TSGS#20(03)0216

Source: **TSG-SA WG4**

Title: CR to TS 26.173 - MMS compatible input/output option for fixed-point AMR-WB source code (Release 5)

Document for: **Approval**

Agenda Item: **7.4.3**

The following CR, agreed at the TSG-SA WG4 meeting #26, is presented to TSG SA #20 for approval.

Spec	CR	Rev	Phase	Subject	Cat	Vers	WG	Meeting	S4 doc
26.173	017	1	Rel-5	MMS compatible input/output option for fixed-point AMR-WB source code	F	5.6.0	S4	TSG-SA WG4#26	S4-030400

3GPP TSG-SA4 Meeting #26
Paris, France, 5-9 May 2003

Tdoc ⌘S4-(03)0400

CR-Form-v7

CHANGE REQUEST

⌘ **26.173** **CR** **017** ⌘**rev** **1** ⌘ Current version: **5.6.0** ⌘

*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:	⌘	MMS compatible input/output option for fixed-point AMR-WB source code
Source:	⌘	TSG SA WG4
Work item code:	⌘	AMRWB
Date:	⌘	10/06/2003
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-WB fixed-point to encode and decode files according to the AMR-WB MIME file storage format, which is used e.g. by the MMS service.
Summary of change:	⌘	New input/output option.
Consequences if not approved:	⌘	Codec can not operate with bitstreams in AMR MIME file storage format.

Clauses affected:	⌘	2, 6.3; source files coder.c, decoder.c, bits.c, mime_io.tab

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

Other comments:	⌘	

Changes to the specification document:

2. 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 26.174: "AMR Wideband Speech Codec; Test sequences".

[2] 3GPP TS 26.190: "AMR Wideband Speech Codec; Speech transcoding".

[3] 3GPP TS 26.191: "AMR Wideband Speech Codec; Substitution and muting of lost frames".

[4] 3GPP TS 26.192: "AMR Wideband Speech Codec; Comfort noise aspects".

[5] 3GPP TS 26.193: "AMR Wideband Speech Codec; Source controlled rate operation".

[6] 3GPP TS 26.194: "AMR Wideband Speech Codec; Voice Activity Detection".

[7] 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 following available formats.

NOTE ON DEFAULT 3GPP AND ITU BITSTREAM FORMATS:
ITU stream format gives very limited possibilities to distinguish NO_DATA and SID_FIRST frame types at the beginning of a stream. In some very limited cases for which some instance between encoder and decoder cuts of the first hangover period frames (e.g. handovers, editing of the stream), the output of the decoder is different depending on the stream format, ITU or default 3GPP.

Default 3GPP format:

This is the default format used in 3GPP. This format shall be used when the codec is tested against the test vectors.

TYPE_OF_FRAME_TYPE	FRAME_TYPE	MODE	B1	B2	…	Bnn

Each box corresponds to one `Word16` value in the bitstream file, for a total of 3+nn words or 6+2nn bytes per frame, where nn is the number of encoded bits in the frame. Each encoded bit is represented as follows: Bit 0 = 0xff81, Bit 1 = 0x007f. The fields have the following meaning:

`TYPE_OF_FRAME_TYPE` transmit frame type, which is one of
`TX_TYPE` (`0x6b21`)
`RX_TYPE` (`0x6b20`)

```
If TYPE_OF_FRAME_TYPE is TX_TYPE,

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

If TYPE_OF_FRAME_TYPE is RX_TYPE,

FRAME_TYPE     transmit       frame       type,       which       is       one       of
                    RX_SPEECH_GOOD  (0x0000)
                    RX_SPEECH_PROBABLY_DEGRADED  (0x0001)
                    RX_SPEECH_LOST  (0x0002)
                    RX_SPEECH_BAD   (0x0003)
                    RX_SID_FIRST    (0x0004)
                    RX_SID_UPDATE   (0x0005)
                    RX_SID_BAD      (0x0006)
                    RX_NO_DATA      (0x0007)

B0…B2nn        speech encoder parameter bits (i.e. the bitstream itself). Each Bx either has the
               value 0x0081 (for bit 0) or 0x007F (for bit 1).

MODE_INFO      encoding       mode       information,       which       is       one       of
                    6.60  kbit/s mode  (0x0000)
                    8.85  kbit/s mode  (0x0001)
                    12.65 kbit/s mode  (0x0002)
                    14.25 kbit/s mode  (0x0003)
                    15.85 kbit/s mode  (0x0004)
                    18.25 kbit/s mode  (0x0005)
                    19.85 kbit/s mode  (0x0006)
                    23.05 kbit/s mode  (0x0007)
                    23.85 kbit/s mode  (0x0008)
```

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

ITU format (activated with command line parameter -itu)

SYNC_WORD	DATA_LENGTH	B1	B2	…	Bnn

Each box corresponds to one `Word16` value in the bitstream file, for a total of 2+nn words or 4+2nn bytes per frame, where nn is the number of encoded bits in the frame. Each encoded bit is represented as follows: Bit 0 = 0x007f, Bit 1 = 0x0081. The fields have the following meaning:

`SYNC_WORD` Word to ensure correct frame synchronization between the encoder and the decoder. It is also used to indicate the occurrences of bad frames.

```
In the encoder output:   (0x6b21)
In the decoder input:    Good frames (0x6b21)
                         Bad frames  (0x6b20)
```

`DATA_LENGTH` Length of the speech data. Codec mode and frame type is extracted in the decoder using this parameter:

DATA _LENGTH	PREVIOUS FRAME	CODEC MODE	FRAMETYPE
0	RX_SPEECH_GOOD/	DTX	RX_SID_FIRST

	RX_SPEECH_BAD		
0	OTHER THAN RX_SPEECH_GOOD/ RX_SPEECH_BAD	DTX	RX_NO_DATA
35	-	DTX	RX_SID_UPDATE/ RX_SID_BAD
132	-	6.60 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
177	-	8.85 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
253	-	12.65 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
285	-	14.25 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
317	-	15.85 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
365	-	18.25 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
397	-	19.85 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
461	-	23.05 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD
477	-	23.85 kbit/s	RX_SPEECH_GOOD/ RX_SPEECH_BAD

MIME/file storage format (activated with command line parameter -mime)

Detailed description of the AMR-WB single channel MIME/file storage format can be found in [7] (sections 5.1 and 5.3). This format is used e.g. by the Multimedia Messaging Service (MMS).

