Short time spectrum analysis-synthesis of speech

Dusan Vujcic
SHORT TIME SPECTRUM
ANALYSIS–SYNTHESIS OF SPEECH

by
Dusan D. Vujcic

A Thesis Submitted
in
Partial Fulfilment
of the
Requirements for the Degree of

MASTER OF SCIENCE
in
Electrical Engineering

Approved by:

Prof. Robert Houde
(Thesis Advisor)

Prof. Kenneth Hsu

Prof. A. F. Reithmeier

Prof. Harvey Rhody

Prof. S. Madhu
(Department Head)

Department Of Electrical Engineering

COLLEGE OF ENGINEERING

Rochester Institute Of Technology
Rochester, New York
May, 1985
Reproduction of this material for personal use is permitted by the author of the thesis. Mass reproduction and distribution is not permitted without the author's approval.
ABSTRACT

This thesis presents an extensive study of the short time spectrum analysis-synthesis of speech. This type of speech processing attempts to synthesize speech without the short time Fourier transform phase information. Several existing analysis-synthesis methods are presented. In addition, five other methods are attempted experimentally. The results of these experiments are explained, identifying some very important properties of the short time spectrum analysis-synthesis. One of the methods was shown to synthesize a high fidelity speech. Finally, a comparison is made between this method and other known methods which are based on short time Fourier transform approach.
TABLE OF CONTENTS

1. INTRODUCTION

2. SHORT TIME FOURIER TRANSFORM ANALYSIS-SYNTHESIS (STFTAS)
   2.1 Terminal-Analog Model of Vocal System
   2.2 Definition of STFTAS
   2.3 Selection of STFTAS Parameters
   2.4 Evaluation of STFTAS

3. INTRODUCTION TO SHORT TIME SPECTRUM ANALYSIS-SYNTHESIS OF SPEECH (STSASS)
   3.1 Introduction
   3.2 Definition of STSASS
   3.3 STSASS vs. STFTAS
   3.4 Previous Research of STSASS Method

4. RESEARCH OF SHORT TIME SPECTRUM ANALYSIS-SYNTHESIS METHODS
   4.1 Method 1 - Synthesis From Real Power Spectrum
   4.2 Method 2 - Synthesis From Imaginary Power Spectrum
   4.3 Method 3 - Synthesis From Quantized Phase Spectrum
   4.4 Method 4 - Synthesis by Phase Prediction
   4.5 Method 5 - Synthesis From Modified Power Spectrum

5. EVALUATION OF STSASS METHODS
   5.1 Method Comparison
   5.2 Affect of Analysis-Synthesis Parameters on Synthesis Quality
   5.3 Practical Considerations

6. CONCLUSION

7. APPENDIX
   A.1 Speech Analysis-Synthesis System (SASS) Hardware Overview
   A.2 Speech Analysis-Synthesis System (SASS) Software Overview
   A.3 Computer Listing of STSASS Algorithm

8. REFERENCES
ACKNOWLEDGEMENTS

I wish to thank Dr. Robert Houde for his valuable suggestions and continuous encouragements. Without Him this Thesis would not be possible.

I wish to thank Dr. Charles Parkins of Strong Memorial Hospital, Rochester, N.Y., for confirming my experimental results.

I wish to thank my wife Svetlana for her patience.
1. INTRODUCTION

Short time Fourier transform analysis-synthesis (STFTAS) is a method of speech processing based on fundamental concepts of describing speech. Simulating the operation of the human brain, this method offers a high degree of accuracy when synthesizing speech.

Gaining the popularity in the early 1970's, STFTAS has been a focus of several leading speech processing experts. This sudden interest in STFTAS is attributed to the technological developments in the microelectronics field. The inventions of microprocessors, fast-Fourier-transform (FFT) integrated circuits, floating-point co-processors, and many other similar devices, have made it possible to execute millions of floating point arithmetic and trigonometric operations needed to complete STFTAS.

The short time spectrum analysis-synthesis of speech (STSASS) is a special branch of STFTAS. The main attraction of this method is the synthesis of speech from its frequency power spectrum without the phase information. There are several benefits of this method compared to STFTAS. These include information (bit) reduction, less computation, and an operation with an all positive, all real frequency domain, which is a much simpler medium for analysis of speech than the complex frequency domain.

Being at its infant stages of development, STSASS has been researched very little. This paper will present several methods of STSASS implementation, show experimental results, and back them with theory.

We now explain the organization of the following material. Section 2 proceeds to explain the general method of STFTAS, and the theory behind it. Selection of analysis-synthesis parameters is shown, and an overall evaluation of the method presented. Following is section 3 with the historical information on STSASS research attempted by others. Section 4 presents five methods of STSASS suggested by this author. Varying in success and quality, each method reveals a valuable piece of information regarding STSASS. In section 5 we find the discussion of STSASS, as results from section 4 are combined for an overall evaluation. A most successful analysis-synthesis method is chosen and the selection of parameters presented. Section 6 makes the concluding remarks.
2. SHORT TIME FOURIER TRANSFORM ANALYSIS-SYNTHESIS

2.1 Terminal-Analog Model of Vocal System

In order to fully understand the subject of short time Fourier transform analysis-synthesis (STFTAS) of speech, we must first present and explain the model of the vocal system, as developed by Shafer and Rabiner (ref. 8).

Figure 2.1. A shows the "terminal-analog" model of the vocal system. Two mutually exclusive sources of sound are assumed to generate the speech. These are the voiced and the unvoiced speech signals.

The voiced signal, \( v(n) \), is generated by the unit sample train generator, whose samples are separated by the speech pitch period. The pitch is the fundamental period of the speech signal. Typical pitch periods are in the 5 to 10 millisecond range. The unvoiced portion, \( u(n) \), is the output of the white noise generator.

The linear time varying filter, \( t(n,m) \), receives either \( v(n) \) or \( u(n) \), and coupled with the vocal tract parameters, generates the speech output, \( x(n) \). Time varying filter \( t(n,m) \) is a function of variable \( n \), quantized time, and variable \( m \), quantized time describing time previous to \( n \). The filter’s transfer function defines the response of the unit sample applied to its input \( n-m \) samples earlier.

The system \( t(n,m) \) varies in time due to the changes in the vocal tract. However, the model may be simplified by realizing that the vocal tract is a lossy resonator, whose impulse response has a very sharp decay. Slow changes in the vocal tract with respect to the input and output waveforms permit us to call \( t(n,m) \) "quasi-stationary". \( t(n,m) \) may also be viewed as a time varying system whose "memory" is too short to detect any system changes. Therefore, \( t(n,m) \) can be approximated as \( t(m) \).

Finally, we express the output \( x(n) \) in terms of the inputs of the time varying filter. Concerned only with the voiced speech, we get

\[
x(n) = t(n,m) \ast v(n)
\]

(2.1.1)
FIGURE 2.1.A - Terminal-Analog Model of Vocal System

PITCH PERIOD

UNIT SAMPLE TRAIN GENERATOR

V(n)

VOCAL TRACT PARAMETERS

LINEAR TIME-VARYING FILTER

t(n,m)

SPEECH SAMPLES

STATIONARY WHITE NOISE GENERATOR

u(n)
Using the definition of convolution, it can be shown (ref. 4) that

\[ x(n) = \sum_{m=+\infty}^{m=-\infty} t(n,m)v(n-m) \]  

(2.1.2)

Even though the summation is over all values of \( m \), we recall that \( t(n,m) \) has a short "memory", and thus only the most recent values of \( m \) will contribute to \( x(n) \).

2.2 Definition of Short Time Fourier Transform Analysis-Synthesis

Short time Fourier transform analysis-synthesis (STFTAS) is a process which attempts to generate an intermediate representation of a speech signal.

STFTAS is performed on a speech signal \( x(n) \), which is assumed to be defined for all \( n \). Ideally, STFTAS ultimate goal would be to "freeze" the speech signal for any value of \( n \) and express the frequency content of speech at that particular instant. Then, the variable \( n \) would be increased by an infinitesimal amount, and the frequency content found again. Such a frequency representation of speech is of an extreme value, as it allows the speech signal to be modified and/or enhanced. Manipulations of speech in time domain have always been far more difficult to achieve.

The frequency representation of speech explained above is not possible. STFTAS method attempts to represent speech based on the above approach, with some necessary modifications.

Step1 - Instead of a single value of \( n \), we must define a speech window, or collection of \( n \)'s for which we are able to obtain the frequency representation of speech. The analysis-synthesis process begins by selecting a small portion of the speech signal \( x(n) \) to be analyzed. This selection is achieved by finding \( x_m(n) \) such that

\[ x_m(n) = x(n)w_m(n) \]  

(2.2.1)

where \( w_m(n) \) is the filtering function which isolates \( x_m(n) \) from \( x(n) \).
In general, \( w_m(n) \) is in the form

\[
\begin{align*}
    w_m(n) &> 0 \quad \text{for } m-1 < n < m+N \quad (2.2.2) \\
    &= 0 \quad \text{otherwise}
\end{align*}
\]

where \( m \) is a constant, and \( N \) is the number of speech samples. It will be shown that it is necessary that the frequency spectrum of \( w_m(n) \) resemble that of a low pass filter response, in order to avoid aliasing problems. \( x_m(n) \) will be referred to as the short time analysis section.

Step 2 - Once \( x_m(n) \) is found, a complex Fourier transform \( X_m(k) \) is calculated by

\[
X_m(k) = \sum_{n=m}^{m+N-1} x_m(n)e^{-j2\pi k(n-m)/N} \quad (2.2.3)
\]

Step 3 - \( X_m(k) \) may now be modified to \( X'_m(k) \). Since the methods of frequency domain modification are beyond the scope of this project, we will assume that \( X_m(k) \) is not modified.

Step 4 - We move back to the time domain by taking the inverse Fourier transform generated in Step 3. The resulting waveform is called the short time synthesis section, \( y_m(n) \), where

\[
y_m(n) = \sum_{k=0}^{N-1} X'_m(k)e^{-j2\pi k(n-m)/N} \quad (2.2.4)
\]

This completes the first phase of the short time Fourier transform analysis-synthesis. The synthesis section \( y_m(n) \) is the synthesized waveshape from \( X'_m(k) \), the modified frequency domain of the analysis section, \( x_m(n) \). Waveform \( y_m(n) \) must be saved as we proceed to step 5.
Step 5 STFTAS continues by sliding the analysis filter $W_m(n)$ in time by $s$ samples. Range of $s$ is

$$0 < s < N+1$$

(2.2.5)

where $N$ is the number of samples in $W_m(n)$. We now repeat steps step 1 through step 4, generating a new synthesis section $y_m(n)$ every time. Assuming that the very first $y_m(n)$ generated is defined as $y_m(n)$, the next one as $y_{m+1}(n)$, we can express the synthesized speech as the sum of synthesis sections

$$y(n) = \sum_{m=-\infty}^{\infty} y_m(n)$$

(2.2.6)

Therefore, if $s < N$, we explain $y(n)$ as an overlapped-and-added sum of the short time synthesis sections obtained from the modified Fourier transform of the short time analysis sections $X_m(n)$.

The above procedure is a general description of STFTAS method. As an algorithm, the above procedure is inefficient, as we must save all of the synthesis sections, $y_m(n)$, before finding $y(n)$.

A new method is presented which is more efficient and requires less storage. The main idea is to define the original speech $x(n)$ as

$$x(n) = \sum_{a} x_a(n)$$

(2.2.7)

where $x_a(n)$ are non-overlapping analysis windows of $x(n)$, and $a$ is a constant, marking the start of the analysis window. We define $x_a(n)$ as

$$x_a(n) = x(n) \quad \text{for } a-1 < n < a+N$$

$$= 0 \quad \text{elsewhere}$$

(2.2.8)

The synthesized speech $y(n)$ is defined in the similar manner

$$y(n) = \sum_{a} y_a(n)$$

(2.2.9)
where \( y\neq(n) \) are non-verlapping synthesis windows of \( y(n) \). We define \( y\neq(n) \) as

\[
y\neq(n) = y(n) \quad \text{for } a-1 < n < a+N \quad (2.2.10)
\]

\[
y\neq(n) = 0 \quad \text{elsewhere}
\]

The method of synthesizing \( y(n) \) from \( x(n) \) by synthesizing one window at a time is now presented. By definition of the overlap-and-add method, \( y\neq(n) \) is made up of short time synthesis sections \( y_m(n) \), or

\[
y\neq(n) = \sum_{m} y_m(n) \quad (2.2.11)
\]

\[
m = a-N+s, a-N+2s, ... , 0, a+s, a+2s, a+N-2s, a+N-s
\]

where \( s \) is the separation between windows. It can be shown that the total number of sections making up \( y\neq(n) \) is

\[
W = 2(N/s) - 1 \quad (2.2.12)
\]

Therefore, the alternate procedure for synthesizing \( y(n) \) from \( x(n) \) is:

Step1-Choose \( x_0(n) \) as the starting analysis window by letting \( a=0 \), and \( m=a-N+s \).

Step2-Find analysis section \( x_m(n) \) by multiplying \( x(n) \) by \( w_m(n) \).

Step3-Find \( X_m(k) \) and modify to \( X'_m(k) \).

Step4-Find synthesis section \( y_m(n) \) from \( X'_m(k) \).

Step5-Increase \( m \) by \( s \) and repeat Step2 through Step4. When \( m=a+N \), go to Step6.

Step6-Add \( y_m(n) \) as in equation 2.2.11 to get \( y\neq(n) \).

Step7-Increment \( a \) by \( N \). Keep only \( y_m(n) \) for \( m > a-N \). Go to Step2.
FIGURE 2.2. Graphical Interpretation of STFTAS Algorithm

\[ X(n) \]

\[ X_A(n) \quad X_{A0}(n) \quad X_{A1}(n) \quad X_{A2}(n) \quad X_{A3}(n) \quad X_{A4}(n) \quad X_{A5}(n) \]

\[ X_m(n) \quad X_{m0}(n) \quad X_{m1}(n) \quad X_{m2}(n) \quad X_{m3}(n) \quad X_{m4}(n) \quad X_{m5}(n) \quad X_{m6}(n) \]

\[ X_m^2(k) \quad X_{m0}(k) \quad X_{m1}(k) \quad X_{m2}(k) \quad X_{m3}(k) \quad X_{m4}(k) \quad X_{m5}(k) \quad X_{m6}(k) \]

\[ Y_m(n) \quad Y_{m0}(n) \quad Y_{m1}(n) \quad Y_{m2}(n) \quad Y_{m3}(n) \quad Y_{m4}(n) \quad Y_{m5}(n) \quad Y_{m6}(n) \]

\[ Y_A(n) \quad Y_{A0}(n) \quad Y_{A1}(n) \quad Y_{A2}(n) \quad Y_{A3}(n) \quad Y_{A4}(n) \quad Y_{A5}(n) \]

\[ y(n) \]
Figure 2.2. A illustrates the above procedure graphically. Appendix 3.A shows the computer listing of the above algorithm written in the 68B09 micoprocessor assembler language.

2.3 Selection of Short Time Fourier Transform Analysis-Synthesis Parameters

This section will present a design of STFTAS parameters such as sampling period $T$, number of samples per analysis window $N$, separation between analysis windows $s$, and filtering window $w_m(n)$.

