AN2812
Application note

Vocoder demonstration using a Speex audio codec on STM32F101xx and STM32F103xx microcontrollers

Introduction

This application note describes how to implement the codec Speex software on the STM32F101xx and STM32F103xx microcontrollers to build a vocoder application.

Speex is a free audio codec dedicated to speech encoding and decoding. It provides a high level of compression with a good quality of sound. That makes it a high-performance solution for any application using message playback or a voice recorder, like answering machines, building and home safety systems, intercoms, smart appliances, voice recorders or walkie-talkies.

Contents

List of tables

List of figures

1 Speex codec overview

The Speex codec is an open-source, patent- and royalty-free software dedicated to speech compression and decompression.

Speex is based on CELP (code-excited linear prediction) and designed to compress voice at bitrates ranging from 2 to 44 kbps.

The features of Speex include:
– Narrowband (8 kHz), wideband (16 kHz), and ultrawideband (32 kHz) compression in the same bitstream
– Intensity stereo encoding
– Packet loss concealment
– Variable bitrate operation (VBR)
– Voice activity detection (VAD)
– Discontinuous transmission (DTX)
– Fixed-point port
– Acoustic echo canceller
– Noise suppression

Note that Speex has a number of features that are not present in other codecs, such as intensity stereo encoding, integration of multiple sampling rates in the same bitstream (embedded coding), and a VBR mode.

For more information about the Speex codec, please refer to the Speex website: *www.speex.org*.

Note: *This application note was developed with the release 1.2rc1 of the Speex codec.*

2 The vocoder application

2.1 Vocoder application overview

The vocoder is a speech processing application that provides human voice treatment such as encoding, decoding, filtering and amplifying.

The typical implementation of a vocoder application consists of a speech processing module and audio input/output interfaces.

In this application note, the Speex codec firmware serves as the speech processing module, and the audio input/output interfaces are integrated using the embedded resources of the STM32F101xx and STM32F103xx, which are the ADC (for the input) and the PWM (for the output).

The STM32F101xx and STM32F103xx microcontroller family provides multi-channels ADCs with a 12-bit resolution and a 1 µs conversion time, as well as 16-bit timers with a PWM generation capably. These features can be used to build low-cost audio applications with high-performance results.

2.2 Hardware description

The hardware is made up of two audio stages: an input and an output, to capture and play voice (refer to *Figure 1*).

In this application note, voice capture is performed by the microcontroller's internal ADC that must be connected to a microphone through a pre-amplifier.

Voice generation is performed by the microcontroller's internal timer configured in PWM mode. The PWM output signal is low-pass filtered to remove high-frequency components, then amplified and sent to the speaker.

Figure 1. STM32F101xx and STM32F103xx vocoder hardware description
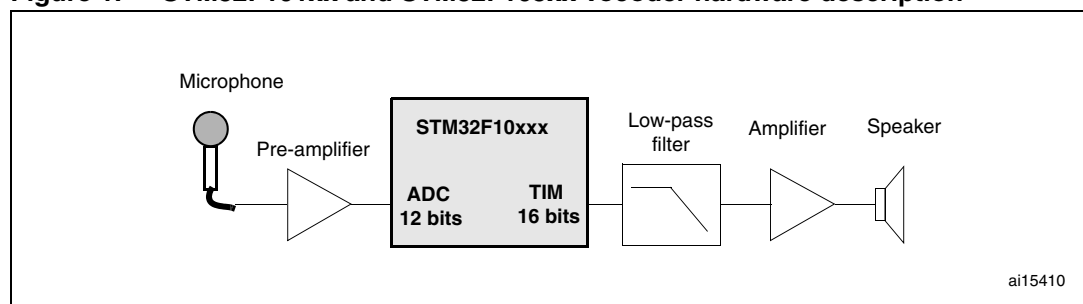
This application note was developed using the STM32F103-STK board from OLIMEX (OLIMEX website is *www.OLIMEX.com*).

This board contains an input audio interface and an output audio interface connected to two jacks themselves connected to headphones (or a speaker) and a microphone (refer to *Figure 2*).

The device implemented on the OLIMEX board is the STM32F103RBT6 which has 128 Kbytes of Flash memory and 20 Kbytes of RAM.

For more information about the STM32F103-STK board, please refer to the following web address: *http://www.olimex.com/dev/stm32-103stk.html*.

