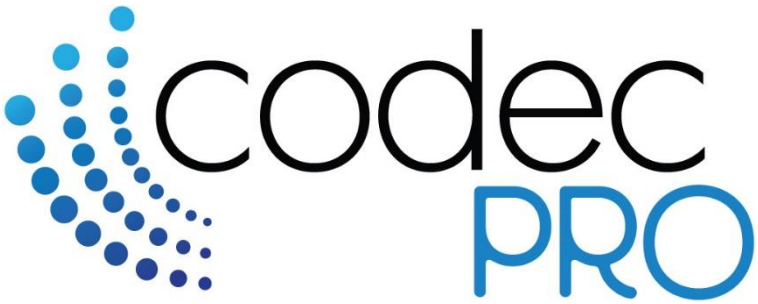

Experimental AMR Codec

(NON-COMMERCIAL use only)

Technical Documentation

Version 1.1 Revision A
February 2024

CodecPro Incorporated
2162 Laurier East
Montreal, QC H2H 1C2 CANADA
sales@codecpro.com
www.codecpro.com

Contents

Revision history

November 2013 First release of this document.

February 2024 Changed package content section to reflect the use of Visual Studio 2022.

References

[1] 3GPP 1999 TS 26.071, “AMR speech Codec; General description.” http://www.3gpp.org/ftp/Specs/html-info/26071.htm

[2] IETF (2007) RFC 4867, “Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codec.” http://www.ietf.org/rfc/rfc4867.txt

Experimental AMR Codec

We have chosen to offer a free AMR codec implementation that enables the developer community to better appreciate the strengths and capabilities of this technology.

The source code published by the standardization body is copyrighted by many organizations. Consequently, contributions to open forums are made in the form of object code, under specific licenses that place certain restrictions on the object code but do not prevent the implementation from being used in harmony with the typical open-source project licenses.

Please note that this Experimental AMR Codec is intended for personal or community experimentation usage. It cannot be used for commercial purposes as specified in the license terms to which it was agreed to during the download process from CodecPro's web site. For a commercial version please contact CodecPro at sales@codecpro.com.

AMR codec

AMR is an adaptive multi-rate narrowband speech codec with eight bit rate modes ranging from 4.75 kbps to 12.2 kbps and an additional low bit rate background noise mode. The codec includes a voice activity detector, a comfort noise generator and an error concealment mechanism, all of which improve speech quality over lossy transmission mediums. For a general description, please see [1].

The RTP payload format defined in [2] enables the use of AMR in RTP packet-switched networks in applications like streaming and provides interoperability with existing codec transport formats on non-IP networks.

Data input/output format

The input to the encoder and output from the decoder is 16-bit linear PCM speech, sampled at 8 kHz.

Bitstream frame layout

The output from the encoder and input to the decoder use the bitstream format illustrated in the following graphic:

Coding mode and Quality indicator	**Bitstream data**
(1 byte)	(size depends on mode)

The first byte of every frame (Byte 0) contains the coding mode and a quality indicator. The rest of the bytes in the frame are the encoded speech bitstream.

Byte 0 layout

Byte 0 contains the coding mode in bits 1 to 4 and the quality of this frame (0 for bad frame or 1 for good frame) in bit 5. The other bits in Byte 0 are not used.

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Pad	——— Mode ———				Quality	Pad	Pad

Modes

Possible mode values are 0-7, 8 and 15. Each mode has a corresponding number of bytes per frame, as shown in the following table. Modes 8 and 15 are automatically generated by the encoder when Voice Activity Detection (VAD) is enabled. Modes 9-14 are not used.

Mode	Bit Rate	Bytes per frame (zero-padded)
0	4.75 kbps	13
1	5.15 kbps	14
2	5.90 kbps	16
3	6.70 kbps	18
4	7.40 kbps	20
5	7.95 kbps	21
6	10.2 kbps	27
7	12.2 kbps	32
8	SID	6
15	no data	1

Discontinuous Transmission (DTX) mode

In a typical telephone conversation, voice transmission alternates frequently between the speaking parties, leaving long pauses of silence. These pauses can be efficiently represented as background noise and transmitted at a much lower bit rate than speech. The discontinuous transmission (DTX) mode is used to encode frames that contain only background noise.

