SIMULATION AND IMPLEMENTATION OF A LINEAR PREDICTIVE CODER

By

D. S. F. CHAN

A Master's Thesis

Submitted in partial fulfilment of the requirements

for the award of

Master of Philosophy

of the Loughborough University of Technology

Dec 85

010024/01

S Y N O P S I S

SIMULATION AND IMPLEMENTATION OF A LINEAR PREDICTIVE CODER

The main objective of this research was to design and build a Linear Predictive Coder (LPC) based on the TMS320 processor, and to incorporate this in the design of a low bit rate voice coding server for a Cambridge Ring. In order to decide on a suitable algorithm for the LPC, extensive simulations were carried out on a BBC computer. The computer used was interfaced to a frame store which, although its original purpose was to store video information, acted as a suitable store for speech. Up to six seconds of speech could be fed in from a microphone in real time for analysis. The BBC was fitted with a second processor, but in spite of this the processing times were very slow. However after complete processing, i.e. analysis and synthesis, the reconstituted speech could be read out from the frame store in real time to a loudspeaker or headphones in order to judge the quality. After deciding on a suitable algorithm for the LPC the program was translated into TMS320 assembly code so that one TMS320 was responsible for analysis and one for synthesis. Two sets of TMS320 development boards were used in this real time implementation experiment so that substantial hardware development could be minimized. Parallel data lines and interrupt technique were used for parameters transfer from the analyser to the synthesiser and speech input and output were through two analogue/digital boards. The performance of the coder was assessed by informal subjective listening tests.

Limitations of the TMS320 processor in implementing LPC are discussed and the design of the voice coding server for the Cambridge Ring based on this research is outlined.

ACKNOWLEDGEMENT

I would like to thank my supervisor, Professor J.W.R. Griffiths for his help and assistance throughout the course of the research program and also for his effort to obtain the studentship for me.

Special thanks also to Dr.P.J.Patrick for his valuable assistance and helpful discussions.

Finally, I wish to express my deepest gratitude to my parents for their moral and financial support through the course of my education.

LIST OF PRINCIPAL SYMBOLS AND ABBREVIATIONS

ACF	-	Autocorrelation Function
ADC	-	Analog to Digital Converter
AIB	-	Analog Interface Board
A_k	-	Cross-sectional area of the k^{th} tube of a lossless tubes model
A(x,t)	-	"Area Function" of an acoustic tube at position x and time t.
A(z)	-	Inverse filter transfer function
CORR	-	Correction coefficient of an interpolator
c	-	Velocity of sound in an acoustic tube
DAC	-	Digital to Analog Converter
E_n	-	Short time average prediction error
EVM	-	Evaluation Module
e(n)	-	Prediction error
f_s	-	Sampling frequency
G	-	LPC parameter for gain
G(z)	-	Glottal pulse model transfer function
g(n)	-	Synthetic glottal pulse wave
H(z)	-	Vocal system transfer function
K(i)	-	LPC parameter for the i^{th} reflection coefficient
k_i	-	The i^{th} PARCOR coefficient
LPC	-	Linear Predictive Coding/Coder
ℓ	-	Overall length of a human vocal tract
NOISE	-	Random noise generator output

OS - Mean value of a speech segment

P_o - Pitch period of a speech segment

P_I - Pitch detector output

PITCH - LPC parameter for pitch period

P(Z) - Z-transform of p(x,t)

p - Order of a linear predictor

p(x,t) - Sound pressure in an acoustic tube at position x and time t

R(i) - Autocorrelation function coefficient at i^{th} sample lag.

R(Z) - Radiation model transfer function

r_k - Reflection coefficient of the k^{th} tube of a lossless tubes model

SIFT - Simplified Inverse Filter Tracking algorithm

s(n) - Speech signal

$\tilde{s}(n)$ - Predicted speech signal

T - Sampling period

THRE - Threshold value for centre-clipping

T[ ] - Centre-clipping transformation

T'[ ] - 3 -level centre-clipping transformation

U (Z) - Z-transform of u(x,t)

u(x,t) - Volume velocity flow in an acoustic tube at position x and time t.

$u_k^+(t)$ - Positive going travelling wave in the k^{th} tube of a lossless tubes model

$u_k^-(t)$ - Negative going travelling wave in the k^{th} tube of a lossless tubes model

V(Z) - Vocal tract model transfer function

V/UV - Voiced/Unvoiced

W_n - A 220 points Hanning window

$w(n)$ -	A finite window
$X(n), X_n$ -	Sampled speech signal for LPC analysis
$Z_L(s)$ -	Radiation impedance at the lips
α_k -	The k^{th} prediction coefficient of a linear predictor
β -	A scaling constant
ρ -	Density of air in an acoustic tube
μ -	Pre-emphasis coefficient

C O N T E N T

Page

CHAPTER ONE - INTRODUCTION

1.1 INTRODUCTION

The material contained in this thesis relates to the development of a real-time Linear Predictive Speech Coder based on the Texas TMS32010 signal processor. This work is part of the voice communication experiment of the UNIVERSE (UNIVersities Extended Ring and Satellite Experiment) Project. In the next section, the nature of Project UNIVERSE and how the work presented here relates to it will be briefly discussed. Finally, in section 1.3, the organization of this thesis is outlined.

1.2 MOTIVATION OF THE WORK

The object of Project UNIVERSE was to investigate the facilities which can be developed for allowing business communication over a concatenation of terrestrial and small dish satellite networks (1). UNIVERSE had seven participating organizations, three from British industry and four academic groups. They were GEC-Marconi Research Laboratories, Logica Ltd., British Telecom., Cambridge University, Loughborough University, Rutherford Appleton Laboratory (RAL) and University College London (UCL).

In order to carry out the investigation experimentally a number of small earth stations were sited in most of the participants' premises. These stations can communicate at 1 Mbps via the OTS (Orbital Test Satellite) Research Satellite as shown in Fig.1.1. At each site there are one or more Cambridge Rings, capable of a local user data bandwidth of 4 Mbps. The Rings are connected to various service hosts, local servers, computers driving the earth stations and computers containing gateways to other networks. To complement the network, a number of application experiments have been developed. These experiments include:

1) The development of the Distributed Operation System, the Universe Support Environment (USE) and the Distributed File (DF) System. These packages permit remote file handling and transfer and the remote use of software support facilities.

2) The development of a set of distributed network support facilities including General Purpose Server and Data Encryption.

3) The business communication experiments. These include communication facilities over the network using distributed Teletex, Videotex, Packet Voice and Image Transfer.

Voice transmission has been included in the UNIVERSE network for three reasons, i.e.

1) To provide a "talk-back" facility to assist in the development of other experiments. It is extremely convenient, for example, to be able to pick up a telephone and talk over the same network to the location of an equipment failure.

2) The experience of real-time service operation is required to satisfy the need to test the network with such services. The lessons learned will be of great assistance in designing network operation with any real-time services, e.g. process control.

3) Speech service is a very visible demonstrator of the capabilities (and some of the limits) of the network.

Existing voice stations are in the form of standard telephones connected to a special codec board designed by the Marconi Research Centre. It makes use of the AMI S3506 codec chip and is configurated to provide two full duplex circuits for the UNIVERSE network. The codec provides a standard 64 Kbit/s PCM speech data stream. The codec board accesses the Cambridge Ring using a UNIVERSE Z80 "small server" which inserts the data stream into "Basic Block" packets for transmission over the network, and subsequently these packets are stored in the voice server or passed to the remote telephone for replay. The 1 Mbit/s satellite is capable of transmitting a total of only about 15 duplex 64 Kbit/s speech circuits simultaneously even if there is no other communication in progress. This is quite small and there is therefore considerable interest in the use of data compression speech encoding systems. It was suggested that Linear Predictive Coding should be the first data compression scheme to be experimented with. This is because LPC is a known practical data compression algorithm and theoretically it can reduce the information rate of speech down to as low as 2.4 Kbit/s. Once LPC can be implemented on the Cambridge Ring, other less complicated data compression schemes such as Transform Coding or Sub-band Coding could then be experimented with using the same system.

The remaining chapters of this thesis describe the development of the LPC analysis and synthesis algorithms. This includes simulation and implementation of the algorithms using a BBC computer and TMS32010 processors respectively. The performance of the LPC coder was judged both in simulation and real-time implementation under "minimum error situation" (i.e. no transmission errors and using unquantized parameters for synthesis).

1.3 ORGANIZATION OF THE THESIS

Following this introductory Chapter, Chapter Two describes digital models for speech signals. Integrating these models together forms the basic configuration of an LPC synthesizer. Chapter Two also gives a brief introduction to Linear Prediction theory of speech signals and shows how Linear Prediction can be used to estimate the parameters needed for the LPC synthesizer. Chapter Three describes the development of the LPC simulation programs, namely the LPC Analysis program and the LPC Synthesis program. These programs define the analysis and synthesis algorithms which were implemented in real-time by TMS32010 processors. Chapter Four describes the transformation of the LPC simulation programs into TMS32010 assembly codes. Details of the TMS32010 software operations and the hardware involved in the implementation experiment are given. Finally results of informal subjective listening tests on the coder are discussed.

In Chapter Five, the limitations of the TMS32010 processor in implementing LPC algorithms are discussed and the original design of an LPC voice coding server for a Cambridge Ring based on this research is outlined.

CHAPTER TWO - LINEAR PREDICTIVE CODING SYSTEM FOR SPEECH SIGNALS

2.1 INTRODUCTION

This Chapter first examines the mechanism of human speech production. Digital models for speech signals are then described. These include a vocal tract model, a radiation model and a glottal excitation model. Integrating all these models together forms the basic configuration of an LPC synthesizer. The rest of this Chapter gives a brief introduction to linear predictive analysis of speech signals and shows how linear prediction can be used to estimate the reflection coefficients needed for the LPC synthesizer by comparing the all-pole model produced by linear predictive analysis and the transfer function of the vocal tract model. This also reveals the basic configuration of a linear predictive coding system for speech communication.

2.2 MECHANISM OF SPEECH PRODUCTION (2)

The schematic diagram of human speech production mechanism is shown in Fig.2.1. The vocal tract begins at the glottis and ends at the lips. In an adult male the vocal tract is about 17 cm. long. The cross-sectional area of the vocal tract determined by the positions of tongue, lips, jaw and velum varies from zero to 20 cm^2. When the velum is lowered the nasal tract is acoustically coupled to the vocal tract to produce the nasal sounds of speech.

Fig.2.2 shows the functional diagram of the vocal apparatus. The diagram also includes the sub-glottal system composed of the lungs, bronchi and trachea. This sub-glottal system serves as a source of energy for the production of speech. Speech sounds can be classified into three distinct classes according to their mode of operation. They are the voiced sounds, fricative or unvoiced sounds and plosive sounds. Voiced sounds are produced by forcing air through the glottis with the tension of the vocal cords adjusted so that they vibrate in a relaxation oscillation, thereby producing quasi-periodic pulses of air which excite the vocal tract. Fricative or unvoiced sounds are generated by forming a constriction at some point in the vocal tract and forcing air through the constriction at a high enough velocity to produce turbulence. This creates a broad spectrum noise source to excite the vocal tract. Plosive sounds result from making a complete closure (usually towards the mouth end), building up pressure behind the closure and suddenly releasing it.

The vocal tract and nasal tract are shown in Fig.2.2 as tubes of non-uniform cross-sectional area. As sound propagates down these tubes, the frequency spectrum is shaped by the frequency selectivity of the tubes. The resonance frequencies of the vocal tract tube are termed formant frequencies or simply formants. The formant frequencies depend upon the shape and dimensions of the vocal tract. Different sounds are formed by varying the shape of the vocal tract. Thus, the spectral properties of the speech signal vary with time as the vocal tract shape varies.

2.3 DIGITAL MODELS FOR SPEECH SIGNALS (3)

In order to obtain a practical model for speech production, the human vocal system is divided into three main parts. They are the vocal tract, the radiation at the lips and the glottal excitation. It is assumed that these three parts can be uncoupled from each other so that they can be modelled individually.

2.3.1 The Vocal Tract Model

It can be seen from Fig.2.2 that the vocal tract and the nasal tract can be modelled as tubes of non-uniform cross-sectional area. However, in order to obtain a useful vocal tract model, it is assumed that the effects of the nasal tract can be ignored. The vocal tract can then be modelled as a tube of non-uniform time varying cross-section as shown in Fig.2.3. With the further simplifying assumption that there are no losses inside the tube, Portnoff (4) has shown that the sound waves in the tube satisfy the following pair of equations

$$-\frac{\partial p}{\partial x} = \rho \frac{\partial (u/A)}{\partial t} \qquad (2.1a)$$

$$-\frac{\partial u}{\partial x} = \frac{1}{\rho c^2} \frac{\partial (pA)}{\partial t} + \frac{\partial A}{\partial t} \qquad (2.1b)$$

where

$p = p(x,t)$ is the variation in sound pressure in the tube at position x and time t

$u = u(x,t)$ is the variation in volume velocity flow at position x and time t.

ρ is the density of air in the tube

c is the velocity of sound

$A = A(x,t)$ is the "area function" of the tube; i.e. the value of cross-sectional area normal to the axis of the tube as a function of distance along the tube and as a function of time.

Closed form solutions to Eqs.(2.1) are not possible except for the simplest configuration. One approach to solve Eqs.(2.1) is to model the vocal tract as interconnected lossless acoustic tubes as shown in Fig.2.4. The cross-sectional areas A_k of the tubes are chosen so as to approximate the area function $A(x)$ of the vocal tract. If a large number of tubes of short length is used, it is reasonable to expect the resonant frequencies of the concatenated tubes to be close to those of a tube with continuously varying area function.

Solving Eqs.(2.1) for the k^{th} tube and applying continuity conditions at the junction between the k^{th} and (k+1)st tubes, it can be shown (3) that:

$$u^+_{k+1}(t) = (1+r_k)u^+_k(t-\tau_k) + r_k\, u^-_{k+1}(t) \qquad (2.2a)$$

$$u^-_k(t+\tau_k) = -r_k u^+_k(t-\tau_k) + (1-r_k)\, u^-_{k+1}(t) \qquad (2.2b)$$

where $\tau_k = {}^{\ell k}/_{c}$ is the time for a wave to travel the length of the k^{th} tube and u^+_k and u^-_k are positive and negative going travelling waves in the k^{th} tube. The quantity

$$r_k = \left[\frac{A_{k+1} - A_k}{A_{k+1} + A_k}\right] \qquad (2.3)$$

is called the reflection coefficient for the k^{th} junction. Since the areas are all positive, it can be shown that

$$-1 \le r_k \le 1 \qquad (2.4)$$

The signal flow graph representation of Eqs.(2.2) is shown in Fig.2.5. Hence an N-tube model as in Fig.2.4 would have N sets of forward and backward delays and N-1 junctions each characterized by a reflection coefficient.

Applying boundary conditions at the lips to the N^{th} tube of the system gives the output termination as shown in Fig.2.6, whereas applying boundary conditions at the glottis to the 1st tube of the system and assuming the glottal impedance is infinite gives the input termination as shown in Fig.2.7.

At the present stage,wave propagation in the human vocal tract can be represented by an N-tube model with flow graph as shown in Fig.2.8.

By further assuming that all tubes are of equal length, each delay in Fig.2.8 can then be set equal to

$$\tau = \ell/Nc \qquad (2.5)$$

where ℓ is the overall length of the vocal tract.

It can be shown (3) that if the input to the system (i.e. the excitation) is band limited to frequencies below $\pi/2\tau$, then we can sample the input with period $T = 2\tau$. Hence a discrete-time model for the vocal tract can be obtained by replacing each τ sec delay in Fig.2.8 by a ½ sample delay (since $\tau = T/2$) as shown in Fig.2.9. The half sample delays imply an interpolation half-way between sample values and this is very difficult to implement. A more practical configuration can be obtained by moving the delays in the upper branches to the corresponding branches directly below. Fig.2.10 shows the modified discrete-time system. The advantage of this form is that difference equations can be written for this system and these difference equations can be used iteratively to compute samples of the output from samples of the input.

By mathematical induction, it can be shown (3) that the transfer function of the discrete-time vocal tract model is of the form

$$V(Z) = \frac{U_L(Z)}{U_G(Z)}$$

$$= \frac{Z^{-N/2} \prod_{k=1}^{N} (1+r_k)}{D(Z)} \tag{2.6}$$

where D(Z) can be determined by the recursive formula

$$D_o(Z) = 1 \tag{2.7a}$$

$$D_k(Z) = D_{k-1}(Z) + r_k Z^{-k} D_{k-1}(Z^{-1}) \qquad k=1, 2, \ldots, N \tag{2.7b}$$

$$D(Z) = D_N(Z) \tag{2.7c}$$

2.3.2 The Radiation Model

The human vocal tract tube is actually terminated with the opening between the lips. A reasonable model for the effect of radiation at the lips is shown in Fig.2.11a which shows the lip opening as an orifice in a sphere. In this model, at low frequencies, the opening can be considered as a radiating surface, with the radiated sound waves being diffracted by the spherical baffle that represents the head. The resulting diffraction effects are complicated and difficult to represent. However, if the radiating surface (lip opening) is small compared to the size of the sphere, a reasonable approximation assumes that the radiating surface is set in a plane baffle of infinite extent as shown in Fig.2.11b. In such a case, it can be shown (5) that the sinusoidal steady state relation between the complex amplitudes of pressure and volume velocity at the lips is

$$P(\ell,s) = Z_L(s)\, U(\ell,s) \tag{2.8}$$

where $P(\ell,s)$ and $U(\ell,s)$ are the Laplace Transforms of $p(\ell,t)$ and $u(\ell,t)$ respectively and the "radiation impedance" at the lips is approximately of the form (5)

$$Z_L(s) = \frac{sR_rL_r}{R_r + sL_r} \tag{2.9}$$

R_r and L_r are termed as "radiation resistance" and "radiation inductance" respectively. Values of R_r and L_r that provide a good approximation to the infinite baffle are (5)

$$R_r = \frac{128}{9\pi^2} \tag{2.10a}$$

$$L_r = \frac{8a}{3\pi c} \tag{2.10b}$$

where a is the radius of the opening and c is the velocity of sound.

In a discrete-time model, the corresponding relationship desired is of the form

$$P_L(Z) = R(Z)\, U_L(Z) \tag{2.11}$$

where $P_L(Z)$ and $U_L(Z)$ are Z-transforms of $p(\ell,t)$ and $u(\ell,t)$, the sampled versions of the band limited pressure and volume velocity. One approach to obtain $R(Z)$ is to use the Bilinear Transform method. It can be shown that a reasonable approximation to the radiation effect at the lips is of the form

$$R(Z) = (1-Z^{-1}) \tag{2.12}$$

i.e. a first backward difference. Fig.2.12 shows how this radiation model can be cascaded to the vocal tract model.

2.3.3 The Glottal Excitation Model

In section 2.2, we have identified 3 major mechanisms of excitation, namely voiced, unvoiced and plosive. In the present glottal excitation modelling, however, we assume the excitation in the vocal tract is either:

1. Voiced excitation - Air flow from the lungs is modulated by the vocal cord vibration, resulting in a quasi-periodic pulse-like excitation.

or

2. Unvoiced excitation - Air flow from the lungs becomes turbulent as the air passes through a constriction in the vocal tract resulting in noise-like excitation.

Thus glottal excitation modelling requires a source that can provide either a quasi-periodic pulse waveform or a random noise waveform.

In the case of voiced speech, the excitation waveform must appear somewhat like the one as shown in Fig.2.13. A convenient way to represent the generation of the glottal wave is shown in Fig.2.14. The impulse train generator produces a sequence of unit impulses which are spaced by the desired pitch period. This signal in turn excites a linear system whose impulse response g(n) has the desired glottal wave shape. A gain control, A_V, controls the intensity of the voiced excitation. Rosenberg (6) in a study of the effect of glottal pulse shape on speech quality, found that the natural glottal pulse waveform could be replaced by a synthetic pulse waveform of the form:

$$g(n) = \tfrac{1}{2}\left[1-\cos(\pi n/N1)\right] \qquad o \leq n \leq N1$$
$$= \cos\left[\pi(n-N1)/2N2\right] \qquad N1 \leq n \leq N1 + N2$$
$$= 0 \qquad \text{otherwise} \qquad (2.13)$$

This waveshape is very similar in appearance to the pulses as shown in Fig.2.13. Since g(n) in Eq.(2.13) has infinite length, G(Z) has only zeros. However an all pole model is often more desirable.

For unvoiced sounds, the excitation is much simpler. All that is required is a source of random noise and a gain parameter, A_N, to control the intensity of the unvoiced excitation. For discrete-time models, a random number generator can provide a source of flat-spectrum noise. The probability distribution of the noise samples does not appear to be critical.

2.3.4 The Digital Model for Speech Production

Integrating the vocal tract model, the radiation model and the glottal excitation model together, we obtain a digital model for speech production as shown in Fig.2.15. By switching between the voiced and unvoiced excitation generators, we can model the changing mode of excitation. In the following sections a description of how linear predictive analysis can be used to determine the reflection coefficients is given. In Chapter Three, algorithms used to evaluate the pitch period and the gain G will be discussed.

2.4 LINEAR PREDICTIVE ANALYSIS (7) (8)

Fig.2.16 shows a simplified discrete-time model for speech production. In this model, the composite spectrum effects of radiation, vocal tract and glottal excitation are represented by a time varying digital filter $\hat{H}(Z)$. This system is excited by an impulse train for voiced speech or a random noise sequence for unvoiced speech. G is the parameter which controls the intensity of the excitation.

In linear predictive analysis, the signal s(n) is considered to be the output of the system $\hat{H}(Z)$ with input u(n) such that the following relation holds

$$s(n) = -\sum_{k=1}^{p} \alpha_k \, s(n-k) + G \sum_{i=0}^{q} b_i \, u(n-i) \qquad b_0=1 \qquad (2.14)$$

Eq. (2.14) implies that the output s(n) is a linear function of past outputs and present and past inputs. That is s(n) is predictable from linear combinations of past outputs and inputs. A special case of this model which is very useful for the analysis of speech is called the all-pole model, where bi = 0, $1 \leq i \leq q$, so that Eq.(2.14) becomes

$$s(n) = -\sum_{k=1}^{p} \alpha_k \, s(n-k) + Gu(n) \qquad (2.15)$$

Hence H(Z) has the form

$$H(Z) = \frac{S(Z)}{U(Z)} = \frac{G}{1 + \sum_{k=1}^{p} \alpha_k \, Z^{-k}} \qquad (2.16)$$

Since it is assumed that characteristic of the input u(n) is unknown, the signal s(n) can be predicted only approximately from a linear-combination of its past samples. Let this approximation of s(n) be $\tilde{s}(n)$ where

$$\tilde{s}(n) = -\sum_{k=1}^{p} \alpha_k \, s(n-k) \qquad (2.17)$$

where p is called the order of prediction.
Then the error between the actual value s(n) and the predicted value $\tilde{s}(n)$ is given by

$$e(n) = s(n) - \tilde{s}(n) = s(n) + \sum_{k=1}^{p} \alpha_k \, s(n-k) \qquad (2.18)$$

From Eq.(2.18) it can be seen that the prediction error sequence is the output of a system whose transfer function is

$$A(Z) = 1 + \sum_{k=1}^{p} \alpha_k Z^{-k} \tag{2.19}$$

which is the inverse filter for the system H(Z) of Eq.(2.16) i.e.

$$H(Z) = \frac{G}{A(Z)} \tag{2.20}$$

The basic problem of linear prediction analysis is to determine a set of predictor coefficients $\{\alpha_k\}$ directly from the speech signal in such a manner as to obtain a good estimate of the spectral properties of the speech signal through the use of Eq.(2.20). Because of the time varying nature of the speech signal, the predictor coefficients must be estimated from short segments of the speech signal. The basic approach is to find a set of predictor coefficients that will minimize the mean squared prediction error over a short segment of the speech waveform. The resulting parameters are then assumed to be the parameters of the system function H(Z) in the model for speech production.

The short time average prediction error is defined as

$$E_n = \sum_m e_n^2(m) \tag{2.21}$$

$$= \sum_m (s_n(m) - \tilde{s}_n(m))^2 \tag{2.22}$$

where $s_n(m)$ is a segment of speech that has been selected in the vicinity of sample n, i.e.

$$s_n(m) = s(m + n) \tag{2.23}$$

The approach that will be used to determine the limit of the summations in Eqs.(2.21) and (2.22) is called the Autocorrelation

Method. Assume the short segment of the speech waveform consists of N samples; the autocorrelation method assumes that the waveform segment $s_n(m)$ is identically zero outside the interval $0 \leq m \leq N-1$. This can be expressed as

$$s_n(m) = s(m + n)\ w(m) \qquad (2.24)$$

where w(m) is a finite length window (e.g. a Hanning window) that is identically zero outside the interval $0 \leq m \leq N-1$. Hence the corresponding prediction error, $e_n(m)$, for a pth order predictor, will be nonzero over the interval $0 \leq m \leq N-1 + p$. Thus, for this case, E_n can be properly expressed as:

$$E_n = \sum_{m=0}^{N+p-1} e_n^2(m) \qquad (2.25)$$

$$= \sum_{m=0}^{N+p-1} (s_n(m) - \tilde{s}_n(m)^2) \qquad (2.26)$$

$$= \sum_{m=0}^{N+p-1} \left[s_n(m) + \sum_{k=1}^{p} \alpha_k\, s_n\ (m-k)\right]^2 \qquad (2.27)$$

We can find the values of α_k that minimize E_n in Eq.(2.27) by setting

$$\frac{\partial E_n}{\partial \alpha_i} = 0 \qquad i = 1,2,\ldots p \qquad (2.28)$$

thereby obtaining the equations

$$\sum_{m=0}^{N+p-1} s_n\ (m-i)\ s_n(m) = - \sum_{k=1}^{p} \alpha_k \sum_{m=0}^{N+p-1} s_n(m-i)\, s_n(m-k) \qquad 1 \leq i \leq p \qquad 0 \leq k \leq p \qquad (2.29)$$

Since $s_n(m)$ is zero outside the interval $0 \leq m \leq N-1$, it can be shown that Eq.(2.29) can be expressed as:

$$\sum_{m=0}^{N-1-i} s_n(m+i)\, s_n(m) = -\sum_{k=1}^{p} \alpha_k \sum_{m=0}^{N-1-(i-k)} s_n(m) s_n(m+i-k)$$

$$1 \leq i \leq p$$

$$0 \leq k \leq p \qquad (2.30)$$

It can be seen that both sides of Eq.(2.30) are the short-time autocorrelation functions of $s_n(m)$. Autocorrelation functions are even functions, hence Eq.(2.30) becomes

$$\sum_{k=1}^{p} \alpha_k R_n(|i-k|) = -R_n(i) \quad 1 \leq i \leq p \qquad (2.31)$$

where

$$R_n(k) = \sum_{m=0}^{N-1-k} s_n(m)\, s_n(m+k) \qquad (2.32)$$

The set of equations given by Eq.(2.31) can be expressed in matrix form as

$$\begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & --- & R_n(p-1) \\ R_n(1) & R_n(0) & R_n(1) & --- & R_n(p-2) \\ R_n(2) & R_n(1) & R_n(0) & --- & R_n(p-3) \\ --- & --- & --- & --- & --- \\ --- & --- & --- & --- & --- \\ --- & --- & --- & --- & --- \\ R_n(p-1) & R_n(p-2) & R_n(p-3) & --- & R_n(0) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ --- \\ --- \\ --- \\ \alpha_p \end{bmatrix} = \begin{bmatrix} -R_n(1) \\ -R_n(2) \\ -R_n(3) \\ --- \\ --- \\ --- \\ -R_n(p) \end{bmatrix}$$

(2.33)

The p x p matrix of autocorrelation values is a Toeplitz matrix, i.e. it is symmetric and all the elements along a given diagonal are equal. To solve for the optimum predictor coefficients, we must first compute the quantities $R_n(k)$ for $0 \leq k \leq p$. Once this is done, we only have to solve Eq.(2.33) to obtain the α_k. Durbin's recursive method to solve for α_k will be discussed in the next section so as to find out the relationship between linear predictive analysis and the acoustic model for speech production.

2.5 RELATIONSHIP BETWEEN THE LINEAR PREDICTION MODEL AND THE ACOUSTIC TUBE MODEL

To find out how the linear prediction model relates to the acoustic tube model, we first examine the solution for Eq.(2.33). By exploiting the Toeplitz nature of the matrix of coefficients several efficient recursive procedures have been devised for solving this system of equations. The most efficient method known for solving this particular system of equations is Durbin's recursive procedure (7) which can be stated as follows:

$$E_n^{(0)} = R_n(0) \tag{2.34}$$

$$k_i = -\left[R_n(i) + \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R_n(i-j)\right] / E_n^{(i-1)} \quad 1 \leq i \leq p \tag{2.35}$$

$$\alpha_i^{(i)} = k_i \tag{2.36}$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} + k_i\, \alpha_{i-j}^{(i-1)} \qquad 1 \leq j \leq i-1 \tag{2.37}$$

$$E_n^{(i)} = (1-k_i^{\,2})\; E_n^{(i-1)} \tag{2.38}$$

Eqs. (2.35) to (2.38) are solved recursively for i=1, 2...p and the final solution is given by

$$\alpha_j = \alpha_j^{(p)} \qquad 1 \leq j \leq p \tag{2.39}$$

It can be seen that in the process of solving for the predictor coefficients for a predictor of order p, the solutions for the predictor coefficients of all orders less than p have also been obtained, i.e. $\alpha_j^{(i)}$ is the j^{th} prediction coefficient for a predictor of order i. Therefore at the i^{th} stage of this procedure, the set of coefficients $\{\alpha_j^{(i)}$ $j = 1, 2, \ldots i\}$ are the coefficients of the i^{th} order optimum linear predictor. Using these coefficients we can define

$$A^{(i)}(Z) = 1 + \sum_{k=1}^{i} \alpha_k^{(i)} Z^{-k} \tag{2.40}$$

to be the transfer function of the i^{th} order inverse filter (or prediction error filter). By substituting Eqs.(2.36) and (2.37) into Eq.(2.40), we obtain a recurrence formula for $A^{(i)}(Z)$ in terms of $A^{(i-1)}(Z)$, i.e.

$$A^{(i)}(Z) = A^{(i-1)}(Z) + k_i Z^{-i} A^{(i-1)}(Z^{-1}) \tag{2.41}$$

Hence the polynomial

$$A(Z) = 1 + \sum_{k=1}^{p} \alpha_k Z^{-k} \tag{2.42}$$

obtained by linear prediction analysis could be obtained by the recursion

$$A^{(0)}(Z) = 1 \tag{2.43a}$$

$$A^{(i)}(Z) = A^{(i-1)}(Z) + k_i Z^{-i} A^{(i-1)}(Z^{-1}) \tag{2.43b}$$

$$A(Z) = A^{(p)}(Z) \tag{2.43c}$$

where the parameters $\{k_i\}$ are called the PARCOR coefficients, which can be determined by Durbin's procedure. By comparing Eqs.(2.7) and Eqs.(2.43) it can be seen that the system function

$$H(Z) = \frac{G}{A(Z)} \tag{2.44}$$

obtained by linear prediction analysis has the same form as the system function of the lossless tube model consisting of p sections. If

$$r_i = k_i \tag{2.45}$$

then

$$D(Z) = A(Z) \tag{2.46}$$

Using Eqs.(2.3) and (2.45) it can be shown that the areas of the equivalent tube model are related to the PARCOR coefficients by

$$A_{i+1} = \left[\frac{1 + k_i}{1 - k_i}\right] A_i \tag{2.47}$$

i.e. the PARCOR coefficient gives a ratio between areas of adjacent sections. Thus the areas of the equivalent tube model are not absolutely determined and any convenient normalization will produce a tube model with the same transfer function.

Comparing Fig.2.16 and Fig.2.15 it can be seen that the transfer function H(Z) includes the effects due to glottal excitation and radiation at the lips. Hence the "area function" obtained using Eq.(2.47) cannot be said to be the area function of the human vocal tract. However, Wakita (9) has shown that if pre-emphasis is used prior to linear predictive analysis to remove the effects due to the glottal pulse and radiation then the resulting area functions are often very similar to vocal tract configuration that would be used in human speech.

2.6 THE LINEAR PREDICTIVE CODING SYSTEM

Fig.2.17 shows the basic configuration of the LPC experiment. The LPC analyser consists of a reflection coefficient estimator, a pitch detector, a gain estimator and a voiced/unvoiced decision

The LPC synthesizer is the one shown in Fig.2.15. The analyser extracts LPC parameters from the input speech signal and transmits them to the synthesizer which then uses the parameters to reconstruct the speech. In order to verify the actual performance of the LPC algorithms, coding and decoding of the parameters were discarded in the LPC experiment so that unquantized LPC parameters were used for speech synthesis. The transmission channel between the analyser and the synthesizer was also assumed to be perfect, i.e. no transmission errors.

CHAPTER THREE - LINEAR PREDICTIVE CODER SIMULATION

3.1 INTRODUCTION

This Chapter describes the LPC simulation in detail. A brief description of the equipment used is first given. Then algorithms of the LPC analyser and the LPC synthesizer are explained. Finally simulation results of two segments of speech are discussed.

3.2 SIMULATION EQUIPMENT

The equipment used for simulation was developed by M.J. Fairfield and P.J.Patrick at the Electrical and Electronic Department, University of Technology, Loughborough, U.K. It consists of a basic BBC computer system, a 6502 second processor, an analog board, a Beebex card, an ADC/DAC board and a framestore. The interconnections between these items are shown in Fig.3.1.

The menu of the data flow control program in the BBC computer is shown in Fig.3.2. In order to store speech segments on a BBC disk for simulation, the "INPUT SPEECH" operation is first chosen to allow 8 seconds of speech to be input through a microphone, filtered and sampled at 8 KHz. Each sample is converted into a 12-bit code which is then stored temporarily in the framestore using two bytes per sample as shown in Fig.3.3. The "STORE SPEECH" operation is then used to transfer data sequentially from the framestore to a BBC computer floppy disk. The speech file can then be examined, analysed or processed. To judge the quality of the processed speech, the "RETRIEVE SPEECH" operation is first chosen to transfer the processed speech data from a floppy disk to the framestore. The "OUTPUT SPEECH" operation is then used to transfer the data in the framestore to the DAC at a frequency of 8 KHz so that the processed speech can be listened to through a loudspeaker. Finally, the "RESET FSTORE" operation is used to reset every byte of the 128 K memory inside the framestore to >FF and choosing the "EXIT" operation allows the BBC computer to operate in the edit mode.

Two six-second speech segments were processed. They are the "AUDIO" and the "LAMB", i.e.

a) AUDIO (male voice)

"This audio tape is part of the training module on time management, from a series produced by the British Gas."

b) LAMB (female voice)

"Mary had a little lamb, its fleece was white as snow, and everywhere that Mary went...."

3.3 THE LPC ANALYSER

This section describes the components of the LPC analyser. As shown in Fig.2.17 the analyser includes a reflection coefficient estimator, a pitch detector, a gain estimator and a V/UV decision. The reflection coefficient estimator is based on the Le Roux and Gueguen recursion method, whereas the pitch detector is a modified version of the centre-clipped autocorrelation method. The principle of conservation of energy is used to derive the gain estimator, and the criterion for the V/UV decision is determined according to statistical information. We first define the prediction order and the analysis interval of the LPC analyser.

3.3.1 Prediction Order

The prediction order of the LPC analyser depends on the number of sections of the lattice filter which is used for the LPC synthesis and the choice of number of sections of the lattice filter depends upon the sampling rate chosen to represent the speech signal. In section 2.3.1 it was mentioned that

$$T = 2\tau \tag{3.1}$$

where T is the sampling period and τ is the one way propagation time in a single section of the lattice filter.

If there are p sections, for a human vocal tract length, ℓ, and the speed of sound c,

$$\tau = \ell/cp \tag{3.2}$$

substituting Eq.(3.2) into Eq.(3.1) and rearranging, we have

$$p = \frac{2\ell}{cT} \tag{3.3}$$

The sampling frequency, f_s, was chosen as 8 KHz in the LPC experiment and therefore, using ℓ = 17.5 cm and c = 35000cm/sec, we have p = 8. However, in order to account for non-ideal circumstances and possible zeros in the speech spectrum, the prediction order of the LPC analyser was chosen to be 10, i.e. p = 10.

3.3.2 Analysis Interval

LPC analysis is actually a kind of short-term spectral analysis and hence it assumes the signal being analysed to be stationary within the analysis interval. It is therefore necessary to perform LPC analysis within an interval where vocal tract movement is negligible. This implies that the shorter the analysis interval is, the more accurate the spectral estimation. However, the data within the analysis interval will also be used for pitch detection using the autocorrelation function method which requires the presence of at least two pitch periods within the detection frame. It is possible to have a pitch frequency as low as 70 Hz for some speech signals and that means that a data frame of 28.5 ms is needed. In order to compromise between the desires to detect low fundamental frequency and to minimize the averaging of the time-varying speech signal, an analysis interval of 25 ms was chosen in the LPC experiment. This is equivalent to 200 data samples per analysis frame for $f_s = 8$ KHz.

3.3.3 <u>The Reflection Coefficient Estimator</u>

The configuration of the reflection coefficient estimator is shown in Fig.3.4. Basically, input speech waveform is divided into overlapping blocks and a smooth window function is applied to each block as shown in Fig.3.5. Each block is then pre-emphasised before being used to compute the normalized auto-correlation function for 10 lags $\{NR(i), i = 0...10\}$. NR(i) is then used to determine the first 10 reflection coefficients, using the LeRoux and Gueguen procedure.

3.3.3.1 <u>Windowing</u>

It can be seen from Fig.3.5 that during reflection coefficients estimation, even if no window is explicitly introduced, there is a rectangular window implicit in the treatment of the data sequence, because only a given sequence of 220 samples $\{X(n), n=0 \ldots 219\}$ is utilized in the estimation. It has been shown in Chapter Two that in linear predictive analysis a model spectrum $G^2/|A(\exp(j\theta))|^2$ is being used to represent a data spectrum $|X(\exp(j\theta))|^2$. If no explicit windowing is carried out, discontinuities between values of X(0), X(219) and the numerical values of zero (outside of the implicit rectangular window) can cause spectral distortion. For this reason, a Hanning window was used in the LPC experiment. The shape of the Hanning window is shown in Fig.3.6. The windowed data WX(n) could then be expressed as

$$WX(n) = X(n) * 0.5 * (1 - \cos 2\pi n/219) \quad n=0,\ldots,219 \quad (3.4)$$

3.3.3.2 Pre-emphasis

In order to model the human vocal tract accurately, the reflection coefficients of the lattice filter must be determined from speech waveform which is pre-processed so that the effects of the glottal excitation and radiation at the lips are removed. Wakita's (9) experiments have shown that this can be done by a pre-emphasis of the form $[1 - \mu Z^{-1}]$ where μ is near unity. For $\mu = 1$, the result is an approximate + 6dB/octave slope. This will result in a slight upward shift for the estimated formant frequency location with respect to no pre-emphasis ($\mu = 0$). In the LPC experiment, a factor of $\mu = 0.95$ was chosen so that the pre-emphasised data could be expressed as

$$PX(n) = WX(n)-0.95*WX(n-1) \qquad n=0,\ldots,219 \tag{3.5}$$

3.3.3.3 The Normalized Autocorrelation Function

The calculation of the normalized autocorrelation function which is needed for the determination of the reflection coefficients is straightforward. Utilizing Eq.(2.32) with N = 220, we have

$$AR(i) = \sum_{m=0}^{220-1-i} PX(m) * PX(m+i) \tag{3.6}$$

Since the order of prediction is 10, the autocorrelation function needed is $\{AR(0), AR(1)\ldots AR(10)\}$. Therefore Eq.(3.6) should be calculated for $i = 0,\ldots, 10$. The autocorrelation function is then normalized with respect to AR(0).
The normalized autocorrelation function can then be expressed as

$$NR(i) = AR(i)/AR(0) \qquad i = 0 \ldots 10 \tag{3.7}$$

3.3.3.4 The Le Roux and Gueguen Method

Several recursive methods have been proposed to determine reflection coefficients from the autocorrelation function. One of them is Durbin's recursive procedure which was discussed in section (2.5). However very little is known about the range of magnitude of the intermediate variables that appear during the recursion and this causes troublesome scaling problems when the procedure is carried out using fixed-point arithmetic digital signal processors (e.g. TMS 32010).

This problem was solved by a method introduced by J.Le Roux and C.Gueguen (10). This method was derived from Durbin's recursive procedure with new intermediate variables introduced using inner product formulation. The flow diagram of the Le Roux and Gueguen procedure for a 10th order LPC is shown in Fig.3.7. It was shown that all the intermediate variables lie between -1 and +1 and hence implementation can be conducted using fixed point arithmetic. According to experimental results, Le Roux and Gueguen claimed that the differences between the results obtained by their method using 16 bit fixed-point arithmetic and usual algorithms implemented using floating point processors is less than 0.005 on K(10).

3.3.4 The Pitch Detector

There are many practical algorithms being proposed for pitch extraction (11). However, in a paper by Oh and Un (12) it was reported that for pitch extraction of noisy speech, algorithms that use an autocorrelation function (ACF) yield better results than others. Methods using an autocorrelation function are based on the fact that if the pitch period of a sampled speech segment is P_o samples, the autocorrelation function of the segment will attain a maximum at samples 0, $\pm P_o$, $\pm 2P_o$,....

The pitch period can then be estimated by locating the second maximum of the ACF. However, in cases when the autocorrelation peaks due to the vocal tract response are larger than those due to the periodicity of the vocal excitation, the simple procedure of picking the largest peak in the ACF will fail. To overcome this problem, it is useful to pre-process the speech segment before calculating the ACF so as to make the periodicity more prominent while suppressing other distracting features. Techniques which perform this type of operation on a signal are called "Spectrum Flattener" since their objective is to remove the effects of the vocal tract transfer function, thereby bringing each harmonic to the same amplitude level as in the case of a periodic impulse train. Numerous spectrum flattening techniques have been proposed. However, a technique called "centre-clipping" suggested by Sondhi (13) appears to be the easiest to implement.

Sondhi's autocorrelation method with centre-clipping is shown in Fig.3.8. Basically input speech is divided into blocks (no overlapping). Each block of data with d.c. offset removed is centre-clipped and then the autocorrelation function is calculated. The pitch period P_O can then be estimated by locating the maximum peak of the ACF. In the LPC experiment a speech wave was divided into 25ms blocks, i.e. 200 samples per frame. This means that if Sondhi's method were used for pitch detection, a 200 points autocorrelation function would have to be evaluated for each frame. However, it was realized that the TMS32010 can only calculate up to a 128 points autocorrelation function in a "pipe-line" fashion. Beyond that a cumbersome data handling procedure would be needed. To overcome this problem, decimation and interpolation techniques are used to modify Sondhi's method and the modified method is shown in Fig.3.9.

The ½ decimator is used to down sample the input from 200 data/frame to 100 data/frame. Then Sondhi's method gives a crude estimation for the pitch period. A quadratic interpolator is then used to estimate a more accurate value for the pitch period. In fact, this method is very similar to the SIFT algorithm (11) proposed by J.D.Markel although the SIFT algorithm utilizes inverse filtering for spectral flattening whereas this method uses centre-clipping.

