COVERING NOTE

GENERAL SECRETARIAT OF THE INTERNATIONAL TELECOMMUNICATION UNION

Geneva, 26 May 2014

ITU – TELECOMMUNICATION STANDARDIZATION SECTOR

Subject: Erratum 1 (05/2014) to Recommendation ITU-T G.728 (2012), ***Coding of speech at 16 kbit/s using low-delay code excited linear prediction***

1) The codevector components associated to channel index 36 and channel index 42 contain wrong signs. Correct the signs of the codevector components indicated with underlining as shown below:

Channel index	Codevector components				
....	....	....	....	....	....
36	-3837	-1831	6397	2545	-2848
....	....	....	....	....	....
82	-45	1198	2160	-1449	2203
....	....	....	....	....	....

2) Table G.5 contains extraneous characters. Correct Table G.5, Integer values of gain codebook related arrays, to read as follows (underlining indicates values that are being rectified):

Table G.5 – Integer values of gain codebook related array

Array index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
GQ (Q13)	4224	7392	12936	22638	-4224	-7392	-12936	-22638
GB (Q13)	5808	10164	17787	*	-5808	-10164	-17787	*
G2 (Q12)	4224	7392	12936	22638	-4224	-7392	-12936	-22638
GSQ (Q11)	545	1668	5107	15640	545	1668	5107	15640
* Can be any arbitrary value (not used).								

Union internationale des télécommunications
Place des Nations 1211 GENEVE 20
Suisse – Switzerland – Suiza – Швейцария – 瑞士 – سويسرا

3) In clause G.2.2, some of the indentation in the main loop has been lost. Correct the indentation of the Recursion module as shown below, where additional clarification has been added to identify loop boundaries, and scratch variable IP has been replaced with scratch variable IA to avoid confusion with pointer IP in the main code.

```
RECURSION:

  For MINC = MINC0 + 1, MINC0 + 2, ..., LPC, do the following indented lines
    AA0 = 0
    For IA = 2, 3, ..., MINC, do the next 3 lines
      N1 = MINC - IA + 2
      P = RTMP(N1) * ATMP(IA)
      AA0 = AA0 + P                        | 32 bits for SUM
    AA0 = AA0 << 1

    AA0 = AA0 << NRS
    AA1 = RTMP(MINC + 1) << 16
    AA0 = AA0 + AA1                        |
    SIGN = RND(AA0)                        | Save high word sign
    NUM = SIGN
    If NUM < 0, set NUM = -NUM
    If NUM ≥ ALPHATMP, go to FAILED             |
    Call SIMPDIV(NUM, ALPHATMP, AA0)            | Divide to get RC
    AA2 = AA0 << 15                        | AA2 stores 17-bit RC
    RC = RND(AA2)
    If SIGN > 0, set RC = -RC

                                  | Now update ALPHATMP
    AA1 = ALPHATMP << 16
    P = RC * SIGN
    AA1 = AA1 + (P << 1)
    If AA1 ≤ 0, go to FAILED
    ALPHATMP = RND(AA1)

    MH = MINC/2 + 1               | Fractional part of MINC/2 truncated;
| MH = integer
| Begin to update predictor
| coefficients
    For IA = 2, 3, 4, ..., MH, do the following 24 lines
      IB = MINC - IA + 2
      AA0 = ATMP(IA) <<16                       | Load AA0 high word
      P = RC * ATMP(IB)                         | Q15/16 RC, so << 1
      AA0 = AA0 + (P << 1)
      If AA0 overflowed, then do the following 5 lines
        NRS = NRS + 1
        For LP = 2, 3, ..., MINC, set ATMP(LP) = ATMP(LP) >> 1
        AA0 = ATMP(IA) <<16                | First re-scale ATMP
        P = RC * ATMP(IB)                       | Next re-calculate
        AA0 = AA0 + (P << 1)                    | overflowed AA0

      AA1 = ATMP(IB) <<16
      P = RC * ATMP(IA)
      AA1 = AA1 + (P << 1)
      If AA1 overflowed, then do the following 8 lines
        NRS = NRS + 1
        For LP = 2, 3, ..., MINC, set ATMP(LP) = ATMP(LP) >> 1
        AA0 = ATMP(IA) <<16                | First re-scale ATMP(IA)
        P = RC * ATMP(IB)                       | Next re-calculate AA0
        AA0 = AA0 + (P << 1)                    |
        AA1 = ATMP(IB) << 16                    | Next re-scale ATMP(IB)
        P = RC * ATMP(IA)                       | Next re-calculate
        AA1 = AA1 + (P << 1)                    | overflowed AA1

      ATMP(IA) = RND(AA0)
```

```
      ATMP(IB) = RND(AA1)

                                      | Update ATMP(MINC + 1)
    AA0 = AA2 >> NRS | AA2 contains 17-bit RC
    AA0 = RND(AA0) | Output in low word of AA0
    If SIGN > 0, set AA0 = -AA0
    ATMP(MINC + 1) = AA0 | Low word stored in ATMP

Repeat the above indented lines for the next MINC
```