Figure 2.   OLIMEX STM32F103-STK audio input/output stages
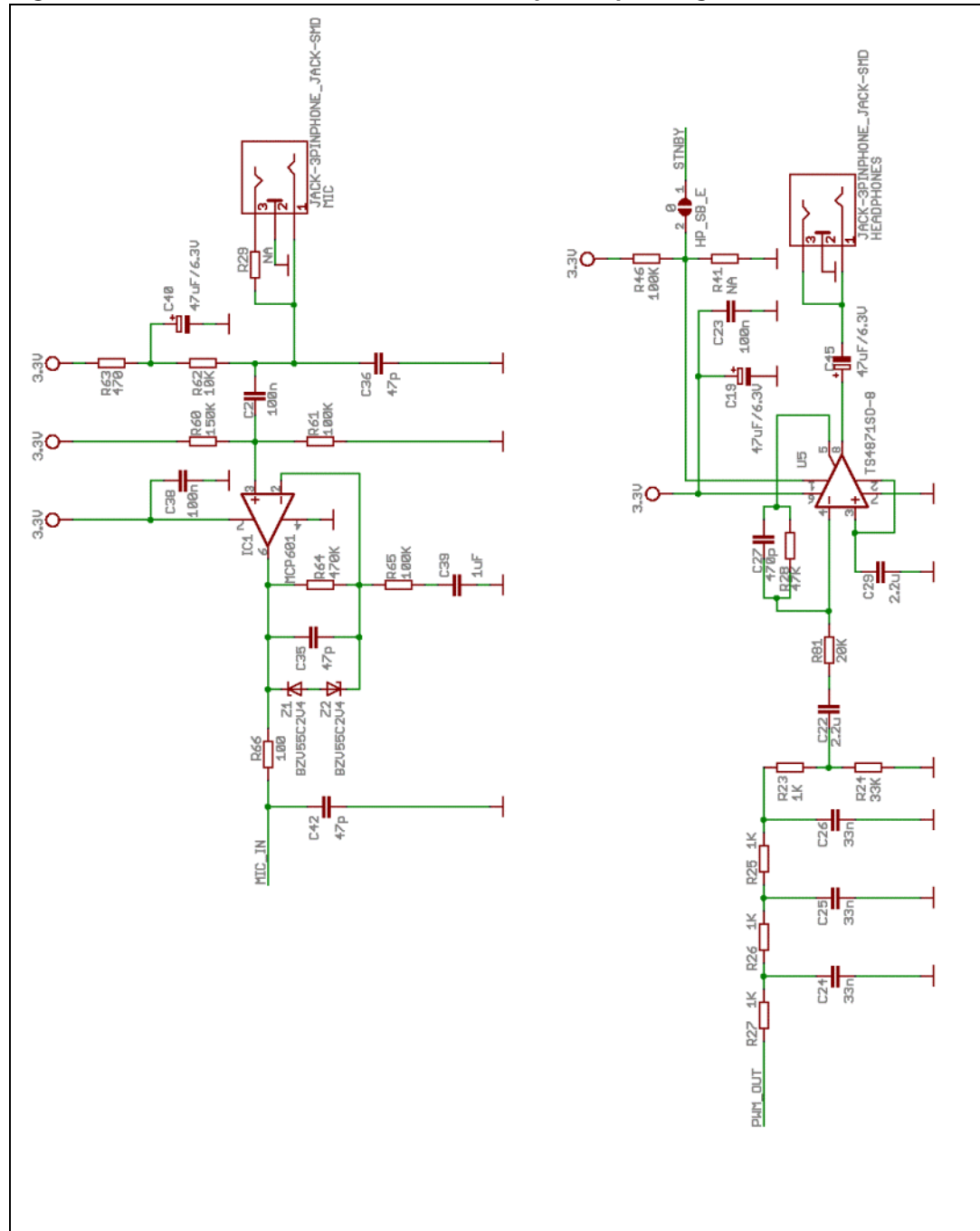
2.3 Speex codec implementation

2.3.1 Speex codec settings

In this application note the Speex codec is implemented to support narrowband voices (sampled at 8 kHz) and to improve the encoding at a bitrate equal to 8 kbps. This gives a compression ratio equal to 16:1.

The Speex codec has a parameter called "quality", that is used to control the bitrate. This parameter is configured in the encoding settings.

To set the bitrate to 8 kbps, the quality parameter must be equal to 4. Since the Speex is implemented at a **constant** bitrate (equal to 8 kbps), the quality parameter must be kept unchanged.

In this model of implementation, with the quality parameter equal to 4, the complexity of the encoder can only be set to a value lower than or equal to 2. Indeed, a complexity value greater than 2 would cause the codec to use extra CPU cycles, which is not in line with the expected quality level. To obtain the best quality, it is recommended to set the complexity to a value lower than or equal to 2.

2.3.2 Speex codec optimization

The Speex codec was modified to have a constant bitrate of 8 kbps and a complexity value lower than or equal to 2 with a view to reducing memory usage and increasing performance and, in so doing, optimizing the STM32F101xx and STM32F103xx microcontroller resources.

The files, structures and constant tables referring to the unused modes were removed and some functions were modified to fit the implementation model.

On the other hand, the following functions, which are among the functions that use most of the CPU load cycles, were written in assembly to take advantage of the ARM Cortex™-M3 core (thumb-2 instructions set):

● `filter_mem16`
● `fir_mem16`
● `iir_mem16`
● `inner_prod`
● `vq_nbest`

The Speex codec implementation on the STM32F101xx and STM32F103xx was developed and validated in the scope of this application note, using two different toolchains:

● IAR Embedded Workbench for ARM (Version: 5.11)
● Keil µVision3 RealView MDK-ARM (Version: 3.22)

The assembly code was developed using the ARM IAR assembler for the IAR EWARM toolchain and, using the ARM embedded assembler for the Keil RVMDK toolchain.

All modified and added files, that contain the assembly code, are provided separately in the **SpeexLib/stm32** directory in the firmware package.

Note: *The implementation could be easily tailored to any other toolchain (compiler & assembly) such as GNU.*

2.3.3 Speex codec requirements

Note that Speex encoding cannot run on STM32F101xx access line devices because the members of this family have a maximum frequency of 36 MHz and, at this frequency, the CPU load for encoding is of about 95%. The current firmware version therefore does not allow encoder implementation on the STM32F101xx access line family.
In this application note, Speex codec encoding is implemented on the STM32F103xx performance line family.

The resources required by the Speex codec running on the STM32F103xx at 72 MHz are given in *Table 1*.

Table 1.   Speex implementation requirements

Parameters	Resource requirement[1]
Encoder Flash memory size in bytes	31844
Encoder RAM size in bytes	6456
Decoder Flash memory size in bytes	31636
Decoder RAM size in bytes	3680
Encoder + decoder Flash memory size in bytes	31904
Encoder + decoder RAM size in bytes	7216
Encoding CPU load in %	52
Decoding CPU load in %	8

1. These values have been measured with the Keil RVMDK 3.22 toolchain at the -O3 optimization level, without using microlib.