Changes to the c source-code:

Changes in file *coder.c*

Lines 21 - 54:

```
/*-------------------------------------------------------------------*
 * CODER.C                                                           *
 * ~~~~~~~~~~                                                        *
 * Main program of the AMR WB ACELP wideband coder.                  *
 *                                                                   *
 *    Usage : coder (-dtx) (-itu | -mime) mode speech_file  bitstream_file *
 *                                                                   *
 *    Format for speech_file:                                        *
 *      Speech is read from a binary file of 16 bits data.           *
 *                                                                   *
 *    Format for bitstream_file (default):                           *
 *                                    *
 *        1 word (2-byte) for the type of frame type                 *
 *          (TX_FRAME_TYPE or RX_FRAME_TYPE)                         *
 *        1 word (2-byte) for the frame type                         *
 *          (see dtx.h for possible values)                          *
```

```
 *        1 word (2-byte) for the mode indication                    *
 *          (see bits.h for possible values)                         *
 *        N words (2-byte) containning N bits.                       *
 *          Bit 0 = 0xff81 and Bit 1 = 0x007f                        *
 *                                                                   *
 *    if option -itu defined:                                      *
 *        1 word (2-byte) for sync word (0x6b21)                     *
 *        1 word (2-byte) frame length N                             *
 *        N words (2-byte) containing N 'soft' bits                  *
 *             (bit 0 = 0x007f, bit 1 = 0x008f)                      *
 *                                                                   *
*    if option -mime defined:                                     *
 *        AMR-WB MIME/storage format, see RFC 3267 (sections 5.1 and 5.3) for
details *
 *                                                                   *
*    mode = 0..8 (bit rate = 6.60 to 23.85 k)                     *
 *                                                                   *
 *    -dtx if DTX is ON                                              *
 *-------------------------------------------------------------------*/
```

Lines 75 - 77:

```
  fprintf(stderr, "
================================================================================
==================\n");
  fprintf(stderr, " AMR Wideband Codec 3GPP TS26.190 / ITU-T G.722.2, April
30March 10, 2003. Version %s.\n", CODEC_VERSION);
  fprintf(stderr, "
================================================================================
==================\n");
```

Lines 84 – 115:

```
    if ((argc < 4) || (argc > 6))
    {
        fprintf(stderr, "Usage : coder  (-dtx) (-itu| -mime) mode speech_file
bitstream_file\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "Format for speech_file:\n");
        fprintf(stderr, "  Speech is read form a binary file of 16 bits
data.\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "Format for bitstream_file (default):\n");
        fprintf(stderr, "  One word (2-byte) to indicate type of frame
type.\n");
        fprintf(stderr, "  One word (2-byte) to indicate frame type.\n");
        fprintf(stderr, "  One word (2-byte) to indicate mode.\n");
        fprintf(stderr, "  N words (2-byte) containing N bits (bit 0 = 0xff81,
bit 1 = 0x007f).\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "  if option -itu defined:\n");
    fprintf(stderr, "  One word (2-byte) for sync word (0x6b21)\n");
    fprintf(stderr, "  One word (2-byte) for frame length N.\n");
    fprintf(stderr, "  N words (2-byte) containing N bits (bit 0 = 0x007f, bit
1 = 0x0081).\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "  if option -mime defined:\n");
    fprintf(stderr, "  AMR-WB MIME/storage format, see RFC 3267 (sections 5.1
and 5.3) for details.\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "mode: 0 to 8 (9 bits rates) or\n");
        fprintf(stderr, "      -modefile filename\n");
        fprintf(stderr, "
=====================================================================\n");
        fprintf(stderr, " mode   :  (0)  (1)   (2)   (3)   (4)   (5)   (6)   (7)
(8)     \n");
        fprintf(stderr, " bitrate: 6.60 8.85 12.65 14.25 15.85 18.25 19.85 23.05
23.85 kbit/s\n");
        fprintf(stderr, "
=====================================================================\n");
```

```
        fprintf(stderr, "\n");
        fprintf(stderr, "-dtx if DTX is ON, default is OFF\n");
        fprintf(stderr, "\n");
        exit(0);
    }
```

Lines 118 – 127:

```
  bitstreamformat = 0;
   if (strcmp(argv[1], "-itu") == 0)
  {
      bitstreamformat = 1;
      argv++;
    fprintf(stderr, "Input bitstream format: ITU\n");
   } else
  {
    if (strcmp(argv[1], "-mime") == 0)
    {
      bitstreamformat = 2;
      argv++;
      fprintf(stderr, "Input bitstream format: MIME\n");
    } else
    {
      fprintf(stderr, "Input bitstream format: Default\n");
    }
    }
```

Lines 195 – 203:

```
    fprintf(stderr, "\n --- Running ---\n");

    /* If MIME/storage format selected, write the magic number at the beginning
of the bitstream file */
  if (bitstreamformat == 2)
  {
    fwrite("#!AMR-WB\n", sizeof(char), 9, f_serial);
  }

    frame = 0;
```

Changes in file *decoder.c*

Lines 8 – 10:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Lines 21 – 51:

```
/*-------------------------------------------------------------------*
 * DECODER.C                                                         *
 * ~~~~~~~~~                                                         *
 * Main program of the AMR WB ACELP wideband decoder.                *
 *                                                                   *
 *    Usage : decoder (-itu| -mime) bitstream_file synth_file bit_rate      *
 *                                                                   *
 *    Format for bitstream_file (default):                           *
 *                                      *
 *        1 word (2-byte) for the type of frame type                 *
 *          (TX_FRAME_TYPE or RX_FRAME_TYPE)                         *
 *        1 word (2-byte) for the frame type                         *
 *          (see dtx.h for possible values)                          *
 *        1 word (2-byte) for the mode indication                    *
 *          (see bits.h for possible values)                         *
 *        N words (2-byte) containning N bits.                       *
 *          Bit 0 = 0xff81 and Bit 1 = 0x007f                        *
 *                                      *
```

```
 *    if option -itu defined:                                          *
 *        1 word (2-byte) for sync word (0x6b21)                          *
 *        1 word (2-byte) frame length N                                  *
 *        N words (2-byte) containing N 'soft' bits                       *
 *              (bit 0 = 0x007f, bit 1 = 0x0081)                          *
 *                                                                        *
 *    if option -mime defined:                                         *
 *        AMR-WB MIME/storage format, see RFC 3267 (sections 5.1 and 5.3) for details *
 *                                                                       *
 *    Format for synth_file:                                           *
 *      Synthesis is written to a binary file of 16 bits data.         *
 *                                                                     *
 *---------------------------------------------------------------------*/
```

