



Using the ADPCM Algorithm in Dialogic[®] Voice Processing Applications



Executive Summary

This application note describes the algorithm known as Adaptive Differential Pulse Code Modulation (ADPCM) as used in Dialogic® Voice Processing applications. The ADPCM or Voice Operated eXchange (VOX) is in audio file format, optimized for storing digitized voice data at a low sampling rate. These files are commonly used in telephony applications.

This application note covers information on encoding audio in ADPCM format for use in Dialogic® Voice Processing applications.

Table of Contents

Introduction.....	2
VOX File Format Specification.....	2
ADPCM Encoding Algorithm.....	2
ADPCM Decoding Algorithm.....	4
Calculation of Step Size.....	4
Initial Conditions.....	5

Introduction

This application note provides the steps to implement the Adaptive Differential Pulse Code Modulation (ADPCM) algorithm or commonly referred to as Voice Operated eXchange (VOX) as used in Dialogic® Voice Processing applications. The following topics are covered herein:

- VOX file format for voice data files
- ADPCM encoding algorithm
- ADPCM decoding algorithm
- Step size determination
- Initial and reset conditions

VOX File Format Specification

VOX files are flat binary files containing digitized voice data samples. Each byte contains two samples. There is a direct relationship between positional offset within the file and time, as expressed in the following formula:

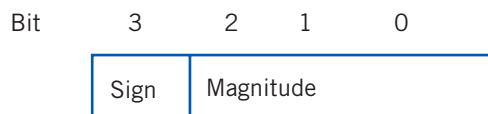
$$T(i) = 2i * I/SR$$

where: T(i) is the time offset in seconds from the beginning of the file of byte number “i” within the file and SR is the sampling rate in samples per second.

The encoding within each byte is as follows:



The encoding within each sample is ADPCM. This is a differential coding scheme in which each sample approximates the difference between the present input value and the previous one. The weighting of the magnitude portion of the difference is adaptive (non-linear); that is, it can change after each sample.



Sign: Positive (0) or negative (1) sample.

Magnitude: Change (0 to 7) from previous sample.

ADPCM Encoding Algorithm

Figure 1 shows a block diagram of the ADPCM encoding process. A linear input sample X(n) is compared to the previous estimate of that input X(n-1). The difference, d(n), along with the present step size, ss(n), is presented to the encoder logic. This logic, described below, produces an ADPCM output sample. This output sample is also used to update the step size calculation ss(n+1), and is presented to the decoder to compute the linear estimate of the input sample.