These values were determined from a test demonstration that integrates only the Speex encoder and decoder with no additional treatment. The Speex codec is configured in the narrowband mode, with the quality parameter set to 4 and the complexity equal to 1.

2.4   Vocoder firmware description

The firmware is made up of three applications that illustrate the typical use of a vocoder:

1. Playback application: playing a message stored in the encoded format.
2. Record and Play:
    a) A speech is encoded and stored: recording phase
    b) The stored message is decoded and played: playing phase
3. Loopback application: basic implementation of the full-duplex codec. A voice message is captured and encoded. It is then immediately decoded and played in real time.

These applications have been assembled into a single demonstration that runs on the STM32F103-STK board from OLIMEX.

### 2.4.1 Playback application

The playback application consists in decoding and playing a Speex-encoded message that has been stored into the embedded STM32F101xx and STM32F103xx Flash memory.

The main of the playback application uses two buffers to decode the stored message. Once one buffer has been played, the application reuses it to decode another frame.

The decoded data is played using TIM2's handling interrupt function, which is called at a rate of 8 kHz.

Figure 3: Playback application flowchart shows in detail the principle of the playback application.

Figure 3.   Playback application flowchart
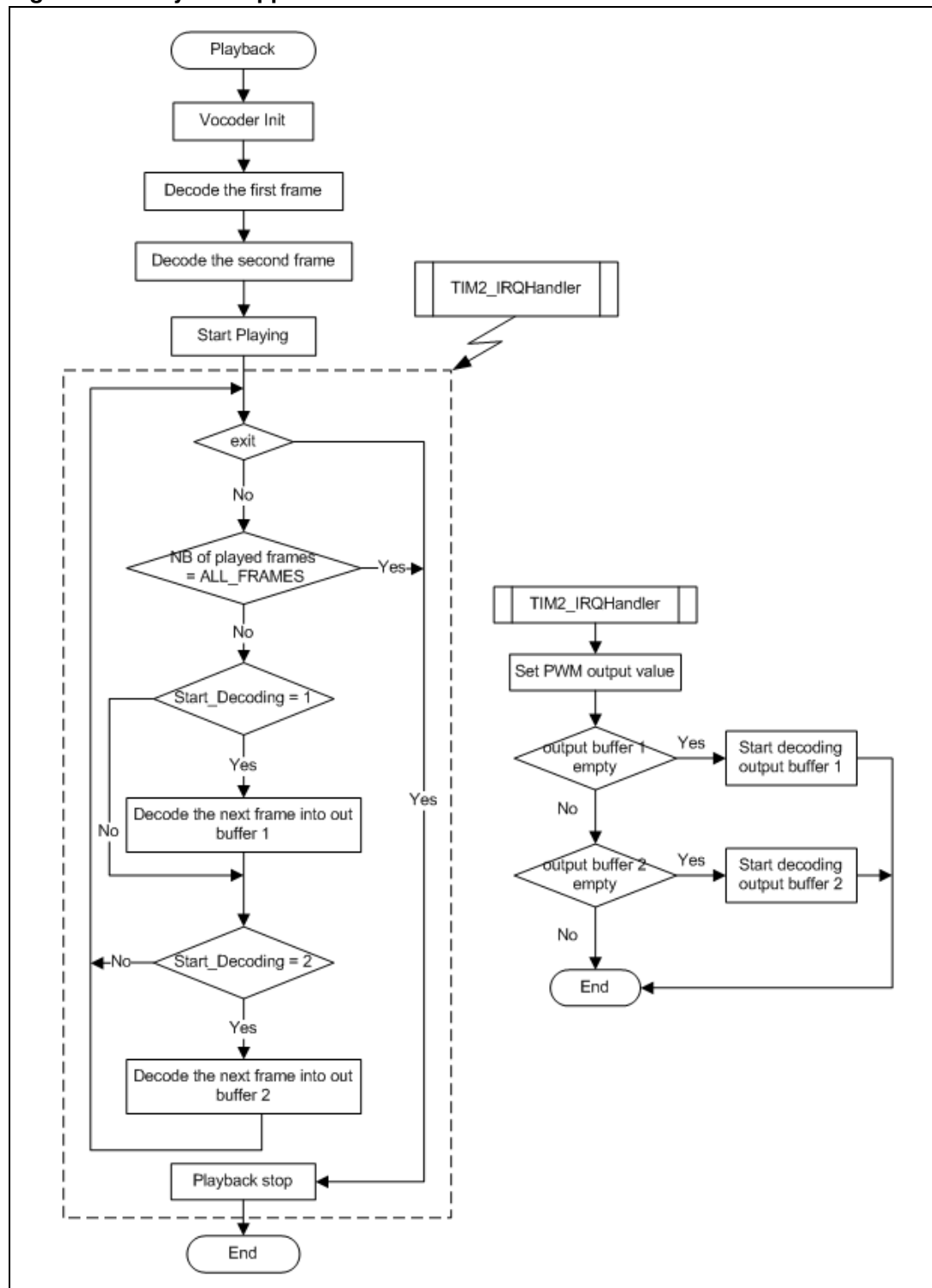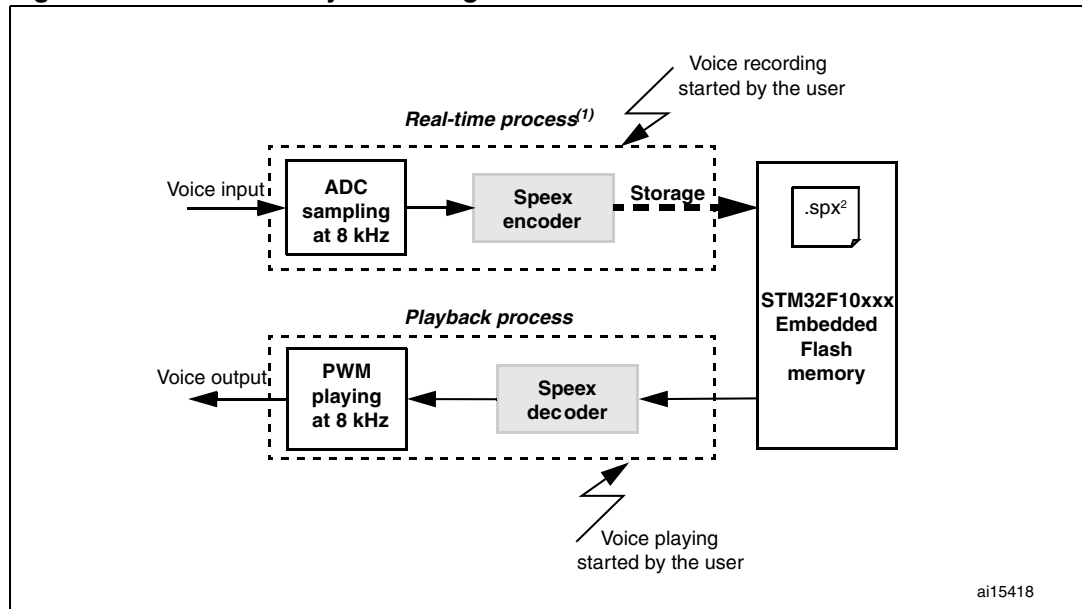
2.4.2 Record and play application