Lines 68 – 72:

```
  Word16 bitstreamformat;
  RX_State *rx_state;

  char magic[10];
    void *st;
```

Lines 75 – 77:

```
      fprintf(stderr, "
================================================================================
==================\n");
      fprintf(stderr, " AMR Wideband Codec 3GPP TS26.190 / ITU-T G.722.2, April 30March 10, 2003. Version %s.\n", CODEC_VERSION);
      fprintf(stderr, "
================================================================================
==================\n");
```

Lines 84 – 106:

```
    if (argc != 3 && argc != 4)
    {
        fprintf(stderr, "Usage : decoder  (-itu | -mime) bitstream_file
synth_file\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "Format for bitstream_file: (default)\n");
        fprintf(stderr, "  One word (2-byte) to indicate type of frame
type.\n");
        fprintf(stderr, "  One word (2-byte) to indicate frame type.\n");
        fprintf(stderr, "  One word (2-byte) to indicate mode.\n");
        fprintf(stderr, "  N words (2-byte) containing N bits (bit 0 = 0xff81,
bit 1 = 0x007f).\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "  if option -itu defined:\n");
     fprintf(stderr, "  One word (2-byte) for sync word (good frames: 0x6b21,
bad frames: 0x6b20)\n");
     fprintf(stderr, "  One word (2-byte) for frame length N.\n");
     fprintf(stderr, "  N words (2-byte) containing N bits (bit 0 = 0x007f, bit
1 = 0x0081).\n");
     fprintf(stderr, "\n");
        fprintf(stderr, "  if option -mime defined:\n");
     fprintf(stderr, "  AMR-WB MIME/storage format, see RFC 3267 (sections 5.1
and 5.3) for details.\n");
        fprintf(stderr, "\n");
        fprintf(stderr, "Format for synth_file:\n");
        fprintf(stderr, "  Synthesis is written to a binary file of 16 bits
data.\n");
        fprintf(stderr, "\n");
        exit(0);
    }
```

Lines 109 – 126:

```
  bitstreamformat = 0;
```

```
    if (strcmp(argv[1], "-itu") == 0)
  {
      bitstreamformat = 1;
      argv++;
    fprintf(stderr, "Input bitstream format: ITU\n");
    } else
  {
    if (strcmp(argv[1], "-mime") == 0)
    {
      bitstreamformat = 2;
      argv++;
      fprintf(stderr, "Input bitstream format: MIME\n");
    } else
    {
      fprintf(stderr, "Input bitstream format: Default\n");
    }
  }
```

Lines 155 – 171:

```
    fprintf(stderr, "\n --- Running ---\n");

  /* read and verify magic number if MIME/storage format specified */
  if (bitstreamformat == 2)
  {
    fread(magic, sizeof(char), 9, f_serial);

    if (strncmp(magic, "#!AMR-WB\n", 9))
    {
      fprintf(stderr, "%s%s\n", "Invalid magic number: ", magic);
      fclose(f_serial);
      fclose(f_synth);
      exit(0);
    }
  }

    frame = 0;
```

Changes in file *bits.c*

Lines 7 – 18:

```
#include <stdlib.h>
#include <stdio.h>
#include "typedef.h"
#include "basic_op.h"
#include "cnst.h"
#include "bits.h"
#include "acelp.h"
#include "count.h"
#include "dtx.h"

#include "mime_io.tab"
```

Lines 62 – 65:

```
   Word16 i, frame_type;
   Word16 stream[SIZE_MAX];
   UWord8 temp;
   UWord8 *stream_ptr;
```

Lines 103 – 217:

```
   if(bitstreamformat == 0)         /* default file format */
   {
     stream[0] = TX_FRAME_TYPE;
     stream[1] = frame_type;
     stream[2] = mode;
```

```
      for (i = 0; i < nb_of_bits[coding_mode]; i++)
      {
        stream[3 + i] = prms[i];
      }

      fwrite(stream, sizeof(Word16), 3 + nb_of_bits[coding_mode], fp);
    } else                  /* ITU file format */
    {
      if (bitstreamformat == 1)   /* ITU file format */
      {
        stream[0] = 0x6b21;

        if(frame_type != TX_NO_DATA && frame_type != TX_SID_FIRST)
        {
          stream[1]=nb_of_bits[coding_mode];
          for (i = 0; i < nb_of_bits[coding_mode]; i++)
          {
            if(prms[i] == BIT_0){
              stream[2 + i] = BIT_0_ITU;
            }
            else{
              stream[2 + i] = BIT_1_ITU;
            }
          }
          fwrite(stream, sizeof(Word16), 2 + nb_of_bits[coding_mode], fp);
        } else
        {
          stream[1] = 0;
          fwrite(stream, sizeof(Word16), 2, fp);
        }
      } else                /* MIME/storage file format */
      {
#define MRSID 9
        /* change mode index in case of SID frame */
        if (coding_mode == MRDTX)
        {
          coding_mode = MRSID;

          if (frame_type == TX_SID_FIRST)
          {
            for (i = 0; i < NBBITS_SID; i++) prms[i] = BIT_0;
          }
        }

        /* we cannot handle unspecified frame types (modes 10 - 13) */
        /* -> force NO_DATA frame */
        if (coding_mode < 0 || coding_mode > 15 || (coding_mode > MRSID &&
coding_mode < 14))
        {
          coding_mode = 15;
        }

        /* mark empty frames between SID updates as NO_DATA frames */
        if (coding_mode == MRSID && frame_type == TX_NO_DATA)
        {
          coding_mode = 15;
        }

        /* set pointer for packed frame, note that we handle data as bytes */
          stream_ptr = (UWord8*)stream;

        /* insert table of contents (ToC) byte at the beginning of the packet */
        *stream_ptr = toc_byte[coding_mode];
        stream_ptr++;

        temp = 0;

        /* sort and pack AMR-WB speech or SID bits */
        for (i = 1; i < unpacked_size[coding_mode] + 1; i++)
        {
          if (prms[sort_ptr[coding_mode][i-1]] == BIT_1)
```

```
            {
               temp++;
            }

            if (i % 8)
            {
               temp <<= 1;
            }
            else
            {
               *stream_ptr = temp;
               stream_ptr++;
               temp = 0;
            }
         }

         /* insert SID type indication and speech mode in case of SID frame */
         if (coding_mode == MRSID)
         {
            if (frame_type == TX_SID_UPDATE)
            {
               temp++;
            }
            temp <<= 4;

            temp += mode & 0x000F;
         }

         /* insert unused bits (zeros) at the tail of the last byte */
         if (unused_size[coding_mode])
         {
            temp <<= (unused_size[coding_mode] - 1);
         }
         *stream_ptr = temp;

         /* write packed frame into file (1 byte added to cover ToC entry) */
         fwrite(stream, sizeof(UWord8), 1 + packed_size[coding_mode], fp);
      }
   }
```

