

Real Time Linear Predictive Coder of Speech

by

Kenneth Charles Walter Mikolajek

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Electrical Engineering

Winnipeg, Manitoba, 1981

(c) Kenneth Charles Walter Mikolajek, 1981

REAL TIME LINEAR PREDICTIVE CODER OF SPEECH

BY

KENNETH CHARLES WALTER MIKOLAJEK

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

MASTER OF SCIENCE

© 1981

Permission has been granted to the LIBRARY OF THE UNIVER-
SITY OF MANITOBA to lend or sell copies of this thesis, to
the NATIONAL LIBRARY OF CANADA to microfilm this
thesis and to lend or sell copies of the film, and UNIVERSITY
MICROFILMS to publish an abstract of this thesis.

The author reserves other publication rights, and neither the
thesis nor extensive extracts from it may be printed or other-
wise reproduced without the author's written permission.

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Ken Mikolajek

Kenneth Charles Walter Mikolajek

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Ken Mikolajek

Kenneth Charles Walter Mikolajek

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

A speech research facility based on bit-slice architecture has been developed and tested. The processor was successfully programmed to produce autocorrelation values, reflection coefficients, prediction coefficients, and the frame error of a speech signal in real time. The software is written on a program and a microprogram level.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation to the individuals who have contributed to this research. The author extends his appreciation to Dr. W. Kinsner for suggesting the topic of Linear Predictive Coding and for arranging financial support from the Natural Sciences and Engineering Research Council and the Industrial Applications of Microelectronics Center. Also, Dr. Kinsner offered guidance in directing the field of study which led to an understanding of the topic.

The numerous technical discussions about LPC and the speech signal with Luc Mahieu, Andreas Weirich, and Victor Shkawrytko were invaluable to completion of this work. I also would like express appreciation to Jim Reimer, Greg Smith, Jack Sill and Balraj Joll for many ideas which were implemented relating to hardware.

The author would also like to thank Debbie Strachan for the care taken in typing the manuscript and finally, Leona Kalra for her moral support.

CONTENTS

ABSTRACT	iv
ACKNOWLEDGEMENTS	v

Chapter

page

I.	INTRODUCTION	1
II.	LINEAR PREDICTION CODING THEORY OF SPEECH	2
	The Speech Signal	2
	Speech Production	2
	Excitation	4
	Vocal Tract Model	6
	Speech Perception	12
	Linear Predictive Coding Theory	18
	The LPC Model	18
	LPC Parameter Generation	20
	Autocorrelation Method	22
	Covariance Method	23
	Lattice Formulations	24
	Comparison of Methods	24
III.	ALGORITHM DESCRIPTION	26
	Windowing	26
	Autocorrelation Calculation	30
	Predictor Solution	33
	Computational Considerations	35
IV.	SYSTEM OVERVIEW	36
	The Bit Slice Approach	36
	System Hardware	37
	Components	37
	System Hardware Operation	39
	System Software Operation	40
V.	HARDWARE DESCRIPTION	41
	Preprocessor and A/D Converter	41
	Preprocessor	41
	Sampling	44
	Program Memory	47
	The LPCMK	49

Bit Slice Processor Operation	50
Pipeline Register	55
Microinstruction Sequencer	57
ALU and Register Set	60
Status	63
I/O Registers	64
Mapping Prom and Vector Register	66
External Bus	67
VI. SOFTWARE / FIRMWARE	68
Program Instructions	69
Program vs. Microprogram	69
Register Allocation	70
Instruction Description	73
Microprogram Software	79
Development	79
Instruction/Microprogram Interface	80
Data Acquisition	80
Major Microprogram Routines	83
Main Program	87
VII. RESULTS	89
Autocorrelation Values	91
Reflection Coefficients	92
Predictor Coefficients	93
Framing Error	93
VIII. CONCLUSIONS	94
IX. RECOMMENDATIONS	95
BIBLIOGRAPHY	97

Appendix

	<u>page</u>
A. LPCMK AND ASSOCIATED HARDWARE	99
B. LPCMK MICROPROGRAM LISTINGS	113
C. PROGRAM MEMORY TESTING LISTINGS	141
D. AMDAHL SIMULATION PROGRAM	145

LIST OF FIGURES

<u>Figure</u>	<u>page</u>
1. Speech Organs	3
2. Glottal Pulse Spectrum	3
3. Lossless Tube Model of the Vocal Tract	7
4. Two Tube model of the Vocal Tract	7
5. Hearing Organs	13
6. Subjective Perception	13
7. Schematic Illustration of the Cochlea Unrolled	16
8. Basilar Displacement	16
9. LPC Model	19
10. 200 Point Hamming Window	29
11. System Hardware Configuration	38
12. Preprocessor and A/D Block Diagram	43
13. Preprocessor Frequency Response	43
14. A/D Conversion Timing Diagram	45
15. Aperture Time	45
16. Interrupt Response Timing	46
17. Program Memory Block Diagram	48
18. A Typical Bit-Slice System Block Diagram	51
19. The LPCMK Block Diagram	54
20. Microinstruction Sequencer Block Diagram	58
21. Internal Bit-slice Block Diagram	61
22. I/O Timing Diagram	65

23.	LPCMK Memory Map	77
24.	Data Acquisition Microroutine Flowchart	82
25.	Main Program Flowchart	88

LIST OF TABLES

<u>Table</u>		<u>page</u>
1.	Allocation of Microinstruction Bits	56
2.	Rotate and Shift Destinations	63
3.	Condition Codes	64
4.	Microprogram vs. Program Tradeoffs	70
5.	LPCMK Internal Register Allocation	71
6.	FLAG Register Bit Assignment	72
7.	LPCMK Instructions	74
8.	LPCMK Real Time LPC Generation Program	87
9.	LPCMK Results	90

Chapter I

INTRODUCTION

The characteristics of the human speech signal have been vigorously studied since the 1950s. The results of work done in speech analysis have successfully been applied to telecommunications, speech synthesis, speech recognition, speaker recognition, and biomedical applications. In this work a speech research facility has been developed that enables the experimental study of linear predictive coding (LPC) of the speech signal.

The LPC data compression technique permits a reduction of the data rate of speech by a ratio of more than 30:1 [1]. This is discussed in detail in Chapter 2. This coding is performed in real time on a sampled speech signal with a specially designed bit slice processor. The algorithms chosen yield ten predictor and reflection coefficients, eleven autocorrelation values and the prediction error for every frame processed. The prediction coefficients are then used for specific purposes such as speech recognition, speech synthesis or sensitivity studies.

This document starts with a brief review of the speech signal and the LPC of speech. The pertinent algorithms are described, followed by the hardware and software description. Finally, the results of the LPC parameter generation are presented and evaluated.

Chapter II

LINEAR PREDICTION CODING THEORY OF SPEECH

2.1 THE SPEECH SIGNAL

To provide a comprehensive description of the speech signal, it is necessary to describe the production of speech and also to look at the human perception of sound. Knowledge of the physical limitations and qualities in the organs responsible for hearing and speaking yield valuable clues about modelling the system and also determine the useful information in the speech signal.

The descriptions are brief and are concerned with the organs on a physical level only.

2.1.1 Speech Production

The vocal organs are illustrated in Fig. 1. The lungs provide a relatively constant pressure which is 1% greater than the atmospheric pressure when a person is talking [2]. During conversation only 15% of the breathing time is typically devoted to inhaling. This contributes to the continuity of speech.

Air from the lungs passes through the larynx, where the vocal cords are located, and then out the mouth and nose. The opening created when the cords are open is known as the glottis. The portion of the vocal system between the glottis and lips, including the nose, is called the vocal tract.

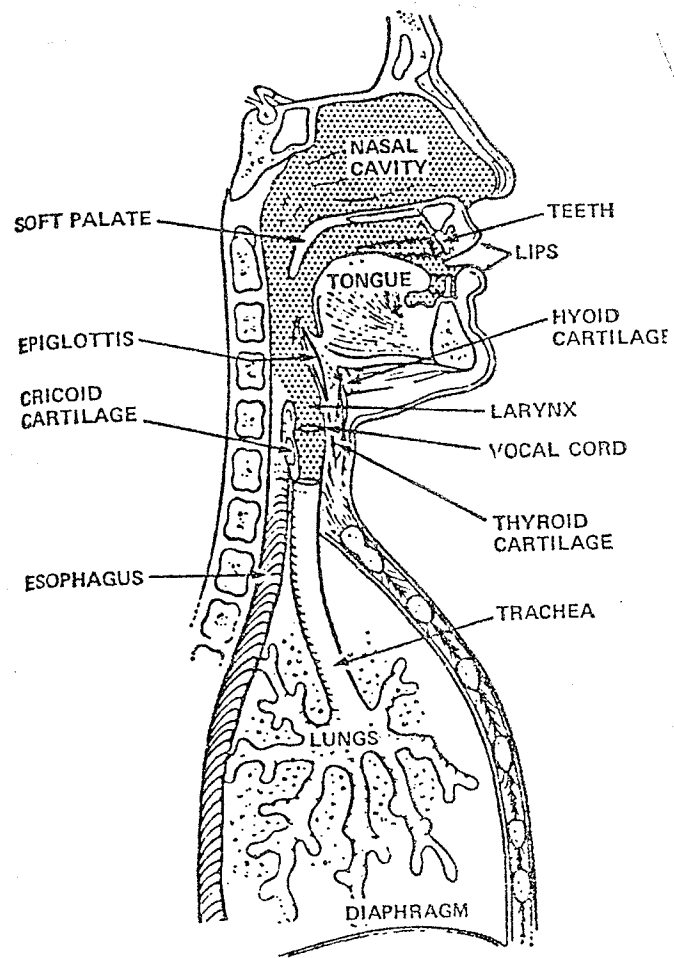


Figure 1: Speech Organs

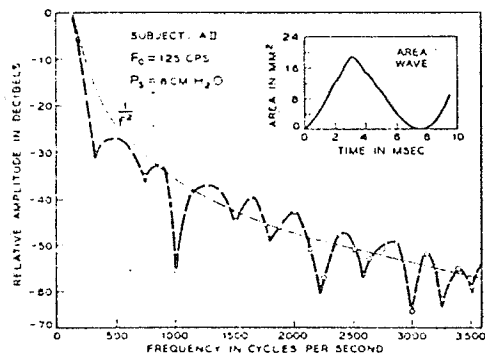


Figure 2: Glottal Pulse Spectrum from [3]

A detailed acoustic theory of the vocal tract must consider the following effects:

- a) Time variation of the local tract shape;
- b) Losses due to heat conduction and viscous friction at the vocal tract walls;
- c) Softness of the vocal tract walls;
- d) Radiation of sound at the lips;
- e) Nasal coupling; and
- f) Excitation of sound in the vocal tract.

The complete theory including all these effects is not yet available and is beyond the scope of this work. Simplifications will be made as required.

The vocal cords are responsible for modifying the air flow from the lungs before it enters the vocal tract. Here, the flow of air is modulated by articulators to produce speech. The vocal tract is approximately 17 cm long in the adult male and is deformed in cross-sectional area by the articulators; i.e., the tongue, lips, jaw, teeth and velum.

2.1.1.1 Excitation

The primary source of excitation to the vocal tract are the vocal cords. Three basic types of excitation are possible; voiced, unvoiced, and plosive.

Voiced excitation is the best understood since the physical source, (the vocal cords) can be more easily monitored and measured during excitation. Voiced excitation is produced by the vocal cords vibrating at various frequencies. Sufficient air pressure is built up when the glot-

tis is closed to force the cords open. The opened vocal cords form a venturi allowing air from the lungs to flow through the glottis and then the vocal tract. Bernoulli's Law states that when a fluid (air) flows through an orifice, in this case the glottis, the pressure is lower in the constriction than on either side. Consequently when the pressure is adjusted properly on the vocal cords, a sustained vibration is created. Thus the cords will open when there is sufficient pressure and will close completely once the air flow has begun.

A sustained air oscillation can be maintained. Its period depends on the following three factors:

- a) Air pressure in the lungs;
- b) Tension and stiffness of the vocal cords; and
- c) Area of the glottis under rest conditions.

Figure 2 shows a typical glottal pulse as a triangular wave, along with its associated frequency amplitude spectrum [3]. In reality the energy in the pulse is not symmetrically distributed. In normal speech, the periodic glottal pulse frequency, also known as pitch, extends from 60 to 350 Hz. Speech sounds associated with voiced excitations include vowels such as 'e' in beet, and voiced consonants such as 'r' in zebra.

One source for unvoiced excitation is a constriction in the vocal tract or turbulent air flow around a sharp edge, such as the teeth. The resulting speech sound is a hiss-like noise, called a fricative sound. Another type of unvoiced excitation is caused by aspiration or or free flow of the air from the lung through the vocal tract. In this case, all the articulators in the vocal tract are used to shape the sound. Whispering is a common example of this excitation source. Un-

voiced excitation is noted for having a lower energy and flatter frequency spectrum than voiced excitation. These characteristics are used to distinguish between the two types of excitations.

Although unvoiced excitation could occur anywhere in the vocal tract it is assumed that all excitation is applied at the vocal cords. Another simplification made for the sake of a reasonable speech model is that of a single excitation source. At any time, the excitation provided to the system is either voiced or unvoiced. In reality, however, there are numerous examples of mixed-source excitation models [4].

2.1.1.2 Vocal Tract Model

With the above assumptions, the vocal tract is seen as the system under excitation at the vocal cords. As mentioned earlier, the articulators of the vocal tract are used to adjust its cross-sectional area and hardness to provide the speech sound. The system is time-varying and is assumed as linear to aid in speech analysis. The articulators used to alter the system are limited by their inertia in their rate of change. The vocal tract is therefore assumed stationary for 10-30 msec.