3.3.4.1 The ½ Decimator

In the LPC experiment, input speech was sampled at 8 KHz. A ½ decimation is equivalent to reducing the sampling frequency to 4 KHz. In order to avoid aliasing distortion, the 8 KHz sampled speech must first be low-pass filtered before decimation. In fact the ½ decimation involves just passing the 8 KHz sampled speech through a low pass filter and takes alternate outputs of the filter as the decimator output. The filter chosen was a 1 KHz cutoff, third order Butterworth low pass filter as shown in Fig.3.10. The coefficients of the filter were determined using the Bilinear Transformation technique (14). The output of the decimator can be expressed as

$$DX(m) = FX(2*m+1) \qquad m = 0, 1, \ldots 99 \qquad (3.8)$$

where FX is the output of the low pass filter.

3.3.4.2 The d.c. Offset Extractor

The mean of the data should be extracted before calculating the autocorrelation function. Although speech is a zero mean process over long intervals, considerable bias can exist during a single frame. This bias within the frame can lead to shape distortion of the desired autocorrelation function and this will result in wrong pitch period estimation. The mean extraction operation includes calculating the mean of DX(m) $m = 0, \ldots, 99$ and subtracting it from each of the samples, i.e.

$$OS = \sum_{m=0}^{99} DX(m)/100 \qquad (3.9)$$

$$RX(m) = DX(m) - OS \qquad m = 0, \ldots, 99 \qquad (3.10)$$

3.3.4.3 Centre-Clipping

Centre-clipping of speech was first used by Licklider and Pollack (15) in an experiment in which they showed that whereas speech that has been infinitely peak clipped is highly intelligible, even a few percent of centre clipping drastically reduces intelligibility. This is because infinite peak-clipping retains the formants of the speech signal (although it introduces a few secondary formants), whereas centre-clipping destroys formant structure while retaining the periodicity. It is the removal of formant structure that is so important for pitch detection.

In the original scheme proposed by Sondhi, the centre-clipped speech signal is obtained by a non-linear transformation

$$CX(m) = T\,[RX(m)] \qquad (3.11)$$

where $T\,[\;]$ is as shown in Fig.3.11.

It has been found that a clearer indication of periodicity in the autocorrelation function is obtained for a higher clipping level. However, it is possible that the amplitude of the signal may vary appreciably across the duration of the speech segment, so that if the clipping level is set too high, there is a possibility that much of the waveform will fall below the clipping level and be lost. For this reason Sondhi's original proposal was to set the clipping level at 30% of the maximum amplitude across the whole speech segment. A procedure which permits a greater percentage to be used is to find the peak amplitude in both the first third and last third of the segment and set the clipping level at a fixed percentage of the smaller of these two maximum levels. The percentage used in the LPC experiment was 60%, and hence the threshold THRE, could be calculated as

$$\text{THRE} = 0.6 * \text{MIN}\left[\text{MAX}\left[|\text{RX}(m)|\right],\ \text{MAX}\left[|\text{RX}(n)|\right]\right] \qquad m = 0,\ldots,32 \quad n = 67,\ldots,99 \tag{3.12}$$

The output of the centre-clipping process CX(m) could then be calculated as

$$\begin{aligned} \text{CX}(m) &= \text{sgn}\left[\text{RX}(m)\right] * \left[|\text{RX}(m)| - \text{THRE}\right] & |\text{RX}(m)| \geq \text{THRE} \\ &= 0 & |\text{RX}(m)| < \text{THRE} \end{aligned} \qquad m = 0,\ldots,99 \tag{3.13}$$

However, overflow problems may occur if we use the CX(m) in Eq.(3.13) to calculate the autocorrelation function using only 16 bit fixed-point arithmetic. One simple method of solving this problem is to replace T[ ] in Fig.3.11 by a 3-level centre clipping function T'[ ] as shown in Fig.3.12 (16), i.e. the amplitude of CX(m) is hardlimited to unity. Hence for the worst case when THRE = 0, the maximum amplitude of the autocorrelation function is 100 which is within the 16 bit range. It has been shown that the shape of the auto-correlation function calculated using 3-level centre-clipped data is very similar to the one using ordinary centre-clipped data. In the LPC experiment, 3-level centre clipping was used and hence CX(m) was calculated as

$$\begin{aligned} \text{CX}(m) &= \text{sgn}\left[\text{RX}(m)\right] & |\text{RX}(m)| \geq \text{THRE} \\ &= 0 & |\text{RX}(m)| < \text{THRE} \end{aligned} \qquad m = 0,\ldots,99 \tag{3.14}$$

Fig.3.13 shows a speech segment and its corresponding Fourier spectrum. Fig.3.14 and Fig.3.15 show the effects of centre-clipping and 3-level centre clipping on the frequency spectrum of the speech segment. It can be seen that both centre-clipping processes give similar spectra-flattening effects on the original speech spectrum.

3.3.4.4 The Autocorrelation Function

The calculation of autocorrelation function for the pitch detector is very similar to the one described in section 3.3.3.3, except that N = 100, and the ACF is calculated up to 99 lags, i.e.

$$DR(m) = \sum_{i=0}^{100-1-m} CX(i) * CX(i+m) \qquad m = 0, \ldots, 99 \tag{3.15}$$

Fig.3.16 and Fig.3.17 show the autocorrelation functions calculated using the centre-clipped data shown in Fig.3.14a and Fig.3.15a respectively. It can be seen that both ACF are very similar in shape, as we have mentioned in the previous section.

3.3.4.5 Peak Picking

As we have mentioned in section 3.3.4, if the pitch period of a speech segment is P_o samples, the autocorrelation function of the segment attains a maximum at samples $0, \pm P_o, \pm 2P_o, \ldots$ However, because of the finite length of the windowed speech segment involved in the computation of DR(m), there is less and less data involved in the computation as m increases. In a simple case where the speech segment is a sinusoidal wave, a relationship between the maximums is $DR(0) > DR(P_o) > DR(2P_o), \ldots$ Therefore instead of using complicated pattern recognition techniques, a simple way to find P_o is to locate the maximum peak across the autocorrelation function but excluding DR(0). In the LPC experiment the searching procedure was started from DR(15) since samples in the vicinity of DR(0) might have amplitudes greater than $DR(P_O)$. The flowchart of the searching operation is shown in Fig.3.18. The result of the searching procedure, P_D, however, is not the required pitch period, since the time scale of DR(m) is compressed by a factor of two due to the decimation process. The next section will describe how to "time re-scale" DR(m) and estimate a more accurate value for the pitch period using an interpolation technique.

3.3.4.6 The 2/1 Interpolator

"Time rescaling" of DR(m) is simply expanding the time scale of DR(m) by a factor of two. Hence

$$PR(2 * m) = DR(m) \qquad m = 0, 1, \ldots, P_D, \ldots 99 \tag{3.16}$$

where PR(n), n = 0, 1 ... 198, 199 is the "time rescaled" DR(m). $DR(P_D)$ is then rescaled to $PR(2P_D)$. Fig.3.19 shows the vicinity of $PR(2P_D)$ in the time domain. It can be seen that in order to give a more accurate estimation for the pitch period, it is necessary to find the interpolation equation F(t). In the LPC experiment, a quadratic interpolator was employed for this purpose. From Appendix I, it can be shown that F(t) can be expressed as

$$F(t) = PR(2P_D-2)\,\theta_O(t) + PR(2P_D)\,\theta_1(t) + PR(2P_D+2)\,\theta_2(t) \quad (3.1$$

where $\theta_O(t) = \frac{1}{8}\left[(t-2P_D)\ (t-2P_D - 2)\right]$ (3.18a)

$$\theta_1(t) = \frac{-1}{4}\left[(t-2P_D + 2)\ (t-2P_D-2)\right] \tag{3.18b}$$

$$\theta_2(t) = \frac{1}{8}\left[(t-2P_D + 2)\ (t-2P_D)\right] \tag{3.18c}$$

In order to find the value of t when F(t) reaches maximum, we differentiate F(t) with respect to t and set the resulting expression to zero. i.e.

$$PR(2P_D-2)\,\frac{d\theta_O(t)}{dt} + PR(2P_D)\,\frac{d\theta_1(t)}{dt} + PR(2P_D+2)\,\frac{d\theta_2(t)}{dt} = 0 \qquad (3.19)$$

Evaluating the derivatives of Eqs.(3.18) with respect to t, substituting into Eq.(3.19) and rearranging terms, we have

$$t\Big|_{peak} = 2P_D + \frac{[PR(2P_D-2)-PR(2P_D+2)]}{[PR(2P_D-2)-2*PR(2P_D)+PR(2P_D+2)]}$$

$$= 2P_D + CORR. \qquad (3.20)$$

where CORR is termed the "correction coefficient" of the interpolator. Hence Eq.(3.20) gives a better estimation for the pitch period. However, because of the nature of the LPC synthesizer, an integer value for the pitch period is required. Therefore in the LPC experiment, the output of the pitch detector, P_I, was defined as

$$P_I = 2P_D + 1 \qquad CORR \geq 0.5 \qquad (3.21a)$$

$$= 2P_D - 1 \qquad CORR \leq -0.5 \qquad (3.21b)$$

$$= 2P_D \qquad \text{otherwise} \qquad (3.21c)$$

Fig.3.20 shows the flow chart of the interpolation procedure. It can be seen that rearrangements are made so as to avoid divisions.

3.3.5 Voiced/Unvoiced (V/UV) Decision

A reliable pattern recognition approach to V/UV decision of speech was proposed by Atal and Rabiner (17). It involves calculating: 1) the energy of the speech segment; 2) zero crossing rate; 3) normalized autocorrelation coefficient at unit sample delay; 4) first prediction coefficient and 5) energy of the prediction error. Then according to statistical information concerning the five measured parameters, a distance measure technique is used to make the V/UV decision. However, due to the present TMS32010 technology, and the limited time available for the V/UV decision operation, the above pattern recognition approach appears to be impracticable for the present LPC experiment. Therefore, in the LPC experiment, the normalized autocorrelation coefficient at unit sample delay was chosen to be the only parameter used for the V/UV decision. This is because this parameter is a by-product in the calculation of the reflection coefficients and hence no further calculation is needed. It was also found that this parameter is a reliable measure in V/UV decision for most speech segments of the two testing speeches "AUDIO" and "LAMB" (Section 3.2).

In fact the V/UV decision parameter is NR(1) calculated by Eq.(3.7). NR(1) is the correlation between adjacent speech samples and, by definition, varies between -1 and +1. Due to the concentration of low-frequency energy in voiced sounds, adjacent samples of voiced speech waveform are highly correlated and NR(1) is close to unity. On the other hand NR(1) is close to -1 for unvoiced speech.

The threshold value of NR(1) for the V/UV decision depends on the input filtering processes and the pre-emphasis factor μ being used. Hence it can only be determined by trial and error procedure, and in the LPC experiment, it was set at 0.2.

This means that any speech segment having a value of NR(1) greater than or equal to 0.2 is classified as voiced. Otherwise that segment is classified as unvoiced.

This V/UV decision is actually incorporated with the pitch detector in such a way that if the speech segment being analysed is classified as unvoiced, the final estimate value of the pitch period, PITCH, is set to zero. Otherwise PITCH is set equal to the output of the pitch detector, i.e.

$$\text{PITCH} = 0 \qquad \text{NR}(1) < 0.2 \tag{3.22a}$$

$$= P_I \qquad \text{NR}(1) \geq 0.2 \tag{3.22b}$$

where PITCH is the final estimate of the pitch period. It can be seen from Eqs.(3.22) that the V/UV parameter is already embedded in the value of PITCH, i.e.

$$\text{V/UV} = \text{voiced} \qquad \text{PITCH} \neq 0 \tag{3.23a}$$

$$= \text{unvoiced} \qquad \text{PITCH} = 0 \tag{3.23b}$$

3.3.6 <u>The Gain Estimator</u>

An accurate method of estimating the gain G for the lattice filter V(Z) (Fig.2.15) is first passing the speech segment being analysed through a filter with transfer function $^1/V(Z)$ and then evaluating G using the r.m.s. value of the filter output. However this inverse filtering process was found to be impracticable for the present LPC experiment.

A less accurate but faster approach (that was actually used in the LPC experiment) is to use AR(0), which has already been calculated in the reflection coefficient estimation procedure (Section 3.3.3.3), to calculate G. Although AR(0) is the r.m.s. value of the pre-emphasised windowed speech segment, it is reasonable to assume the energy of the glottal excitation is roughly proportional to AR(0). Therefore in the LPC experiment, the gain G was calculated as

$$G = \beta\sqrt{AR(0)} \tag{3.24}$$

where β is a scaling constant and was determined by trial and error procedure so that the output amplitude of the LPC synthesizer would not cause arithmetic overflow.

3.3.7 The Complete LPC Analyser

Fig.3.21 shows the complete LPC analyser configuration. Speech X(n) is input to two main devices, namely, the reflection coefficient estimator and the pitch detector. The normalized autocorrelation coefficient NR(1) is used to modify the output of the pitch detector so as to decide the final value of the pitch period, PITCH. A by-product of the reflection coefficient estimation procedure, AR(0), is used to estimate the gain parameter G. Therefore 12 parameters are extracted from each frame of speech. They are 10 reflection coefficients K(i), i=1, ...10; the gain G and the pitch period PITCH. These parameters are then transmitted to the LPC synthesizer which reconstructs the speech through a lattice filter.

3.4 THE LPC SYNTHESIZER

This section describes the components of the LPC synthesizer, which is based on the digital models described in Chapter Two. The digital models, however, are modified so as to speed up the synthesis process. As we have shown in Fig.2.15, the synthesizer includes a vocal tract model, a radiation model, a glottal waveform generator, a random noise generator and a voiced/unvoiced switch.

3.4.1 The Vocal Tract and Radiation Models

The vocal tract model used in the LPC experiment is based on the lattice filter shown in Fig.2.10 whereas the radiation model is based on Eq.(2.12). The two models are cascaded together as shown in Fig.2.12. It has been shown in Section 3.3.1 that the number of sections of the lattice filter was chosen to be 10. Fig.3.22 shows a 10th order lattice filter with infinite glottal impedance. It can be seen that each junction requires 4 multiplications and 2 additions. Since one multiplication in the TMS32010 requires one more instruction than one addition, it is of interest to consider another junction structure which may require fewer multiplications. This can easily be derived by considering a typical junction as depicted in Fig.3.23a. The difference equations represented by this diagram are:

$$u^+(n) = (1+r)\ w^+(n) + r\ u^-(n) \qquad (3.25a)$$

$$w^-(n) = -rw^+(n) + (1-r)\ u^-(n) \qquad (3.25b)$$

Rearranging terms, we have

$$u^+(n) = w^+(n) \quad + \quad r * [w^+(n) + u^-(n)] \qquad (3.26a)$$

$$w^-(n) = u^-(n) \quad - \quad r * [w^+(n) + u^-(n)] \qquad (3.26b)$$

Since the term $r * [w^+(n) + u^-(n)]$ occurs in both equations this configuration requires only one multiplication and three additions as shown in Fig.3.23b. Fig.3.24 shows the lattice filter which uses the one multiplier structure, and this was the lattice filter used in the LPC experiment. $u_G(n)$ is the glottal excitation input to the lattice filter and $u_L(n)$ is the filter output.

It has been shown in section 2.3.2 that the radiation effect at the lips can be modelled approximately using a network of the form $[1 - Z^{-1}]$. This network is shown in Fig.3.25. The synthesizer output $\tilde{s}(n)$ can then be expressed as

$$\tilde{s}(n) = u_L(n) * [1 - Z^{-1}] \tag{3.27}$$

3.4.2 The Glottal Pulse Generator

The glottal pulse generator used in the LPC experiment is based on the configuration as shown in Fig.2.14. It has been found that the width of the glottal pulse varies for different pitch periods (5). This means that the glottal pulse model G(Z) would have to be a time-variant filter. In order to avoid complex algorithms for evaluating the transfer function G(Z) for different pitch periods, a fixed glottal pulse waveform was used in the LPC experiment. The glottal pulse waveform was determined using Eqs.(2.13) with N1 = 14 and N2 = 6. Fig.3.26 shows the pulse waveform and its corresponding fourier spectrum. It can be seen that the effect of the glottal pulse in the frequency domain is to introduce a low pass filtering effect. Fig.3.27 shows the glottal pulse generator used in the LPC experiment. The glottal pulse was stored in an array GP(n) n = 0, ..., 20 and was output according to the subroutine with flow chart shown in Fig.3.28.

3.4.3 The Random Noise Generator (18)

The noise generator used in the LPC experiment was actually a shift register. The length of the shift register was chosen to be 11-BIT so that the fundamental period of the pseudo-random sequence produced is long compared to an analysis/ synthesis interval. The operation of the shift register is shown in Fig.3.29. The digit B_o must be preset to 1 for initialization and a number can then be calculated for every right shift by the expression

$$\text{NOISE} = \sum_{i=0}^{10} B_i * 2^i - 1024 \tag{3.28}$$

where NOISE is the output of the random noise generator. The signal NOISE is a zero mean, 1023 to -1023 uniformly distributed pseudo-random sequence. The fundamental period of the sequence is 2047 samples which is more than ten times the length of an analysis/synthesis interval (200 samples). Fig.3.30 shows a segment of the number sequence and its corresponding frequency spectrum. It can be seen that the sequence possesses a noise-like frequency spectrum and this shows that the signal NOISE is a good approximation to the unvoiced excitation for the vocal tract filter.

3.4.4 The Complete LPC Synthesizer

Fig.3.31 shows the complete LPC synthesizer configuration. The parameters which operate the synthesizer are the pitch period PITCH, the gain G and 10 reflection coefficients. The V/UV switch is operated in such a way that if PITCH = 0, then it is switched to UV, otherwise it is switched to V. The synthesizer receives a new set of parameters for every 25 ms. However, parameters are updated only at the beginning of a pitch period. This technique of speech synthesis is called "Pitch Synchronous Synthesis", and has been found to be a much more effective synthesis strategy than the process of updating the parameters at the beginning of each frame ("Asynchronous Synthesis").

3.5 THE LPC SIMULATION

Two simulation programs were written, namely the LPC analyser [LPC.ANY] and the LPC synthesizer [LPC.SYN]. [LPC.ANY] and

[LPC.SYN] were written according to the algorithms described in Sections 3.3 and 3.4 respectively. Fig.3.32 shows the file handling configuration of the LPC simulation. As we mentioned in Section 3.2, two speech files were processed, viz. [AUDIO] and [LAMB] . The [LPC.ANY] program generated a set of parameter files for each speech file. The parameter files consisted of a pitch file [XXX.PIT] ; a reflection coefficient file [XXX.RC] and a gain file [XXX.G], where XXX is the first three letters of a speech file filename. The [LPC.SYN] program then used the parameter files to reconstruct the original speech and the synthesized speech samples were stored in an output file [XXX.OUT] .

According to informal subjective listening tests, the two synthesized speeches were very intelligible but with some distortion at speech segments with long pitch periods. This was because the pitch periods of those particular segments were so long that there were less than two pitch periods within the analysis interval. This resulted in wrong pitch period estimation and hence the distortion.

CHAPTER FOUR - LINEAR PREDICTIVE CODER IMPLEMENTATION

4.1 INTRODUCTION

This chapter first gives a brief description of a TMS32010 software development system which was developed during the course of the present work. Two TMS32010 were involved in the implementation experiment. One was operated as the analyser and the other as the synthesizer. The TMS32010 software is based on the algorithms described in Chapter Three and is explained in Sections 4.3 and 4.4. Section 4.5 describes the communication between the analyser and the synthesizer. Finally results of informal subjective listening tests on the coder are discussed.

4.2 TMS32010 SOFTWARE DEVELOPMENT SYSTEM

The TMS32010 software development system was built around the TMS32010 Digital Signal Processor Evaluation Module (EVM) (19). The TMS32010 EVM is a single board development system for the TMS32010. The EVM can stand alone as a development system using the on-board text editor for the creation of TMS32010 assembly language text files (20). It also provides the facility for using audio tape as a mass storage media. The EVM can accept text files from a host computer through one of the two EIA ports or from the audio tape interface. In either situation, the resident assembler will convert the incoming text into executable code in just one pass by automatically resolving labels after the first assembly pass is complete. The object code is stored in a 4K-word memory space allowing the utilization of the entire TMS32010 address space for program development.

The EVM operating system can be divided into four segments, namely the debug monitor, the assembler/reverse assembler, the text editor and the TMS2764 PROM utility. The EVM firmwave supports three ports for the operation of inputting and outputting data (text and object code) for storage and/or display. Two of the ports conform to EIA RS232C specifications and are called Port 1 and Port 2. The third port, Port 3, is an audio tape connnection.

It was found that the audio tape storage system is very slow because port 3 can only operate at 300 baud. Therefore a BBC microcomputer system was connected to Port 2 of the EVM as shown in Fig.4.1 so that the disk storage of the BBC system could be used as a mass storage media for the EVM. A terminal was connected to port 1 of the EVM so that it could control the EVM under normal operation mode and could communicate with the BBC system via the transparency mode. Incorporated with the BBC software, the development system provides useful facilities for TMS32010 program development. These include:

1) TMS32010 text programs can be created using the EVM text editor. The text programs can then be transferred to the BBC system and stored in a floppy disk.

2) A TMS32010 text program which is stored in a BBC disk can be transferred from the BBC system to the EVM. The EVM can either accept the text program into its text editor for editing or use its assembler to convert the text program into TMS32010 machine code for program debugging or real-time testing.

3) The contents of the TSM32010 program memory and data memory can be transferred from the EVM to the BBC system for analysis.

4) Hardcopies of text programs listings, reverse assembled programs listings and assembler label tables can be obtained from the Epson printer.

4.3 THE LPC ANALYSER

This section describes the TMS32010 subroutines for the LPC analyser. The algorithms of the subroutines are based on the procedures described in section 3.3. Some of the algorithms were re-organised so that they could be implemented by the TMS32010 in a more effective way. The technique of single buffering analysis is also described.

4.3.1 The Reflection Coefficient Estimator

Fig.4.2 shows the main TMS32010 software subroutines for the reflection coefficient estimator. They include a Windowing/Pre-emphasis/Autocorrelating network subroutine, an autocorrelation function normalization subroutine and a LeRoux and Gueguen recursion subroutine. Variables in these subroutines with magnitude less than unity were all represented in 16 bits Q15 format.

4.3.1.1 Windowing, Pre-emphasis, Autocorrelating Network

The windowing, pre-emphasis and autocorrelating operations described in Section 3.3.3 were all involved in the processing of 220 data samples per analysis interval. They were integrated together as a digital network so as to facilitate the implementation of the operations using the TMS32010. The digital network is shown in Fig.4.3 with inputs X_n and W_n. X_n were 220 speech samples stored in program memory >F1C to >FF7 and W_n were data of a 220 points Hanning Window stored in program memory >E20 to >EFC. Intermediate variables of the network must be initialized at the beginning of each analysis interval, i.e.

$$AR_i = 0 \qquad (4.1a)$$

$$D_i = 0 \qquad i = 0, \ldots, 10 \qquad (4.1b)$$

X_n and W_n were input to the network synchronously starting from X_o and W_o respectively. After X_{219} and W_{219} were input to the network, outputs AR_i $i=0,\ldots,10$ were the required autocorrelation function. This method of calculating the autocorrelation function is called the "Contribution Method". The values of AR_i were all represented in double precision, i.e. 32 bits, so as to increase the input dynamic range. The coefficients had to be normalized with respect to AR_o before being used to determine the reflection coefficients.

4.3.1.2 Normalization of the Autocorrelation Function

Normalization of the autocorrelation function involves the process of dividing the entire autocorrelation function by the autocorrelation coefficient at zero lag. The autocorrelation function coefficients AR_i determined by the network shown in Fig.4.3 were all represented in 32-bits, and 32-bit division in the TMS32010 is not simple. However, TMS32010 supports 16-bit division in a very convenient way by using a special instruction called "Condition Subtract (SUBC)". Hence it was necessary to transform the 32-bit autocorrelation coefficients into 16-bit representation. The transformation was divided into two parts as shown in Fig.4.4.

First the number of leading zeros of the 32-bit AR_o was counted. If the number of leading zeros was greater than 16, then the shift counter SCNT would be set equal to zero. Otherwise SCNT would be set equal to the number of leading zeros.
If SCNT was zero, then MR_i $i=0, \ldots, 10$, the modified autocorrelation coefficients, would be set equal to the lower 16-bits of AR_i. Otherwise AR_i would be shifted to the left by

SCNT-1 bits and MR_i would be set equal to the higher 16 bits of AR_i. The flowcharts of the leading zeros counting subroutine and the shifting subroutine are shown in Fig.4.5 and Fig.4.6 respectively.

The transformed autocorrelation coefficients MR_i were then used for the normalization process which was mainly dividing MR_i by MR_o for i=0,...,10. Fig.4.7 shows the flowcharts of the normalization subroutines.

4.3.1.3 The Le Roux and Gueguen Method

The TMS32010 subroutine for the Le Roux and Gueguen recursion procedure was directly transformed from the flow chart depicted in Fig.3.7. Inputs to the subroutine were the normalized autocorrelation function NR_i i=0,...,10. Auxiliary registers ARØ and AR1 of the TMS32010 were used as loop counters for the recursion process. The division subroutine DIV as shown in Fig.4.7a was also used for the determination of the reflection coefficients. The resulting reflection coefficients were all represented in 16 bits Q15 format and were stored temporarily in program memory >CA2 to >CAB before being transmitted to the synthesizer. Fig.4.8 shows the flowchart of the reflection coefficient estimator main program.

4.3.2 Pitch Detector

The TMS32010 software for the pitch detector was written according to the algorithms described in Sections 3.3.4 and 3.3.5. Fig.4.9 shows the flowchart of the pitch detector main program. It can be seen that the V/UV decision subroutine was integrated into the pitch detector program so that the pitch period value at the end of the program would be final. Inputs to the pitch detector program were 200 data samples stored in program memory >F30 to >FF7. The filter coefficients for the decimation process were stored in program memory >D90 to >D96 and were transferred to the data memory when needed.

The final value of the pitch period, PITCH, was represented in 16 bits 2's complement format and was stored temporarily in program memory >CAO before being transmitted to the synthesizer.

4.3.3 The Gain Estimator

In the LPC simulation program, the gain G was calculated according to Eq.(3.24). However, in the LPC implementation experiment, the scaling procedure was done at the synthesizer so that at the TMS32010 analyser, the gain G was calculated as

$$G = \sqrt{AR_o} \qquad (6.2)$$

where AR_o was a 32-bit number and was calculated during the reflection coefficient estimation process (Section 4.3.1.1). The square root of AR_o was calculated by an iterative process called "Mid-point Method". The flowchart of the gain estimator subroutine is given in Fig.4.10. It can be seen that the accuracy of the square root process was set to 1 so that the resulting G would be an integer. The final value of the gain G was represented in 16 bits 2's complement format and was stored temporarily in program memory >CA1 before being transmitted to the synthesizer.

4.3.4 Single Buffering Analysis

Fig.4.11 shows the TMS32010 software structure for the LPC analyser. It can be seen that the TMS32010 analyser program was divided into a foreground routine and a background routine.

The background routine was mainly responsible for the data overlapping process, the LPC analysis and the transmission of the LPC parameters to the synthesizer whereas the foreground routine was responsible for handling input speech samples.

The foreground routine was actually an interrupt handling subroutine and was activated by an interrupt from the A/D converter every 125 μs (i.e. the sampling frequency was at 8 KHz). It can be seen from Fig.4.11 that a single buffering scheme was used for data handling because the LPC analysis was operated on a 25 ms block basis. At the start of the analyser program, Buffer A1 would be cleared and the foreground routine would start inputting speech samples into the buffer. At the same time, the background routine would start functioning. The data overlapping process was accomplished by inserting the last 20 samples of the previous frame (stored in program memory >F00 to >F13) as the first 20 samples of the present frame (program memory >F1C to >F2F). Then the data in program memory >F1C to >FF7 (i.e. overlapping data and content of Buffer A2) would be analysed and the extracted LPC parameters would then be transmitted to the LPC synthesizer using simple interrupt-handshaking technique. Then the background routine would enter an idle state so as to wait until Buffer A1 was full. As the background routine resumed its operation from the idle state, the last 20 samples of Buffer A2 would be stored into program memory >F00 to >F13 for the data overlapping process and Buffer A2 would be loaded with the content of Buffer A1. The foreground routine would then be initialized and the whole process would repeat.

The analyser program was written in structural form so that each subroutine could be tested individually and any amendments to the LPC algorithm would not be difficult.

4.4 THE LPC SYNTHESIZER

This section describes the TMS32010 subroutines for the LPC synthesizer. The algorithms of the subroutines are based on the procedures described in Section 3.4. The synthesis procedure was re-organised so as to avoid arithmetic overflow and to speed up the synthesis process. Finally the technique of double buffering / pitch synchronous synthesis is described.

4.4.1 The Lattice Filter Subroutine

The TMS32010 lattice filter subroutine was based on the one multiplier 10th order lattice filter as shown in Fig.3.24. It can be seen that the operation of the filter merely involves multiplications and additions. The parameter G received from the LPC analyser, was scaled by a factor of 0.032 (i.e. $\beta = 0.032$ in Eq.(3.24))before it was used to control the intensity of the filter output instead of the intensity of the excitation. The reason for this re-arrangement was to avoid arithmetic overflow during the calculation of the filter intermediate variables. Fig.4.12 shows the flowchart of the lattice filter subroutine. It can be seen that the subroutine includes the radiation network and is a one sample process.

4.4.2 The Voiced Excitation Synthesis Subroutine

The voiced excitation source of the TMS32010 LPC synthesizer was based on the glottal pulse generator as described in Section 3.4.2. The glottal pulse as shown in Fig.3.26a was scaled by a factor of 100 before being input to the vocal tract lattice filter so that the filter could produce sufficient output level. Fig.4.13 shows the flowchart of the routine which operates the voiced excitation LPC synthesis for one pitch period.

4.4.3 The Unvoiced Excitation Synthesis Subroutine

The unvoiced excitation source of the TMS32010 LPC synthesizer was based on the random number generator as shown in Fig.3.29. A 16-bit register RNDREG in the TMS32010 data memory was used as the shift register and it was preset to 1 in the initialization procedure. Fig.4.14 shows the flowchart of the random noise generator subroutine which produces a noise sample and Fig.4.15 shows the flowchart of the routine which operates the unvoiced excitation LPC synthesis for one sample. It can be seen that the output of the noise generator, NOISE, was scaled by a factor of 0.015 before it was input to the lattice filter so as to avoid arithmetic overflow during the calculation of the filter intermediate variables.

4.4.4 Doubling Buffering/Pitch Synchronous Synthesis

Fig.4.16 shows the TMS32010 software structure for the LPC synthesizer. It can be seen that the synthesizer program was divided into a foreground routine and a background routine. The background routine was mainly responsible for the LPC synthesis whereas the foreground routine was responsible for handling incoming LPC parameters received from the analyser. The pitch synchronous synthesis technique was used for the voiced excitation synthesis and therefore a double buffering scheme was employed for the updating procedure of the LPC parameters. Three buffer zones, S1, S2 and S3, each consisting of 12 locations in the TMS32010 data memory, were used for the double buffering scheme. Initially the background routine would operate LPC synthesis using the LPC parameters stored in Buffer S3, and the Flag UF was set to 1. When the synthesizer was connected to the analyser, the synthesizer would receive 12 LPC parameters from the analyser every 25 ms. The foreground of the synthesizer would store the incoming parameters in Buffer S1. After a whole set of parameters (i.e. the pitch, the gain and the reflection coefficients) was received and stored in Buffer S1, the foreground routine would then load Buffer S2 with the content of Buffer S1 and set UF to 0. The background routine would check the status of UF after having completed one voiced excitation synthesis routine or one unvoiced excitation synthesis routine. If the status of UF was detected as 0, the LPC parameters in Buffer S3 would be updated with the parameters stored in Buffer S2 and UF would be reset to 1. The synthesis procedure would then start again. Fig.4.17 and Fig.4.18 show the flowcharts of the background and foreground routines respectively. It can be seen that the synthesizer would keep on synthesising speech using the same set of LPC parameters until another set of parameters was received. Hence this LPC synthesizer could also operate properly when a silence compression scheme is applied to the transmission strategy of the LPC parameters.

4.5 THE LPC IMPLEMENTATION EXPERIMENT

The equipment used in the LPC real-time implementation experiment consisted of two TMS32010 Evaluation Modules (EVM) two TMS32010 Analog Interface Boards (AIB) (21), one audio tape recorder, one loudspeaker and one terminal. The interconnections between these items are shown in Fig.4.19. The two EVMs were connected in a Master/slave configuration so that the terminal could control the Master EVM (LPC analyser) via the terminal emulator mode and the Slave EVM (LPC synthesizer) via the transparency mode. Each EVM was connected to an AIB via an emulation cable. The AIB consists of one analog to digital conversion channel, one digital to analog conversion channel, two 16-bit input buffers and one 16-bit output buffer. Recorded speech stored in the audio tape recorder was input to the LPC analyser (Master EVM) via the analyser's AIB. The extracted LPC parameters were then passed to the LPC synthesizer (Slave EVM) through the AIBs' 16-bit output and input buffers. The LPC synthesizer would then use the parameters to reconstruct the original speech and the synthesized speech was output to the loudspeaker via the synthesizer's AIB.

According to informal subjective listening tests, it was found that the synthesized speech was highly intelligible, but with machine-like quality. Distortion was significant at speech segments with long pitch period as we have discussed in Section 3.5. It was also found that voiced fricative speech was not well synthesised. This was mainly due to the simple dichotomy of the voiced/unvoiced excitation employed in the LPC synthesizer. However, despite the above limitations, the performance of the single channel LPC coder was judged to be satisfactory as far as low-noise clear spoken speech was concerned. Therefore it is believed that if input speech to the LPC analyser were preprocessed so as to remove background noise, the quality of the synthesised speech would be greatly improved.

CHAPTER FIVE - CONCLUSION AND SUGGESTIONS FOR FURTHER WORK

5.1 INTRODUCTION

At present, TMS32010 software has been developed for real-time implementation of linear predictive coding of speech signals. However, due to the "one (TMS32010) chip for analysis and one chip for synthesis" structure of the LPC coder and the limitations of the TMS32010 processor, crude approximations were made in the estimation of the gain of the lattice filter and smoothing procedures could not be applied in the pitch detection process. These shortcomings lead to the degradation of the quality of the synthesised speech. The solution for this problem is a multi (TMS32010) chip structure for the LPC coder. However, this would involve complicated timing problems and the resulting coder would be comparatively expensive. Therefore as far as cost is concerned a "one chip for analysis and one chip for synthesis" structure seems to be practical for an LPC coder. In the remaining sections of this chapter, the limitations of TMS32010 in implementing LPC of speech signals are discussed and the original design of an LPC voice coder for a Cambridge Ring based on this research is outlined.

5.2 LIMITATIONS OF TMS32010 IN IMPLEMENTING LPC OF SPEECH SIGNALS

In the TMS32010 analysis program, input speech signals are analysed on block basis and the duration of each block is 25 ms. The complete analysis procedure (i.e. the pitch detection, the reflection coefficient estimation, the gain estimation and the V/UV decision), the data input subroutine and the parameters transmission subroutine consume a total time of 21 ms which is 84% of an analysis interval. This means that there is only 4 ms left for parameters coding and packing subroutines if 2.4 k bit/sec transmission rate is desired. It is obvious that there is no room to implement more sophisticated LPC analysis procedure as long as the LPC coder has a "one chip for analysis and one chip for synthesis" structure. Even though a multi-chip structure may be proposed for an LPC coder so that more sophisticated algorithms may be implemented, the limitations of TMS32010 in implementing LPC algorithms on speech signals must be considered when designing such a system.

One of the reasons why the TMS32010 LPC analysis procedure consumes so much time (84% of an analysis interval) is that the TMS32010 data memory is not large enough. Although data can be stored in TMS32010 program memory, TMS32010 programs can only perform arithmetic operations with operands stored in TMS32010 data memory. The size of TMS32010 data memory is just 144 words x 16 bits which is smaller than the size of an analysis interval (200 samples). Therefore, during the LPC analysis (especially the pitch detection process), blocks of data were transferred between the TMS32010 program memory and data memory. Unfortunately this kind of data transfer is very time consuming. It takes 3 instruction cycles to complete one transfer either from program memory to data memory or vice versa.

Another factor which prolongs the analysis time is that the TMS32010 only provides two auxiliary registers, ARØ and AR1. These two registers can be used as loop counters and/or data pointers for recursive procedures. However, the number of auxiliary registers is not enough for some complex recursion processes such as the Le Roux and Gueguen procedure. Therefore during these processes, some locations of the TMS32010 data memory were used as loop counters and data pointers. In this way, however, these loop counters and data pointers do not have the advantage of autoincrement and autodecrement facilities as ARØ and AR1 do. The counters and pointers, however, must be incremented or decremented after one recursive loop and this consumes 2 instruction cycles for every increment/decrement process. The time spent on these updating procedures could be very considerable if the order of the loop is large and especially when nested loops are involved.

The TMS32010 can be considered as a general-purpose microprocessor with special instructions for digital signal processing. However, it only provides one single-vectored hardware interrupt (INIT) and one software interrupt (BIO). This can only support simple input/output functions so that in the LPC implementation experiment, both the analyser and synthesizer used up all interrupt lines available for data input/output and parameters transfer. Therefore for a practical LPC coder where LPC parameters are transmitted in a serial manner (i.e. bit by bit), it is suggested that the TMS32010 processors should be incorporated with a host processor (e.g. 8086) in such a way that the host processor handles all input/output operations and LPC parameters transfer whereas the TMS32010 processors only perform the LPC analysis and synthesis.

5.3 ORIGINAL DESIGN OF AN LPC VOICE CODING SERVER FOR A CAMBRIDGE RING

Fig.5.1 shows one possible hardware configuration to implement the LPC vocoder using the Texas Instrument Technology on a Cambridge Ring. The interface between the vocoder unit and the Cambridge Ring is the VMI-1 which already exists. The LPC vocoder unit consists of 4 major parts, namely the I/O board, the 8086 host computer, and two TMS32010 processors each with 4K x 16 program memory. The 8086 controls data flow between the I/O board, the TMS32010 processors and the

VMI-1 via the Intel Multi-Bus. The software of the 8086 and the design of the actual hardware circuit depend on the function of each item of the vocoder.

The TMS32010 LPC programs should first be stored in a ROM which can be accessed by the 8086. After the vocoder unit is reset, the 8086 should be able to transfer the LPC program in the ROM to the program memory of the TMS32010 processors so that one TMS32010 operates the LPC analysis and the other operates the LPC synthesis. The I/O board, after being initialized, should be able to sample incoming speech at 8 KHz and generates a 16-bit linear PCM code for each sample. The 8086 stores the samples in its main memory temporarily until 200 samples have been received. Then the whole block of data is transferred onto the program memory of the LPC analyser. The analyser TMS32010 does the LPC analysis and places the fixed point parameters into the program memory buffer. The 8086 then accesses the parameters, encodes and packs them into Basic Blocks (BBs). The LPC BBs are then transmitted to the distance vocoder unit through the VMI-1 interface. The distance vocoder unit should have the same configuration as in Fig.5.1 so that its 8086, after having received the LPC BBs, should be able to unpack and decode the parameters. The parameters are then transferred onto the program memory of the LPC synthesizer. The synthesizer TMS32010 accesses the parameters which are then used to produce synthesized speech samples from the LPC lattice filter. The synthesised speech data is stored in the program memory buffer. The 8086 accesses the processed speech and transfers the data to the I/O board for analog reconstruction. Since the two vocoder units have the same configuration, full-duplex speech communication is accomplished.

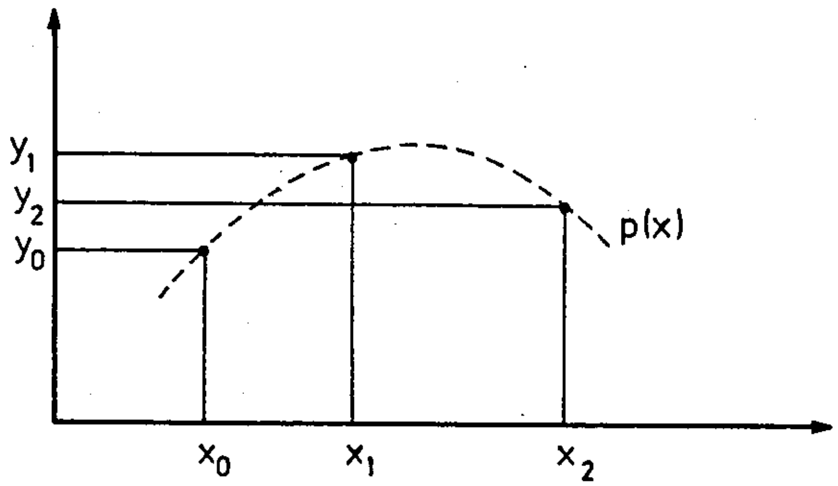
5.4 CONCLUDING REMARKS

The LPC voice server as depicted in Fig.5.1 was actually designed before the beginning of the present work. However, during the course of the present work, it was found that the VMI-1 interface would not operate in the duplex mode. Therefore the construction of the LPC voice coding server for the Cambridge Ring will have to be abandoned unless another interface is built.

Although the LPC voice coding server is unlikely to be built, TMS32010 software has been developed to implement the LPC vocoder algorithm in real-time. The algorithm is especially suitable to implement 2.4K bit/s LPC. Due to the compact size of TMS32010, the dimensions of the vocoder unit as depicted in Fig.5.1 would be much smaller than a conventional vocoder unit. Therefore the TMS32010 vocoder system is very suitable for mobile communication. Actually the TMS32010 vocoder unit can be interfaced to other types of communication channel such as H.F. links,telephone lines or cellular radio network. The operation of the vocoder unit would be the same as described in section 5.3.