Lines 263 – 474:

```
Word16 Read_serial(FILE * fp, Word16 prms[], Word16 * frame_type, Word16 * mode,
RX_State *st, Word16 bitstreamformat)
{
   Word16 n, n1, type_of_frame_type, coding_mode, datalen, i;
   UWord8 toc, q, temp, *packet_ptr, packet[64];

   if(bitstreamformat == 0)        /* default file format */
   {
      n = (Word16) fread(&type_of_frame_type, sizeof(Word16), 1, fp);
      n = (Word16) (n + fread(frame_type, sizeof(Word16), 1, fp));
      n = (Word16) (n + fread(mode, sizeof(Word16), 1, fp));
      coding_mode = *mode;
      if(*mode < 0 || *mode > NUM_OF_MODES-1)
      {
         fprintf(stderr, "Invalid mode received: %d (check file format).\n",
*mode);
         exit(-1);
      }
      if (n == 3)
      {
         if (type_of_frame_type == TX_FRAME_TYPE)
         {
            switch (*frame_type)
            {
            case TX_SPEECH:
               *frame_type = RX_SPEECH_GOOD;
               break;
            case TX_SID_FIRST:
               *frame_type = RX_SID_FIRST;
```

```
            break;
          case TX_SID_UPDATE:
            *frame_type = RX_SID_UPDATE;
            break;
          case TX_NO_DATA:
            *frame_type = RX_NO_DATA;
            break;
          }
        } else if (type_of_frame_type != RX_FRAME_TYPE)
        {
          fprintf(stderr, "Wrong type of frame type:%d.\n", type_of_frame_type);
        }

        if ((*frame_type == RX_SID_FIRST) | (*frame_type == RX_SID_UPDATE) |
(*frame_type == RX_NO_DATA) | (*frame_type == RX_SID_BAD))
        {
          coding_mode = MRDTX;
        }
        n = (Word16) fread(prms, sizeof(Word16), nb_of_bits[coding_mode], fp);
        if (n != nb_of_bits[coding_mode])
          n = 0;
      }
      return (n);
    } else                  /* ITU file format */
    {
      if (bitstreamformat == 1)   /* ITU file format */
      {
        n = (Word16) fread(&type_of_frame_type, sizeof(Word16), 1, fp);
        n = (Word16)(n+fread(&datalen, sizeof(Word16), 1, fp));

        if(n == 2)
        {
          if(datalen == 0)                /* RX_NO_DATA frame type */
          {
            if(st->prev_ft == RX_SPEECH_GOOD || st->prev_ft == RX_SPEECH_BAD)
            {
              *frame_type = RX_SID_FIRST;
            } else
            {
              *frame_type = RX_NO_DATA;
            }
            *mode = st->prev_mode;
          }
          else{

            coding_mode = -1;
            for(i=NUM_OF_MODES-1; i>=0; i--)
            {
              if(datalen == nb_of_bits[i])
              {
                coding_mode = i;
              }
            }
            if(coding_mode == -1)
            {
              fprintf(stderr, "\n\n ERROR: Invalid number of data bits received
[%d]\n\n", datalen);
              exit(-1);
            }

            if(coding_mode == NUM_OF_MODES-1)  /* DTX frame type */
            {
              if(type_of_frame_type == 0x6b20) /* bad SID frame */
              {
                *frame_type = RX_SID_BAD;
              } else                           /* correct SID frame */
              {
                *frame_type = RX_SID_UPDATE;
              }
              *mode = st->prev_mode;
            } else
```

```
            {
              if(type_of_frame_type == 0x6b20)
              {
                *frame_type = RX_SPEECH_BAD;
              } else
              {
                *frame_type = RX_SPEECH_GOOD;
              }
              *mode = coding_mode;
            }
          }
        }
        n1 = fread(prms, sizeof(Word16), datalen, fp);
        n += n1;
        for(i=0; i<n1; i++){
          if(prms[i] <= BIT_0_ITU) prms[i] = BIT_0;
          else            prms[i] = BIT_1;
        }
        st->prev_mode = *mode;
        st->prev_ft = *frame_type;
        return(n);

    } else               /* MIME/storage file format */
    {
        /* read ToC byte, return immediately if no more data available */
        if (fread(&toc, sizeof(UWord8), 1, fp) == 0)
        {
            return 0;
        }

        /* extract q and mode from ToC */
        q  = (toc >> 2) & 0x01;
        *mode = (toc >> 3) & 0x0F;

        /* read speech bits, return with empty frame if mismatch between mode
info and available data */
        if ((Word16)fread(packet, sizeof(UWord8), packed_size[*mode], fp) !=
packed_size[*mode])
        {
            return 0;
        }

        packet_ptr = (UWord8*)packet;
        temp = *packet_ptr;
        packet_ptr++;

        /* unpack and unsort speech or SID bits */
        for (i = 1; i < unpacked_size[*mode] + 1; i++)
        {
            if (temp & 0x80)  prms[sort_ptr[*mode][i-1]] = BIT_1;
            else       prms[sort_ptr[*mode][i-1]] = BIT_0;

            if (i % 8)
            {
                temp <<= 1;
            }
            else
            {
                temp = *packet_ptr;
                packet_ptr++;
            }
        }

        /* set frame type */
        switch (*mode)
        {
        case MODE_7k:
        case MODE_9k:
        case MODE_12k:
        case MODE_14k:
        case MODE_16k:
```

```
        case MODE_18k:
        case MODE_20k:
        case MODE_23k:
        case MODE_24k:
          if (q) *frame_type = RX_SPEECH_GOOD;
          else   *frame_type = RX_SPEECH_BAD;
          break;
        case MRSID:
          if (q)
          {
            if (temp & 0x80)  *frame_type = RX_SID_UPDATE;
            else              *frame_type = RX_SID_FIRST;
          }
          else
          {
            *frame_type = RX_SID_BAD;
          }

          /* read speech mode indication */
          coding_mode = (temp >> 3) & 0x0F;

          /* set mode index */
          *mode = st->prev_mode;
          break;
        case 14:   /* SPEECH_LOST */
          *frame_type = RX_SPEECH_LOST;
          *mode = st->prev_mode;
          break;
        case 15:   /* NO_DATA */
          *frame_type = RX_NO_DATA;
          *mode = st->prev_mode;
          break;
        default:   /* replace frame with unused mode index by NO_DATA frame */
          *frame_type = RX_NO_DATA;
          *mode = st->prev_mode;
          break;
        }

        st->prev_mode = *mode;

        /* return 1 to indicate succesfully parsed frame */
        return 1;
    }
#undef MRSID
   }

}
```