Figure 3 shows a simple model of the vocal tract as a concatenation of lossless tubes of different diameters. This model has been investigated [5] by looking at the boundary conditions and junctions between tubes of differing diameters in terms of reflected waves. The k-th reflection coefficient (r_k) is given by

$$r_k = \frac{A_{k+1} - A_k}{A_{k+1} + A_k} \quad (1)$$

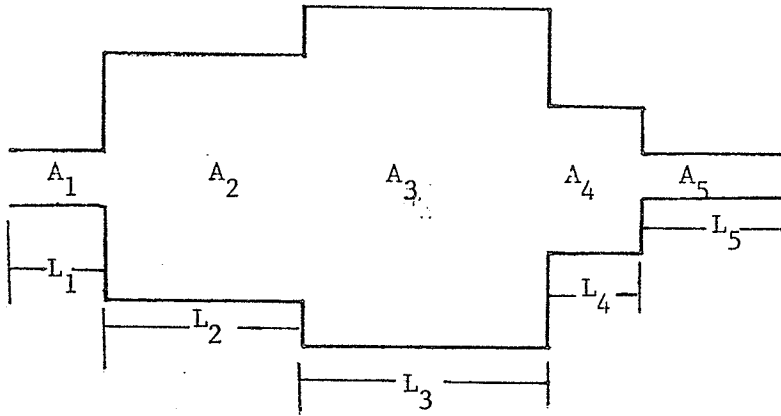


Figure 3: Lossless Tube Model of the Vocal Tract

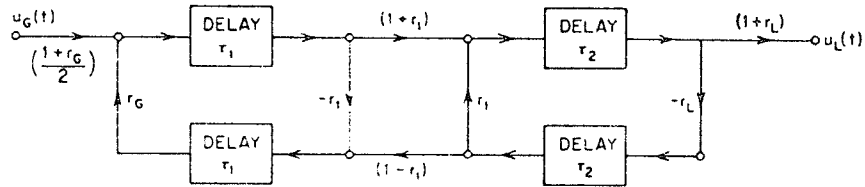


Figure 4: Two Tube Model of the Vocal Tract

where A_k is the cross-sectional area of the k-th tube. Since all $A_k > 0$, all the reflection coefficients are confined to the following range

$$-1 \leq r_k \leq 1 \quad (2)$$

At the acoustic frequencies under consideration the wavelengths of the sound waves are comparable to the length of the vocal tract. For the following analysis, only the frequencies between $f = 0-4000$ Hz are considered so that wave propagation can be assumed to occur in a lengthwise direction, along the vocal tract. The shortest sound wavelength (l_w) is given by

$$l_w = \frac{c}{f} = \frac{35000}{4000} = 8.75 \text{ cm.} \approx \frac{1}{2} l_v \quad (3)$$

where l_v is the vocal tract length and c , the speed of sound is given as 35000 cm/sec.

A complete 2 tube model is shown in Fig. 4 with the frequency response $V_a(\Omega)$ given by [5]

$$V_a(\Omega) = \frac{V_L(\Omega)}{V_G(\Omega)}$$

$$= \frac{.5(1+r_G)(1+r_L)\{1+r_1 e^{-j\Omega(T_1+T_2)}\}}{1+r_1 r_G e^{-j\Omega 2T_1} + r_1 r_2 e^{-j\Omega 2T_2} + r_2 r_G e^{-j\Omega(T_1+T_2)}} \quad (4)$$

Where V_L and V_G are the transforms of the volume velocity at the lips and the glottis, respectively, R_L and R_G are the reflection coefficients at the lips and glottis, respectively, and T_1 and T_2 are the times required for sound to travel through sections 1 and 2.

In the signal flow graph model to be considered all sections are assumed to be of equal length. This is valid if an adequate number of sections are used to represent the vocal tract. Thus the propagation delays in each section T are the same and the system can be modelled with an infinite impulse response (IIR) digital filter.

The lossless tube model is the same as that of Fig. 4 except that the delays T_1 and T_2 are the same T' . When an impulse is supplied at the glottis, the volume velocity at the lips ($v_a(t)$) is given by

$$v_a(t) = a_0 \delta(t - NT') + \sum_{k=1}^{\infty} a_k \delta(t - NT' - 2kT') \quad (5)$$

The soonest an impulse can reach the output is NT' , where T' is the propagation time for one section and N is the number of sections. The successive reflected waves will reach the output 'ad infinitum' $2T'$ apart. The frequency response is

$$\begin{aligned} V_a(s) &= \sum_{k=0}^{\infty} a_k e^{-s(N+2k)T'} \\ &= e^{-sNT'} \sum_{k=0}^{\infty} a_k e^{-s2T'k} \end{aligned} \quad (6)$$

The factor

$$V_a'(s) = \sum_{k=0}^{\infty} a_k e^{-s2T'k} \quad (7)$$

represents the resonance properties of the system. The frequency response is obtained by replacing s with $j\Omega$ to get

$$V_a'(\Omega) = \sum_{k=0}^{\infty} a_k e^{j\Omega k 2T'} \quad (8)$$

with

$$V_a'(\Omega + \frac{2\pi}{2T'}) = V_a'(\Omega) \quad (9)$$

The response is periodic and, by Shannon's theorem, aliasing is avoided when sampling is done with period $2T'$ and all excitation is less than $1/2T'$ Hz.

The sampling period of $2T'$ makes sense intuitively since a change in the output volume velocity should only occur after a wave has undergone a backwards and then a forward reflection. This corresponds to a time of $2T'$.

All information regarding the system is contained in the reflection coefficients r_k of the individual boundaries. The boundary condition at the lips ($r_L=0$) and glottis ($r_G=1$) are assumed the same as for the previous model of Fig. 4.

The choice of number of sections depends on the length of the vocal tract and the sampling rate chosen. Since the frequency response of the lossless tube is periodic, only a band of frequencies can be approximated in the model. These frequencies are given by

$$|F| < \frac{1}{(2T)} \text{ T-Sampling period} \quad (10)$$

This requires $T=2T'$ where T' is the one-way propagation time of a single section. With N sections giving total length l , $T'=l/cN$, where c is the speed of sound.

The denominator representing the transfer function is similar to that in Eq. (4) except $T_1 = T_2$. The order of this denominator is N with $N/2$ complex conjugate poles necessary to provide the resonances in the above mentioned frequency band.

The following relationship describes the highest frequency component from the vocal tract

$$\frac{1}{2T} = \frac{1}{4T'} = \frac{Nc}{4L} = \frac{N}{2}(1000 \text{ Hz}) \quad (11)$$

This implies that there is one resonance for each 1000 Hz in a vocal tract of length 17.5 cm. A resonance of the vocal tract is known as a formant. Shorter vocal tract lengths have wider formant spacing.

In summary, the vocal tract is modelled as a linear, quasi-stationary system. The excitation is assumed to be provided at the vocal cords and the response of the vocal tract results in speech sounds. The existence of formants are successfully predicted by the theory of lossless acoustic tubes and are an important feature of the speech signal.

2.1.2 Speech Perception

Some understanding of the human perception of sound in general and speech in particular is important for the successful analysis of speech. Only the relevant information of the speech signal need be retained in the model. This refers directly to that which can be perceived by the ear. With better understanding of the perceptual process, speech research effort, including analysis, can be optimized. Because of the complexity of the topic, this discussion of speech perception is necessarily limited to the physical function of the hearing organs, not including the auditory nerve and detailed interpretation of sound. The hearing organs are organized into three groups depending on their location in the head. These are the outer ear, the middle ear and the inner ear. A schematic diagram of the ear is show in Fig. 5.

The outer ear is that portion outside the head to the point where the ear canal meets the eardrum. In man the ear canal is about 2.7 cm in length and is an acoustic resonator that amplifies sound waves near its resonance frequencies, usually 3-4 kHz.

The effect of the ear canal on perception can be seen clearly in Fig. 6. The threshold of hearing shows a global minimum at about 3 kHz when subjective loudness is plotted against frequency. Zero dB is defined as the threshold of hearing for a sinusoid of 1000 Hz. At 3-4 kHz frequencies the sound pressure at the eardrum is two to four times greater than the sound pressure entering the ear canal.

The middle ear consists of the malleus (hammer), incus (anvil) and stapes (stirrup), also called the auditory ossicles. These are three small bones that mechanically link the ear drum to the inner ear. The two major functions performed by the middle ear are:

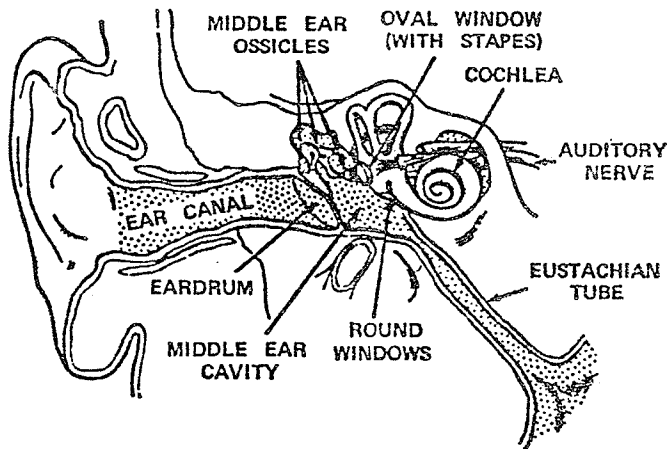


Figure 5: Hearing Organs

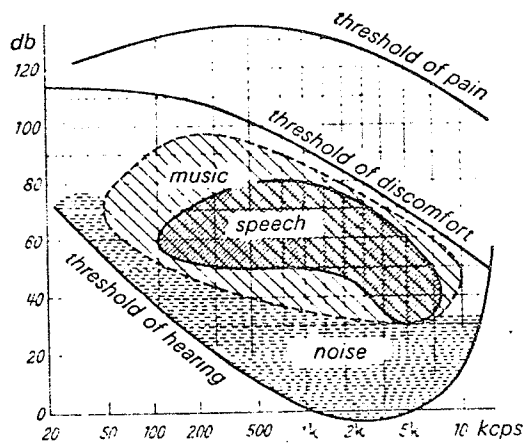


Figure 6: Subjective Perception

- a) Impedance transformation from the outer ear air medium to the inner ear's fluid medium; and
- b) Protection for the inner ear against sudden loud sounds.

The air-filled cavity of the middle ear is connected to the mouth cavity by the eustachian tube, which is normally closed. The pressure difference that can build up between the middle ear and surrounding air is particularly noticeable if the outside air pressure changes rapidly. Swallowing will momentarily open the tube to equalize the pressure.

The auditory ossicles are suspended by ligaments in the cavity walls. The hammer is rigidly attached on one side and covers more than one-half the eardrum area. It transmits the energy of eardrum motion to the anvil which causes a displacement in the stirrup. The total force on the stirrup acts only on the oval window, which interfaces the middle ear to the inner ear. The middle ear system acts as a lever that produces pressure on the oval window about 35 times greater than it would be otherwise.

The protective function of the middle ear prevents damage by loud noises to the inner ear, and also performs some initial processing on the signal. One mechanism is provided by two muscles reacting together in a reflex response to loud sounds. The middle of the eardrum is connected by one of the so-called tympanic muscles to the inner region of the head while another is connected to the stirrup. In response to loud sounds the eardrum is pulled in while the stirrup is pulled away from the oval window slightly. Thus the efficiency of the middle ear is reduced for louder sounds.

Contraction of the middle ear muscles increases with sound intensity so the ossicles stay in contact with each other, even at high levels. This controls distortion that might otherwise result.

The inner ear consists of the cochlea, vestibular apparatus (not normally used for detecting audio vibrations) and the auditory nerve termination. The cochlea is normally coiled like a snail shell in a flat spiral of 2 1/2 turns. The cochlea is schematically shown in Fig. 7 as if it were stretched out. The chamber is filled with a colorless liquid and the length of the canal in the spiral conch is about 35 mm. The cross-sectional area at the stirrup end is about 4 mm^2 and the area decreases to about 1 mm^2 at the tip. The partition in the center is filled with a different liquid and is bounded by a bony shelf, a gelatinous membrane called the Basilar membrane and another membrane known as Peissner's membrane.

The inner ear is connected to the stirrup at the oval window. When vibrating, the stirrup acts as a piston producing a volume displacement of the cochlea fluid. Since the cochlea is essentially rigid and the fluid is incompressible, the round window serves to absorb the fluid displacement. Among the cells residing in the Basilar membrane partition are approximately 3000 sensory (hair) cells on which the auditory nerve endings terminate. Since the Basilar membrane is stiffer and less massive at its narrow basal end and more compliant and more massive at the broad end, its resonant properties vary continuously along its length.

Very slow vibrations of the stapes (less than 20 Hz) result in a back-and-forth motion of the fluid of the scala vestibuli and the scala

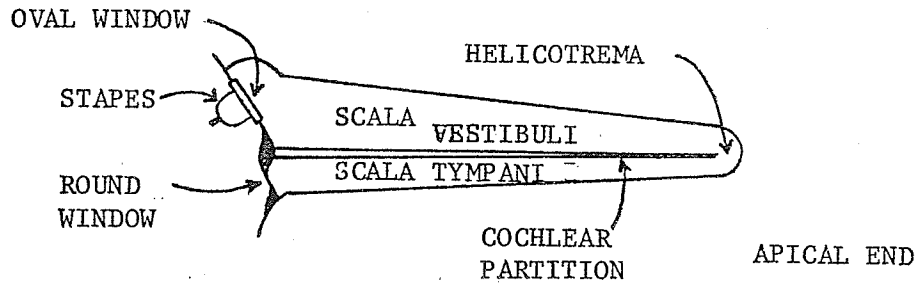


Figure 7: Schematic Illustration of the Cochlea Unrolled

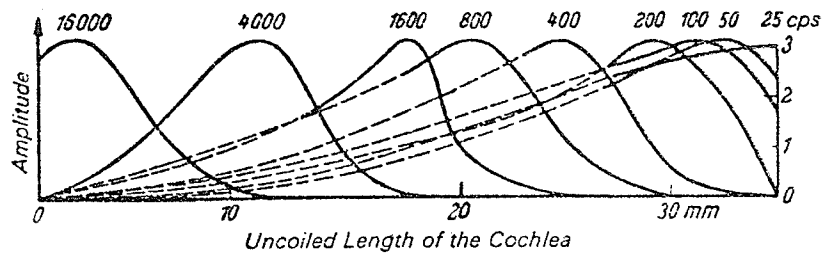


Figure 8: Basilar Displacement

tympani through the opening at the helicotrema. At higher frequencies, vibrations are transmitted through the yielding cochlea at points which depend on the frequency content of the signal.

The relative displacements of the Basilar membrane, normalized to one, are shown graphically in Fig. 8 for different sinusoidal frequencies. It should be noted that the displacements for the Basilar membrane at the different frequencies have roughly the same Q. This explains why frequency differentiation is more selective for lower frequencies and time resolution is better at higher frequencies. More recent measurements [3] suggest that the mechanical response is somewhat dependant on sound intensity.

The range of frequencies perceived by the ear are depicted in Fig. 6. Typically frequency perception extends from 50-16,000 Hz. It is interesting to note that the lowest frequency heard has a period very close to the pseudostationary period of the vocal tract, 20 msec. Events that happen slower than 20 msec apart appear to the listener as time-separated while those which are quicker are distinguished in the frequency domain.

When white noise is applied to a subject's ear the uncertainty of frequency differentiation is greater than that for sinusoids. The maximum information content in terms of frequency and loudness is found between 700 and 5,000 Hz and does not increase with loudness as it does for sinusoids. Perception tests done with noise are more applicable to speech because a pure sine wave is almost never encountered in speech.

