

I. Overview

Sensory's SonicNet™ technology transmits information between one or more products using Sensory's RSC-4x line of microprocessors, using a speaker and/or microphone to send and receive information respectively. It can send the information discretely, or embed it within other audio output signals, like speech, music and sound effects. Since most RSC-4x applications already use microphones and speakers for speech recognition, SonicNet™ adds a layer of low-speed digital communication without adding additional components to an already existing system.

II. SonicNet™ Technology

SonicNet™ transmits 4 or 8 bits of data (a "token") using a sequence of tones. Each SonicNet™ tone (named t0 through t4) is a sine wave burst lasting about 55 msec. Each tone is separated from neighboring tones by a period of 110 msec. In order to transmit a 4-bit token, SonicNet™ will use a sequence of 5 tones lasting a total of 715 msec. For an 8-bit token, it will use a sequence of 8 tones lasting a total of 1.21 seconds.

SonicNet™ Tones				
t0	t1	t2	t3	t4
~7600 Hz	~7800 Hz	~8000 Hz	~8200 Hz	~8400 Hz

SonicNet™ as implemented on the RSC-4x requires oscillators which are accurate and stable to $\pm 0.6\%$. If an RSC-4x is used as the sender and receiver, this means they must both have stable oscillators. However, sonic tone sequences do not have to be transmitted by an RSC-4x based product. For example, they can be embedded and sent in a TV audio signal and received by an RSC-4x based product. In this case, similar to an RSC-4x based SonicNet™ transmitter, the TV set would also require a frequency stability of $\pm 0.6\%$ for SonicNet™ technology to work. In special cases, a SonicNet™ transmitter and receiver can be reconfigured to accept $\pm 1.0\%$ frequency stability, but this will tend to cause false-accept errors on noise, so this approach is only recommended when absolutely necessary.

The goal of Sensory's SonicNet™ technology is to transmit information over short range (1-5 meters) using high frequency tones that are close to the upper limit of hearing and are also hidden from a listener's perception by embedding them in audio signals containing speech or other lower frequency sounds. Sonic tone amplitudes can be configured to be one of three levels, corresponding to one-half of full scale, one-fourth of full scale, or one-eighth of full scale. A relatively long transmission range requires that sonic tones be louder, while hiding sonic tones in an audio signal requires they be softer. The application designer must balance these conflicting requirements to achieve an optimal design. In addition, the audio signal plus sonic tones must not saturate the dynamic range of the digital output otherwise this will produce noise, clicks, or pops in the received signal.

III. SonicNet™ Hardware Design

The following are recommended hardware considerations developers should consider when creating a SonicNet™ application. Consult Sensory's "Speech Recognition Hardware Design Guide" (80-0073), for a list of other hardware recommendations.

A. Setting Minimum/Maximum Separation Distance between Transmitter and Receiver.

The maximum separation distance between the transmitting speaker and the receiving microphone depends on room acoustics and echoes. Generally speaking, for a separation distance of up to 20 feet, the sonic tone gain

should be 1/2, for a 10 foot separation distance, the gain should be 1/4, and for a 5 foot separation distance, the gain should be 1/8. If a higher sonic tone gain is selected, and if the separation distance is less than about two feet, the receiving electronics may malfunction due to saturation. So there is an optimum range over which SonicNet™ works properly for a given sonic tone amplitude. If sonic tones are well-hidden in sound effects or music, or if it is not important whether the sonic tones are perceived by the user, then a gain of 1/2 can be selected, provided that operation over short distances is not required.

B. Selecting Microphone and Speaker.

The separation distances referenced above assume selection of proper high-quality microphones and speakers.

Many brands of electret microphones have flat frequency responses to 20 kHz, which is more than adequate. However, not all microphones meet this specification. Avoid microphones that do not have flat frequency responses to 20 kHz.

Selection of speakers is more difficult because many, but not all, inexpensive speakers are satisfactory and meet specifications. Thus, the best approach is to try several speakers and select one that produces audio signals that are sufficiently loud and that works as a sonic tone component to the distances required by the application.

