Technical Specification Group Services and System Aspects **TSGS#21(03)0447**
Meeting #21, Frankfurt, Germany
22-25 September 2003

Source: **TSG-SA WG4**

Title: CR to TS 26.204 - Possible decoder LPC coefficients overflow (Release 5)

Document for: **Approval**

Agenda Item: **7.4.3**

The following CR, agreed at the TSG-SA WG4 meeting #28, is presented to TSG SA #21 for approval.

Spec	CR	Rev	Phase	Subject	Cat	Vers	WG	Meeting	S4 doc
26.204	008		Rel-5	Possible decoder LPC coefficients overflow	F	5.1.0	S4	TSG-SA WG4#28	S4-030635

3GPP TSG-SA4 Meeting #28
Erlangen, Germany, 1-5 Sep 2003

Tdoc ⌘S4-030635

CR-Form-v7

CHANGE REQUEST

⌘ **26.204** **CR** **008** ⌘ **rev** **-** ⌘ Current version: **5.1.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: UICC apps⌘ [ ] ME [X] Radio Access Network [ ] Core Network [X]

Title:	⌘	Possible decoder LPC coefficients overflow
Source:	⌘	TSG SA WG4
Work item code:	⌘	AMRWB-FP
Date:	⌘	22/9/2003
Category:	⌘	**F**
Release:	⌘	Rel-5

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:

2	*(GSM Phase 2)*
R96	*(Release 1996)*
R97	*(Release 1997)*
R98	*(Release 1998)*
R99	*(Release 1999)*
Rel-4	*(Release 4)*
Rel-5	*(Release 5)*
Rel-6	*(Release 6)*

Reason for change:	⌘	AMR-WB decoder can produce unstable output during DTX-operation.
Summary of change:	⌘	Conversion from ISP to LPC coefficients is changed, so that LPC coefficients cannot overflow in decoder comfort noise generation. Synthesis is changed to support scaled LPC coefficients.
Consequences if not approved:	⌘	Decoder synthesis filter may be unstable causing uncontrolled ouput when DTX is used.

Clauses affected:	⌘	dec_lpc.c, dec_lpc.h, dec_main.c and dec_util.c

	Y	N		
Other specs affected: ⌘		X	Other core specifications	⌘
		X	Test specifications	
		X	O&M Specifications	

Other comments:	⌘	For background see Tdoc S4-030635 26.173 Possible decoder LPC coefficients overflow

Changes to the C-code:

1. How the code is changed in the file *dec_lpc.c*

Lines 6-7:

```
#include <math.h>

#include "typedef.h"
#include "dec_util.h"
```