2.3.1 Sampling period $T$

Bandlimited by $B$ hertz, human speech must be sampled at the rate of

$$\frac{1}{T} = F > 2B \quad (2.3.1.1)$$

to avoid aliasing effects. If $F$ is roughly equal to $2B$, the aliasing effects may only be removed with an ideal low pass filter. If $F \gg 2B$ a low quality low pass filter would be required, however, the the number of samples/second would increase, thus increasing the processing time. Given the above considerations, it is recommended that

$$T = \frac{1}{3B} \quad (2.3.1.2)$$

2.3.2 Filtering window $w_m(n)$

The selection of $w_m(n)$ is not trivial. Several important requirements must be met. In order to be selected, $w_m(n)$ must be such that the synthesized waveshape, $y(n)$, equals the original waveshape, $x(n)$, providing that the frequency content of the analysis section ($X_m(k)$) was not modified. From equation 2.2.1, we can express $X_m(k)$ as

$$X_m(k) = \sum w_m(n)x(n)e^{-j2\pi k(n-m)/N} \quad (2.3.2.1)$$

Note again that above equation follows from $x(n)$ being multiplied by a sliding window $w_m(n)$ every $s$ samples. By taking the inverse Fourier
Transform of $X_m(k)$, we get

\[ y_m(n) = \frac{1}{N} \sum_{k=0}^{N-1} (\sum w(n) x(n) e^{j2\pi k(n-m)/N}) e^{-j2\pi k(n-m)/N} \]  

(2.3.2.2)

It can be shown (ref. 1) that the above equation simplifies to

\[ y_m(n) = x(n) w_m(n) \]  

(2.3.2.3)

By overlap-and-add method, we generate the synthesized waveshape

\[ y(n) = \sum_{m} y_m(n) = x(n) \sum_{m} w_m(n) \]  

(2.3.2.4)

Finally, we must find such $w_m(n)$ so that

\[ \sum_{m} w_m(n) = 1 \]  

(2.3.2.5)

Allen (ref. 3) has shown using the Poisson's summation formula that the above equation holds true for any function $w_m(n)$ which is bandlimited to $1/2T$, and summed over all $n$, $w_m(n) = 1$. From equations 2.3.2.5 and 2.3.2.6, it follows that

\[ y(n) = x(n) \]  

(2.3.2.6)

The above equation says that the synthesized waveshape, $y(n)$, is identical to the original waveshape, $x(n)$, providing a proper filtering analysis window $w_m(n)$ has been chosen.
The cutoff frequency of \( w_m(n) \) must now be determined. Recalling that multiplication of \( x(n) \) by \( w_m(n) \) in the time domain results in the circular convolution in the frequency domain, the resulting frequency spectrum may be thought of as a sum of filter banks \( B \) wide, where \( B \) is the bandwidth of \( w_m(n) \). By letting

\[
Bw < B_p \quad (2.3.2.7)
\]

where \( B_p \) is the period of speech pitch, the filter banks will be separated and aliasing problems avoided. This separation permits frequency spectrum modifications and eventual synthesis. Figures 2.3.2.A through 2.3.2.C show how the spectrum of the analysis section \( x_m(n) \) is modified by \( w_m(n) \), and what occurs if the equation 2.3.2.8 is not satisfied.

A commonly used filtering window is the Hamming window

\[
w_m(n) = 0.54 + 0.46 \cos(2\pi T(n-m)/N) \quad \text{for} \quad m-1 < n < m+N \quad (2.3.2.8)
\]

\[
w_m(n) = 0 \quad \text{otherwise}
\]

where \( N \) is the number of samples of \( w_m(n) \). Figures 2.3.2.D and 2.3.2.E show the time and frequency domain representation of the Hamming window.

Other filtering functions may, of course, be used. It is interesting to mention that an obvious window function such as \( w_m(n) = u(n-m) - u(n-m+N-1) \) has a frequency spectrum similar to that of the Hamming window, as shown in Figure 2.3.2.E. ( \( U(n) \) is the unit step function. \( U(n) = 0 \) for \( n < 0 \), and \( U(n) = 1 \) elsewhere.) The major exception is that the first side lobe is only 12db down from the main lobe, thus making it a poor low-pass filter.

Choosing the filtering function may be achieved by analyzing the response of the human cochlea, an organ which connects the outer ear with the brain. Some early research performed at Rochester Institute of Technology has shown that the cochlea is made up of roughly 300 baseband filters which are in the form

\[
w_m(t) = \exp(-a(t-t)) \exp(j\omega t) \quad \text{for} \quad t < t_0 \quad (2.3.2.9)
\]
FIGURE 2.3.2.A - Frequency Content of Speech Waveform

FIGURE 2.3.2.B - Frequency Spectrum After Proper Filtering Function Applied

FIGURE 2.3.2.C - Frequency Spectrum After Improper Filtering Function Applied
FIGURE 2.3.2.D - Hamming Window - Time Domain

FIGURE 2.3.2.E - Hamming Window - Frequency Domain
where \( t \) is time, \( t_0 \) is the current time, \( a \) is a constant, and \( \omega_c \) is a center radian frequency of the filter. As a result, it was suggested that the filtering window may be in the form

\[
W_m(n) = e^{-qT(N-1-n+m)} \quad \text{for} \quad m-1 < n < m+N \quad (2.3.2.10)
\]

\[
= 0 \quad \text{otherwise}
\]

where \( q \) is a constant, \( T \) is the sampling period, and \( N \) is the number of samples in \( W_m(n) \). The frequency response of this function resembles that of the low filter.

### 2.3.3 Number of Samples Per Analysis Section, \( N \)

Choosing \( N \) requires two considerations. First is the relationship between \( N \) and the particular \( W_m(n) \) used. The second consideration is related to the speech pitch. In this explanation we choose to utilize the Hamming window.

From Figures 2.3.2.D and 2.3.2.E we can show that the cutoff frequency for the Hamming window occurs at

\[
B_h = \frac{2}{NT} \quad (2.3.3.1)
\]

\( N \) must be chosen so that

\[
\frac{4}{NT} < \text{pitch frequency} \quad (2.3.3.2)
\]

thus allowing the pitch harmonics to be resolved. Relating the equation 2.3.3.2 to the pitch period of speech, \( P(n) \), we conclude that

\[
\frac{1}{P(n)} > \frac{4}{NT}
\]

or simply

\[
NT > 4P(n) \quad (2.3.3.3)
\]

Therefore, the length of the Hamming window, and thus the analysis window must be at least 4 pitch periods long.
2.3.4 Separation Between Analysis Sections, s

This parameter depends on the cutoff frequency of the filtering function. In general, if \( w_m(n) \) is bandlimited to frequency \( F \), then, using the Nyquist rate as criteria, we conclude that \( w(n) \) must be applied at the rate \( 2F \). For the case of the Hamming window we use the equation 2.3.3.1 to derive

\[
s = \frac{NT}{4}
\]  

(2.3.4.1)

Combining sections 2.3.1 through 2.3.4, we can show that for a speech signal whose pitch is roughly 200 hertz, and whose bandwidth is about 3300 hertz, we should sample the speech at a rate of 10 khertz, and analyze the speech with the Hamming window at least 20 milliseconds long. The separation between the analysis sections should not be greater than 5 milliseconds. Figures 2.3.4.A - 2.3.4.D show an example of a Hamming window being applied to the original waveshape (.A), the resulting time domain waveshape (.B) and its frequency spectrum (.C), and the overlapped-and-added windows being combined to generate the synthesized waveshape (.D).

2.4 Evaluation Of Short Time Fourier Transform Analysis-Synthesis

STFTAS method of speech analysis, spectral modification and synthesis is a very attractive one since its representation of speech in short time captures all of the speech attributes that can be modified.

This type of signal processing is very useful in a field of the speech enhancement, where the noise may be removed from speech by modifying the short time Fourier transform. Other applications (ref. 9,10,11) also use STFTAS as the means of speech processing. The principles of STFTAS are also applied in video signal processing.

SFTAS method of speech processing has been tested on several occasions. In cases when speech was not modified, the synthesis reproduced the original speech with negligible errors (ref 3). In cases when speech was modified, synthesized speech was also very intelligible.
FIGURE 2.3.4.A Hamming Window Applied on Some Signal

FIGURE 2.3.4.B Result of Multiplication In A

FIGURE 2.3.4.C Frequency Spectrum of Waveshape In B

FIGURE 2.3.4.D Example of Overlap-And-Add Method Using Hamming Window
The most attractive aspect of STFTAS is its ability to generate the medium in which speech can be modified and then reconstructed. The very straightforward procedure is also important. Such processes as pitch tracking, and differentiation between the voiced and the unvoiced speech is not necessary as it is the case with a channel vocoder.

The main weakness of this method is a relatively high degree of mathematical complexity. For each loop that generates a speech section to be overlapped-and-added, one must execute 1 multiplication/sample in the time domain, execute an N-point complex Fourier transform, execute the frequency modification algorithm (if any), perform an inverse N-point Fourier transform, and finally add the resulting waveshape to the overlap-and-add running sum. With a separation between analysis sections equal to s, one must process N/s times more data relative to the end result.

The technological advances made in the fields of VLSI integrated circuits have made STFTAS more attractive. Fast Fourier transform (FFT) algorithms can now be executed by an integrated circuit (IC) in a matter of milliseconds. For example the state-of-the-art IC from Texas Instruments completes a 128-point FFT in only 1 millisecond.

The optimum goal of any signal processing method is to be realizable and operational in real time, as well as cost justifiable. Technological advancements have placed the short time Fourier transform at a very threshold of these requirements. The future should bring an even greater popularity of the STFTAS method.
3. INTRODUCTION TO SHORT TIME SPECTRUM ANALYSIS-SYNTHESIS OF SPEECH

3.1 Introduction

A special case of short time Fourier transform analysis-synthesis (STFTAS) is the short time spectrum analysis-synthesis of speech (STSASS). The objective of this method is to represent the speech with its frequency power spectrum only. This would allow frequency domain modifications which are insensitive to the speech phase information. Finally, the speech must be reconstructable from its short time frequency spectrum domain.

Synthesis from the frequency power spectrum is a subject rarely explored due to the predisposition that the correct phase information must be present for speech to be intelligible after synthesis. Recently (1981-1982), Massachusetts Institute of Technology (M.I.T.) exerted a major effort in analyzing the speech reconstruction from the short time Fourier transform magnitude. M.I.T. research is the only known major study of STSASS attempted thus far.

This section’s purpose is to familiarize the reader with the basic concepts of STSASS, explain its advantages over the STFTAS method, and present and evaluate the M.I.T. study.

3.2 Definition of Short Time Spectrum Analysis-Synthesis of Speech

Short time Fourier spectrum analysis-synthesis of speech (STSASS) varies in only one aspect from the short time Fourier transform analysis-synthesis (STFTAS). Recalling from section 2, the short time Fourier transform of the analysis section was described as

\[ X_m(k) = c(k)e^{j\theta(k)} \]  

(3.2.1)

where \( c(k) \) is the magnitude of \( X_m(k) \) at quantized frequency \( k \). In STSASS method we modify \( X_m(k) \) to \( |X_m(k)| \), which is equivalent to equation 3.2.1 with \( \theta(k)=0 \). Therefore,
\[ X_m(k) = c(k) \quad (3.2.2) \]

Thus \( X_m(k) \) is an all real, all positive frequency domain containing the magnitude of each frequency component \( k \).

3.3 STSASS Vs. STFTAS

There are numerous advantages of STSASS over STFTAS:

A. The amount of information is cut by over 50\% compared to STFTAS. The savings come from the property that the imaginary part of \( X_m(k) \) is zero. An additional information cut is a result of \( X_m(k) \) being always positive, thus not requiring a sign bit. For example, for \( X_m(k) \) having 12 bits of precision, the information rate can be cut from 24 bits (12 for real, 12 for imaginary) to 11 bits, with a total reduction of 54\% In speech communication systems the amount of information is important from many aspects, such as speed, storage, etc. In general, the reduced speech information simplifies the system complexity, reducing the cost.

B. The amount of mathematical manipulation is reduced using the STSASS method. The reduction comes when calculating the synthesized section \( y_m(n) \), from the power spectrum of the analysis section, \( X_m(k) \). When the inverse Fourier transform of \( X_m(k) \) is executed, several short cuts may be taken since \( X_m(k) \) is an all real, all positive frequency domain. Using the definition of the inverse Fourier transform

\[
y_m(n) = \frac{1}{N} \sum_{k=0}^{N-1} X^*_m(k) e^{-j2\pi kn/N} \quad (3.3.1)
\]

where \( X^*_m(k) \) is the complex conjugate of \( X_m(k) \). Another form of the above equation is
\[ y^m(n) = \frac{1}{N} \sum_{k=0}^{N-1} (c_r(k) - j c_i(k)) \left( \cos\left(\frac{2\pi kn}{N}\right) + j \sin\left(\frac{2\pi kn}{N}\right) \right) \quad (3.3.2) \]

where \( c_r(k) \) is the real part of \( X^m(k) \), and \( c_i(k) \) is the imaginary part of \( X^m(k) \) at frequency \( k \). Note that the complex multiplication for each value of \( k \) requires 4 real number multiplications and two additions.

In the case of the STSASS, we showed that \( c_i(k) = 0 \) for all \( k \). This simplifies the above equation to

\[ y^m(n) = \frac{1}{N} \sum_{k=0}^{N-1} c_r(k) \left( \cos\left(\frac{2\pi kn}{N}\right) + j \sin\left(\frac{2\pi kn}{N}\right) \right) \quad (3.3.3) \]

Furthermore, from the properties of the Fourier transform it can be shown that \( y^m(n) \) is also an all real function. This allows us to simplify even further to

\[ y^m(n) = \frac{1}{N} \sum_{k=0}^{N-1} c_r(k) \cos\left(\frac{2\pi kn}{N}\right) \quad (3.3.4) \]

Finally, we can also show that \( y^m(n) \) is an even function, so that \( y^m(0) = y^m(N-1) \), \( y^m(1) = y^m(N-2) \) etc. Therefore

\[ y^m(n) = y^m(N-n) = \frac{1}{N} \sum_{k=0}^{N/2-1} c_r(k) \cos\left(\frac{2\pi kn}{N}\right) \quad (3.3.5) \]

Comparing to equation 3.3.2, we only need to perform a single multiplication for every value of \( k \). Assuming that the calculations are performed with a digital computer, we (conservatively) estimate the savings in calculation time of about 85%.

C. Another advantage of STSASS is the very absence of phase information. This is important from two aspects: psychological and practical. The psychological aspect comes from the very nature of phase - in general, we do not perceive the phase with the same ease as we do the magnitude. Phase is somewhat of a low-information parameter, as we can
learn very little about a waveform. On the other hand, knowledge of the frequency power spectrum tells us immediately about the frequency content, and the energy stored in the signal. The practical aspect comes from an ability to analyze the signals whose phase is unknown. For example, let us assume we know that the speech signal is corrupted by white noise whose phase is not known, with a known signal-to-noise ratio. We could attempt to rid the speech of the noise by implementing the STSASS method and performing a "spectral subtraction" on all $X_m(k)$'s, for all values of $k$. The "spectral subtraction" would simply subtract the spectral magnitude of noise from each $X_m(k)$. Thus the speech could be enhanced, without any knowledge of the phase of white noise.

3.4 Previous Research of STSASS Methods

The only known research of STSASS took place at Massachusetts Institute of Technology (M.I.T.). This section describes the results of the research (ref.7).

The object of the M.I.T. study was to reconstruct the original waveshape from its short time Fourier transform magnitude $|X_m(k)|$, as defined in Eq.

\[ |X_m(k)| = \left| \sum_{n=0}^{N-1} X_m(n)e^{-j2\pi k n/N} \right| \]

where $X_m(n)$ is the short time analysis section $N$ samples long.

The synthesis method is based on two theorems:

Theorem 1 - Let $z(n)$ be a sequence that is zero outside $-1 < n < N+1$. Suppose $z(0)$ is non-zero. Define $|Z(k)|$ as a frequency domain magnitude of $z(n)$ for $-1 < n < N$. Then $|Z(k)|$ and the sample $z(0)$ uniquely specify sample $z(N)$. 

22
The proof lies in showing that the autocorrelation function of \( z(n) \) relates to \( x(0) \) and \( x(N) \) as

\[
R(n) = z(0)z(N) \quad (3.4.2)
\]

where \( R(n) \) is the autocorrelation function

\[
R(n) = z(n)^*z(-n) \quad (3.4.3)
\]

An alternate way to calculate \( R(n) \) is by

\[
R(n) = F^{-1}(|Z(k)|^2) \quad (3.4.4)
\]

where \( R(n) \) must be obtained with a \( 2N+1 \) point inverse Fourier transform to avoid aliasing. From equation 3.4.2

\[
z(N) = R(n)/z(0) \quad (3.4.5)
\]

Theorem 2 - Let \( z(n) \) be a sequence that is zero outside -1 < \( n < N+1 \). Suppose \( z(0) \) is non-zero. Define \( |Z(k)| \) as a frequency domain magnitude of \( z(n) \) for -1 < \( n < N \). Then \( |Z(k)| \) and the \( p \) samples of \( z(n) \) for -1 < \( n < p+1 \) uniquely specify the complete sequence \( z(n) \) if and only if \( p > (N+1)/2 + 1 \).