C. Avoiding Absorbent Material In Front Of Microphone and Speaker.

Sound absorbing material in front of the microphone or speaker can degrade the performance of otherwise satisfactory components. The material covering microphones and speakers must be the minimum possible.

D. Designing Proper Frequency Response into the Electronic Circuit.

The electronic circuits connected to the microphone and the speaker must have upper frequency cutoffs of not less than 20 kHz. Several typical circuits have cutoffs at ~10 kHz and are not acceptable. The proper frequency responses can be achieved by proper selection of resistors and capacitors. The reference schematic shown the RSC-4128 data sheet (80-0206) is a good example of this.

E. Using DAC or PWM Outputs on RSC-4x Chips.

Sensory's RSC-4x chips have two audio output ports: the DAC and the PWM. The PWM output can directly drive a speaker while the DAC output requires an external audio amplifier to drive the speaker. Thus, while the PWM output requires fewer external components, it spreads the frequency of the sonic tones over a wider frequency range and has a smaller maximum separation distance over which SonicNet™ works. The DAC output with an external amplifier is therefore recommended unless cost restrictions eliminate this possibility.

IV. SonicNet™ Software Design

The following are recommended software considerations developers should consider when creating a SonicNet™ application. Consult Sensory's "Speech Recognition Software Design Guide" (80-0305), for a list of other software recommendations.

A. Setting Rejection Parameter to Manage False-Accept/False-Reject Errors.

The SxDetectToken parameter, *rejection*, which has allowable values of 0, 1, or 2, affects both the false-accept rate and the maximum range of reception of true signals. For *rejection* = 0, in a noisy environment, there may be very many false-accepts per hour. For *rejection* = 1, the rate of false-accepts is decreased substantially at the expense of decreasing the maximum separation distance to about 80% of the distance achievable for

rejection = 0. If audible detection of the sonic tones by a user is not an issue, *rejection* = 1 should be used, and the sonic tone gain should be set to 1/2. If audible detection of the sonic tones by a user is an issue, then *rejection* = 0 may be used. Applications should only use *rejection* = 2 when the noise environment is severe.

B. Recovering From Missed or Unexpected SonicNet™ Tokens.

The application program knows where in the game flow it is at any time. If it fails to receive a SonicNet™ token within a specified time, or if it receives a token different from that expected at that point in the flow, the application program should take steps to repeat the SonicNet™ transmission, or jump to some default activity in place of the expected SonicNet™ response to hide the failure..

C. Controlling the Timing between SonicNet™ Token Reception and Response.

Sensory's FluentChip™ allows for the transmitter to embed a SonicNet™ tokens at any point in the audio waveform. FluentChip's™ SxDetectToken API call returns control to the calling program immediately upon receiving a valid token. It may be that the receiver will need to delay its response relative to this moment of reception so as to let the transmitter finish outputting the entire audio waveform.

D. Placing SonicNet Tokens at Optimal Locations in the Audio Signal.

When embedding tokens in audio waveforms, they should be placed in the audio signal at locations where the amplitude of the audio signal is large, in order to better hide the tones from perception.

E. Combining SonicNet™ Technology with other FluentChip technologies.

Unfortunately, it is not currently possible for Sensory's RSC-4x chips to perform SonicNet™ detection simultaneously with any Sensory technologies other than SX speech.

F. Sending Multiple SonicNet Tokens

If an application must transmit more than one sonic tone sequence at a time because more than 24 different signals are to be transmitted, the string of sonic tone sequences should be separated by at least 0.25 seconds in order for the room echoes from previous sonic tone sequences to fade

G. Considerations When Tokens Are Embedded in SX Speech and Sound Effects.

The application can use the `_SxAddToken` function (`SxAddToken` macro in assembly language) to start transmission of a token that is added to ongoing SX speech or other sound output. There are several considerations and limitations that must be taken into account:

- SX speech must be encoded at the 9.3 KHz rate.
- Embedding tones in "sentences" played via `_PlaySnd` rather than individual SX sounds may not work correctly and should be avoided. ("Sentences" as used here are lists of SX sounds and optional silences periods created with the QuickSynthesis4 tool).
- The output volume of speech playback may need to be reduced by as much as one half because the addition of tones during playback may cause overflows that will produce audible noise and distortion.