Lines 246-336:

```
void D_LPC_isp_a_conversion(Word16 isp[], Word16 a[], Word32 adaptive_scaling,
                            Word16 m)
{
   Word32 j, i, nc, tmax, q, q_sug, r;
   Word32 f1[NC16k + 1], f2[NC16k];
   Word32 t0;
   Word16 hi, lo;

   nc = m >> 1;

   if(nc > 8)
   {
      D_LPC_isp_pol_get(&isp[0], f1, nc, 1);

      for(i = 0; i <= nc; i++)
      {
         f1[i] = (f1[i] << 2);
      }
   }
   else
   {
      D_LPC_isp_pol_get(&isp[0], f1, nc, 0);
   }

   if(nc > 8)
   {
      D_LPC_isp_pol_get(&isp[1], f2, nc - 1, 1);

      for(i = 0; i <= nc - 1; i++)
      {
         f2[i] = (f2[i] << 2);
      }
   }
   else
   {
      D_LPC_isp_pol_get(&isp[1], f2, nc - 1, 0);
   }

   /*
    *  Multiply F2(z) by (1 - z^-2)
    */
   for(i = nc - 1; i > 1; i--)
   {
      f2[i] = f2[i] - f2[i - 2];   /* f2[i] -= f2[i-2]; */
   }

   /*
    *  Scale F1(z) by (1+isp[m-1]) and F2(z) by (1-isp[m-1])
    */
   for(i = 0; i < nc; i++)
   {
      /* f1[i] *= (1.0 + isp[M-1]); */
      D_UTIL_l_extract(f1[i], &hi, &lo);
      t0 = D_UTIL_mpy_32_16(hi, lo, isp[m - 1]);
      f1[i] = f1[i] + t0;

      /* f2[i] *= (1.0 - isp[M-1]); */
      D_UTIL_l_extract(f2[i], &hi, &lo);
      t0 = D_UTIL_mpy_32_16(hi, lo, isp[m - 1]);
      f2[i] = f2[i] - t0;
   }
```

```
   /*
    *  A(z) = (F1(z)+F2(z))/2
    *  F1(z) is symmetric and F2(z) is antisymmetric
    */

   /* a[0] = 1.0; */
   a[0] = 4096;
   tmax = 1;


   for(i = 1, j = m - 1; i < nc; i++, j--)
   {
      /* a[i] = 0.5*(f1[i] + f2[i]); */
      t0 = f1[i] + f2[i];   /* f1[i] + f2[i] */
      tmax |= labs(t0);

      a[i] = (Word16)((t0 + 0x800) >> 12);   /* from Q23 to Q12 and * 0.5 */

      /* a[j] = 0.5*(f1[i] - f2[i]); */
      t0 = (f1[i] - f2[i]);   /* f1[i] - f2[i] */
      tmax |= labs(t0);

      a[j] = (Word16)((t0 + 0x800) >> 12);   /* from Q23 to Q12 and * 0.5 */
   }

   /* rescale data if overflow has occured and reprocess the loop */

    if (adaptive_scaling)
    {
      q = 4 - D_UTIL_norm_l(tmax);        /* adaptive scaling enabled */
    }
    else
    {
      q = 0;                              /* adaptive scaling disabled */
    }

    if (q > 0)
    {
      q_sug = 12 + q;
      r = 1 << (q_sug - 1);

      for (i = 1, j = m - 1; i < nc; i++, j--)
        {
          /* a[i] = 0.5*(f1[i] + f2[i]); */
          t0 = f1[i] + f2[i];           /* f1[i] + f2[i]              */
          a[i] = (Word16)((t0 + r) >> q_sug); /* from Q23 to Q12 and * 0.5 */

          /* a[j] = 0.5*(f1[i] - f2[i]); */
          t0 = f1[i] - f2[i];           /* f1[i] - f2[i]              */
          a[j] = (Word16)((t0 + r) >> q_sug); /* from Q23 to Q12 and * 0.5 */
        }
      a[0] = (Word16)(a[0] >> q);
    }
    else
    {
      q_sug = 12;
      r = 1 << (q_sug - 1);
      q     = 0;
    }


   /* a[NC] = 0.5*f1[NC]*(1.0 + isp[M-1]); */
   D_UTIL_l_extract(f1[nc], &hi, &lo);
   t0 = D_UTIL_mpy_32_16(hi, lo, isp[m - 1]);
   t0 = (f1[nc] + t0);f1[nc] + t0;
   a[nc] = (Word16)((t0 + 0x800) >> 12);r) >> q_sug);  /* from Q23 to Q12 and * 0.5 */

   /* a[m] = isp[m-1]; */
   a[m] = (Word16)((isp[m - 1] + 0x4) >> 3);>> (2 + q)) + 1); /* from Q15 to Q12 */
   a[m] = (Word16)(a[m] >> 1);


   return;
}
```

Lines 620-646:

```
void D_LPC_int_isp_find(Word16 isp_old[], Word16 isp_new[],
                        const Word16 frac[], Word16 Az[])
{
```

```
   Word32 tmp, i, k, fac_old, fac_new;
   Word16 isp[M];

   for(k = 0; k < 3; k++)
   {
      fac_new = frac[k];
      fac_old = (32767 - fac_new) + 1;   /* 1.0 - fac_new */

      for(i = 0; i < M; i++)
      {
         tmp = isp_old[i] * fac_old;
         tmp += isp_new[i] * fac_new;
         isp[i] = (Word16)((tmp + 0x4000) >> 15);
      }

      D_LPC_isp_a_conversion(isp, Az, 0, M);
      Az += MP1;
   }

   /* 4th subframe: isp_new (frac=1.0) */
   D_LPC_isp_a_conversion(isp_new, Az, 0, M);

   return;
}
```

2. How the code is changed in the file *dec_lpc.h*

Lines 11-21:

```
void D_LPC_isf_noise_d(Word16 *indice, Word16 *isf_q);
void D_LPC_isf_isp_conversion(Word16 isf[], Word16 isp[], Word16 m);
void D_LPC_isp_a_conversion(Word16 isp[], Word16Word32 a[], Word16 m);
void D_LPC_a_weight(Word16 a[], Word16D_LPC_a_weight(Word32 a[], Word32 ap[], Word16 gamma, Word16
m);
void D_LPC_isf_2s3s_decode(Word16 *indice, Word16 *isf_q, Word16* past_isfq,
                           Word16 *isfold, Word16 *isf_buf, Word16 bfi);
void D_LPC_isf_2s5s_decode(Word16 *indice, Word16 *isf_q, Word16 *past_isfq,
                           Word16 *isfold, Word16 *isf_buf, Word16 bfi);
void D_LPC_int_isp_find(Word16 isp_old[], Word16 isp_new[],
                        const Word16 frac[], Word16Word32 Az[]);
void D_LPC_isf_extrapolation(Word16 HfIsf[]);
```