The basic assumptions of the speech model which have been justified from the point of view of both the ear and vocal organs are:

- a) Pseudostationary for 20 msec; and
- b) Bandlimited to $1/2T < 4$ kHz.

2.2 LINEAR PREDICTIVE CODING THEORY

Linear Predictive Coding is widely used as an efficient means of representing a complex speech signal by a small number of parameters. Typically the reduction in data using LPC is about 30:1. A typical LPC vocoder [6] reduces the data rate necessary for speech transmission to 2400 BPS (Bit Per Second). This is a source coding method where only the parameters for the model of the vocal tract are calculated. The excitation, voiced or unvoiced, and the gain must be calculated separately for the reconstruction of speech from the parameters.

2.2.1 The LPC Model

The justification for the use of LPC is that a linear combination of past samples of the speech signal can be used to predict the next value. In the most general case for linear prediction, the past and present inputs are also used to predict the next value. This is a time domain method which depends on the pseudostationary linear model of the vocal tract shown in Fig. 9.

The all-pole model which is used is also known as the autoregressive model. It can be justified on the basis of perceptual effects of removing the zeros or anti-resonances from the speech signal.

With non-nasal sounds the vocal tract transfer function has no zeros. For these sounds, therefore, zeros are not necessary. With unvoiced and nasal sounds, the zeros lie within the unit circle in the z-plane.

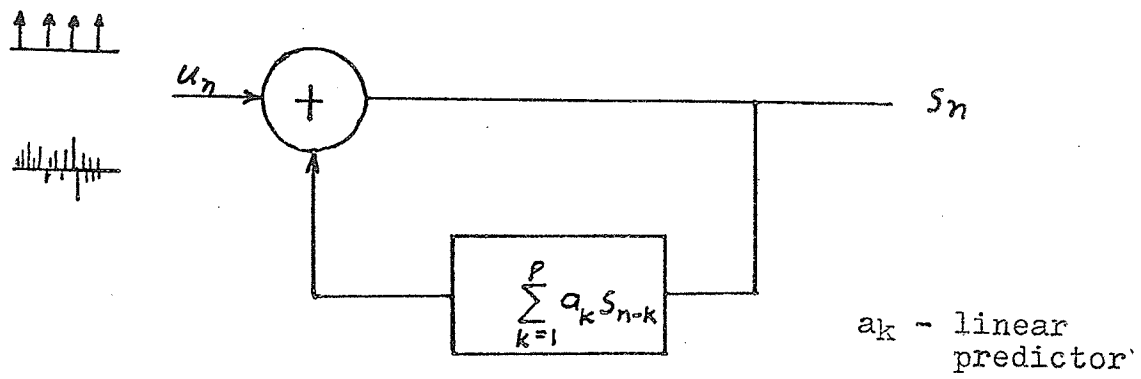


Figure 9: LPC Model

Thus, they can be approximated with arbitrary accuracy by multiple poles in the transfer function denominator [5]. In most cases the zeros contribute only to spectral balance by affecting the formant bandwidth or energy distribution. The formant locations are determined solely by the poles.

It was noted in Sec. 2.1.2 that one resonance will exist per 1000 Hz in a vocal tract of length 17.5 cm. As a rough measurement, it is sufficient to obtain two poles to represent the influence of the glottal flow and radiation of the speech wave and two poles for each resonance. The effect of zeros is not normally considered here. Therefore, if the signal is bandlimited to 4 kHz, approximately 10 predictor coefficients are needed.

2.2.2 LPC Parameter Generation

The signal s_n can be described at any time as a linear combination of P past values and the present input u_n for the all-pole model. This is shown by

$$s_n = -\sum_{k=1}^P a_k s_{n-k} + G u_n \quad (12)$$

where a_k are the predictor coefficients and G is the gain. The system transfer function reduces to

$$H(z) = \frac{G}{1 + \sum_{k=1}^P a_k z^{-k}} \quad (13)$$

which can be realized by a digital infinite impulse response (IIR) filter.

The approximated signal s' is obtained by a linearly weighted sum of the past values and is shown as follows

$$s'_n = -\sum_{k=1}^P a_k s_{n-k} \quad (14)$$

The resulting error e_n known as the residual is described by

$$e_n = s_n - s'_n = s_n + \sum_{k=1}^P a_k s_{n-k} \quad (15)$$

The residual must be minimized to get the LPC coefficients.

A deterministic instead of a random signal is assumed, allowing results to be expressed in terms of actual rather than expected values. The method of least squares is almost universally used for minimizing the error. This method is expressed as

$$E = \sum_n e_n^2 = \sum_n \left(s_n + \sum_{k=1}^P a_k s_{n-k} \right)^2 \quad (16)$$

$$\frac{\partial E}{\partial a_i} = 0 \quad 1 \leq i \leq P \quad (17)$$

where E is known as the error signal energy. The solution yields the normal equations shown below, in which the range of summation has not yet been defined.

$$\sum_{k=1}^P a_k \sum_n s_{n-k} s_{n-i} = -\sum_n s_n s_{n-i} \quad 1 \leq i \leq P \quad (18)$$

$$E_P = \sum_n s_n^2 + \sum_{k=1}^P (a_k \sum_n s_n s_{n-k}) \quad (19)$$

For any set of signals s_n , there will be P equations in P unknowns that can be solved for the predictor coefficients (a_k). The range of summation is specified for two cases, which yield two distinct methods for estimating the parameters.

2.2.2.1 Autocorrelation Method

With this method the error is minimized over all time, $-\infty < n < \infty$.

The solution for the normal equation is

$$\sum_{k=1}^P a_k R(i-k) = -R(i) \quad 1 \leq i \leq P \quad (20)$$

$$E_P = R(0) + \sum_{k=1}^P a_k R(k) \quad (21)$$

where

$$R(i) = \sum_{n=-\infty}^{\infty} s_n s_{n+i} \quad (22)$$

is the autocorrelation function of s_n . We can see that $R(i) = R(-i)$, is an even function of i .

The autocorrelation matrix formed by the $R(i-k)$ in Eq. (20) is a symmetric Toeplitz matrix. All terms along each diagonal are the same. This is important when considering solutions for the predictor coefficients.

Since the signal is generally known only over a finite interval, 0 to $N-1$, a window function multiplies the signal so it is zero outside the known interval. This can be expressed by

$$s'_n = \begin{cases} s_n w_n & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

where s'_n is the windowed signal and w_n is the window function. The autocorrelation then becomes

$$R(i) = \sum_{n=0}^{N-1-i} s'_n s'_{n+i} \quad i \geq 0 \quad (24)$$

2.2.2.2 Covariance Method

With this method error is minimized over the finite interval $0 \leq n \leq N-1$, giving the normal equations

$$\sum_{k=1}^P a_k \zeta_{ki} = \zeta_{0i} \quad 1 \leq i \leq P \quad (25)$$

$$E_p = \zeta_{00} + \sum_{k=1}^P a_k \zeta_{0k} \quad (26)$$

where

$$\zeta_{ik} = \sum_{n=0}^{N-1} s_{n-i} s_{n-k} \quad (27)$$

which is the covariance of the signal s_n over the given interval. Note that the P samples preceding the frame under consideration are necessary to compute the covariance. The ζ_{ki} in Eq. (25) form the covariance matrix which is symmetric, but not Toeplitz.

The covariance method reduces to the autocorrelation method as N goes to infinity. Also note that windowing is not necessary over the frame since error is minimized only over the interval considered.

2.2.2.3 Lattice Formulations

This class of methods combine the autocorrelation and covariance methods. With these methods the predictor coefficients are obtained directly from the speech sample. There is no need for an intermediate correlation matrix to be calculated. Also, stability of the filter yielded is guaranteed without use of a window.

2.2.3 Comparison of Methods

The number of multiplications required for computation of the autocorrelation and covariance methods are similar if $N \gg P$ where N is the number of samples considered in the frame and P is the number of predictor coefficients. A modified lattice method exists that obtains the partial correlation coefficients, also called PARCOR or reflection coefficients, with the same computational efficiency as the covariance method [5]. However, this method requires more calculation to obtain the parameters. Therefore, to obtain the predictor parameters it is less efficient.

Stability is also an important criterion in the choice of a method. The LPC model can be described in the z -plane as

$$\text{where } H(z) = \frac{G}{A(z)} \quad (28)$$

$$A(z) = 1 - \sum_{k=1}^P a_k z^{-k} \quad (29)$$

This system is stable if the zeros of $A(z)$ are inside the unit circle in the z -plane.

The parameters obtained by the covariance method could result in an unstable filter; i.e. stability is not guaranteed by this method. The

poles, which are located outside the unit circle for an unstable filter, could be reflected inside to obtain a stable filter with the same frequency response. This approach, however, results in increased computation time.

For the lattice method the filter obtained by the resulting predictor polynomial is guaranteed to be stable since the predictor coefficients are obtained from the PARCOR coefficients which, by definition, are stable.

The autocorrelation method was chosen because of its efficient calculation and the fact that the resulting filter is theoretically guaranteed to be stable. Care must be taken in the calculation since finite register length effects might cause an unstable filter. When the speech signal is spectrally flattened these undesirable effects are minimized. Therefore, with spectral pre-emphasis, a stable filter will result with smaller register lengths. This is discussed in more detail in Chapter 5.

The test of the PARCOR coefficients to ascertain the following relation

$$-1 \leq k_i \leq 1 \tag{30}$$

is necessary and sufficient to ensure stability [5]. It will be shown that PARCOR coefficients are a byproduct of the recursive relation, known as Durbin's algorithm, used to calculate the LPC coefficients.

Chapter III

ALGORITHM DESCRIPTION

Meeting the objective of using the autocorrelation method in real time requires that all calculations be done within the time period of a single frame. Algorithm selection is based on efficiency of calculation and the information provided at completion. The following three main transformations are performed on the data, as required by the autocorrelation method:

- a) Windowing;
- b) Autocorrelation matrix calculation; and
- c) Parameter generation.

These are discussed in some detail below, with justification given for the method used.

3.1 WINDOWING

It was shown in Sec. 2.2.2.1 that since the signal is known only for samples 0 to N-1, windowing in the form of Eq. (23) is necessary. The simplest window considered is the rectangular window, given by

$$W_n = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (31)$$

Use of this window to obtain predictor coefficients results in a very large prediction error at the start of and immediately following the frame. This can be seen by looking at Eq. (14) repeated below.

$$s'_n = \sum_{k=1}^P a_k s'_{n-k} \quad -\infty < n < \infty \quad (14)$$

For $n=0$ the s'_n are predicted using the past P values which have been set to zero by w_n . The estimated value is zero, but it is very unlikely that the actual value will be zero, resulting in a large prediction error. After the first P values are predicted the error will generally become smaller since the actual signal values are used to predict the next value.

A similar argument is used for prediction of samples past $N-1$. According to the window these values should be predicted to be zero, but this is unlikely since the preceding P values are used to predict them.

The rectangular window yields large errors at the beginning of the speech frame and is considered unsuitable for use with the autocorrelation method. Effects of the edges can be minimized by de-emphasizing samples near the edges with the use of a different window. A Hamming window was chosen with its mathematical expression as follows

$$W_n = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (32)$$

The use of this window gradually reduces the weight applied to the samples as the edge of the frame is approached.

The frame length N chosen is an important consideration. If the window doesn't encompass at least one pitch period the error can vary dra-

matically, depending on the position of the window with respect to the pitch period. A window which is too long will cause variations in the voice signal to be lost. It is good practice to include 2-3 pitch periods in the frame but this is a difficult estimation to make.

For a female voice the period can be 2 msec and for a low-pitched male 25 msec. A 20 msec frame was chosen resulting in 200 samples per frame at 10 kHz sampling rate. Therefore, a 200 point Hamming window will be used and is illustrated in Fig. 10. Other reasons for choosing this sampling rate are discussed in Chapter 6.

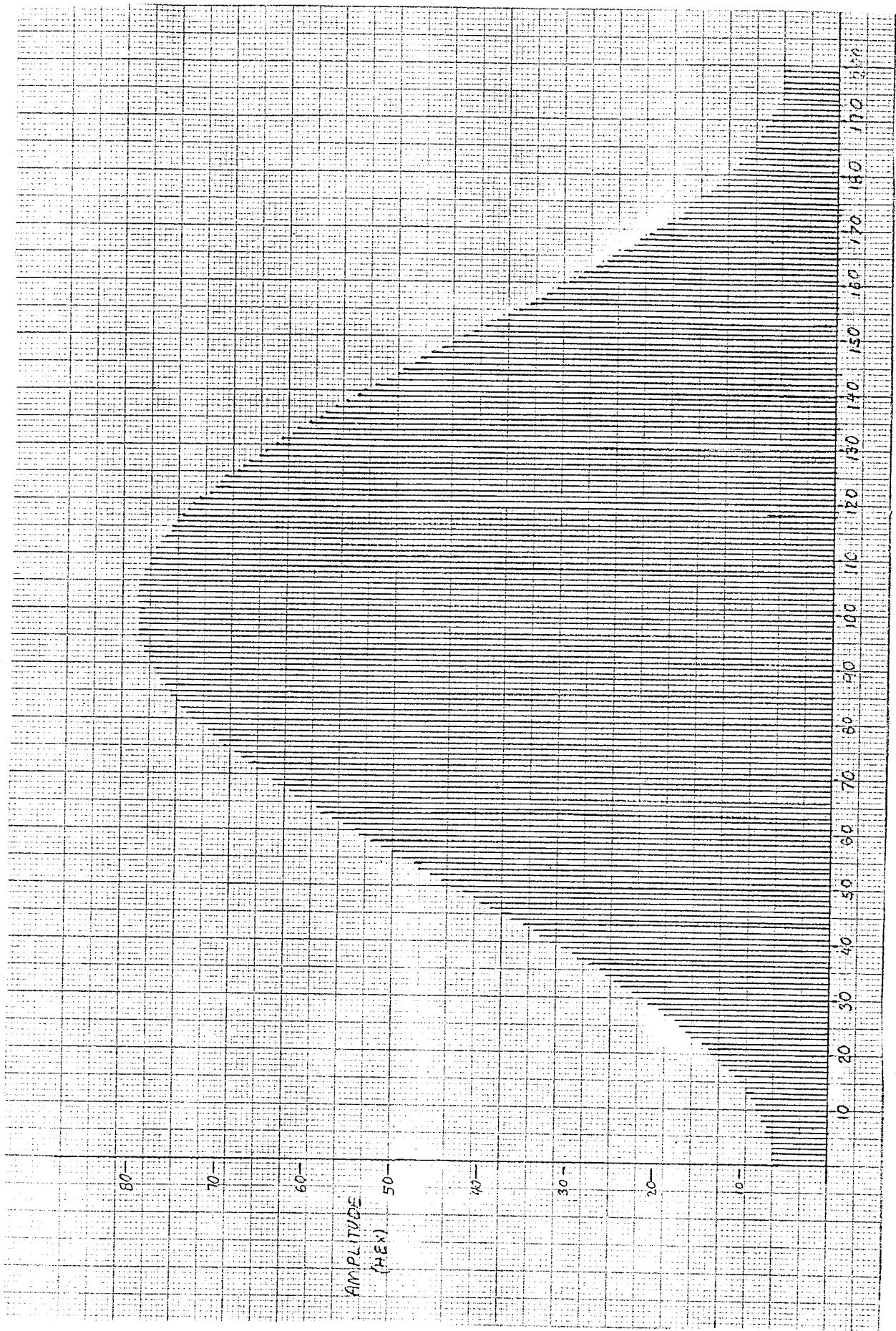


Figure 10: 200 Point Hamming Window

3.2 AUTOCORRELATION CALCULATION

The short time autocorrelation $R(k)$ is calculated using the Hamming windowed samples. This is defined by

$$R(k) = \sum_{m=0}^{N-1-k} \{x(m) w'(m)\} \{x(m+k) w'(m+k)\} \quad (33)$$

The values for $R(k)$, $0 \leq k \leq P$ are necessary for the autocorrelation method. Using the above definition, approximately $(P+1)N$ multiplications are required for $P \ll N$, not including windowing. $R(k)$ has some useful properties, some of which can be exploited in the calculation. These properties are:

- a) It is an even function, $R(k) = R(-k)$;
- b) Maximum value is at $k=0$ i.e. $|R(k)| \leq R(0)$; and
- c) $R(0)$ is the energy in the frame.