New file *mime_io.tab*

```
#include <stdio.h>
#include "typedef.h"

static UWord8 toc_byte[16] = {0x04, 0x0C, 0x14, 0x1C, 0x24, 0x2C, 0x34, 0x3C,
                              0x44, 0x4C, 0x54, 0x5C, 0x64, 0x6C, 0x74, 0x7C};

/* number of speech bits for all modes */
static Word16 unpacked_size[16] = {132,  177, 253, 285, 317, 365, 397, 461,
                                   477,   35,   0,   0,   0,   0,   0,   0};

/* size of packed frame for each mode, excluding TOC byte */
static Word16 packed_size[16] = {17, 23, 32, 36, 40, 46, 50, 58,
                                 60,  5,  0,  0,  0,  0,  0,  0};

/* number of unused speech bits in packed format for each mode */
static Word16 unused_size[16] = {4, 7, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0,
0};

/* sorting tables for all modes */
```

```
static Word16 sort_660[132] = {
     0,   5,   6,   7,  61,  84, 107, 130,  62,  85,
     8,   4,  37,  38,  39,  40,  58,  81, 104, 127,
    60,  83, 106, 129, 108, 131, 128,  41,  42,  80,
   126,   1,   3,  57, 103,  82, 105,  59,   2,  63,
   109, 110,  86,  19,  22,  23,  64,  87,  18,  20,
    21,  17,  13,  88,  43,  89,  65, 111,  14,  24,
    25,  26,  27,  28,  15,  16,  44,  90,  66, 112,
     9,  11,  10,  12,  67, 113,  29,  30,  31,  32,
    34,  33,  35,  36,  45,  51,  68,  74,  91,  97,
   114, 120,  46,  69,  92, 115,  52,  75,  98, 121,
    47,  70,  93, 116,  53,  76,  99, 122,  48,  71,
    94, 117,  54,  77, 100, 123,  49,  72,  95, 118,
    55,  78, 101, 124,  50,  73,  96, 119,  56,  79,
   102, 125
};

static Word16 sort_885[177] = {
     0,   4,   6,   7,   5,   3,  47,  48,  49, 112,
   113, 114,  75, 106, 140, 171,  80, 111, 145, 176,
    77, 108, 142, 173,  78, 109, 143, 174,  79, 110,
   144, 175,  76, 107, 141, 172,  50, 115,  51,   2,
     1,  81, 116, 146,  19,  21,  12,  17,  18,  20,
    16,  25,  13,  10,  14,  24,  23,  22,  26,   8,
    15,  52, 117,  31,  82, 147,   9,  33,  11,  83,
   148,  53, 118,  28,  27,  84, 149,  34,  35,  29,
    46,  32,  30,  54, 119,  37,  36,  39,  38,  40,
    85, 150,  41,  42,  43,  44,  45,  55,  60,  65,
    70,  86,  91,  96, 101, 120, 125, 130, 135, 151,
   156, 161, 166,  56,  87, 121, 152,  61,  92, 126,
   157,  66,  97, 131, 162,  71, 102, 136, 167,  57,
    88, 122, 153,  62,  93, 127, 158,  67,  98, 132,
   163,  72, 103, 137, 168,  58,  89, 123, 154,  63,
    94, 128, 159,  68,  99, 133, 164,  73, 104, 138,
   169,  59,  90, 124, 155,  64,  95, 129, 160,  69,
   100, 134, 165,  74, 105, 139, 170
};

static Word16 sort_1265[253] = {
     0,   4,   6,  93, 143, 196, 246,   7,   5,   3,
    47,  48,  49,  50,  51, 150, 151, 152, 153, 154,
    94, 144, 197, 247,  99, 149, 202, 252,  96, 146,
   199, 249,  97, 147, 200, 250, 100, 203,  98, 148,
   201, 251,  95, 145, 198, 248,  52,   2,   1, 101,
   204, 155,  19,  21,  12,  17,  18,  20,  16,  25,
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   156,  31, 102, 205,   9,  33,  11, 103, 206,  54,
   157,  28,  27, 104, 207,  34,  35,  29,  46,  32,
    30,  55, 158,  37,  36,  39,  38,  40, 105, 208,
    41,  42,  43,  44,  45,  56, 106, 159, 209,  57,
    66,  75,  84, 107, 116, 125, 134, 160, 169, 178,
   187, 210, 219, 228, 237,  58, 108, 161, 211,  62,
   112, 165, 215,  67, 117, 170, 220,  71, 121, 174,
   224,  76, 126, 179, 229,  80, 130, 183, 233,  85,
   135, 188, 238,  89, 139, 192, 242,  59, 109, 162,
   212,  63, 113, 166, 216,  68, 118, 171, 221,  72,
   122, 175, 225,  77, 127, 180, 230,  81, 131, 184,
   234,  86, 136, 189, 239,  90, 140, 193, 243,  60,
   110, 163, 213,  64, 114, 167, 217,  69, 119, 172,
   222,  73, 123, 176, 226,  78, 128, 181, 231,  82,
   132, 185, 235,  87, 137, 190, 240,  91, 141, 194,
   244,  61, 111, 164, 214,  65, 115, 168, 218,  70,
   120, 173, 223,  74, 124, 177, 227,  79, 129, 182,
   232,  83, 133, 186, 236,  88, 138, 191, 241,  92,
   142, 195, 245
};

static Word16 sort_1425[285] = {
     0,   4,   6, 101, 159, 220, 278,   7,   5,   3,
    47,  48,  49,  50,  51, 166, 167, 168, 169, 170,
```

```
   102, 160, 221, 279, 107, 165, 226, 284, 104, 162,
   223, 281, 105, 163, 224, 282, 108, 227, 106, 164,
   225, 283, 103, 161, 222, 280,  52,   2,   1, 109,