3. How the code is changed in the file *dec_lpc.h*

Line 13:

```
void D_LPC_isp_a_conversion(Word16 isp[], Word16 a[], Word32 adaptive_scaling,

                            Word16 m);
```

4. How the code is changed in the file *dec_main.c*

Lines 327-359:

```
if(newDTXState != SPEECH) /* CNG mode */
   {
      /*
       * increase slightly energy of noise below 200 Hz
       * Convert ISFs to the cosine domain
       */
      D_LPC_isf_isp_conversion(isf, ispnew, M);
      D_LPC_isp_a_conversion(ispnew, Aq, 1, M);
      memcpy(isf_tmp, st->mem_isf, M * sizeof(Word16));

      for(i_subfr = 0; i_subfr < L_FRAME; i_subfr += L_SUBFR)
      {
         j = (i_subfr >> 6);

         for(i = 0; i < M; i++)
         {
            L_tmp = (isf_tmp[i] * (32767 - D_ROM_interpol_frac[j])) << 1;
            L_tmp = L_tmp + ((isf[i] * D_ROM_interpol_frac[j]) << 1);
            HfIsf[i] = (Word16)((L_tmp + 0x8000) >> 16);
```

```
         }

         D_UTIL_dec_synthesis(Aq, &exc2[i_subfr], 0, &synth16k[i_subfr * 5 /4],
            (Word16) 1, HfIsf, mode, newDTXState, bfi, st);
      }

      /* reset speech coder memories */
      D_MAIN_reset(st, 0);
      memcpy(st->mem_isf, isf, M * sizeof(Word16));
      st->mem_bfi = bfi;
      st->dtx_decSt->mem_dtx_global_state = (UWord8)newDTXState;

      return(0);
   }
```

5. How the code is changed in the file *dec_util.c*

Lines 659-697:

```
static void D_UTIL_synthesis_32(Word16 a[], Word16 m, Word16 exc[],
                                Word16 Qnew, Word16 sig_hi[], Word16 sig_lo[],
                                Word16 lg)
{
   Word32 i, j, a0, s;
   Word32 tmp, tmp2;

   /* See if a[0] is scaled */
   s = D_UTIL_norm_s((Word16)a[0]) - 2;


   a0 = a[0] >> (4 + Qnew);   /* input / 16 and >>Qnew */

   /* Do the filtering. */
   for(i = 0; i < lg; i++)
   {
      tmp = 0;

      for(j = 1; j <= m; j++)
      {
         tmp -= sig_lo[i - j] * a[j];
      }

      tmp = tmp >> (15 - 4);   /* -4 : sig_lo[i] << 4 */

      tmp2 = exc[i] * a0;

      for(j = 1; j <= m; j++)
      {
         tmp2 -= sig_hi[i - j] * a[j];
      }

      tmp += tmp2 << 1;
      tmp <<= s;


      /* sig_hi = bit16 to bit31 of synthesis */
      sig_hi[i] = (Word16)(tmp >> 13);

      /* sig_lo = bit4 to bit15 of synthesis */
      sig_lo[i] = (Word16)((tmp  >> 1) - (sig_hi[i] * 4096));
   }

   return;
}
```

Lines 972-1006:

```
static void D_UTIL_synthesis(Word16 a[], Word16 m, Word16 x[], Word16 y[],
                             Word16 lg, Word16 mem[], Word16 update)
{
   Word32 i, j, tmp, s;
   Word16 y_buf[L_SUBFR16k + M16k], a0;
   Word16 *yy;

   yy = &y_buf[m];

   /* See if a[0] is scaled */
   s = D_UTIL_norm_s(a[0]) - 2;
```

```
   /* copy initial filter states into synthesis buffer */
   memcpy(y_buf, mem, m * sizeof(Word16));

   a0 = (Word16)(a[0] >> 1);   /* input / 2 */

   /* Do the filtering. */
   for(i = 0; i < lg; i++)
   {
      tmp = x[i] * a0;

      for(j = 1; j <= m; j++)
      {
         tmp -= a[j] * yy[i - j];
      }
      tmp <<= s;


      y[i] = yy[i] = (Word16)((tmp + 0x800) >> 12);
   }

   /* Update memory if required */
   if(update)
   {
      memcpy(mem, &yy[lg - m], m * sizeof(Word16));
   }

   return;
}
```

Lines 1310-1316:

```
if((mode <= MODE_7k) & (newDTXState == SPEECH))
   {
      D_LPC_isf_extrapolation(HfIsf);
      D_LPC_isp_a_conversion(HfIsf, HfA, 0, M16k);
      D_LPC_a_weight(HfA, Ap, 29491, M16k);   /* fac=0.9 */
      D_UTIL_synthesis(Ap, M16k, HF, HF, L_SUBFR16k, st->mem_syn_hf, 1);
   }
```