Technical Specification Group Services and System Aspects ***TSGS#19(03)0088***
Meeting #19, Birmingham, UK, 17 - 20 March 2003

Source: **TSG-SA WG4**

Title: CRs to TS 26.104 - Correction to floating-point implementation for AMR (R99, Rel4 and Rel 5), and MMS compatible input/output option (Rel 5)

Document for: **Approval**

Agenda Item: **7.4.3**

The following CRs, agreed at the TSG-SA WG4 meeting #25, are presented to TSG SA #19 for approval.

Spec	CR	Rev	Phase	Subject	Cat	Vers	WG	Meeting	S4 doc
26.104	021	1	Rel-5	MMS compatible i/o format	F	5.0.0	S4	TSG-SA WG4#25	S4-030086
26.104	022		R99	Correction to floating-point implementation of sp_dec.c	F	3.4.0	S4	TSG-SA WG4#25	S4-030037
26.104	023		Rel-4	Correction to floating-point implementation of sp_dec.c	A	4.3.0	S4	TSG-SA WG4#25	S4-030038
26.104	024		Rel-5	Correction to floating-point implementation of sp_dec.c	A	5.0.0	S4	TSG-SA WG4#25	S4-030039

3GPP TSG-SA4 Meeting #25
San Francisco, USA, 20-24 January 2003

Tdoc ⌘S4-030086

CR-Form-v7

CHANGE REQUEST

⌘ **26.104** **CR** **021** ⌘**rev** **1** ⌘ Current version: **5.0.0** ⌘

*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 ☒ Radio Access Network ☐ Core Network ☒

Title:	⌘	MMS compatible i/o format option
Source:	⌘	TSG SA WG4
Work item code:	⌘	AMR
Date:	⌘	18/03/2003
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:	⌘	Modifications proposed by this document enable usage of AMR floating-point to encode and decode files according to the AMR MIME file storage format, which is used e.g. by the MMS service.
Summary of change:	⌘	New input/output format option.
Consequences if not approved:	⌘	Codec can not operate with bitstreams in AMR MIME file storage format.

Clauses affected:	⌘	2, 6.3, encoder.c, interf_enc.c, decoder.c, interf_dec.c, interf_rom.h

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

Other comments:	⌘	

Changes to the specification document:

2. Normative references

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TS 26.074: "AMR Speech Codec; Test sequences".

[2] 3GPP TS 26.090: "AMR Speech Codec; Speech transcoding".

[3] 3GPP TS 26.091: "AMR Speech Codec; Substitution and muting of lost frames".

[4] 3GPP TS 26.092: "AMR Speech Codec; Comfort noise aspects".

[5] 3GPP TS 26.093: "AMR Speech Codec; Source controlled rate operation".

[6] 3GPP TS 26.094: "AMR Speech Codec; Voice Activity Detection".

[7] 3GPP TS 26.073: "ANSI-C code for the Adaptive Multi Rate speech codec".

[8] 3GPP TS 26.101: "AMR Speech Codec Frame Structure".

[9] RFC 3267 “A Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs, June 2002.

6.3 Parameter bitstream file (encoder output / decoder input)

The files produced by the speech encoder/expected by the speech decoder contain an arbitrary number of frames in the format described in RFC 3267 [9], sections 5.1 and 5.3.

By using preprocessor definition encoder/decoder can optionally use AMR Interface Format 2. The format is described in TS 26.101 [8] Annex A.

By using another preprocessor definition encoder/decoder can optionally use format compatible with the existing AMR fixed-point C-code. Frame format is following.

FRAME_TYPE	B1	B2	…	B244	MODE_INFO	*unused1*	…	*unused4*

Each box corresponds to one `Word16` value in the bitstream file, for a total of 250 words or 500 bytes per frame. The fields have the following meaning:

FRAME_TYPE transmit frame type, which is one of
TX_SPEECH (0x0000)
TX_SID_FIRST (0x0001)
TX_SID_UPDATE (0x0002)
TX_NO_DATA (0x0003)

B0…B244 **speech encoder parameter bits (i.e. the bitstream itself). Each B*x* either has the value 0x0000 or 0x0001. Only mode MR122 really uses all 244 bits; for the other modes, only the first *n* bits are used (35 ≤ n ≤ 204). The remaining bits are unused (written as 0x0000)**

MODE_INFO **encoding mode information, which is one of**
MR475 (0x0000)
MR515 (0x0001)
MR59 (0x0002)
MR67 (0x0003)
MR74 (0x0004)
MR795 (0x0005)
MR102 (0x0006)
MR122 (0x0007)

unused1…4 **unused, written as 0x0000**

As indicated in section 6.1 above, the byte order depends on the host architecture.