For each lag $m = 0$ it was noted by Blankenship [5] that most of the input samples appear twice as multiplicands in the calculation. For $k=1$ this is illustrated as follows

$$\begin{aligned} R(1) &= x(0)x(1) + x(1)x(2) + x(2)x(3) + x(3)x(4) + \dots \\ &= x(1)(x(0) + x(2)) + x(3)(x(2) + x(4)) + \dots \end{aligned} \quad (34)$$

Except for $k=0$ the number of multiplications are reduced by approximately 1/2 resulting in about $(1+P/2)N$ multiplications where $P \ll N$.

Kendall modified Blankenship's algorithm for use, recursively, with the unmodified short time autocorrelation defined by

$$R(k) = \sum_{m=0}^{N-1-k} x(m)x(m+k) \quad (35)$$

where x is the windowed signal. N is required to be even for this algorithm and so was previously set to be 200.

Kendall's algorithm is used for its computational efficiency and is expressed below

$$R(k) = \sum_{m=0}^{(N-k)/2-1} \{x(2m) + x(2m+k+1)\} \{x(2m+k) + x(2m+1)\} - A(k) - B(k) \quad \text{for even } k \quad (36)$$

$$R(k) = \sum_{m=0}^{(N-k-1)/2-1} \{x(2m) + x(2m+k+1)\} \{x(2m+k) + x(2m+1)\} - A(k) - B(k) - x(N-1-k)x(N-1) \quad \text{for odd } k$$

where $A(k)$ and $B(k)$ are obtained recursively, (37)

$$A(k) = A(k+2) + x(N-2-k)x(N-1-k) \quad \text{for even } k \quad (38)$$

$$B(k) = B(k+2) + x(k)x(k+1) \quad \text{for even } k \quad (39)$$

$$A(k) = A(k+1) \quad \text{for odd } k \quad (40)$$

$$B(k) = B(k+2) + x(k)x(k+1) \quad \text{for odd } k \quad (41)$$

It is convenient to evaluate $R(k)$ from $k=0$ to $k=P$ to ascertain the need for and to perform normalization. This is used to prevent overflow

and will be discussed later. When starting the algorithm at $k=0$, $A(0)$ and $B(0)$ are needed first. The expansion of the recursive relation is shown as

$$A(0) = x(0)x(1) + x(2)x(3) + \dots + x(N-2)x(N-1) \quad (42)$$

$$B(0) = x(N-2)x(N-1) + x(N-3)x(N-4) + \dots + x(1)x(0) \quad (43)$$

Thus $A(0) = B(0)$ will be the initial condition for a modified recursive relation where k is even. This is shown as

$$A(k) = A(k-2) - x(k-2)x(k-1) \quad k \text{ is even} \quad (44)$$

$$B(k) = B(k-2) - x(N-k+1)x(N-1) \quad k \text{ is even} \quad (45)$$

For $k=1$, we have

$$A(1) = A(0) \quad (46)$$

$$B(1) = x(N-3)x(N-2) + x(N-5)x(N-4) + \dots + x(1)x(2) \quad (47)$$

For k odd the modified relation is

$$A(k) = A(k-1) \quad \text{for } k \text{ odd} \quad (48)$$

$$B(k) = B(k-2) - x(N-k+1)x(N-k) \quad \text{for } k \text{ odd} \quad (49)$$

3.3 PREDICTOR SOLUTION

The normal equations are presented below in matrix form for the autocorrelation method.

$$\begin{bmatrix} R(0) & R(1) & \dots & R(P-1) \\ R(1) & R(0) & \dots & R(P-2) \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ R(P-1) & \dots & \dots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ \cdot \\ \cdot \\ a_p \end{bmatrix} = \begin{bmatrix} R(1) \\ R(2) \\ \cdot \\ \cdot \\ \cdot \\ R(P) \end{bmatrix} \quad (50)$$

The autocorrelation matrix and the right hand side are known using Kendall's algorithm to solve for the first P lags. The matrix being symmetric Toeplitz led to the development of Durbin's recursive solution [5]. To date this is the most efficient algorithm known for solution of the predictor coefficients a_k . Although the primary objective of solving for the a_k is met, other valuable information is obtained with this algorithm. The PARCOR, prediction coefficients and the total mean squared error E are yielded. Durbin's algorithm is expressed below as

$$E^{(0)} = R(0) \quad (51)$$

$$k_1 = \frac{R(1)}{E^{(0)}} \quad (52)$$

$$E_1 = (1 - k_1^2) E^{(0)} \quad (53)$$

$$k_i = \frac{\{R(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)\}}{E^{(i-1)}} \quad (54)$$

$$a_i^{(i)} = k_i \quad (55)$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)} \quad 1 \leq j \leq P \quad (56)$$

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \quad (57)$$

These equations are solved recursively for $i=2,3,\dots,P$ with the final solution given by

$$a_j = a_j^{(P)} \quad 1 \leq j \leq P \quad (58)$$

It can be seen that the solution for predictor coefficients of all orders less than P have been obtained.

System stability can be easily monitored by checking the PARCOR coefficients k_i as they are calculated, so that $-1 \leq k_i \leq 1$ holds.

The following data are saved for each frame processed:

- a) Predictor coefficients for order P ;
- b) PARCOR coefficients k_1, k_2, \dots, k_P ; and
- c) Total mean squared error for frame E .

3.4 COMPUTATIONAL CONSIDERATIONS

The number of calculations are considered on the basis of using the autocorrelation method with 10 predictor coefficients. For large N the number of multiplications required for the autocorrelation solution far exceeds that of the predictor coefficient solution, even when Kendall's algorithm is used. Subtractions are considered computationally the same as additions and divisions the same as multiplications.

With $N=200$, about 1500 multiplications per frame must be done requiring the performance of 75,000 multiplications per second. A very important aspect of the computational procedure is that of data shuffling. This is optimized as described later to reduce the time required.

Chapter IV

SYSTEM OVERVIEW

Implementation of the algorithms is done with a specially designed processor which will henceforth be called the LPCMK. The system has been designed so as to achieve the following two objectives:

- a) Speed of operation; and
- b) Flexibility.

Speed is important because of the amount of calculation and data shuffling required for real time operation. Sufficient flexibility must exist to meet the objective of a speech research facility. If the LPCMK were dedicated to a single task the research potential would be too limited.

4.1 THE BIT SLICE APPROACH

To meet the speed requirements, the use of standard microprocessors would require the implementation of parallel processing. An alternative approach was chosen with the design of a special purpose processor.

Bit slice components are used which have the following major advantages over standard microcomputers:

- a) The processor architecture can be tailored to meet the user's needs;
- b) The instruction set is defined by the user; and
- c) Instruction execution is much faster.

Any standard microprocessor will have a fixed instruction set, fixed architecture, and be slower than a bit slice processor. The major disadvantage of the bit slice approach is the higher cost in terms of power consumption, development time, and component cost.

In a typical bit slice processor, including the LPCMK, there are two levels of instructions. The most primitive are microinstructions, each of which is 48-bits wide in LPCMK. A program using a series of microinstructions is referred to as a microprogram. A program level instruction is used to call a microprogram. Access to operands is controlled by the microprogram, whose function and length depend totally on the users definition.

4.2 SYSTEM HARDWARE

4.2.1 Components

A block diagram of the system configuration is shown in Fig. 11. It consists of the Motorola Mace development system, including the EXORcisor, the LPCMK, the program memory, and the preprocessor with analog to digital converter (A/D). The latter three items each occupy one standard S-100 wirewrap protoboard.

The Mace functions as a very fast (50 nsec access time) writeable control store (WCS) for the LPCMK's microinstructions. After program development the Mace can be replaced with PROM memory, which is normally just as fast.

The LPCMK is the complete bit slice processor. When operating, it supplies addresses to the Mace to access the microinstructions. These microinstructions determine the operation of the LPCMK, which include

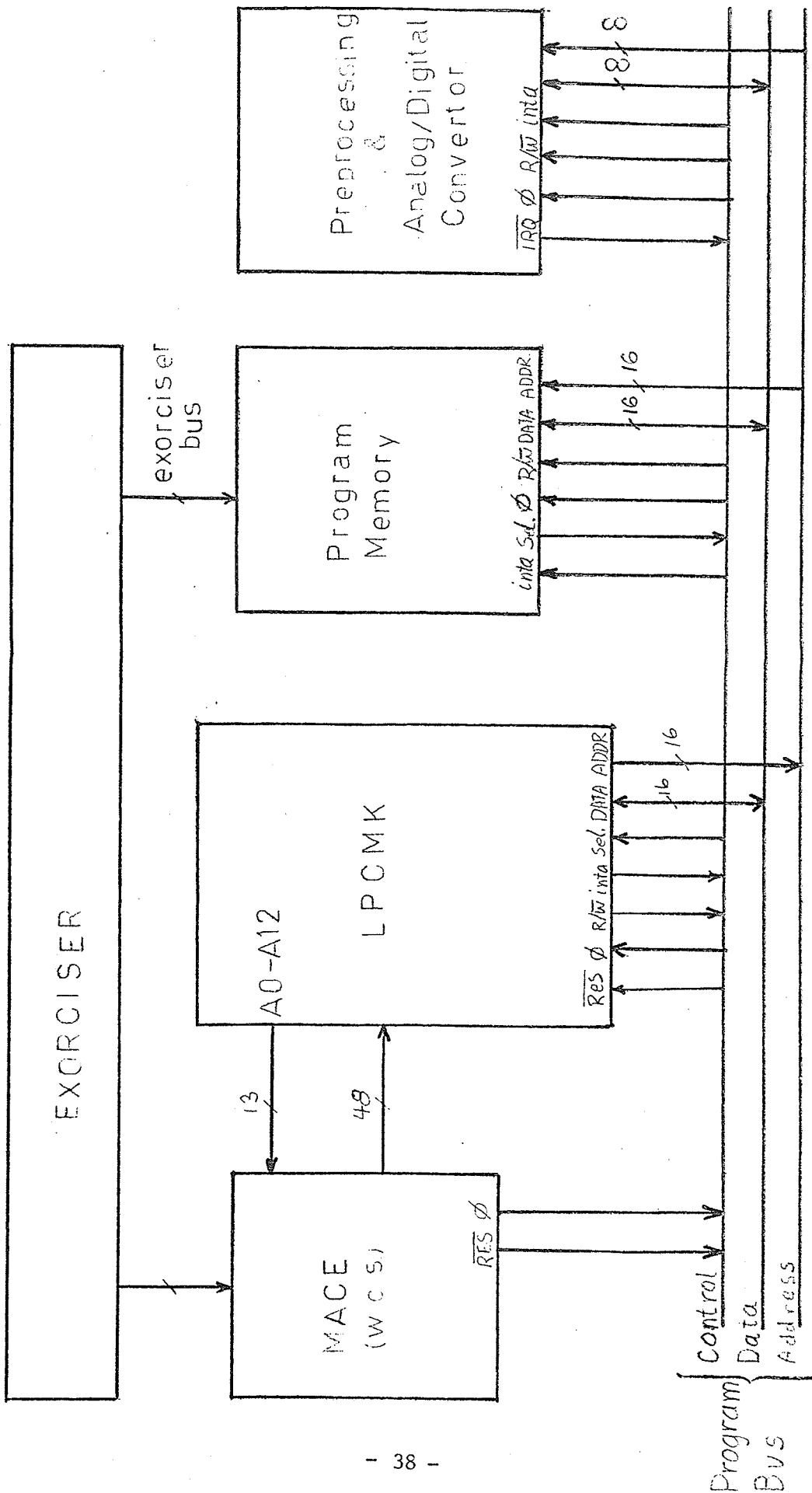


Figure 11: System Hardware Configuration

fetching data and instructions from the program memory. Also, the LPCMK fetches data from the A/D converter and processes it.

The program for the LPCMK and results of the computations are stored in the program memory. Various buffers used during program and microprogram execution are also located in the program memory. This memory is accessed by the EXORciser as well for the purpose of loading the program and data input and output (I/O).

The preprocessor with A/D converter supplies speech samples to the LPCMK. When the analog speech signal has been converted to digital form, an interrupt request is sent to the LPCMK indicating the data is ready. One sample of speech data can then be accessed by the LPCMK.

4.2.2 System Hardware Operation

There are two operational modes for the system. They are the EXOR and the LPCMK mode. These modes are selected by the combination of a switch setting on the program memory card and the MBUG software, which is discussed in Sec. 4.3.

The EXOR mode allows the Motorola EXORciser to access the program memory and the Mace WCS. During microprogram development, the Mace is loaded with microinstructions and the program memory is loaded with the program, including constants, from the EXORciser. After the LPCMK program has been executed, the results are read from the program memory in this mode.

While in the EXOR mode the LPCMK data bus and address bus lines are tri-stated. The EXORciser then has control over the program memory since the LPCMK address is disabled.

The LPCMK is the bus master when the LPCMK mode is chosen. The EXOR-ciser bus is tri-stated and the LPCMK accesses the program memory, pre-processor and A/D. The LPCMK address is enabled allowing microprogram operation. Data is acquired and processed to yield LPC coefficients in this mode.