APPENDIX I

A1 QUADRATIC INTERPOLATION (22)



Consider three points (x_o, y_o), (x_1, y_1) and (x_2, y_2) on the x-y co-ordinate. The quadratic interpolation equation p(x) which passes through the three points can be determined by the expression:

$$p(x) = y_o\ \theta_o(x) + y_1\ \theta_1(x) + y_2\ \theta_2(x) \qquad \text{(A1.1)}$$

where

$$\theta_o(x) = \frac{(x-x_1)\ (x-x_2)}{(x_o-x_1)\ (x_o-x_2)} \qquad \text{(A1.2)}$$

$$\theta_1(x) = \frac{(x-x_o)\ (x-x_2)}{(x_1-x_o)\ (x_1-x_2)} \qquad \text{(A1.3)}$$

$$\theta_2(x) = \frac{(x-x_o)\ (x-x_1)}{(x_2-x_o)\ (x_2-x_1)} \qquad \text{(A1.4)}$$

APPENDIX 2

A2 THE TMS32010 SOFTWARE DEVELOPMENT SYSTEM PROGRAM LISTING

```
 10 REM********************************
 20 REM    TMS32010-BBC COMMUNICATION
 30 REM
 40 REM           D.S.F.CHAN
 50 REM
 60 REM********************************
 65
 70 MODE 3
 80 REPEAT
 90 PROCINIT
100 INPUT "COMMAND (S/L/F/P/C): ",A$
110 IF A$="S"THEN PROCSAVE
120 IF A$="L"THEN PROCLOAD
130 IF A$="F"THEN *CAT
140 IF A$="P"THEN PROCPRINT
150 IF A$="C"THEN GOTO 170
160 UNTIL FALSE
170 END
180
190 DEF PROCINIT
200 *FX2,1
210 *FX3,1
220 *FX7,4
230 *FX8,4
240 *FX229,1
250 OSBYTE=&FFF4
260 ENDPROC
270
280 DEF PROCSAVE
290 DIM START 1000
300 FOR I%=0 TO 2 STEP2:P%=START
310 [OPTI%
320 .LOOP1
330 CLD
340 LDX £254
350 LDA £128
360 JSR &FFF4
370 CPX £0
380 BEQ LOOP1
390 LDX £1
400 LDA £145
410 JSR &FFF4
420 CPY £62
430 BEQ LOOP3
440 JMP LOOP1
450 .LOOP3
```

```
460 TYA
470 LDY &70
480 JSR &FFD4
490 TAY
500 CPY £60
510 BEQ LOOP6
520 .LOOP4
530 CLD
540 LDX £254
550 LDA £128
560 JSR &FFF4
570 CPX £0
580 BEQ LOOP4
590 LDX £1
600 LDA £145
610 JSR &FFF4
620 JMP LOOP3
630 .LOOP6
640 LDY &600
650 RTS:]NEXT I%
660 INPUT "FILENAME: ",FILE$
670 IF RIGHT$(FILE$,4)="HELP" THEN GOTO 730
680 Y%=OPENOUT (FILE$)
690 ?&70=Y%
700 CALL START
710 CLOSE £Y%
720 PRINT
730 ENDPROC
740
750 DEF PROCLOAD
760 INPUT "FILENAME: ",FILE$
770 IF RIGHT$(FILE$,4)="HELP" THEN GOTO 890
780 Y=OPENIN (FILE$)
790 IF ADVAL(-2)>0 THEN B%=GET
800 IF B%<>13 THEN GOTO 790
810 A%=138:X%=2:J=0
820 REPEAT
830 IF ADVAL(-3)>0 THEN B%=BGET£Y:Y%=B%:CALL OSBYTE:J=J+1
840 IF J>400 THEN FOR Z=1 TO 5000:NEXT Z:J=0
850 UNTIL B%=60
860 Y%=13:CALL OSBYTE
870 Y%=10:CALL OSBYTE
880 CLOSE£Y
890 ENDPROC
900
910 DEF PROCPRINT
920 INPUT "SELECT VDU/PRINT/FILE/CONTROL(V/P/F/C): ",C$
930 IF C$="C" THEN GOTO 1090
940 IF C$="F" THEN *CAT
950 IF C$="F" THEN GOTO 920
960 INPUT "FILENAME: ",FILE$
970 IF RIGHT$(FILE$,4)="HELP" THEN GOTO 1090
980 IF C$="P" THEN VDU 2
990 Y=OPENIN(FILE$)
1000 PRINT:PRINT"FILE: ",FILE$:PRINT
1010 REPEAT
1020 B%=BGET£Y
1030 VDU B%
1040 UNTIL B%=60
1050 VDU 10:VDU 13
1060 CLOSE £Y
1070 VDU3
1080 GOTO 920
1090 ENDPROC
```

APPENDIX 3

A3 THE LPC SIMULATION PROGRAM LISTING

```
 10 REM**************************
 20 REM LPC ANALYSIS SIMULATION
 30 REM
 40 REM     D.S.F.CHAN
 50 REM
 60 REM**************************
 70
 80 MODE 3
 90 CLS
100 PROCINIT
110 PTR£IN=400*STARTBK
120
130 FOR BLOCK=STARTBK TO ENDBK
140 PROCINPUT
150
160 REM**************************
170 REM  REF-COEFF AND GAIN
180 REM**************************
190
200 PROCSTORELAP
210 PROCPREEMP
220 PROCWINDOW
230 PROCACORR
240 PROCENERGY
250 PROCLANDG
260 PROCOVERLAP
270
280 REM**************************
290 REM  PITCH  DETECTION
300 REM**************************
310
320 PROCCLEAR
330
340 FOR I=0 TO 198 STEP 2
350 FIN=P(I)
360 PROCFILTER
370 FIN=P(I+1)
380 PROCFILTER
390 PA(I/2)=FOUT
400 NEXT I
410
420 PROCDCCUT
430 PROCTHRHLD
440 PROCCLIP
450 PROCCORR
460 PROCPEAK
```

```
470 PROCINTERP
480
490 REM***************************
500 REM  OUTPUT LPC10 PARAMETERS
510 REM***************************
520
530 PROCOUTPUT
540
550 NEXT BLOCK
560 PROCCLOSE
570 END
580
590
600 DEF PROCINIT
610 DIM W(220),A(220),AR(10),K(10),X(30),OL(20)
620 DIM P(200),PA(100),PR(100)
630
640 A1=1.45902906:A2=-0.910368999:A3=0.197825187
650 B0=0.0316893439:B1=0.0950680317:B2=0.0950680317:B3=0.031689
   3439
660
670
680 EMPREF=0
690 FOR I=0 TO 19
700 A(I)=0
710 NEXT I
720
730 PRINT:INPUT"SOURCE FILENAME:",F$
740 PRINT:INPUT"STARTING BLOCK:",STARTBK
750 PRINT:INPUT"ENDING BLOCK:",ENDBK
760
770 IN=OPENIN(F$)
780 *DR.1
790 RC=OPENOUT(LEFT$(F$,3)+".RC")
800 G=OPENOUT(LEFT$(F$,3)+".G")
810 PIT=OPENOUT(LEFT$(F$,3)+".PIT")
820 *DR.0
830
840 FOR I=0 TO 219
850 W(I)=0.5*(1-COS(2*PI*I/219))
860 NEXT I
870
880 ENDPROC
890
900
910 DEF PROCINPUT
920 FOR I=0 TO 199
930 A=BGET£IN
940 B=BGET£IN
950 SAMPLE=A*64+B-2050
960 P(I)=SAMPLE
970 A(I+20)=SAMPLE
980 NEXT I
990 ENDPROC
1000
1010
1020 DEF PROCSTORELAP
1030 FOR I=0 TO 19
1040 OL(I)=A(I+200)
1050 NEXT I
1060 ENDPROC
1070
1080
1090 DEF PROCPREEMP
1100 FOR I=0 TO 219
1110 PRE=A(I)-0.95*EMPREF
```

```
1120 EMPREF=A(I)
1130 A(I)=PRE
1140 NEXT I
1150 ENDPROC
1160
1170
1180 DEF PROCWINDOW
1190 FOR I=0 TO 219
1200 A(I)=A(I)*W(I)
1210 NEXT I
1220 ENDPROC
1230
1240
1250 DEF PROCACORR
1260 FOR I=0 TO 10
1270 AR(I)=0
1280 FOR J=0 TO 219-I
1290 AR(I)=AR(I)+A(J)*A(J+I)
1300 NEXT J
1310 NEXT I
1320 ENDPROC
1330
1340
1350 DEF PROCENERGY
1360 GAIN=SQR(AR(0))
1370 ENDPROC
1380
1390
1400 DEF PROCLANDG
1410 FOR I=10 TO 0 STEP -1
1420 AR(I)=AR(I)/AR(0)
1430 NEXT I
1440 X(0)=AR(0)
1450 X(21)=0
1460 FOR J=1 TO 10
1470 X(2*J-1)=AR(J)
1480 X(2*J)=AR(J)
1490 NEXT J
1500 FOR J=1 TO 10
1510 K(J)=-X(1)/X(0)
1520 IF J=10 THEN ENDPROC
1530 FOR I=0 TO 2*(10-J) STEP 2
1540 X(I)=X(I)+K(J)*X(I+1)
1550 X(I+1)=K(J)*X(I+2)+X(I+3)
1560 NEXT I
1570 NEXT J
1580 ENDPROC
1590
1600
1610 DEF PROCOVERLAP
1620 FOR I=0 TO 19
1630 A(I)=OL(I)
1640 NEXT I
1650 ENDPROC
1660
1670 DEF PROCCLEAR
1680 D1=0:D2=0:D3=0:D4=0
1690 ENDPROC
1700
1710 DEF PROCFILTER
1720 FB=D2*A1+D3*A2+D4*A3
1730 D1=FB+FIN
1740 FOUT=B0*D1+B1*D2+B2*D3+B3*D4
1750 D4=D3:D3=D2:D2=D1
1760 ENDPROC
1770
```

```
1780 DEF PROCDCCUT
1790 OS=0
1800 FOR I=0 TO 99
1810 OS=OS+PA(I)
1820 NEXT I
1830 OS=OS/100
1840 FOR I=0 TO 99
1850 PA(I)=PA(I)-OS
1860 NEXT I
1870 ENDPROC
1880
1890 DEF PROCTHRHLD
1900 TH1=0:TH2=0:TH3=0
1910 FOR I=0 TO 33
1920 IF ABS(PA(I))>TH1 THEN TH1=ABS(PA(I))
1930 NEXT I
1940 FOR I=34 TO 66
1950 IF ABS(PA(I))>TH2 THEN TH2=ABS(PA(I))
1960 NEXT I
1970 FOR I=67 TO 99
1980 IF ABS(PA(I))>TH3 THEN TH3=ABS(PA(I))
1990 NEXT I
2000 THRE=TH1
2010 IF TH2<THRE THEN THRE=TH2
2020 IF TH3<THRE THEN THRE=TH3
2030 ENDPROC
2040
2050 DEF PROCCLIP
2060 THRE=THRE*0.6
2070 FOR I=0 TO 99
2080 IF ABS(PA(I))<=THRE THEN PA(I)=0 ELSE PA(I)=5*SGN(PA(I))
2090 NEXT I
2100 ENDPROC
2110
2120 DEF PROCCORR
2130 FOR J=0 TO 99
2140 PR(J)=0
2150 FOR I=0 TO 99-J
2160 PR(J)=PR(J)+PA(I)*PA(I+J)
2170 NEXT I
2180 NEXT J
2190 ENDPROC
2200
2210 DEF PROCPEAK
2220 P1=0:RXX=0
2230 FOR J=15 TO 99
2240 IF PR(J)>RXX THEN P1=J:RXX=PR(J)
2250 NEXT J
2260 ENDPROC
2270
2280 DEF PROCINTERP
2290 Y0=PR(P1-1):Y1=PR(P1):Y2=PR(P1+1)
2300 COMP1=2*(Y0-Y2)
2310 COMP2=(Y0-2*Y1+Y2)
2320 XX=0
2330 IF COMP1=0 THEN XX=0:GOTO 2370
2340 IF COMP2=0 THEN XX=0:GOTO 2370
2350 IF COMP1-COMP2>=0 THEN XX=1:GOTO 2370
2360 IF COMP1+COMP2<=0 THEN XX=-1:GOTO 2370
2370 PITCH=2*P1+XX
2380 ENDPROC
2390
2400 DEF PROCOUTPUT
2410 *DR.1
2420 PRINT
2430 PRINT"BLOCK=";BLOCK
```

```
2440 PRINT"GAIN=";GAIN
2450 PRINT£G,BLOCK:PRINT£G,GAIN
2460 PRINT"PITCH=";PITCH
2470 PRINT£PIT,BLOCK:PRINT£PIT,PITCH
2480 PRINT£RC,BLOCK
2490 FOR I=1 TO 10
2500 PRINT"K(";I;")=";K(I)
2510 PRINT£RC,K(I)
2520 NEXT I
2530 *DR.0
2540 ENDPROC
2550
2560 DEF PROCCLOSE
2570 CLOSE£IN
2580 *DR.1
2590 CLOSE£G
2600 CLOSE£RC
2610 CLOSE£PIT
2620 *DR.0
2630 ENDPROC
```

```
 10 REM********************************
 20 REM
 30 REM   LPC SYNTHESIS SIMULATION
 40 REM
 50 REM   D.S.F.CHAN
 60 REM
 70 REM********************************
 80
 90 MODE3
100 PROCINIT
110 UF=1:PROCINPUT
120 K=1
130 IF UF=1 THEN PROCUPDATE:UF=0
140 IF PPITCH=0 THEN PROCUNVOICE:GOTO 130
150 FOR Z=0 TO 20
160 IN=GP(Z)*100:PROCLATTICE:PROCOUTPUT:K=K+1
170 NEXT Z
180 IF K>200 THEN PROCINPUT:UF=1:K=1
190 P=PPITCH-20
200 IN=0:PROCLATTICE:PROCOUTPUT:K=K+1:P=P-1
210 IF K>200 THEN PROCINPUT:UF=1:K=1
220 IF P<0 THEN GOTO 130 ELSE GOTO 200
230 END
240
250 DEF PROCINIT
260 DIM F(10),G(10),D(10),NK(10),PK(10),GP(20)
270 FOR I=1 TO 10
280 F(I)=0:G(I)=0:D(I)=0:NK(I)=0:PK(I)=0
290 NEXT I
300 AMAX=0:TEMP=0:RNDREG=1
310 FOR I=0 TO 14
320 GP(I)=0.5*(1-COS(PI*I/14))
330 NEXT I
340 FOR I=15 TO 20
350 GP(I)=COS(PI*(I-14)/12)
360 NEXT I
370
380 INPUT"INPUT SYNTHESIS FILENAME",F$
390 SPOUT=OPENOUT(LEFT$(F$,3)+"/OUT")
400 *DR.1
410 G=OPENIN(LEFT$(F$,3)+".G")
420 RC=OPENIN(LEFT$(F$,3)+".RC")
430 PIT=OPENIN(LEFT$(F$,3)+".PIT")
440 *DR.0
450 ENDPROC
460
470 DEF PROCINPUT
480 *DR.1
490 INPUT£G,B:INPUT£G,NGAIN
500 INPUT£PIT,B:INPUT£PIT,NPITCH
510 INPUT£RC,B
520 FOR I=1 TO 10
530 INPUT£RC,NK(I)
540 NEXT I
550 *DR.0
560 IF NK(1)>0.15 THEN NPITCH=0:NGAIN=NGAIN/4900 ELSE NGAIN=NGA
   IN*SQR(NPITCH)/500
570 PRINT "B=";B;" P=";NPITCH;" G=";NGAIN;" K=";NK(1);" M=";INT
```

```
    (AMAX)
 580 ENDPROC
 590
 600 DEF PROCLATTICE
 610 F(1)=IN+D(1)
 620 FOR I=2 TO 10
 630 F(I)=F(I-1)+PK(I-1)*(F(I-1)+D(I))
 640 NEXT I
 650 OUT=(F(10)+F(10)*PK(10))*PGAIN
 660 FOR I=1 TO 9
 670 G(I)=-(F(I)+D(I+1))*PK(I)+D(I+1)
 680 NEXT I
 690 G(10)=-F(10)*PK(10)
 700 FOR I=1 TO 10
 710 D(I)=G(I)
 720 NEXT I
 730 ENDPROC
 740
 750 DEF PROCOUTPUT
 760 SOUT=OUT-TEMP
 770 TEMP=OUT
 780 DD=SOUT
 790 IF ABS(DD)>AMAX THEN AMAX=ABS(DD)
 800 IF AMAX>2040 THEN PRINT"                                  OVERFLOW!!
    !!!!!!!"
 810 DD=DD+2050
 820 A=DD MOD 64
 830 B=DD DIV 64
 840 BPUT£SPOUT,B
 850 BPUT£SPOUT,A
 860 ENDPROC
 870
 880 DEF PROCUPDATE
 890 PPITCH=NPITCH
 900 PGAIN=NGAIN
 910 FOR I=1 TO 10
 920 PK(I)=NK(I)
 930 NEXT I
 940 ENDPROC
 950
 960 DEF PROCUNVOICE
 970 REPEAT
 980 PROCNOISE
 990 IN=NOISE
1000 PROCLATTICE
1010 PROCOUTPUT
1020 K=K+1
1030 UNTIL K>200
1040 PROCINPUT
1050 UF=1:K=1
1060 ENDPROC
1070
1080 DEF PROCNOISE
1090 RNDIN=RNDREG AND &00000001
1100 RNDOUT=RNDREG AND &00000200
1110 RNDOUT=RNDOUT/(2^9)
1120 RNDOUT=RNDOUT EOR RNDIN
1130 RNDOUT=RNDOUT*(2^11)
1140 RNDREG=RNDREG+RNDOUT
1150 RNDREG=RNDREG AND &0000FFFE
1160 RNDREG=RNDREG/2
1170 NOISE=(1024-RNDREG)*100/1024
1180 ENDPROC
1190
1200
```

APPENDIX 4

A4 THE LPC TMS32010 PROGRAM LISTING

FILE: SAMWIN

```
*
************************************
*
*        TMS32010  LPC  ANALYSER
*
*            D.S.F.CHAN
*
************************************
*
         AORG  >F1C
         DATA  -39,-29,-1,28,37,10,-24,-53,-78,-93
         DATA  -89,-82,-79,-96,-150,-204,-256,-304,-325,-360
         DATA  -353,-169,31,66,87,100,143,287,421,428
         DATA  439,448,338,273,244,202,217,216,130,59
         DATA  32,-15,-22,-26,-64,-80,-112,-136,-135,-121
         DATA  -95,-57,-13,-2,23,44,50,54,22,-32
         DATA  -77,-90,-78,-53,-43,-41,-64,-90,-99,-126
         DATA  -149,-164,-184,-199,-209,-232,-285,-348,-377,-2
         DATA  -17,71,135,156,147,239,347,359,413,432
         DATA  320,256,224,197,231,228,164,114,107,80
         DATA  50,18,-56,-106,-139,-140,-109,-83,-61,-49
         DATA  -64,-69,-58,-61,-43,-36,-55,-62,-63,-45
         DATA  -11,9,10,-5,-28,-63,-100,-143,-172,-186
         DATA  -189,-199,-232,-261,-288,-313,-257,-97,-13,13
         DATA  75,97,167,295,348,371,410,376,288,253
         DATA  246,255,281,240,147,90,40,-13,-25,-38
         DATA  -33,-11,-26,-37,-54,-82,-77,-68,-61,-63
         DATA  -80,-110,-121,-119,-93,-33,19,47,22,-40
         DATA  -90,-105,-95,-85,-87,-116,-129,-135,-148,-158
         DATA  -186,-219,-256,-293,-225,-89,-17,47,111,147
         DATA  231,299,301,331,358,347,320,275,230,205
         DATA  182,144,123,128,115,72,17,-16,-34,-37
*
*************************
* WINDOW FUNCTION
*************************
*
         AORG  >E20
         DATA  0,1,3,8,13,21,30,41,54,68
         DATA  84,101,120,141,163,187,212,239,267,297
         DATA  328,361,395,430,467,505,544,584,626,669
         DATA  713,758,804,851,900,949,999,1050,1101,1154
         DATA  1207,1261,1315,1371,1426,1483,1539,1596,1654,171
```

```
        DATA  1770,1828,1887,1945,2004,2063,2121,2180,2239,229
        DATA  2355,2413,2471,2528,2585,2642,2698,2753,2808,286
        DATA  2916,2969,3021,3072,3122,3172,3221,3268,3315,336
        DATA  3405,3449,3491,3532,3572,3611,3648,3684,3719,375
        DATA  3784,3814,3843,3871,3897,3921,3944,3966,3986,400
        DATA  4020,4035,4049,4060,4071,4079,4086,4091,4094,409
        DATA  4096,4094,4091,4086,4079,4071,4060,4049,4035,402
        DATA  4004,3986,3966,3944,3921,3897,3871,3843,3814,378
        DATA  3752,3719,3684,3648,3611,3572,3532,3491,3449,340
        DATA  3361,3315,3268,3221,3172,3122,3072,3021,2969,291
        DATA  2862,2808,2753,2698,2642,2585,2528,2471,2413,235
        DATA  2297,2239,2180,2121,2063,2004,1945,1887,1828,177
        DATA  1712,1654,1596,1539,1483,1426,1371,1315,1261,120
        DATA  1154,1101,1050,999,949,900,851,804,758,713
        DATA  669,626,584,544,505,467,430,395,361,328
        DATA  297,267,239,212,187,163,141,120,101,84
        DATA  68,54,41,30,21,13,8,3,1,0
*
*************************
* FILTER COEFFICIENTS
* A1..A3 BO......B3
*************************
*
        AORG  >D90
        DATA  5976,-3728,810,129,389,389,129
*
*
<
```

FILE: PITMAIN

```
***************************************
* PITCH DETECTION TESTING PROGRAM
***************************************
*
*************************
*INTERRUPT ADDRESS ASG.
*************************
*
INTDAT  EQU   >7B
INTMSK  EQU   >7C
INTNDT  EQU   >7D
INTPMA  EQU   >7E
UNITY   EQU   >7F
*
INTSTU  EQU   >0
INTACH  EQU   >1
INTACL  EQU   >2
*
*************************
* PROGRAM MEMORY MAPPING
*************************
*
FE4     EQU   >FE4
F1C     EQU   >F1C
F00     EQU   >F00
F30     EQU   >F30
E20     EQU   >E20
DB0     EQU   >DB0
D90     EQU   >D90
CC0     EQU   >CC0
CA0     EQU   >CA0
CA1     EQU   >CA1
CA2     EQU   >CA2
CAB     EQU   >CAB
*
```

```
*
******************************
* INITIALIZATION
******************************
*
         AORG   >0
         B      START
         B      INTSUR
*
         AORG   >A
*
CNTRL    EQU    >0
CLCK     EQU    >1
MODE     EQU    >E
SAMRAT   EQU    4095
MASK     EQU    >7FF
*
START    DINT
         CALL   INTIAL
FRAME    DINT
         CALL   AGAIN
         EINT
*
*
****************************
*   DECIMATION
****************************
*
PFIN     EQU    >64
PA1      EQU    >65
PA2      EQU    >66
PA3      EQU    >67
PB0      EQU    >68
PB1      EQU    >69
PB2      EQU    >6A
PB3      EQU    >6B
PFB      EQU    >6C
PD1      EQU    >6D
PD2      EQU    >6E
PD3      EQU    >6F
PD4      EQU    >70
PFOUT    EQU    >71
PRDADD   EQU    >72
*
         CALL   CLEAR
         CALL   PCOEFF
*
         LARK   0,99
         LARK   1,0
PFDSL    CALL   PDMOV
         CALL   PFILTR
         CALL   PDMOV
         CALL   PFILTR
         CALL   PSTORE
         LARP   0
         BANZ   PFDSL
*
*
******************************
* REMOVAL OF DC OFFSET
******************************
*
POS      EQU    >64
*
         CALL   PDCCUT
*
```

```
00820 *
00830 ******************************
00840 * THRESHOLDING & CENTRE-CLIP
00850 ******************************
00860 *
00870 PTHRE    EQU    >64
00880 PTHR1    EQU    >65
00890 PTHR2    EQU    >66
00900 PTHR3    EQU    >67
00910 *
00920          CALL   PTHRLD
00930          CALL   PCCLIP
00940 *
00950 *
00960 ******************************
00970 * AUTOCORRELATION
00980 ******************************
00990 *
01000 PDELAY   EQU    >64
01010 PDLARO   EQU    >65
01020 PATDAT   EQU    >66
01030 PCORPM   EQU    >67
01040 *
01050          CALL   PCORR
01060          CALL   PCORMV
01070 *
01080 *
01090 ******************************
01100 * PEAK PICKING , INTERPOLATION
01110 * AND PITCH OUTPUT SUBROUTINE
01120 ******************************
01130 *
01140 PRXX     EQU    >64
01150 PP1      EQU    >65
01160 PY0      EQU    >66
01170 PY1      EQU    >67
01180 PY2      EQU    >68
01190 PCOMP1   EQU    >69
01200 PCOMP2   EQU    >6A
01210 PXX      EQU    >6B
01220 PPITCH   EQU    >6C
01230 *
01240          CALL   PPEAK
01250          CALL   PINTRP
01260          CALL   POUT
01270 *
01280          NOP
01281          NOP
01282          NOP
<
FILE:     ANYMAIN


00010 *************************************
00020 * LPC10 ANALYSIS TESTING PROGRAM
00030 *************************************
00040 *
00430 ******************************
00440 * STORE OVERLAPPING DATA
00450 ******************************
00460 *
00470          CALL   CLEAR
00480          CALL   ASTOVL
00490 *
00500 *
00510 ******************************
```

```
* PRE-EMPHASIS , WINDOWING
* SHIFTING , AUTOCORRELATION
******************************
*
AROH     EQU   >0
AROL     EQU   >1
AR10H    EQU   >14
AR10L    EQU   >15
AF1C     EQU   >16
AE20     EQU   >17
AWINDT   EQU   >18
AINPUT   EQU   >19
AEMREF   EQU   >1A
AINSHF   EQU   >1B
AD0      EQU   >1C
AD1      EQU   >1D
AD2      EQU   >1E
AD3      EQU   >1F
AD4      EQU   >20
AD5      EQU   >21
AD6      EQU   >22
AD7      EQU   >23
AD8      EQU   >24
AD9      EQU   >25
AD10     EQU   >26
AAR0     EQU   >79
AAR1     EQU   >7A
*
         CALL  CLEAR
*
         LT    UNITY
         MPYK  F1C
         PAC
         SACL  AF1C
         MPYK  E20
         PAC
         SACL  AE20
*
         LARK  0,219
PPWSAL   CALL  APREMP
         CALL  AWIN
         CALL  ASHFT
         CALL  AACORR
         LARP  0
         BANZ  PPWSAL
*
*
******************************
* CALCULATE SEGMENT GAIN
******************************
*
AA       EQU   >16
AB       EQU   >17
AC       EQU   >18
*
         CALL  AENGRY
*
*
******************************
* PRE-NORMALIZATION (RO...R10)
******************************
*
AR0      EQU   >16
AR1      EQU   >17
AR10     EQU   >20
ACNTER   EQU   >21
```

```
ASFCNT  EQU   >22
AREF    EQU   >23
*
        CALL  APNORM
*
*
*****************************
* NORMALIZATION RI=RI/RO
*****************************
*
ANUMER  EQU   >72
ADENOM  EQU   >73
AQUOT   EQU   >74
ATMSGN  EQU   >75
AMULT1  EQU   >76
AMULT2  EQU   >77
AMANS   EQU   >78
*
        CALL  ANORM
*
*
*******************************
* L AND G ITERATION
*******************************
*
AX0     EQU   >21
AX1     EQU   >22
AX18    EQU   >33
AX20    EQU   >35
AX21    EQU   >36
AK1     EQU   >37
AK10    EQU   >40
AKCNT   EQU   >41
*
        CALL  ALAG
        CALL  AOUT
*
*
*******************************
* RESTORE OVERLAPPING DATA
* AT THE FRONT OF FRAME
*******************************
*
        CALL  ARST
        NOP
        NOP
        NOP
<
FILE:    ENDMAIN


*
****************************************
* TRANSMIT PARAMETERS TO RECEIVER
****************************************
*
        CALL  CLEAR
        CALL  COEFXF
        DINT
        CALL  XMIT
        EINT
*
****************************************
* MOVE NEWFRAME TO >F30.......>FF7
****************************************
*
```

```
NEWP1    EQU    >0
NEWP2    EQU    >1
NEWDAT   EQU    >2
*
MORE     LAC    INTNDT
         BNZ    MORE
         CALL   NEWFRM
         NOP
         NOP
         NOP
         B      FRAME
         NOP
         NOP
         NOP
*
*
<
FILE:     PITSUBR


*
*******************************************************
* PCOEFF SUBROUTINE
* SET UP FILTER COEFF. A1...A3 , B0.......B3
* D1=D2=D3=D4=0 : PUT >F30 INTO PRDADD
*******************************************************
*
PCOEFF   LT     UNITY
         MPYK   D90
         PAC
         LARK   0,6
         LARK   1,PA1
PCOFL1   LARP   1
         TBLR   *+,0
         ADD    UNITY
         BANZ   PCOFL1
*
         ZAC
         SACL   PD1
         SACL   PD2
         SACL   PD3
         SACL   PD4
*
         LT     UNITY
         MPYK   F30
         PAC
         SACL   PRDADD
*
         RET
*
*
****************************************
* PDMOV SUBROUTINE
* MOVE DATA IN PM (PRDADD) INTO DM (PFIN)
****************************************
*
PDMOV    LAC    PRDADD
         TBLR   PFIN
         ADD    UNITY
         SACL   PRDADD
*
         RET
*
*
****************************************
* PSTORE SUBROUTINE
```

```
* STORE PFOUT OF THE LPF INTO DM (AR1)
*************************************
*
PSTORE  LAC   PFOUT
        LARP  1
        SACL  *+
*
        RET
*
*
*************************************
* PFILTR SUBROUTINE
* PASS PFIN --> LPF(A1..A3,B0..B3)
* WITH OUTPUT IN PFOUT
*************************************
*
PFILTR  ZAC
        LT    PD2
        MPY   PA1
        LTA   PD3
        MPY   PA2
        LTA   PD4
        MPY   PA3
        APAC
        SACH  PFB,4
*
        LAC   PFB
        ADD   PFIN
        SACL  PD1
*
        ZAC
        LT    PD4
        MPY   PB3
        LTD   PD3
        MPY   PB2
        LTD   PD2
        MPY   PB1
        LTD   PD1
        MPY   PB0
        APAC
        SACH  PFOUT,4
*
        RET
*
*
**************************************
* PDCCUT SUBROUTINE
* MEAN  OF THE SPEECH SEGMENT REMOVED
**************************************
*
PDCCUT  ZAC
        LARK  0,99
        LARP  0
PDCTL1  ADD   *
        BANZ  PDCTL1
        SACL  PDS
*
        LT    PDS
        MPYK  +41
        PAC
        SACH  PDS,4
*
        LARK  0,99
        LARP  0
PDCTL2  LAC   *
        SUB   PDS
```

```
        SACL   *
        BANZ   PDCTL2
*
        RET
*
*
****************************************
* PTHRLD SUBROUTINE
* FIND MAX(0-33),MAX(34-66),MAX(67-99)
* THRESHOLD=0.6*MAX(SMALLEST)
****************************************
*
PTHRLD  ZAC
        SACL   PTHR1
        SACL   PTHR2
        SACL   PTHR3
*
        LARK   0,33
        LARP   0
PTHRL1  LAC    *
        ABS
        SUB    PTHR1
        BLZ    PTHLT1
        LAC    *
        ABS
        SACL   PTHR1
PTHLT1  BANZ   PTHRL1
*
        LARK   0,66
        LARK   1,32
PTHRL2  LARP   0
        LAC    *
        ABS
        SUB    PTHR2
        BLZ    PTHLT2
        LAC    *
        ABS
        SACL   PTHR2
PTHLT2  MAR    *-
        LARP   1
        BANZ   PTHRL2
*
        LARK   0,99
        LARK   1,32
PTHRL3  LARP   0
        LAC    *
        ABS
        SUB    PTHR3
        BLZ    PTHLT3
        LAC    *
        ABS
        SACL   PTHR3
PTHLT3  MAR    *-
        LARP   1
        BANZ   PTHRL3
*
        LAC    PTHR1
        SACL   PTHRE
        SUB    PTHR2
        BLZ    PTHLT4
        LAC    PTHR2
        SACL   PTHRE
PTHLT4  LAC    PTHRE
        SUB    PTHR3
        BLZ    PTHLT5
        LAC    PTHR3
```

```
                SACL    PTHRE
*
PTHLT5          LT      PTHRE
                MPYK    +2457
                PAC
                SACH    PTHRE,4
*
                RET
*
*
****************************************
* PCCLIP SUBROUTONE
* CENTRE CLIP AI I=0....99 WITH PTHRE
****************************************
*
PCCLIP          LARK    0,99
                LARP    0
*
PCLPL1          LAC     PTHRE
                BZ      PCLPL3
                LAC     *
                ABS
                SUB     PTHRE
                BLEZ    PCLPL3
                LAC     *
                BGZ     PCLPL2
                LT    UNITY
                MPYK -5
                PAC
                SACL    *
                B       PCLPL4
PCLPL2          LACK    +5
                SACL    *
                B       PCLPL4
PCLPL3          ZAC
                SACL    *
PCLPL4          BANZ    PCLPL1
*
                RET
*
*
****************************************
* PCORR SUBROUTINE
* AR1=99;AR0=99-X WHERE X=NO. OF DELAY
****************************************
*
PCORR           LT      UNITY
                MPYK    DB0
                PAC
                SACL    PCORPM
*
                ZAC
                SACL    PDELAY
*
PCORL1          LACK    99
                SUB     PDELAY
                SACL    PDLAR0
                BLZ     PCOROK
*
                LAR     0,PDLAR0
                LARK    1,99
*
                ZAC
                MPYK    0
                LARP    0
PCORL2          LTA     *,1
```

```
                MPY     *-,0
                BANZ    PCORL2
                APAC
                SACL    PATDAT
*
                LAC     PCORPM
                TBLW    PATDAT
                ADD     UNITY
                SACL    PCORPM
*
                LAC     PDELAY
                ADD     UNITY
                SACL    PDELAY
                B       PCORL1
*
PCOROK          RET
*
*
****************************************
* PCORMV SUBROUTINE
* MOVE PM(>DB0->E13) --->  DM(0-99)
****************************************
*
PCORMV          LT      UNITY
                MPYK    DB0
                PAC
*
                LARK    0,0
                LARK    1,99
PCMVL1          LARP    0
                TBLR    *+,1
                ADD     UNITY
                BANZ    PCMVL1
*
                RET
*
*
****************************************
* PPEAK SUBROUTINE
* FIND MAX OF DM(15-98)
****************************************
*
PPEAK           ZAC
                SACL    PRXX
*
                LARK    0,15
                LARK    1,83
*
PPKL1           LARP    0
                LAC     *
                SUB     PRXX
                BLZ     PPKL2
                LAC     *
                SACL    PRXX
                SAR     0,PP1
PPKL2           MAR     *+,1
                BANZ    PPKL1
*
                RET
*
*
***************************************
* PINTRP SUBROUTINE
* QUADRATIC INTERPOLATION
* XMAX=X1+(Y0-Y2)/(Y0-2Y1+Y2)
***************************************
```

```
*
PINTRP  LAR    0,PP1
        LARP   0
        MAR    *-
        LAC    *+
        SACL   PY0
        LAC    *+
        SACL   PY1
        LAC    *
        SACL   PY2
*
        LAC    PY0
        SUB    PY2
        SACL   PCOMP1
        LT     PCOMP1
        MPYK   +2
        PAC
        SACL   PCOMP1
*
        LT     PY1
        MPYK   -2
        PAC
        ADD    PY0
        ADD    PY2
        SACL   PCOMP2
*
        ZAC
        SACL   PXX
*
        LAC    PCOMP1
        BZ     PINTR1
        LAC    PCOMP2
        BZ     PINTR1
*
        LAC    PCOMP1
        SUB    PCOMP2
        BLZ    PINTR2
        LACK   1
        SACL   PXX
        B      PINTR1
PINTR2  LAC    PCOMP1
        ADD    PCOMP2
        BGZ    PINTR1
        ZAC
        SUB    UNITY
        SACL   PXX
PINTR1  LT     PP1
        MPYK   +2
        PAC
        ADD    PXX
        SACL   PPITCH
*
        RET
*
*
****************************************
* POUT SUBROUTINE
* OUTPUT PPITCH TO PM( >CA0 )
****************************************
*
POUT    LT     UNITY
        MPYK   CA0
        PAC
        TBLW   PPITCH
*
        RET
```

```
*
*
**************************************
* INTIAL SUBROUTINE
* DEFINE SAMPLE RATE & SAMPLE MODE
* DEFINE UNITY & INPUT DATA MASK
**************************************
*
INTIAL  LACK  1
        SACL  UNITY
*
        LT    UNITY
        MPYK  SAMRAT
        PAC
        SACL  CLCK
*
        LACK  MODE
        SACL  CNTRL
*
        OUT   CLCK,1
        OUT   CNTRL,0
*
        LT    UNITY
        MPYK  MASK
        PAC
        SACL  INTMSK
        LAC   INTMSK,4
        SACL  INTMSK
*
        RET
*
*
**************************************
* AGAIN SUBROUTINE
* DEFINE INPUT STARTING ADDRESS (PM)
* AND INPUT DATA COUNTER
**************************************
*
AGAIN   LT    UNITY
        MPYK  CCO
        PAC
        SACL  INTPMA
*
        LACK  196
        SACL  INTNDT
*
        RET
*
*
```

<

FILE: ANYSUBR

```
**************************************
* ASTOVL SUBROUTINE
* MOVE PM(>FE4->FF7)---> PM(>F00->F13)
**************************************
*
ASTOVL  LT    UNITY
        MPYK  FE4
        PAC
*
        LARK  0,0
        LARK  1,19
ASTOL1  LARP  0
        TBLR  *+,1
```

```
        ADD   UNITY
        BANZ  ASTOL1
*
        LT    UNITY
        MPYK  F00
        PAC
*
        LARK  0,0
        LARK  1,19
ASTOL2  LARP  0
        TBLW  *+,1
        ADD   UNITY
        BANZ  ASTOL2
*
        RET
*
*
****************************************
* APREMP SUBROUTINE
* INPUT DATA & PREEMPHASIS [1-0.4Z^-1]
****************************************
*
APREMP  LAC   AF1C
        TBLR  AINPUT
        ADD   UNITY
        SACL  AF1C
*
        LT    AEMREF
        MPYK  -3684
        PAC
        SACH  AEMREF,4
*
        LAC   AEMREF
        ADD   AINPUT
        SACL  AINSHF
*
        LAC   AINPUT
        SACL  AEMREF
*
        RET
*
*
****************************************
* AWIN SUBROUTINE
* INPUT WINDOW DATA AWINDT
* AINSHF=AINSHF*AWINDT
****************************************
*
AWIN    LAC   AE20
        TBLR  AWINDT
        ADD   UNITY
        SACL  AE20
*
        LT    AWINDT
        MPY   AINSHF
        PAC
        SACH  AINSHF,4
*
        RET
*
*
****************************************
* ASHFT SUBROUTINE
* AD(I)=AD(I-1),I=10......0;AD0=AINSHF
****************************************
*
```

```
ASHFT    SAR   0,AARO
*
         LARK  0,AD9
         LARK  1,10
ASFTL1   LARP  0
         DMOV  *-,1
         BANZ  ASFTL1
*
         LAR   0,AARO
*
         RET
*
*
*************************************
* AACORR SUBROUTINE
* AR(I)=AR(I)+ADO*ADI    I=0..........10
*************************************
*
AACORR   SAR   0,AARO
*
         LT    ADO
         LARK  0,AROH
         LARK  1,ADO
ACORL1   LARP  0
         ZALH  *+
         ADDS  *-,1
         MPY   *+,0
         APAC
         SACH  *+
         SACL  *+
         SAR   1,AAR1
         LACK  AD10
         SUB   AAR1
         BGEZ  ACORL1
*
         LAR   0,AARO
*
         RET
*
*
**************************************
* AENGRY SUBROUTINE
* A(>7FFF)      C            B(0)
* L____________L___________1
**************************************
*
AENGRY   LACK  +151
         SACL  AA
         LT    AA
         MPYK  +217
         PAC
         SACL  AA
*
         ZAC
         SACL  AB
*
AENGL1   LAC   AA,15
         ADD   AB,15
         SACH  AC
*
         LT    AC
         MPY   AC
         PAC
         SUBH  AROH
         SUBS  AROL
         BZ    AENGL4
```

```
        BLZ   AENGL2
        LAC   AC
        SACL  AA
        B     AENGL3
AENGL2  LAC   AC
        SACL  AB
AENGL3  LAC   AA
        SUB   AB
        ABS
        SUB   UNITY
        BNZ   AENGL1
AENGL4  LT    UNITY
        MPYK  CA1
        PAC
        TBLW  AC
*
        RET
*
*
***************************************
* APNORM SUBROUTINE
* SHIFT AR0 TO A MAX +VE NO. AND SHIFT
* AR1......AR10 ACCORDINGLY (16 BIT)
***************************************
*
APNORM  LAC   AR0H
        BNZ   ASHFL1
        LAC   AR0L
        BLZ   ASHFL1
        LACK  16
        SACL  ACNTER
        B     ASHFL4
*
ASHFL1  LACK  15
        SACL  ACNTER
        LACK  1
        SACL  AREF
ASHFL2  LAC   AR0H
        SUB   AREF
        BGEZ  ASHFL3
        LAC   ACNTER
        SACL  ASFCNT
        B     ASHFL4
ASHFL3  LAC   AREF,1
        SACL  AREF
        LAC   ACNTER
        SUB   UNITY
        SACL  ACNTER
        B     ASHFL2
*
ASHFL4  LARK  0,10
        LARK  1,AR0H
ASHFL5  LAC   ACNTER
        SACL  ASFCNT
ASHFL6  LARP  1
        ZALH  *+
        ADDS  *-
        SACH  *+,1
        ZAC
        LAC   *,1
        SACL  *-
        LAC   ASFCNT
        SUB   UNITY
        SACL  ASFCNT
        BGZ   ASHFL6
        MAR   *+
```

```
        MAR     *+
        LARP    0
        BANZ    ASHFL5
*
        LACK    11
        SACL    ACNTER
        LARK    0,AROH
        LARK    1,ARO
ASHFL7  LARP    0
        LAC     *+
        MAR     *+,1
        SACL    *+
        LAC     ACNTER
        SUB     UNITY
        SACL    ACNTER
        BGZ     ASHFL7
*
        RET
*
*
****************************************
* ADIV SUBROUTINE
* AQUOT=ANUMER/ADENOM
****************************************
*
ADIV    SAR     0,AARO
        SAR     1,AAR1
*
        LARP    0
        LT      ANUMER
        MPY     ADENOM
        PAC
        SACH    ATMSGN
        LAC     ADENOM
        ABS
        SACL    ADENOM
        ZALH    ANUMER
        ABS
        LARK    0,14
ADIVL1  SUBC    ADENOM
        BANZ    ADIVL1
        SACL    AQUOT
        LAC     ATMSGN
        BGEZ    ADIVL2
        ZAC
        SUB     AQUOT
        SACL    AQUOT
*
ADIVL2  LAR     0,AARO
        LAR     1,AAR1
*
        RET
*
*
****************************************
* ANORM SUBROUTINE
* RI=RI/RO  I=10,.........0
****************************************
*
ANORM   LARK    1,10
        LARK    0,AR10
ANORML  LAC     ARO
        SACL    ADENOM
        LARP    0
        LAC     *
        SACL    ANUMER
```

```
        DINT
        CALL  ADIV
        EINT
        LAC   AQUOT
        SACL  *-
        LARP  1
        BANZ  ANORML
*
        RET
*
*
****************************************
* ALAG SUBROUTINE
* ROUX AND GUEGUEN ITERATION
****************************************
*
ALAG    LARK  0,>21
        LARK  1,80
        ZAC
ALAGL1  LARP  0
        SACL  *+,0,1
        BANZ  ALAGL1
*
        LAC   ARO
        SACL  AXO
*
        LARK  0,AR1
        LARK  1,AX1
ALAGL2  LARP  0
        LAC   *+,0,1
        SACL  *+
        SACL  *+
        SAR   0,AARO
        LACK  AR10
        SUB   AARO
        BGEZ  ALAGL2
*
        LARK  0,AK1
ALAGL3  LAC   AX1
        SACL  ANUMER
        LAC   AXO
        SACL  ADENOM
        DINT
        CALL  ADIV
        EINT
        ZAC
        SUB   AQUOT
        LARP  0
        SACL  *
*
        SAR   0,AKCNT
        LACK  AK10
        SUB   AKCNT
        BZ    ALAGL5
*
        LARK  1,AXO
        LARP  1
ALAGL4  MAR   *+
        LT    *-,0
        MPY   *,1
        PAC
        SACH  AMANS,1
        LAC   AMANS
        ADD   *
        SACL  *+
        MAR   *+
```

```
         LT     *+,0
         MPY    *,1
         PAC
         SACH   AMANS,1
         LAC    AMANS
         ADD    *-
         MAR    *-
         SACL   *+,0,1
*
         SAR    1,AAR1
         LACK   AX18
         SUB    AAR1
         BGEZ   ALAGL4
*
         LARP   0
         MAR    *+
         B      ALAGL3
*
ALAGL5   RET
*
*
****************************************
* AOUT   SUBROUTINE
* OUTPUT AK1....AK10 TO PM (>CA2....>CAB)
****************************************
*
AOUT     LT     UNITY
         MPYK   CA2
         PAC
*
         LARK   0,AK1
         LARK   1,9
AOUTL1   LARP   0
         TBLW   *+,1
         ADD    UNITY
         BANZ   AOUTL1
*
         RET
*
*
****************************************
* ARST SUBROUTINE
* TO MOVE PM(>F00..>F13) --> PM(>FIC..>F2F)
****************************************
*
ARST     LT     UNITY
         MPYK   F00
         PAC
*
         LARK   0,0
         LARK   1,19
ARSTL1   LARP   0
         TBLR   *+,1
         ADD    UNITY
         BANZ   ARSTL1
*
         LT     UNITY
         MPYK   F1C
         PAC
*
         LARK   0,0
         LARK   1,19
ARSTL2   LARP   0
         TBLW   *+,1
         ADD    UNITY
         BANZ   ARSTL2
```

```
*
        RET
*
*
<
FILE:     ENDSUBR
```

```
*
***************************************
* COEFXF SUBROUTINE
* TRANSFER PM (>CA0..>CAB) TO
* DM (>0..>B)
***************************************
*
COEFXF  LT    UNITY
        MPYK  CAB
        PAC
*
        LARK  0,>B
        LARP  0
COFXFL  TBLR  *
        SUB   UNITY
        BANZ  COFXFL
*
        RET
*
*
***************************************
* XMIT SUBROUTINE
* TRANSMIT DM(>0...>B) TO OUTPUT PORT 3
***************************************
*
XMIT    LARK  0,0
        LARK  1,11
*
XMITL1  LARP  0
        OUT   *,3
        OUT   *,3
        OUT   *+,3,1
        OUT   5,5
XMITL2  BIOZ  XMITL3
        B     XMITL2
XMITL3  OUT   6,6
        OUT   6,6
        OUT   6,6
        BANZ  XMITL1
*
        RET
*
*
***************************************
* NEWFRM SUBROUTINE
* MOVE PM ( >CC0...>D87) TO
* PM (>F30...>FF7)
***************************************
*
NEWFRM  LT    UNITY
        MPYK  CC0
        PAC
        SACL  NEWP1
        MPYK  F30
        PAC
        SACL  NEWP2
*
        LARK  0,199
```

```
        LARP   0
NEWFRL  LAC    NEWP1
        TBLR   NEWDAT
        ADD    UNITY
        SACL   NEWP1
        LAC    NEWP2
        TBLW   NEWDAT
        ADD    UNITY
        SACL   NEWP2
        BANZ   NEWFRL
*
        RET
*
*
****************************************
* INTSUR SUBROUTINE
* INTERRUPT HANDLING SUBROUTINE
* INPUT DATA XOR >7FF0 --> PM(INTPMA)
****************************************
*
INTSUR  DINT
        LDPK   1
        SST    INTSTU
        SACH   INTACH
        SACL   INTACL
        LDPK   0
*
        IN     INTDAT,2
        LAC    INTDAT
        XOR    INTMSK
        SACL   INTDAT
        LAC    INTDAT,12
        SACH   INTDAT
*
        LAC    INTPMA
        TBLW   INTDAT
        ADD    UNITY
        SACL   INTPMA
*
        LAC    INTNDT
        SUB    UNITY
        SACL   INTNDT
*
        LDPK   1
        ZALH   INTACH
        ADDS   INTACL
        LST    INTSTU
        LDPK   0
*
        EINT
*
        RET
*
*
<
```

