INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

G.728

Annex G
Corrigendum 1
(02/00)

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

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

Coding of speech at 16 kbit/s using low-delay code excited linear prediction

Annex G: 16 kbit/s fixed point specification

Corrigendum 1

ITU-T Recommendation G.728 – Annex G – Corrigendum 1

(Previously CCITT Recommendation)

ITU-T G-SERIES RECOMMENDATIONS

TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS

INTERNATIONAL TELEPHONE CONNECTIONS AND CIRCUITS	G.100–G.199
INTERNATIONAL ANALOGUE CARRIER SYSTEM	
GENERAL CHARACTERISTICS COMMON TO ALL ANALOGUE CARRIER-TRANSMISSION SYSTEMS	G.200–G.299
INDIVIDUAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON METALLIC LINES	G.300–G.399
GENERAL CHARACTERISTICS OF INTERNATIONAL CARRIER TELEPHONE SYSTEMS ON RADIO-RELAY OR SATELLITE LINKS AND INTERCONNECTION WITH METALLIC LINES	G.400–G.449
COORDINATION OF RADIOTELEPHONY AND LINE TELEPHONY	G.450–G.499
TESTING EQUIPMENTS	
TRANSMISSION MEDIA CHARACTERISTICS	G.600–G.699
DIGITAL TRANSMISSION SYSTEMS	
TERMINAL EQUIPMENTS	G.700–G.799
General	G.700–G.709
Coding of analogue signals by pulse code modulation	G.710–G.719
Coding of analogue signals by methods other than PCM	**G.720–G.729**
Principal characteristics of primary multiplex equipment	G.730–G.739
Principal characteristics of second order multiplex equipment	G.740–G.749
Principal characteristics of higher order multiplex equipment	G.750–G.759
Principal characteristics of transcoder and digital multiplication equipment	G.760–G.769
Operations, administration and maintenance features of transmission equipment	G.770–G.779
Principal characteristics of multiplexing equipment for the synchronous digital hierarchy	G.780–G.789
Other terminal equipment	G.790–G.799
DIGITAL NETWORKS	G.800–G.899
DIGITAL SECTIONS AND DIGITAL LINE SYSTEM	G.900–G.999

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION G.728

CODING OF SPEECH AT 16 kbit/s USING LOW-DELAY CODE EXCITED LINEAR PREDICTION

ANNEX G

16 kbit/s fixed point specification

CORRIGENDUM 1

Source

Corrigendum 1 to Annex G to ITU-T Recommendation G.728 was prepared by ITU-T Study Group 16 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on 17 February 2000.

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.

© ITU 2000

All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the ITU.

Recommendation G.728

CODING OF SPEECH AT 16 kbit/s USING LOW-DELAY CODE EXCITED LINEAR PREDICTION

ANNEX G

16 kbit/s fixed point specification

CORRIGENDUM 1

(Geneva, 2000)

1) Subclause G.1.3.4

Sentences in the second and third paragraphs should be deleted as follows:

G.1.3.4 Division

Division is not used nearly as frequently as addition or multiplication. The only divisions used are scalar floating point divisions. The numerator and denominator are represented in normalized format, as is the quotient. The quotient's NLS is calculated by subtracting the NLS of the denominator from that of the numerator and adding 14. To explain this 14, consider the case where the numerator was slightly larger than the denominator and both had NLS equal 0. The quotient would have NLS equal 14 in this case and would be properly normalized. If the numerator's mantissa is less than the denominator's, then the numerator should be left shifted by 1 bit and its NLS increased by 1 in order to compute the NLS of the quotient. This guarantees that the mantissa of the quotient will be in normalized format.

~~Division occurs within Durbin's recursion, a routine requiring full 16-bit precision in the result. Therefore, approximate division routines are not sufficient.~~ The mantissa of the result must have full 16-bit precision including rounding of the 17-bit result. Pseudo-code for such a division is given below.