4.3 SYSTEM SOFTWARE OPERATION

Microprogram development is facilitated by use of the Motorola micro-assembler called MASM [7]. The first step requires the creation of a definition file in which the desired bit fields are assigned to mnemonics or labels defined by the user. These mnemonics are then used in microprogram development to build the microinstruction. In assembly of the microprogram reference must be made to the appropriate definition table, which is created by assembly of the definition file.

The assembled program is then loaded into the Mace and operated by the Motorola program MBUG [7]. MBUG controls the system clock and reset and enables the LPCMK address to the Mace WCS.

Chapter V
HARDWARE DESCRIPTION

The system described in Chapter 5 consists of the following components:

- a) Preprocessor with A/D converter;
- b) Program memory;
- c) LPCMK bit-slice processor; and
- d) EXORciser with Mace development system.

In this Chapter the first 3 items are described in detail. The Mace is considered in [7].

5.1 PREPROCESSOR AND A/D CONVERTER

5.1.1 Preprocessor

The preprocessor is responsible for converting the approximately 10 mV peak-to-peak analog signal from the microphone into a useable form for the A/D converter. The three aspects which must be considered in the analog processing are:

- a) A sufficient signal voltage must be applied to the A/D converter;
- b) Bandlimiting of the signal to $|F| < 1/|2T|$, where T is the sampling period, is necessary to avoid aliasing; and
- c) Spectral pre-emphasis is desirable to spectrally flatten the signal spectrum.



A block diagram of the entire preprocessor and A/D conversion is shown in Fig. 12 and the circuit schematic can be found in Appendix I.

The first stage consists of an amplifier with a gain $A_v = V_o/V_i = 1000$. This supplies the antialiasing filter with a ± 5 V maximum input signal.

A VCVS two stage, four pole Butterworth Saliient-Key filter was chosen for the antialiasing filter. This is suitable for use with speech processing because it has no ripple in the pass band. The 3 dB point is set at 3.5 kHz to assure sufficient band limiting. To avoid aliasing an insignificant amount of signal energy must exist above 5kHz. The spectrum amplitude obtained is shown in Fig. 13. It was shown in Sec. 2.1.1 that there is approximately one formant for each 1000 Hz in the vocal tract. The signal used will therefore contain about 4 formants.

It was previously stated in Sec. 2.3.4 that spectral pre-emphasis aids in reducing roundoff error when computing the autocorrelation [5]. This was found to be unnecessary with the word lengths used when the results were examined. These are presented in Chapter 8. The study of these effects is not included in the scope of this work, but will be mentioned in Sec. 6.1 when the software implementation is considered.

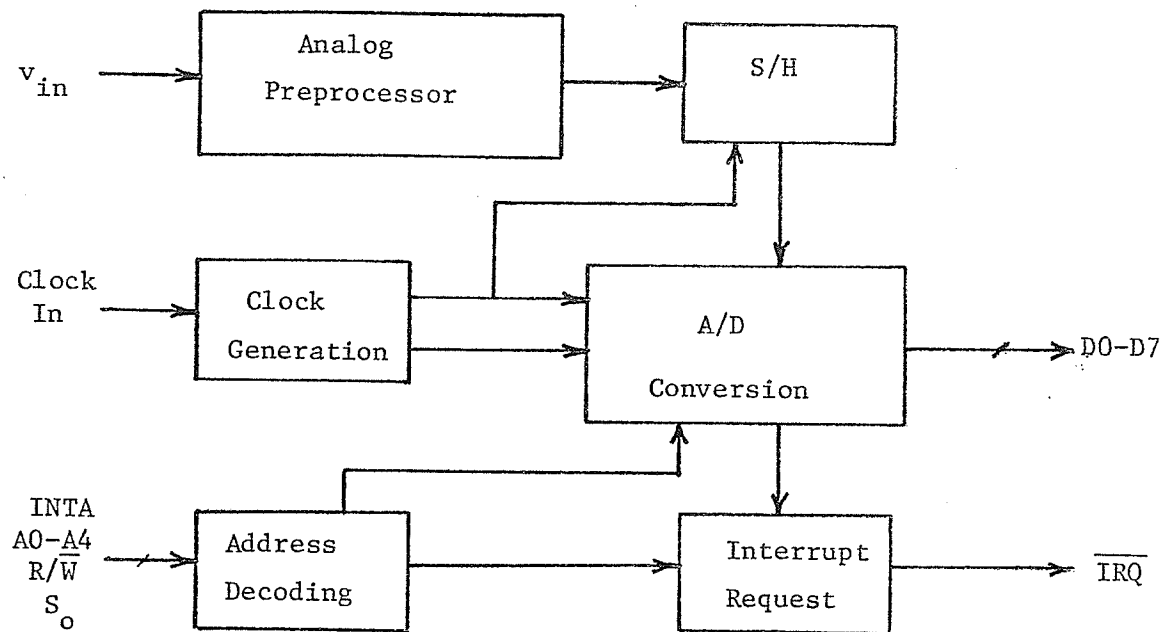


Figure 12: Preprocessor and A/D Block Diagram

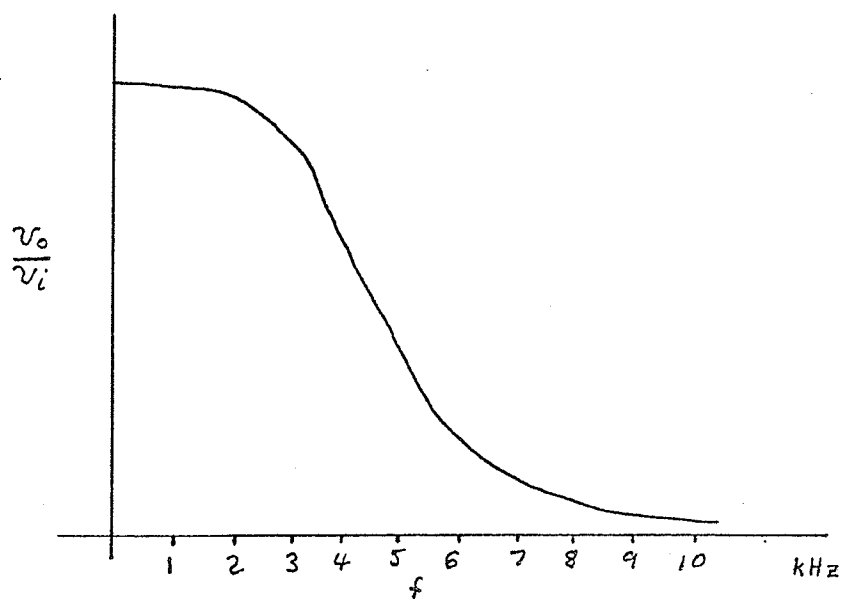


Figure 13: Preprocessor Frequency Response

5.1.2 Sampling

Although the signal is band limited to $f < 4000$ Hz, a sampling rate of approximately 10kHz was chosen. At this rate, aliasing is avoided with a safety margin. Also, as a matter of convenience, the conversion clock T_{cc} is obtained from the system clock T_{sy} which has a period of 200 nsec. A conversion clock of 625 kHz was used. This is obtained by $T_{cc} = T_{sy}/8$, implemented with a simple frequency divider. The start-of-conversion clock T_{sc} resulted from the following relation.

$$T_{sc} = T_{cc}/4 \cdot T_{cc}/8 \cdot T_{cc}/10 \cdot T_{cc}/32 \cdot T_{cc}/64 \quad (59)$$

This was implemented with a logical AND gate and resulted in a start-of-conversion pulse T_{sc} lasting $2T_{cc}$ sec every 102.4 usec. This relation is shown in Fig. 14. The timing is required for the A/D converter chosen. The actual sampling rate is 9,766 Hz.

According to Fig. 15, a sample and hold (S/H) is required for the 4 kHz signal since the conversion time is 64 usec at the chosen conversion clock period. The aperture time of a 4 kHz signal is 300 nsec. Sample acquisition is triggered by T_{sc} .

The leading edge of the end-of-conversion signal \overline{EOC} , from the A/D converter triggers the interrupt request circuitry \overline{IRQ} . This signal is applied to the LPCMK program bus. When the LPCMK system is operating, the \overline{IRQ} is responded to with an interrupt acknowledge, INTA. This procedure is illustrated in Fig. 16. The A/D is port-mapped, so the converter's address is provided with the INTA, enabling the tri-state output buffer. Digital speech data is then read by the LPCMK at the same time the \overline{IRQ} is reset.

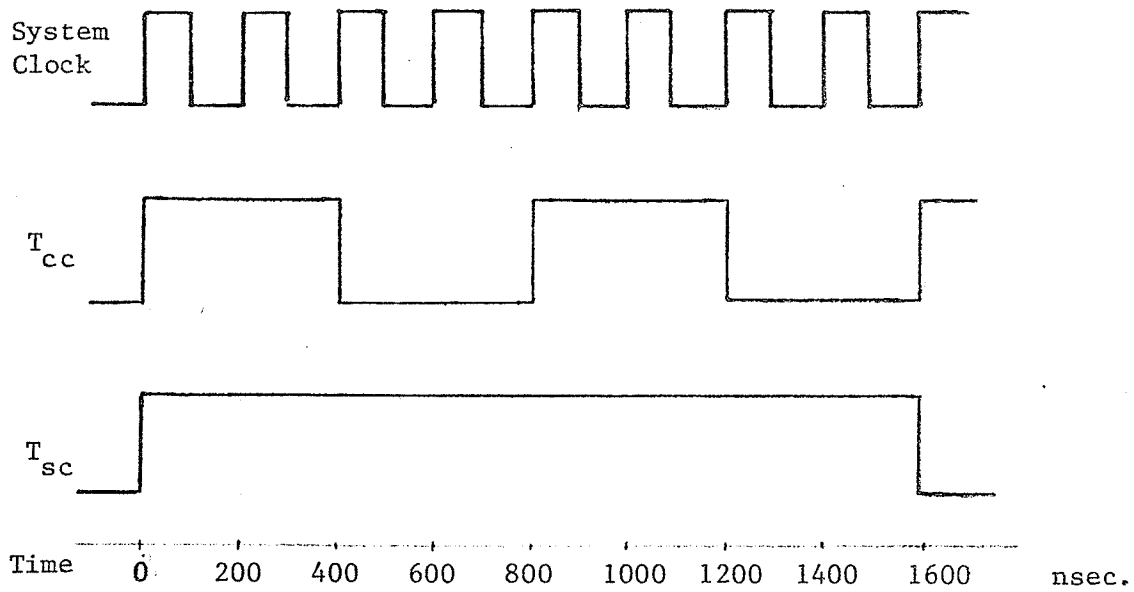


Figure 14: A/D Conversion Timing Diagram

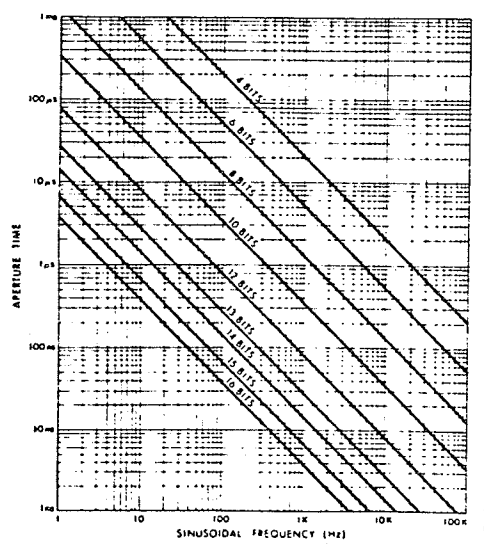


Figure 15: Aperture Time from [18]

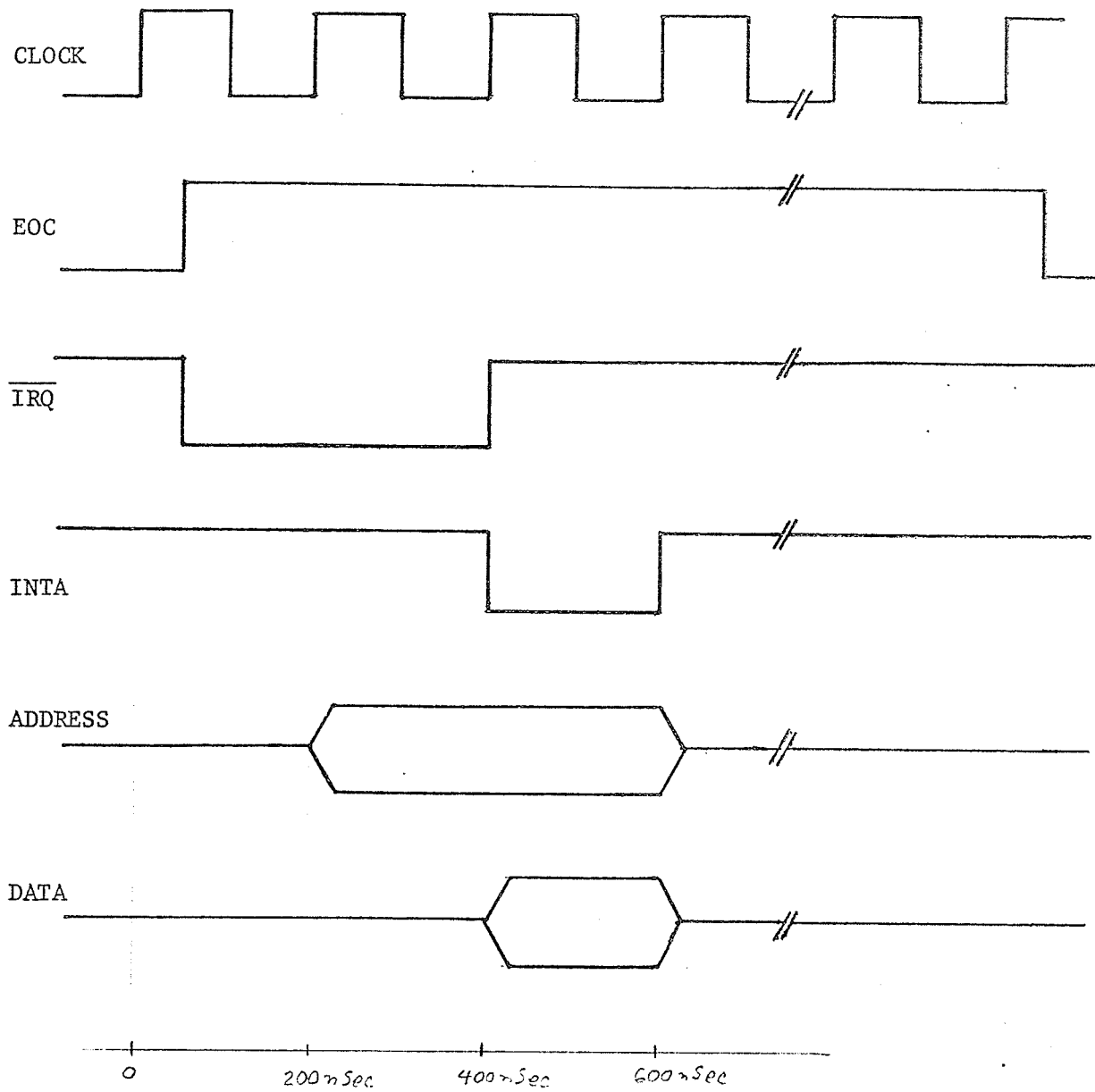


Figure 16: Interrupt Response Timing

A digital to analog D/A converter is provided and is used for testing the A/D converter.