FILE: CLEAR

```
************************************
* CLEAR SUBROUTINE
* CLEAR DM ( >0..............>7A )
************************************
*
CLEAR   ZAC
        LARP   0
        LARK   0,>7A
```

```
00090 CLRL1    SACL   *
00100          BANZ   CLRL1
00110 *
00120          RET
00130 *
00140 *
<
FILE:      END

00010 *
00020          END
00030 *
<
```

FILE: SYMAIN

```
*
****************************************
*
*    TMS32010 LPC SYNTHESIZER
*
*         D.S.F.CHAN
*
****************************************
*
          AORG   >0
          B      START
          B      INT
*
          AORG   >A
*
CNTRL     EQU    >0
CLCK      EQU    >1
MODE      EQU    >E
SAMRAT    EQU    4095
MASK      EQU    >800
PULSE     EQU    500
*
LD        EQU    >2A
STORE     EQU    >2B
NOISE     EQU    >2C
RNDREG    EQU    >2D
RNDIN     EQU    >2E
RNDOUT    EQU    >2F
UF        EQU    >30
K         EQU    >31
P         EQU    >32
F1        EQU    >33
F2        EQU    >34
F3        EQU    >35
F4        EQU    >36
F5        EQU    >37
F6        EQU    >38
F7        EQU    >39
F8        EQU    >3A
F9        EQU    >3B
F10       EQU    >3C
G1        EQU    >3D
G2        EQU    >3E
G3        EQU    >3F
G4        EQU    >40
G5        EQU    >41
G6        EQU    >42
G7        EQU    >43
G8        EQU    >44
G9        EQU    >45
G10       EQU    >46
D1        EQU    >47
D2        EQU    >48
D3        EQU    >49
D4        EQU    >4A
D5        EQU    >4B
D6        EQU    >4C
D7        EQU    >4D
D8        EQU    >4E
D9        EQU    >4F
D10       EQU    >50
LATIN     EQU    >51
LATOUT    EQU    >52
```

```
LATTEM  EQU   >53
PPITCH  EQU   >54
PGAIN   EQU   >55
PK1     EQU   >56
PK2     EQU   >57
PK3     EQU   >58
PK4     EQU   >59
PK5     EQU   >5A
PK6     EQU   >5B
PK7     EQU   >5C
PK8     EQU   >5D
PK9     EQU   >5E
PK10    EQU   >5F
NPITCH  EQU   >60
NGAIN   EQU   >61
NK1     EQU   >62
NK2     EQU   >63
NK3     EQU   >64
NK4     EQU   >65
NK5     EQU   >66
NK6     EQU   >67
NK7     EQU   >68
NK8     EQU   >69
NK9     EQU   >6A
NK10    EQU   >6B
IPITCH  EQU   >6C
IGAIN   EQU   >6D
IK1     EQU   >6E
IK2     EQU   >6F
IK3     EQU   >70
IK4     EQU   >71
IK5     EQU   >72
IK6     EQU   >73
IK7     EQU   >74
IK8     EQU   >75
IK9     EQU   >76
IK10    EQU   >77
INTSTU  EQU   >78
INTACH  EQU   >79
INTACL  EQU   >7A
INTAR0  EQU   >7B
INTAR1  EQU   >7C
INTADD  EQU   >7D
OPMSK   EQU   >7E
UNITY   EQU   >7F
*
*
****************************************
* SYNTHESISING MAIN PROGRAM
****************************************
*
*
START   DINT
        CALL  INTIAL
        CALL  CLEAR
        ZAC
        SACL  UF
        SACL  LD
        EINT
SYL1    LAC   UF
        BNZ   SYL2
        DINT
        CALL  UPDATE
        EINT
        CALL  DECIS
        LACK  1
```

```
        SACL  UF
SYL2    CALL  CLEAR
        LAC   PPITCH
        BNZ   SYL4
SYL3    CALL  PRBS
        LAC   NOISE
        SACL  LATIN
        CALL  LATICE
        LAC   UF
        BNZ   SYL3
        B     SYL1
SYL4    LACK  0
        SACL  LATIN
        CALL  LATICE
        LACK  1
        SACL  LATIN
        CALL  LATICE
        LACK  4
        SACL  LATIN
        CALL  LATICE
        LACK  10
        SACL  LATIN
        CALL  LATICE
        LACK  18
        SACL  LATIN
        CALL  LATICE
        LACK  28
        SACL  LATIN
        CALL  LATICE
        LACK  38
        SACL  LATIN
        CALL  LATICE
        LACK  50
        SACL  LATIN
        CALL  LATICE
        LACK  61
        SACL  LATIN
        CALL  LATICE
        LACK  71
        SACL  LATIN
        CALL  LATICE
        LACK  81
        SACL  LATIN
        CALL  LATICE
        LACK  89
        SACL  LATIN
        CALL  LATICE
        LACK  95
        SACL  LATIN
        CALL  LATICE
        LACK  98
        SACL  LATIN
        CALL  LATICE
        LACK  100
        SACL  LATIN
        CALL  LATICE
        LACK  96
        SACL  LATIN
        CALL  LATICE
        LACK  86
        SACL  LATIN
        CALL  LATICE
        LACK  70
        SACL  LATIN
        CALL  LATICE
        LACK  50
```

```
            SACL  LATIN
            CALL  LATICE
            LACK  25
            SACL  LATIN
            CALL  LATICE
            LACK  0
            SACL  LATIN
            CALL  LATICE
            LACK  20
            SACL  K
            LAC   PPITCH
            SUB   K
            SACL  P
SYL5        ZAC
            SACL  LATIN
            CALL  LATICE
            LAC   P
            SUB   UNITY
            SACL  P
            BNZ   SYL5
            B     SYL1
            NOP
            NOP
            NOP
*
*
<
FILE:      SYSUBR1


*
*
***************************************
* SYNTHESIS SUBROUTINES
***************************************
*
*
***************************************
* INTIAL SUBROUTINES
* UNITY=1:RNDREG=1
* DEFINE SAMPLING RATE & SAMPLING MODE
* DEFINE O/P MASK & I/P STARTING ADDRESS
***************************************
*
INTIAL      LACK  1
            SACL  UNITY
            SACL  RNDREG
*
            LT    UNITY
            MPYK  SAMRAT
            PAC
            SACL  CLCK
*
            MPYK  PULSE
            PAC
            SACL  K
*
            LACK  MODE
            SACL  CNTRL
*
            OUT   CLCK,1
            OUT   CNTRL,0
*
            LT    UNITY
            MPYK  MASK
            PAC
```

```
        SACL  OPMSK
        LAC   OPMSK,4
        SACL  OPMSK
*
        LACK  IPITCH
        SACL  INTADD
*
        RET
*
*
*************************************
* CLEAR SUBROUTINE
* CLEAR DM F1...F10,G1...G10,D1...D10
*************************************
*
CLEAR   ZAC
        SACL  F1
        SACL  F2
        SACL  F3
        SACL  F4
        SACL  F5
        SACL  F6
        SACL  F7
        SACL  F8
        SACL  F9
        SACL  F10
        SACL  G1
        SACL  G2
        SACL  G3
        SACL  G4
        SACL  G5
        SACL  G6
        SACL  G7
        SACL  G8
        SACL  G9
        SACL  G10
        SACL  D1
        SACL  D2
        SACL  D3
        SACL  D4
        SACL  D5
        SACL  D6
        SACL  D7
        SACL  D8
        SACL  D9
        SACL  D10
*
        RET
*
*
*************************************
* UPDATE SUBROUTINE
* P( PARAMETERS )=N( PARAMETERS )
*************************************
*
UPDATE  LAC   NPITCH
        SACL  PPITCH
        LAC   NGAIN
        SACL  PGAIN
        LAC   NK1
        SACL  PK1
        LAC   NK2
        SACL  PK2
        LAC   NK3
        SACL  PK3
        LAC   NK4
```

```
        SACL  PK4
        LAC   NK5
        SACL  PK5
        LAC   NK6
        SACL  PK6
        LAC   NK7
        SACL  PK7
        LAC   NK8
        SACL  PK8
        LAC   NK9
        SACL  PK9
        LAC   NK10
        SACL  PK10
*
        RET
*
*
**************************************************
* DECIS SUBROUTINE
* IF PPITCH >=195 THEN PPITCH=0:PGAIN=0
* IF PK1>-0.2 THEN PPITCH=0
* PGAIN=PGAIN*32/1000
**************************************************
*
DECIS   LACK  195
        SUB   PPITCH
        BGZ   DECL1
        ZAC
        SACL  PPITCH
        SACL  PGAIN
        B     DECL2
DECL1   LACK  58
        SACL  LATTEM
        LT    LATTEM
        MPYK  141
        PAC
        ADD   PK1
        BLEZ  DECL2
        ZAC
        SACL  PPITCH
*
DECL2   LT    PGAIN
        MPYK  1048
        PAC
        SACH  PGAIN,1
*
        RET
*
*
**************************************************
* PRBS SUBROUTINE
* PSEUDO RANDOM BINARY SEQUENCE
**************************************************
*
PRBS    LAC   RNDREG
        AND   UNITY
        SACL  RNDIN
        LAC   RNDREG,7
        SACH  RNDOUT
        LAC   RNDOUT
        AND   UNITY
        XOR   RNDIN
        SACL  RNDOUT
        LAC   RNDOUT,11
        ADD   RNDREG
        SACL  RNDREG
```

```
        LAC   RNDREG,15
        SACH  RNDREG
        LAC   RNDREG
        LT    UNITY
        MPYK  1024
        SPAC
        SACL  NOISE
        LAC   NOISE,12
        SACH  NOISE
        NOP
*
*
        RET
*
*
<
FILE:     SYSUBR2


*
*
*************************************
* LATICE SUBROUTINE
* 10TH ORDER LATTICE FILTER OPERATION
*************************************
*
LATICE  LAC   LATIN
        ADD   D1
        SACL  F1
*
        LAC   F1
        ADD   D2
        SACL  LATTEM
        LT    LATTEM
        MPY   PK1
        PAC
        SACH  LATTEM,1
        LAC   F1
        ADD   LATTEM
        SACL  F2
        LAC   D2
        SUB   LATTEM
        SACL  G1
*
        LAC   F2
        ADD   D3
        SACL  LATTEM
        LT    LATTEM
        MPY   PK2
        PAC
        SACH  LATTEM,1
        LAC   F2
        ADD   LATTEM
        SACL  F3
        LAC   D3
        SUB   LATTEM
        SACL  G2
*
        LAC   F3
        ADD   D4
        SACL  LATTEM
        LT    LATTEM
        MPY   PK3
        PAC
        SACH  LATTEM,1
        LAC   F3
```

```
        ADD   LATTEM
        SACL  F4
        LAC   D4
        SUB   LATTEM
        SACL  G3
*
        LAC   F4
        ADD   D5
        SACL  LATTEM
        LT    LATTEM
        MPY   PK4
        PAC
        SACH  LATTEM,1
        LAC   F4
        ADD   LATTEM
        SACL  F5
        LAC   D5
        SUB   LATTEM
        SACL  G4
*
        LAC   F5
        ADD   D6
        SACL  LATTEM
        LT    LATTEM
        MPY   PK5
        PAC
        SACH  LATTEM,1
        LAC   F5
        ADD   LATTEM
        SACL  F6
        LAC   D6
        SUB   LATTEM
        SACL  G5
*
        LAC   F6
        ADD   D7
        SACL  LATTEM
        LT    LATTEM
        MPY   PK6
        PAC
        SACH  LATTEM,1
        LAC   F6
        ADD   LATTEM
        SACL  F7
        LAC   D7
        SUB   LATTEM
        SACL  G6
*
        LAC   F7
        ADD   D8
        SACL  LATTEM
        LT    LATTEM
        MPY   PK7
        PAC
        SACH  LATTEM,1
        LAC   F7
        ADD   LATTEM
        SACL  F8
        LAC   D8
        SUB   LATTEM
        SACL  G7
*
        LAC   F8
        ADD   D9
        SACL  LATTEM
        LT    LATTEM
```

```
        MPY    PK8
        PAC
        SACH   LATTEM,1
        LAC    F8
        ADD    LATTEM
        SACL   F9
        LAC    D9
        SUB    LATTEM
        SACL   G8
*
        LAC    F9
        ADD    D10
        SACL   LATTEM
        LT     LATTEM
        MPY    PK9
        PAC
        SACH   LATTEM,1
        LAC    F9
        ADD    LATTEM
        SACL   F10
        LAC    D10
        SUB    LATTEM
        SACL   G9
*
        LT     F10
        MPY    PK10
        PAC
        SACH   LATTEM,1
        ZAC
        SUB    LATTEM
        SACL   G10
        LAC    F10
        ADD    LATTEM
        SACL   LATTEM
        LT     LATTEM
        MPY    PGAIN
        PAC
        SACL   LATOUT
*
        LAC    G1
        SACL   D1
        LAC    G2
        SACL   D2
        LAC    G3
        SACL   D3
        LAC    G4
        SACL   D4
        LAC    G5
        SACL   D5
        LAC    G6
        SACL   D6
        LAC    G7
        SACL   D7
        LAC    G8
        SACL   D8
        LAC    G9
        SACL   D9
        LAC    G10
        SACL   D10
*
        LAC    LD
        SACL   STORE
        LT     STORE
        MPYK   3684
        PAC
        SACH   STORE,4
```

```
        LAC    STORE
        ADD    LATOUT
        SACL   LATOUT
        SACL   LD
*
WAIT    BIOZ   OUTPUT
        B      WAIT
OUTPUT  LAC    LATOUT,1
        XOR    OPMSK
        SACL   LATOUT
        OUT    LATOUT,2
*
        RET
*
*
****************************************
* INT SUBROUTINE
* INTERRUPT HANDLING SUBROUTINE
* AFTER HAVING INPUT 12 PARAMETERS
* PUSH I( PARAMETERS )=N( PARAMETERS )
****************************************
*
INT     DINT
        SST    INTSTU
        SACH   INTACH
        SACL   INTACL
        SAR    1,INTAR1
        SAR    0,INTAR0
*
        LAR    0,INTADD
        LARP   0
        OUT    6,6
        IN     *,4
        IN     *,4
        IN     *+,4
        OUT    5,5
        OUT    5,5
        OUT    5,5
        OUT    7,7
        SAR    0,INTADD
        LACK   IK10
        SUB    INTADD
        BGEZ   INTL1
*
        LACK   IPITCH
        SACL   INTADD
        LAC    IPITCH
        SACL   NPITCH
        LAC    IGAIN
        SACL   NGAIN
        LAC    IK1
        SACL   NK1
        LAC    IK2
        SACL   NK2
        LAC    IK3
        SACL   NK3
        LAC    IK4
        SACL   NK4
        LAC    IK5
        SACL   NK5
        LAC    IK6
        SACL   NK6
        LAC    IK7
        SACL   NK7
        LAC    IK8
        SACL   NK8
```

```
02300           LAC    IK9
02310           SACL   NK9
02320           LAC    IK10
02330           SACL   NK10
02335 *
02336           ZAC
02337           SACL   UF
02338 *
02340 *
02350 INTL1     ZALH   INTACH
02360           ADDS   INTACL
02370           LAR    0,INTAR0
02380           LAR    1,INTAR1
02390           LST    INTSTU
02400           EINT
02410 *
02420           RET
02430 *
02440 *
<
FILE:      END
```

```
00010 *
00020 ****************************************
00030 * ENDING STATEMENT
00040 ****************************************
00050 *
00060           END
00070 *
00080 *
<
```

R E F E R E N C E

1. WILBUR, Steve. "The Universe Project", Electronics & Power. May 1983, pp.394-398.

2. FLANAGAN, J.L., COBER,C.H., RABINER,L.R., SCHAFER,R.W., and UMEDA, N. "Synthetic Voices for Computers", IEEE Spectrum, Vol.7, No.10, pp.22-45, Oct.1970.

3. RABINER, L.R., SCHAFER, R.W. Digital Processing of Speech Signals, Prentice-Hall,1978.

4. PORTNOFF, M.R. "A Quasi-One-Dimensional Digital Simulation for the Time Varying Vocal Tract", M.S.Thesis, Dept.of Elect.Engr., MIT, Cambridge Mass., USA, June 1973.

5. FLANAGAN, J.L. Speech Analysis, Synthesis and Perception, 2nd Ed., Springer-Verlag, New York, 1972.

6. ROSENBERG, A.E. "Effects of Glottal Pulse Shape on the Quality of Natural Vowels", J.Acoust. Soc. Am., Vol.49, No.2, pp.583-590, Feb.1971.

7. MAKHOUL, J. "Linear Prediction : A Tutorial Review", PROC. IEEE, Vol.63, pp.561-580, 1975.

8. MARKEL,J.D. and GRAY,A.H.Jr., Linear Prediction of Speech, Springer-Verlag, New York, 1976.

9. WAKITA, H. "Direct Estimation of the Vocal Tract Shape by Inverse Filtering of Acoustic Speech Waveforms", IEEE Trans.on Audio and Electro acoustics Vol.AU-21, No.5, pp.417-427, Oct.1973.

10. LE ROUX,J., and GUEGUEN, G. "A Fixed Point Computation of Partial Correlation Coefficients", IEEE Transactions on ASSP. June 1977, pp.257-259.

11. HESS, W. Pitch Determination of Speech Signals, Algorithms and Devices, Springer Verlag, New York, 1983.

12. OH, C.A., UN, C.K. "A Performance Comparison of Pitch Extractor Algorithms for Noisy Speech". Private Communication, Dept.of Electrical Eng., Korea Advanced Inst. of Science & Technology, Seoul, Korea.

13. SONDHI, M.M. "New Methods of Pitch Extraction", IEEE Transactions on Audio and Acoustics, Vol.16, No.2, pp.262-266. June 1968.

14. BOZIC, S.M. Digital and Kalman Filtering, Edward Arnold, 1981.

15. LICKLIDER, J.C.R., and POLLACK,I. "Effects of Differentiation, Integration and Infinite Peak Picking upon the Intelligibility of Speech", J.Acoust.Soc. Am., Vol.20, pp.42-50, Jan.1948.

16. DUBNOWSKI, J.J., SCHAFER,R.W., and RABINER,L.R. "Real-Time Digital Hardware Pitch Detector", IEEE Transaction on ASSP. Vol.24, No.1, pp.2-8. Feb.1976.

17. ATAL,B.S., RABINER,L.R. "A Pattern Recognition Approach to Voiced-Unvoiced-Silence Classification with Application to Speech Recognition", IEEE Transactions on ASSP. Vol.24, No.3, pp.201-202. June 1976.

18. HUGHES,M.T.G. "The Simulation of Chance Phenomena", IUIEC Seminar, No.53.

19. Texa Instrument. TMS32010 Evaluation Module Users' Guide, 1984.

20. Texa Instrument. TMS32010 Users' Guide, 1983.

21. Texa Instrument. TMS32010 Analog.Interface Board Users' Guide, 1983.

22. KREYSZIG, Erwin. Advanced Engineering Mathematics. John Wiley and Sons,Inc., 1972. Section 18.4.

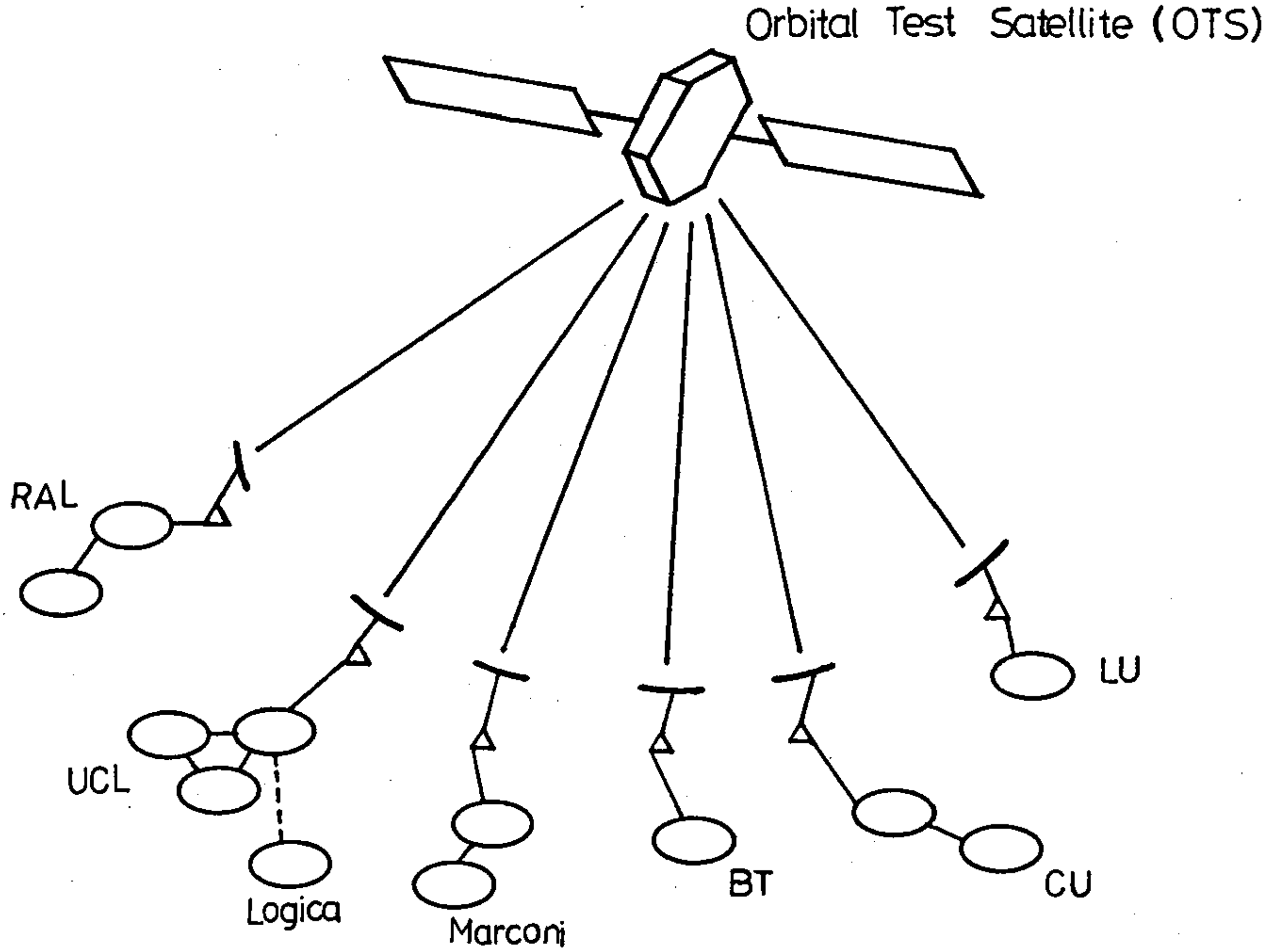


FIG.1.1 THE PROJECT UNIVERSE NETWORK

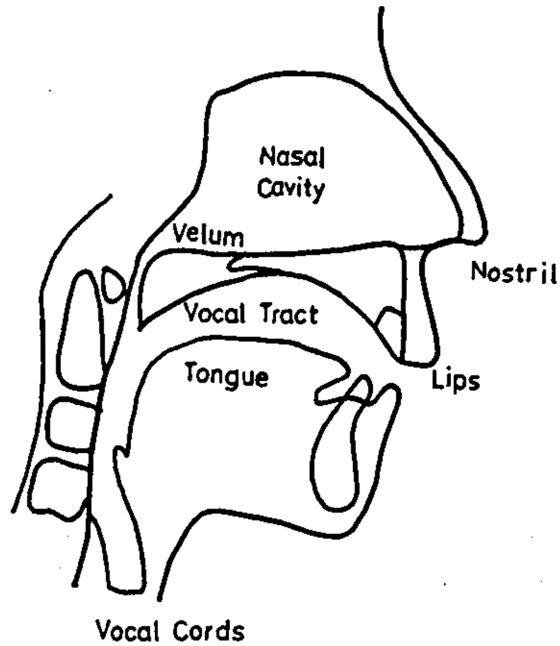


FIG.2.1 SCHEMATIC DIAGRAM OF THE HUMAN VOCAL TRACT

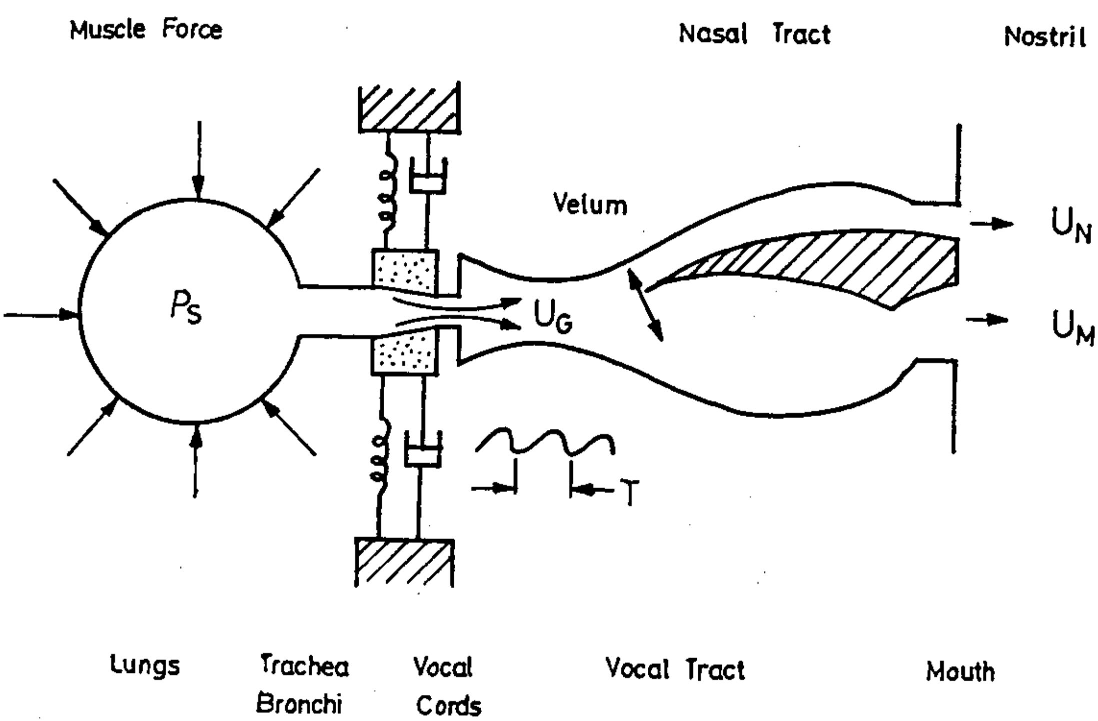


FIG. 2.2 FUNCTION DIAGRAM OF THE VOCAL APPARATUS
(AFTER FLANAGAN)