   228, 171,  19,  21,  12,  17,  18,  20,  16,  25,
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   172,  31, 110, 229,   9,  33,  11, 111, 230,  54,
   173,  28,  27, 112, 231,  34,  35,  29,  46,  32,
    30,  55, 174,  37,  36,  39,  38,  40, 113, 232,
    41,  42,  43,  44,  45,  56, 114, 175, 233,  62,
   120, 181, 239,  75, 133, 194, 252,  57, 115, 176,
   234,  63, 121, 182, 240,  70, 128, 189, 247,  76,
   134, 195, 253,  83, 141, 202, 260,  92, 150, 211,
   269,  84, 142, 203, 261,  93, 151, 212, 270,  85,
   143, 204, 262,  94, 152, 213, 271,  86, 144, 205,
   263,  95, 153, 214, 272,  64, 122, 183, 241,  77,
   135, 196, 254,  65, 123, 184, 242,  78, 136, 197,
   255,  87, 145, 206, 264,  96, 154, 215, 273,  58,
   116, 177, 235,  66, 124, 185, 243,  71, 129, 190,
   248,  79, 137, 198, 256,  88, 146, 207, 265,  97,
   155, 216, 274,  59, 117, 178, 236,  67, 125, 186,
   244,  72, 130, 191, 249,  80, 138, 199, 257,  89,
   147, 208, 266,  98, 156, 217, 275,  60, 118, 179,
   237,  68, 126, 187, 245,  73, 131, 192, 250,  81,
   139, 200, 258,  90, 148, 209, 267,  99, 157, 218,
   276,  61, 119, 180, 238,  69, 127, 188, 246,  74,
   132, 193, 251,  82, 140, 201, 259,  91, 149, 210,
   268, 100, 158, 219, 277
};

static Word16 sort_1585[317] = {
     0,   4,   6, 109, 175, 244, 310,   7,   5,   3,
    47,  48,  49,  50,  51, 182, 183, 184, 185, 186,
   110, 176, 245, 311, 115, 181, 250, 316, 112, 178,
   247, 313, 113, 179, 248, 314, 116, 251, 114, 180,
   249, 315, 111, 177, 246, 312,  52,   2,   1, 117,
   252, 187,  19,  21,  12,  17,  18,  20,  16,  25,
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   188,  31, 118, 253,   9,  33,  11, 119, 254,  54,
   189,  28,  27, 120, 255,  34,  35,  29,  46,  32,
    30,  55, 190,  37,  36,  39,  38,  40, 121, 256,
    41,  42,  43,  44,  45,  56, 122, 191, 257,  63,
   129, 198, 264,  76, 142, 211, 277,  89, 155, 224,
   290, 102, 168, 237, 303,  57, 123, 192, 258,  70,
   136, 205, 271,  83, 149, 218, 284,  96, 162, 231,
   297,  62, 128, 197, 263,  75, 141, 210, 276,  88,
   154, 223, 289, 101, 167, 236, 302,  58, 124, 193,
   259,  71, 137, 206, 272,  84, 150, 219, 285,  97,
   163, 232, 298,  59, 125, 194, 260,  64, 130, 199,
   265,  67, 133, 202, 268,  72, 138, 207, 273,  77,
   143, 212, 278,  80, 146, 215, 281,  85, 151, 220,
   286,  90, 156, 225, 291,  93, 159, 228, 294,  98,
   164, 233, 299, 103, 169, 238, 304, 106, 172, 241,
   307,  60, 126, 195, 261,  65, 131, 200, 266,  68,
   134, 203, 269,  73, 139, 208, 274,  78, 144, 213,
   279,  81, 147, 216, 282,  86, 152, 221, 287,  91,
   157, 226, 292,  94, 160, 229, 295,  99, 165, 234,
   300, 104, 170, 239, 305, 107, 173, 242, 308,  61,
   127, 196, 262,  66, 132, 201, 267,  69, 135, 204,
   270,  74, 140, 209, 275,  79, 145, 214, 280,  82,
   148, 217, 283,  87, 153, 222, 288,  92, 158, 227,
   293,  95, 161, 230, 296, 100, 166, 235, 301, 105,
   171, 240, 306, 108, 174, 243, 309
};

static Word16 sort_1825[365] = {
     0,   4,   6, 121, 199, 280, 358,   7,   5,   3,
    47,  48,  49,  50,  51, 206, 207, 208, 209, 210,
   122, 200, 281, 359, 127, 205, 286, 364, 124, 202,
   283, 361, 125, 203, 284, 362, 128, 287, 126, 204,
   285, 363, 123, 201, 282, 360,  52,   2,   1, 129,
   288, 211,  19,  21,  12,  17,  18,  20,  16,  25,
```

```
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   212,  31, 130, 289,   9,  33,  11, 131, 290,  54,
   213,  28,  27, 132, 291,  34,  35,  29,  46,  32,
    30,  55, 214,  37,  36,  39,  38,  40, 133, 292,
    41,  42,  43,  44,  45,  56, 134, 215, 293, 198,
   299, 136, 120, 138,  60, 279,  58,  62, 357, 139,
   140, 295, 156,  57, 219, 297,  63, 217, 137, 170,
   300, 222,  64, 106,  61,  78, 294,  92, 142, 141,
   135, 221, 296, 301, 343,  59, 298, 184, 329, 315,
   220, 216, 265, 251, 218, 237, 352, 223, 157,  86,
   171,  87, 164, 351, 111, 302,  65, 178, 115, 323,
    72, 192, 101, 179,  93,  73, 193, 151, 337, 309,
   143, 274,  69, 324, 165, 150,  97, 338, 110, 310,
   330, 273,  68, 107, 175, 245, 114,  79, 113, 189,
   246, 259, 174,  71, 185,  96, 344, 100, 322,  83,
   334, 316, 333, 252, 161, 348, 147,  82, 269, 232,
   260, 308, 353, 347, 163, 231, 306, 320, 188, 270,