5.2 PROGRAM MEMORY

The program memory is accessible from both the EXORciser microcomputer and the LPCMK. A switch is provided on the memory board to choose its mode of operation. This select signal is applied to the LPCMK program bus and serves to tri-state the LPCMK address and data bus when the EXOR mode is selected. A block diagram of the memory is shown in Fig. 17 and its schematic diagram is located in Appendix 1.

An important consideration is the data bus size difference between the EXORciser and the LPCMK. The former uses a standard microprocessor, the M6800, that requires an 8 bit data word and the LPCMK has a 16 bit data word. This disparity is resolved by use of tri-state bus transceivers for the EXORciser interface. The program memory is divided into two 4K x 8 bit banks which are alternately chosen by the EXORciser. The data bus of the LPCMK is tied directly to the memory data pins.

The memories used were 4K X 1 static RAMs with 120 nsec access times. This speed is sufficient to enable data access in one clock cycle of the LPCMK. To get the required 16 bit data width, 16 devices are needed. When the EXOR mode is chosen the LSB (Least Significant Bit) of the EXORciser's address is used with the chip select circuitry. This allows a 16 bit word accessed by the LPCMK to appear as two consecutive 8-bit words in the EXORciser's address space for ease in interpreting data.

The EXORciser bus is used directly by the memory, which is mapped to A000 - BFFF hexadecimal for the EXORciser. For the LPCMK the memory ap-

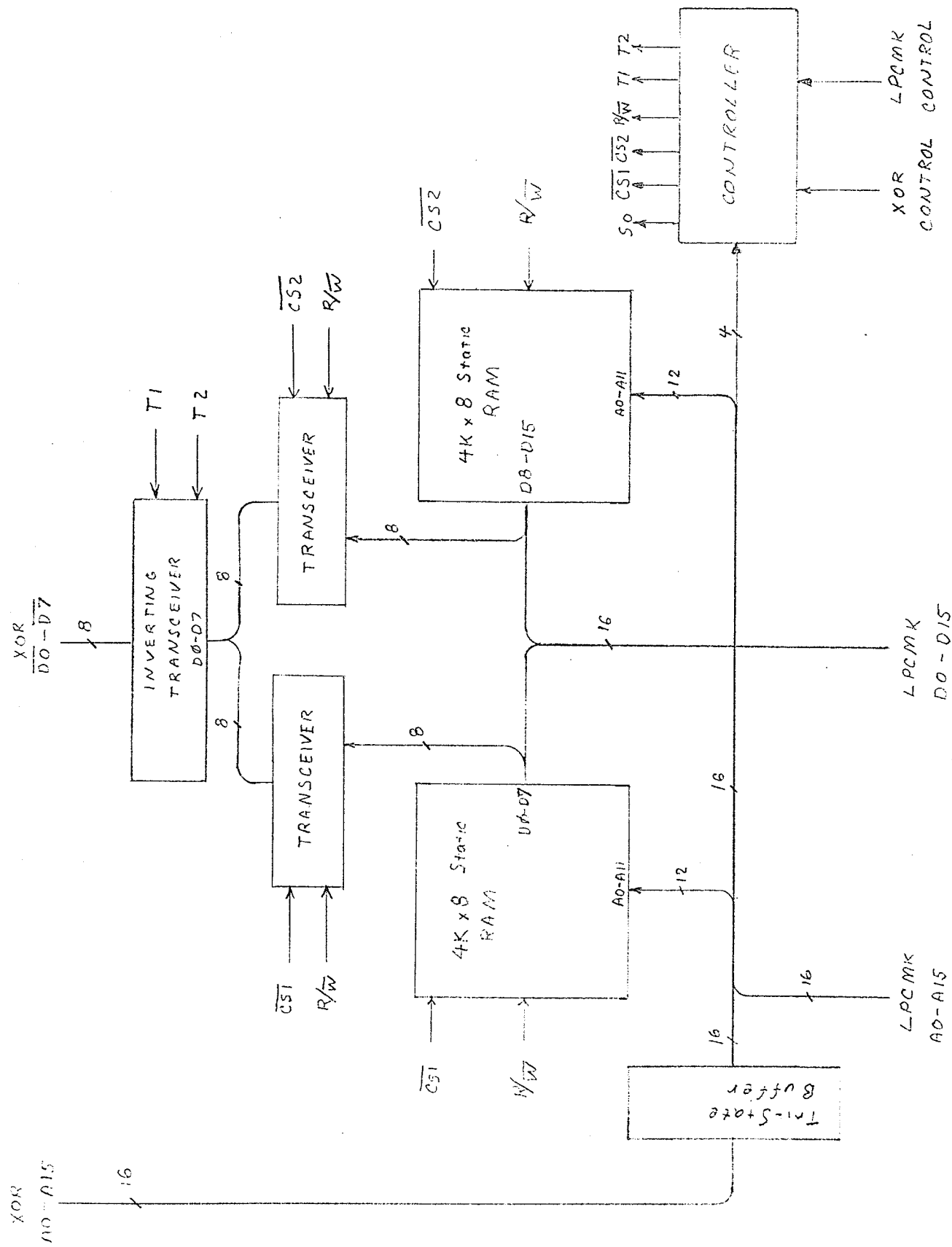


Figure 17: Program Memory Block Diagram

pears from 000-FFF hex, representing 4K words. This is adequate for storing over 2 seconds of predictor coefficients, autocorrelation values, PARCOR coefficients and error values.

The memory write and read timing is described in Sec. 5.3. A memory test program was written for EXORciser operation which requires 10 hours to run. A similar memory test program was written for the LPCMK that has a 4 minute execution time illustrating the speed advantage of the bit-slice approach. These test programs can be seen in Appendix 3 with accompanying flow charts.

5.3 THE LPCMK

The processor is the heart of the LPC research facility. Because of the advantages with the bit-slice approach, real time parameter generation is possible. The specially designed processor, called the LPCMK, is described here. One of the advantages mentioned for the bit-slice approach is the flexibility in architecture design. The following features are included in the LPCMK:

- a) A 48 bit wide microinstruction word;
- b) A 16 bit internal and external data bus;
- c) A 16 bit external address bus; and
- d) Single level interrupt servicing.

The major sections of the LPCMK are as follows:

- a) Pipeline Registers;
- b) Microinstruction Sequencer with immediate data buffers;
- c) Bit-slice ALU and register set (also called the bit-slice unit) with shift logic;

- d) Status register;
- e) I/O registers; and
- f) Mapping PROM and Vector Register.

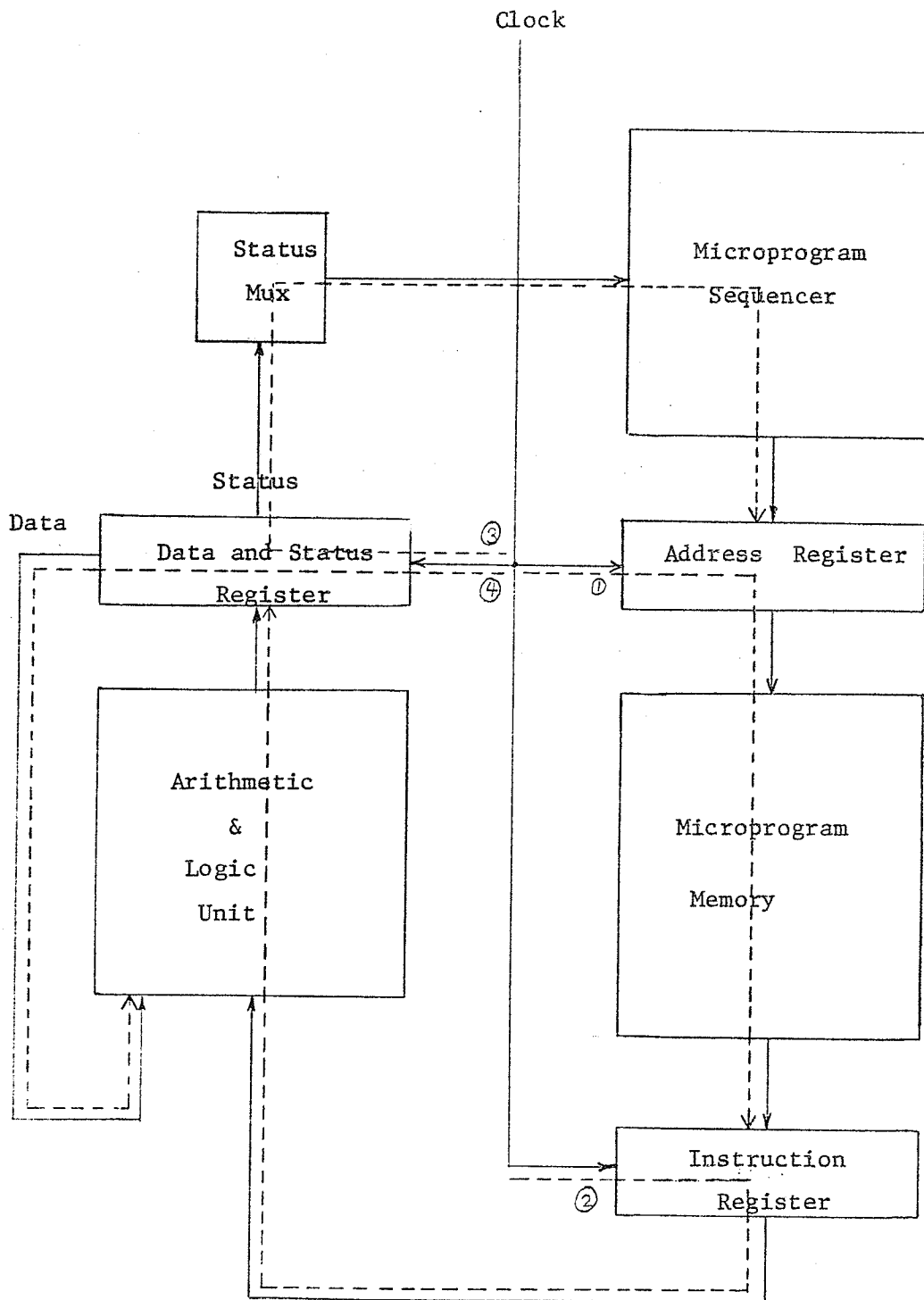
The components are discussed in detail following a description of bit-slice processor operation. The circuit schematic for each section is shown in Appendix 1.

5.3.1 Bit Slice Processor Operation

All events in the bit-slice processor are triggered by the leading edge of the system clock. This clock is provided by the Mace and was chosen to be 5MHz with a 50% duty cycle. A single clock period is the time required for the execution of one microinstruction. The devices used are static and so will not lose their state if the clock is stopped. This is valuable for program or hardware debugging. The standard procedure of inserting software interrupts where breakpoints are desired need not be applied.

A block diagram of a typical system is shown in Fig. 18. Since all events are started at the rising clock edges it is necessary for the combinational actions of the processor to settle before the next clock pulse arrives. Therefore the longest combinational signal path determines the maximum clock rate and hence the time required for each microinstruction.

Pipelining refers to the overlapping of events in time to reduce the total execution times. The location of pipelining registers determine the processor operating characteristics. In Fig. 18, the Instruction, Data and Address pipeline registers are shown at typical locations. If



1 , 2 , 3 , 4 Combinational delay paths.

Figure 18: A Typical Bit-Slice System Block Diagram

all three were used, the shortest clock period would depend on the longest delay path, possibly the microinstruction memory access time. Any combination of these three could be used providing different characteristics for operation [8].

An Instruction-Data architecture is used requiring the following two sets of registers:

- a) Instruction registers which receive and hold the microinstruction from the microinstruction memory; and
- b) Data Registers having the two purposes listed below;
 - i) Registers holding data from the previous operation are located in the AMD2901 ALU; and
 - ii) Status Registers that contain the status of the previous ALU operation or other information available at the previous clock edge.

The architecture chosen effectively causes an increase in the maximum clock speed permitted. The actual increase depends on the specific microinstruction being executed since not all propagation delay paths are equal. The major limitation in speed is that of the Mace WCS which has a 50 nsec access time. The memory is emitter coupled logic which is very fast. This, plus the delay of the microinstruction sequencer gives a minimum delay of about 120 nsec. The Mace provides the system clock in increments of 25 nsec with a 50 % duty cycle. Operation at periods of 175 nsec was possible before system failure. The processor was observed to fail with a 150 nsec clock period when performing the memory test program.

A clock period of 200 nsec was chosen to avoid electromagnetic noise problems on the LPCMK program bus. This is adequate for performing LPC in real time and allows program memory access in one microinstruction period.

A block diagram for the LPCMK is shown in Fig. 19. Microcoding considerations are presented in the following sections and in Chapter 6.