Changes to the c source-code:

Changes in file *encoder.c*

Lines 29 – 33:

```
#ifndef ETSI
#ifndef IF2
#define AMR_MAGIC_NUMBER "#!AMR\n"
#endif
#endif
```

Lines 188 – 193:

```
#ifndef ETSI
#ifndef IF2
   /* write magic number to indicate single channel AMR file storage format */
     bytes = fwrite(AMR_MAGIC_NUMBER, sizeof(char), strlen(AMR_MAGIC_NUMBER),
file_encoded);
#endif
#endif
```

Lines 225 – 233:

```
#ifndef ETSI
#ifdef IF2
   fprintf ( stderr, "\n%s%i%s%i%s\n", "Frame structure AMR IF2: ", frames, "
frames, ", bytes, " bytes.");
#else
   fprintf ( stderr, "\n%s%i%s%i%s\n", "Frame structure AMR MIME file storage
format: ", frames, " frames, ", bytes, " bytes.");
#endif
#else
   fprintf ( stderr, "\n%s%i%s\n", "Frame structure AMR ETSI: ", frames, "
frames. ");
#endif
```

Changes in file *interf_enc.c*

Lines 181 – 545

```
#else
```

```
#ifndef IF2

/*
 * EncoderMMS
 *
 *
 * Parameters:
 *    mode                  I: AMR mode
 *    param                 I: Encoder output parameters
 *    stream                O: packed speech frame
 *    frame_type            I: frame type (DTX)
 *    speech_mode           I: speech mode (DTX)
 *
 * Function:
 *    Pack encoder output parameters to octet structure according
 *    importance table and AMR file storage format according to
 *    RFC 3267.
 * Returns:
 *    number of octets
 */
static int EncoderMMS( enum Mode mode, Word16 *param, UWord8 *stream, enum
      TXFrameType frame_type, enum Mode speech_mode )
{
   Word32 j = 0, k;
   Word16 *mask;

   memset(stream, 0, block_size[mode]);

   *stream = toc_byte[mode];
   stream++;

   if ( mode == 15 ) {
      return 1;
   }
   else if ( mode == MRDTX ) {
      mask = order_MRDTX;

      for ( j = 1; j < 36; j++ ) {
         if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
         mask += 2;

         if ( j % 8 )
            *stream <<= 1;
         else
            stream++;
      }

      /* add SID type information */
      if ( frame_type == TX_SID_UPDATE )
         *stream += 0x01;
      *stream <<= 3;

      /* speech mode indication */
      *stream += ( unsigned char )(speech_mode & 0x0007);

    *stream <<= 1;

      /* don't shift at the end of the function */
      return 6;
   }
   else if ( mode == MR475 ) {
      mask = order_MR475;

      for ( j = 1; j < 96; j++ ) {
         if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
         mask += 2;

         if ( j % 8 )
            *stream <<= 1;
```

```
        else
            stream++;
    }
  }
  else if ( mode == MR515 ) {
    mask = order_MR515;

    for ( j = 1; j < 104; j++ ) {
        if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
        mask += 2;

        if ( j % 8 )
            *stream <<= 1;
        else
            stream++;
    }
  }
  else if ( mode == MR59 ) {
    mask = order_MR59;

    for ( j = 1; j < 119; j++ ) {
        if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
        mask += 2;

        if ( j % 8 )
            *stream <<= 1;
        else
            stream++;
    }
  }
  else if ( mode == MR67 ) {
    mask = order_MR67;

    for ( j = 1; j < 135; j++ ) {
        if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
        mask += 2;

        if ( j % 8 )
            *stream <<= 1;
        else
            stream++;
    }
  }
  else if ( mode == MR74 ) {
    mask = order_MR74;

    for ( j = 1; j < 149; j++ ) {
        if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
        mask += 2;

        if ( j % 8 )
            *stream <<= 1;
        else
            stream++;
    }
  }
  else if ( mode == MR795 ) {
    mask = order_MR795;

    for ( j = 1; j < 160; j++ ) {