The proof again makes the use of the autocorrelation function in equation 3.4.4. Expanding, we get

\[
\begin{bmatrix}
1 & z(0) & | & 1 & 1 & z(N) & | & 1 & R(N) & 1 \\
1 & z(1) & z(0) & | & 1 & 1 & z(N-1) & | & 1 & R(N-1) & 1 \\
1 & z(2) & z(1) & z(0) & | & 1 & 1 & z(N-2) & | & 1 & R(N-2) & 1 \\
1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & z(N/2) & z(N/2-1) & \ldots z(0) & | & 1 & 1 & z(N/2) & | & 1 & R(N/2) & 1
\end{bmatrix}
\]

Solving the matrix equation for unknown \( z(n) \) samples now becomes trivial.
The study offers three methods of synthesizing $y(n)$ from the short time Fourier transform magnitude of the original signal $x(n)$. The following is the brief description of those methods. The experimental results, carried out on PDP11/50 computer with 64 bit resolution, are also presented.

Method 1 - Sequential Extrapolation

This method is very closely based on Theorems 1 and 2. The attempt is to reconstruct the signal with the minimum amount of information, given $s$ known samples, where $s$ is the overlap between the short time analysis sections.

Step 1 - Initialization: $x(n)$ is known for $-1 < n < s$. Set $m=2$ and $lX_1(k)$.

Step 2 - Extrapolation: Use known samples of $x(n)$ along with the autocorrelation function (derived from $lX_m(k)$) to find unknown samples in region $(m-1)L-1 < n < ms$.

Step 3 - Increment $k$

Step 4 - Repeat Step 1 through Step 3 while $x(n)$ exists.

This method was tested using the sampling rate of $10\,\text{kHz}$, quantization of 12 bits/sample, and 128 sample short time analysis sections. The speech was synthesized exactly when separation was set to $s=1$. When $s$ was increased, the synthesized waveform sharply decreased in quality. At $s=4$, the waveform was unintelligible. The error was blamed on the additive error. An error generated in calculating the unknown samples of $x(n)$ will propagate to the next set of calculations, thus decreasing the quality of the synthesized waveshape. From this we may conclude that the known samples must be as accurate as possible, since the measurement error will propagate through each successive calculation.
Method 2 - Least Squares Sequential Extrapolation

As an attempt to rid the dependancy of the overlap s, this method uses more information than Method 1. The synthesis of the unknown samples is based on minimizing the error function E, where

\[
E = \sum_{m=-\infty}^{\infty} (R(m) - U(m))^2 \quad (3.4.6)
\]

where \( R(m) \) is the autocorrelation function derived from \(|X_m(k)|\), and \( U(m) \) represents the autocorrelation function of the synthesized section. The goal is to minimize \( E \) so that the synthesized section closely resembles the original. To minimize \( E \), we must set its derivatives with respect to the unknown samples to zero. For \( s \) unknowns, this will yield \( s \) simultaneous cubic equations with \( s \) unknowns. If only 1 sample is unknown, then it was shown that that sample is equal to

\[
y_m(N-1) = \frac{\sum_{m=1}^{N-1} (R(m) - T(m))y_m(N-1-m)}{\sum_{m=1}^{N-1} y_m^2(N-1-m)} \quad (3.4.7)
\]

where \( T(m) \) is the autocorrelation of the sequence obtained from \( y_m(n) \) by setting \( y_m(N-1) \) equal to zero.

The experimental results showed that the dependancy on the overlap \( s \) decreased. This time synthesis of waveshapes of up to \( s=30 \) were synthesized with speech quality. It must be pointed out that this improvement heavily increased the calculation time. At \( s=40 \), the quality sharply decreased and the synthesized speech was non-intelligible.
Method 3 - Iterative Extrapolation

This method decreases the dependence of synthesized speech quality from overlap $s$ by improving the extrapolation step in Method 2. Similar to Method 2 approach, more information than needed is used to derive the unknown samples of $x(n)$. The technique alternates between the time and frequency domain, imposing the known constraints. The goal is to have the technique converge to the unknown samples. The lengthy procedure does not permit more discussion of this method. More importantly, Method 3 was said not to converge for all $x(n)$, without specifying those waveshapes for which it converges. In some cases, this method was capable of synthesizing speech with $s=40$. The speech quality was retained, although the synthesized waveshape noticeably varied from the original. In other cases, the synthesized waveshape did not converge to the unknown samples, making the synthesized waveshape unintelligible.

We now summarize the above methods. The above discussion has shown that the synthesis from the short time Fourier transform magnitude is possible, providing that the initial $s$ samples are known, where $s$ is the overlap between the short time sections. All methods used the properties relating the autocorrelation function of the short time section to its known and unknown samples, as shown in Theorems 1 and 2. Three methods of synthesis were presented. Method 1 showed an extreme sensitivity to the overlap $s$, while the Method 3 attempted to remove that dependency by using more information than required. Although the improvement was significant, the method did not always converge to the unknown samples. Method 2 always synthesized an intelligible waveshape. The improvement is overshadowed by increased calculation requirements. The main drawback of all three methods is the requirement that of some samples of the original waveshape be known.
4. RESEARCH OF SHORT TIME SPECTRUM ANALYSIS-SYNTHESIS METHODS

This section investigates several short time spectrum analysis-synthesis (STSASS) methods. The approach taken is very fundamental - establish a STSASS method which will be shown to be comparable or simpler in complexity from the existing short time Fourier transform methods utilizing phase. An additional objective will be to synthesize speech exactly, that is, the listener should not be able to differentiate between the synthesized and the original speech waveshape.

Several STSASS methods are explored. Explanation of each method includes its definition, presentation of experimental results, and their analysis.

It must be pointed out that the goal of this paper is to synthesize the speech by discarding the phase information. This does not imply that the synthesis method would necessarily use the magnitude of the short time Fourier transform.

All of the experimental work was performed by the author on the Speech Analysis-Synthesis System (SASS), a multi-board microcomputer described briefly in Appendix A.1 (hardware design) and A.2 (software design).

Several experiments were attempted for each method, each time making a slight modification by varying one of the analysis-synthesis parameters. Each waveshape generated by an experiment is shown in the time and frequency domain, along with all of the parameters used in the experiment. Note that every experiment made an attempt to synthesize the speech from an original waveshape (Figure 4.B) which did not change for any of the experiments. This permits not only the comparison of the synthesized waveshape with the original waveshape, but the comparison of various synthesized waveshapes as well.
FIGURE 4.B - Speech Waveform Under Analysis-Synthesis, Methods 1,2,3,5

Number of samples per analysis-synthesis section .................. N=128
Sampling period in microseconds ........................................ T=100
Horizontal resolution in microseconds/grid ......................... X_t=700
Horizontal resolution in hertz per grid .............................. X_f=136
Start of print in multiples of T/2 .................................... t_0=4096
4.1 Method 1 - Synthesis From Real Frequency Power Spectrum

We have already shown that the short time Fourier transform may be expressed as a Fourier transform of the analysis section $x_m(n)$, or

$$X_m(k) = \sum_{n=m}^{m+N-1} x_m(n)e^{-j2\pi(n-m)k/N}$$

where $k$ is the quantized frequency, $n$ is the quantized time, $x_m(n)$ is the portion of the original speech signal $x(n)$ being analyzed, $m$ is a constant, and $N$ is the number of samples that make up the $x_m(n)$.

$X_m(k)$ is a complex number made up of a real and the imaginary frequency terms. Another way of expressing $X_m(k)$ is by

$$X_m(k) = a(k) + jb(k)$$

where $a(k)$ is the real frequency component, and $b(k)$ the imaginary frequency component.

Another useful representation is

$$X_m(k) = c(k)e^{j\phi(k)}$$

where

$$c(k) = (a(k)^2 + b(k)^2)^{1/2}$$

and

$$\phi(k) = \tan^{-1}(b(k)/a(k))$$

The object of method 1 is to attempt the synthesis by forcing

$$\phi(k) = 0$$
Therefore, the frequency domain will contain all real, all positive frequency components. In this form

\[ X_m(k) = c(k) \quad (4.1.5) \]

4.1.1 Experiment 1.1

The key to this experiment is zero overlap, or

\[ s = N \quad (4.1.1.1) \]

where \( s \) is the separation between the analysis sections, and \( N \) is the number of samples in the analysis section. Looking at the synthesized waveshape, Figure 4.1.1.A, we observe no resemblance to the original waveshape. The frequency spectrum matched the original one. This, however, is of no significance because of the zero overlap. There is a total destruction of the original pitch.

The synthesized waveshape has two very sharp "edges" per synthesized window. These are the result of taking the inverse Fourier transform from the frequency domain whose imaginary terms are all zero, and all real terms are positive. It can be shown that each synthesized section is in the form

\[ y_m(n) = \frac{1}{N} \sum_{k=0}^{N-1} c(k) \cos(2\pi kn/N) \quad (4.1.1.2) \]

where \( k/(NT) \) is the incremental frequency at which the frequency spectrum is evaluated. In the case of this experiment, \( k/(NT)=78 \text{ Hz} \) for \( k=1 \), \( k/(NT)=156 \text{ Hz} \) for \( k=2 \), etc.

Because of the zero overlap, there is no reason to run an overlap-and-add algorithm, and therefore
FIGURE 4.1.1. A Waveshape Synthesized With Experiment 1.1

Analysis-synthesis method..............................................1
Number of samples per analysis-synthesis section...............N=128
Separation between analysis sections...............................s=128
Sampling period in microseconds.................................T=100
Pre-filtering function.................................................f1=U(n)-U(n-N)
Post-filtering function................................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid.....................Xt=700
Horizontal resolution in hertz per grid..........................Xf=136
Start of print in multiples of T/2.................................t0=4096
\[ y\,(n) = y\,(n) \]  

(4.1.1.3)

Thus a synthesized section \( y\) yields the synthesized window \( y\). From equation 4.1.1.2, we observe that \( y\,(n) \) is made up of \( N \) cosine terms, all in phase! So for \( n=0 \)

\[
N - 1 \\
y\,(0) = \frac{1}{N} \sum_{k=0}^{N-1} c(k) = y\,(N-1)
\]

(4.1.1.4)

which explains the sharp "edges" at \( n=0 \), and \( n=N-1 \). From now we will refer to the phenomenon observed in equation 4.1.1.4 as the "edge effect".

Listening to the synthesized waveshape provides a surprising amount of intelligibility, even though the pitch is destroyed. This tells us that the human ear and brain are somewhat insensitive to the speech phase, and quite sensitive to the pitch (or the lack of it). The synthesized speech sounds mechanical and contains a large degree of low frequency noise.

Observing Figure 4.1.A, we see that the "edges" occur every \( N \) samples. With the sampling rate of \( T=0.00001 \) seconds, and \( N=128 \), an edge occurs at the rate of about \( 1/NT=80 \) Hz, which explains the low frequency noise.

4.1.2 Experiment 1.2

This experiment attacks the issue of overlapping the analysis-synthesis sections and synthesizing them into speech. For this experiment, the separation between the analysis sections is set to \( s=16 \) samples, so the overlap factor is

\[ N/s = 8 \]

implying that to generate the synthesis window \( y\), we must overlap-and-add 15 \((2(N/s)-1)\) synthesized sections.

From section 2, we know that the orthodox STFTAS method would generate a near perfect speech waveform. Unfortunately, our experiment generated grossly unacceptable results, as shown in Figure 4.1.2.A. The speech fidelity significantly decreased compared to the experiment 1.1. This
FIGURE 4.1.2. A Waveshape Synthesized With Experiment 1.2

Analysis-synthesis method........................................................................1
Number of samples per analysis-synthesis section................................. N=128
Separation between analysis sections.................................................... s=16
Sampling period in microseconds......................................................... T=100
Pre-filtering function........................................................................... f1=U(n)-U(n-N)
Post-filtering function.......................................................................... f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid......................................... Xt=700
Horizontal resolution in hertz per grid............................................... Xf=136
Start of print in multiples of T/2........................................................ t0=4096
failure is mainly attributed to the fatal "edges" already encountered. The contribution of each overlap-and-add section is separated by $s=16$ samples, or about 1.6 milliseconds. Thus every 1.6 milliseconds an edge appears in the synthesized waveshape. Not surprisingly, the frequency spectrum of the synthesized waveshape is heavily dominated by frequency $1/sT$, or about 600 Hz, and its harmonics.

4.1.3 Experiment 1.3

This experiment increases the overlap factor to its maximum, by making the separation $s=1$. The analysis windows are now separated by a single sample, or .1 millisecond. The object of this experiment is to rid the synthesized waveshape of the "edge effect" encountered in experiment 1.1 and 1.2.

Figure 4.1.3.A shows the resulting synthesized waveshape. No longer plagued by the "edge effect", the synthesized waveshape has lost all of its medium and high frequency components. Most of the frequency domain energy is contained in the d.c., or $k=0$ term (about 90%). (Note that Figure 4.1.3 shows only about 1/2 of the d.c. term magnitude.)

The synthesized speech was not intelligible, as most of the original frequency spectrum components were destroyed. Before explaining the results, we perform experiment 1.4.

4.1.4 Experiment 1.4

Recalling the discussion of the window filtering function, $w_m(n)$, we execute the exact algorithm as in experiment 1.3, with an exception of letting $w_m(n)$ be a Hamming window. The resulting synthesized waveshape is shown in figure 4.1.4.A. Very similar in its infidelity and the frequency content as the waveshape generated in experiment 1.3, this waveform appears even richer in its d.c. content.

We now proceed to explain the reasons for obtaining the waveshape with 3 distinct phenomena. These are the lack of medium and high frequency components, a large d.c. term, and a presence of a sinusoidal component matching the pitch of the original waveshape.
FIGURE 4.1.3. A Waveshape Synthesized With Experiment 1.3

Analysis-synthesis method.................................................1
Number of samples per analysis-synthesis section..................N=128
Separation between analysis sections.................................s=1
Sampling period in microseconds........................................T=100
Pre-filtering function......................................................f1=U(n)-U(n-N)
Post-filtering function......................................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid............................Xt=700
Horizontal resolution in hertz per grid.................................Xf=136
Start of print in multiples of T/2.......................................t0=4096
FIGURE 4.1.4. A Waveshape Synthesized With Experiment 1.4

Analysis-synthesis method........................................1
Number of samples per analysis-synthesis section..............N=128
Separation between analysis sections..........................s=1
Sampling period in microseconds...............................T=100
Pre-filtering function...........................................f1=.54+.46\cos(2\pi n/N)
Post-filtering function..........................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid...................Xt=700
Horizontal resolution in hertz per grid.........................Xf=136
Start of print in multiples of T/2................................t0=4096
The lack of the medium and high frequency components is explained by the Fourier series representation of the original waveshape. One important assumption must be made - original waveshape is "quasi-stationary", meaning that the frequency content of analysis section \(m\) varies very slightly with respect to the frequency content of analysis section \(m+1\).

From Fourier series, we can represent the original analysis section as

\[
x_m(n) = \frac{1}{N} \sum_{k=0}^{N-1} c(k) \cos(2\pi kn/N + \theta(k))
\]

(4.1.4.1)

where \(c(k)\) is the value of \(X_m(k)\) at frequency \(k\). Using superposition, let us now attempt to synthesize \(y_m(n)\) by synthesizing one cosine term at a time. In particular, let us select a value of \(k\) such that \(k=i\).

Substituting in equation 4.1.4.1, we get

\[
x_m(n,i) = \frac{1}{N} c(i) \cos(2\pi in/N + \theta(i))
\]

(4.1.4.2)

We now use the overlap-and-method to synthesize the window \(y_m(n,i)\).