~~If either the numerator or denominator is not initially stored in scalar floating point, it must first be converted to this format. The function FLOAT(.) is used in the pseudo-code to represent such conversions. The argument could be either single precision or double precision fixed point.~~

2) Subclause G.2.1

a) Change the last sentence of the second paragraph as follows:

G.2.1 Changes in the backward vector gain adapter (block 20)

NOTE – This subclause refers to 3.8/G.728. Readers should familiarize themselves with 3.8/G.728 before attempting to understand this subclause. The changes outlined in this subclause pertain to the once-per-vector computations for the backward vector gain adpater. Wherever possible, the same notation used in Recommendation G.728 has been used here.

In this subclause we briefly describe the once-per-vector backward vector gain adapter operations in Recommendation G.728 as implemented in floating point. We then describe a mathematically equivalent method which can be more easily and accurately implemented on fixed point processors. Tables for values required by this alternate method are given in ~~the addendum to this annex~~G.5.

b) *For consistency, all figure names should be followed by "/G.728" as follows:*

2) 10 $\log_{10} P[y_j]$, the dB value of the power of the best shape codevector selected from the shape codebook. (In a sense, this is like a conventional predictive coder for the gain, but operated in the logarithmic domain.)

Figure G.1/G.728 shows the block schematic of this mathematically equivalent method. Since there are only 4 possible $|g_i|$ and 128 possible $P[y_j]$ values, we can precompute their dB values and store them in two log-gain tables (blocks 93 and 94 in Figure G.1/G.728).

The delay units 91 and 92 make available the best gain and shape codebook indices chosen in the excitation codebook search of the previous vector. These two indices are used to look up the values of 20 $\log_{10} |g_i|$ and 10 $\log_{10} P[y_j]$ from the log-gain tables in blocks 93 and 94. The 1-sample delay unit 95 holds the previous predicted (and possibly range limited) log-gain $\hat{\delta}\ (n-1)$. The adder 96 adds the outputs of blocks 93, 94, and 95 to produce an unclipped $\hat{\delta}\ (n-1)$ according to equation (G-14). Then, the limiter 97 enforces the inequality in equation (G-9) by clipping the output of the adder 96 at –32 dB if it is less than –32 dB.

The ouput of the limiter 97 is mathematically equivalent to the output of the adder 42 in Figure 6/G.728. Therefore, blocks 43 through 46 in Figure G.1/G.728 are identical to their counterparts in Figure 6/G.728. The operation of the log-gain limiter 98 is similar to the limiter 47 in Figure 6/G.728, except that the allowed range has been shifted down by 32 dB. The adder 99 adds the log-gain offset value of 32 dB, stored in block 41, to the output of the log-gain limiter 98. The resulting log-gain value is then converted to the linear domain by the inverse logarithm calculator 48, which is identical to its counterparts in Figure 6/G.728. This completes the descriptions of the mathematically equivalent method for fixed point implementation.

The equivalent method shown in Figure G.1/G.728 has two important advantages over the original method in Figure 6/G.728.

a) It eliminates the need to calculate the logarithm function (block 40 in Figure 6/G.728). In DSP implementations, the logarithm function is usually calculated using a power series expansion and typically takes a large number of instruction cycles to calculate. Thus, replacing the logarithm calculation by a table look-up could mean a considerable saving in DSP cycles. Also, the table entries can be pre-computed to the maximum desired.

b) It is likely to give more accurate numerical results than the original method when a fixed point processor is used. Due to backward adaptation, there is a feedback loop in the gain adaptation process. In Figure 6/G.728, this feedback loop is very long. It goes from the inverse logarithm calculator 48 to the gain scaling unit 21 (in Figure 2/G.728), and then back to blocks 67, 39, 40 and 42 through 48. The more computations done in this loop, the more likely that numerical errors due to finite precision may accumulate in the feedback loop. This is especially true if the fixed point processor does not always achieve the maximum possible accuracy for the logarithm function. In contrast, the feedback loop in Figure G.1/G.728 is as tight as it can be. Note that the gain scaling unit, the energy and power calculation for $e(n)$, the logarithm calculator, and even the adder for restoring the log-gain offset are now all out of the feedback loop. Except for blocks 43 through 46 that are common in both methods, the feedback loop only involves two limiters and two additions, which can be implemented with very high precision by fixed point processors.

c) "P" should be added at the first column of the second raw and "AA0" should be added at the footnote of the table for variables of Block 50:

The following table lists all variables in the above pseudo-code with their representation format for easy reference.