When AMR operates in DTX mode, a voice activity detector (VAD) on the transmission (TX) side evaluates whether a frame contains any voice data. In the absence of speech, a silence information descriptor (SID) frame, which contains characteristics describing the background noise, is transmitted. On the reception (RX) side, a comfort noise generator (CNG) is used to synthesize background noise based on the SID frame parameters. On the TX side, the encoder generates 'no data' frames until it detects a change in the input signal (as background noise or speech).

Bad Frame Indicator (BFI)

To hide the effects of a corrupted or lost speech frame, the decoder can perform error concealment, using information from previous good frames. The `iBadFrame` argument is used to indicate the quality of each frame to the decoder and to enable error concealment. (See D_IF_amr_decode on page 10.)

Package Contents

cp_ amr.lib	Single channel, 64-bit static library of the AMR codec for Windows. Built using Microsoft Visual Studio 2022 and the following command line options: /permissive- /GS /W3 /Gy /Zc:wchar_t /Zi /Gm- /O2 /sdl /Zc:inline /fp:fast /D "NDEBUG" /D "_LIB" /D "_UNICODE" /D "UNICODE" /errorReport:prompt /WX- /Zc:forScope /Gd /Oi /MT /FC /Fa"x64\Release\" /EHsc /nologo /Fo"x64\Release\" /Fp"x64\Release\cp_amrnb.pch" /diagnostics:column
cp_amr.h	Header file with API prototypes to cp_amr.lib and constant definitions used by decoder.c and encoder.c.
decoder.c	Sample application that demonstrates how to use the cp_amrwb.lib decoder API.
encoder.c	Sample application that demonstrates how to use the cp_amrwb.lib encoder API.

AMR API functions

E_IF_amrnb_init

Description

Initializes resources needed for the encoder. To “reset” the encoder during normal operation, call `E_IF_amrnb_exit` and then call this function again.

Syntax

```
#include "typedef.h"
#include "cp_amr.h"

Word32 E_IF_amrnb_init(
       void);
```

Arguments

none

Return value

-1 if failed, 0 otherwise.

E_IF_amrnb_encode

Description

Encodes one frame of 16-bit linear PCM speech data.

Syntax

```
#include "typedef.h"
#include "cp_amr.h "

Word32 E_IF_amrnb_encode(
      Word16 iMode,
      Word16 *pSpeech,
      UWord8 *pBitstream,
      Word16 *iDtx
);
```

Arguments

`iMode`	(Input) Encoding mode {0..7}.
`pSpeech`	(Input) Buffer containing one frame of speech samples.
`pBitstream`	(Output) Buffer containing one frame of encoded bitstream data.
`iDtx`	(Input) Set to 1 to enable voice activity detection (VAD), 0 otherwise. For a description of codec operation when VAD is enabled, please see “Discontinuous Transmission (DTX) mode” on page 7.

Return value

Frame size in bytes.

E_IF_amrnb_exit

Description

Frees resources needed for the encoder.

Syntax

```
#include "typedef.h"
#include "cp_amr.h"

Word32 E_IF_amrnb_exit(
       void);
```

Arguments

none

Return value

none

D_IF_amrnb_init

Description

Initializes the resources needed for the decoder. To “reset” the decoder during normal operation, call `D_IF_amrnb_exit` and then call this function again.

Syntax

```
#include "typedef.h"
#include "cp_amr.h"

Word32 D_IF_amrnb_init(
       void
);
```

Arguments

none

Return value

-1 if failed, 0 otherwise.

D_IF_amrnb_decode

Description

Decodes one frame of encoded bitstream data.

Syntax

```
#include "typedef.h"
#include "cp_amr.h"

void D_IF_amrnb_decode(
       UWord8 *pBitstream,
       Word16 *pSynthSpeech,
       Word32 iBadFrame
);
```

Arguments

`pBitstream`	(Input) Buffer containing one frame of encoded bitstream data.
`pSynthSpeech`	(Output) Buffer containing one frame of speech samples.
`iBadFrame`	(Input) Bad frame indicator is used to inform the decoder of problems in the received frame. Set to 0 to indicate a good frame. Set to one to indicate bad or lost frame.

Return value

None

D_IF_amrnb_exit

Description

Frees resources needed for the decoder.

Syntax

```
#include "typedef.h"
#include "cp_amr.h"

Word32 D_IF_amrnb_exit(
       void);
```

Arguments

none

Return value

none