Recalling the definition of method 1, we see that the contribution of each synthesis section \(y_m(n,i)\) to \(y(n,i)\) will be

\[
y_m(n,i) = \frac{1}{N} c(i) \cos(2\pi in/N)
\]

(4.1.4.3)

since \(\theta(i)\) is forced to 0. Introducing the overlap factor \(N/s\), we express the synthesis window as the sum of all \(y_m(n,i)\) terms. This leads to

\[
y_m(n,i) = \frac{1}{N} \sum c(i) \cos(2\pi i(n-m)/N + 2\pi im/N)
\]

(4.1.4.4)

\(m\)

\(m = a-N+s, a-N+2s, \ldots, a+s, a+2s, \ldots, a+N-3s, a+N-2s, a+N-s\)

Summation valid for all \(n\), answer only valid for \(a-1 < n < a+N\)

This is a very important equation, since we learn from it why Method 1 is not successful, and possibly how to improve it. Above equation tells us that the synthesized waveform of frequency component \(i\) is made up of cosine waves, all equal in magnitude, but separated in phase. The phase angles are uniformly distributed around the circle, separated by \(2\pi is/N\). Using the phasor representation of each cosine, we can plot phasors around
the unit circle. For example, for \( i = 8, s = 1, T = 0.1 \) millisecond, and \( N = 128 \),
the phasors are separated by \( 2\pi (8)(1)/128 \), or roughly 22.5 degrees. By
simple vector addition, all of the phasors cancel each other, and we are
left with
\[
y\mathbf{*}(n,i) = 0 \quad \text{for } 0 < i < N \quad (4.1.4.5)
\]
This explains why all medium and high frequencies were missing from the
waveshape generated by experiment 1.4. We will refer to this phenomenon as
the "phasor cancellation".
Now we explain the dominating presence of the d.c. term. Looking back
at the equation 4.1.4.4 and letting \( i = 0 \), we get
\[
y\mathbf{*}(n,0) = \frac{1}{N} \sum_{m} c(0) \cos(2\pi 0 ((n-m)/N + 2\pi 0 m/N) \quad (4.1.4.6)
\]
or
\[
y\mathbf{*}(n,0) = \frac{1}{N} (N/s)c(0) = c(0)/s \quad (4.1.4.7)
\]
Realizing that the d.c. offset of \( x\mathbf{*}(n) \) is only \( c(0)/N \), we see that
\( y\mathbf{*}(n) \) d.c. term is \( N/s \) times greater. For \( s = 1 \), it is \( N \) times greater.
This means that each one of the sections making up \( y\mathbf{*}(n) \) contributes to
the d.c. term.
The third phenomenon is the existence of some very low frequency
components in the frequency spectrum of the synthesis window \( y\mathbf{*}(n) \),
which is contrary to equation 4.1.4.5. We explain this by amending our
previous assumption that the magnitude of all frequency components does
not vary from analysis section \( m \) to analysis section \( m+1 \). We correct
ourselves by making a suggestion that the low frequency components, such
as pitch, do change somewhat more noticeably compared with other
frequencies. This correction shall remain unproven.
4.2 Method 2 - Synthesis From Imaginary Power Spectrum

In Method 1 we have shown several reasons why synthesis from real magnitude was not successful. One of the problems noticed was the so-called "edge-effect".

Method 2 attempts to remove the "edge-effect" problem by making a slight change in the synthesis procedure. In this method we modify the frequency spectrum $X_m(k)$ from

$$X_m(k) = c(k)e^{j\phi(k)}$$  \hspace{1cm} (4.2.1)

to

$$X_m'(k) = c(k)e^{j\phi'/2} = j c(k)$$  \hspace{1cm} (4.2.2)

The resulting frequency spectrum's real part is zero, while the imaginary part contains the magnitude.

The reason for this attempt stems from the very definition of the Fourier transform, which tells us that an even function (eg. cosine) transforms to an all real frequency spectrum, while an odd function (eg. sine) transforms to an all imaginary frequency spectrum. Anticipating two "edges" per synthesized section $y_m(n)$, we expect a positive and a negative edge to occur. By overlapping the sections, positive and negative edges may cancel, thus eliminating the "edge-effect".

4.2.1 Experiment 2.1

This experiment had the same parameters as experiment 1.1. Not surprisingly, the results are similar in nature: destruction of pitch, and two sharp "edges" of opposite polarity, since an odd function was synthesized. The (in)fidelity was equal to that of experiment 1.1.

The synthesized waveshape, shown in Figure 4.2.1.A, can be shown to be in the form
FIGURE 4.2.1. A Waveshape Synthesized With Experiment 2.1

Analysis-synthesis method...............2
Number of samples per analysis-synthesis section...........N=128
Separation between analysis sections.......................s=128
Sampling period in microseconds.........................T=100
Pre-filtering function........................................f1=U(n)−U(n−N)
Post-filtering function........................................f2=U(n)−U(n−N)
Horizontal resolution in microseconds/grid...............Xt=700
Horizontal resolution in hertz per grid...................Xf=136
Start of print in multiples of T/2.........................t0=4096
\[ y^*(n) = \frac{1}{N} \sum_{k=0}^{N-1} c(k) \sin\left(\frac{2\pi k(n-a)}{N}\right) \text{ for } a-1 < n < a+N \quad (4.2.1.1) \]

From the above equation we see that \( y^*(n) \) is made up of \( N \) sine waves, each of different frequency, and all in phase. As a result, the edges are of the opposite polarity.

4.2.2 Experiment 2.2

Using the same parameters as in experiment 1.2, the synthesis is accomplished with no improvements. The edge effect is still present as the anticipated "edge" cancellation did not occur. The results are shown in Figure 4.2.2.A.

4.2.3 Experiment 2.3

Again, using the identical set of parameters as in experiment 1.3, we arrive at a nearly identical result (Figure 4.2.3.A). The only difference is the lack of the large d.c. component. We proceed to experiment 2.4, where this phenomenon is explained.

4.2.4 Experiment 2.4

Parameters used equal those from experiment 1.4. The major difference here is the lack of the enormous d.c. term seen in 1.4.

As in experiment 1.4, we observe (Figure 4.2.4) the "phasor cancellation" of medium and high frequency components. Without being repetitious, it can be shown that by using the superposition, we can synthesize one sine wave at a time, and then combine the result to get the
FIGURE 4.2.2. A Waveshape Synthesized With Experiment 2.2

Analysis-synthesis method .............................................. 2
Number of samples per analysis-synthesis section .................. N=128
Separation between analysis sections ................................. s=16
Sampling period in microseconds ...................................... T=100
Pre-filtering function ......................................................... f1=U(n)-U(n-N)
Post-filtering function ......................................................... f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid ......................... X_t=700
Horizontal resolution in hertz per grid ............................... X_f=136
Start of print in multiples of T/2 ...................................... t_0=4096
FIGURE 4.2.3. A Waveshape Synthesized With Experiment 2.3

Analysis-synthesis method: N=128
Number of samples per analysis-synthesis section: N=128
Separation between analysis sections: s=1
Sampling period in microseconds: T=100
Pre-filtering function: \( f_1 = U(n) - U(n-N) \)
Post-filtering function: \( f_2 = U(n) - U(n-N) \)
Horizontal resolution in microseconds/grid: \( X_t = 700 \)
Horizontal resolution in hertz per grid: \( X_f = 136 \)
Start of print in multiples of \( T/2 \): \( t_0 = 4096 \)
FIGURE 4.2.4. A Waveshape Synthesized With Experiment 2.4

Analysis-synthesis method .......................................................... 2
Number of samples per analysis-synthesis section ....................... N=128
Separation between analysis sections ........................................... s=1
Sampling period in microseconds ............................................... T=100
Pre-filtering function ............................................................... \( f_1 = 0.54 + 0.46 \cos(p\pi/n) \)
Post-filtering function .............................................................. \( f_2 = u(n) - u(n-N) \)
Horizontal resolution in microseconds/grid ............................... \( X_t = 700 \)
Horizontal resolution in hertz per grid ..................................... \( X_f = 136 \)
Start of print in multiples of \( T/2 \) .......................................... \( t_0 = 4096 \)
synthesized waveshape. The contribution of each frequency component \( i \) can be shown to be

\[
y_s(n,i) = \frac{1}{N} \sum_c(i) \sin(2\pi i(n-m)/N + 2\pi im/N) = 0 \quad (4.2.4.1)
\]

\( m = a-N+s, a-N+2s, \ldots, 0, a+s, a+2s, \ldots a+N-2s, N-s \)

Summation valid for all \( n \), answer only valid for \( a-1 < n < a+N \)

The presence of low frequencies is attributed to the varying pitch within the analysis window.

Note that the lack of the d.c. term is derived by letting \( i=0 \) in equation 4.2.4.1, which yeilds

\[
y_s(n,0) = 0 \quad (4.2.4.2)
\]

The d.c. component present in the synthesized waveform may be attributed to the d.c. offset in the original analysis window, \( x_m(n) \), figure 4.1.

4.3 Method 3 - Synthesis From Quantized Phase Spectrum

Methods 1 and 2 had proven to be unsuccessful synthesis methods. We have observed the problem termed "phasor cancellation". Method 3 makes an attempt to avoid this phenomenon.

Recall that Fourier transform of the short time analysis section is

\[
x_m(k) = c(k)e^{j\theta(k)} \quad (4.3.1)
\]

where \( c(i) \) is the magnitude of \( x_m(k) \) at \( k=i \). While Method 1 and 2 modified the phase angle \( \theta(k) \) to 0 or \( \pi/2 \), Method 3 changes it to

\[
\text{modified } \theta(k) = 0 \quad \text{if } -\pi/2 < \theta(k) < = \pi/2 \quad (4.3.2)
\]

\[= \pi \quad \text{elsewhere} \]
Simplifying equation 4.3.1,

\[ X_m(k) = c_1(k) - c_2(k) \quad (4.3.3) \]

where

\[ c_1(k) \geq 0 \quad \text{for modified } \theta(k) = 0 \]
\[ = 0 \quad \text{for modified } \theta(k) = \pi \]
\[ c_2(k) \geq 0 \quad \text{for modified } \theta(k) = \pi \]
\[ = 0 \quad \text{for modified } \theta(k) = 0 \]

The above equation tells us that the frequency spectrum phase will be retained in its quantized form only. The real part of the frequency domain contains the magnitude, which is negative for some frequencies. The imaginary part is zero.

Note that we are slightly deviating from the original plan of synthesizing from an all real, all positive frequency domain. As it will be shown later, this experiment will help us discover a method that does just that.

4.3.1 Experiment 3.1

This experiment attempts the synthesis from quantized phase spectrum, with N=s. Figure 4.3.1.A shows the resulting waveshape. The most significant difference between this and the previous two methods is the lack of the "edge effect". This difference stems from the very definition of Method 3.
FIGURE 4.3.1. A Waveshape Synthesized With Experiment 3.1

Analysis-synthesis method........................................3
Number of samples per analysis-synthesis section...............N=128
Separation between analysis sections............................s=128
Sampling period in microseconds.................................T=100
Pre-filtering function..............................................f1=U(n)-U(n-N)
Post-filtering function..............................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid..................Xt=700
Horizontal resolution in hertz per grid.........................Xf=136
Start of print in multiples of T/2..............................t0=4096
Looking back at equation 4.3.3, we can express the synthesis section $y_m(n)$ as

$$y_m(n) = \frac{1}{N} \sum_{k=0}^{N-1} c_1(k) \cos(2\pi kn/N) - c_2(k) \cos(2\pi kn/N) \quad (4.3.1.1)$$

Note that, statistically speaking, number of $c_1(k)$ non-zero terms will roughly equal that of $c_2(k)$ non-zero terms. Therefore, the cosine waves making up $y_m(n)$ will no longer add in phase, thus not creating previously discussed "edge effect".

The fidelity of the synthesized speech was only slightly better than those of experiments 1.1 and 2.1: intelligible, yet very mechanical with no pitch recognition, and without the low frequency noise, which is a direct result of eliminating the "edge effect" phenomenon.

4.3.2 Experiment 3.2

In this experiment N=128 samples, while the separation is set at $s=16$ samples. The results were much improved over those of 3.1. Observing the resulting waveshape, Figure 4.3.2.A, we notice several indications of high fidelity as the pitch is preserved, and the frequency spectrum of the synthesized window very much resembles that of the original analysis window, Figure 4.1.

Listening test did not produce the quality as high as the similarity between the synthesis window and the analysis window suggests. Although pitch existence was quite noticable, high frequency components seemed distorted. In general, the higher the frequency, the greater the distortion.

4.3.3 Experiment 3.3

Method 3 is repeated using the maximum overlap, or $s=1$. The results of synthesis are shown Figure 4.3.3.A. The synthesized window was reconstructed from the analysis window in all of the aspects. We see a distinct presence of pitch, near perfect reproduction of the time and frequency domain, and a high fidelity speech reconstruction.
FIGURE 4.3.2 A Waveshape Synthesized With Experiment 3.2

Analysis-synthesis method

Number of samples per analysis-synthesis section

Separation between analysis sections

Sampling period in microseconds

Pre-filtering function

Post-filtering function

Horizontal resolution in microseconds/grid

Horizontal resolution in hertz per grid

Start of print in multiples of T/2
FIGURE 4.3.3. A Waveshape Synthesized With Experiment 3.3

Analysis-synthesis method.................................................................3
Number of samples per analysis-synthesis section.........................N=128
Separation between analysis sections.............................................s=1
Sampling period in microseconds.................................................T=100
Pre-filtering function.................................................................f1=U(n)-U(n-N)
Post-filtering function...............................................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid...............................Xt=700
Horizontal resolution in hertz per grid......................................Xf=136
Start of print in multiples of T/2...............................................t0=4096
Careful analysis explains the success of this method. Again we take the approach of showing how a single frequency sinusoid is synthesized. Using equation 4.1.4.4, we can express the synthesis window for \( k=i \) as

\[
y(n,i) = \frac{1}{N} \sum_{m} c(i) \cos\left(\frac{2\pi i (n-m)}{N} + \frac{2\pi im}{N}\right)
\]

where \( c(i) \) is a function of \( c_1(i) \) and \( c_2(i) \).

We now derive \( c(i) \). Assumption is made again that there will be little variation in the individual sinusoids within the analysis window. As a result

\[
c_1(i) = c_2(i)
\]

Now, for \(-\pi/2 < \theta(i) \leq \pi/2\)

\[
y(n,i) = \frac{1}{N} \sum_{m} c_1(i) \cos\left(\frac{2\pi i (n-m)}{N} + \frac{2\pi im}{N}\right)
\]

and for \( \pi/2 < \theta(i) \leq 3\pi/2\)

\[
y(n,i) = \frac{1}{N} \sum_{m} c_2(i) \cos\left(\frac{2\pi i (n-m)}{N} + \frac{2\pi im}{N} + \pi\right)
\]

By adding equations 4.3.3.3 and 4.3.3.4, and using the equality in 4.3.3.2,

\[
y(n,i) = \frac{1}{N} \sum_{m} 2c_1(i) \cos\left(\frac{2\pi i (n-m)}{N} + \frac{2\pi im}{N}\right)
\]

if \(-\pi/2 < \theta(i) \leq \pi/2\)

\[= 0\] otherwise

Term \(2\pi im/N\) is the phase shift with respect to the sinusoid \( i \) of the original analysis window. For the sake of clarity, let us assume that the analysis window started at such \( n=a \) that frequency component \( i \) had a phase of \( \theta(i) = 0 \). Then if we execute Method 3, the synthesized component of
frequency $i$ can be represented as a phasor sum, as shown in Figure 4.3.3.B. Note that the resultant vector has the phase angle similar to that of the original sinusoid. Assumption that the original sinusoid has zero phase does not represent a special case. For any $\theta(i)$, all of the phasors will be such that the resultant vector will point in the direction of $\theta(i)$, as shown in Figure 4.3.3.C.

From experiments 3.1 to 3.3, we have seen a direct relationship between the fidelity of the synthesized speech and the overlap separation $s$. From Figure 4.3.3.B, it is clear that the ideal synthesis would require an infinite number of phasors, which would then perfectly reconstruct the original sinusoid and its phase. As $i$ approaches the bandwidth, the number of phasors decrease. At $i = \text{bandwidth}$, $s=1$, and providing $x(n)$ is being sampled at the Nyquist rate, only 2 phasors exist, both pointing in the same direction. Needless to say, the distortion of the highest frequency component will be the highest. To improve the fidelity, we may choose to either make $s$ smaller, or increase the sampling frequency of $x(n)$.

4.3.3 Experiment 3.4

Nearly identical to experiment 3.3, this experiment utilizes a Hamming window. The experiment yields nearly identical results, as shown in Figure 4.3.4.A. In order to determine which method is more precise, a computer of high accuracy is required for the time/frequency domain comparison between the original and synthesized speech.

Method 3 has a major drawback as this is not a true synthesis from the frequency power spectrum, because we must retain the quantized phase information. Valuable knowledge obtained in this experiment will allow us to improve the results even more.
FIGURE 4.3.3.B - Phasor Representation of Synthesis In Experiment 3.3
Assumption: Sinusoid being synthesized has a phase of zero.