        if ( param[ * mask] & *( mask + 1 ) )
            *stream += 0x01;
        mask += 2;

        if ( j % 8 )
            *stream <<= 1;
        else
```

```
            stream++;
        }
    }
    else if ( mode == MR102 ) {
        mask = order_MR102;

        for ( j = 1; j < 205; j++ ) {
            if ( param[ * mask] & *( mask + 1 ) )
                *stream += 0x01;
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
    }
    else if ( mode == MR122 ) {
        mask = order_MR122;

        for ( j = 1; j < 245; j++ ) {
            if ( param[ * mask] & *( mask + 1 ) )
                *stream += 0x01;
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
    }

    /* shift remaining bits */
    if ( k = j % 8 )  *stream <<= ( 8 - k );
    return( (int)block_size[mode] );
}

#else

/*
 * Encoder3GPP
 *
 *
 * Parameters:
 *    mode                  I: AMR mode
 *    param                 I: Encoder output parameters
 *    stream                O: packed speech frame
 *    frame_type            I: frame type (DTX)
 *    speech_mode           I: speech mode (DTX)
 *
 * Function:
 *    Pack encoder output parameters to octet structure according
 *    importance table.
 * Returns:
 *    number of octets
 */
static int Encoder3GPP( enum Mode mode, Word16 *param, UWord8 *stream, enum
      TXFrameType frame_type, enum Mode speech_mode )
{
   Word32 j = 0;
   Word16 *mask;

   memset(stream, 0, block_size[mode]);

   if ( mode == 15 ) {
      *stream = 0xF;
      return 1;
   }
   else if ( mode == MRDTX ) {
      mask = order_MRDTX;
      *stream = 0x40;
```

```
   for ( j = 5; j < 40; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }

   /* add SID type information */
   if ( frame_type == TX_SID_UPDATE )
      *stream += 0x80;
   stream++;

   /* speech mode indication */
   *stream = ( unsigned char )speech_mode;

   /* don't shift at the end of the function */
   return 6;
}
else if ( mode == MR475 ) {
   mask = order_MR475;
   *stream = 0;

   for ( j = 5; j < 100; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR515 ) {
   mask = order_MR515;
   *stream = 0x8;

   for ( j = 5; j < 108; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR59 ) {
   mask = order_MR59;
   *stream = 0x10;

   for ( j = 5; j < 123; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR67 ) {
   mask = order_MR67;
   *stream = 0x18;
```

```
   for ( j = 5; j < 139; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR74 ) {
   mask = order_MR74;
   *stream = 0x20;

   for ( j = 5; j < 153; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR795 ) {
   mask = order_MR795;
   *stream = 0x28;

   for ( j = 5; j < 164; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR102 ) {
   mask = order_MR102;
   *stream = 0x30;

   for ( j = 5; j < 209; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
}
else if ( mode == MR122 ) {
   mask = order_MR122;
   *stream = 0x38;

   for ( j = 5; j < 249; j++ ) {
      if ( param[ * mask] & *( mask + 1 ) )
         *stream += 0x80;
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
```

```
   }

   /* shift remaining bits */
   *stream >>= ( 8 - j % 8 );
   return( (int)block_size[mode] );
}
#endif
#endif
```

Lines 725 – 739

```
#ifndef ETSI
#ifdef IF2
   return Encoder3GPP( used_mode, prm, serial, txFrameType, mode );

#else
   return EncoderMMS( used_mode, prm, serial, txFrameType, mode );

#endif
#else

   Prm2Bits( used_mode, prm, &serial[1] );
   serial[0] = ( Word16 )txFrameType;
   serial[245] = ( Word16 )mode;
   return 500;
#endif
```

Changes in file *decoder.c*

Lines 17 – 22

```
#ifndef ETSI
#ifndef IF2
#include <string.h>
#define AMR_MAGIC_NUMBER "#!AMR\n"
#endif
#endif
```