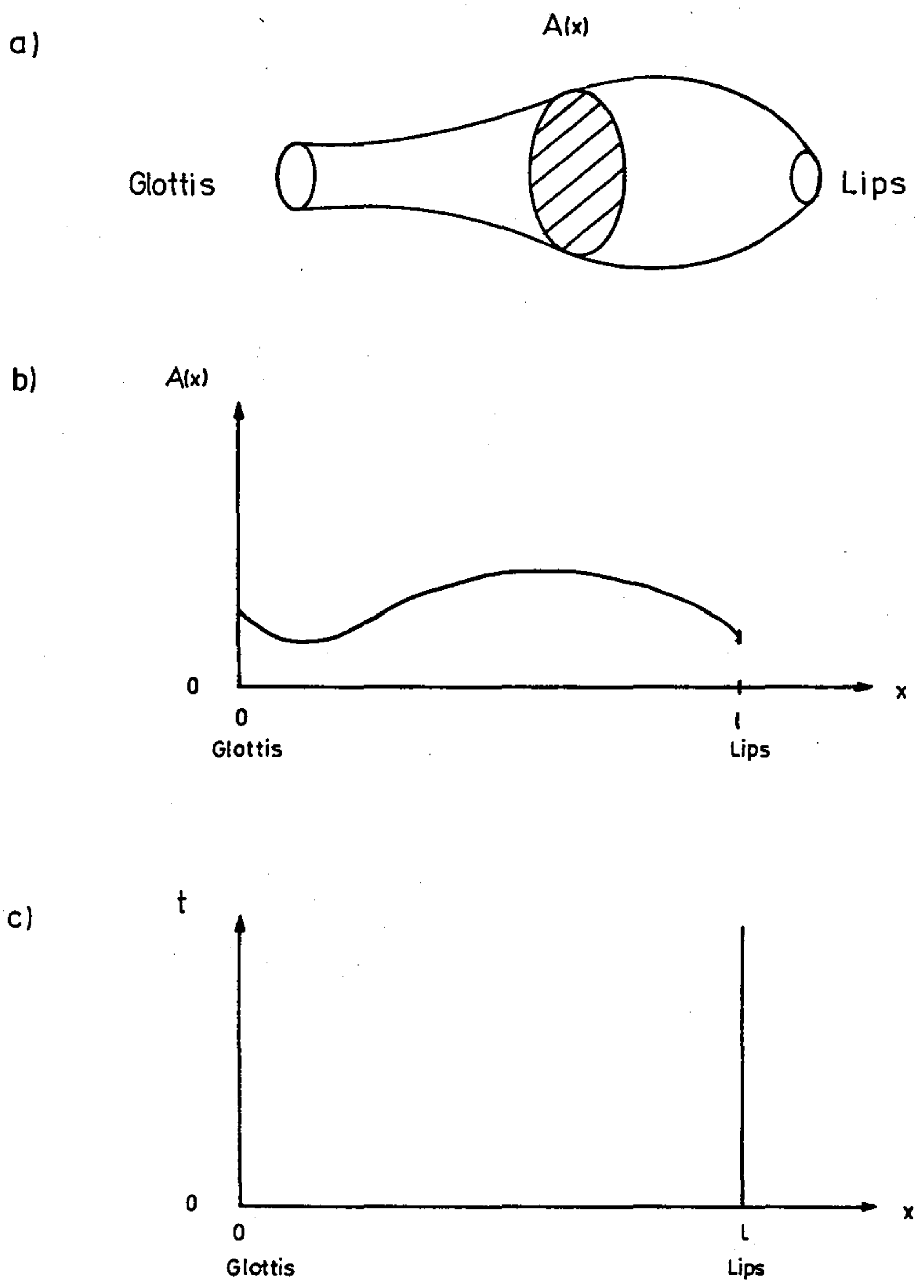


FIG.2.3 a) THE VOCAL TRACT MODEL
b) THE VOCAL TRACT MODEL AREA FUNCTION
c) THE x- t PLANE

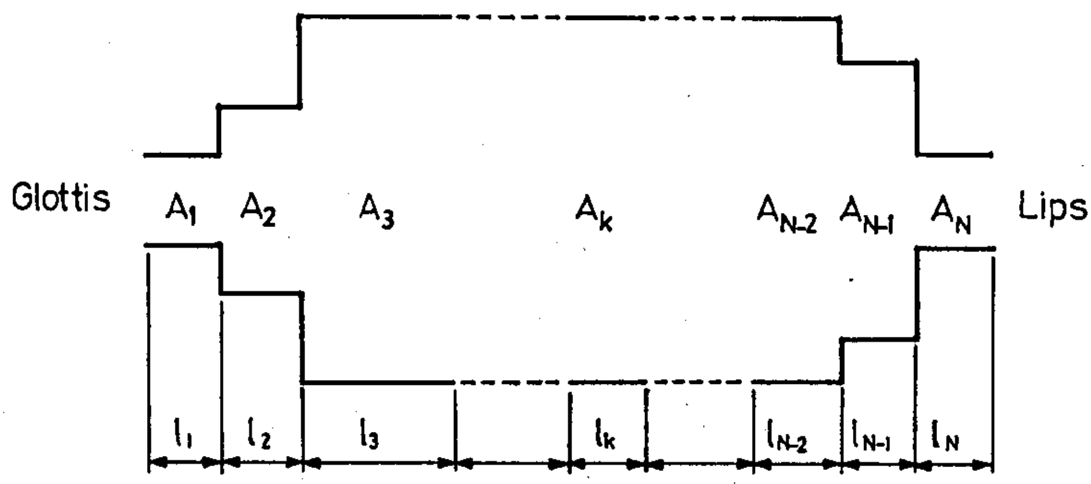


FIG. 2.4 THE CONCATENATION OF N LOSSLESS ACOUSTIC TUBES

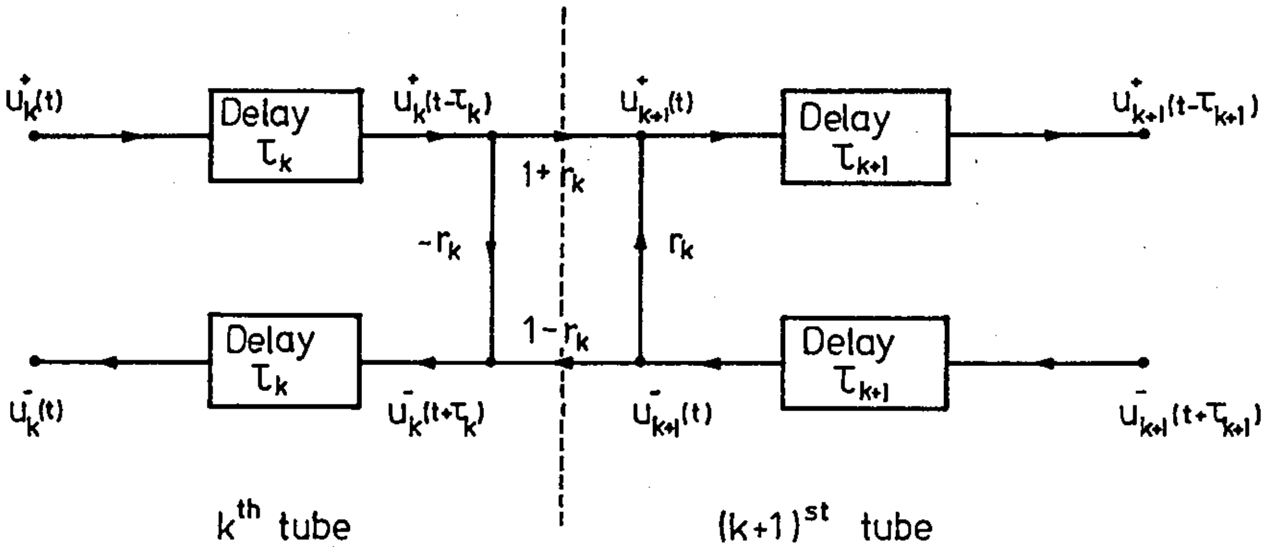


FIG. 2.5 THE JUNCTION BETWEEN TWO LOSSLESS TUBES

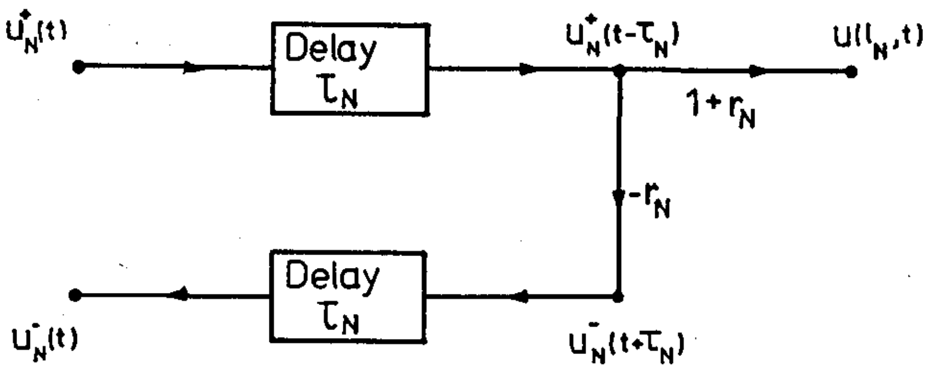


FIG. 2.6 THE TERMINATION AT LIP END OF A CONCATENATION OF LOSSLESS TUBES

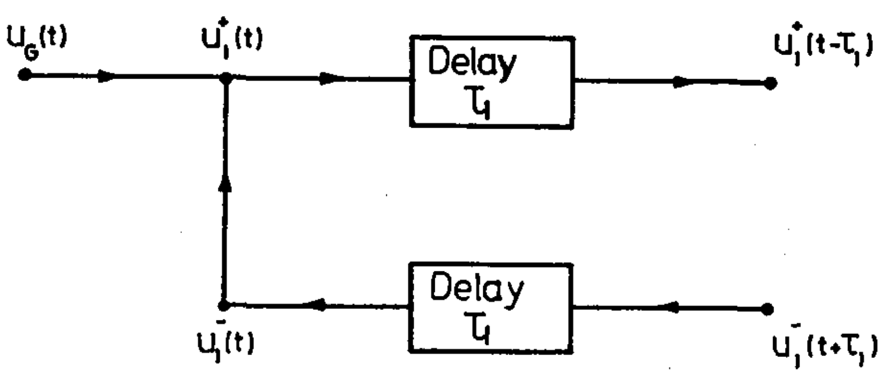


FIG. 2.7 THE TERMINATION AT GLOTTAL END OF A CONCATENATION OF LOSSLESS TUBES

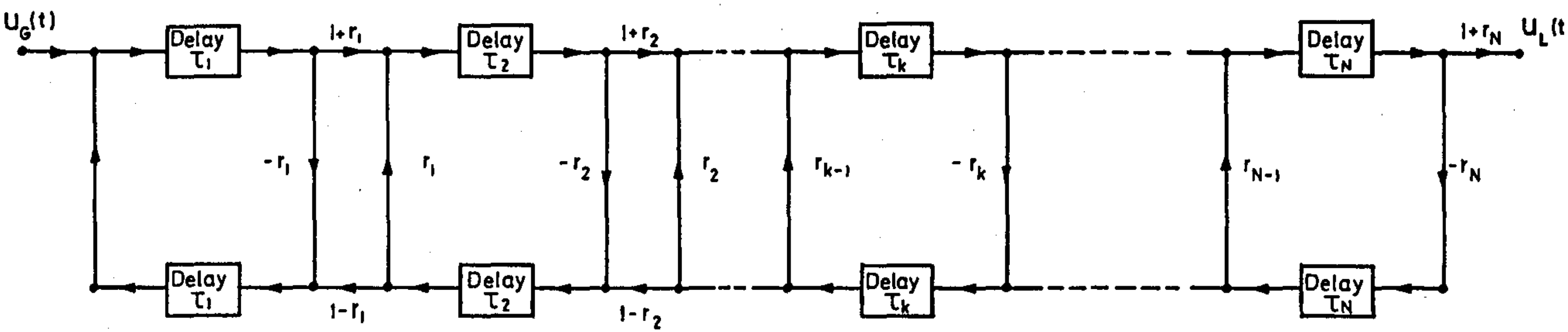


FIG.2.8 THE N-SECTION UNIFORM LOSSLESS TUBE MODEL

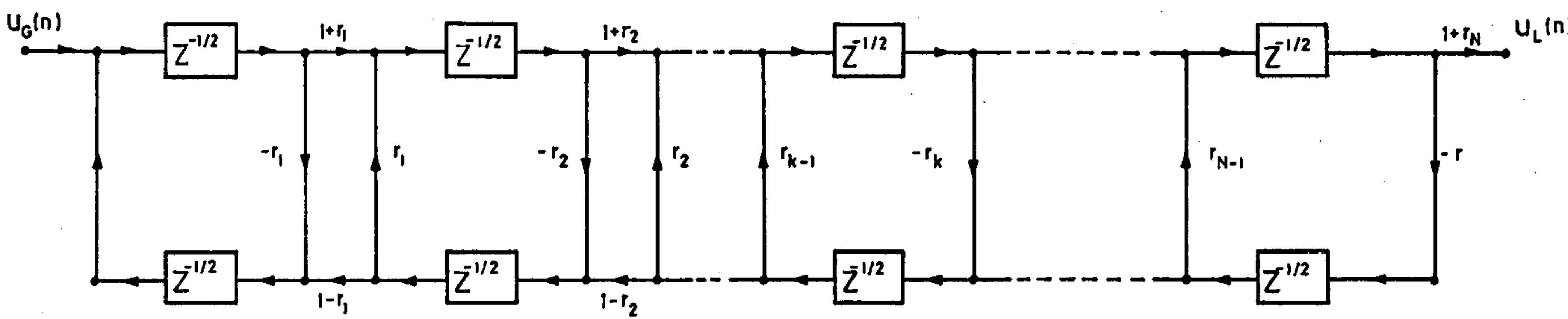


FIG.2.9 THE DISCRETE-TIME SYSTEM OF THE LOSSLESS TUBE MODEL OF THE VOCAL TRACT

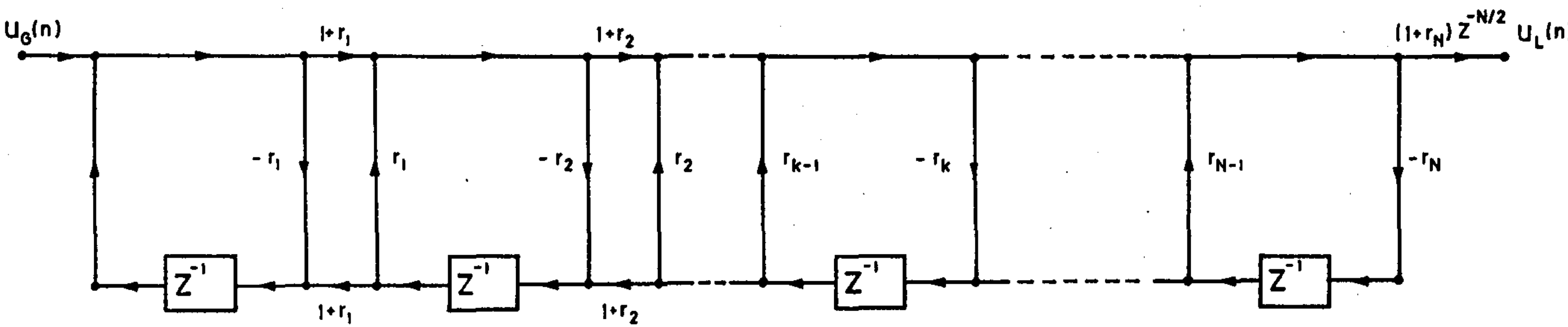


FIG.2.10 THE DISCRETE-TIME SYSTEM OF THE LOSSLESS TUBE MODEL OF THE VOCAL TRACT USING WHOLE DELAYS IN LADDER PARTS

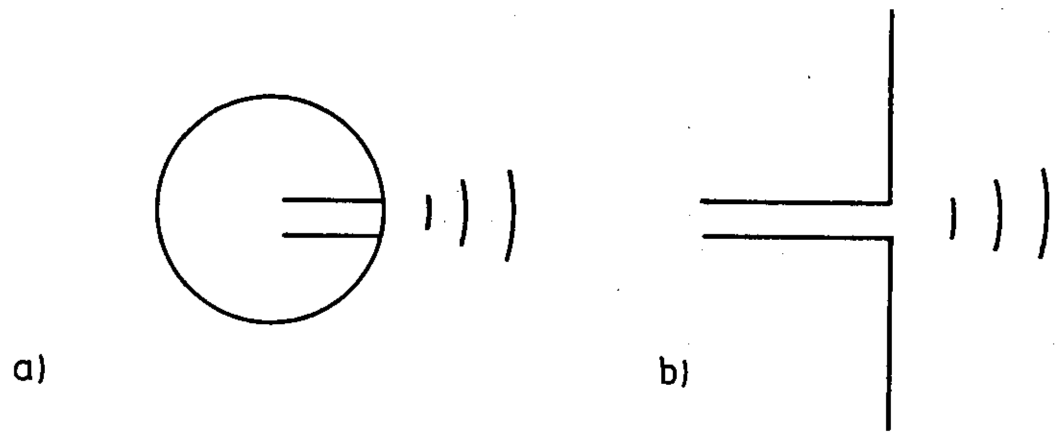


FIG. 2.11 a) THE RADIATION FROM A SPHERICAL BAFFLE
b) THE RADIATION FROM AN INFINITE PLANE BAFFLE

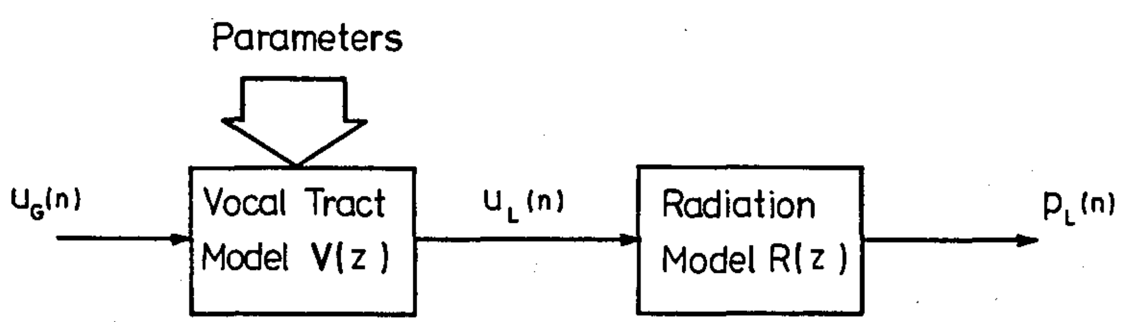


FIG. 2.12 BLOCK DIAGRAM OF THE VOCAL TRACT MODEL INCLUDING THE RADIATION EFFECT

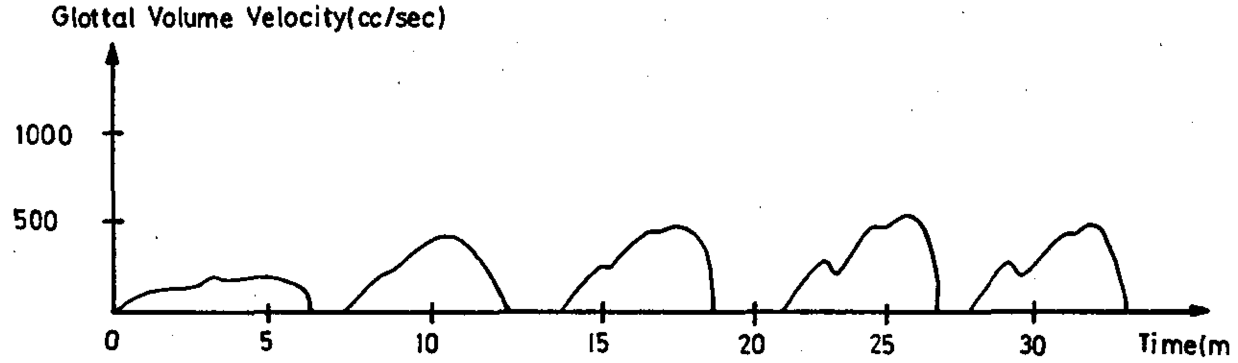


FIG.2.13 AN EXAMPLE OF GLOTTAL VOLUME VELOCITY AT MOUTH

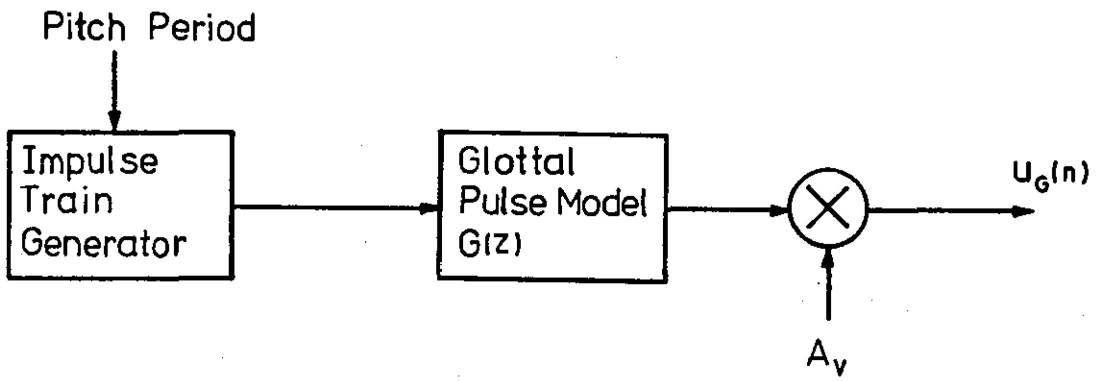


FIG.2.14 THE GLOTTAL PULSE GENERATOR

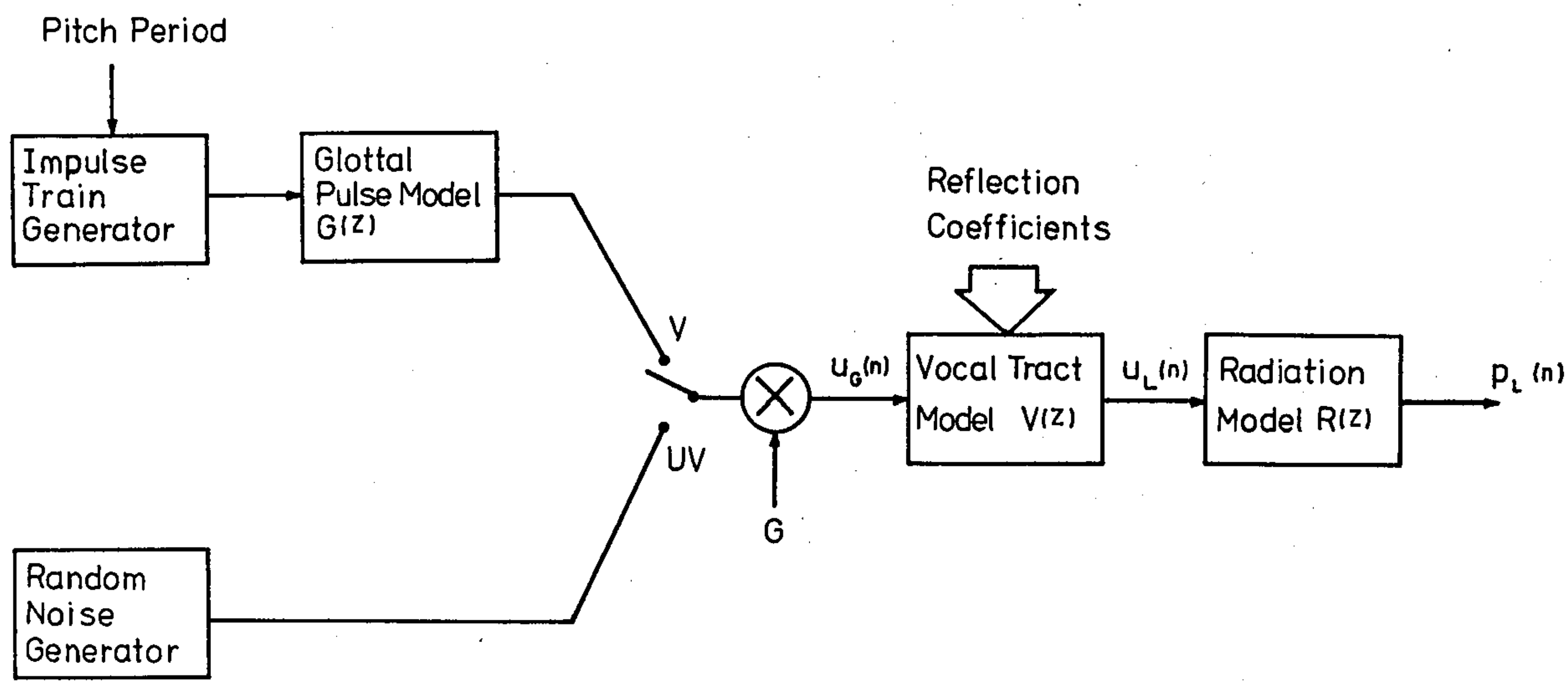


FIG.2.15 THE DISCRETE-TIME MODEL FOR SPEECH PRODUCTION

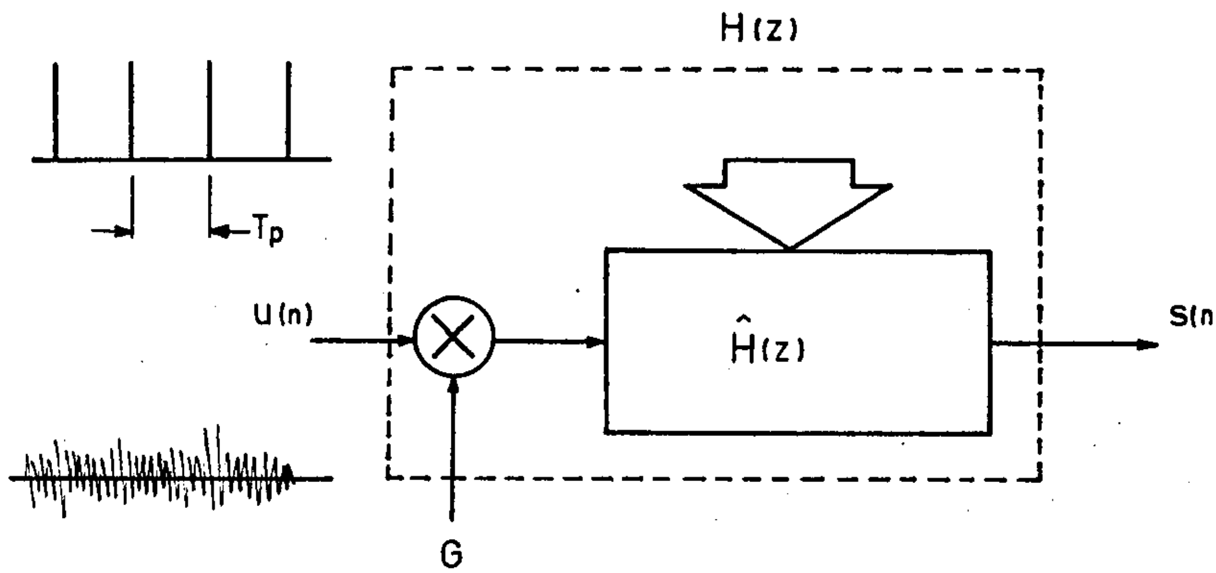


FIG.2.16 THE SIMPLIFIED DISCRETE-TIME MODEL FOR SPEECH PRODUCTION

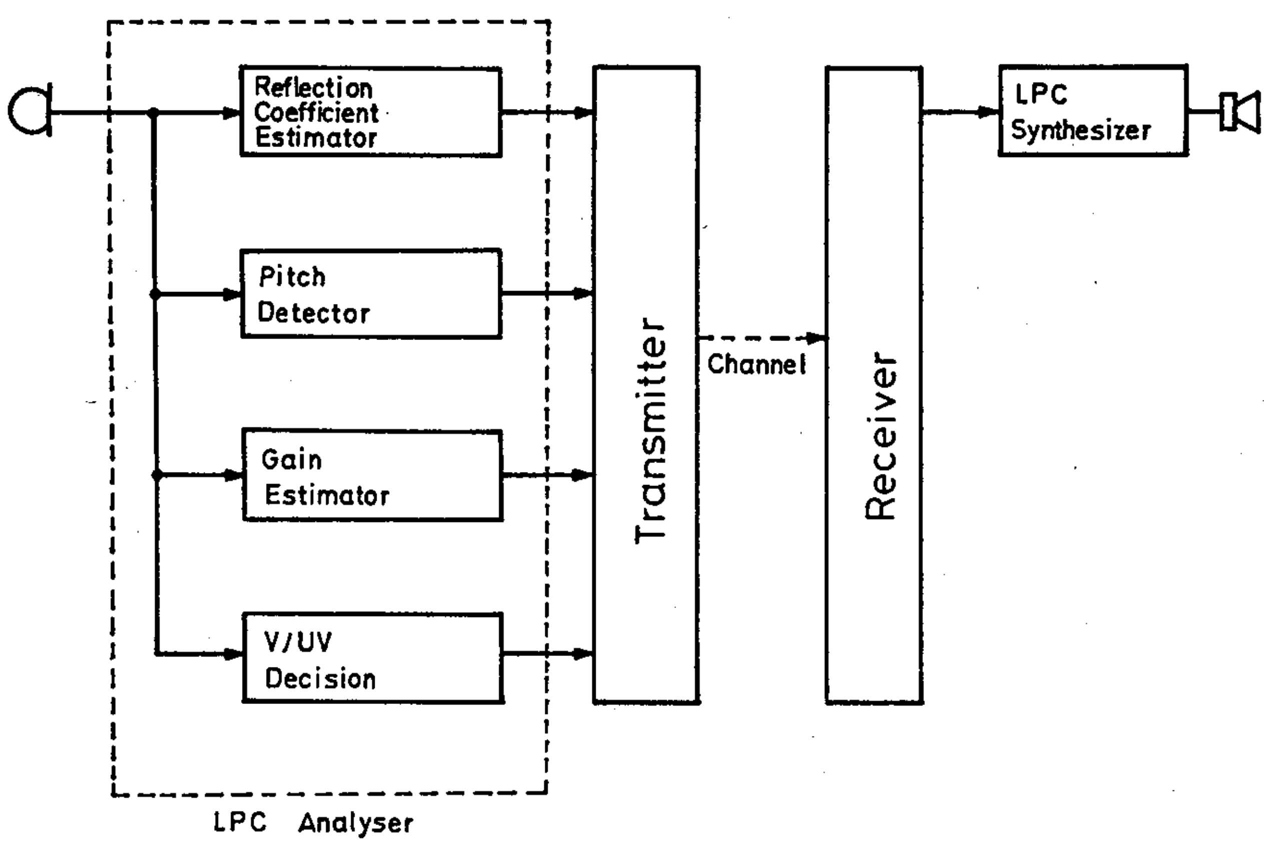


FIG.2.17 BASIC CONFIGURATION OF THE LPC EXPERIMENT

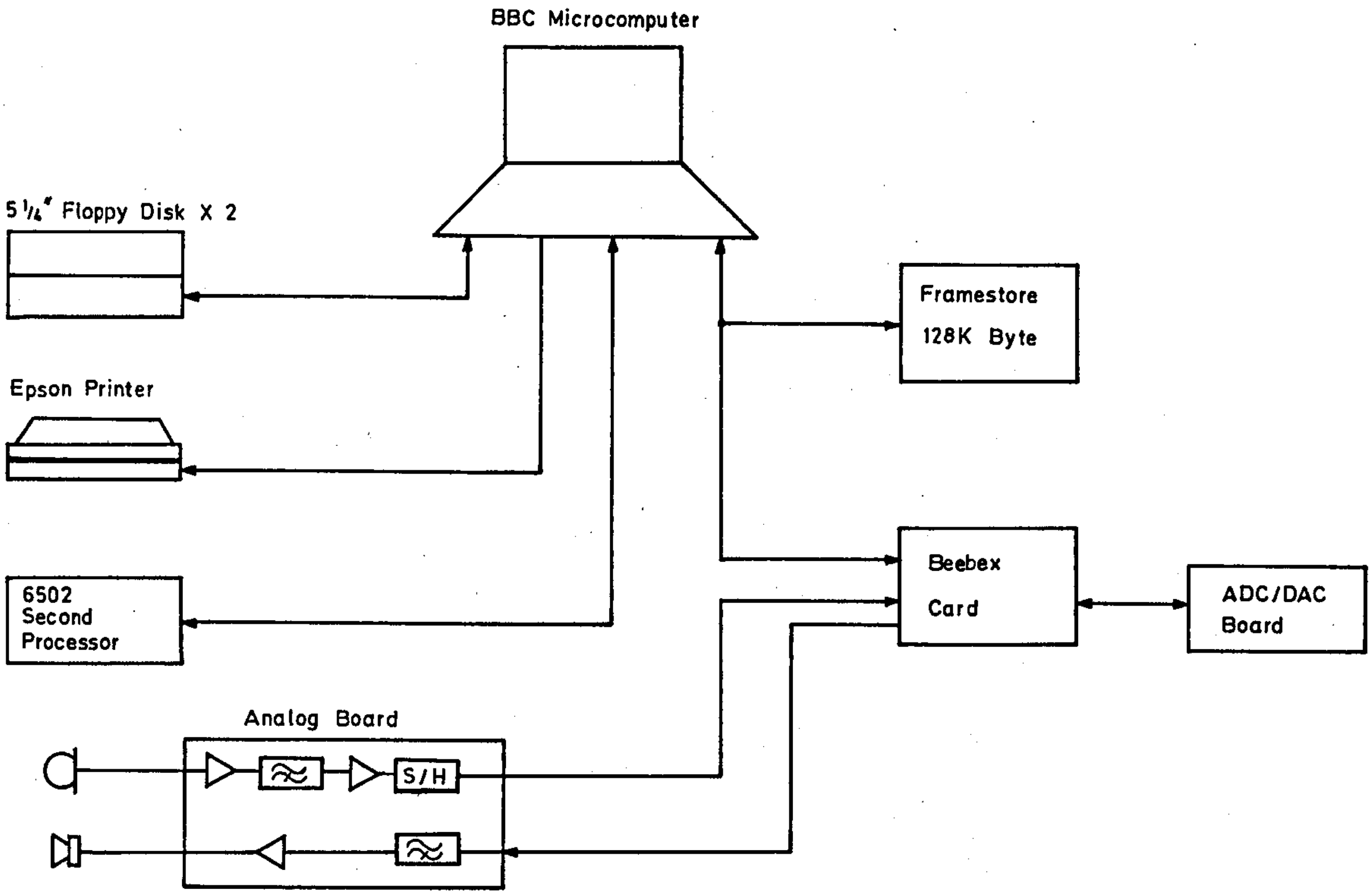


FIG.3.1 THE SIMULATION EQUIPMENT CONFIGURATION

```
    8 kHz SAMPLING OF SPEECH
      ( 12-BIT LINEAR PCM)
   LIST OF OPERATIONS :-
  (1) INPUT SPEECH
  (2) OUTPUT SPEECH
  (3) STORE SPEECH
  (4) RETRIEVE SPEECH
  (5) RESET FSTORE
  (6) EXIT
 WHICH OPERATION CODE ?
```

FIG.3.2 THE MENU OF THE DATA FLOW CONTROL PROGRAM

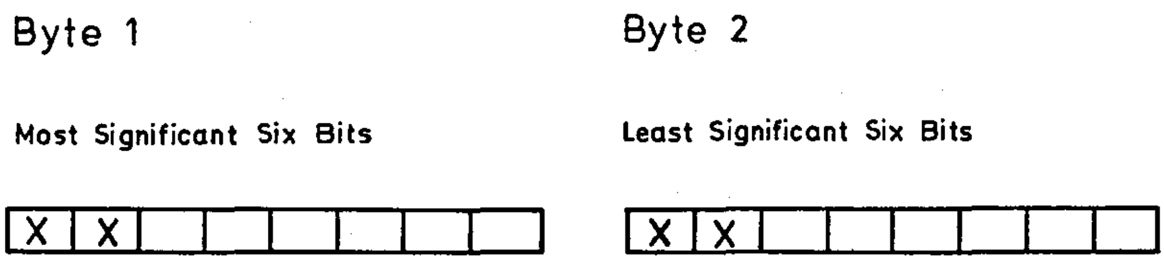


FIG.3.3 DATA FORMAT OF A SPEECH SAMPLE IN THE FRAMESTORE

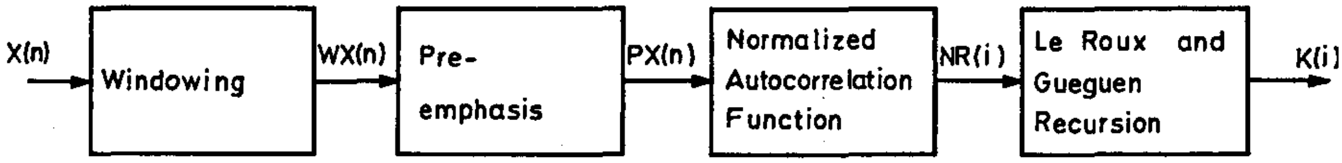


FIG.3.4 THE REFLECTION COEFFICIENT ESTIMATOR

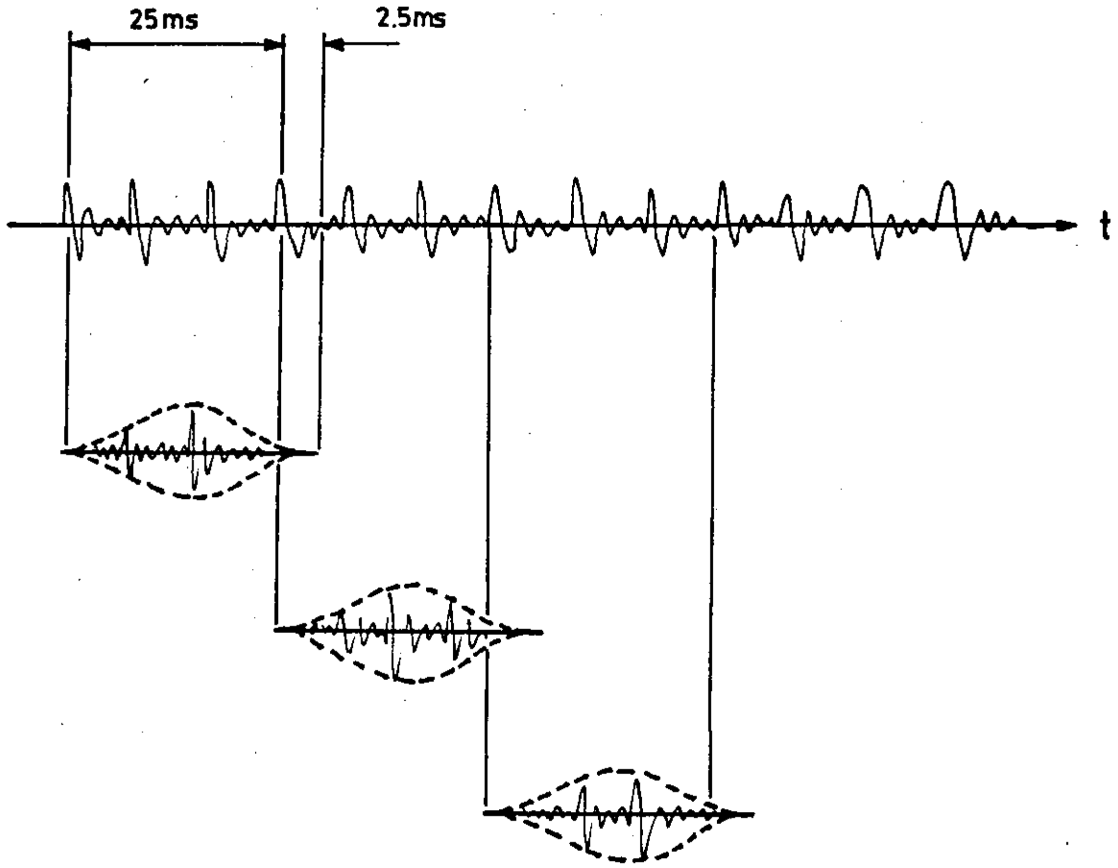
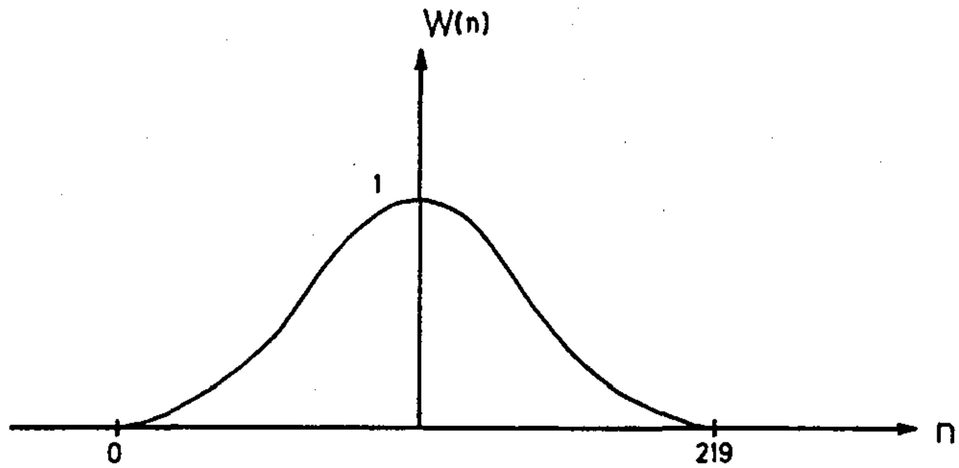


FIG.3.5 WINDOWED AND OVERLAPPING DATA BLOCKS



$W(n) = 0.5 * (1 - COS(2\pi n / 219))$

FIG.3.6 THE 220 POINTS HANNING WINDOW

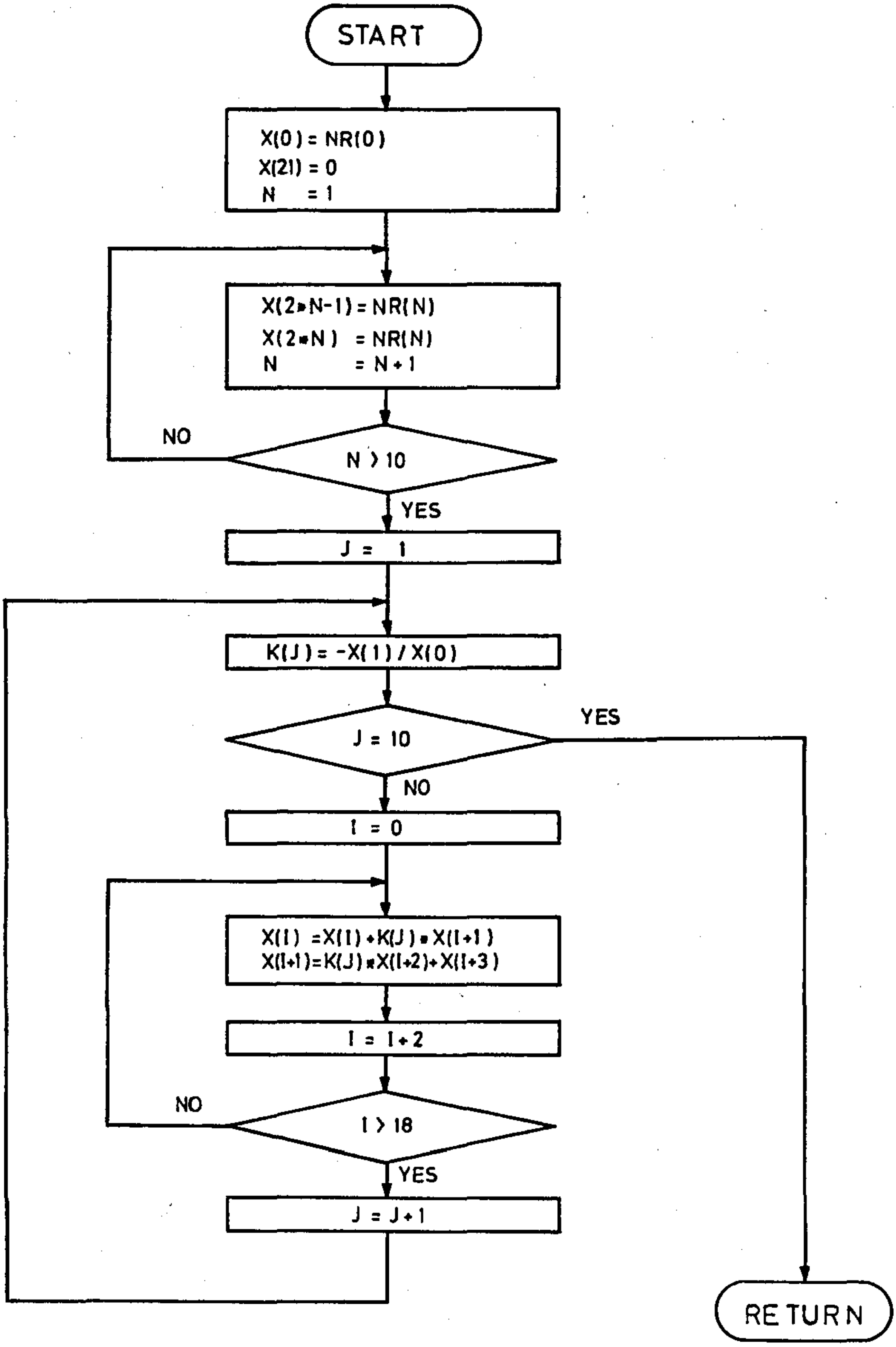


FIG.3.7 FLOWCHART OF THE LE ROUX AND GUEGUEN METHOD FOR A 10th ORDER LPC

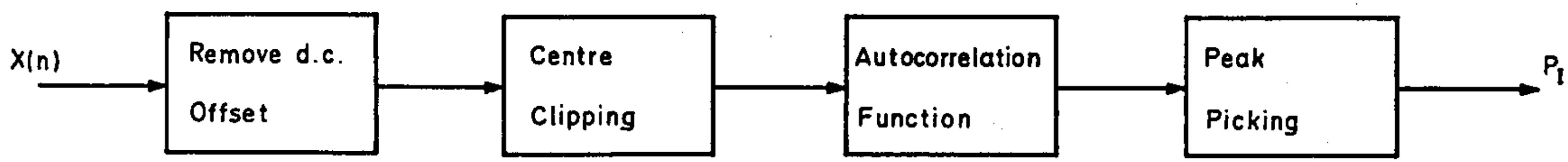


FIG.3.8 SONDHI'S METHOD FOR PITCH DETECTION

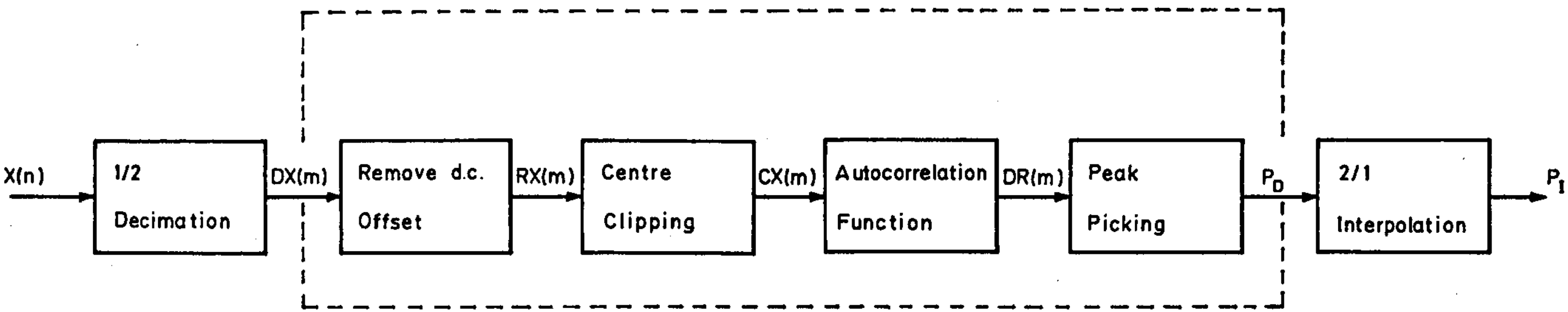
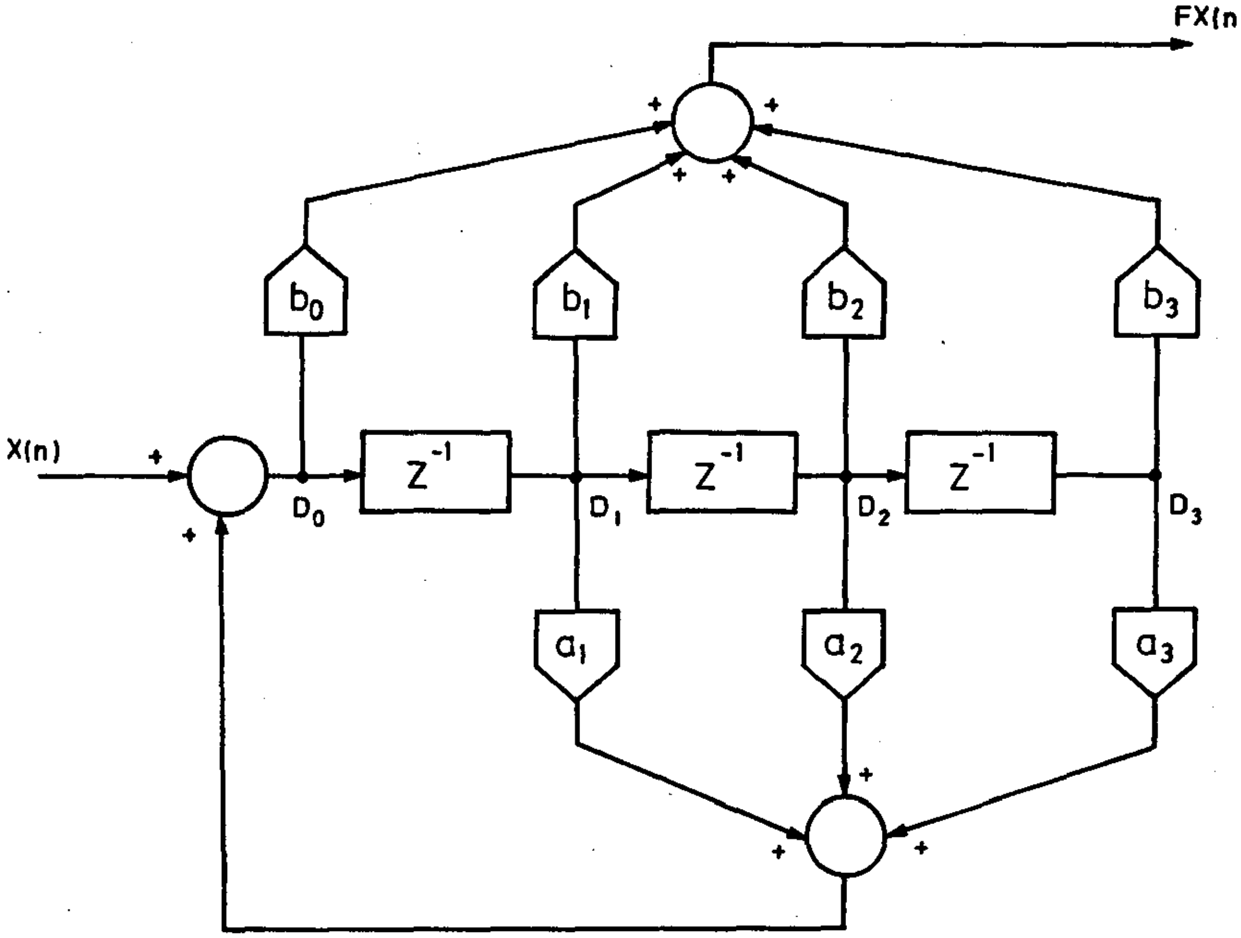


FIG.3.9 THE MODIFIED SONDHI'S METHOD FOR PITCH DETECTION



$a_1 = 1.45902906$

$a_2 = -0.910368999$

$a_3 = 0.197825187$

$b_0 = 0.0316893439$

$b_1 = 0.0950680317$

$b_2 = 0.0950680317$

$b_3 = 0.0316893439$

FIG. 3.10 THE 3rd ORDER BUTTERWORTH LOW PASS FILTER

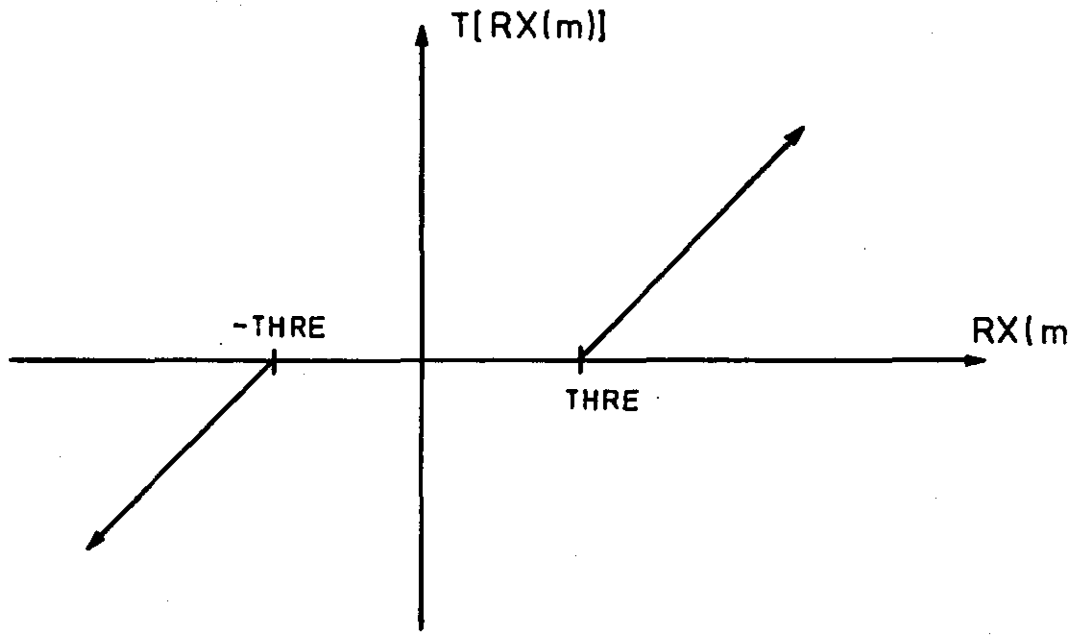


FIG.3.11 THE CENTRE-CLIPPING FUNCTION

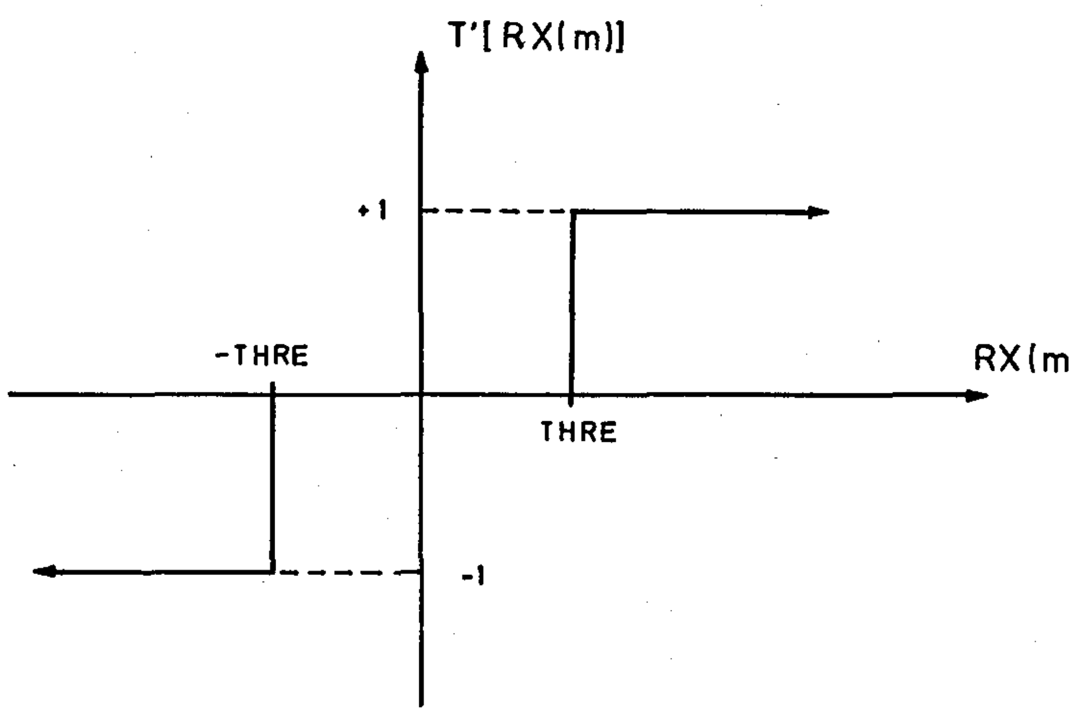
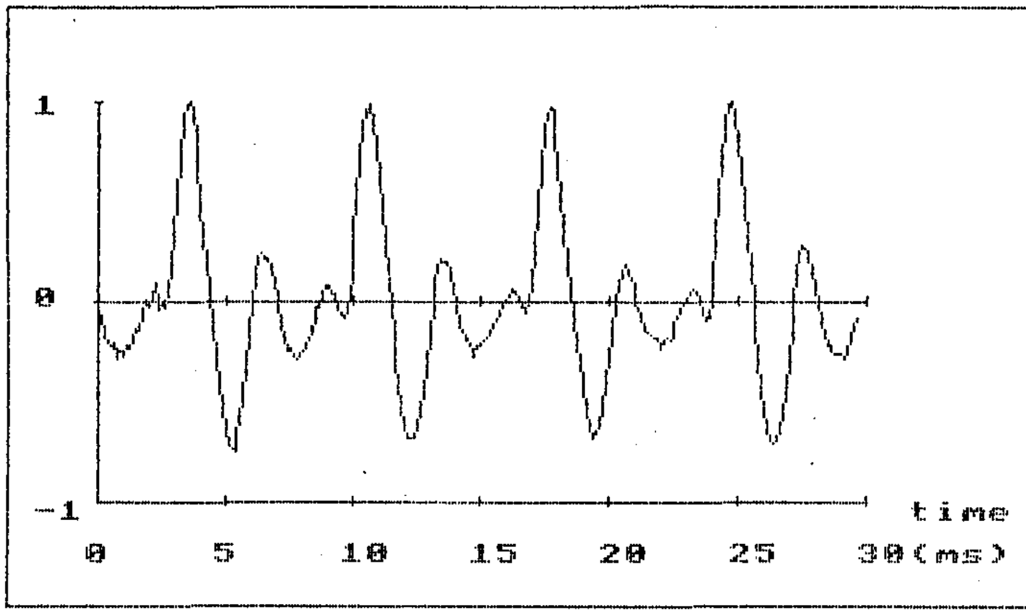