This application is very useful to store voice messages with few memory requirements. The user can record and store a voice message into the Flash memory embedded in the STM32F103xx, and then play it back.

With the demonstration you can store up to 1 minute of voice into the embedded Flash memory. For that, it is necessary to have 59 Kbytes of Flash memory used as data storage EEPROM by the firmware.

For comparison purposes, a raw PCM format used alone (without the Speex codec) to store 1 minute of voice requires 938 Kbytes of memory space. This implies the use of an external memory, which amounts to an additional cost.

The block diagram of the record and play application is represented in *Figure 4*. The recording and playing processes are independent so that you only need to make one recording to then play it back as many times as required.

Figure 4.   Record & Play block diagram

1. Capture, encoding and storage are performed in real time while speaking.

2. The .spx extension refers to a speech message compressed with the Speex codec.

The main codes for recording and playing in the record and play application are quite similar.

The main code for recording encodes the data and stores them into the embedded Flash memory, whereas the main code for playing reads and decodes the data from the embedded Flash memory.

The TIM3 interrupt handling function (TIM3_IRQHandler) controls speech input and output in the input/output stage made up of the ADC and the PWM.

TIM3 has been configured to generate an interrupt every 125 μs (8 kHz). At every call, the TIM3_IRQHandler checks whether it has been called for speech recording or for speech playing. Accordingly, the TIM3_IRQHandler performs either voice capture via the ADC or voice playing via the PWM.

The interaction between the two blocks, the Speex codec and the voice input/output stage is managed by using two different buffers.

While the codec is working on the first buffer, the voice I/O stage uses the second one. When the I/O reaches the end of the buffer, the codex and I/O stage swap buffers, the codec takes the second buffer and the I/O stage takes the first buffer, and vice versa.