- The SX interrupt handlers that are installed with the SonicNet™ library use more processor cycles than the default versions. For this reason it is easy to run out of cycles for SX playback, even when SonicNet™ tones are not being produced. It is usually necessary to select only one of the audio output options (DAC_OUT or PWM_OUT) in CONFIG.MCA rather than having both enabled, which is the default situation. Having a single output option will save some processor cycles.

V. SonicNet™ Application

A SonicNet™ sample application is provided with Sensory's FluentChip™. For complete details on the various API calls and functions discussed here, refer to the Sensory FluentChip™ Reference .chm file.

In order to use SonicNet™, the application must add the library file "sonicnet.mcl" to the project. This library file is located in the "lib" folder.

SonicNet™ applications will need to decide whether 4-bit tokens (16 different values) or 8-bit tokens (256 different values) will be used. The number of bits per token is passed to each of the three SonicNet™ functions and must be the same for the transmitter and receiver. Note that tokens are passed to and from the SonicNet™ functions as 16-bit values, because it is anticipated that the number of bits per token may get larger in future.

When transmitting a token, the application must select one of three available tone tables. The tone tables are part of the SonicNet™ library. To save memory, applications should use only one table. The tables differ in the amplitude of the tones:

SonicNetToneTable2	Tones are one-half of full scale
SonicNetToneTable4	Tones are one-fourth of full scale
SonicNetToneTable8	Tones are one-eighth of full scale

The tone table is declared in C as follows:

```
extern cdata uchar SonicNetToneTable2[];
#define TONE_TABLE      (uint)SonicNetToneTable2
```

The equivalent declaration in assembly language would be:

```
.extrn (const) SonicNetToneTable2
TONE_TABLE = SonicNetToneTable2
```

Use the _SxPlayToken function to transmit a token on its own (not embedded in speech). The following example transmits an 8-bit token stored in a variable called txToken (in this example, the token value is 128):

```
uint txToken;
...
txToken = 128;          // token to transmit
...
_SxPlayToken(TONE_TABLE, txToken, 8);
```

_SxPlayToken will return after transmission is complete. Note that _SxPlayToken calls _PlaySnd with a pointer to a special sequence of commands stored in code space ROM. This special sequence of commands will initiate the token transmission. During token transmission the SX handler will be called periodically and the SX memory handler will be called to retrieve this special command sequence.

Adding a token to other synthesized audio output requires adding code to both the main application, and the SX_H callout handler. In this example, `sx_h_token`, `sx_h_delay`, and `sx_h_tonetable` are global variables defined in main program module, but used in the callout handler module.

```

uchar sx_h_delay;
uint sx_h_token;
uint sx_h_tonetable;
...

sx_h_token = 128;
sx_h_delay = 4;
sx_h_tonetable = TONE_TABLE;
_SxTalk((long)&SX_here_i_am, SX_VOLUME);

```

Here, the main application program has set a counter called “`sx_h_delay`” to time out in 4 callouts (about 100 mS).

```

void _SxTalkHandler()
{
    . . .
    if (sx_h_delay) {
        if (--sx_h_delay == 0) {
            _SxAddToken(sx_h_tonetable, sx_h_token, 8);
        }
    }
}

```

`_SxAddToken()` starts the transmission a token that is added to the audio output currently being played out to the DAC or PWM, and returns immediately.

Detection of tokens is accomplished via the function `_SxDetectToken`. It has three arguments: an optional timeout period, a noise rejection level, and the number of bits per token.

`_SxDetectToken` will call a handler function provided by the application. The handler function can trigger an abort exit of `_SxDetectToken`.