Lines 70 – 81

```
#ifndef ETSI
   unsigned char analysis[32];
   enum Mode dec_mode;
#ifdef IF2
   short block_size[16]={ 12, 13, 15, 17, 18, 20, 25, 30, 5, 0, 0, 0, 0, 0, 0, 0
};
#else
   char magic[8];
   short block_size[16]={ 12, 13, 15, 17, 19, 20, 26, 31, 5, 0, 0, 0, 0, 0, 0, 0
};
#endif
#else
   short analysis[250];
#endif
```

Lines 108 – 140

```
#ifndef ETSI
#ifndef IF2
   /* read and verify magic number */
   fread( magic, sizeof( char ), strlen( AMR_MAGIC_NUMBER ), file_analysis );
   if ( strncmp( magic, AMR_MAGIC_NUMBER, strlen( AMR_MAGIC_NUMBER ) ) ) {
      fprintf( stderr, "%s%s\n", "Invalid magic number: ", magic );
      fclose( file_speech );
      fclose( file_analysis );
      return 1;
   }
#endif
```

```
#endif

#ifndef ETSI

   /* find mode, read file */
   while (fread(analysis, sizeof (unsigned char), 1, file_analysis ) > 0)
   {
#ifdef IF2
      dec_mode = analysis[0] & 0x000F;
#else
      dec_mode = (analysis[0] >> 3) & 0x000F;
#endif
      switch (dec_mode){
      case 0:
         read_size = 12;
         break;
      case 1:
         read_size = 13;
         break;
      case 2:
         read_size = 15;
         break;
      case 3:
         read_size = 17;
         break;
      case 4:
         read_size = 18;
         break;
      case 5:
         read_size = 20;
         break;
      case 6:
         read_size = 25;
         break;
      case 7:
         read_size = 30;
         break;
      case 8:
         read_size = 5;
         break;
      case 15:
         read_size = 0;
      default:
         read_size = 0;
         break;
      };
     read_size = block_size[dec_mode];

      fread(&analysis[1], sizeof (char), read_size, file_analysis );
#else

   read_size = 250;
   /* read file */
   while (fread(analysis, sizeof (short), read_size, file_analysis ) > 0)
   {
#endif
```