The principle is fully described in the flowcharts shown in *Figure 5*, *Figure 6* and *Figure 7*, which present the main codes for recording and playing, and the TIM3 interrupt handling routine.
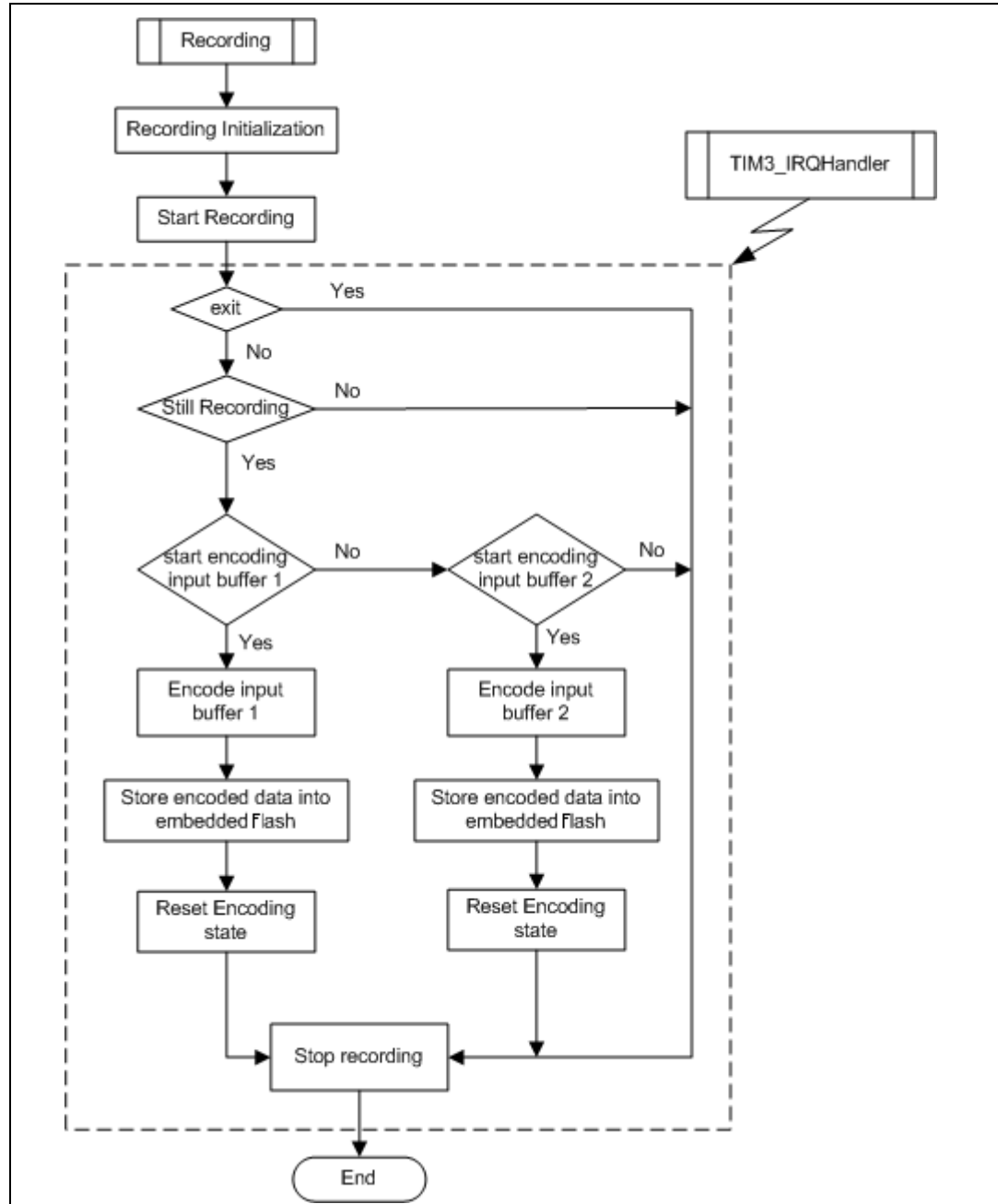
Figure 5.   Main record application flowchart