FIGURE 4.3.3.C - Phasor Representation of Synthesis In Experiment 3.3
Assumption: Sinusoid being synthesized has a phase of 40 degrees.
FIGURE 4.3.4. A Waveshape Synthesized With Experiment 3.4

Analysis-synthesis method........................................3
Number of samples per analysis-synthesis section...............N=128
Separation between analysis sections.............................s=1
Sampling period in microseconds....................................T=100
Pre-filtering function.................................................f1=.54+.46COS(pin/N)
Post-filtering function.................................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid.........................Xt=700
Horizontal resolution in hertz per grid................................Xf=136
Start of print in multiples of T/2..................................t0=4096
4.4 Method 4 - Synthesis by Phase Prediction

Although Method 3 generated a near perfect synthesis, the requirement of quantized phase forces a continued search for a new STSASS method not requiring any phase information in frequency domain.

This method attempts the synthesis by predicting the phase from its frequency spectrum. The phase prediction algorithm is based on the Fourier series representation of the original signal, $x(n)$. Concentrating on a single analysis window between $n=0$ and $n=N-1$, we have

$$N-1$$
$$x_a(n) = \frac{1}{N} \sum_{k=0}^{N-1} c(k)e^{j2\pi kn/N}$$

(4.4.1)

or

$$x_a(n) = \frac{1}{N} \sum_{k=0}^{N-1} (a(k)\cos(2\pi kn/N) + j b(k)\sin(2\pi kn/N))$$

(4.4.2)

where

$$c(k) = (a(k)^2 + b(k)^2)^{1/2}$$

$$\theta(k) = \tan^{-1}(b(k)/a(k))$$

For a single frequency component $k=i$, we show a proof that the phase could be predicted. Let us assume that the phase angle $\theta(i)$ is known to be $\theta(i) = \theta_1$ for a particular analysis section $x_m(n)$. The sinusoid of frequency $i$ has a period of

$$T_i = NT/i$$

(4.4.3)

where $T$ is the sampling period of $x(n)$, and $N$ is the number of frequency terms in an analysis section. Therefore, the sinusoid $i$ has a phase $\theta_1$
every $T_i$ seconds.

From the definition of STSASS, the analysis sections are separated by $s$ samples. If $s$ is equal to $T_i$, then the phase of the next analysis section can be predicted to be $\theta_i + 2\pi i$. Similarly, if $s$ is equal to $1/2$ of $T_i$, then the phase of the next analysis section can be predicted to be $\theta_i + \pi i$.

Generalizing the above discussion, we conclude that for any frequency $i$

$$\theta_{m+1}(i) = \theta_m(i) + 2\pi i s/N \quad (4.4.4)$$

where $\theta_m(i)$ is the known phase of the sinusoid $i$ of analysis section $X_m$.

There are several disadvantages to this method of analysis-synthesis:

A. Equation 4.4.4 makes the assumption that a frequency component $i$ is a continuous sinusoid, whose phase can be predicted. This assumption may be incorrect, especially at a transition of voiced and unvoiced speech.

B. Calculation of the phase must be very precise and with minimum error, otherwise the error will increase rapidly from one analysis section to next. For example, if we generate a 0.1% error each time we calculate $\theta_m(i)$, the error would propagate at 0.001/(sT) per second.

C. Requirement that the initial phase must be known for all $N$ frequency components is a burden since we are attempting to synthesize with no phase information. An alternate approach is to simply set the initial phase for each frequency component to 0. It can be shown that this will lead to results similar Method 1, as we would again be faced with the "edge effect" problem, since all sinusoids would be in phase. Setting random initial phases has been attempted during experimental research at Strong Memorial Hospital, Rochester, NY. The quality of synthesized speech was said to be very mechanical, and there was a lack of pitch.

In view of the above discussion no experiments were conducted.

4.5 Method 5 - Synthesis From Modified Power Spectrum

This method of analysis-synthesis is very similar to Method 3, as an attempt is made to synthesize the speech from an all real, all positive
frequency spectrum.

This method makes a modification to the calculation of the frequency spectrum. Recalling the equation 4.3.1 derived in method 3, we express the modified frequency domain magnitude as

$$X_m(k) = c(k)$$  \hspace{1cm} (4.5.1)

where

$$c(k) = \left( a(k)^2 + b(k)^2 \right)^{1/2}$$  \hspace{1cm} (4.5.2)

if $$-\pi/2 < \tan^{-1}(b(k)/a(k)) \leq \pi/2$$, 0 otherwise

Above equation says that if the phase of the frequency component is between $$-\pi/2$$ and $$\pi/2$$, the magnitude $$c(k)$$ is calculated in the traditional way. For frequency components whose phase falls out of the specified range, $$c(k) = 0$$.

4.5.1 Experiment 5.1

This experiment uses the same parameters as experiment 3.3, mainly s=1, and N=128. Figure 4.5.1.A shows the resulting waveshape. Comparing it with the original waveshape, we see a near perfect reproduction. Listening test reveals all signs of high fidelity speech, with no noticeable differences compared to the original.

The explanation for the high quality of synthesis lies in the results of Method 3, where we have shown that each sinusoid can be synthesized separately. Recall that

$$y_m(n,i) = \left( \frac{1}{N} \right) \sum_{m} c_1(i) \cos(2\pi kn/N) - c_2(i) \cos(2\pi kn/N)$$  \hspace{1cm} (4.5.1)

We now force an all real, all positive frequency domain by forcing

$$c_2(i) = 0$$  \hspace{1cm} (4.5.2)
FIGURE 4.5.1 A Waveshape Synthesized With Experiment 5.1

Analysis-synthesis method

Number of samples per analysis-synthesis section \( N = 128 \)

Separation between analysis sections \( s = 1 \)

Sampling rate in microseconds \( T = 100 \)

Pre-filtering function \( f_1 = U(n) - U(n-N) \)

Post-filtering function \( f_2 = U(n) - U(n-N) \)

Horizontal resolution in microseconds/grid \( X_t = 700 \)

Horizontal resolution in hertz/grid \( X_f = 700 \)
Thus

\[ y_m(n) = \frac{1}{N} \sum_{m} c_m \cos(2\pi k n/N + 2\pi i m/N) \]  
(4.5.3)

Note that the only difference between this equation and equation 4.3.3.5 is a factor of 2. We observe that in Method 3 the information provided by \( c_2(k) \) term is redundant and thus Method 5 ends up utilizing the same amount of information as method 3. All of the synthesis-fidelity vs. analysis-synthesis-parameters discussion in Method 3 applies here as well.

Figure 4.5.1.B shows a time domain illustration of Method 5 synthesis applied to a single sinusoid. The frequency domain example is very similar to that of Figure 4.3.3.B and 4.3.3.C.

4.5.2 Experiment 5.2

Using identical parameters as in 5.1, we only change the filtering function to Hamming window. There was neither noticeable change in the time/frequency domain of the synthesized waveshape, nor change in the speech quality compared with the results of experiment 5.1.
FIGURE 4.5.1.B Example: Synthesis of Sinusoid Using Method 5
FIGURE 4.5.2. A Waveshape Synthesized With Experiment 5.2

Analysis-synthesis method..........................................................5
Number of samples per analysis-synthesis section.................N=128
Separation between analysis sections.................................s=1
Sampling rate in microseconds..........................................T=100
Pre-filtering function.....................................................f1=.54+.46\cos(2\pi n/N)
Post-filtering function....................................................f2=U(n)-U(n-N)
Horizontal resolution in microseconds/grid.........................X_t=700
Horizontal resolution in hertz/grid....................................X_f=700
5. EVALUATION OF STSASS METHODS

5.1 Method Comparison

Section 4 presented the research and results of short time spectrum analysis-synthesis methods (STSASS) utilizing very little or no phase information.

In Method 1 we synthesized directly from the short time Fourier transform magnitude. As the separation (parameter 's') between analysis sections was decreased, the quality of the synthesized speech also decreased. At s=1, the synthesized speech was unintelligible. Two distinct phenomena were noted. At large values of s, we observed the "edge effect", an allignment of all frequency components in phase with each other, causing a pitch destruction. At small values of s, we detected the "phasor cancellation", which destroyed all medium and high frequency components.

Method 2 very slightly deviated from Method 1, as the short time Fourier transform magnitude was moved from the real part of frequency domain to the imaginary part, while the real part was forced to zero. The main reason behind this attempt was to rid the synthesized speech of the "edge effect". The experiments revealed no improvements over Method 1.

Method 3 attempted to use the minimum amount of phase information by assigning a sign to the short time Fourier transform magnitude. Those frequency components whose phase angle was in the -90 to +90 degree range were unmodified, while all others were assigned a negative sign. The goal of this approach was to force the frequency components not to add in phase upon synthesis, thus avoiding the "edge effect". The experiments produced satisfactory results. In addition, at low values of analysis section separation (s=1,2,4,8) the synthesized speech showed all signs of high fidelity. The investigation of Method 5 experiments revealed a correlation between small analysis section separation and synthesis quality.

Not satisfied with the requirement that the phase must be retained in its quantized form, Method 4 attempted the synthesis by phase prediction. Given the analysis section separation, s, number of samples per analysis-synthesis section, N, the frequency component, i, and the known
phase angle of the previous synthesis section was shown to be enough information in predicting the phase of the synthesized component. Method 4 proved to have several problems such as a requirement for knowledge of the initial phase angle of each frequency component, propagation of error from the calculated phase angle to the next calculation, and the inability to predict the affect on phase as speech makes a transition from unvoiced to voiced speech.

Method 5 is a modified version of Method 3. The major difference is the zeroing of frequency components whose short time Fourier transform magnitude was assigned a negative sign. It was shown that this apparent loss of data is only the elimination of redundant information. Method 5 generated a crisp, high fidelity speech, whose quality improved as the separation between analysis section approached s=1. This method of synthesis is judged to be the best. Even though the synthesized speech was not constructed from the short time Fourier transform magnitude, the synthesis medium was an all real, all positive frequency domain. Since all non zero frequency components equaled those of the frequency magnitude spectrum, Method 5 can be thought of as the synthesis from the modified short time Fourier transform magnitude.

5.2 Affect of Analysis-Synthesis Parameters on Synthesis Quality

This section attempts to present the affect of analysis-synthesis parameters on the quality of speech synthesized with Method 5. To aid the explanation, numerous speech waveforms have been synthesized using varying parameters. Table 5.2.A (#1-#16) shows the experiments conducted, and parameters used. The quality of the results has been evaluated based on intelligibility. Rather than giving a quantitative value of quality, a correlation is made with analysis-synthesis parameters and the intelligibility of synthesized speech. Little effort will be devoted to explaining the dependency of speech quality on certain parameters, as all of the discussion may be found in the previous sections.
### TABLE 5.A - List of STSASS Experiments and Their Parameters

<table>
<thead>
<tr>
<th>Exp#</th>
<th>1/T</th>
<th>N</th>
<th>s</th>
<th>f1*</th>
<th>f2*</th>
<th>A</th>
<th>Tape**</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10Khz</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3-8</td>
<td>original</td>
</tr>
<tr>
<td>2</td>
<td>10Khz</td>
<td>128</td>
<td>128</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>1</td>
<td>9-14</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>10Khz</td>
<td>128</td>
<td>32</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>2</td>
<td>18-22</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>10Khz</td>
<td>128</td>
<td>16</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>4</td>
<td>25-28</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>10Khz</td>
<td>128</td>
<td>8</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>8</td>
<td>30-35</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>10Khz</td>
<td>256</td>
<td>8</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>16</td>
<td>35-40</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>10Khz</td>
<td>128</td>
<td>8</td>
<td>H(n)</td>
<td>U(0,N)</td>
<td>2</td>
<td>40-43</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>10Khz</td>
<td>256</td>
<td>16</td>
<td>H(n)</td>
<td>U(0,N)</td>
<td>2</td>
<td>45-50</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>10Khz</td>
<td>128</td>
<td>8</td>
<td>H(n)</td>
<td>H(n)</td>
<td>2</td>
<td>50-53</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10Khz</td>
<td>128</td>
<td>16</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>56-59</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>10Khz</td>
<td>256</td>
<td>32</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>60-64</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>10Khz</td>
<td>256</td>
<td>64</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>65-68</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>10Khz</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>68-70</td>
<td>original</td>
</tr>
<tr>
<td>14</td>
<td>10Khz</td>
<td>128</td>
<td>1</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>64</td>
<td>70-73</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>10Khz</td>
<td>64</td>
<td>4</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>4</td>
<td>75-78</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>10Khz</td>
<td>64</td>
<td>4</td>
<td>U(0,N)</td>
<td>U(0,N)</td>
<td>4</td>
<td>85-88</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>10Khz</td>
<td>128</td>
<td>128</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>92-96</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>10Khz</td>
<td>128</td>
<td>1</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>98-102</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>10Khz</td>
<td>128</td>
<td>128</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>104-108</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>10Khz</td>
<td>128</td>
<td>1</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>110-114</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>10Khz</td>
<td>128</td>
<td>128</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>116-120</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>10Khz</td>
<td>128</td>
<td>1</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>122-126</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td>10Khz</td>
<td>128</td>
<td>128</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>128-132</td>
<td>5</td>
</tr>
<tr>
<td>24</td>
<td>10Khz</td>
<td>128</td>
<td>1</td>
<td>H(n)</td>
<td>H(n)</td>
<td>4</td>
<td>134-138</td>
<td>5</td>
</tr>
</tbody>
</table>

Notes:  
* H(n) = .54+.46\cos(2\pi n/N), \quad U(0,N) = u(n-0) - u(n-N)
** Cassette tape counter, full count is 700 for 90 min. tape
! Waveshape generated by exp. 14 was analyzed-synthesized
5.2.1 Sampling Period, T

It was already shown that the "safe" sampling rate relates to the speech bandwidth $\text{Bw}$ as $1/T > 3\text{Bw}$. Increasing the sampling period to the order of Nyquist rate of sampling is risky as the aliasing may occur. Decreasing $T$ improves the synthesis quality since the phase of high frequency components can be estimated more accurately with Method 5.

5.2.2 Number of Samples Per Analysis-Synthesis Section, $N$

Three values of $N$ were attempted: $N=64$, $N=128$, and $N=256$. In general, intelligibility of synthesized speech was directly related to the increasing $N$. It was shown in section 2 that $N$ should be at least four times as long as the pitch period. This requirement was satisfied at $N=256$. Compared with experiments using $N=128$, speech synthesized with $N=256$ was very slightly superior in quality, with the noticable bass response improvement. At $N=64$ quality significantly decreased, as the high frequency components appeared to be distorted.

5.2.3 Separation Between Analysis-Synthesis Sections, $s$

Parameter $s$ has a very strong influence on the quality of speech synthesized with Method 5. Smaller values of $s$ synthesize higher quality speech than the large values of $s$. At $s=1$ the synthesized speech is near perfect. Quality is retained at $s=2$. At $s=4$ quality decreases, while $s=8$ can be considered to be the threshold of fidelity. In general, as $s$ increases, phase of frequency components is distorted. The distortion first strikes the high frequency components and works its way toward low frequencies as $s$ is increased.

5.2.4 Pre-Filtering Function, $w_m(n)$

Hamming window ($w_m(n) = .54 + .46 \cos(2\pi n/N)$) and unit window ($w_m = u(n) - u(n-N)$) were attempted. In all cases, Hamming window helped synthesize slightly superior speech waveform.
5.2.5 Post-Filtering Function, \( w'(n) \)

When some values of parameter \( s \) were used (eq. \( s=16, s=32, s=64, \) etc) the synthesized speech was distorted due to overlapping-and-adding of the synthesis sections. The region of overlap was discontinuous and thus the distortion was generated, with a fundamental frequency component of \( 1/(sT) \). To help attenuate the regions of overlap, post-filtering was introduced. The filter was applied to the synthesis section \( y_m(n) \), generating \( y'_m(n) \), where

\[
y'_m(n) = y_m(n)w'_m(n) \quad (5.2.5.1)
\]

\( y'_m(n) \) was then used in the overlap-and-add algorithm.

In the experiments conducted we used a Hamming window for post-filtering, as the quality of the synthesized speech was greatly improved. Use of post-filtering was not needed for small analysis-synthesis section separations (\( s=4, s=2, s=1 \)).