Changes in file *interf_dec.c*

Lines 175 – 548

```
#else

#ifndef IF2

/*
 * DecoderMMS
 *
 *
```

```
 * Parameters:
 *    param              O: AMR parameters
 *    stream             I: input bitstream
 *    frame_type         O: frame type
 *    speech_mode        O: speech mode in DTX
 *
 * Function:
 *    AMR file storage format frame to decoder parameters
 *
 * Returns:
 *    mode               used mode
 */
enum Mode DecoderMMS( Word16 *param, UWord8 *stream, enum RXFrameType
                      *frame_type, enum Mode *speech_mode, Word16 *q_bit )
{
   enum Mode mode;
   Word32 j;
   Word16 *mask;


   memset( param, 0, PRMNO_MR122 <<1 );
   *q_bit = 0x01 & (*stream >> 2);
   mode = 0x0F & (*stream >> 3);
   stream++;

   if ( mode == MRDTX ) {
      mask = order_MRDTX;

      for ( j = 1; j < 36; j++ ) {
         if ( *stream & 0x80 )
            param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
         mask += 2;

         if ( j % 8 )
            *stream <<= 1;
         else
            stream++;
      }

      /* get SID type bit */

      *frame_type = RX_SID_FIRST;
      if (*stream & 0x80)
         *frame_type = RX_SID_UPDATE;

      /* since there is update, use it */
      /* *frame_type = RX_SID_UPDATE; */

      /* speech mode indicator */
    *speech_mode = (*stream >> 4) && 0x07;

   }
   else if ( mode == 15 ) {
      *frame_type = RX_NO_DATA;
   }
   else if ( mode == MR475 ) {
      mask = order_MR475;

      for ( j = 1; j < 96; j++ ) {
         if ( *stream & 0x80 )
            param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
         mask += 2;

         if ( j % 8 )
            *stream <<= 1;
         else
            stream++;
      }
      *frame_type = RX_SPEECH_GOOD;
   }
   else if ( mode == MR515 ) {
```

```
        mask = order_MR515;

        for ( j = 1; j < 104; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else if ( mode == MR59 ) {
        mask = order_MR59;

        for ( j = 1; j < 119; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else if ( mode == MR67 ) {
        mask = order_MR67;

        for ( j = 1; j < 135; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else if ( mode == MR74 ) {
        mask = order_MR74;

        for ( j = 1; j < 149; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else if ( mode == MR795 ) {
        mask = order_MR795;

        for ( j = 1; j < 160; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
```

```
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else if ( mode == MR102 ) {
        mask = order_MR102;

        for ( j = 1; j < 205; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else if ( mode == MR122 ) {
        mask = order_MR122;

        for ( j = 1; j < 245; j++ ) {
            if ( *stream & 0x80 )
                param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
            mask += 2;

            if ( j % 8 )
                *stream <<= 1;
            else
                stream++;
        }
        *frame_type = RX_SPEECH_GOOD;
    }
    else
        *frame_type = RX_SPEECH_BAD;
    return mode;
}

#else

/*
 * Decoder3GPP
 *
 *
 * Parameters:
 *    param             O: AMR parameters
 *    stream            I: input bitstream
 *    frame_type        O: frame type
 *    speech_mode       O: speech mode in DTX
 *
 * Function:
 *    Resets state memory
 *
 * Returns:
 *    mode              used mode
 */
enum Mode Decoder3GPP( Word16 *param, UWord8 *stream, enum RXFrameType
                       *frame_type, enum Mode *speech_mode )
{
    enum Mode mode;
    Word32 j;
    Word16 *mask;


    memset( param, 0, PRMNO_MR122 <<1 );
    mode = 0xF & *stream;
    *stream >>= 4;

    if ( mode == MRDTX ) {
        mask = order_MRDTX;
```

```
   for ( j = 5; j < 40; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }

   /* get SID type bit */

   *frame_type = RX_SID_FIRST;
   if (*stream)
      *frame_type = RX_SID_UPDATE;

   /* since there is update, use it */
   /* *frame_type = RX_SID_UPDATE; */
   stream++;

   /* speech mode indicator */
   *speech_mode = *stream;
}
else if ( mode == 15 ) {
   *frame_type = RX_NO_DATA;
}
else if ( mode == MR475 ) {
   mask = order_MR475;

   for ( j = 5; j < 100; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
}
else if ( mode == MR515 ) {
   mask = order_MR515;

   for ( j = 5; j < 108; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
}
else if ( mode == MR59 ) {
   mask = order_MR59;

   for ( j = 5; j < 123; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
```

```
}
else if ( mode == MR67 ) {
   mask = order_MR67;

   for ( j = 5; j < 139; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
}
else if ( mode == MR74 ) {
   mask = order_MR74;

   for ( j = 5; j < 153; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
}
else if ( mode == MR795 ) {
   mask = order_MR795;

   for ( j = 5; j < 164; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
}
else if ( mode == MR102 ) {
   mask = order_MR102;

   for ( j = 5; j < 209; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
      else
         stream++;
   }
   *frame_type = RX_SPEECH_GOOD;
}
else if ( mode == MR122 ) {
   mask = order_MR122;

   for ( j = 5; j < 249; j++ ) {
      if ( *stream & 0x1 )
         param[ * mask] = ( short )( param[ * mask] + *( mask + 1 ) );
      mask += 2;

      if ( j % 8 )
         *stream >>= 1;
```

```
            else
               stream++;
         }
         *frame_type = RX_SPEECH_GOOD;
      }
      else
         *frame_type = RX_SPEECH_BAD;
      return mode;
   }
#endif
#endif
```