FIG.3.12 THE 3-LEVEL CENTRE-CLIPPING FUNCTION

a)



b)

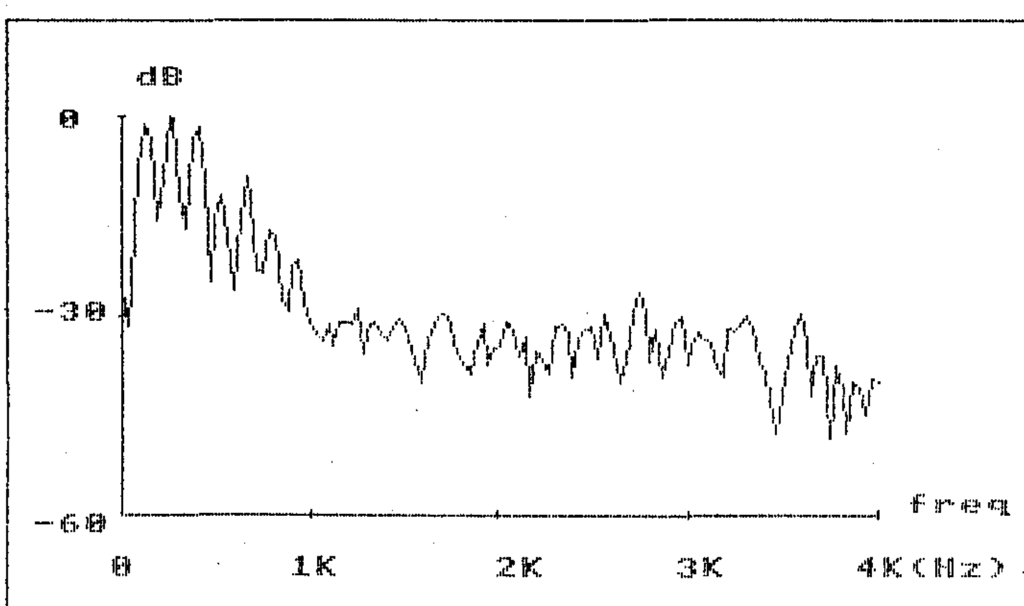
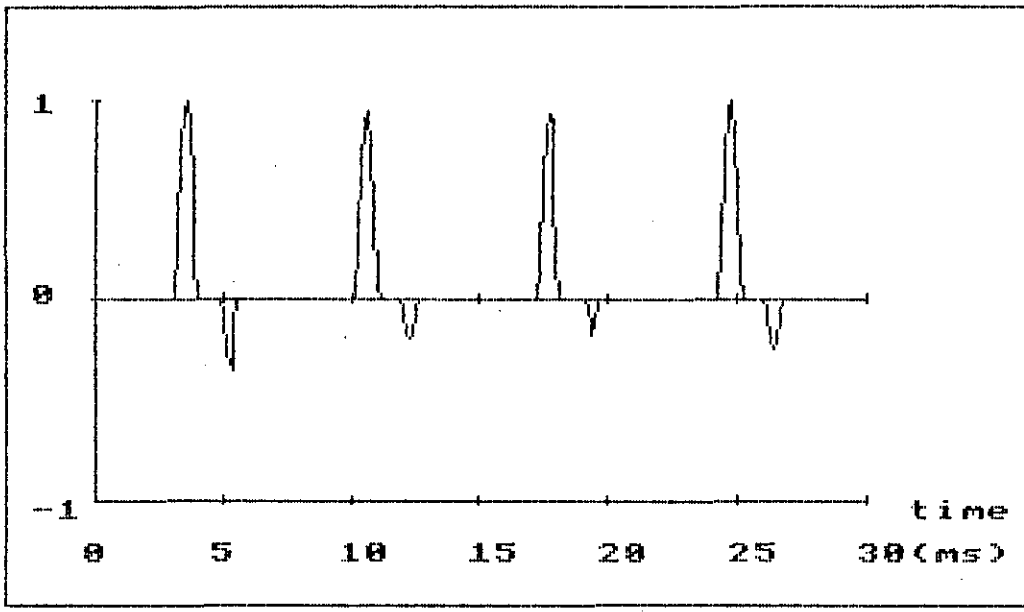


FIG. 3.13 a) A SPEECH SEGMENT FROM UTTERANCE 'ONE'

b) THE CORRESPONDING FOURIER SPECTRUM

a)



b)

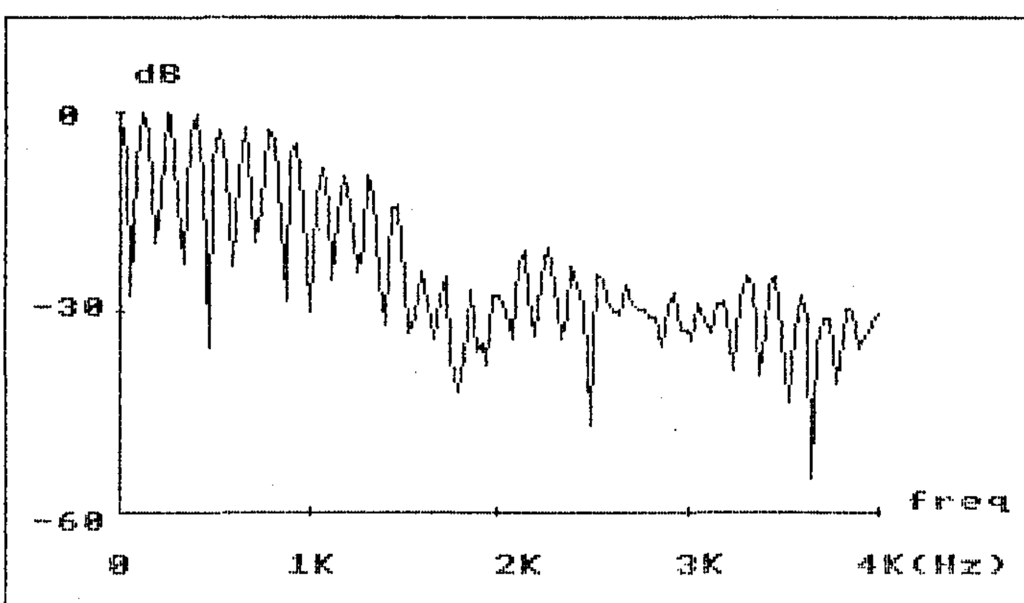
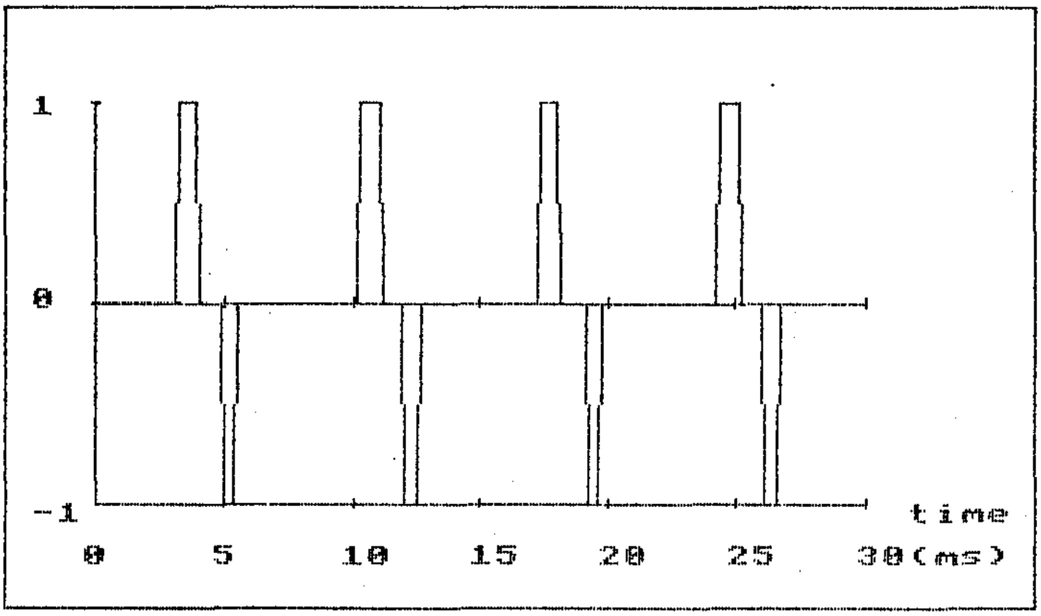


FIG.3.14 a) THE CENTRE-CLIPPING OF FIG.3.13 a

b) THE FOURIER SPECTRUM OF THE CLIPPED DATA

a)



b)

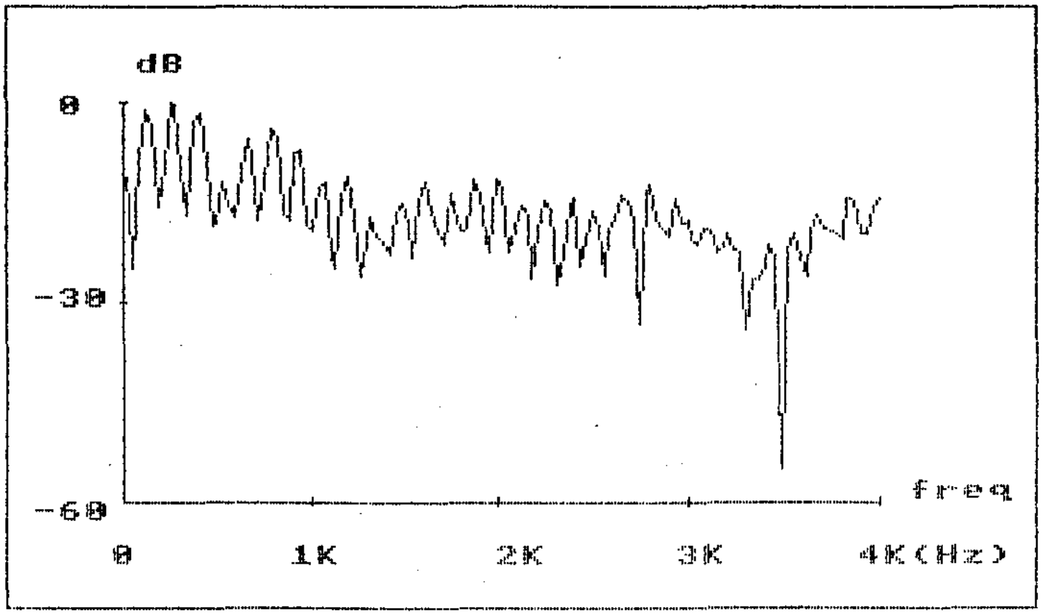


FIG.3.15 a) THE 3-LEVEL CENTRE-CLIPPING OF FIG.3.13a

b) THE FOURIER SPECTRUM OF THE CLIPPED DATA

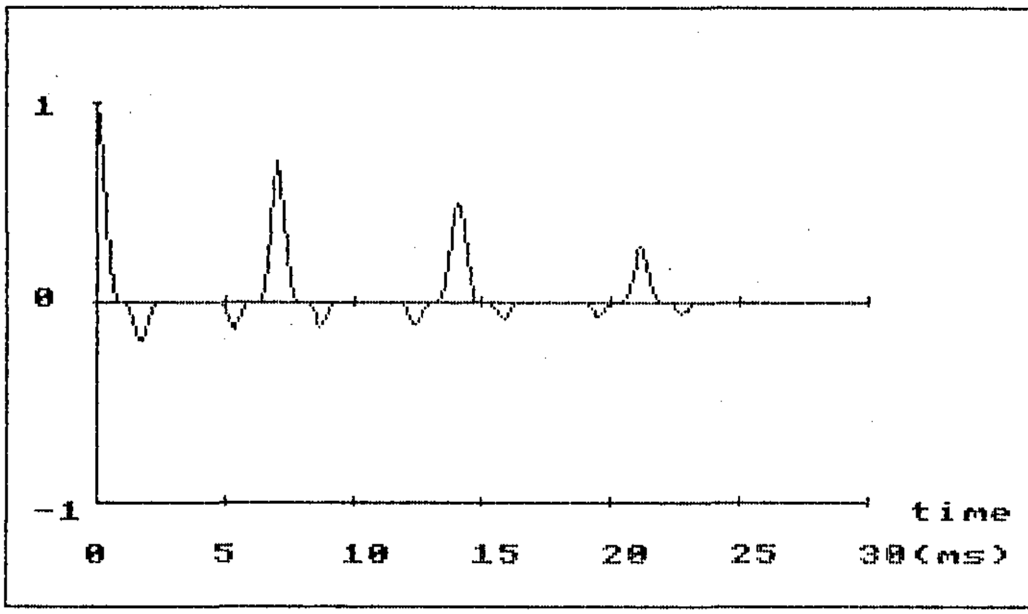


FIG. 3.16 THE AUTOCORRELATION FUNCTION OF FIG. 3.14 a

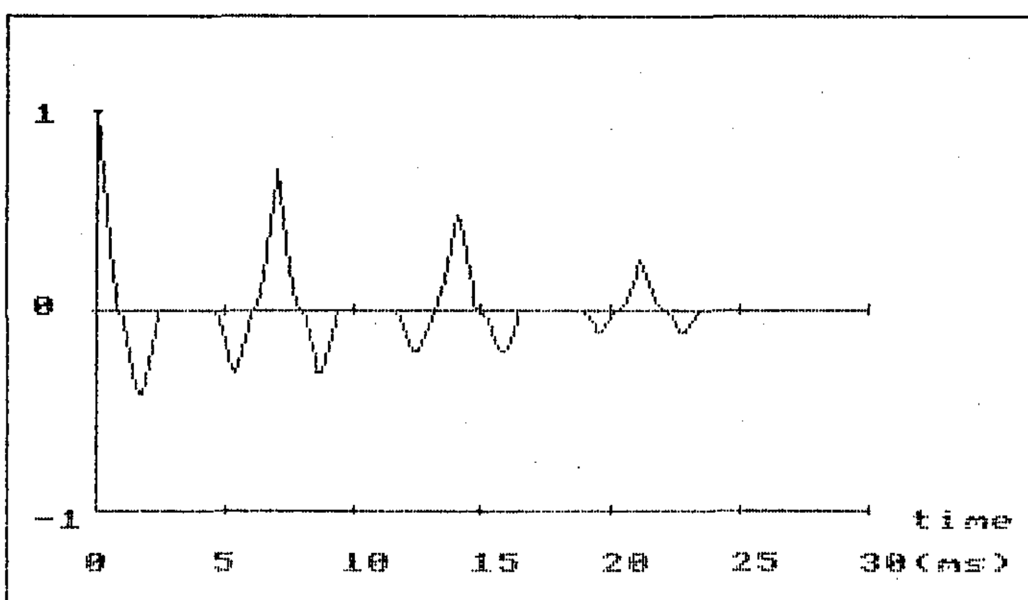


FIG. 3.17 THE AUTOCORRELATION FUNCTION OF FIG. 3.15 a

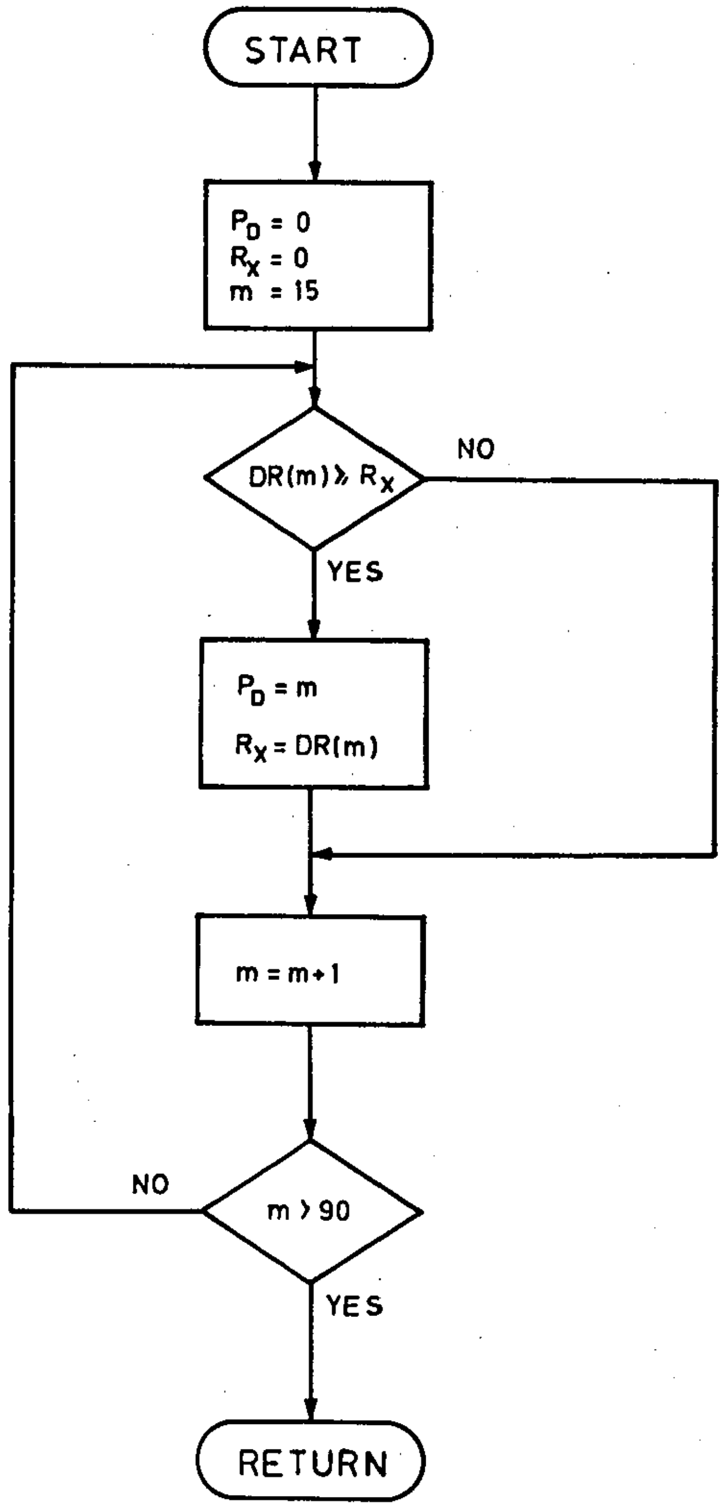


FIG.3.18 FLOWCHART OF THE PEAK PICKING PROCEDURE

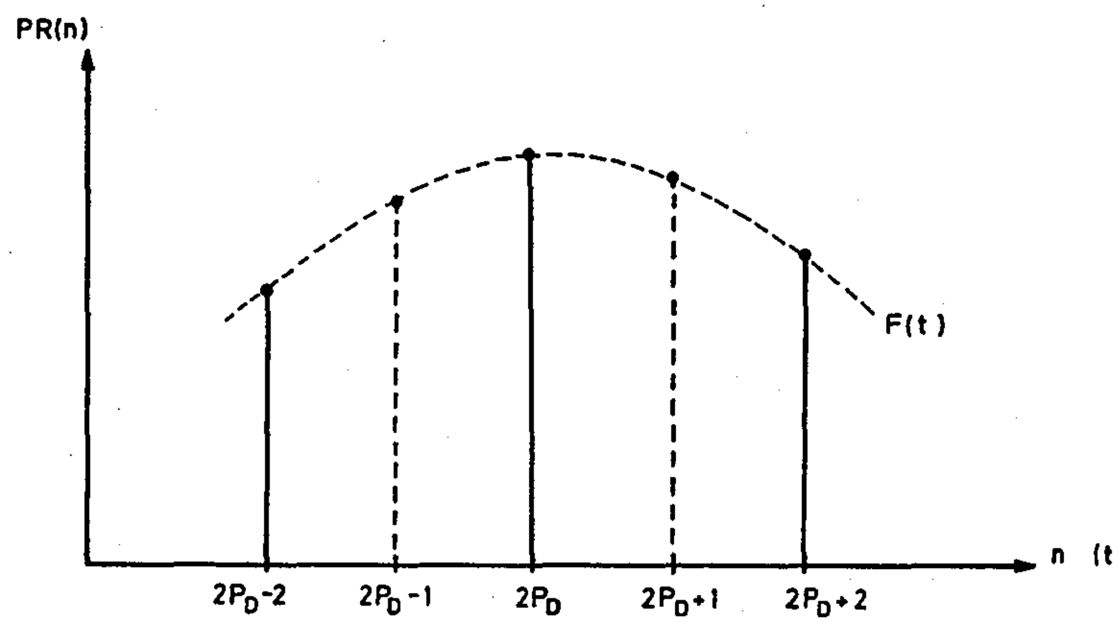


FIG.3.19 THE 2/1 QUADRATIC INTERPOLATION

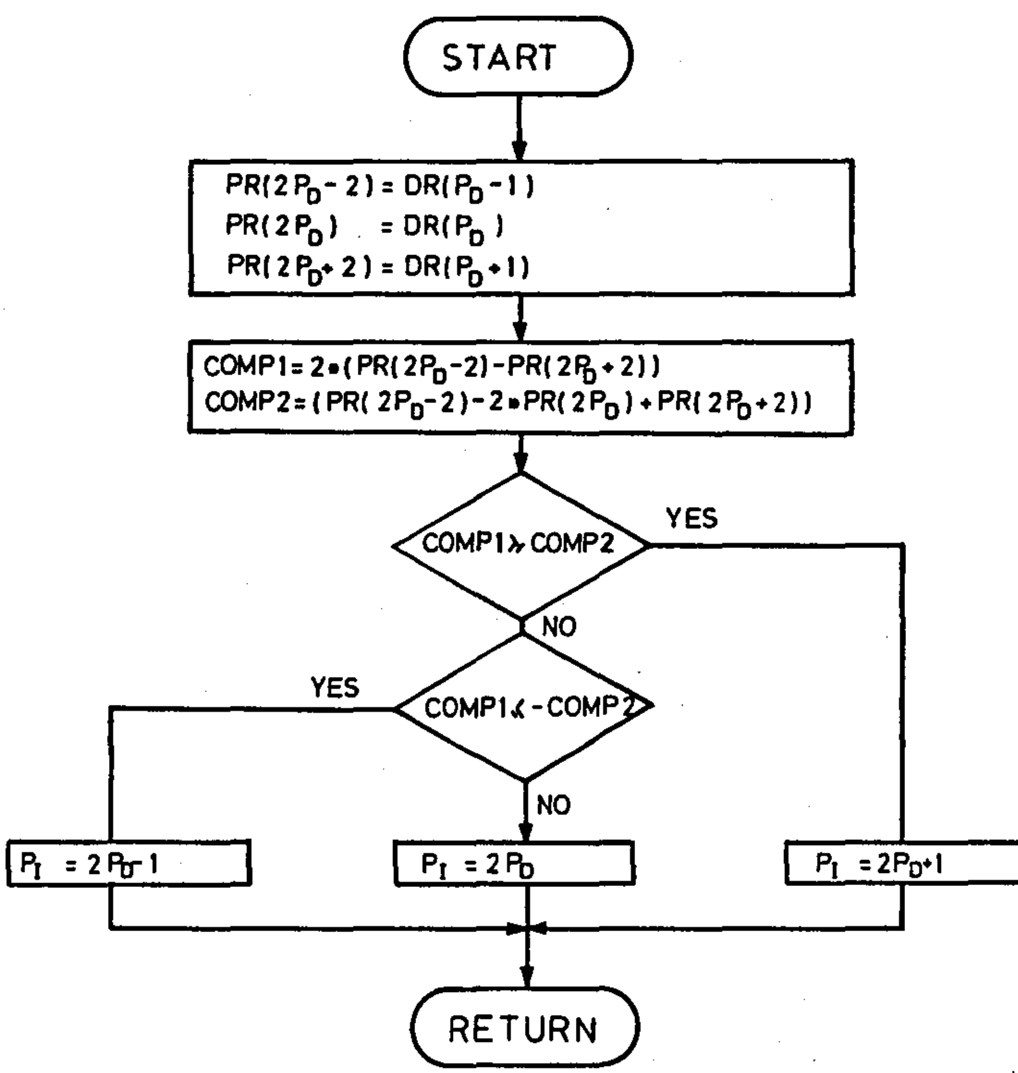


FIG.3.20 FLOWCHART OF THE INTERPOLATION PROCEDURE

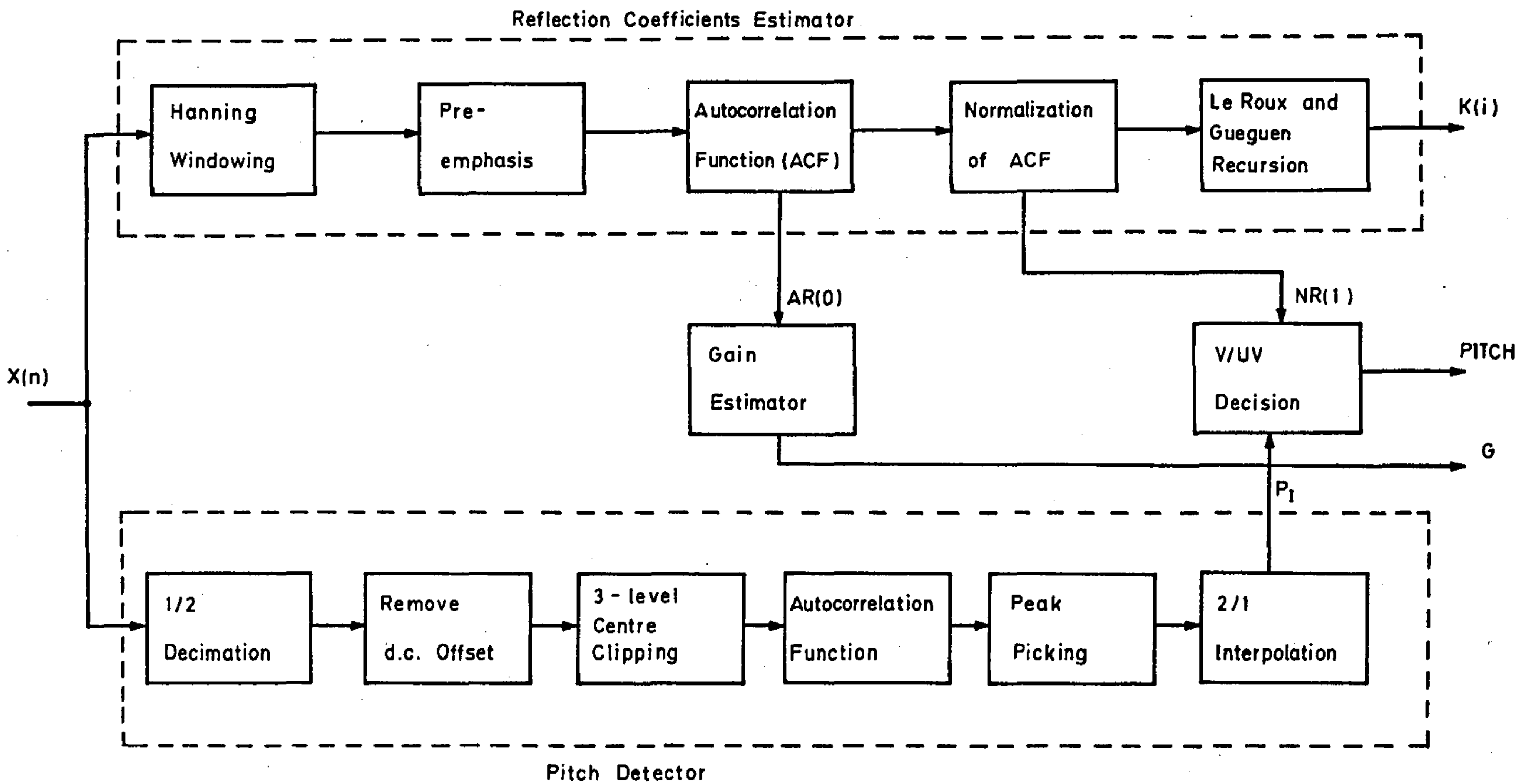


FIG.3.21 THE COMPLETE LPC ANALYSER CONFIGURATION

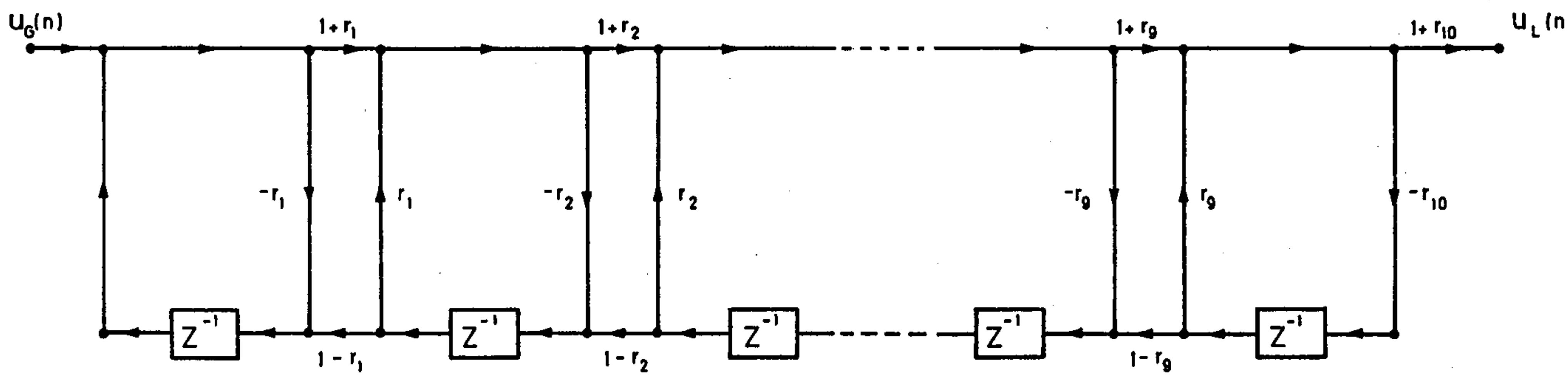


FIG.3.22 THE 10th ORDER VOCAL TRACT LATTICE FILTER (INFINITE GLOTTAL IMPEDANCE)

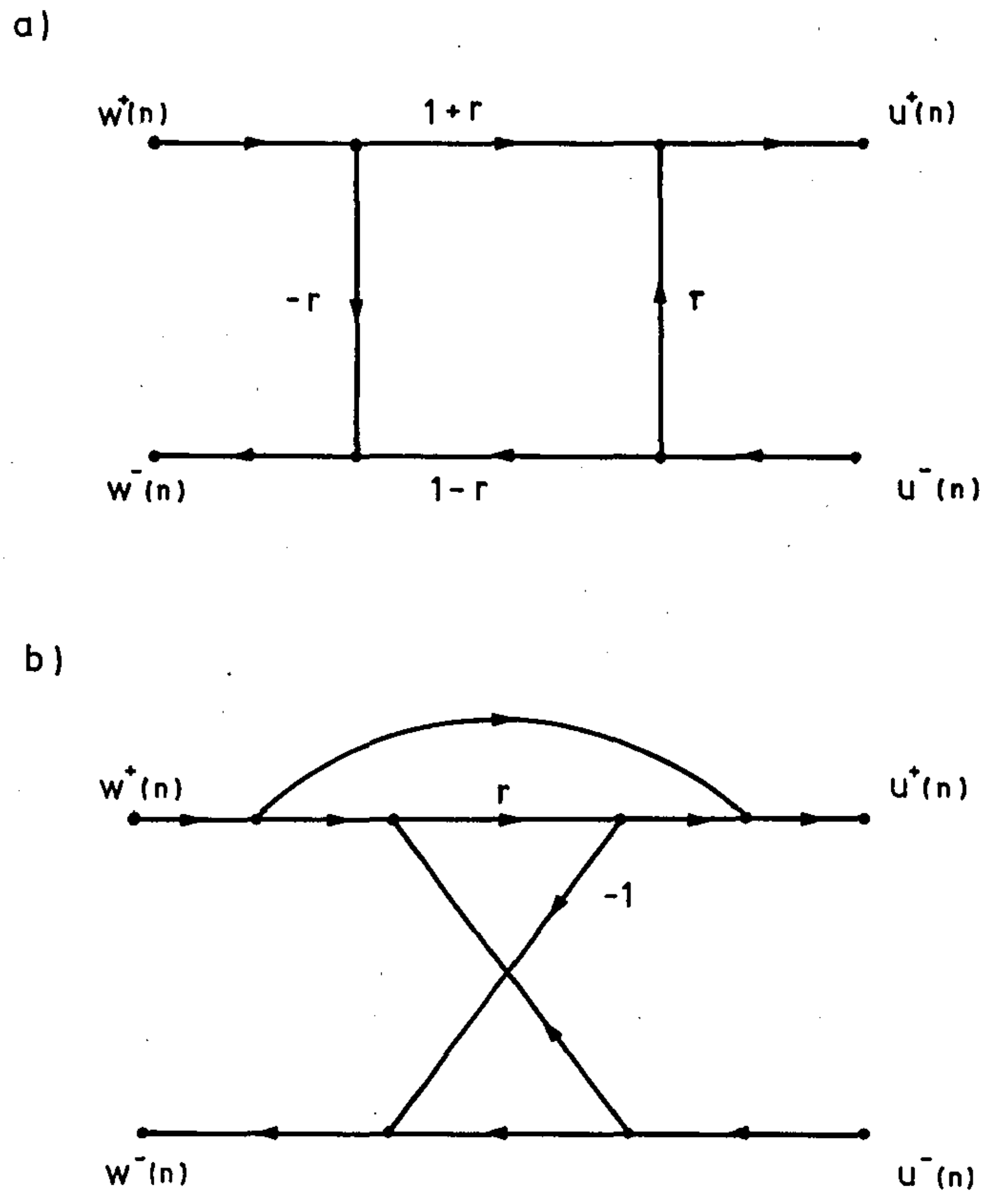


FIG.3.23 a) THE FOUR MULTIPLIER REPRESENTATION OF A LOSSLESS TUBE JUNCTION

b) THE CORRESPONDING ONE MULTIPLIER CONFIGURATION

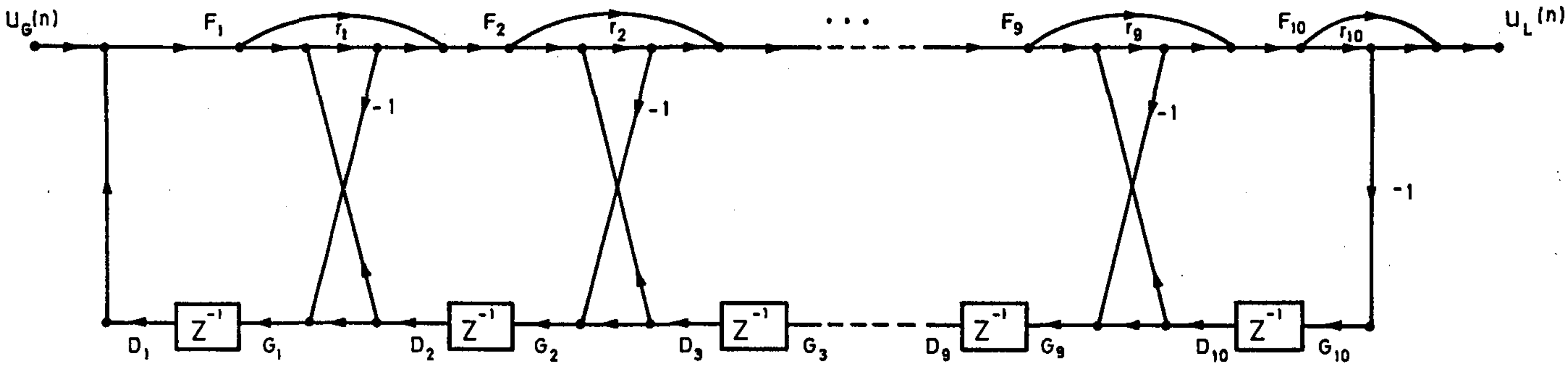


FIG.3.24 THE 10th ORDER LATTICE FILTER USING ONE MULTIPLIER JUNCTION

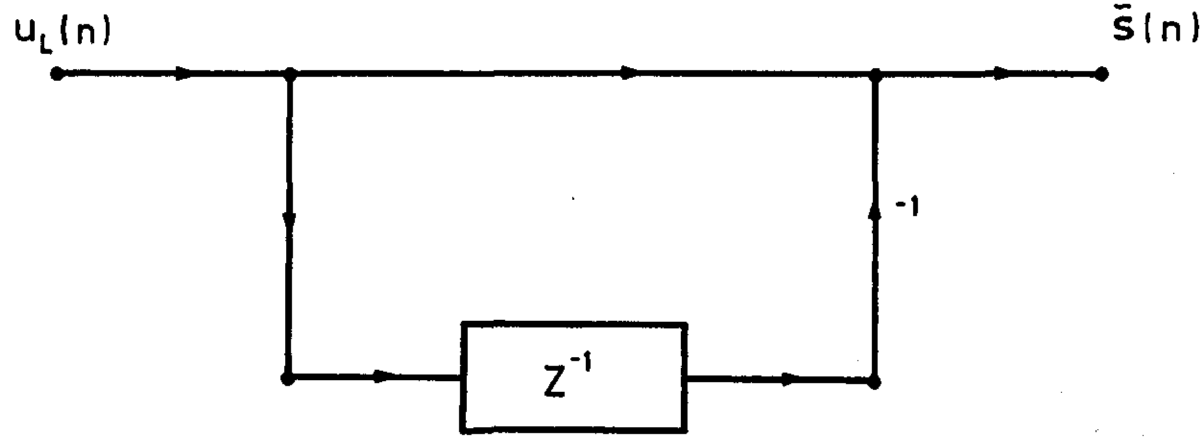
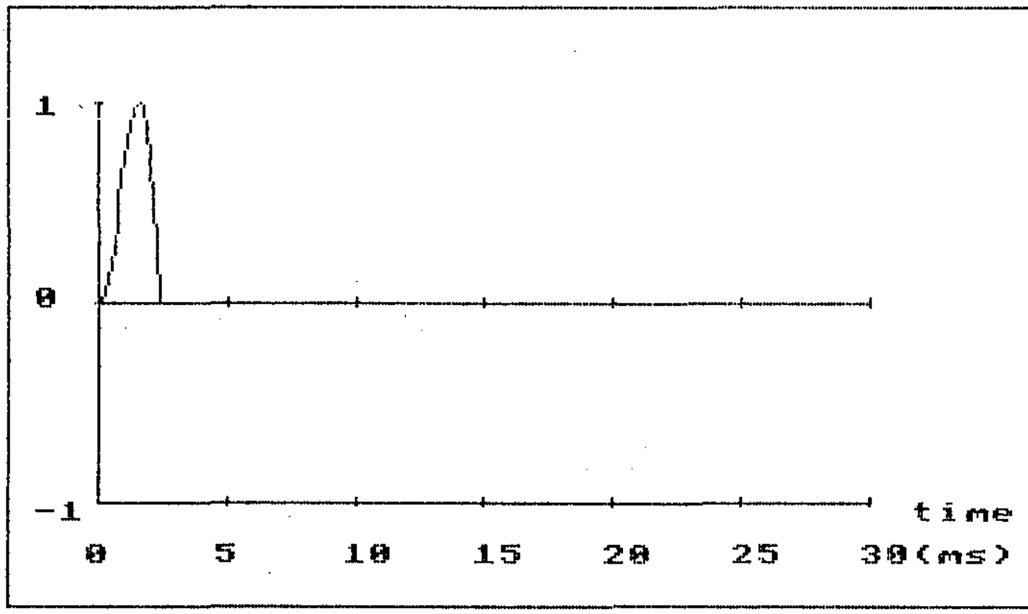
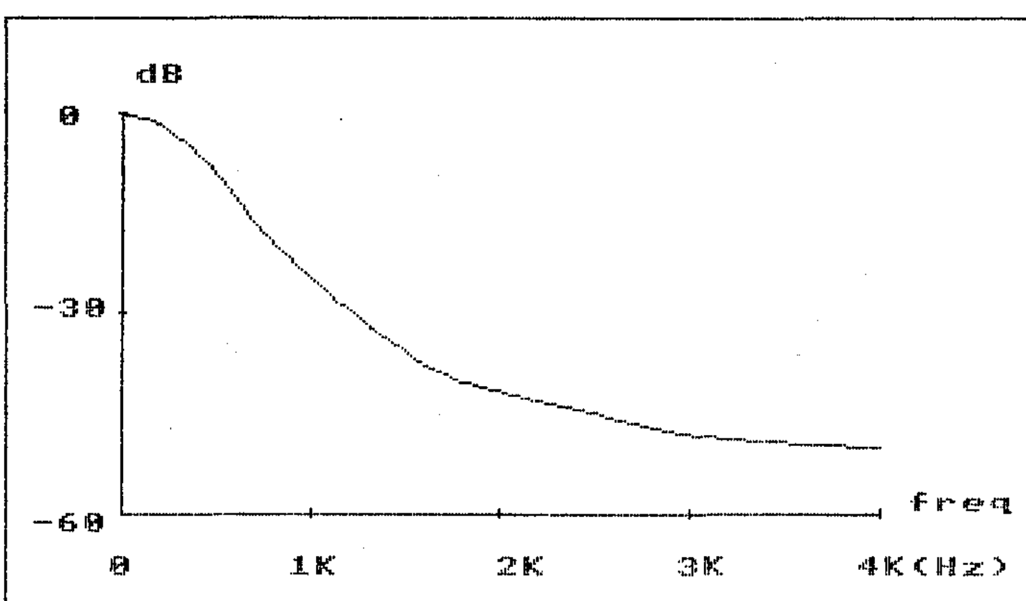