A 16-bit value is returned by `_SxDetectToken`.

```

// Call _SxDetectToken with a 10 second timeout. Timeout is specified
// in units of about 1/73 second (about 13.7 msec). A timeout of zero
// means no timeout.
//
// It will return with:
//     0xffff      timed out
//     0xfffe      abort signalled from _SxDetectTokenHandler
//     0..(N-1)    token detected

uint result;

...

result = _SxDetectToken(10*73, REJECTION_LEVEL, NUM_BITS);

```

The Interactive Speech™ Product Line

The Interactive Speech line of ICs and software was developed to “bring life to products” through advanced speech recognition and audio technologies. It is designed for cost-sensitive consumer-electronic applications such as home electronics, home automation, toys, and personal communication. The product line includes the award-winning RSC-4x general-purpose microcontrollers and tools, the *VR Stamp™* 40 pin DIP module and tools, the SC series of speech and music synthesis microcontrollers. Our suite of software development kits are designed to run on non-Sensory processors and DSP's, and support most popular operating systems.

RSC Microcontrollers and Tools

The RSC product family contains low-cost 8-bit speech-optimized microcontrollers designed for use in consumer electronics. All members of the RSC family are fully integrated and include A/D, pre-amplifier, D/A, ROM, and RAM circuitry. The RSC family can perform a full range of speech/audio functions including speech recognition, speaker verification, speech and music synthesis, and voice recording/playback. The family is supported by a complete suite of evaluation and development toolkits.

Speech Recognition Modules and Tools

The VR Stamp™ is a complete speech recognition module based on the RSC-4x and is ideal for fast design and easy production. A low-noise audio channel and standardized 40-pin DIP footprint allow rapid prototyping, less debugging, and shorter time to market. The *VR Stamp Toolkit* includes everything needed to get started today, including VR Stamps, Module Programming Board, sample applications, and a complete set of development tools featuring the Phyton IDE and limited-life C compiler, QuickSynthesis™ 4 and Quick T2SI-Lite™ speech tools.

SC Microcontrollers and Tools

The SC-6x product family features the highest quality speech synthesis ICs at the lowest data rate in the industry. The line includes a 12.32 MIPS processor for high-quality, low data-rate speech compression and MIDI music synthesis, with plenty of power left over for other processing and control functions. Members of the SC-6x line can store as much as 37 minutes of speech on-chip and include as many as 64 I/O pins for external interfacing. Integrating this broad range of features into a single chip enables developers to create products with high quality, long duration speech at very competitive price points.

FluentSoft™ Technology

FluentSoft™ Recognizer is the engine powering the FluentSoft™ SDK. It provides a noise-robust, large-vocabulary, speaker-independent solution with continuous digit recognition and word-spotting capabilities. This small-footprint software recognizes up to 5,000 words; runs on non-Sensory processors including Intel XScale, TI OMAP, and ARM9 platforms; and supports operating systems such as MS Windows, Linux, and Symbian.

3Dmsg™ Technology

3Dmsg's (www.3Dmsg.com) Animated Speech technology offers animated avatars with advanced speech recognition and synthesis capabilities for use in smartphones, language trainers, and kiosk applications. Facial expressions can be configured to show emotions and lip synchronization can be automatically driven from voice or text data.

Important notices:

Sensory Incorporated (Sensory, Inc.) reserves the right to make changes, without notice, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Sensory, Inc. assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Sensory, Inc. makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Safety Policy:

Sensory, Inc. products are not designed for use in any systems where malfunction of a Sensory, Inc. product can reasonably be expected to result in a personal injury, including but not limited to life support appliances and devices. Sensory, Inc. customers using or selling Sensory Incorporated products for use in such applications do so at their own risk and agree to fully indemnify Sensory, Inc. for any damages resulting from such improper use or sale.



575 N. Pastoria Ave., Sunnyvale, CA 94085
Tel: (408) 625-3300 Fax: (408) 625-3350

© 2007 SENSORY, INC. ALL RIGHTS RESERVED.
Sensory is registered by the U.S. Patent and Trademark Office.

All other trademarks or registered trademarks are the property of their respective owners.