5.2.6 Attenuating Factor, \( A \)

Parameter \( A \) is needed to adjust the magnitude of the synthesized speech so that it is comparable to the original one. The experiments showed that \( A \) can be determined by

\[
A = 0.707A_1A_2(s/N) \quad (5.2.6.1)
\]

where constants \( A_1 \) and \( A_2 \) are a function of the pre-filtering and post-filtering functions used in the synthesis. For a Hamming window a factor of 2 should be used, while a unit window requires a factor of 1. Term \( (s/N) \) is needed to account for the overlap-and-adding, while 0.707 is derived from the very definition of Method 5. The most logical time for applying factor \( A \) is once the synthesized window \( y_m(n) \) was calculated. The resulting waveshape is

\[
y'_m(n) = Ay_m(n) \quad (5.2.5.2)
\]

Quality of synthesized speech does not vary with \( A \), unless an improperly selected \( A \) causes the speech playback system to clip the waveform.
5.3 Practical Considerations

At this time we evaluate Method 5 as a practical means of analysis-synthesis of speech in a real system. Frequent comparisons will be made to the already existing methods, mainly the M.I.T. method (section 3) and the general short time Fourier transform analysis-synthesis (STFTAS) method (section 2). The reader will also be presented an overview of considerations which must be taken when designing a short time analysis-synthesis system.

Method 5 achieves the synthesis by modifying the short time Fourier transform magnitude. The modification was shown to be necessary in order to preserve an "encoded" phase information in the modified magnitude spectrum. The modification was shown to rid the magnitude spectrum of only the "redundant" information.

Although the phase information was required for calculating the modified frequency magnitude, in its final form the frequency domain was all real and positive. This implies that in order to use Method 5, we must be provided the phase information in some form. This is also the case with the general STFTAS method, while the M.I.T. method does not require any phase knowledge (however, the initial samples must be known). Therefore, any system which attempts to reconstruct the signal from the short time Fourier transform magnitude may not utilize Method 5. However, these types of systems are rare. The most common use of the short time analysis-synthesis approach is for the speech enhancement. This type of application does offer the phase information.

The quality of synthesized speech using Method 5 was shown to vary from near perfect to poor, depending on the choice of parameters. In all cases the speech was intelligible. The other two methods also showed the ability to reproduce speech with high fidelity, although the M.I.T. method was quite sensitive to the choice of parameters.

The amount of information per analysis section used by Method 5 is roughly 1/2 of that of the M.I.T. method and 1/4 of that of the STFTAS method. This is very important for systems which may choose to communicate with other systems through the short time frequency domain medium.
However, Method 5 requires a smaller degree of separation between the analysis sections than the STFTAS method, in order to obtain an equal quality synthesis.

From the calculation time point of view, Method 5 offers a great simplicity in calculating the short time synthesis section (see section 3.3). Given the fairly small separation requirement of Method 5 and M.I.T. method, STFTAS method is most likely the most efficient when it comes to synthesizing high fidelity speech. At less than optimized quality, Method 5 requires less calculation.

Both STFTAS and Method 5 are realizable in a real-time system. Typical system would employ parallel Fourier transform and inverse Fourier transform stages. Method 5 also requires the way of calculating the modified short time frequency magnitude. M.I.T. method's real time realization is doubtful due to the great complexity.

From the above discussion, it should be apparent that decision about which method to use strictly depends on the analysis-synthesis system and its specifications. Method 5 was shown to have some advantages, as well as disadvantages over the other two methods. It seems that the most beneficial application of Method 5 is in systems where speech must be enhanced or modified to a given frequency domain magnitude whose phase is unknown or unspecified. The issue of frequency domain modification of speech using Method 5 was beyond the limits of this paper. However, this is a very important issue which should be investigated so that the true worth of Method 5 can be determined.
6. CONCLUSION

This thesis presented a study of short time spectrum analysis-synthesis of speech (STSASS), which is a special branch of short time Fourier transform analysis-synthesis (STFTAS). STFTAS method was defined. STSASS was then described, along with a presentation of STSASS efforts previously attempted by others. Five methods of STSASS were then presented, analyzed in detail, and backed up with experimental data generated with Speech Analysis Synthesis System (see Appendix A.1 and A.2) One of the methods was shown to synthesize a near perfect speech. This was achieved with a slight modification of the short time Fourier transform magnitude. The affect of analysis-synthesis parameters on the quality of the synthesized speech was discussed. Finally, practical design considerations were imposed on the STSASS methods. The method discovered here was shown to have several advantages as well as disadvantages over other methods. It was suggested that this method be farther analyzed for its feasibilty in the field of short time frequency modification and enhancement of speech.
7. APPENDIX

A.1 Speech Analysis-Synthesis System (SASS) Hardware Overview

A.2 Speech Analysis-Synthesis System (SASS) Software Overview

A.3 Computer Listing of STSASS Algorithm
APPENDIX A.1

Speech Analysis - Synthesis System Hardware Overview

The Speech Analysis - Synthesis System (SASS) is a microprocessor controlled multi-board circuit designed to analyze speech and experiment with methods of speech synthesis. In an attempt to provide a high level of flexibility, many special features have been designed into SASS, such as a serial link to a printer and computer terminal, 32 character alphanumeric vacuum-flourescent display, and many others.

SASS was designed using a modular approach. It is made up of 4 circuit boards (Figure A.1.A), each one having a specific function.

A.1.1 Central Processing Unit (CPU)

This is the "brain" of SASS. Equipped with a microprocessor, CPU controls all of the SASS data transfers. CPU includes the following features:

A. MC68B09 8 bit microprocessor
B. 10K bytes of static RAM
C. 20K bytes of PROM
D. RS232C serial port to computer terminal
E. RS232C serial port to printer
F. Trackball interface
G. Audio transducer with programmable volume, frequency
H. Watchdog timer
I. Power fail detector
J. Full microprocessor bus interface to an external module
FIGURE A.1.A - SASS Board Level Block Diagram
A.1.2 Input/Output (I/O) Module

This module expands the capability of CPU module by adding the following functions:
A. Two 32K byte blocks of battery backed static RAM
B. 32 character vacuum flourescent display (VFD)
C. Digital portion of A/D converter
D. Interface to external keyboard module
E. Interface to an audio A/D - D/A converter module

A.1.3 Keyboard Module

Keyboard module provides an operator interface to SASS. It includes the following:
A. 88 key-switch matrix with N-key rollover
B. 8 LEDs for visual indication

A.1.4 Audio Module

Speech signals processed by SASS are received and transmitted through the Audio module, which includes:
A. Microphone input with a low-pass filter
B. Analog portion of 12 bit analog-to-digital (ADC) converter
C. 12 bit Digital-to-analog (DAC) converter
D. Buffered audio output with a low pass filter
Speech Analysis-Synthesis System (SASS) Software Overview

This section describes the software package written for the Speech Analysis - Synthesis System (SASS). SASS software is intended for the research purposes in the field of speech analysis and synthesis. Equipped with a variety of arithmetic functions, SASS software is capable of performing anything from a floating-binary-point number addition, subtraction, multiplication, and division, to more complicated operations such as a complex fast Fourier transform algorithm and calculation of the magnitude of a complex number.

SASS software resides in the PROM. Nearly 20K bytes long, the code has been written in the MC68B09 microprocessor assembler language and is made up of roughly 90 modules (subroutines). The main objectives of the design were high speed and high level of flexibility.

MC68B09 is an 8 data bit, 16 addresses bit microprocessor. Internally, the microprocessor contains six 16 bit registers. Capable of executing over 1500 instructions, MC68B09 is regarded as one of the most powerful 8 bit microprocessors available on the market today.

For developmental/debugging purposes, SASS software includes a diagnostic Monitor program. Used in conjunction with the computer terminal, the Monitor program is capable of reading and writing specified locations in the memory, copying a block of memory to a specified address, jumping to any part of the SASS program, executing a single SASS subroutine, displaying the contents of the microprocessor’s internal registers, etc. Monitor program is an extremely valuable part of the SASS software and that without it the developmental efforts would be seriously impeded.
SASS operating system uses the keyboard inputs to determine the mode of operation. There are four possible modes:

- Transmit Mode
- Fast Analysis-Synthesis Mode
- Slow Analysis-Synthesis Mode
- Print Mode

Due to their complexity, most modes are subdivided into submodes. The following sections present a brief description of all modes and submodes.

1. Transmit Mode

All keyboard generated codes are passed to the serial port. In this mode SASS acts as a dumb keyboard.

2. Fast Analysis-Synthesis Mode

This mode, made up of 5 submodes, has the responsible for the speech analysis and synthesis. The analysis is performed on the sampled speech input stored in the 64K byte section of RAM. The result of the analysis is a synthesized speech stored in the 64K RAM.

2.1 Configure Parameters Submode - allows operator to specify all of the parameters of the Fast Analysis-Synthesis Mode. Each parameter name is displayed on the vacuum fluorescent display (VFD) one at a time. The operator enters the parameter value which is followed by another parameter name. The configuration process is completed once all parameters have been entered. Parameters include sampling period of speech, pre-filtering and post-filtering functions, separation between the analysis sections, length of analysis sections, analysis-synthesis method, etc.

2.2 Review Mode Parameters Submode - displays on the VFD all of the Fast Analysis-Synthesis Mode parameters entered during configuration.
2.3 Initialize A/D Input Submode - initializes the speech sampling from analog-to-digital converter. Once initialized, the speech input is sampled and data stored in the 64K RAM section.

2.4 Initialize D/A Output Submode - initializes the playback of the speech stored in the 64K section. Once initialized, data is transferred from the 64K RAM to the digital-to-analog converter.

2.5 Fast Analyze-Synthesize Submode - performs short time spectrum analysis-synthesis on the sampled speech stored in the 64K RAM using the parameters specified by the operator.

3. Slow Analysis-Synthesis Mode

This mode of operation is similar to the Fast Analysis-Synthesis Mode in a way that both modes perform the analysis and synthesis of speech. However, while the main goal of Fast Analysis-Synthesis mode is speech synthesis, Slow Analysis-Synthesis is concerned with the intermediate steps involved in the speech synthesis. As a result, this mode is capable of printing out a variety of time and frequency domain wave-shapes in order to help the operator explain why a certain synthesis method may or may not be successful. This mode is made up of 4 sub-modes.

3.1 Configure Parameters Submode - allows the operator to specify all of the parameters of the Slow Analysis-Synthesis Mode. See 2.3.1 for more details. Additional parameters include frequency/time domain printout flags for various stages of the analysis-synthesis process.

3.2 Review Mode Parameters Submode - displays on the VFD all of the Slow Analysis-Synthesis Mode parameters entered during the configuration.

3.3 Write To Printer Submode - allows the operator to send all keyboard generated inputs to the printer. This sub-mode is helpful if an operator wishes to add comments to the printouts generated by 3.4. All of the characters sent to the printer will be also displayed on the VFD.
3.4 Slow Analyze-Synthesize Submode - performs the short time spectrum analysis-synthesis on the sampled speech stored in the 64K RAM using the parameters specified by the operator. Various stages of analysis-synthesis sections are printed in both frequency and time domain.

4. Print Mode

This mode of operation allows the operator to print the waveshapes out of the 64K RAM. Printing parameters may be varied by the operator. This mode is made up of 4 sub-modes.

4.1 Configure Parameters Submode - allows the operator to specify all parameters of the Print Mode. Parameters include start/end of printout, vertical sale, time compression factor, D.C. offset, etc.

4.2 Review Mode Parameters Submode - displays on the VFD all of the Print Mode parameters entered during the configuration.

4.3 Write To Printer Submode - allows the operator to send all the keyboard generated inputs to the printer. This sub-mode is helpful if an operator wishes to add comments to the printouts generated by 4.4. All characters sent to the printer will be also displayed on the VFD.

4.4 Print Waveshape Submode - reads the specified range of data out of the 64K RAM and converts each value to a pixel on the graph.
APPENDIX A.3

COMPUTER LISTING OF STSASS ALGORITHM
INDSORI  ONDREJKA  11/21/84  7:00:55 E.S.T. WAS THE ORIGIN

DEST: ENGPR1  FILE: 1624  NAME: RUNFFT  6809  DIST: W90|MFO  RECS:

*******************************************************************************
FILE: RUNFFT 68D9 Q COMBUSTION ENGINEERING, INC.

TITLE 'RUNFFT
NAME RUNFFT
LIST X,B

**************
* RUNFFT *
**************

* THIS MODULE WILL RUN THE COMPLETE SHORT-TIME FAST FOURIER ANALYSIS
* AND SYNTHESIS LOOP ON THE SPEECH SAMPLES LOCATED IN THE MAIN RAM
* STORAGE. RAM STORAGE IS MADE UP OF TWO 32K BYTE SECTIONS. (SEE NOTE)
* 00010000
* 00011000

* THE ANALYSIS-SYNTHESIS RUNS AS FOLLOWS:
* (NOTE: SP1 AND SP2 ARE CONSECUTIVE SCRATCHPAD BUFFERS, EACH .75K BYTES)
* 00012000
* 00013000
* 00014000

* 1. READ PARAMETERS OF ANALYSIS
* 00015000
* 00016000
* 00017000

* 2. INITIALIZE ALL LOCAL VARIABLES AND POINTERS
* 00018000

* 3. ENABLE LOW 32K SECTION OF RAM
* 00019000

* 4. CALCULATE OVERLAP FACTOR ( USED FOR WEIGHTED TABLE ADDITION )
* 00020000

*---------------------LOOP STARTS HERE---------------------
* 00021000

* 5. TRANSFER A SECTION WORTH OF SAMPLES TO SCRATCHPAD BUFFER1 (SP1)
* 00022000

* 6. MULTIPLY BY SELECTED PRE-FILTER WINDOW FUNCTION ( SP1 =/> SP1 )
* 00023000

* 7. BIT-REVERSE TABLE ENTRIES (SP1 =/> SP2)
* 00024000

* 8. ADD IMAGINARY PART TO TABLE GENERATED IN #6 (SP2 =/> SP1+SP2)
* 00025000

* 9. RUN A FAST FOURIER TRANSFORM ALGORITHM ( SP1+SP2 =/> SP1+SP2 )
* 00026000

* 10. FIND MAGNITUDE OF COMPLEX WINDOW ( SP1+SP2 =/> SP1 )
* 00027000

* 11. BIT-REVERSE TABLE ENTRIES (SP1 =/> SP2)
* 00028000

* 12.** ADD IMAGINARY PART TO TABLE GENERATED IN #10 (SP2 =/> SP1+SP2)
* 00029000

* 13. RUN AN INVERSE FFT ALGORITHM ( SP1+SP2 =/> SP1+SP2 )
* 00030000

* 14. COMPRESS THE COMPLEX WINDOW ( SP1+SP2 =/> SP1 )
* 00031000

* 15. ADD DYNAMIC TO STATIC TABLE
* 00032000

* 16. IF SPEECH WINDOW SYNTHESIS COMPLETE
* 00033000

* 17-1.1 THEN DIVIDE FIRST HALF OF DYNAMIC WINDOW BY OVERLAP FACTOR
* 00034000

* 17-1.2 TRANSFER SYNTHESIZED WINDOW TO SCRATCHPAD ( =/> SP1 )
* 00035000

* 17-1.3 COPY UPPER HALF OF STATIC BUFFER TO LOWER HALF
* 00036000

* 17-1.4 FILL UPPER HALF WITH FLOATING POINT ZEROS
* 00037000

* 17-1.5 REinitialize DYNAMIC POINTER.
* 00038000

* 17-1.6 COPY WINDOW FROM SCRATCHPAD TO RAM (SP1 =/> RAM)
* 00039000

* 17-1.7 UPDATE BLOCK POINTER
* 00040000

* 17-1.8 DETERMINE WHETHER TO STAY IN CURRENT 32K BLOCK, GO TO
* 00041000

* ... THE NEXT ONE, OR END.
* 00042000

* 18-2.1 ELSE UPDATE RAM SAMPLE POINTER
* 00043000

* 19. RETURN TO #6. UNTIL ALL ANALYSIS WINDOWS SYNTHESIZED.
* 00044000

* 20. END

* 00045000

* . NOTE: ** STEP 12 MAY DIFFER DEPENDING ON ANALYSIS-SYNTHESIS VERSION

* **********************************************
* ANY KEY INPUTS WILL BE IGNORED. THEY WILL BE TREATED AS AN ERROR,
* AND THE AUDIO FEEDBACK WILL BE GENERATED.
* 00046000

* AN AUDIO CLICK WILL BE GENERATED AFTER A PASS THRU AN FFT ALGORITHM
* 00047000

* 8 AUDIO CLICKS WILL OCCUR UPON THE COMPLETION OF THE SPEECH WINDOW
* 00048000

* SYNTHESIS. AN LED WILL BE TURNED ON UPON THE START OF THIS SUBROUTINE
* 00049000

* AND TURNED OFF UPON COMPLETION.
* 00050000

* NOTE: THE RAM IS MADE UP OF 2 - 32K SECTIONS. A "D" ON THE S1 LINE OF
* 00051000

* 00052000

* 00053000

* 00054000

* 00055000
FILE: RUNFFT 6809 O COMBUSTION ENGINEERING, INC.