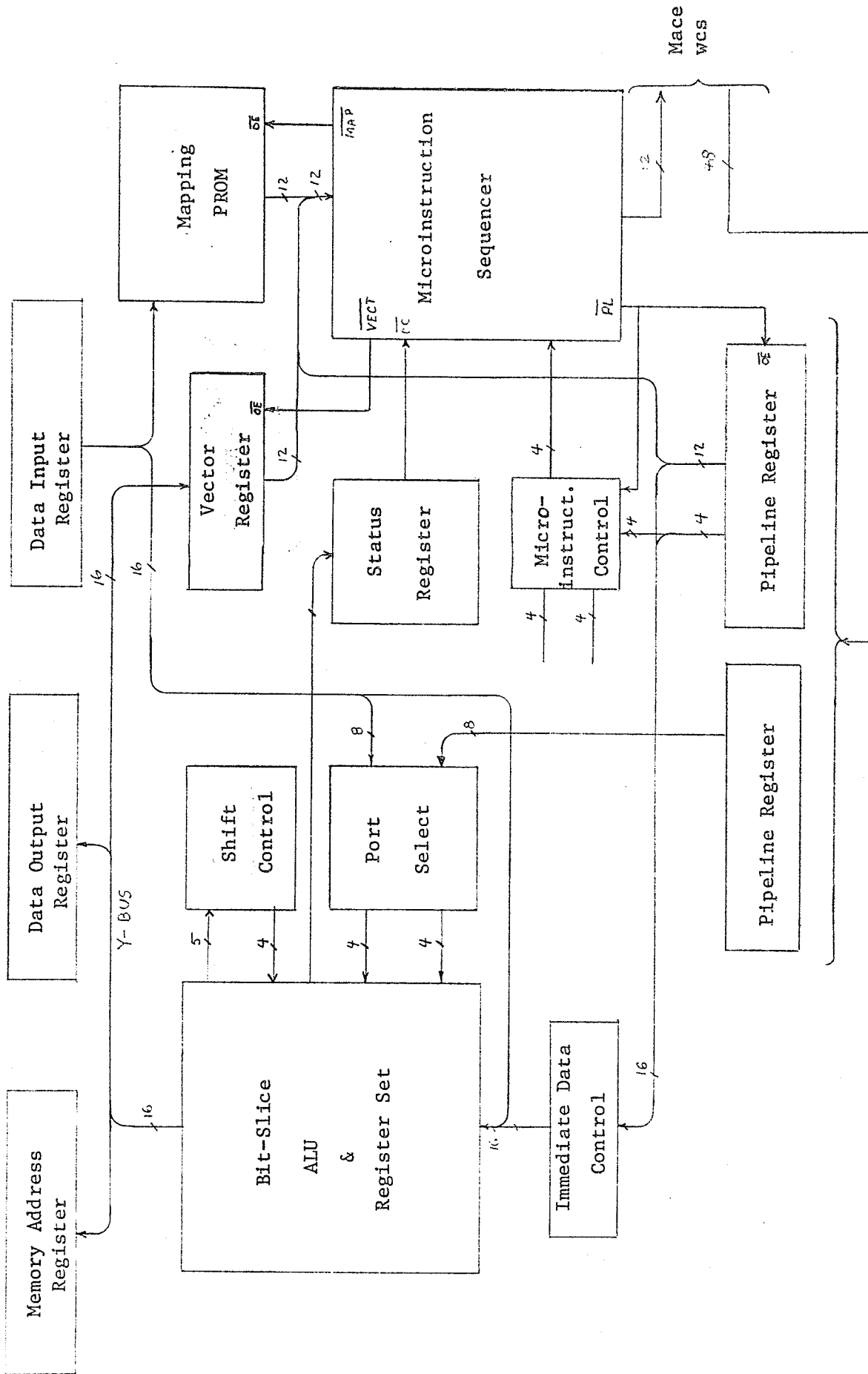


Figure 19: The LPCMK Block Diagram

5.3.2 Pipeline Register

The pipeline register refers specifically to the register that is loaded with the microinstruction. The address to the microinstruction memory is supplied by the sequencer which is discussed in Sec. 5.3.3. The 48-bit wide microinstructions are loaded into the pipeline registers on the rising clock edge. Sufficient settling time is allowed for the microinstruction memory after applying the address. There are 8 of the registers shown schematically in Appendix 1. The specific microinstruction fields are listed in Table 1.

The microinstruction bits are used throughout the LPCMK to control the sequencer, bit-slice unit, other combinational elements of the LPCMK, or as strobes for triggering events such as loading the I/O or status registers.

There are 2K x 48-bits of microinstruction memory in the Mace WCS. Occasionally it is desirable to have immediate data in the microinstruction. It is wasteful in most cases to devote 16 microinstruction bits to the task at all times, therefore the immediate data field is multiplexed with the sequencer field. The 12 next address bits and 4 sequencer instruction bits are also used for 16 immediate data bits. Another bit called the IMD (Immediate Data control) selects between the two modes. When the IMD is low the following changes occur in the microinstruction operation:

- a) Immediate data is supplied to the LPCMK ALU from the pipeline register; and

b) A continue (CONT) instruction is given to the sequencer through hardware control.

TABLE 1
Allocation of Microinstruction Bits

BIT	MNEMONIC	FUNCTION(S)	BIT	MNEMONIC	FUNCTION(S)
0	BR0		27	A0	
1	BR1		28	A1	ALU
2	BR2		29	A2	PORTA
3	BR3		30	A3	IMMEDIATE
4	BR4	MICROPROG.	31	B0	
5	BR5	BRANCH	32	B1	ALU
6	BR6		33	B2	PORTB
7	BR7	IMMED.	34	B3	IMMEDIATE
8	BR8	DATA	35	ABM0	PORT ADDRESS
9	BR9		36	ABM1	SELECT MUX.
10	BR10		37	LADD	LATCH ADDRESS
11	BR11		38	LDV/INTA	INT.ACK./VECT LOAD
12	mI0		39	LDI	LOAD DATA IN
13	mI1	MICROPROG.	40	LDO	LOAD DATA OUT
14	mI2	CONTROL	41	SS0	
15	mI3		42	SS1	STATUS
16	IMD	IMMEDIATE DATA CNT.	43	SS2	SELECT
17	I0		44	ROTO	ROTATE
18	I1	ALU	45	ROT1	SELECT
19	I2	SOURCE	46	STAT	STATUS CONTROL
20	I3		47	R/ \bar{W}	READ/WRITE
21	I4	ALU ALU			
22	I5	FUNCTION CONTROL			
23	I6				
24	I7	ALU			
25	I8	DESTIN.			
26	C0	ALU CARRY-IN			

The price paid for multiplexing the microinstruction bits is in the IMD control bit and the limitation of the CONT instruction imposed upon microprogram development. The LPCMK block diagram is shown in Fig. 19.

Another consideration is that of system reset or power-up microinstruction address generation. This is done through hardware by supplying the sequencer instruction ZERO when the $\overline{\text{RESET}}$ pulse goes low. This sets the microinstruction address to location zero to begin execution of the required initialization routines. The reset pulse must be at least one complete clock period and is supplied by the Mace Development system.

5.3.3 Microinstruction Sequencer

The sequencer is responsible for supplying the present address and preparing the next microcomputer address for the program memory. A complete 12 bit sequencer, the AMD2910, is used for program control. It has an address space of 4K words, although there are only 2K words available from the existing configuration of the Mace WCS. This is an enormous amount of memory when the power of each microinstruction is considered.

The microinstruction sequencer, which is illustrated in block diagram form in Fig.20, has three internal registers which are clocked by the rising clock edge.

The microaddress register is loaded with the results of the incrementer which, from the chosen configuration, will always be the next sequential instruction address. The stack pointer and register/counter actions depend on the sequencer instruction being executed.

The sequencer requires a four bit control code giving it a repertoire of 10 conditional and 6 unconditional next addresses.. A condition is assumed to be met if the $\overline{\text{CC}}$ and $\overline{\text{CCEN}}$ is low. The sequencer [9] [10]

IDM2910A Block Diagram

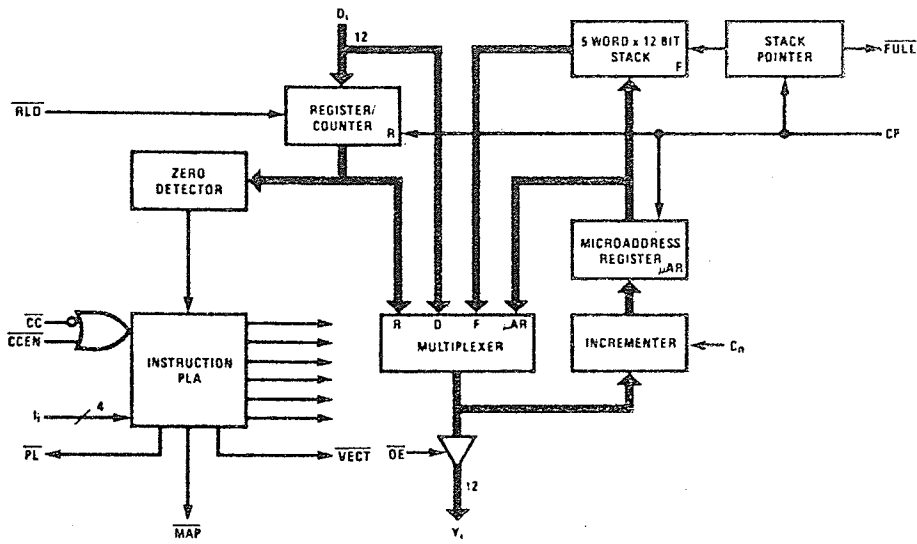


Figure 20: Microinstruction Sequencer Block Diagram

provides the powerful microprogram control required for high processing throughput.

The next microinstruction address will come from one of the following four sources:

- a) The microprogram address registers;
- b) The external direct input lines;
- c) The register/counter containing a previously loaded address; or
- d) The LIFO (last in first out) stack.

When the microinstruction bit allocation is described in Sec. 5.3.3 it is shown that there is almost no overhead in time for program control. There is a cost in space, however, of the microinstruction bits which must be dedicated to the device.

The $\overline{\text{PL}}$, $\overline{\text{MAP}}$, and $\overline{\text{VECT}}$ enable bits are supplied for next direct address control. Only one of these lines is low at any one time. They are controlled by the sequencer control code and simplify the hardware interface to be discussed in Sec. 5.6. The following inputs are tied to ground potential or V_{cc} since their capabilities are not required:

- $\overline{\text{CCEN}}$ - condition code input is always enabled.
- $\overline{\text{RLD}}$ - the register counter can only be loaded by an instruction.
- $\overline{\text{OE}}$ - the address output is always enabled.

The $\overline{\text{FULL}}$ output goes low one microinstruction after the 5 word stack is full. Since nesting of subroutines is not permitted past 5 levels, including interrupt service, this pin is ignored by the LPCMK.

5.3.4 ALU and Register Set

The AMD 2901B was chosen for use as the ALU (arithmetic logic unit) and internal register set [9] [10]. All data paths in this device are 4 bits wide. A block diagram is shown in Fig. 21. The 4-bit-slice can be cascaded to the number of bits required for the application. In the case of the LPCMK, a 16 bit data bus proved sufficient so four bit-slices were cascaded.

A 9-bit instruction is required by the AMD 2901B, which can be divided into three fields as follows:

- a) ALU source operand control;
- b) ALU function control; and
- c) ALU destination control.

Each is a 3-bit field and a carry-in input for the ALU is also required.

A 16-word two port RAM is available as a register set allowing two registers to be accessed simultaneously. Only the register addressed with the port B address can be written into. When the clock pulse is high the data addressed by port A and B addresses are read into their respective latches. The data are latched when the clock pulse goes low avoiding race conditions. Data is also written into the port B register while the clock pulse is low.

The Q-register is most useful for divide or multiply operations. Data is loaded to the Q-register on the low clock level as for the RAM.

The ALU is capable of performing three binary arithmetic and five logic functions. A look-ahead carry generator is used to increase the ALU speed of the cascaded bit-slices. The carry-in to the 16 bit ALU is provided by a microinstruction bit, and will alter the add and subtract

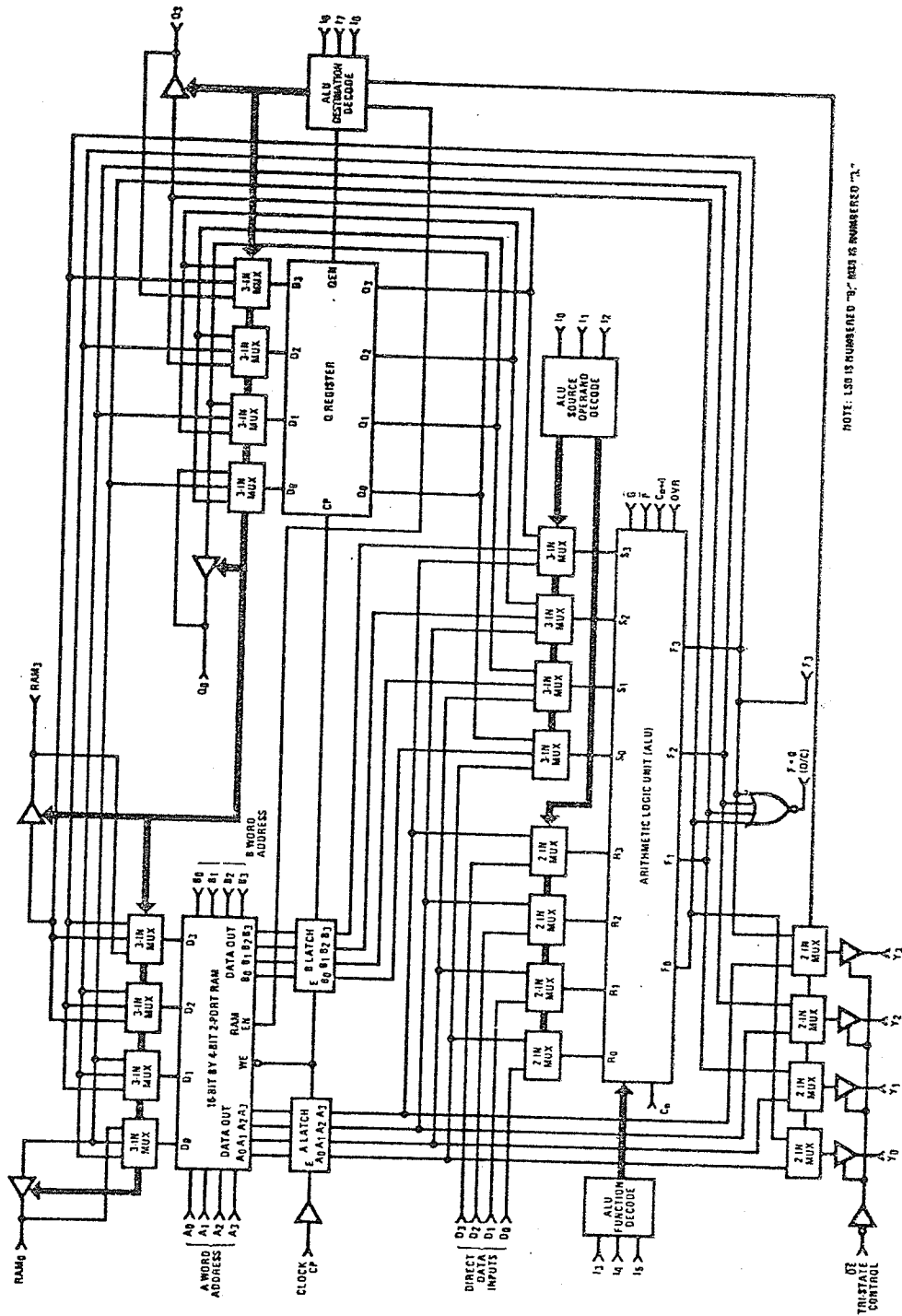


Figure 21: Internal Bit-slice Block Diagram

operations. The status of the ALU is indicated by the zero, carry-out, sign and overflow outputs from the ALU. These are used in the status register which is described in Sec. 6.3.4.