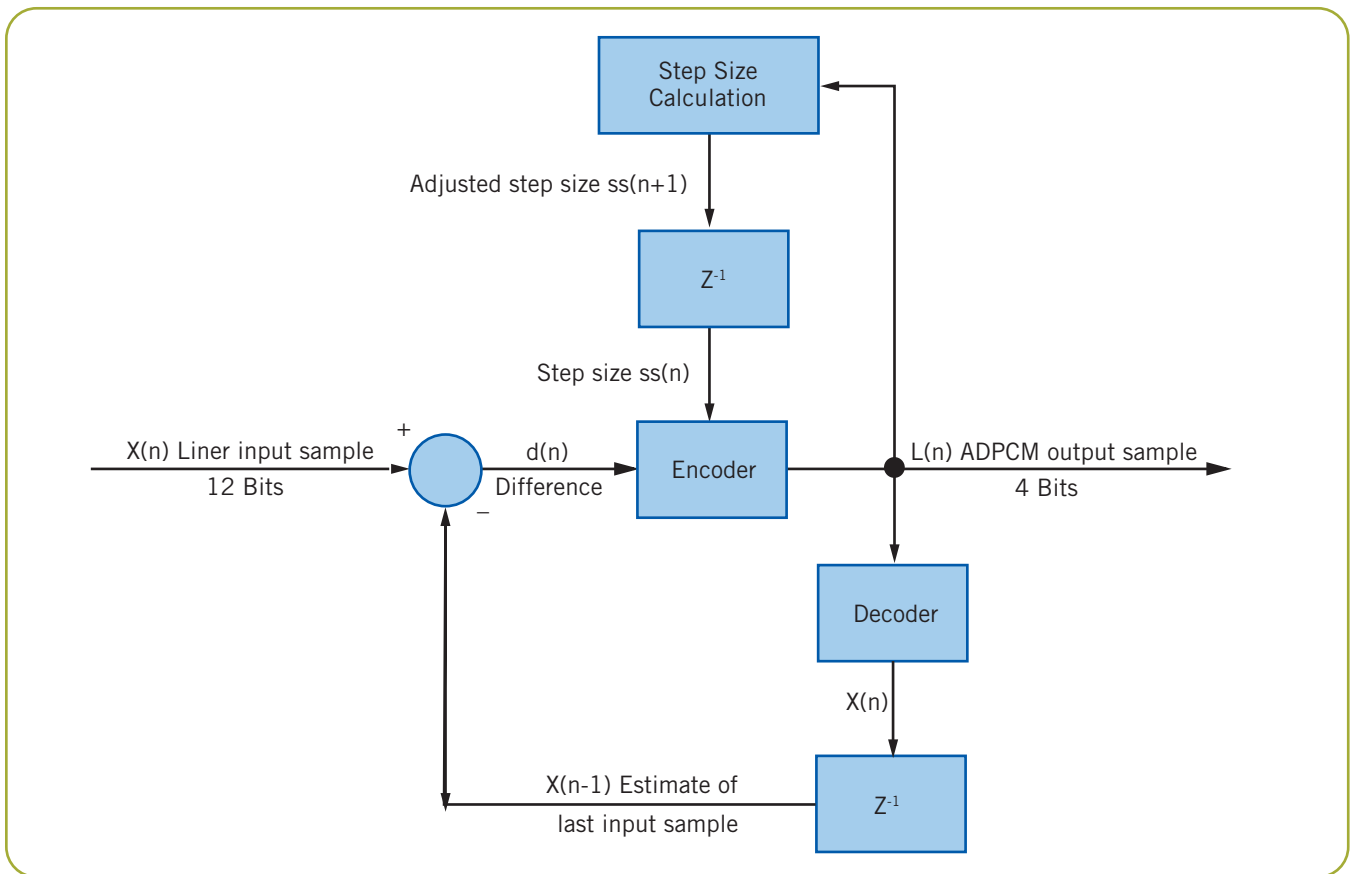


Figure 1. Block Diagram ADPCM Encoding Process

The encoder accepts the differential value, $d(n)$, from the comparator and the step size, and calculates a 4-bit ADPCM code. The following is a representation of this calculation in pseudocode:

```

let B3 = B2 = B1 = B0 = 0
if (d(n) < 0)
then B3 = 1
d(n) = ABS(d(n))
if (d(n) >= ss(n))
then B2 = 1 and d(n) = d(n) - ss(n)
if (d(n) >= ss(n) / 2)
then B1 = 1 and d(n) = d(n) - ss(n) / 2
if (d(n) >= ss(n) / 4)
then B0 = 1
L(n) = (10002 * B3) + (1002 * B2) + (102 * B1) + B0

```

Note: For the calculation of $ss(n)$, see the *Calculation of Step Size* section below.