   146, 177, 266, 350, 256,  85, 149, 116, 191, 160,
   238, 258, 336, 305, 255,  88, 224,  99, 339, 230,
   228, 227, 272, 242, 241, 319, 233, 311, 102,  74,
   180, 275,  66, 194, 152, 325, 172, 247, 244, 261,
   117, 158, 166, 354,  75, 144, 108, 312,  94, 186,
   303,  80, 234,  89, 195, 112, 340, 181, 345, 317,
   326, 276, 239, 167, 118, 313,  70, 355, 327, 253,
   190, 176, 271, 104,  98, 153, 103,  90,  76, 267,
   277, 248, 225, 262, 182,  84, 154, 235, 335, 168,
   331, 196, 341, 249, 162, 307, 148, 349, 263, 321,
   257, 243, 229, 356, 159, 119,  67, 187, 173, 145,
   240,  77, 304, 332, 314, 342, 109, 254,  81, 278,
   105,  91, 346, 318, 183, 250, 197, 328,  95, 155,
   169, 268, 226, 236, 264
};

static Word16 sort_1985[397] = {
     0,   4,   6, 129, 215, 304, 390,   7,   5,   3,
    47,  48,  49,  50,  51, 222, 223, 224, 225, 226,
   130, 216, 305, 391, 135, 221, 310, 396, 132, 218,
   307, 393, 133, 219, 308, 394, 136, 311, 134, 220,
   309, 395, 131, 217, 306, 392,  52,   2,   1, 137,
   312, 227,  19,  21,  12,  17,  18,  20,  16,  25,
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   228,  31, 138, 313,   9,  33,  11, 139, 314,  54,
   229,  28,  27, 140, 315,  34,  35,  29,  46,  32,
    30,  55, 230,  37,  36,  39,  38,  40, 141, 316,
    41,  42,  43,  44,  45,  56, 142, 231, 317,  63,
    73,  92, 340,  82, 324, 149, 353, 159, 334, 165,
   338, 178, 163, 254,  77, 168, 257, 153, 343,  57,
   248, 238,  79, 252, 166,  67,  80, 201, 101, 267,
   143, 164, 341, 255, 339, 187, 376, 318,  78, 328,
   362, 115, 232, 242, 253, 290, 276,  62,  58, 158,
    68,  93, 179, 319, 148, 169, 154,  72, 385, 329,
   333, 344, 102,  83, 144, 233, 323, 124, 243, 192,
   354, 237,  64, 247, 202, 209, 150, 116, 335, 268,
   239, 299, 188, 196, 298,  94, 195, 258, 123, 363,
   384, 109, 325, 371, 170, 370,  84, 110, 295, 180,
    74, 210, 191, 106, 291, 205, 367, 381, 377, 206,
   355, 122, 119, 120, 383, 160, 105, 108, 277, 380,
   294, 284, 285, 345, 208, 269, 249, 366, 386, 300,
   297, 259, 125, 369, 197,  97, 194, 286, 211, 281,
   280, 183, 372,  87, 155, 283,  59, 348, 327, 184,
    76, 111, 330, 203, 349,  69,  98, 152, 145, 189,
    66, 320, 337, 173, 358, 251, 198, 174, 263, 262,
   126, 241, 193,  88, 388, 117,  95, 387, 112, 359,
   287, 244, 103, 272, 301, 171, 162, 234, 273, 127,
   373, 181, 292,  85, 378, 302, 121, 107, 364, 346,
   356, 212, 278, 213,  65, 382, 288, 207, 113, 175,
    99, 296, 374, 368, 199, 260, 185, 336, 331, 161,
   270, 264, 250, 240,  75, 350, 151,  60,  89, 321,
   156, 274, 360, 326,  70, 282, 167, 146, 352,  81,
    91, 389, 266, 245, 177, 235, 190, 256, 204, 342,
   128, 118, 303, 104, 379, 182, 114, 375, 200,  96,
```

```
   293, 172, 214, 365, 279,  86, 289, 351, 347, 357,
   261, 186, 176, 271,  90, 100, 147, 322, 275, 361,
    71, 332,  61, 265, 157, 246, 236
};

static Word16 sort_2305[461] = {
     0,   4,   6, 145, 247, 352, 454,   7,   5,   3,
    47,  48,  49,  50,  51, 254, 255, 256, 257, 258,
   146, 248, 353, 455, 151, 253, 358, 460, 148, 250,
   355, 457, 149, 251, 356, 458, 152, 359, 150, 252,
   357, 459, 147, 249, 354, 456,  52,   2,   1, 153,
   360, 259,  19,  21,  12,  17,  18,  20,  16,  25,
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   260,  31, 154, 361,   9,  33,  11, 155, 362,  54,
   261,  28,  27, 156, 363,  34,  35,  29,  46,  32,
    30,  55, 262,  37,  36,  39,  38,  40, 157, 364,
    41,  42,  43,  44,  45,  56, 158, 263, 365, 181,
   192, 170,  79,  57, 399,  90, 159, 297, 377, 366,
   275,  68, 183, 388, 286, 194, 299, 92 ,  70, 182,
   401, 172,  59,  91,  58, 400, 368, 161,  81, 160,
   264, 171,  80, 389, 390, 378, 379, 193, 298,  69,
   266, 265, 367, 277, 288, 276, 287, 184,  60, 195,
    82,  93,  71, 369, 402, 173, 162, 444, 300, 391,
    98,  76, 278,  61, 267, 374, 135, 411, 167, 102,
   380, 200,  87, 178,  65,  94, 204, 124,  72, 342,
   189, 305, 381, 396, 433, 301, 226, 407, 289, 237,
   113, 215, 185, 128, 309, 403, 116, 320, 196, 331,
   370, 422, 174,  64, 392,  83, 425, 219, 134, 188,
   432, 112, 427, 139, 279, 163, 436, 208, 447, 218,
   236, 229,  97, 294, 385, 230, 166, 268, 177, 443,
   225, 426, 101, 272, 138, 127, 290, 117, 347, 199,