* THE I/O WRITE LATCH ENABLES LOW 32K SECTION OF RAM, AND A "1" THE UPPER 32K BLOCK.
* RECEIVES INITIALIZATION (F1)
  * CONTROL_BYTE (D1)
  * DATA_BYTE (D1)
  * IF INITIALIZATION=FALSE
* CALLS
  * CPARAM COPY_PARAMETERS
  * WRITLAT WRITE_LATCH, I/O
  * TRANSF TRANSFER, SAMPLES FROM RAM TO LOCAL SCRATCHPAD
  * WNDMLT WINDOW_MULTIPLY
  * REXPND, ADD F.P.N. ZEROES IN PLACE OF IMAGINARY NUMBERS
  * IEXPND , ADD F.P.N. ZEROES IN PLACE OF REAL NUMBERS
  * COMPRESS COMPRESS, DELETES IMAGINARY F.P.N. FROM COMPLEX TABLE
  * FFT FAST_FOURIER_TRANSFORM_ALGORITHM
  * FNDMAG FIND_MAGNITUDE, OF TABLE WITH COMPLEX NUMBERS
  * FNDMAG1 FIND_MAGNITUDE, RETAIN SIGN
  * FNDMAG2 FIND_MAGNITUDE, CALL IT ZERO IF REAL < ZERO
  * IFFT INVERSE_FAST_FOURIER_TRANSFORM_ALGORITHM
  * ADDING , ADDS TWO SIGNED F.P.N.'S
  * IRRBACK TRANSFER_BACK, SAMPLES FROM SCRATCHPAD TO RAM
  * COPY , COPY A BLOCK OF BYTES
  * WCDOG , SINCE THIS MODULE COULD TAKE LONG TO EXECUTE
  * CLICK , AUDIO FEEDBACK
  * BITREV BIT_REVERSE, ...TABLE ENTRIES
  * WRTLED WRITE_LEDS
  * COMP COMPLEMENT F.P.N.
* VARIABLES :
* 1.LOCALS
  * Prews (D1) BEFORE_FILTERING_FUNCTION
  * Postws (D1) AFTER_FILTERING_FUNCTION
  * Rate (D1) , SAMPLING PERIOD IN USECONDS
  * Nw (D2) NUMBER_OF SECTIONS TO ANALYZE-SYNTHESIZE
  * Nspw (D1) NUMBER_OF_SAMPLES_PER_SECTION_MINUS_1
  * Nsbw (D1) NUMBER_OF_SAMPLES_BETWEEN_SECTIONS_MINUS_1
  * Strt1 (D2) START_1 , START OF DYNAMIC BUFFER
  * Strt2 (D2) START_2 , MIDDLE OF STATIC BUFFER
  * Strt3 (D2) START_3 , END OF STATIC BUFFER
  * Lomem (F1) LOW_MEMORY , TRUE=LOW32K ON ; FALSE=LOW32K OFF
  * Smplp (D2) SAMPLE_POINTER , FOR SAMPLES IN RAM
  * Blockp (D2) BLOCK_POINTER, OF SYNTHESIZED WINDOW IN RAM
  * Runfs1 (D2) SCRATCHPAD_1 , FOR MISCELLANEOUS CALCULATIONS
  * Runfs2 (D2) SCRATCHPAD_2 , FOR MISCELLANEOUS CALCULATIONS
  * Ovlapf (D1) OVERLAP_FACTOR, SHORT TIME FFT ANLS-SYNTHS PARAM.
  * Vrsion (D1) ANALYSIS - SYNTHESIS VERSION ID.
  * Partbl (D1) (D1X16) PARAMETER_TABLE , WHERE GLOBAL DATA BASE IS COPIED
  * Magvrs (D1) MAGNITUDE VERSION
2.GLOBALS

- STRTD (D3X256X2) STATIC TABLE, 1.5K
- TB1 (D3X256) TABLE_1, SCRATCHPAD BUFFER 1 (SP1)
- TB2 (D3X256) TABLE_2, SCRATCHPAD BUFFER 2 (SP2)
- TB3 (D3X256) TABLE_2, SCRATCHPAD BUFFER 2 (SP2)

2.GLOBALS

- TRUE ($00)
- FALSE ($00)
- STPLAC ($0000) STARTING_PLACE, FOR THE SAMPLE POINTER IN LO32K00138000
- S1CHNG ($00) $1_CHANGE,OPCODE NEEDED TO AFFECT $1_LINE IN I/0001390000
- ZEROEX ($00) ZERO_EXPONENT, EXPONENT OF F.P.N. "0"
- ZERONM ($0000) F.P.N. _ZERO
- MXMNEX ($81) MAXIMUM_NEGATIVE _EXponent
- MNPLNM ($0000) MINIMUM_POSITIVE_F.P.N.

INITIALIZATION: PASS FLAG=TRUE ON USER STACK

XDEF RUNFFT

XREF CPARAM, WRTLAT, TRANSF, WNDMLT, COMPRS, FFT, IFFT
XREF ADDING, TRBACK, TRUE, FALSE, STPLAC, S1CHNG, ZEROEX, ZERONM
XREF MXMNEX, MNPLNM, FNDMAG, COPY, WTCDOG, CLICK, BIJREV, WRTLED
XREF TB1, TB2, STRTD, REXPND, IEXPND, COMP, FNDMG1, FNDMG2

DSCT

AVGVR5 RMB 1
REWS RMB 1
OSTWS RMB 1
RATE RMB 1
W RMB 2
ISPV RMB 1
ISBW RMB 1
THR1 RMB 2
FILE: RUNFFT  6809  Q  COMBUSTION ENGINEERING, INC.

RUNFFT EQU *
PULU A                   READ INITIALIZATION FLAG
CMPA TRUE                INITIALIZE ?
BEQ RUNFDD               YES
*
* ERROR
*
PULU A                   PULL DATA AND THROW AWAY
PULU A                   PULL DATA AND THROW AWAY
JSR CLICK                GENERATE AN AUDIO BEEP
LBRA RUNF99
*
* COPY PARAMETERS
*
RUNFDD LDX #PARTBL       SEND ADDRESS WHERE TO COPY
PSHU X                   TELL HOW MANY TO COPY
LDA #HOWMNY              SEND READ FLAG
PSHU A                   COPY PARAMETERS
LDA .TRUE                GET ACCESS OF PARAMETER TABLE
PSHU A
JSR CPARAM
LDX #PARTBL
*
* READ AND STORE PARAMETERS OUT OF THE TABLE NECESSARY TO RUN
* FAST FOURIER ANALYSIS LOOP.
*
LDA .WST,X               READ FILTER TYPE
STA POSTWS               SAVE POST-FILTER FUNCTION.
ANDA #$0F                SAVE PRE-FILTER FUNCTION.
STA POSTWS
LDA POSTWS
LSRA
LSRA
LSRA
LSRA
STA POSTWS  
LDA RATED,X
STA RATE
LDD NWT,X
STD NW
LDA NSPWT,X
STA NSPW
LDA NSBWT,X
STA NSBW
LDA OVLPFT,X
STA OVLAPF
LDA VRSONT,X
ANDA #$0F
STA VRSION
LDA VRSONT,X
ANDA #$F0
STA MAGVRS

READ...  
...AND STORE SAMPLING RATE IN USECS.
READ HOW MANY SECTIONS TO PROCESS...
...AND STORE THIS NUMBER.
READ...
...AND STORE NUMBER OF SAMPLES/SECT.
READ NUMBER OF SAMPLES BETWEEN SECTIONS.
READ AND STORE...
...NUMBER OF SAMPLES BETWEEN SECTIONS.
READ AND STORE...
...OVERLAP FACTOR ATTENUATOR
READ AND STORE...
...ANALYSIS - SYNTHESIS VERSION ID.
READ AND STORE...
...ANALYSIS - SYNTHESIS VERSION ID.
READ AND STORE...

LDD .#STRTO
STD STRT1
LDB NSPW

INITIALIZE START OF THE STATIC BUFFER.
CALCULATE MIDDLE OF STATIC BUFFER
MIDDLE = START + 3 X NUM. _SAMPLES/WINDOW

CLRA
ADD .#1
STD RUNFS1
ADD RUNFS1
ADD RUNFS1
ADD STRT1
STD .STRT2
ADD RUNFS1
ADD RUNFS1

ADD START OF ABLE
THIS IS A MIDDLE OF BUFFER
DETERMINE END OF STATIC BUFFER
END = START + 2 X MIDDLE_OF_BUFFER
FILE: RUNFFT 6809 Q COMBUSTION ENGINEERING, INC.

ADDD RUNFS1
STO STRT3

THIS IS THE END OF STATIC BUFFER

NOW ENABLE LOW_32K SECTION OF THE RAM

LDA FALSE
PSHU A
LDA S1CHNG
PSHU A
JSR WRTLAT
LDA TRUE
STA LOMEM

FALSE TURNS ON LOW32K, TRUE HIGH32K.
RAM CONTROL IS ON S1 LINE
ENABLE SECTION OF RAM
REMEMBER THAT LOW32K IS ENABLED.

TURN AN LED INDICATOR ON

LDA TRUE
PSHU A
LDA #LEDMSK
PSHU A
JSR WRTLED

TRUE TURNS LED ON
DEFINE LED LOCATION
TURN LED ON

NOW CLEAR STATIC BUFFER WITH FLOATING POINT ZEROES.

LDX .#STRTO
LDA ZEROEX
LDY ZERONM
RUNFD05 STA .X+
STY .X++
CMPX .STRT3
BNE RUNFD05

READ START OF STATIC BUFFER
GET EXPONENT FOR ZERO
GET F.P.N. REPRESENTATION OF ZERO.
STORE A ZERO_EXPONENT
STORE A F.P.N. ZERO
IS STATIC BUFFER "ZEROED"
NO — REPEAT PROCEDURE.

INITIALIZE ("READ") POINTER FOR TAKING SAMPLES OUT OF RAM.
INITIALIZE ("BLOCK") POINTER FOR TAKING BLOCKS OF SAMPLES OUT OF RAM.

LDX .#STPLAC
.SIX SMPLP
.STX BLOCKP

THIS IS THE STARTING PLACE
THIS IS THE READ POINTER
THIS IS THE BLOCK POINTER

INITIALIZATION IS COMPLETE. PROGRAM STARTS HERE.

* INITIALIZE ("READ") POINTER FOR TAKING SAMPLES OUT OF RAM.
* INITIALIZE ("BLOCK") POINTER FOR TAKING BLOCKS OF SAMPLES OUT OF RAM.

LDX .STPLAC
.SIX SMPLP
.STX BLOCKP

THIS IS THE STARTING PLACE
THIS IS THE READ POINTER
THIS IS THE BLOCK POINTER

* INITIALIZATION IS COMPLETE. PROGRAM STARTS HERE.

* INITIALIZATION IS COMPLETE. PROGRAM STARTS HERE.

RJNFT0 EQU *

* REFRESH WATCHDOG TIMER
* JSR WTCDOG

* TRANSFER A SECTION WORTH OF DATA FROM RAM STORAGE.

LDX SMPLP
LDA NSPW
LDY #TB1
PSHU Y
PSHU A
PSHU X
JSR TRANSF

SEND WORKING BUFFER ADDRESS
SEND_NUM_SMPLS/WINDOW-1
SEND_READ_POINTER
TRANSFER SAMPLES FROM RAM
MULTIPLY BY PRE-FILTER FUNCTION
LDX  #TB1
LDA  NSPW
LDB  PPREWS
PSHU B     SEND FUNCTION_SHAPE_TYPE
PSHU A     SEND NUMBER_OF_SAMPLES-1
PSHU X     SEND ADDRESS OF WORKING SPACE
JSR  WNDMLT  MULTIPLY BY PRE-FILTER FUNCTION
BIT REVERSE POSITIONS OF CONTENTS OF THE TABLE
RUNFFT1 LDX  #TB2
LDA  NSPW
LDY  #TB1
PSHU A     SEND_NUM_SMPLS/WINDOW-1
PSHU X     SEND TO_TABLE ADDRESS
PSHU Y     SEND FROM_TABLE ADDRESS
JSR  BITREV  TRANSFER SAMPLES FROM RAM
EXPAND REAL DATA INTO COMPLEX DATA BY FORCING THE IMAGINARY PART TO 0.
LDX  #TB2
LDA  NSPW
LDY  #TB1
PSHU A     SEND_NUM_OF_SAMPLES/SECTION-1
PSHU Y     SEND ADDRESS OF EXPANSION_TABLE
PSHU X     SEND ADDRESS OF TABLE WHERE DATA IS.
JSR  REXPND  EXPAND REAL INTO COMPLEX TABLE.
NOW RUN A FAST-FOURIER-TRANSFORM (FFT) ALGORITHM
LDX  #TB1
LDA  NSPW
PSHU A     SEND NUMBER_OF_SAMPLES/WINDOW-1
PSHU X     SEND ADDRESS .OF TABLE WITH SAMPLES
JSR  FFT  RUN AN FFT ALGORITHM
CLICK AUDIO TRANSCLUDER
JSR  CLICK
FIND THE MAGNITUDE OF COMPLEX FREQUENCY SPECTRUM.
LDA  MAGVRS
CPA  #$20
BNE  RUNFD
LDX  #TB1
LDA  NSPW
PSHU X     SEND ADDRESS OF MAGNITUDE TABLE
PSHU A     SEND NUMBER_OF_SAMPLES/WINDOW-1
PSHU X     SEND ADDRESS OF TABLE WITH SAMPLES
JSR  FNDMG2  FIND MAGNITUDE OF FREQUENCY SPECTRUM.
FILE: RUNFFT 6809 0 COMBUSTION ENGINEERING, INC.

BRA RUNFN9
RUNFN0 LDA MAGVRS
CMPA #$10
BNE RUNFN1
LDX #TB1
LDA NSPW
PSHU X
SEND ADDRESS OF MAGNITUDE TABLE
PSHU A
SEND NUMBER_OF_SAMPLES/WINDOW-1
PSHU X
SEND ADDRESS OF TABLE WITH SAMPLES
JSR FDNGM1
FIND MAGNITUDE OF FREQUENCY SPECTRUM
BRA RUNFN9
RUNFN1 LDA MAGVRS
CMPA #0
* BNE RUNFN2
LDX #TB1
LDA NSPW
PSHU X
SEND ADDRESS OF MAGNITUDE TABLE
PSHU A
SEND NUMBER_OF_SAMPLES/SECTION-1
PSHU X
SEND ADDRESS OF TABLE WITH SAMPLES
JSR FDNGMAG
FIND MAGNITUDE OF FREQUENCY SPECTRUM
BRA RUNFN9
RUNFN2 NOP
RUNFN9 LDA VERSION
CMPA #1
BNE RUNF25
VERSION 1?
LDB NSPW
CLRA
ADDD #1
LDX #TB1
LEAX D,X
LEAX D,X
LEAX D,X
STX RUNFS1
LDX #TB1
LSRA
RORB
LEAX D,X
LEAX D,X
LEAX D,X
RUNF23 LDA +X
LDY +X++
PSHU Y
PSHU A
PSHS X
JSR COMP
PULS X
LEAX -3,X
FILE: RUNFFT  6809  Q  COMBUSTION ENGINEERING, INC.

```
PUU  A
STA  X
PUU  D
STD  X
CMPX RUNFS1
BNE  RUNF23

BIT REVERSE POSITIONS OF CONTENTS OF THE TABLE

RUNF  LDX  #TB2
      LDA  NSPW
      LDY  #TB1
      PSHU  A  SEND_NUM_SMPLS/WINDOW-1
      PSHU  X  SEND TO_TABLE ADDRESS
      PSHU  Y  SEND FROM_TABLE ADDRESS
      JSR  BITREV  TRANSFER SAMPLES FROM RAM

EXPAND REAL DATA INTO COMPLEX DATA BY FORCING THE IMAGINARY PART TO 0.

LDX  #TB2
LDA  NSPW
LDY  #TB1
PSHU  A  SEND_NUM_OF_SAMPLES/SECTION-1
PSHU  Y  SEND ADDRESS OF EXPANSION TABLE
PSHU  X  SEND ADDRESS OF TABLE WHERE DATA IS.
LDA  VRSION  CHECK VERSION
CMPA  #1  EXPAND REALS?
BEQ  RUNFF5  NO.
CMPA  #0
BNE  RUNFFX  DUMMY COMPARE
JSR  REXPND  YES, EXPAND REAL INTO COMPLEX TABLE.
BRA  RUNFF6  CONTINUE

RUNFF  JSR  IEXPND  EXPAND IMAGINARY INTO COMPLEX TABLE

RUNFF6 EQU *
*  NOW RUN AN INVERSE-FAST-FOURIER-TRANSFORM (IFFT) ALGORITHM

LDX  #TB1
LDA  NSPW
PSHU  A  SEND NUMBER_OF_SAMPLES/WINDOW-1
PSHU  X  SEND ADDRESS OF TABLE WITH SAMPLES
JSR  IFFT  RUN AN IFFT ALGORITHM

* COMPRESS IMAGINARY DATA TABLE BY DELETING THE IMAGINARY PART.

LDA  VRSION
CMPA  #1
BNE  RUNFF/  
LDB  NSPW
CLRA  
ADDD  #1
```
KILE* RUNFFT 6809 O COMBUSTION ENGINEERING, INC.