FIG.3.25 THE RADIATION MODEL

a)



b)



<u>FIG.3.26</u> a) ROSENBERG APPROXIMATION TO GLOTTAL PULSE FOR N1=14 AND N2=6

b) THE CORRESPONDING FOURIER SPECTRUM

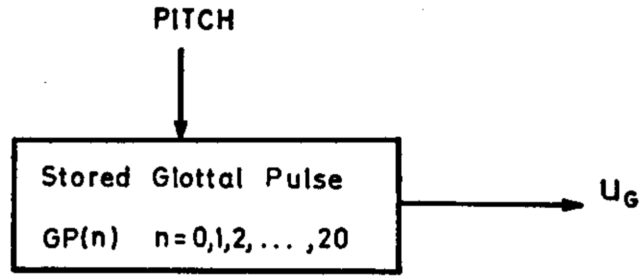


FIG.3.27 THE GLOTTAL PULSE GENERATOR

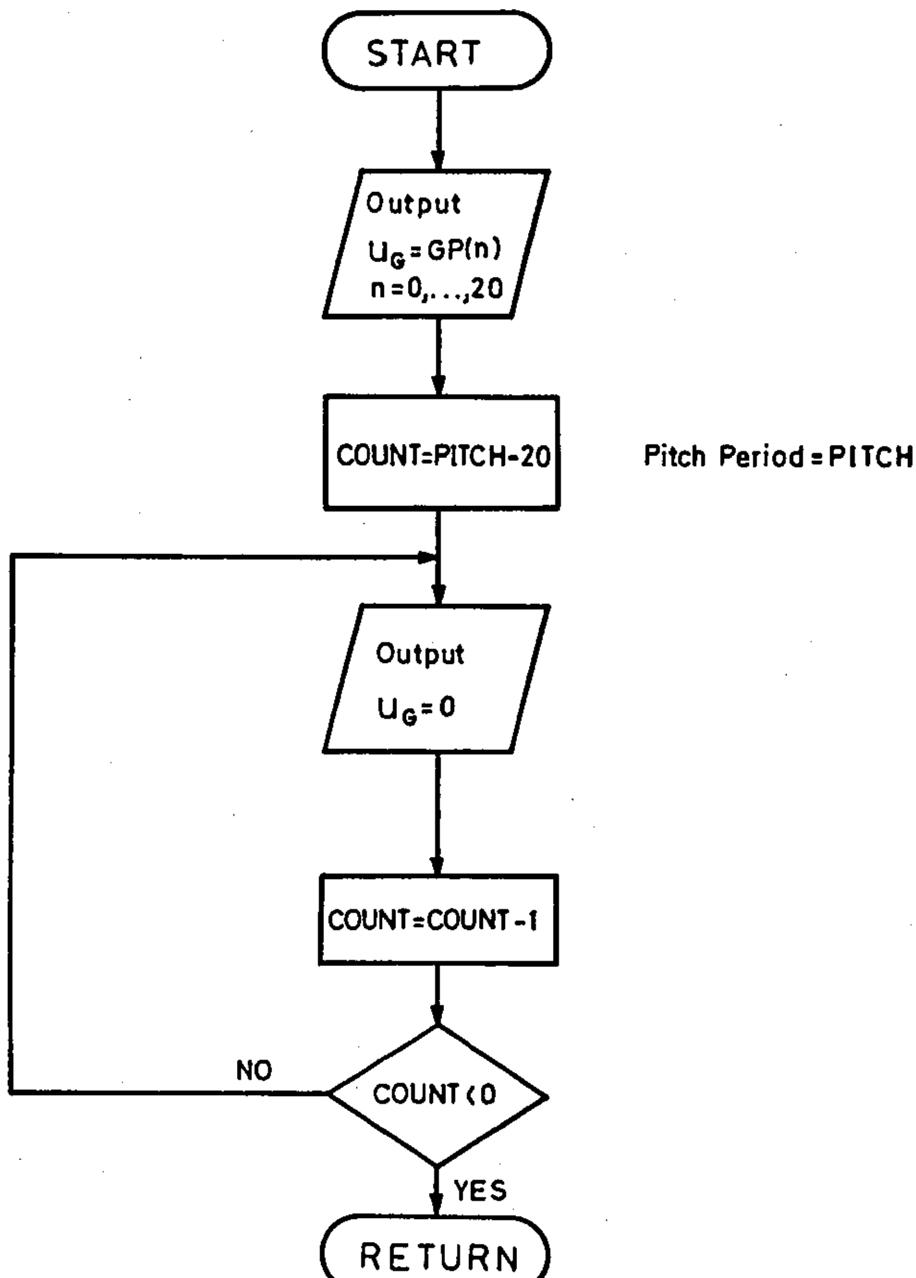


FIG.3.28 FLOWCHART OF THE GLOTTAL PULSE GENERATOR SUBROUTINE

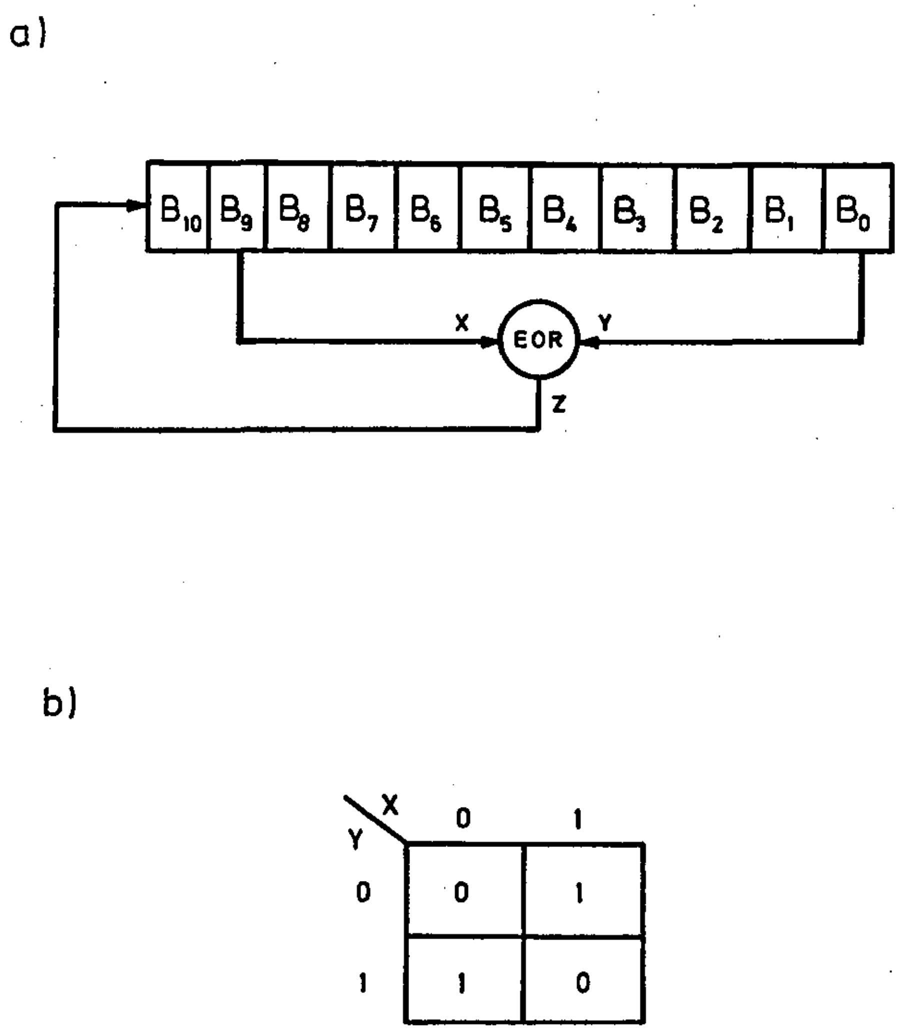
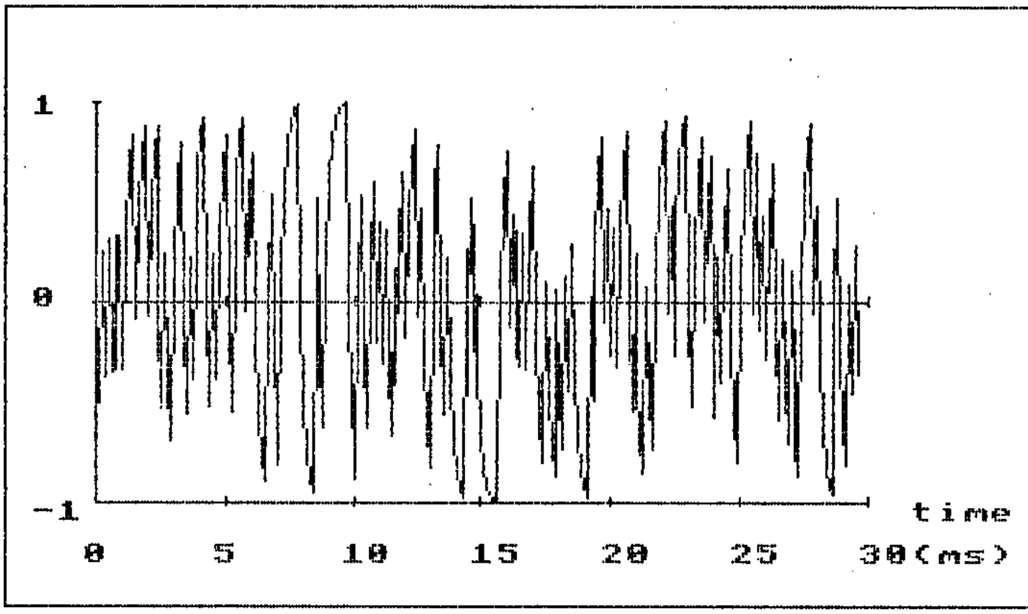


FIG.3.29 a) THE RANDOM NOISE GENERATOR

b) K-MAP OF Z

a)



b)

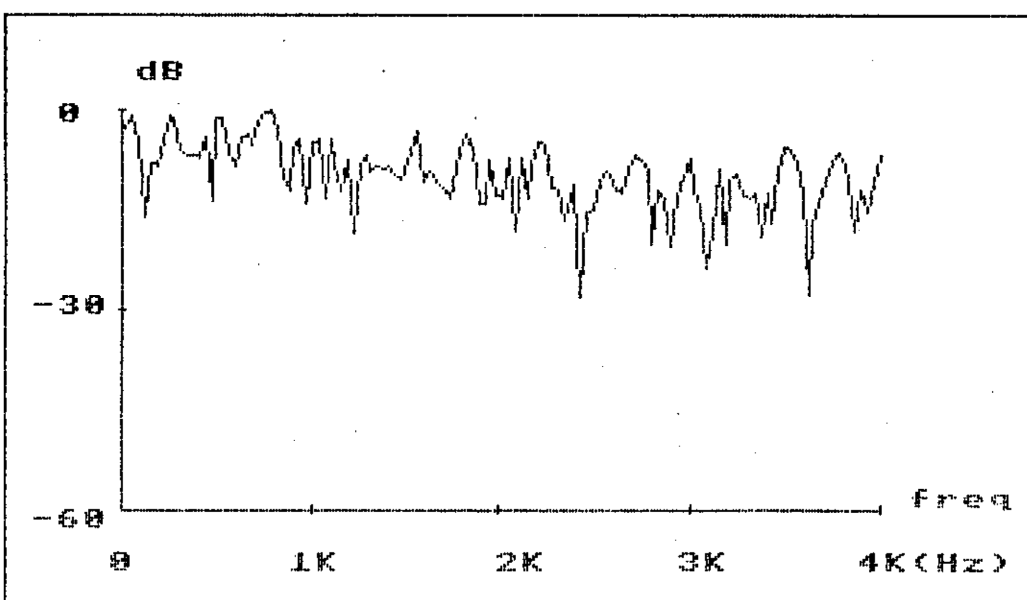


FIG.3.30 a) A SEGMENT OF THE RANDOM NOISE SIGNAL 'NOISE'

b) THE CORRESPONDING FOURIER SPECTRUM

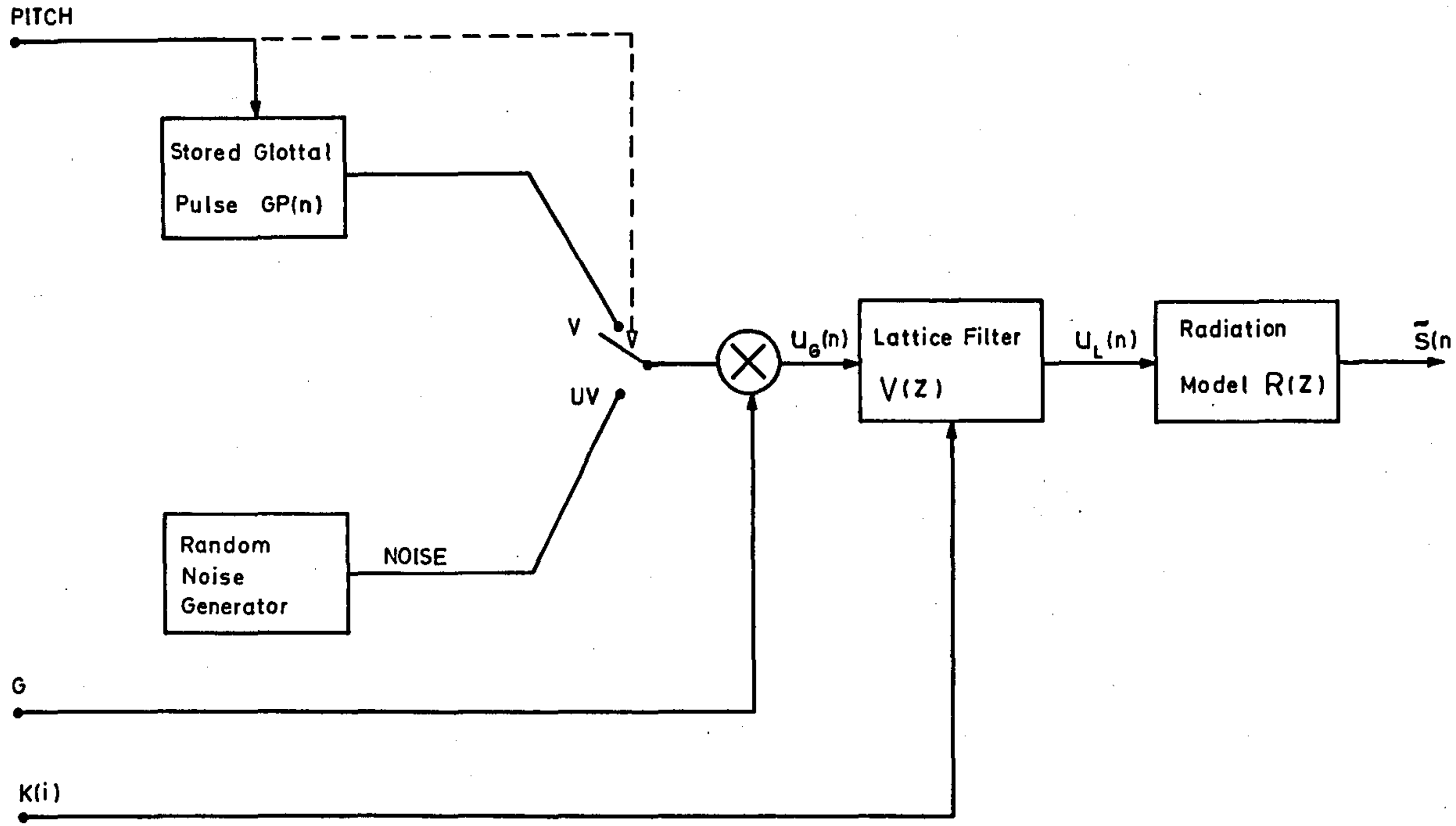


FIG.3.31 THE COMPLETE LPC SYNTHESIZER CONFIGURATION

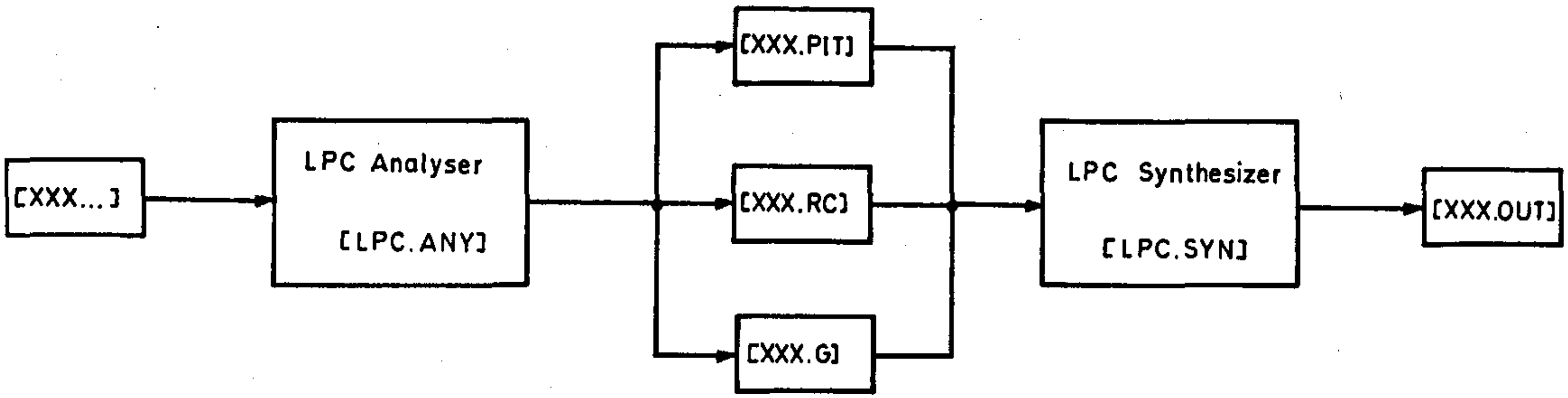


FIG.3.32 FILE HANDLING CONFIGURATION OF THE LPC SIMULATION

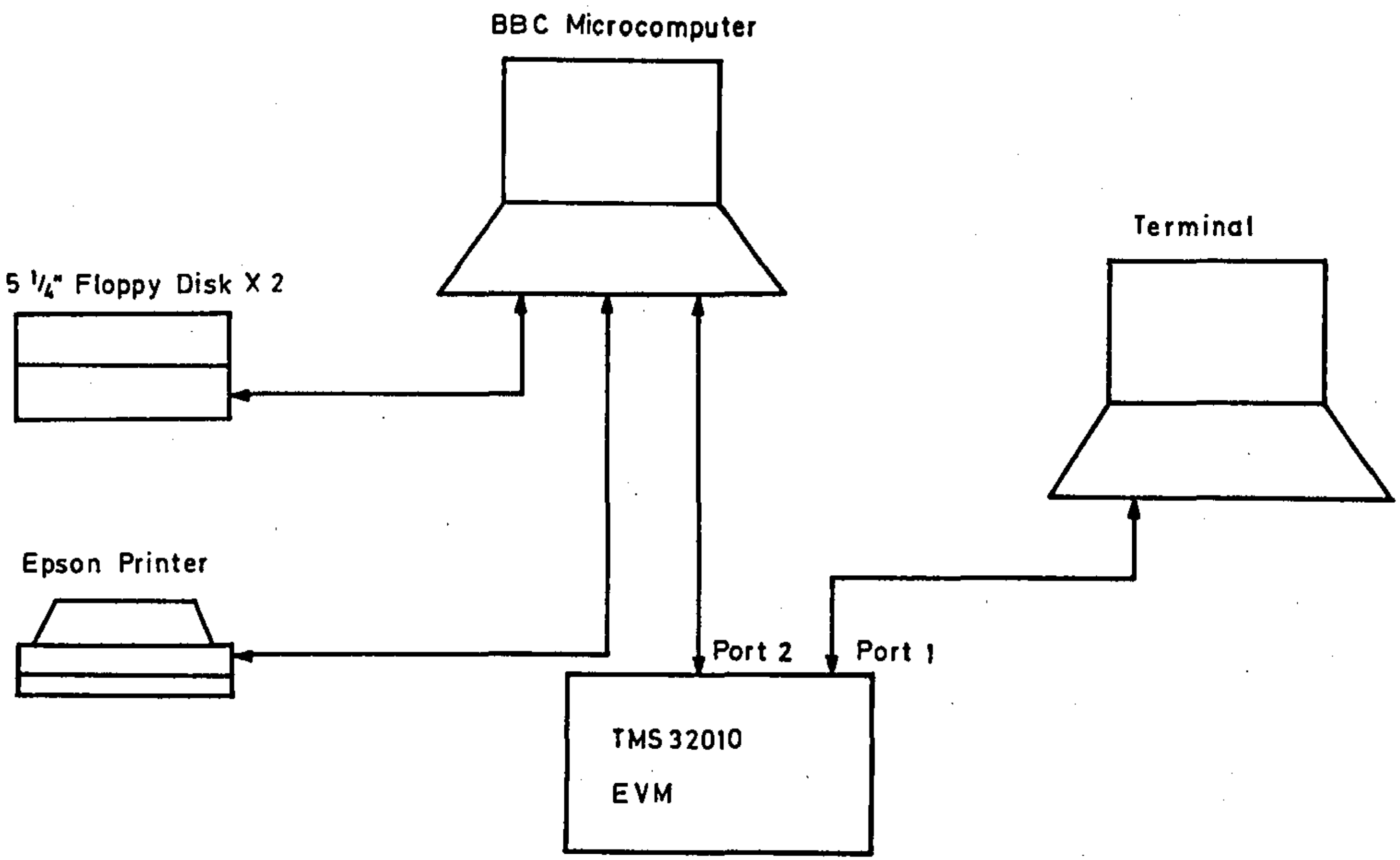


FIG. 4.1 THE TMS32010 SOFTWARE DEVELOPMENT SYSTEM CONFIGURATION

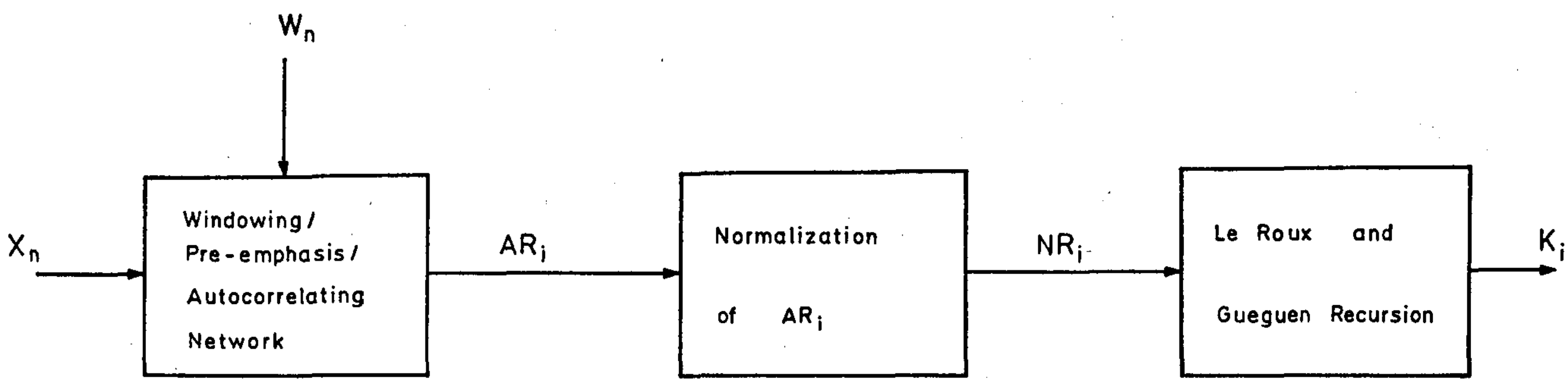


FIG. 4.2 TMS32010 SOFTWARE FOR THE REFLECTION COEFFICIENT ESTIMATOR

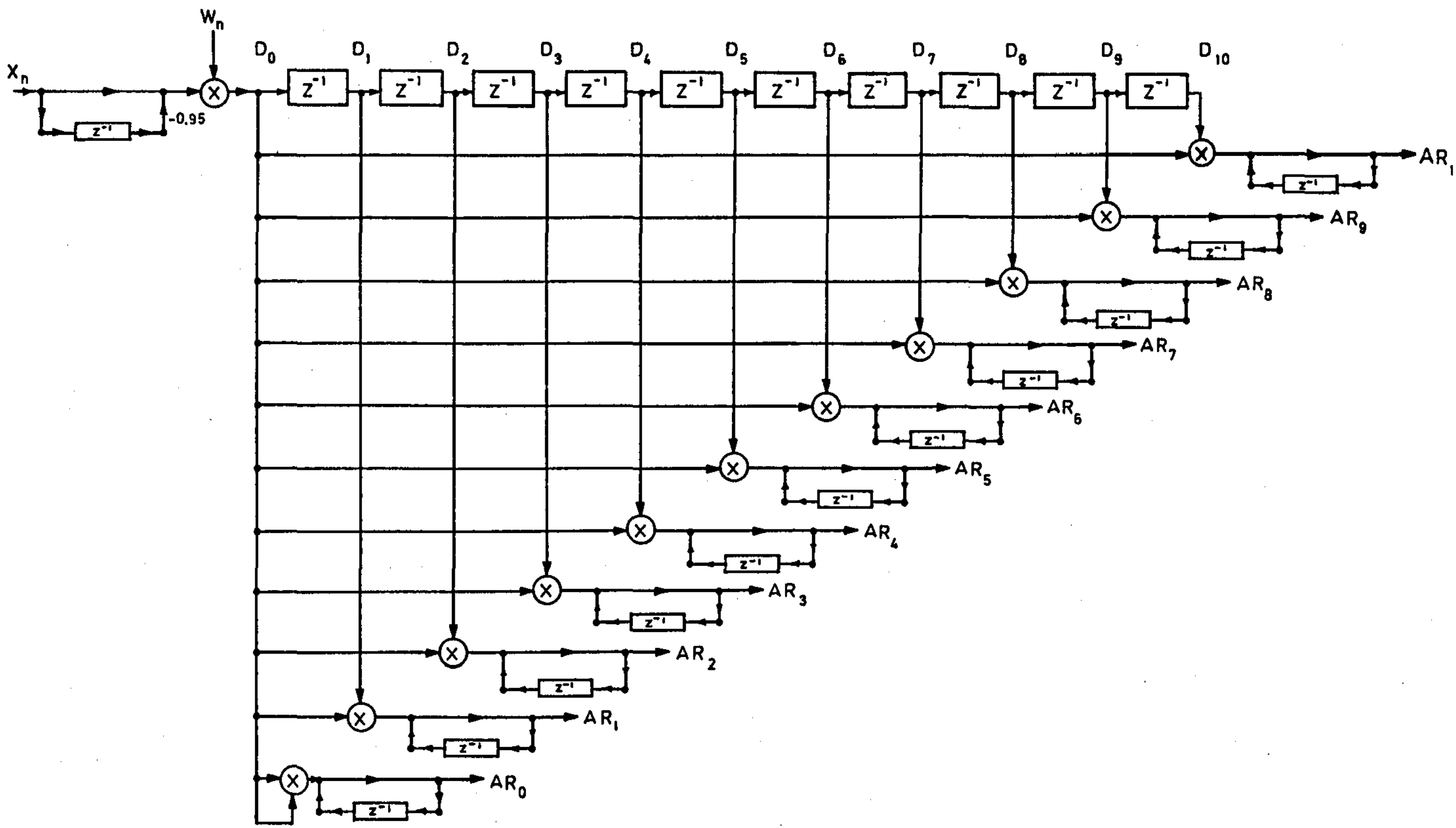


FIG.4.3 THE WINDOWING / PRE-EMPHASIS / AUTOCORRELATING NETWORK

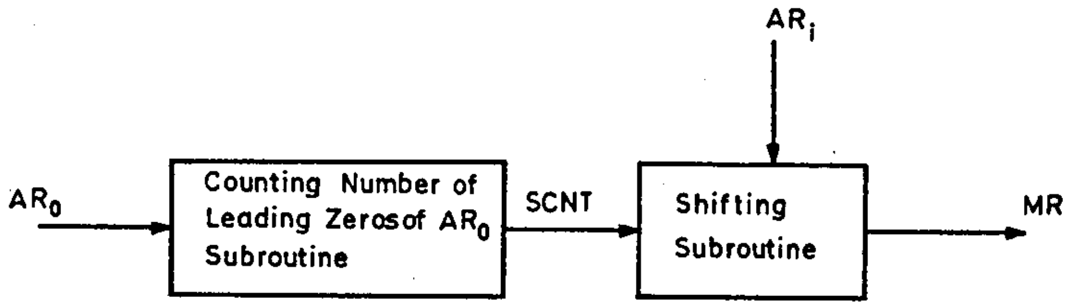


FIG.4.4 AUTOCORRELATION FUNCTION 32-BIT TO 16-BIT TRANSFORMATION

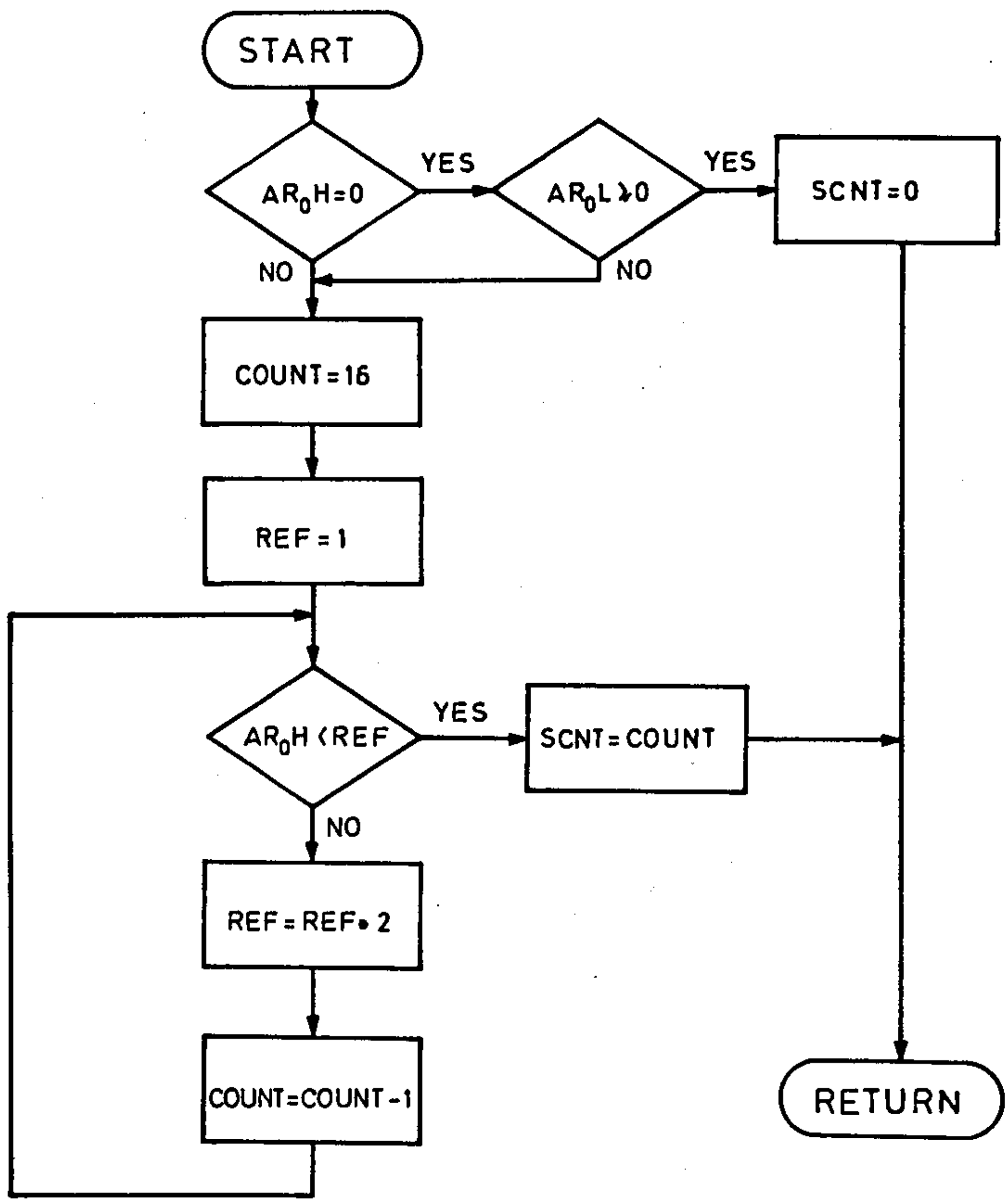


FIG.4.5 FLOWCHART OF THE AR_0 LEADING ZEROS COUNTING SUBROUTINE

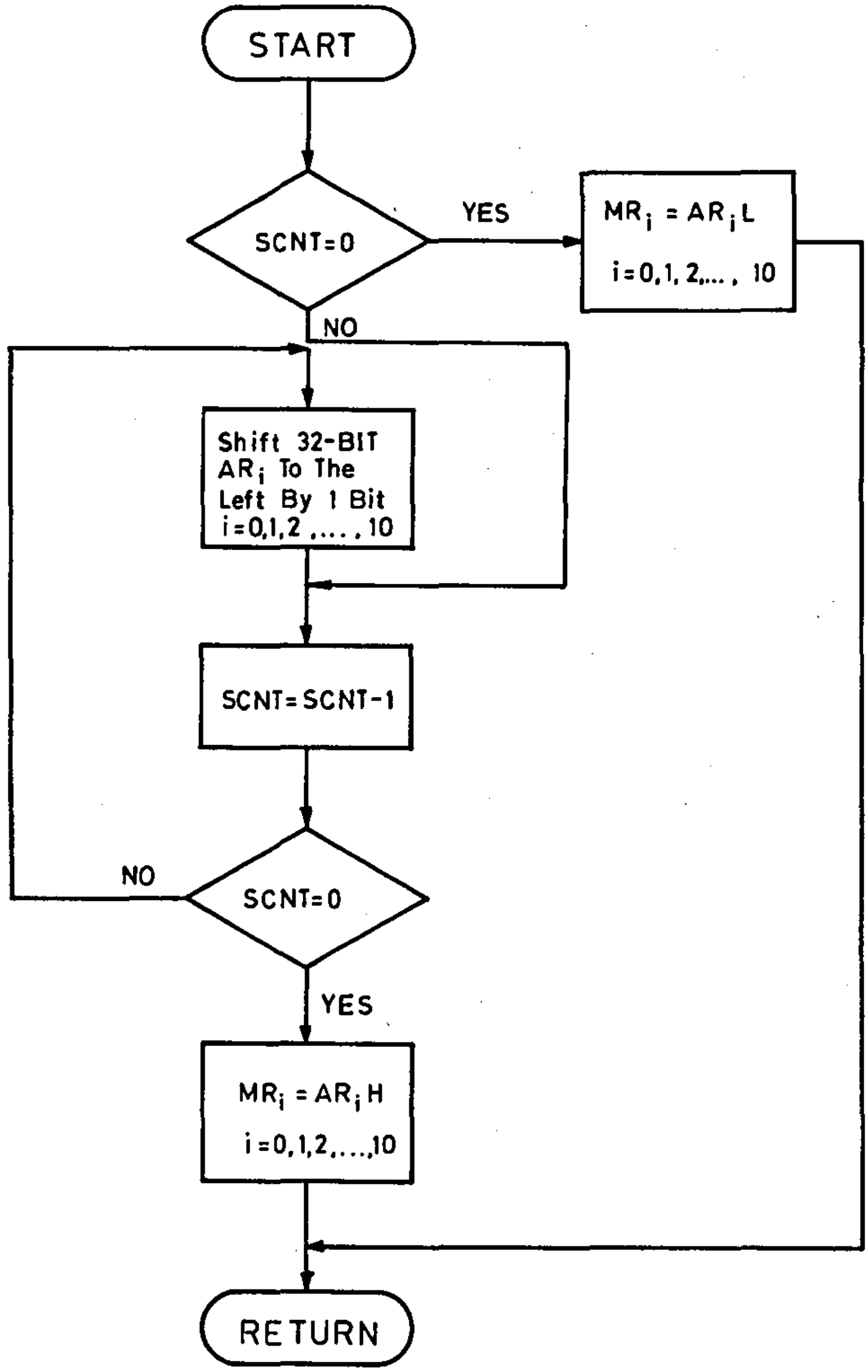


FIG.4.6 FLOWCHART OF THE SHIFTING SUBROUTINE

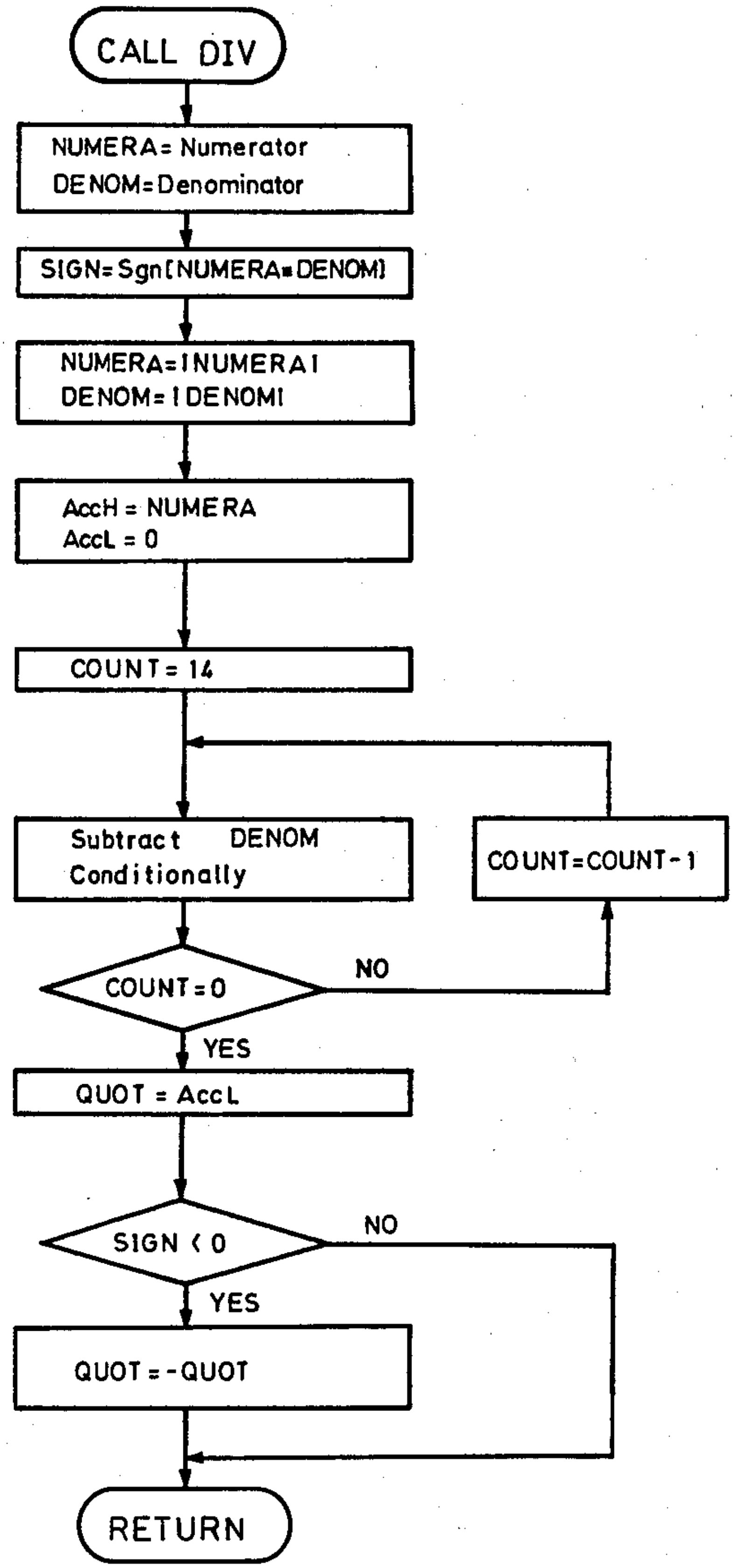


FIG.4.7a FLOWCHART OF THE DIVISION SUBROUTINE (DIV)