Variable	Format	Size	Temp/perm	Old/new
AA0, AA1, AA2, P	DP-integer	1	temp	new
ALPHATMP	SFL	1	temp	old
ATMP	Q13/Q14/Q15	51	perm	old
IB	Integer	1	temp	old
ILLCOND	Logical	1	perm	new
ILLCONDP	logical	1	perm	new
IP	integer	1	temp	old
LP	integer	1	temp	new
MH	integer	1	temp	old
MINC	integer	1	temp	old
NLSATMP	integer	1	temp	new
NRS	integer	1	temp	new
NUM	integer	1	temp	new
RC	Q15	1	temp	new
RC1	Q15	1	temp	new
RTMP	Q15	51	perm	old
SIGN	integer	1	temp	new

SFL	16-bit scalar floating point
DP-integer	32-bit register such as accumulator or product registers (AA0, AA1, AA2 & P)
Integer	16-bit integer
Q13/Q14/Q15	16-bit integer with one of these representations

3) Subclause G.3.2

An equation of the pseudo-code should be corrected as follows:

```
L = 6 – K
J = LPC
AA0 = 0
For LL = 1, ..., L, do the next 3 lines
       AA0 = AA0 – STATELPC(J) * A(J + 1)  |  Multiply – add
       STATELPC(J) = STATELPC(J – 1)       |  Memory shift
       J = J – 1
NLS = NLSSTATE(I) – NLSSTATE(11)
AA1 = AA0 >> NLS

For I = 2, ..., 10, do the next 8 lines
       AA0 = 0
       For LL = 1, 2, ..., IDIM, do the next 3 lines
          AA0 = AA0 – STATELPC(J) * A(J + 1)
          STATELPC(J) = STATELPC(J – 1)   | STATELPC(0) = garbage if J = 1;
          it is OK
          J = J – 1
       NLS = NLSSTATE(I) – NLSSTATE(11)
       AA0 = AA0 >> NLS                   |  Shift to align
       AA1 = AA1 + AA0

If K = 1, go to SHIFT2
L = K – 1
AA0 = 0
For LL = 1, 2, ..., L, do the next 3 lines
       AA0 = AA0 – STATELPC(J) * A(J + 1)
```

4) Subclause G.3.12

a) An unused argument "N3" of FINDNLS should be removed as follows:

G.3.12 Block 36 – Pseudo-code for hybrid windowing module

In this subclause both the floating point and fixed point pseudo-code for block 36 are given. First, the floating point pseudo-code is presented.

```
For N = 1, 2, ..., N2, do the next line
                        SBW(N) = SBW(N + NFRSZ) | Shift the old signal buffer
For N = 1, 2, ..., NFRSZ, do the next line
                        SBW(N2 + N) = STMP(N)   | SBW(N3) is the newest sample
                                                | All SBW are Q2 and represented
                                                | in 15 bits precision

      Call FINDNLS(SBW, ~~N3,~~ N3, 14, NLS)    | Find the amount of left shifts
                                                | needed in the next loop to get
                                                | 2 bits of headroom. We do not
                                                | really need to do the scaling
                                                | We just use NLS
```

b) In the table for variables, the size of "R" should be "11" instead of "1".

Variable	Format	Size	Temp/perm	Old/new
NLS	integer	1	temp	new
NLSREXPW	integer	1	perm	new
NLSTMP	integer	1	temp	new
REXPW	BFL	11	perm	old
R	BFL	11	perm	old
SBW	Q2	60	perm	old
STMP	Q2	20	perm	old
WS	BFL	60	temp	old
BFL Block floating point Integer 16-bit integer				

5) Subclause G.3.14

An unused argument "N3" of FINDNLS should be removed as follows:

G.3.14 Block 43 – Hybrid windowing module

In this subclause both the floating point and fixed point pseudo-code for block 43 are given. First, the floating point pseudo-code is presented.

```
For N = 1, 2, ..., N2, do the next line
            SBLG(N) = SBLG(N + NUPDATE)   | Shift the old signal buffer
For N = 1, 2, ..., NUPDATE, do the next line
            SBLG(N2 + N) = GTMP(N)        | SBLG(N3) is the newest sample
                                          | All SBLG are Q9 and represented
                                          | in 16-bits precision
Call FINDNLS(SBLG, ~~N3,~~ N3, 14, NLS)   | Find the amount of left shifts
NLSTMP = NLS – 1                          | needed in the next loop for 2 bits
                                          | of headroom later
```

6) Subclause G.3.1.7

In a comment line, "NLSWS" should be changed to "NLSTMP":

G.3.17 Block 49 – Hybrid window module for synthesis filter

```
For N = 1, 2, ..., N2, do the next line
    SB(N) = SB(N + NFRSZ)                        |  Shift old part of buffer SB
For N = 1, 2, ..., N6, do the next line
    NLSSB(N) = NLSSB(N + N5)                     |  Shift old NLSSB
For N = 1, 2, ..., NFRSZ, do the next line
    SB(N2 + N) = STTMP(N)                        |  Shift in new part of SB
For N = 1, 2, ..., N5, do the next line
    NLSSB(N6 + N) = NLSSTTMP(N)                  |  Shift in new NLSSB
                                                 |  Now find the minimum NLSSB,
                                                 |  this determines NLSWSTMP
NLSTMP  Min{NLSSB(1), NLSSB(2), ..., NLSSB(N4)}
```

7) Subclause G.3.18

The negative sign "–" should be added as follows:

G.3.18 HWMCORE – Core of hybrid window module

```
      For I = 1, 2, ..., LPO, do the next 11 lines
           AA0 = 0                                |  Compute recursive part
                                                  |  of RREC(I + 1)
           For N = LPO + 1, ..., N1, do the next 2 lines
                P = WS(N) * WS(N – I)
                AA0 = AA0 + P
           AA0 = AA0 >> 1
           AA1 = RREC(I + 1) << NLSATT            |  Scale RREC by 3/4 or 1/2
           AA1 = -AA1 + (RREC(I + 1) << 16)       |
           AA1 = AA1 >> IR
           AA0 = AA0 + AA1
           AA0 = AA0 << NLSRE
           RREC(I + 1) = RND(AA0)                 |  Upper 16 bits of AA0 saved
      Go to FIN_RECUR
```

8) Subclause G.3.20

The variable "AA0" should be changed to "AA1" as follows:

G.3.20 Blocks 71 and 72 – Long-term and short-term postfilters

Blocks 71 and 72 are combined in order to preserve the precision of the intermediate variable TEMP which was passed between them in the floating point pseudo-code. The floating point pseudo-code for both of these blocks is given first.

```
                                               |  Now perform IIR part of filter
      For J = 10, 9, ..., 3, 2, do the next 2 lines
         AA1 = AA1 – STPFIIR(J) * AP(J + 1)    |  AP is Q14, STPFIIR(J) is Q2
         STPFIIR(J) = STPFIIR(J – 1)
      AA1 = AA1 – STPFIIR(1) * AP(2)
         AA0 = AA01 >> 14
                                               |  Now check for saturation
      If AA0 > 32767, set AA0 = 32767
      If AA0 < –32768, set AA0 = –32768
      STPFIIR(1) = AA0
                                               |  Now do spectral compensation
                                               |  tilt filter
```

9) Subclause G.4

For consistency, all table names should be followed by "/G.728" as follows:

G.4 LD-CELP internal variable representations

In this subclause updated versions of Tables 1/G.728 and 2/G.728 are presented. Table G.1/G.728 is a shortened version of Table 1/G.728. It lists only constants which are not inherently integers and are not given elsewhere in the Recommendation. The Equivalent Symbol and Initial Value entries in Table 1/G.728 have been deleted in order to leave space for the Fixed Point Format and representation required for each variable. Table G.2/G.728 is the integer version of Table 2/G.728. The same column has also been deleted from Table 2/G.728 in order to present the fixed point format. Several new variables are listed which relate only to the fixed point specification.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	**Transmission systems and media, digital systems and networks**
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2000