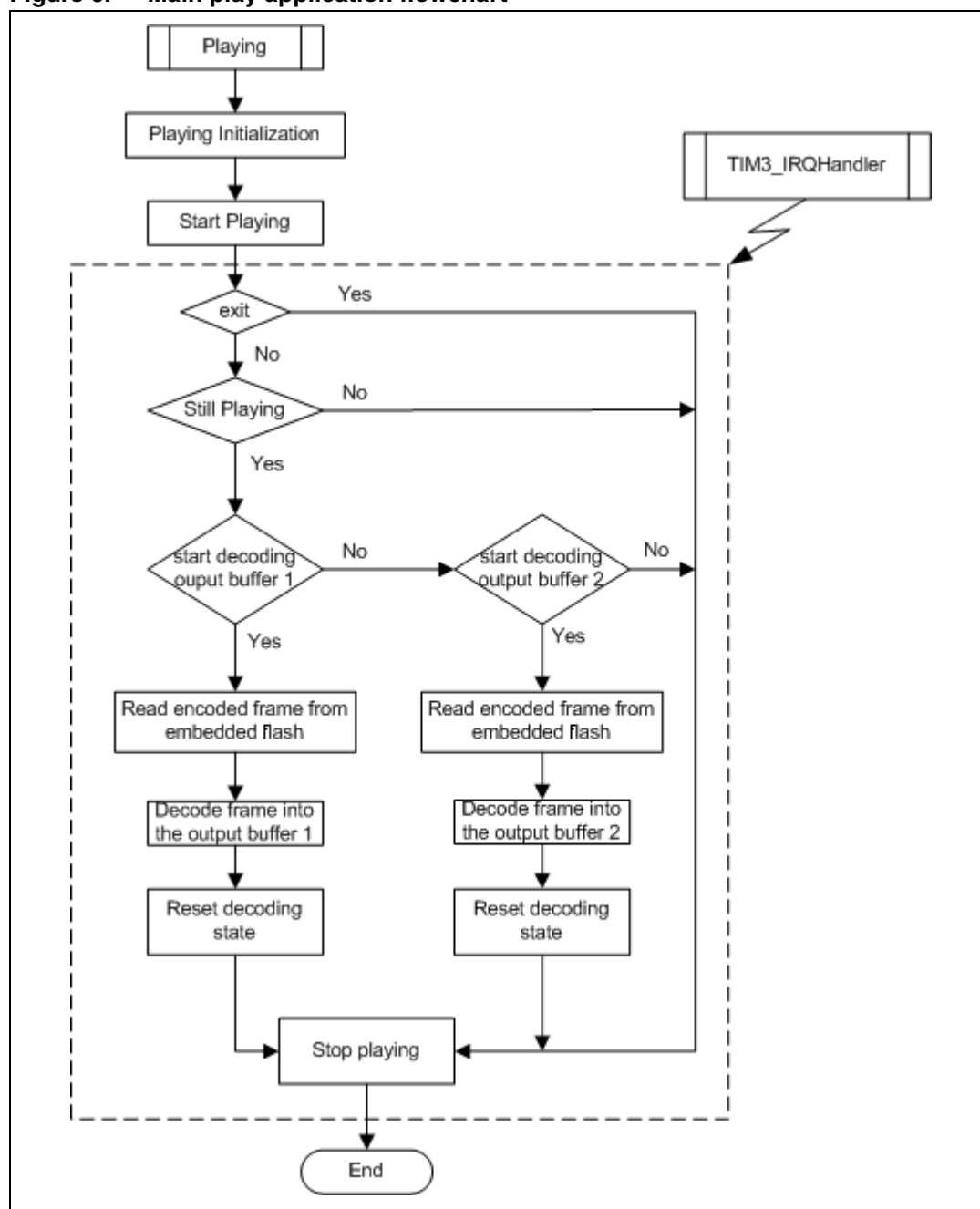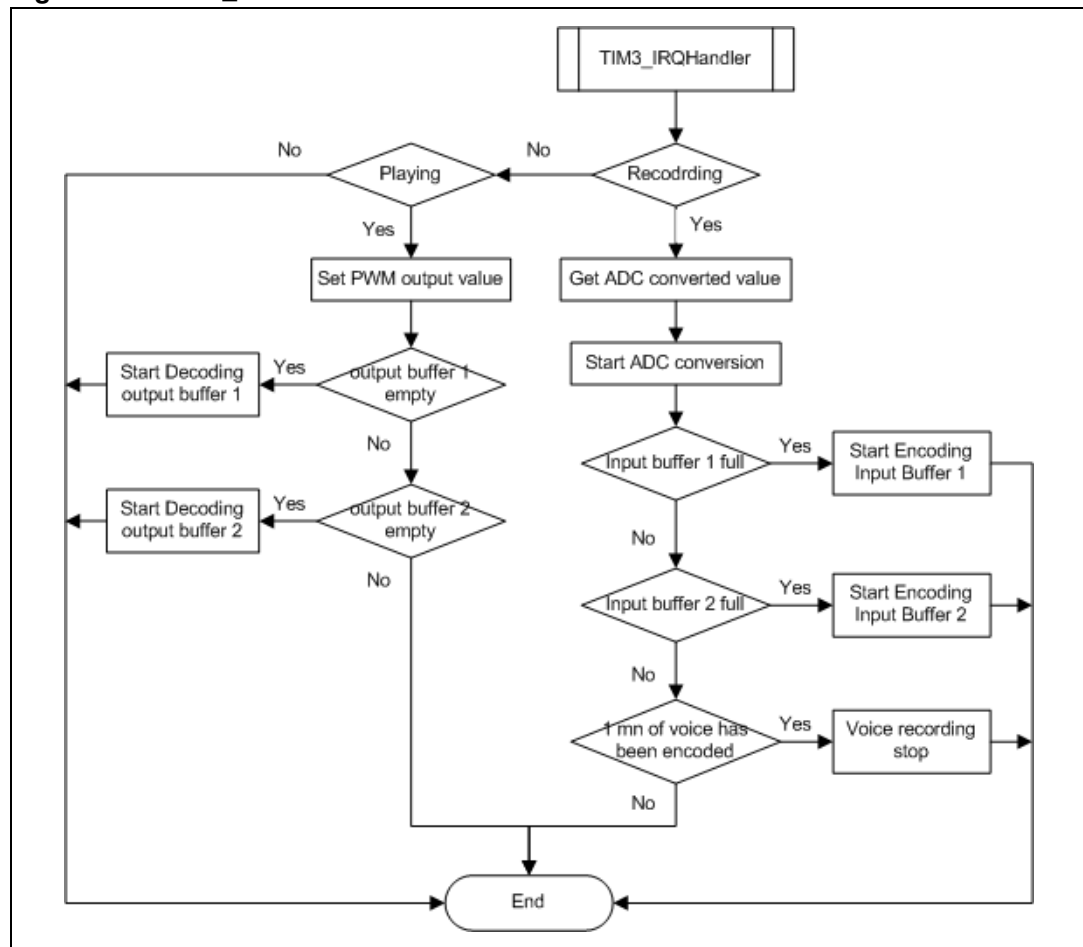**Figure 6.   Main play application flowchart**

Figure 7.   TIM3_IRQHandler flowchart

2.4.3   Loopback application

The loopback application demonstrates how to use the Speex codec in full-duplex mode. At every voice frame, the two encoding and decoding processes are executed in hard real-time. Figure 8 describes the principle of frame management in the loopback application.

Figure 8. **Real-time loopback principle**
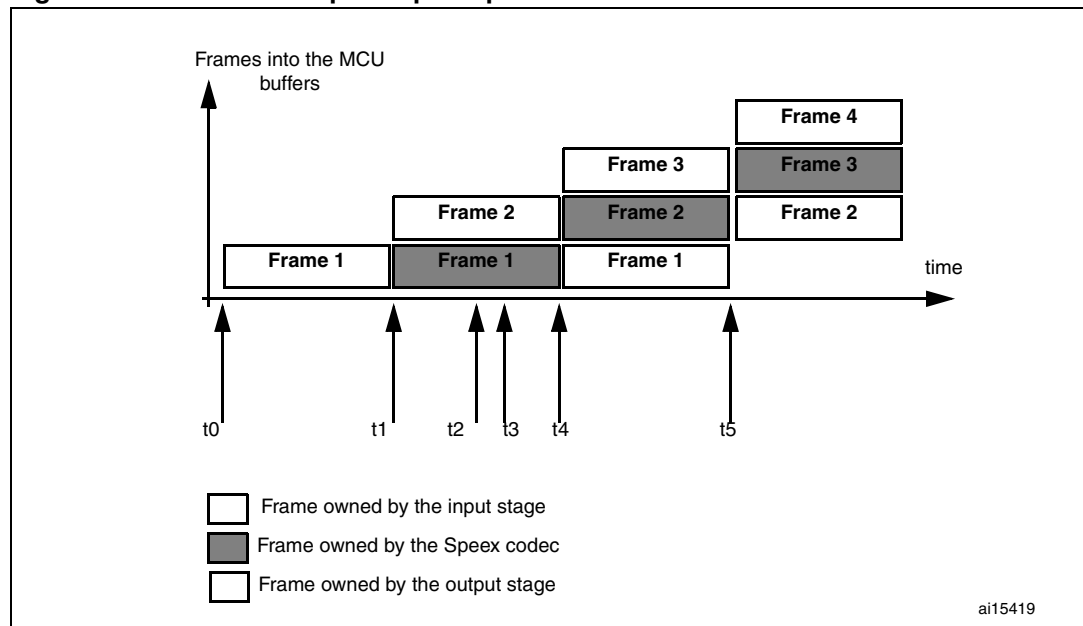
1. At t0:
    – Frame1 starts being recorded,