PSHU D  
LDX #TB1  SEND NUMBER OF BYTES TO COPY  00496000
PSHU #TB1  00497000
PSHU X  SEND "TO" POINTER  00498000
LEAX 3,X  00499000
PSHU X  SEND "FROM" POINTER  00500000
JSR COPY  00501000

JNFF LDX #TB1  00502000
LDA NSPW  00503000
PSHU A  SEND NUM_OF_SAMPLES/WINDOW-1  00504000
PSHU X  SEND ADDRESS OF TABLE WHERE DATA IS.  00505000
JSR COMPR3  00506000
PSHU D  00507000

MULTIPLY THE BY SOME POST-FILTERING FUNCTION

LDX #TB1  00508000
LDA NSPW  00509000
LDB POSTWS  00510000
PSHU B  00511000
PSHU A  SEND NUMBER_OF_SAMPLES_-1  00512000
PSHU X  SEND ADDRESS OF WORKING SPACE  00513000
JSR WNDMLT  00514000

ADD THE SYNTHESIS SECTION JUST GENERATED TO THE STATIC TABLE

RUNF35 LDA NSPW  00515000
INCA  00516000
LDX .STRT1  00517000
LDY #TB1  00518000
READ NUM_OF_SAMPLES/SECTION-1  00519000
...CONTENTS ARE NOW ADDED TO THE  00520000
...STATIC WINDOW, STARTING AT THE  00521000
...BEGINING OF THE DYNAMIC TABLE.  00522000

RUNF40 PSHS A  00523000
LDA ,X+  00524000
READ EXPONENT FROM DYNAMIC TABLE  00525000
PSHS A  00526000
SAVE IT  00527000
LDD ,X++  00528000
READ F.P.N. FROM DYNAMIC TABLE  00529000
PSHU D  00530000
SEND FOR ADDITION  00531000
PULS A  00532000
RECOVER EXPONENT  00533000
PSHU A  00534000
SEND IT FOR ADDITION, TOO.  00535000

LDA ,Y+  00536000
READ EXPONENT FROM STATIC TABLE  00537000
PSHS A  00538000
SAVE IT  00539000
LDD ,Y++  00540000
READ F.P.N. FROM STATIC TABLE  00541000
PSHU D  00542000
SEND FOR ADDITION  00543000
PULS A  00544000
RECOVER EXPONENT  00545000
PSHU A  00546000
SEND IT FOR ADDITION, TOO.  00547000

PSHS X  00548000
SAVE DYNAMIC TABLE POINTER  00549000
PSHS Y  00550000
SAVE STATIC TABLE POINTER
JSR ADDING  00551000
ADD A MEMBER OF STATIC TABLE...
...TO A MEMBER OF THE DYNAMIC TABLE.  00552000

JSR WTCDOG  00553000
PULS Y  00554000
RECOVER STATIC TABLE POINTER  00555000
PULS X  00556000
RECOVER DYNAMIC TABLE POINTER  00557000
LEAX -3,X  REPOSITION STATIC TABLE POINTER 00551000
PULU A  READ EXPONENT OF SUM 00552000
STA ,X+  STORE IN STATIC TABLE 00553000
PULU D  READ F.P.N. SUM 00554000
STD ,X++  STORE IN STATIC TABLE 00555000
PULS A  RECOVER LOOP COUNTER 00556000
DECA  DYNAMIC TO STATIC ADDITION COMPLETE? 00557000
LBNE RUNF40  NO 00558000
CMPX .STRT3  DID START OF DYNAMIC BUFFER REACH ...
*  THE MIDDLE OF STATIC BUFFER? 00559000
*  LBMI RUNF70  NO 00560000
*  DIVIDE FIRST HALF OF THE STATIC TABLE BY THE OVERLAP FACTOR.
*  NOTE THAT THE DIVISION WILL ONLY AFFECT EXPONENT OF THE F.P.N SINCE 00561000
*  THE OVERLAP FACTOR IS A POWER-OF-TWO EXPONENT.
*  LDX #STRTO  GET START OF STATIC TABLE 00562000
LDA NSPW  GET NUM_OF_SAMPLES/SECTION-1 00563000
INCA  USE IT AS A COUNTER 00564000
RUNF42 LDR ,X  READ EXPONENT. POSITIVE? 00565000
BPL RUNF45  YES. 00566000
SUBR OVLAPF  SUBTRACT OVERLAP FACTOR FROM EXPONENT00567000
BMI RUNF50  OVERFLOW? NO. 00568000
LDB MXMNX  STORE A MAXIMUM NEGATIVE EXPONENT 00569000
,STB ,X+  ...AND A MINIMUM POSITIVE F.P.N. 00570000
LDY MNPLNM  00571000
,STY ,X  00572000
LEAX -1,X  00573000
BRA RUNF50  00574000
RUNF45 SUBB OVLAPF  00575000
RUNF50 STB ,X  00576000
LEAX 3,X  MOVE TO NEXT ENTRY IN STATIC .TABLE. 00577000
DECA  ANY MORE ENTRIES IN THE TABLE? 00578000
BNE RUNF42  00579000
*  NOW COPY FIRST HALF OF THE STATIC TABLE INTO SOME SCRATCHPAD. THIS 00580000
*  IS THE SYNTHESIZED SPEECH WINDOW. 00581000
*  LDX #STRTO  GET START OF THE STATIC TABLE 00582000
LDY #TB1  THIS IS THE SCRATCHPAD ADDRESS 00583000
LDB NSPW  USING NUM_SAMPLES/SECTION-1
CLRA  ...CALCULATE NUMBER OF BYTES TO COPY 00584000
ADDD #0001  00585000
STD RUNFS1  00586000
ADDD RUNFS1  00587000
ADDD RUNFS1  00588000
PSHU D  SEND NUMBER OF BYTES TO COPY 00589000
PSHU Y  SEND SCRATCHPAD ("TO") ADDRESS 00590000
PSHU X  SEND STATIC TABLE ("FROM") ADDRESS 00591000
JSR COPY  COPY SYNTHESIZED SPEECH WINDOW 00592000
* NOW COPY A BLOCK FROM UPPER HALF OF STATIC TABLE TO LOWER HALF OF THE STATIC TABLE.

LDX #STRTO  GET START OF THE STATIC TABLE
LDY STRT2  THIS IS THE MIDDLE OF STATIC TABLE USING NUM_SAMPLES/SECTION-1...
LDB NSPW  ...CALCULATE NUMBER OF BYTES TO COPY
ADDD #ODD1  STD RUNFS1
ADDD RUNFS1  ADDD RUNFS1
PSHU D  SEND NUMBER OF BYTES TO COPY
PSHU X  SEND STATIC TABLE ("To") ADDRESS
PSHU Y  SEND MID-STATIC TABLE ("From") ADDRESS
JSR COPY  COPY SYNTHESIZED SPEECH WINDOW

* NOW STORE F.P.N. ZEROS INTO THE UPPER HALF OF THE STATIC BUFFER.

LDX .STRT2  GET THE MIDDLE OF STATIC TABLE
LDA NSPW  GET NUM_SAMPLES/SECTION-1
INCA  THIS WILL BE THE COUNTER
LDB ZEROEX  THIS IS THE EXPONENT FOR ZERO
LDY ZERONM  THIS IS A F.P.N. ZERO
RUNFS55 STB ,X+  STORE A ZERO EXPONENT
SIY ,X++  STORE A ZERO F.P.N.
DECA  ALL LOCATIONS ZEREOED?
BNE RUNFS55 NO.

* REINITIALIZE START OF THE DYNAMIC TABLE

LDB NSBW  GET NUMBER_OF_SAMPLES_BETW_WINDOWS
CLRA  MAKE IT TWO BYTES LONG
ADDD #1  STD RUNFS1
ADDD RUNFS1  ADDD RUNFS1
ADDD RUNFS1  ADDD #STRTO
STD .STRT1

* TRANSFER SYNTHESIZED SPEECH WINDOW TO RAM STORAGE

LDD BLOCKP  READ BLOCK POINTER
SUBD SMPLP  SUBTRACT SAMPLE POINTER
BLT RUNFS60  ARE POINTERS IN OPPOSITE 32K BLOCKS?

* REENABLE TEMPORARILY LOW 32K BLOCK.

LDA FALSE  SET LOW32K_FLAG TO FALSE
PSHU A  RAM CONTROL IS ON S1 LINE
LDA SICHNG  ENABLE SECTION OF RAM
PSHU A
JSR WRTLAT
RUNFS60 LDX #TB1
LDY BLOCKP
LDA NSPW
PSHU Y SEND BLOCK ("TO") POINTER IN RAM
PSHU A SEND NUM_OF_SAMPLES/SECTION-1
PSHU X SEND SCRATCHPAD ("FROM") POINTER
JSR TRBACK TRANSFER SYNTHESIZED SPEECH WINDOW

* CLICK AUDIO TRANSDUCER

* CHECK IF UPPER 32K NEEDS REENABLING

* LDD BLOCKP READ BLOCK POINTER
  SUBD SMPLP SUBTRACT SAMPLE POINTER
  BLT RUNF65 ARE POINTERS IN OPPOSITE 32K BLOCKS?
  NO.

* REENABLE HIGH 32K BLOCK.

* LDA .TRUE. SET LOW32K_FLAG TO TRUE
  PSHU A
  LDA SICHING RAM CONTROL IS ON S1 LINE
  PSHU A
  JSR WRTLAI ENABLE SECTION OF RAM

* DETERMINE IF ANY MORE WINDOWS NEED TO GO THRU ANALYSIS-SYNTHESIS LOOP

RUNF65 LDD NW DECREMENT NUMBER OF SECTIONS TO DO
  SUBD #1 ANY MORE? NO.

* GO TO THE NEXT BLOCK OF WINDOWS, UPDATE BLOCK POINTER

  LDB NSPW READ NUM_SAMPLES/SECTION-1
  CLRA
  ADDD #1
  STD RUNFS1 CALCULATE BLOCK POINTER ADDRESS
  ADDD RUNFS1
  ADDD BLOCKP ADD TO THE OLD POINTER
  STD BLOCKP THIS IS THE NEW BLOCK POINTER
  BPL RUNF68 CROSSED 32K SECTION OF RAM? NO.

* INITIALIZE BLOCK POINTER

  LDD STPLAC
  STD BLOCKP INIT BLOCK POINTER

00661000 00662000
00663000 00664000
00665000 00666000
00667000 00668000
00669000 0066A000
0066B000 0066C000
0066D000 0066E000
0066F000 00670000
00671000 00672000
00673000 00674000
00675000 00676000
00677000 00678000
00679000 0067A000
0067B000 0067C000
0067D000 0067E000
0067F000 00680000
00681000 00682000
00683000 00684000
00685000 00686000
00687000 00688000
00689000 0068A000
0068B000 0068C000
0068D000 0068E000
0068F000 00690000
00691000 00692000
00693000 00694000
00695000 00696000
00697000 00698000
00699000 0069A000
0069B000 0069C000
0069D000 0069E000
0069F000 006A0000
006A1000 006A2000
006A3000 006A4000
006A5000 006A6000
006A7000 006A8000
006A9000 006AA000
006AB000 006AC000
006AD000 006AE000
006AF000 006B0000
006B1000 006B2000
006B3000 006B4000
006B5000 006B6000
006B7000 006B8000
006B9000 006BA000
006BB000 006BC000
006BD000 006BE000
006BF000 006C0000
006C1000 006C2000
006C3000 006C4000
006C5000 006C6000
006C7000 006C8000
006C9000 006CA000
006CB000 006CC000
006CD000 006CE000
006CF000 006D0000
006D1000 006D2000
006D3000 006D4000
006D5000 006D6000
006D7000 006D8000
006D9000 006DA000
006DB000 006DC000
006DD000 006DE000
006DF000 006E0000
006E1000 006E2000
006E3000 006E4000
006E5000 006E6000
006E7000 006E8000
006E9000 006EA000
006EB000 006EC000
006ED000 006EE000
006EF000 006F0000
006F1000 006F2000
006F3000 006F4000
006F5000 006F6000
006F7000 006F8000
006F9000 006FA000
006FB000 006FC000
006FD000 006FE000
006FF000 00700000
00701000 00702000
00703000 00704000
00705000 00706000
00707000 00708000
00709000 0070A000
0070B000 0070C000
0070D000 0070E000
0070F000 00710000
00711000 00712000
00713000 00714000
00715000
**UPDATE THE SAMPLE POINTER**

* RUNFFT
  * LDR  NSBW
  * CLRA
  * ADDD  #1
  * STD  RUNF$1
  * ADDD  RUNF$1
  * ADDD  SMPLP
  * STD  SMPLP
  * LDD  NW
  * SUBD  #1
  * STD  NW

* CHECK IF SAMPLE POINTER HAS CROSSED 32K BOUNDARY

* LDD  SMPLP
  * LBPL  RUNF10
  * BRA  RUNF80

* BLOCK IS INCOMPLETE. ANALYSE-SYNTHESIZE ANOTHER WINDOW.

* RUNFFT
  * LDB  NSBW
  * CLRA
  * ADDD  #1
  * STD  RUNF$1
  * ADDD  RUNF$1
  * ADDD  SMPLP
  * STD  SMPLP
  * LDD  RUNF$1
  * ADDD  RUNF$1
  * ADDD  RUNF$1
  * ADDD  STRT1
  * STD  STRT1
  * LDD  NW
  * SUBD  #1
  * STD  NW

* CHECK IF SAMPLE POINTER HAS CROSSED 32K BOUNDARY

* LDD  SMPLP
  * LBPL  RUNF10

* 32K SECTION OF RAM IS FILLED. ENABLE UPPER 32K IF EMPTY, ELSE END.

* RUNFFT
  * LDA  . TRUE
  * CMPA  LOMEM
  * BNE  RUNF99
  * LDA  FALSE
  * STA  LOMEM
  * COMA
  * PSHU  A
LDA S1CHNG       RAM CONTROL IS ON S1 LINE
PSHU A           ENABLE SECTION OF RAM
JSR WRTRLAT      INITIALIZE BLOCK POINTER

LDD STPLAC
STU SMP LP
LBR A RUNF10     RETURN BACK FOR ANOTHER ANALYSIS...
                 ...SYNTHESIS LOOP.
UNF99 EQU *      

NOW REENABLE LOW_32K SECTION OF THE RAM

LDA FALSE        FALSE TURNS ON LOW32K, TRUE HIGH 32K.
PSHU A
LDA S1CHNG       RAM CONTROL IS ON S1 LINE
PSHU A
JSR WRTRLAT      ENABLE SECTION OF RAM
LDA TRUE         REMEMBER THAT LOW32K IS ENABLED.
STA LOMEM

TURN AN LED INDICATOR OFF

LDA FALSE        FALSE TURNS LED OFF
PSHU A
LDA #LEDMSK      DEFINE LED LOCATION
PSHU A
JSR WRTRLED      TURN LED OFF
RTS
END
REFERENCES


(18) E.Yourdon and L.L.Constantine, "Structured Design", Yourdon,Inc, pp. 3-57 and 76-126, 1978