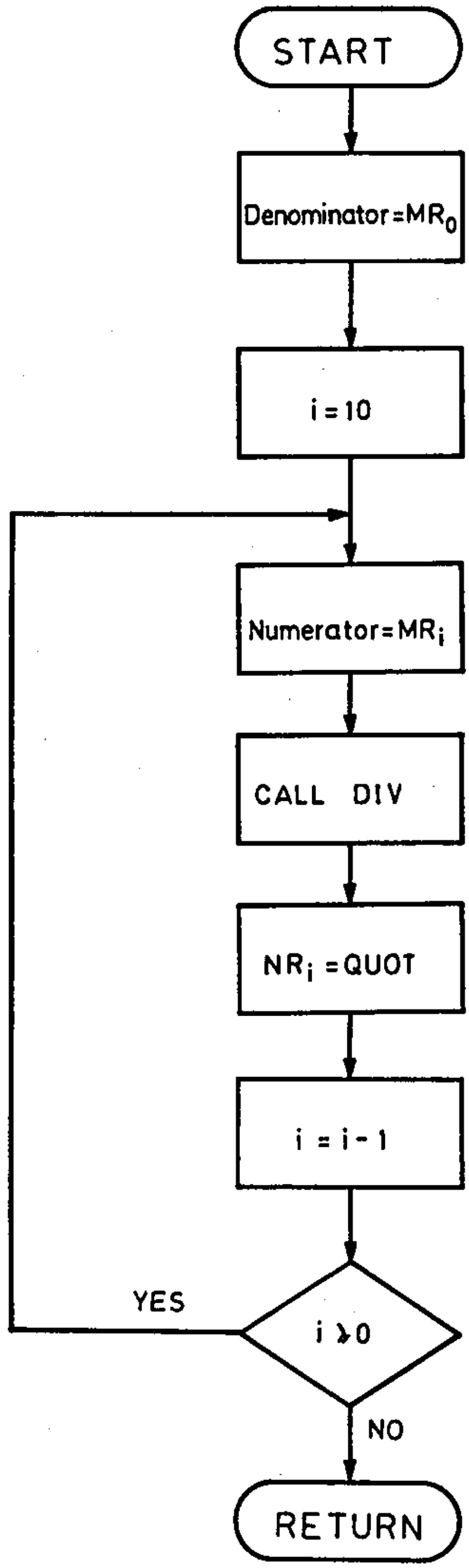


FIG.4.7b FLOWCHART OF THE NORMALIZATION SUBROUTINE

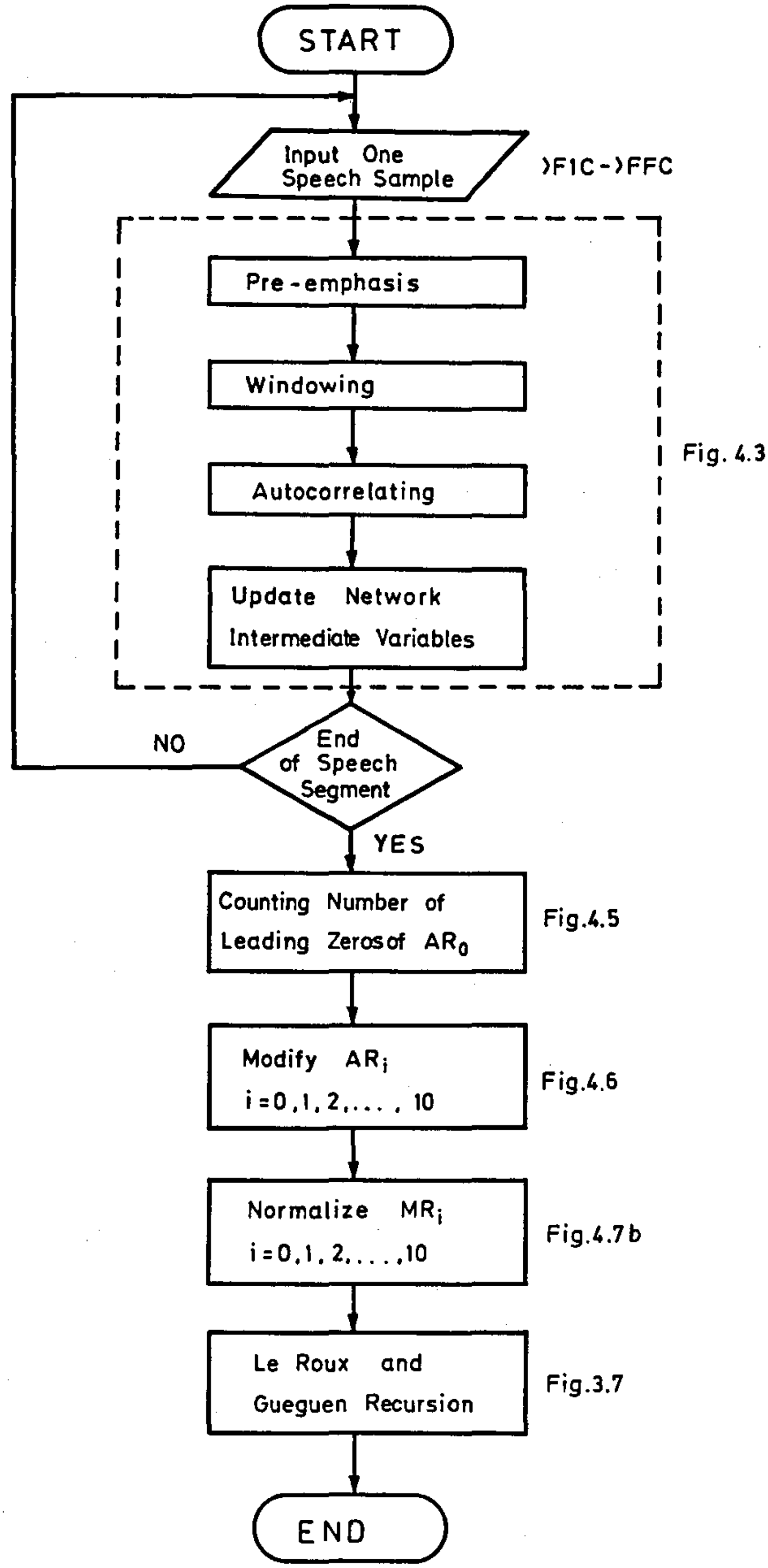


FIG. 4.8 FLOWCHART OF THE REFLECTION COEFFICIENT ESTIMATOR MAIN PROGRAM

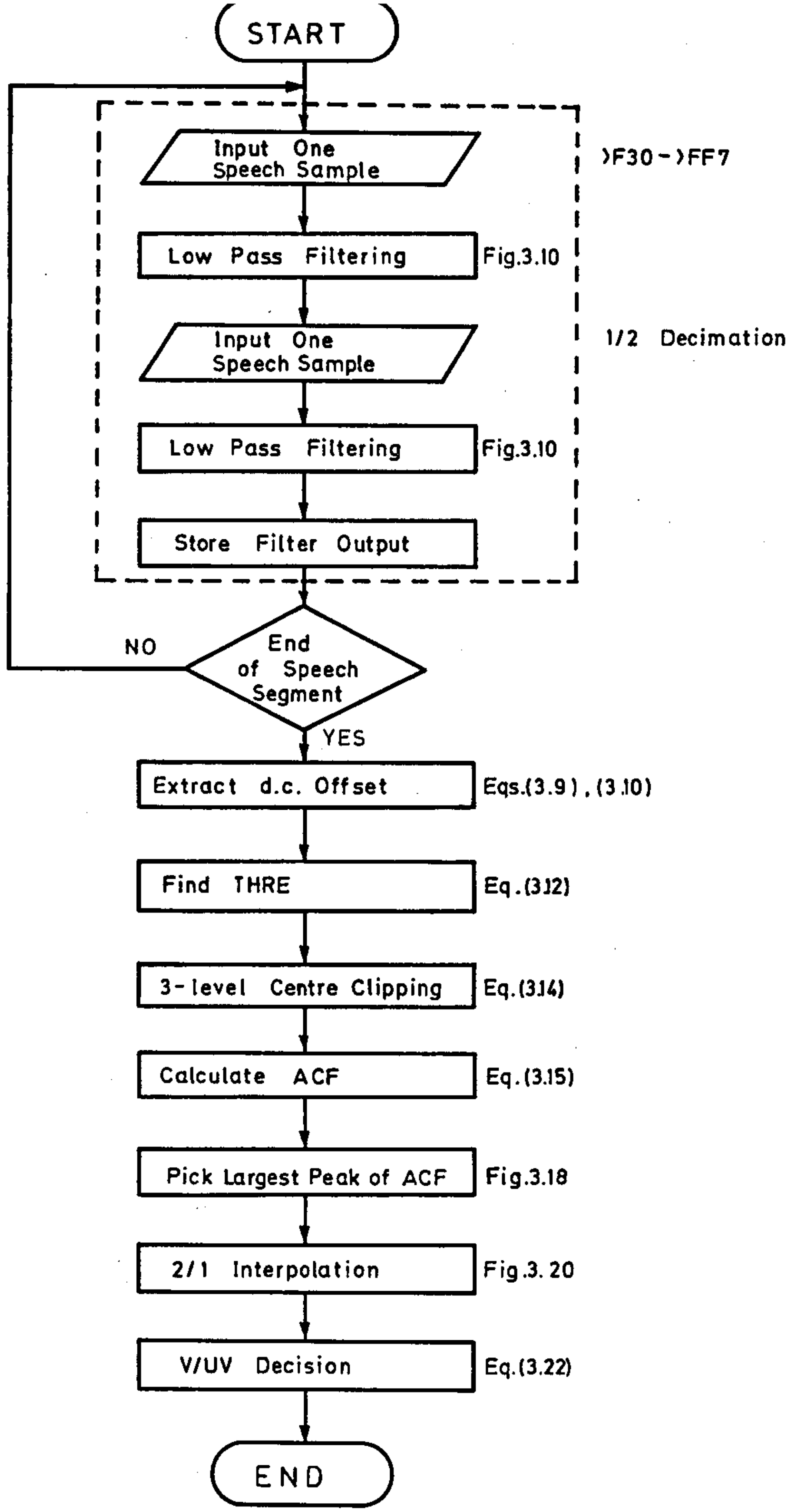


FIG.4.9 FLOWCHART OF THE PITCH DETECTOR MAIN PROGRAM

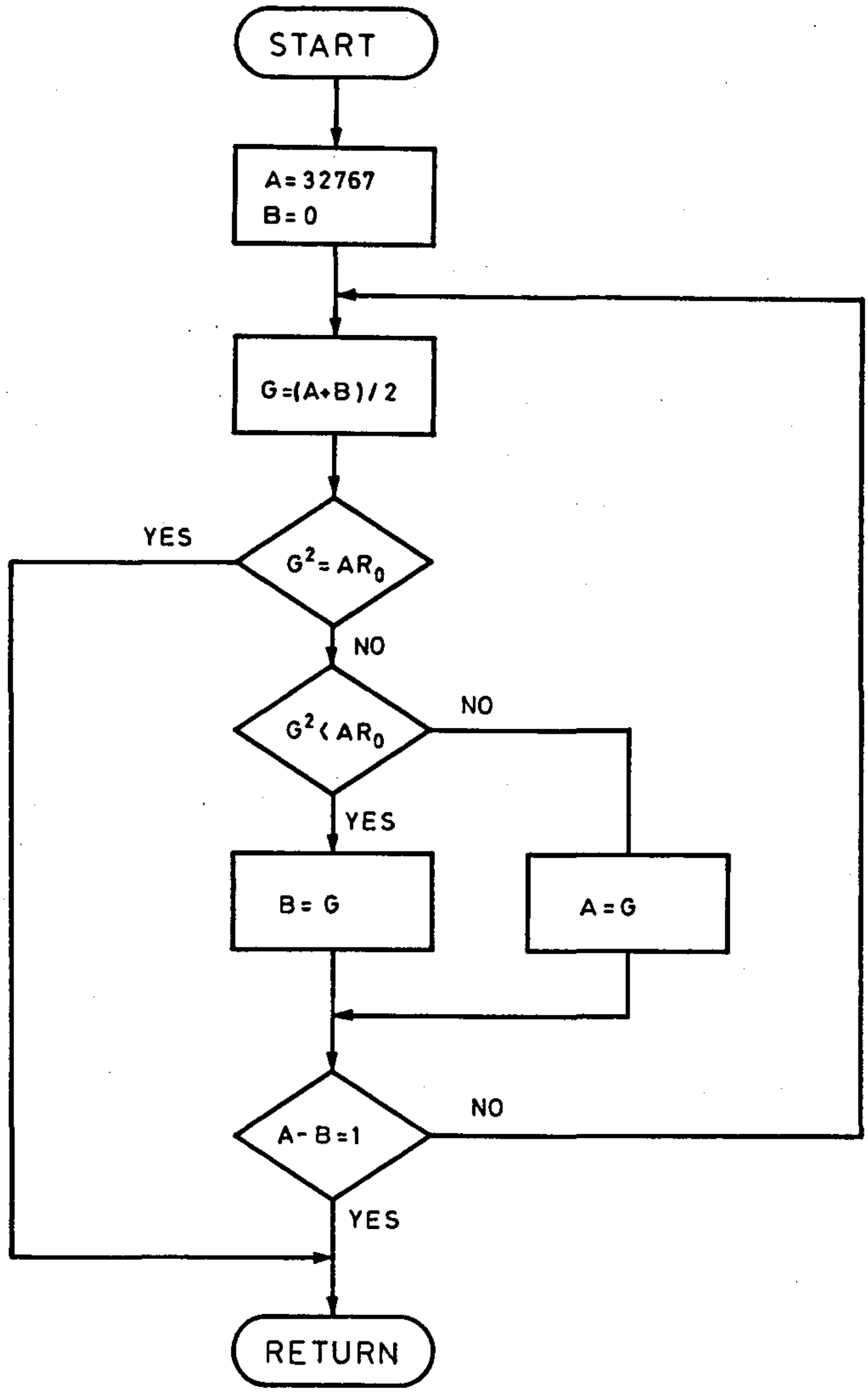


FIG.4.10 FLOWCHART OF THE GAIN ESTIMATOR SUBROUTINE

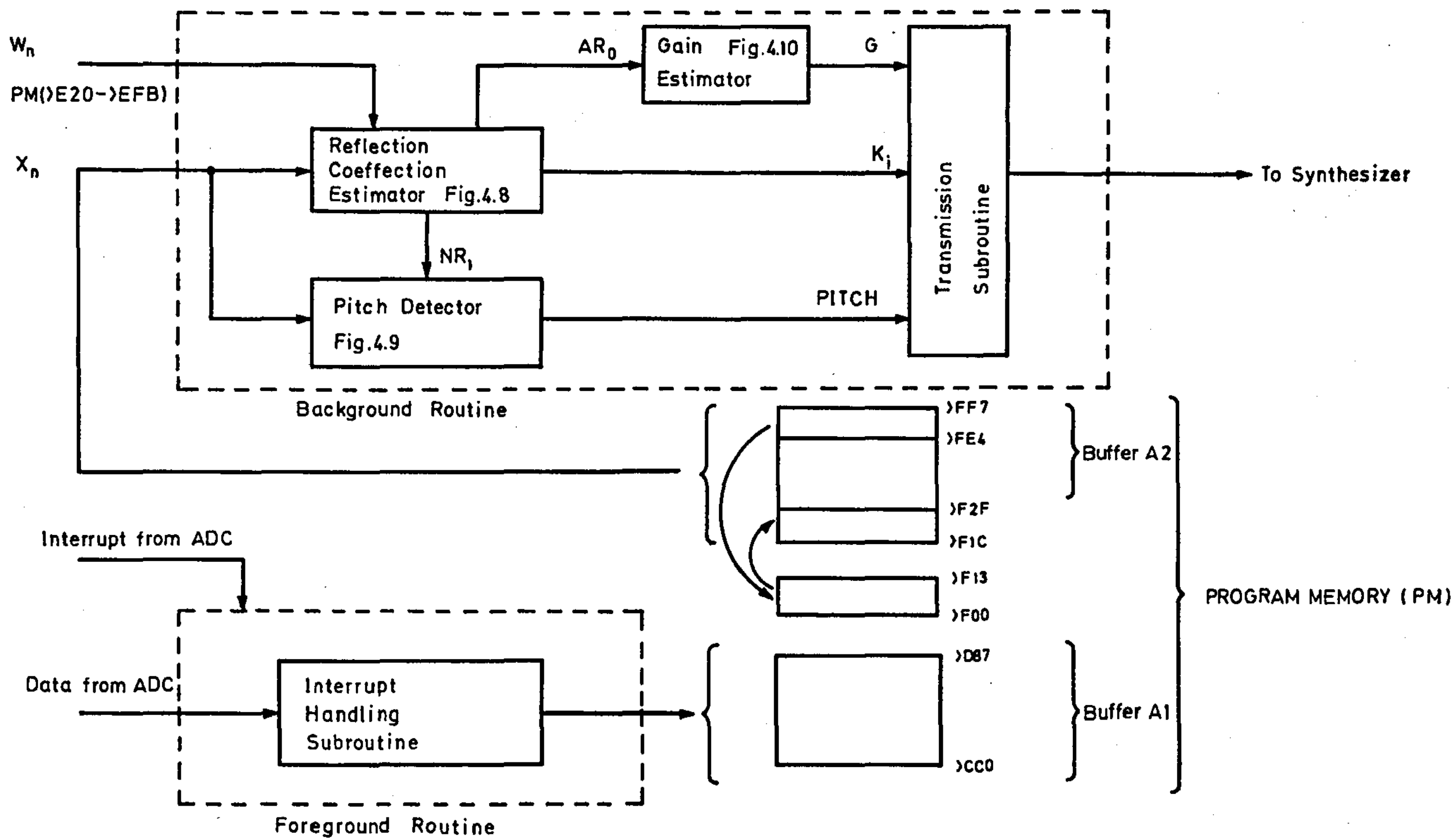


FIG.4.11 TMS32010 SOFTWARE STRUCTURE OF THE LPC ANALYSER

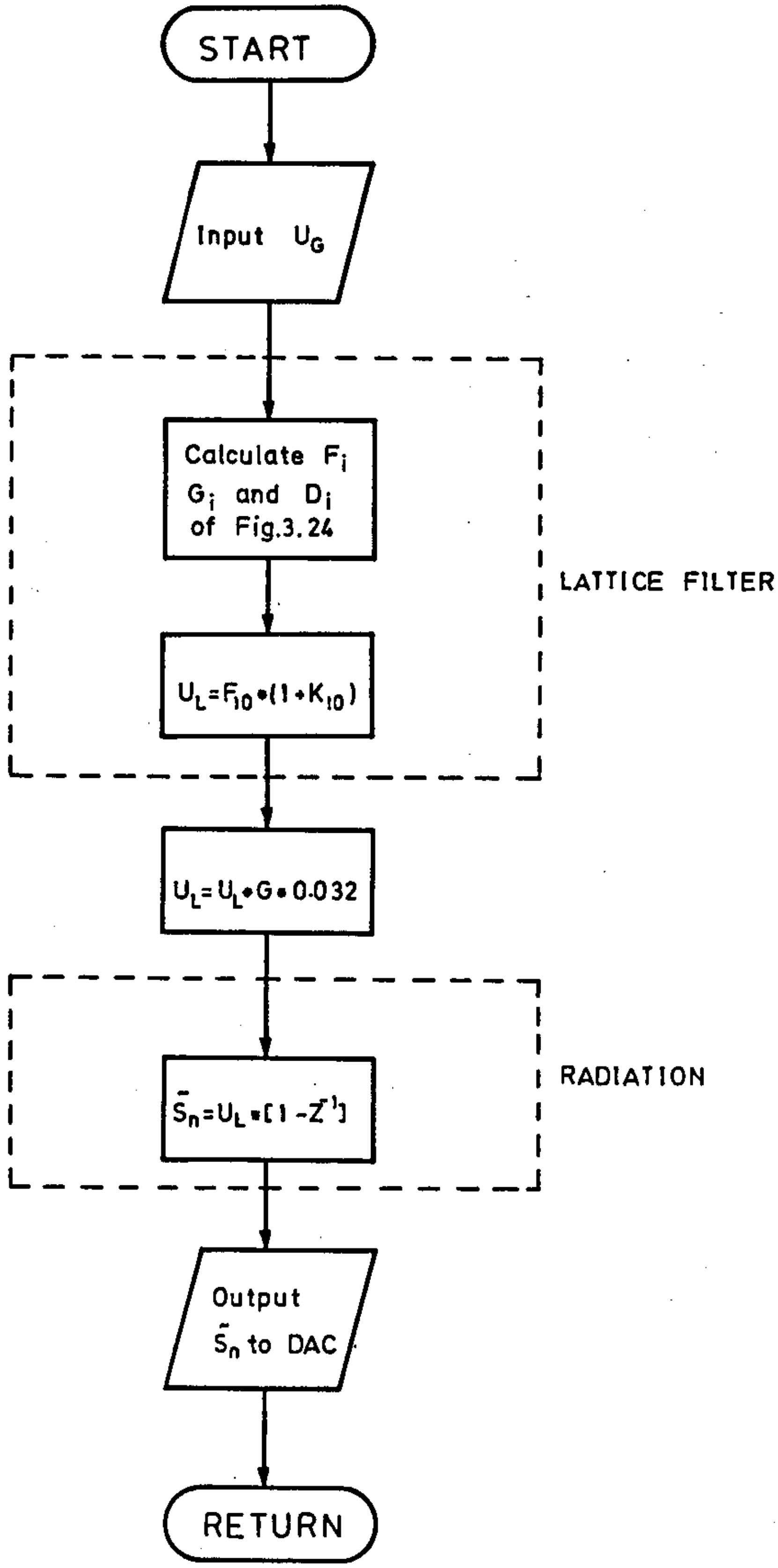


FIG.4.12 FLOWCHART OF THE LATTICE FILTER SUBROUTINE INCLUDING THE RADIATION NETWORK

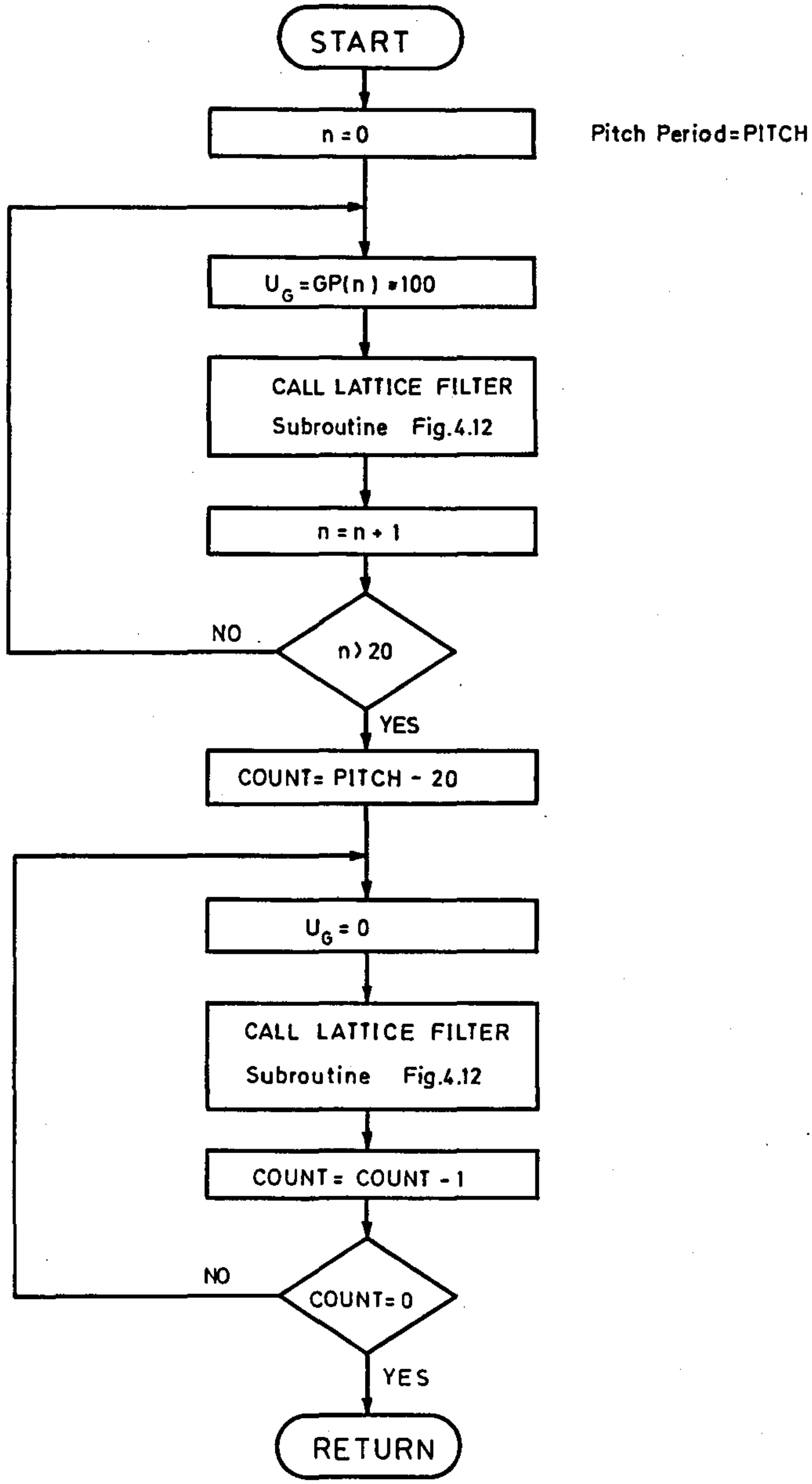


FIG.4.13 FLOWCHART OF THE VOICED EXCITATION LPC SYNTHESIS SUBROUTINE

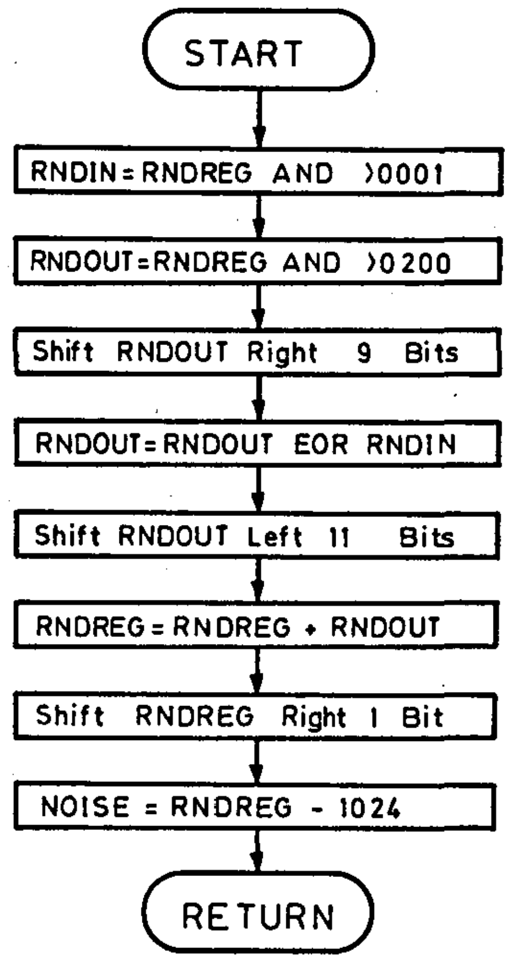


FIG. 4.14 FLOWCHART OF THE RANDOM NOISE GENERATOR SUBROUTINE

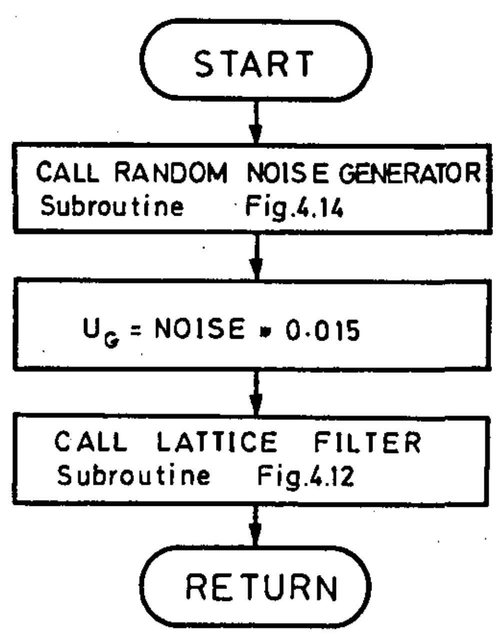


FIG. 4.15 FLOWCHART OF THE UNVOICED EXCITATION LPC SYNTHESIS SUBROUTINE

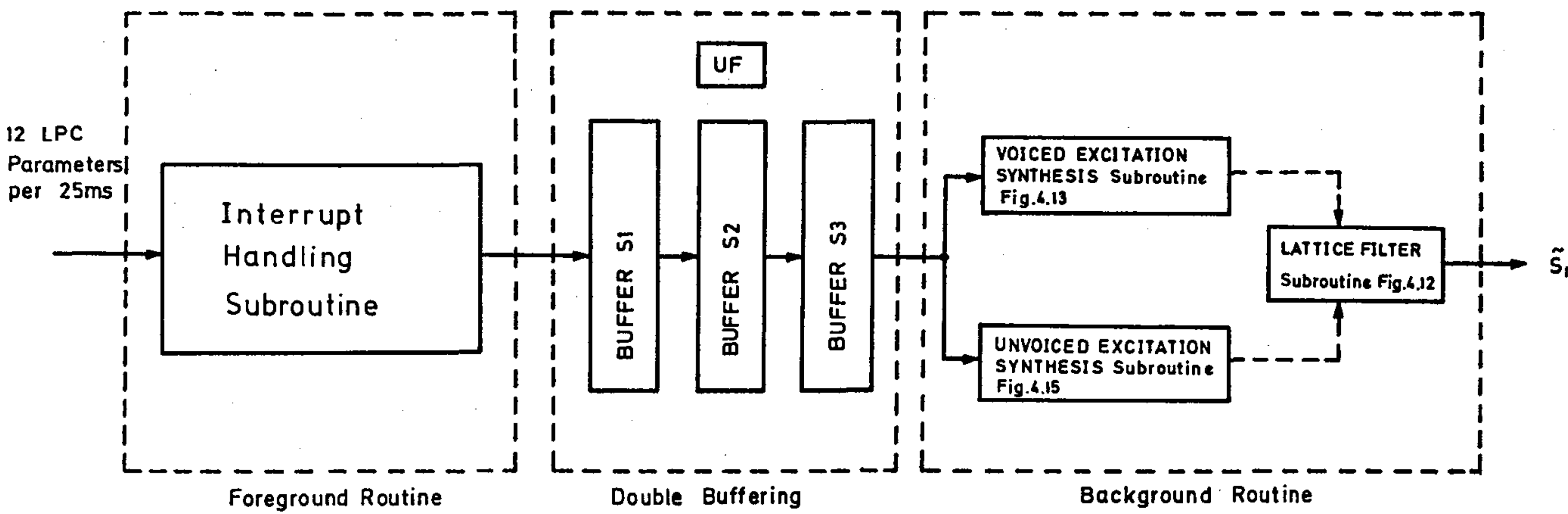


FIG.4.16 TMS32010 SOFTWARE STRUCTURE OF THE LPC SYNTHESIZER

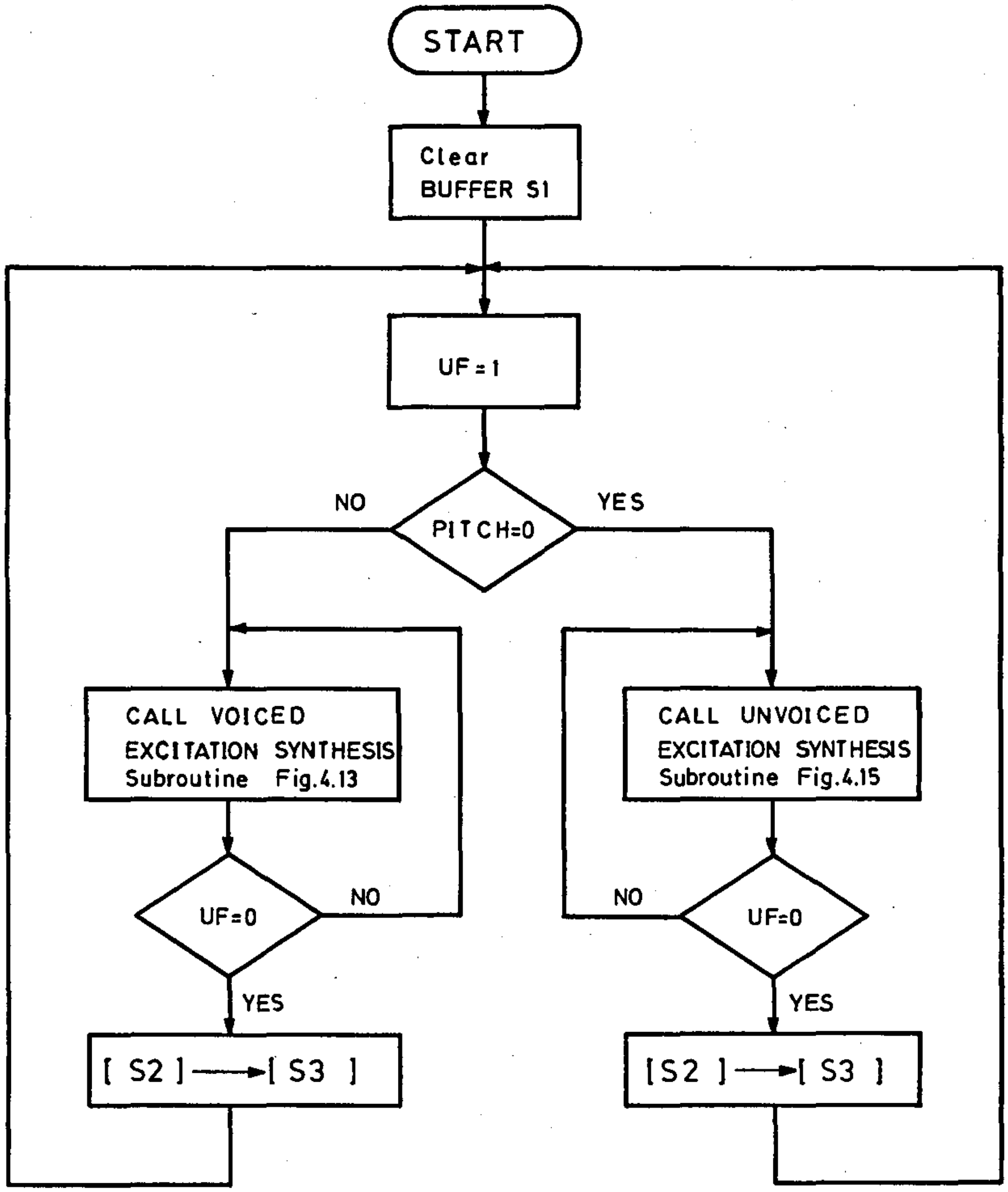


FIG.4.17 FLOWCHART OF THE SYNTHESIZER BACKGROUND ROUTINE

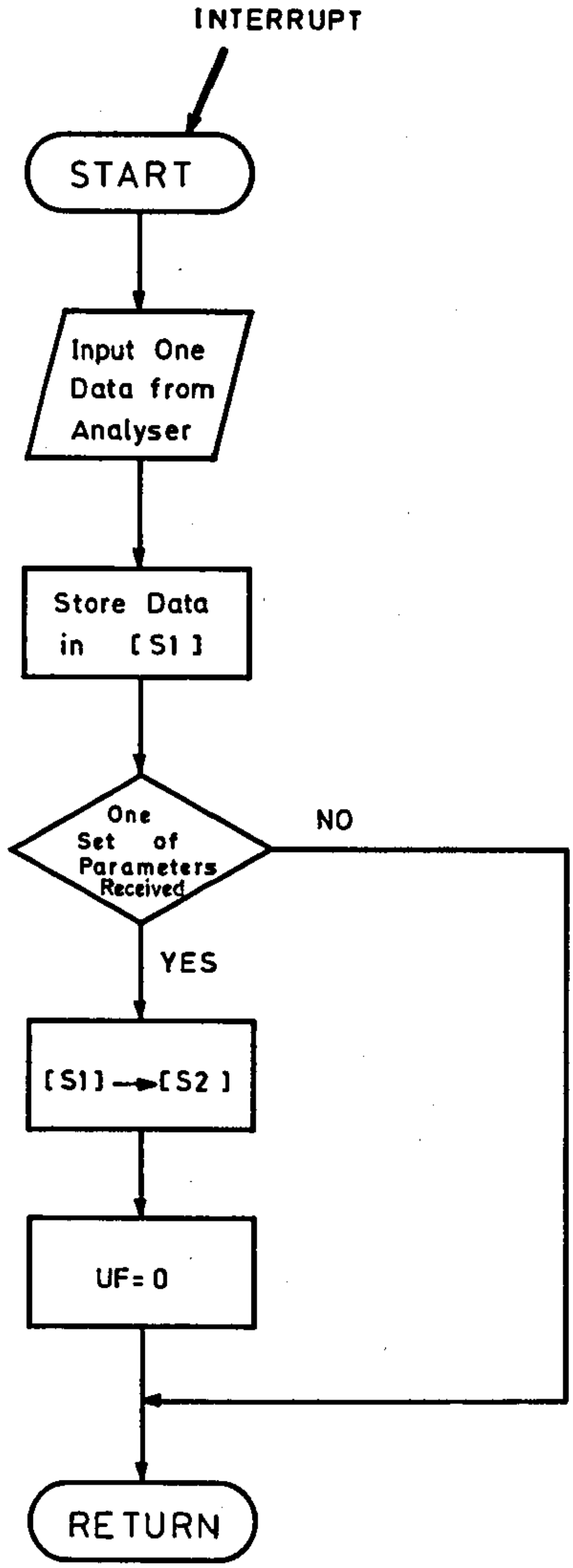


FIG.4.18 FLOWCHART OF THE SYNTHESIZER FOREGROUND ROUTINE

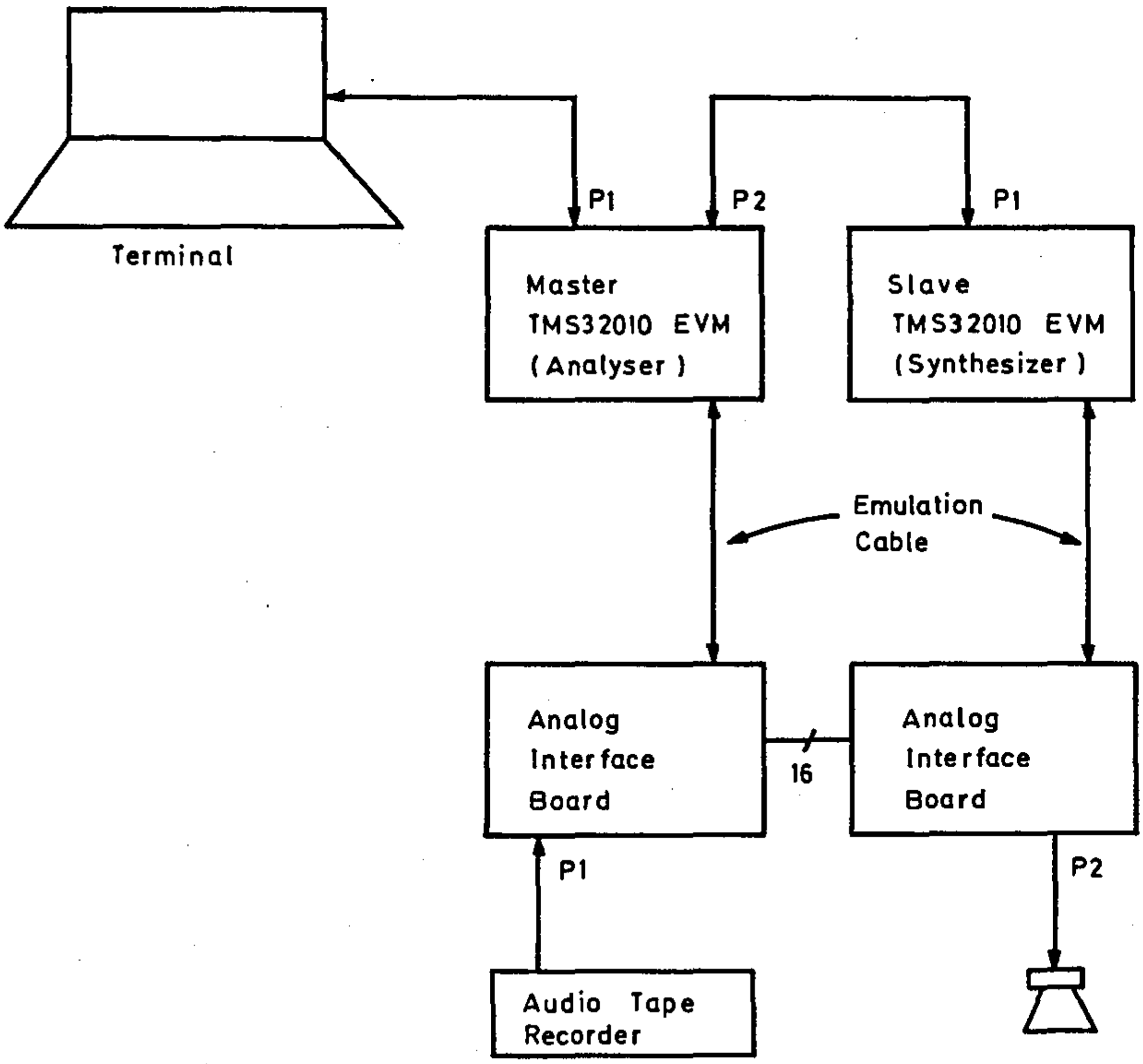


FIG.4.19 THE REAL-TIME IMPLEMENTATION EXPERIMENT CONFIGURATION

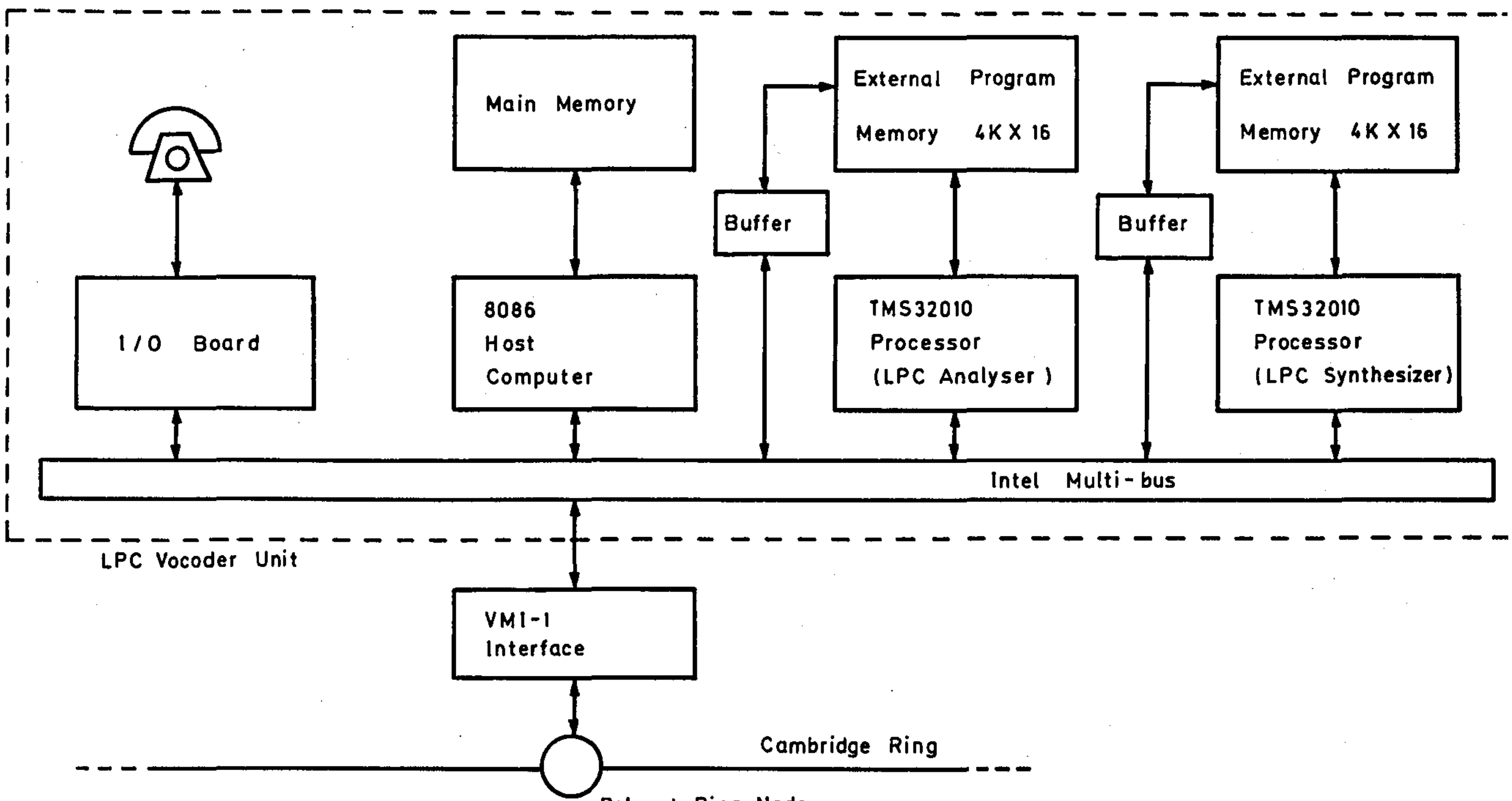


FIG.5.1 HARDWARE CONFIGURATION OF THE LPC VOICE CODING SERVER FOR A CAMBRIDGE RING