2. At t1:
    – Frame1 is ready in the input buffer and starts being encoded,
    – Frame 2 starts being recorded.

3. At t2:
    – Frame1 was encoded, and it starts being decoded,

4. At t3:
    – Frame1 was decoded and is ready in the output buffer,

5. At t4:
    – Frame 1 starts being played,
    – Frame 2 is ready in the input buffer, and starts being encoded,
    – Frame 3 starts being recorded.

6. At t5:
    – Frame1 was played,
    – Frame 2 was encoded/decoded and is being played,
    – Frame 3 is ready in the input buffer, and starts being encoded,
    – Frame 4 starts being recorded.

As described above, during continuous operation, there are always three frames in the RAM memory buffers.

So, we need to distinguish between the input and the output buffers. In total we need 4 buffers of 160 samples (signed short data) to improve the real-time loopback application.

The input/output operations are improved by the TIM2 interrupt handling routine, which is called at a rate of 16 kHz. In fact, for each call, the TIM2_IRQHandler routine performs either the input operation, by getting the ADC converted value, or the output operation by

setting the PWM pulse. This gives an executing rate equal to 8 kHz for input and output operations.

Figure 9 and *Figure 10* give the flowcharts describing the main and interrupt loopback operations.
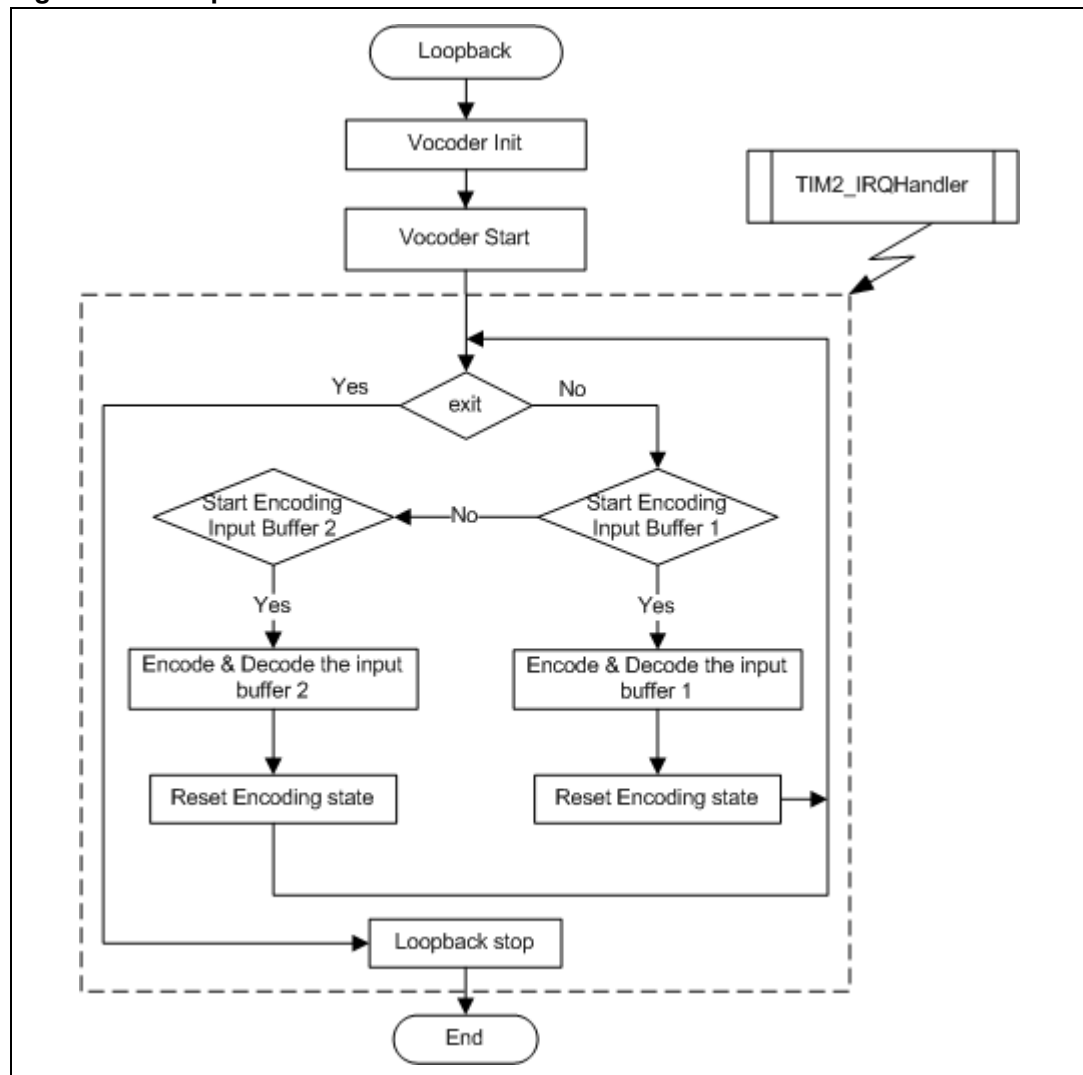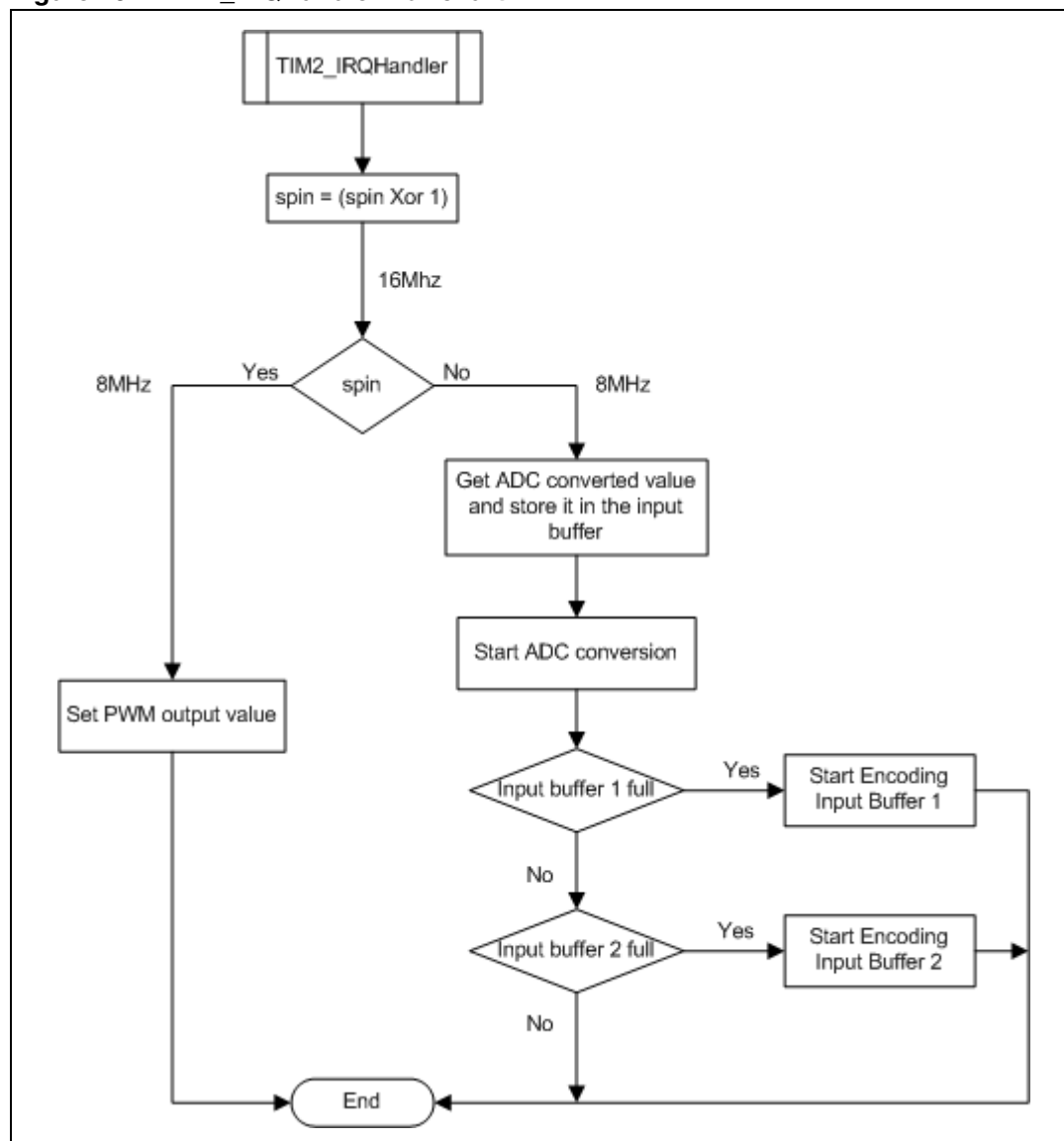
Figure 9.   Loopback flowchart

Figure 10.   TIM2_IRQHandler flowchart

3 Conclusion

This application note describes the implementation of a vocoder solution on the STM32F101xx and STM32F103xx family of microcontrollers using the Speex software codec:

Three different applications are provided:

● Playback application: to play encoded files

● Record and play application: to store a message in internal memory and play it back

● Loopback application: to run a real-time vocoder

Note: *The STM32F101xx access line devices can be used in Playback applications only as the current firmware version does not allow encoder implementation on these devices (whose frequency is limited to 36 MHz). There is no such limitation concerning the STM32F103xx family.*

The STM32F101xx and STM32F103xx microcontroller family supplies other solutions to build audio applications. For instance, the high-density STM32F101xx and STM32F103xx devices provide an embedded 12-bit DAC, which could be used for audio output. They also provide an I^{2}S interface, which could be used to link an external codec without extra hardware.

These alternatives could be used to improve the capabilities of the vocoder and audio applications at a reasonable cost while removing the need for a high-end device or a dedicated DSP engine in addition to a control MCU.

4 Revision history

Table 2. Document revision history

Date	Revision	Changes
15-Sep-2008	1	Initial release.
09-Oct-2008	2	*Table 1: Speex implementation requirements* modified. "Vocoder demo memory requirements" table removed.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.