   414,  95, 140, 240, 410, 395, 209, 129, 283, 346,
   105, 241, 437,  86, 308, 448, 203, 345, 186, 107,
   220, 415, 334, 319, 106, 313, 118, 123,  73, 207,
   421, 214, 384, 373, 438,  62, 371, 341,  75, 449,
   168, 323, 164, 242, 416, 324, 304, 197, 335, 404,
   271,  63, 191, 325,  96, 169, 231, 280, 312, 187,
   406,  84, 201, 100,  67, 382, 175, 336, 202, 330,
   269, 393, 376, 383, 293, 307, 409, 179, 285, 314,
   302, 372, 398, 190, 180,  89,  99, 103, 232,  78,
    88,  77, 136, 387, 165, 198, 394, 125, 176, 428,
    74, 375, 238, 227,  66, 273, 282, 141, 306, 412,
   114,  85, 130, 348, 119, 291, 296, 386, 233, 397,
   303, 405, 284, 445, 423, 221, 210, 205, 450, 108,
   274, 434, 216, 343, 337, 142, 243, 321, 408, 451,
   310, 292, 120, 109, 281, 439, 270, 429, 332, 295,
   418, 211, 315, 222, 326, 131, 430, 244, 327, 349,
   417, 316, 143, 338, 440, 234, 110, 212, 452, 245,
   121, 419, 350, 223, 132, 441, 328, 413, 317, 339,
   126, 104, 137, 446, 344, 239, 435, 115, 333, 206,
   322, 217, 228, 424, 453, 311, 351, 111, 442, 224,
   213, 122, 431, 340, 235, 246, 133, 144, 420, 329,
   318
};

static Word16 sort_2385[477] = {
     0,   4,   6, 145, 251, 360, 466,   7,   5,   3,
    47,  48,  49,  50,  51, 262, 263, 264, 265, 266,
   146, 252, 361, 467, 151, 257, 366, 472, 148, 254,
   363, 469, 149, 255, 364, 470, 156, 371, 150, 256,
   365, 471, 147, 253, 362, 468,  52,   2,   1, 157,
   372, 267,  19,  21,  12,  17,  18,  20,  16,  25,
    13,  10,  14,  24,  23,  22,  26,   8,  15,  53,
   268,  31, 152, 153, 154, 155, 258, 259, 260, 261,
   367, 368, 369, 370, 473, 474, 475, 476, 158, 373,
     9,  33,  11, 159, 374,  54, 269,  28,  27, 160,
   375,  34,  35,  29,  46,  32,  30,  55, 270, 37,
    36,  39,  38,  40, 161, 376,  41,  42,  43,  44,
    45,  56, 162, 271, 377, 185, 196, 174,  79,  57,
   411,  90, 163, 305, 389, 378, 283,  68, 187, 400,
   294, 198, 307,  92,  70, 186, 413, 176,  59,  91,
```

```
    58, 412, 380, 165,  81, 164, 272, 175,  80, 401,
   402, 390, 391, 197, 306,  69, 274, 273, 379, 285,
   296, 284, 295, 188,  60, 199,  82,  93,  71, 381,
   414, 177, 166, 456, 308, 403,  98,  76, 286,  61,
   275, 386, 135, 423, 171, 102, 392, 204,  87, 182,
    65,  94, 208, 124,  72, 350, 193, 313, 393, 408,
   445, 309, 230, 419, 297, 241, 113, 219, 189, 128,
   317, 415, 116, 328, 200, 339, 382, 434, 178,  64,
   404,  83, 437, 223, 134, 192, 444, 112, 439, 139,
   287, 167, 448, 212, 459, 222, 240, 233,  97, 302,
   397, 234, 170, 276, 181, 455, 229, 438, 101, 280,
   138, 127, 298, 117, 355, 203, 426,  95, 140, 244,
   422, 407, 213, 129, 291, 354, 105, 245, 449,  86,
   316, 460, 207, 353, 190, 107, 224, 427, 342, 327,
   106, 321, 118, 123,  73, 211, 433, 218, 396, 385,
   450,  62, 383, 349,  75, 461, 172, 331, 168, 246,
   428, 332, 312, 201, 343, 416, 279,  63, 195, 333,
    96, 173, 235, 288, 320, 191, 418,  84, 205, 100,
    67, 394, 179, 344, 206, 338, 277, 405, 388, 395,
   301, 315, 421, 183, 293, 322, 310, 384, 410, 194,
   184,  89,  99, 103, 236,  78,  88,  77, 136, 399,
   169, 202, 406, 125, 180, 440,  74, 387, 242, 231,
    66, 281, 290, 141, 314, 424, 114,  85, 130, 356,
   119, 299, 304, 398, 237, 409, 311, 417, 292, 457,
   435, 225, 214, 209, 462, 108, 282, 446, 220, 351,
   345, 142, 247, 329, 420, 463, 318, 300, 120, 109,
   289, 451, 278, 441, 340, 303, 430, 215, 323, 226,
   334, 131, 442, 248, 335, 357, 429, 324, 143, 346,
   452, 238, 110, 216, 464, 249, 121, 431, 358, 227,
   132, 453, 336, 425, 325, 347, 126, 104, 137, 458,
   352, 243, 447, 115, 341, 210, 330, 221, 232, 436,
   465, 319, 359, 111, 454, 228, 217, 122, 443, 348,
   239, 250, 133, 144, 432, 337, 326
};

static Word16 sort_SID[35] = {
    0,  1,  2,  3,  4,  5,  6,  7,  8,  9,
   10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
   20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
   30, 31, 32, 33, 34
};

/* pointer table for bit sorting tables */
static Word16 *sort_ptr[16] = { sort_660, sort_885, sort_1265, sort_1425,
sort_1585, sort_1825, sort_1985, sort_2305,
                                sort_2385, sort_SID,      NULL,      NULL,
NULL,      NULL,      NULL,      NULL};
```