | OUTB4.BIN | 102 400 |
| | OUTA5.BIN | 844 800 |

**Table 8 – Distribution directory disk #3**

| Diskette | Filename | Number of bytes |
|---|---|---|

| | | |
|---|---|---|
| Diskette #3 | INCW1G.BIN | 3 072 |
| | INCW2G.BIN | 3 072 |
| Total size: | INCW3G.BIN | 2 048 |
| 1 297 280 bytes | 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 |

**Diskette #4 – Internal state variables**

The specification of fixed point G.728 coder in Annex G/G.728 is bit exact. A short segment of speech is used as a test input to compare the internal representations of all state variables between two different implementations. All of the output vectors should match. To avoid any confusion, all of these outputs are stored in ASCII on disk #4.

**Table 9 – Distribution directory disk #4**

| Size | Filename | Remarks |
|---|---|---|
| 36 100 | a.q14 | a(2) to a(51) in Q14. A "-" means no update for that vector. |
| 7 700 | ap.q14 | ap() in Q14 |
| 8 400 | apf.bf | apf() as the intermediate output of block 50, then Q format |
| 7 700 | apf.q13 | the final apf() in Q13 (converted from apf.bf) |
| 36 900 | atmp.bf | atmp(2) to atmp(51), then IAQ Q format. (block 50 output) |
| 7 700 | awp.q14 | awp(2) to awp(11) in Q14. A "-" means no update for that vector. |
| 7 700 | awz.q14 | awz(2) to awz(11) in Q14. A "-" means no update for that vector. |
| 8 400 | awztmp.bf | awztmp(2) to awztmp(11) in Q13, Q14, or Q15, followed by the Q format in the last column. (block 37 output) |
| 7 700 | az.q14 | az() in Q14 |
| 1 200 | b.q16 | b in Q16 (long-term postfilter coeff. computed in block 84) |
| 14 400 | d.q1 | the newest vector of d() array in Q1 |
| 4 200 | dec.q1 | dec(21:25) in Q1 (new decimated LPC residual for current frame) |
| 15 600 | et.bf | et() in block floating-pt; 5 mantissas and nlset in each line. |
| 3 600 | gain.sf | linear gain used to scale codevector (mantissa, then nlsgain) |
| 4 000 | gaininv.sf | 1/GAIN used to normalize target vector (mantissa, then NLS) |
| 1 400 | gl.q14 | gl in Q14 |
| 1 400 | glb.q14 | glb in Q14 |
| 7 700 | gp.q14 | gp(2) to gp(11) in Q14. A "-" means no update for that vector. |
| 8 400 | gptmp.bf | gptmp(2) to gptmp(11), then gptmp Q format. (block 44 output) |

**Table 9 – Distribution directory disk #4**

| Size | Filename | Remarks |
|---|---|---|
| 2 800 | gstate.q9 | gstate(1) in Q9. (The other 9 gstate() are in previous lines.) |
| 3 500 | gtmp.q9 | gtmp() in Q9. Note the first gtmp() vector has three -16384. |
| 4 200 | h.q13 | h() vector in Q13. A "-" means no update for that vector. |
| 2 000 | ichan.q0 | encoder output channel index "ichan" (one per line) |
| 14 400 | input.q3 | 16-bit linear PCM input vector (fixed Q3, one vector a line) |
| 2 400 | isig.q0 | shape index "is" followed by gain index "ig" in each line |
| 1 400 | kp.q0 | the pitch period kp in Q0 |
| 2 800 | loggain.q9 | log-gain before converting to linear gain (block 48 input) |
| 14 800 | lpfiir.q1 | the 20 elements of lpfiir() corresponding to the current frame |
| 14 400 | output.q3 | decoder (with postfilter) output vector in 16-bit linear PCM |
| 14 400 | pn.q7 | pn() in Q7 (block 13 output) |
| 1 200 | ptap.q14 | ptap in Q14 (output of block 83) |
| 8 400 | r_b36.bf | r(1) to r(11) at block 36 output |
| 8 400 | r_b43.bf | r(1) to r(11) at block 43 output |
| 1 400 | rc1.q15 | rc1 of block 50 in Q15 (the one used to derive tiltz) |
| 5 821 | readme | describes contents of disk #4 |
| 37 600 | rexp.bf | rexp(1) to rexp(51), then nlsrexp. (block 49 output) |
| 9 200 | rexplg.bf | rexplg(1) to rexplg(11), then nlsrexplg. (block 43 output) |
| 9 200 | rexpw.bf | rexpw(1) to rexpw(11), then nlsrexpw. (block 36 output) |
| 36 900 | rtmp.bf | rtmp(1) to rtmp(51) at block 49 output |
| 14 400 | s.q2 | input s() vector after converting input.q3 to Q2 with rounding |
| 3 600 | scale.sf | scale in scalar floating-point (output of block 75) |
| 14 400 | scalefil.q14 | scalefil in Q14 (output of block 76) |
| 14 400 | sst.q0 | Q0 sst(-4:0) after SST() buffer shift (i.e. sst(1:5) >> 2) |
| 14 400 | sst.q2 | sst(1:5) in Q2 |
| 14 400 | st.bf | st() in block floating-point format |
| 15 600 | statelpc.sbf | The newest 5 elements of statelpc() and nlsstate(10) for the current vector |
| 15 600 | stmp.q2 | stmp() in Q2. Its content is up to vector 2 of current frame. |
| 14 800 | stpffir.q2 | stpffir(1:5) after postfiltering the current vector |
| 14 400 | stpfiir.q2 | stpfiir(1:5) after postfiltering the current vector |
| 14 400 | sttmp.sbf | sttmp(), 20 mantissas followed by 4 exponents (nlssttmp()). |
| 17 700 | sw.q2 | sw() in Q2 (block 4 output) |
| 3 200 | sumfil.q2 | sumfil in Q2 at the output of block 74 (AA1 in pseudo-code) |
| 3 200 | sumunfil.q2 | sumunfil in Q2 at the output of block 73 (AA0 in pseudo-code) |
| 14 400 | target.q2 | unnormalized target vector in Q2 (block 11 output) |
| 14 400 | targetn.bf | gain-normalized target vector in block floating-point |
| 15 600 | temp_b72.q2 | temp() at the output of block 72, in Q2 |
| 14 400 | tiltz.q14 | tiltz in Q14 |
| 1 400 | wiir.q2 | newest 5 elements of wiir() in Q2 after weighting filtering |