Lines 676 – 751:

```
#ifndef ETSI
#ifndef IF2
   Word16 q_bit;
#endif
#endif


   s = ( dec_interface_State * )st;

#ifndef ETSI

   /*
    * extract mode information and frametype,
    * octets to parameters
    */
#ifdef IF2
   mode = Decoder3GPP( prm, bits, &frame_type, &speech_mode );
#else
   mode = DecoderMMS( prm, bits, &frame_type, &speech_mode, &q_bit );
   if (!bfi)  bfi = 1 - q_bit;
#endif

   /*
    * if no mode information
    * guess one from the previous frame
    */
   if ( frame_type == RX_SPEECH_BAD ) {
      if ( s->prev_ft > 3 ) {
         frame_type = RX_SID_BAD;
         mode = MRDTX;
      }
      else {
         mode = s->prev_mode;
      }
   }
   else if ( frame_type == RX_NO_DATA ) {
      mode = s->prev_mode;
   }

   if ( bfi == 1 ) {
      if ( mode < 8 ) {
         frame_type = RX_SPEECH_BAD;
      }
      else if ( mode != 15 ) {
         frame_type = RX_SID_BAD;
      }
   }

#else
   bfi = 0;
   frame_type = bits[0];

   switch ( frame_type ) {
      case 0:
         frame_type = RX_SPEECH_GOOD;
         mode = bits[245];
         Bits2Prm( mode, &bits[1], prm );
```

```
            break;

         case 1:
            frame_type = RX_SID_FIRST;
            mode = s->prev_mode;
            break;

         case 2:
            frame_type = RX_SID_UPDATE;
            mode = s->prev_mode;
            Bits2Prm( MRDTX, &bits[1], prm );
            break;

         case 3:
            frame_type = RX_NO_DATA;
            mode = s->prev_mode;
            break;
      }

   #endif
```

Changes in file *interf_rom.h*

Lines 51 – 63

```
#ifndef IF2
#ifndef ETSI
static const UWord8 block_size[16]={ 13, 14, 16, 18, 20, 21, 27, 32,
                                     6 , 0 , 0 , 0 , 0 , 0 , 0 , 1  };

static const UWord8 toc_byte[16]={0x04, 0x0C, 0x14, 0x1C, 0x24, 0x2C, 0x34,
0x3C,
                    0x44, 0x4C, 0x54, 0x5C, 0x64, 0x6C, 0x74, 0x7C};
#endif
#else
/* One encoded frame (bytes) */
static const UWord8 block_size[16]={ 13, 14, 16, 18, 19, 21, 26, 31,
                                     5 , 0 , 0 , 0 , 0 , 0 , 0 , 1  };
#endif
```

3GPP TSG-SA4 Meeting #25
San Francisco, USA, 20-24 January 2003

Tdoc ⌘S4-030037

CR-Form-v7

CHANGE REQUEST

⌘ **26.104** **CR** **022** ⌘ **rev** **-** ⌘ Current version: **3.4.0** ⌘

*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:	⌘	Correction to floating-point implementation of sp_dec.c
Source:	⌘	TSG SA WG4
Work item code:	⌘	AMR
Date:	⌘	18/03/2003
Category:	⌘	**F**
Release:	⌘	R99

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:	⌘	The synthesized speech frames will be empty if NO13BIT is defined.
Summary of change:	⌘	The code is corrected to handle NO13BIT.
Consequences if not approved:	⌘	Improper operation of the codec. Zeros are returned instead of decoded speech.

Clauses affected:	⌘	C code file sp_dec.c

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

Other comments: ⌘

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1. How the code is changed

1.1 File sp_dec.c

1.1.1 Before the change (lines 5675..5678)

```
#ifndef NO13BIT

   Word32 i;
#endif
```

1.1.2 After the change

```
   Word32 i;
```

1.1.3 Before the change (lines 5690..5696)

```
#ifndef NO13BIT

   /* Truncate to 13 bits */
   for ( i = 0; i < L_FRAME; i++ ) {
      synth[i] = ( Word16 )( synth_speech[i] & 0xfff8 );
   }
#endif
```

1.1.4 After the change

```
   for ( i = 0; i < L_FRAME; i++ ) {
#ifndef NO13BIT
      /* Truncate to 13 bits */
      synth[i] = ( Word16 )( synth_speech[i] & 0xfff8 );
#else
      synth[i] = ( Word16 )( synth_speech[i]);
#endif
   }
```

3GPP TSG-SA4 Meeting #25
San Francisco, USA, 20-24 January 2003

Tdoc ⌘S4-030038

CR-Form-v7

CHANGE REQUEST

⌘ **26.104** **CR** **023** ⌘ **rev** **-** ⌘ Current version: **4.3.0** ⌘

*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:	⌘	Correction to floating-point implementation of sp_dec.c
Source:	⌘	TSG SA WG4
Work item code:	⌘	AMR
Date:	⌘	18/03/2003
Category:	⌘	**A**
Release:	⌘	Rel-4

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:	⌘	The synthesized speech frames will be empty if NO13BIT is defined.
Summary of change:	⌘	The code is corrected to handle NO13BIT.
Consequences if not approved:	⌘	Improper operation of the codec. Zeros are returned instead of decoded speech.