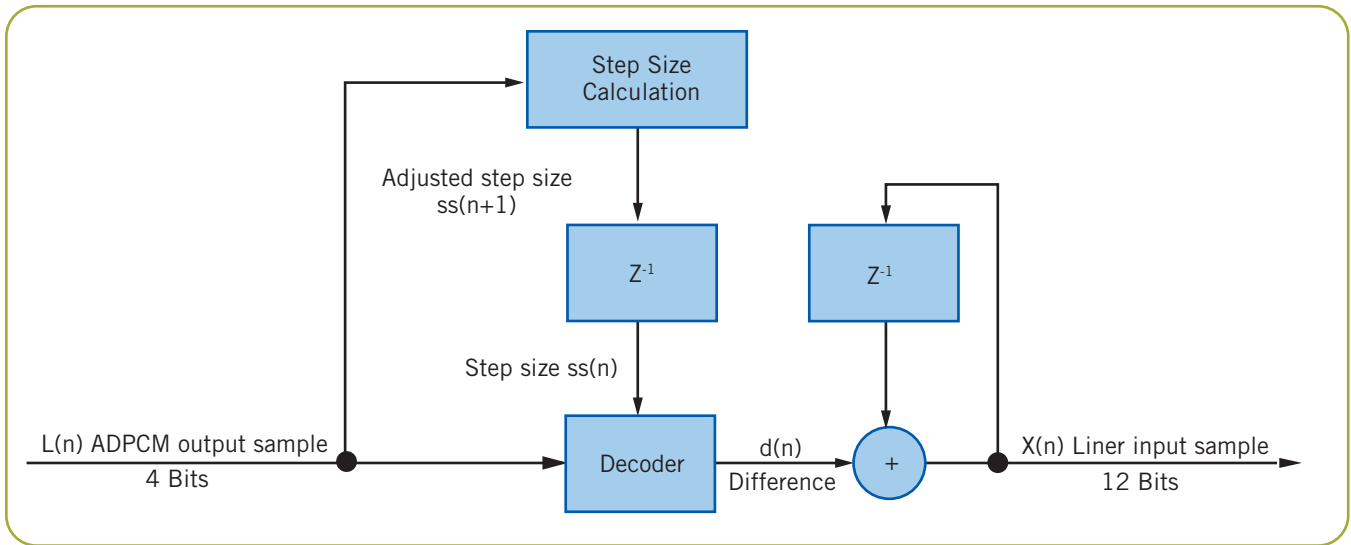


Figure 2. Block Diagram ADPCM Decoding Process

ADPCM Decoding Algorithm

Figure 2 shows a block diagram of the ADPCM decoding process. An ADPCM sample is presented to the decoder. The decoder computes the difference between the previous linear output estimate and the anticipated one. This difference is added to the previous estimate to produce the linear output estimate. The inputted ADPCM sample is also presented to the step size calculator to compute the step size estimate.

The decoder accepts ADPCM code values, $L(n)$, and step size values. It calculates a reproduced differential value, and accumulates an estimated waveform value, X . Here is a pseudocode algorithm:

```

d(n) = (ss(n) * B2) + (ss(n) / 2 * B1) + (ss(n) / 4 * B0) + (ss(n) / 8)
if (B3 = 1)
then d(n) = d(n) * (-1)
X(n) = X(n-1) + d(n)

```

Note: For the calculation of $ss(n)$, see *Calculation of Step Size* section.

Calculation of Step Size

For both the encoding and decoding process, the ADPCM algorithm adjusts the quantizer step size based on the most recent ADPCM value. The step size for the next sample, $n+1$, is calculated with the following equation:

$$ss(n+1) = ss(n) * 1.1M(L(n))$$

This equation can be implemented efficiently using two lookup tables. Table 1 uses the magnitude of the ADPCM code as an index to look up an adjustment factor. The adjustment factor is used to move an index pointer located in Table 2. The index pointer then points to the new step size. Values greater than 3 will increase the step size; values less than 4 will decrease the step size.

L(n) Value M(L(n))	Adjustment Factor
1111 or 0111	+8
1110 0110	+6
1101 0101	+4
1100 0100	+2
1011 0011	-1
1010 0010	-1
1001 0001	-1
1000 0000	-1

Table 1. M(L(n)) Values

No.	Step Size	No.	Step Size	No.	Step Size	No.	Step Size
1	16	13	50	25	157	37	494
2	17	14	55	26	173	38	544
3	19	15	60	27	190	39	598
4	21	16	66	28	209	40	658
5	23	17	73	29	230	41	724
6	25	18	80	30	253	42	796
7	28	19	88	31	279	43	876
8	31	20	97	32	307	44	963
9	34	21	107	33	337	45	1060
10	37	22	118	34	371	46	1166
11	41	23	130	35	408	47	1282
12	45	24	143	36	449	48	1411
						49	1552

Table 2. Calculated Step Sizes

This method of adapting the scale factor with changes in the waveform is optimized for voice signals, not square waves or other non-sinusoidal waveforms.

Initial Conditions

When the ADPCM algorithm is reset, the step size $ss(n)$ is set to the minimum value (16) and the estimated waveform value X is set to zero (half scale). Playback of 48 samples (24 bytes) of plus and minus zero (10002 and 00002) will reset the algorithm. Twenty-four bytes of 08 Hex or 80 Hex will satisfy this requirement. It is necessary to alternate positive and negative zero values because the encoding formula always adds 1/8 of the quantization size. If all values were positive or negative, a DC component would be added that would create a false reference level.

To learn more, visit our site on the World Wide Web at <http://www.dialogic.com>.

Dialogic Corporation

9800 Cavendish Blvd., 5th floor
Montreal, Quebec
CANADA H4M 2V9

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH PRODUCTS OF DIALOGIC CORPORATION OR ITS SUBSIDIARIES ("DIALOGIC"). NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Dialogic may make changes to specifications, product descriptions, and plans at any time, without notice.

Dialogic is a registered trademark of Dialogic Corporation. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic's legal department at 9800 Cavendish Blvd., 5th Floor, Montreal, Quebec, Canada H4M 2V9. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic's trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners. Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement their concepts or applications, which licenses may vary from country to country.

Copyright © 2007 Dialogic Corporation All rights reserved.

07/07 10532-01