**Table 9 – Distribution directory disk #4**

| Size | Filename | Remarks |
|---|---|---|
| 14 400 | y2.q5 | y2() array in Q5. A "-" means no update for that vector. |
| 78 700 | zir.q2 | zir() vector in Q2 |
| 14 400 | zirwfir.q2 | newest 5 elements of zirwfir() after memory update of block 10 |
| 14 400 | zirwiir.q2 | newest 5 elements of zirwiir() after memory update of block 10 |

# ITU-T SOFTWARE

| | |
|---|---|
| G.191 (11.96) | Software Tools Library 96 (STL-96) and STL-96 Manual |
| G.722 Appendix II (03.87) | Digital test sequences for the verification of the G.722 64 kbit/s SB-ADPCM 7 kHz codec |
| G.723.1 Annex A (11.96) | C reference code, test signals and test sequences for the fixed point 5.3 and 6.3 kbit/s dual rate speech coder and for the silence compression scheme, version 5.1 |
| G.723.1 Annex B (11.96) | C reference code and test signals for the floating point 5.3 and 6.3 kbit/s dual rate speech coder, version 5.1F |
| G.723.1 Annex C (11.96) | C reference code and test signals for the scalable channel codec, version 3.1 |
| G.726 Appendix II (03.91) | Digital test sequences for the verification of the G.726 40, 32, 24 and 16 kbit/s ADPCM algorithm |
| G.727 Appendix I (03.91) | Digital test sequences for the verification of the G.727 5-, 4-, 3- and 2-bit/sample embedded ADPCM algorithm |
| G.728 Appendix I (07.95) | Programs and test sequences for implementation verification of the algorithm of the G.728 16 kbit/s LD-CELP speech coder |
| G.729 (03.96) | C Source code and test vectors for implementation verification of the G.729 8 kbit/s CS-ACELP speech coder |
| G.729 Annex A (11.96) | C source code and test vectors for implementation verification of the G.729 reduced complexity 8 kbit/s CS-ACELP speech coder |
| G.729 Annex B (10.96) | C source code and test vectors for implementation verification of the algorithm of the G.729 silence compression scheme |
| P.501 (08.96) | Test signals for use in telephonometry |
| P.861 (08.96) | C reference code of Perceptual Speech Quality Measure (PSQM) |
| Q.921 *bis* (03.93) | Abstract test suite for LAPD conformance testing – Part I: basic rate user side |
| Q.931 *bis* (02.95) | PICS and abstract test suite for ISDN DSS 1 layer 3 – Circuit mode, basic call control conformance testing |
| Q.933 *bis* (10.95) | PICS and abstract test suite for frame mode basic call control conformance testing of PVCs – Section I: user and network sides of user-network interface |
| T.24 (11.94) | Standardized digitized image set |
| T.83 (11.94) | Compliance test data for the generic encoder and decoder for the digital compression and coding of continuous-tone still images |

Printed in Switzerland

Geneva, 1998