Clauses affected:	⌘	C code file sp_dec.c			
		Y	**N**		
Other specs affected:	⌘		X	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1. How the code is changed

1.1 File sp_dec.c

1.1.1 Before the change (lines 5675..5678)

```
#ifndef NO13BIT

   Word32 i;
#endif
```

1.1.2 After the change

```
   Word32 i;
```

1.1.3 Before the change (lines 5690..5696)

```
#ifndef NO13BIT

   /* Truncate to 13 bits */
   for ( i = 0; i < L_FRAME; i++ ) {
      synth[i] = ( Word16 )( synth_speech[i] & 0xfff8 );
   }
#endif
```

1.1.4 After the change

```
   for ( i = 0; i < L_FRAME; i++ ) {
#ifndef NO13BIT
      /* Truncate to 13 bits */
      synth[i] = ( Word16 )( synth_speech[i] & 0xfff8 );
#else
      synth[i] = ( Word16 )( synth_speech[i]);
#endif
   }
```

3GPP TSG-SA4 Meeting #25
San Francisco, USA, 20-24 January 2003

Tdoc ⌘S4-030039

CR-Form-v7

CHANGE REQUEST

⌘ **26.104** **CR** **024** ⌘**rev** **-** ⌘ Current version: **5.0.0** ⌘

*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:	⌘	Correction to floating-point implementation of sp_dec.c
Source:	⌘	TSG SA WG4
Work item code:	⌘	AMR
Date:	⌘	18/03/2003
Category:	⌘	**A**
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:	⌘	The synthesized speech frames will be empty if NO13BIT is defined.
Summary of change:	⌘	The code is corrected to handle NO13BIT.
Consequences if not approved:	⌘	Improper operation of the codec. Zeros are returned instead of decoded speech.

Clauses affected:	⌘	C code file sp_dec.c			
		Y	**N**		
Other specs affected:	⌘		X	Other core specifications	⌘
			X	Test specifications	
			X	O&M Specifications	
Other comments:	⌘				

How to create CRs using this form:
Comprehensive information and tips about how to create CRs can be found at http://www.3gpp.org/specs/CR.htm. Below is a brief summary:

1) Fill out the above form. The symbols above marked ⌘ contain pop-up help information about the field that they are closest to.

2) Obtain the latest version for the release of the specification to which the change is proposed. Use the MS Word "revision marks" feature (also known as "track changes") when making the changes. All 3GPP specifications can be downloaded from the 3GPP server under ftp://ftp.3gpp.org/specs/ For the latest version, look for the directory name with the latest date e.g. 2001-03 contains the specifications resulting from the March 2001 TSG meetings.

3) With "track changes" disabled, paste the entire CR form (use CTRL-A to select it) into the specification just in front of the clause containing the first piece of changed text. Delete those parts of the specification which are not relevant to the change request.

1. How the code is changed

1.1 File sp_dec.c

1.1.1 Before the change (lines 5675..5678)

```
#ifndef NO13BIT

   Word32 i;
#endif
```

1.1.2 After the change

```
   Word32 i;
```

1.1.3 Before the change (lines 5690..5696)

```
#ifndef NO13BIT

   /* Truncate to 13 bits */
   for ( i = 0; i < L_FRAME; i++ ) {
      synth[i] = ( Word16 )( synth_speech[i] & 0xfff8 );
   }
#endif
```

1.1.4 After the change

```
   for ( i = 0; i < L_FRAME; i++ ) {
#ifndef NO13BIT
      /* Truncate to 13 bits */
      synth[i] = ( Word16 )( synth_speech[i] & 0xfff8 );
#else
      synth[i] = ( Word16 )( synth_speech[i]);
#endif
   }
```