The ALU results appear at the output F and have one of the following possible destinations:

- a) ALU data is output to the Y-bus;
- b) ALU data is output to the Y-bus and stored in Q-register;
- c) ALU data is output to the Y-bus and stored in Port B location;
- d) Port A data is output to the Y-bus and ALU data is input to Port B register; or
- e) ALU data is output to the Y-bus and shifted once, up or down to the Port B register. The Q-register can be shifted up or down as well in the same operation.

The maximum clock speed allowed for the AMD 2901B is approximately 15 MHz. A 5 MHz system was implemented, which is well within the AMD 2901B operating specifications. Other propagation delay contributions must be considered when calculating the maximum system clock frequency.

The shift outputs of the AMD 2901B's are tri-stated unless a shift is selected by the destination field of the microinstruction. Shifting circuitry is necessary to supply these inputs which are the most and least significant bits of the Q-register and RAM addressed by Port B. This is implemented in the form of two tri-stateable dual 4-bit multiplexers. The schematic is given in Appendix 1. Two microinstruction bits are allocated to the multiplexers allowing the choice of shifts shown in Table 2. In this table RAM0, RAM15, and Q0, Q15 are the least and most significant bits of the Port B register and Q-register respectively.

TABLE 2

Rotate and Shift Destinations

MNEMONIC	FUNCTION	MNEMONIC	FUNCTION
RZD	0 > RAM15	RDRD	Q0 > RAM15
RQZD	0 > RAM15, 0 > Q15	RQDRD	Q0 > RAM15, RAM0 > Q15
RZU	RAM0 < 0,	RDRU	Q15 > RAM0
RQZU	RAM0 < 0, Q0 < 0	RQDRU	Q15 > RAM0, RAM15 > Q0
RDRD	Q0 < RAM15	SRA	F15 > RAM15 ARITH.
RQDRD	RAM0 > RAM15, Q0 > Q15	DSRA	F15 > RAM15, RAM0 > Q15
RSRU	RAM15 > RAM0	SLA	Q15 > RAM0
RQSU	RAM15 > RAM0, Q15 > Q0	DSLA	Q15 > RAM0, 0 > Q0

5.3.5 Status

Status of the previous microinstruction results is required due to the architecture chosen. Status of the ALU operation, interrupt status or unconditional status met is supplied to the \overline{CC} pin of the sequencer for conditional branch instruction execution. Table 3 shows the status field allocation. Three microinstruction bits are allocated to the status select control allowing 8 different events to be monitored. The status function is implemented with an 8-bit edge triggered register and an 8:1 digital multiplexer.

TABLE 3
Condition Codes

STATUS MNEMONICS	FUNCTION
QQQ	LSB of Q-Register
IRQ	External interrupt
QO	LSB of ALU output
OVR	Overflow
N	Sign
Z	Zero
C	Carry
MET	Condition always met

5.3.6 I/O Registers

An external interface is required for communicating with the Program Memory and Data Acquisition unit. The following three register sets, each consisting of two 8-bit registers, are used for the address and data interface to the external bus:

- a) Memory Address Register (MAR) supplies a 16-bit address;
- b) Data Output Register (DOUT) latches the output from the bit-slice ALU Y-bus; and
- c) Data Input Register (DIN) receives data or program instructions.

One microinstruction bit is reserved for controlling the loading of the MAR, DOUT, and DIN. Figure 22 shows the I/O timing for each of the registers. The MAR and DOUT registers are transparent latches, loading the data while the gate control is high. When the gate goes low the data is latched and will not change. This allows the maximum settling time for the external address and data buses. The MAR output is enabled as long as the LPCMK mode of operation which is described in Sec. 4.2.2

is selected. The DOUT output is enabled when the LPCMK mode is selected and a memory write operation, defined by another microinstruction bit R/\bar{W} , is active. The R/\bar{W} microinstruction bit is also supplied directly to the external bus for use with the data acquisition unit and program memory.

The DIN register is loaded on the leading edge of its clock input. This is done to allow data to be latched in during the same microinstruction that a new address is placed on the address bus. The output of the DIN register is enabled as long as the immediate data mode is not selected. The DIN output is used for instruction fetching and supplying data to the input of the bit-slice ALU.

A program instruction consists of an opcode and an operand. The opcode is loaded into the lower 8-bits of the DIN register and is used to supply an address to the Mapping PROM. When enabled, the Mapping PROM supplies an address to the microinstruction sequencer which is the address of the start of a microprogram that executes the instruction. The operand may be implied or supplied directly. Operand length depends on the instruction involved and the operand acquisition is controlled by the microprogram. Each instruction occupies a minimum of one word (16-bits) in the program memory.

The interrupt acknowledge control bit INTA is required for memory or port access. This microinstruction bit is supplied directly to the LPCMK bus in response to an \overline{IRQ} . It informs the program memory and data acquisition unit that a port access operation is taking place and also resets the \overline{IRQ} .

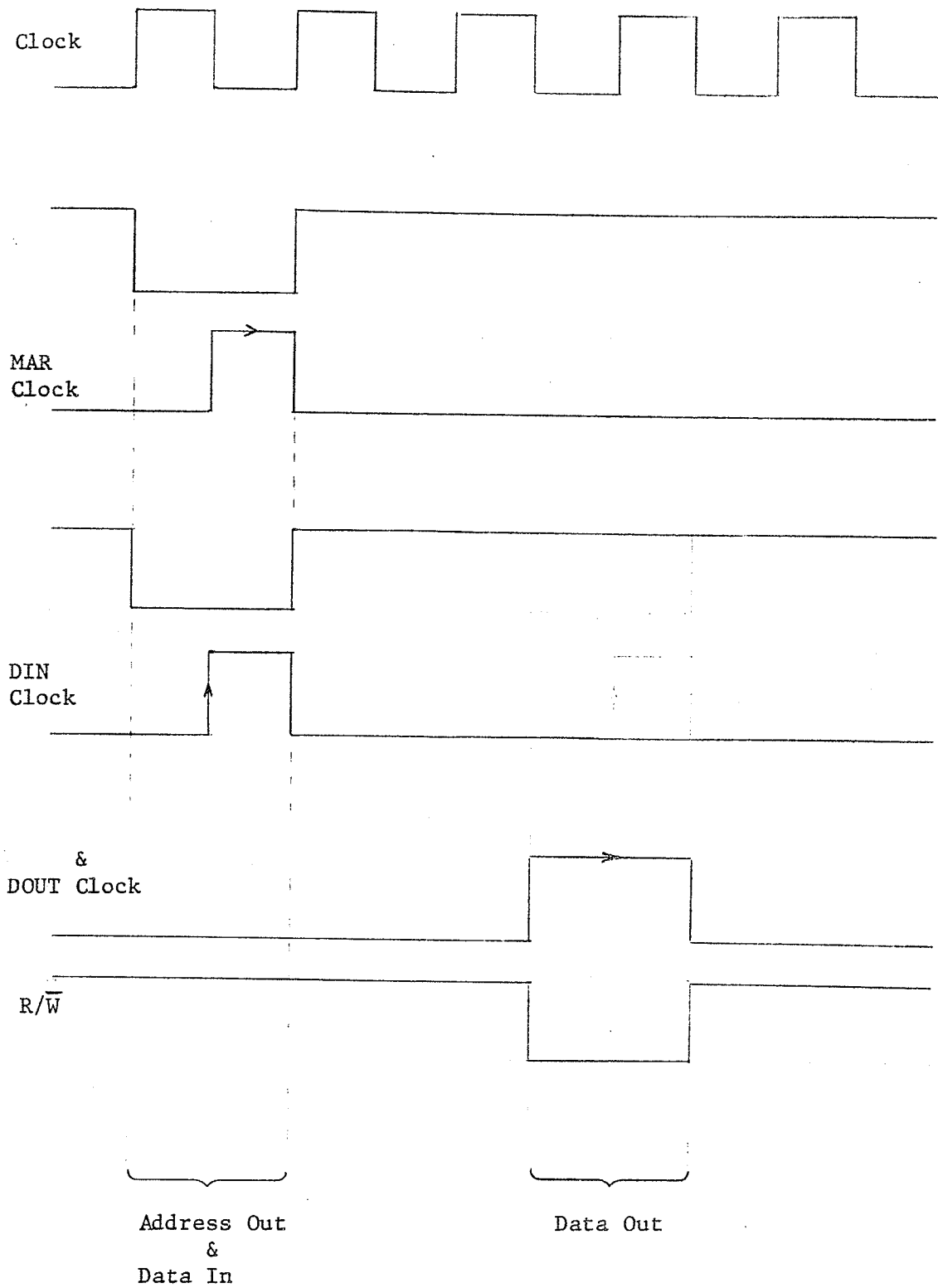


Fig. Input/Output Timing

5.3.7 Mapping Prom and Vector Register

The sequencer next address data input is available from three sources: the pipeline register, Mapping PROM or vector register. These are enabled by the \overline{PL} , \overline{MAP} , and \overline{VECT} outputs of the sequencer. Only one of the enable pins is active(low) at any time. The \overline{MAP} and \overline{VECT} inputs are enabled only during two sequencer microinstructions. When this happens the current microinstruction is latched since the pipeline register holding it is tri-stated. Also, immediate data is not available for ALU use at this time.

The Mapping PROM receives its address from the DIN register. It consists of a 256 x 8-bit and a 256 x 4-bit PROM which form 256 12-bit addresses. These are 256 possible instructions which are discussed in Chapter 6.

The Vector register is loaded from the bit-slice ALU on the first INTA signal. After this it cannot be loaded again. This is useful for setting up a vector address during program initialization. When enabled by \overline{VECT} the next address is supplied by this register.

5.4 EXTERNAL BUS

The External bus connects the preprocessor and A/D, the Program memory, and the LPCMK. It consists of an S-100 motherboard and associated connectors. Alternate pins are tied to system ground to aid in rejecting noise. The External bus assignment is shown in Appendix 1. The RESET and clock are supplied by the Mace Development System.

Chapter VI

SOFTWARE / FIRMWARE

The function of the LPCMK as mentioned earlier is that of a speech research facility. As such, more versatility and a larger memory is required than Hofsteter's vocoder [1]. Since speech synthesis is not required concurrently with LPC parameter generation, the speed advantage of a hardware multiplier is not necessary.

The vocoder [1] is a limited I/O machine with only a microprogrammed level of software. In practise, the microprogram is usually loaded into PROM after development. PROM presently has faster access time than most RAM and also has the advantage of permanently saving the microprogram.

The LPCMK, however, is designed with program level instructions. Program instructions are implemented by accessing microprogram routines through the instruction opcode. This opcode points to a location in the mapping PROM which supplies the next microinstruction address. When the microinstruction routine or microprogram has completed execution the LPCMK fetches another instruction. The microprogram determines the instruction operands.

This versatility permits changes at the program level without altering the microprogram. The advantages of a PROM microprogram store can then be exploited. The microprogram is stored in the Mace WCS and not loaded into PROM, however, since the versatility of microprogramming is desired.

As was described in Sec. 5.2, there is 4K x 16-bit static RAM available to the LPCMK for data and program storage.

The software for real time LPC is written in the following 3 distinct levels:

- a) Program instructions;
- b) Microprogram Instructions; and
- c) Microprogram Interrupt servicing.

6.1 PROGRAM INSTRUCTIONS

6.1.1 Program vs. Microprogram

Table 4 illustrates the tradeoff between program and microprogram instructions in terms of versatility, execution time and development difficulties.

Versatility refers to the number of different useful functions that can be programmed into the machine without changing the microprograms. The machine with a single instruction of GO will have no flexibility, unless its microcode is alterable.

Execution speed is defined as the time required for completion of a task. This includes program and microprogram execution time required for fetching and decoding instructions. The mapping PROM and program memory access times are the limitations in this case. This overhead percentage accumulates as the instructions perform smaller portions of the task.

As the individual microroutines become smaller and perform fewer functions, the level of difficulty in their development decreases. At the same time, program development becomes more involved.

Complete software development in the form of microcode would result in a faster executing program, but would unacceptably limit the versatility.

In view of these considerations it was decided to provide a minimal instruction set of:

- a) Data transfer;
- b) Arithmetic and Logical;
- c) Program control; and
- d) Special purpose instructions.

These are discussed in detail following a description of the internal register allocation.

TABLE 4

Microprogram vs. Program Tradeoffs

Microinst. per Instruction	Speed of Execution	Versatility	Microprog. Development	Program Development
1 uI = 1 Inst.	slowest	most	easy	difficult
1 Inst.= GO	fastest	none	difficult	easy

6.1.2 Register Allocation

There are 16 16-bit registers available for use by the LPCMK ALU. They are allotted the functions shown in Table 5.

TABLE 5

LPCMK Internal Register Allocation

Register	Mnemonic	Function
R0	PROG	Program counter
R1	ACCA	Accumulator
R2	ACCB	Accumulator
R3	ACCC	Accumulator
R4	ACCD	Accumulator
R5	ACCE	Accumulator
R6	AUTOHIGH	MSW for Autocorrelation
R7	AUTOLOW	LSW for Autocorrelation
R8	REFL	ki storage during DURBIN
R9	MCND	MCND and MSW of Dividend
R10	HAMLOC	Hamming window address
R11	M1	Data Acquisition for A0
R12	M2	Data Acquisition for B1
R13	LPP	Autocorrelation buffer loc.
R14	DACBUF	Data Acquisition Buffer loc.
R15	FLAG	Program Status
QREG		for LSW of MULT, DIVIDE & SHIFTS

The instructions can read or write to any of the registers, including the program counter, so care must be taken in designing the program.

The registers can be divided into three general groups, according to their functions as follows:

- a) Program Control;
- b) Interrupt Service; and
- c) Microprogram Execution.

The PROG register is set to the next instruction address by the ALU during execution of the current instruction.

Besides the program counter, the FLAG register is the only register used for controlling the program. At any time the status of program execution can be seen by examining the FLAG register. Its bit assignments can be seen in Table 6. The instruction TEST is used to check status when a conditional branch is to be performed. Its operation is detailed in Sec. 6.1.3.

TABLE 6
FLAG Register Bit Assignment

BIT	LOGICAL VALUE	STATUS
0	0	BUFFER 1 IS BEING FILLED
	1	BUFFER 2 IS BEING FILLED
1	0	NO MEMORY OVERFLOW
	1	MEMORY LIMIT EXCEEDED
2	0	R(0)<1
	1	R(0)>1 OVERFLOW
3	0	NOT END OF WORD
	1	END OF WORD
4	0	NOT START OF WORD
	1	START OF WORD
5-13		NOT USED
14	0	RESULT NOT ZERO
	1	RESULT EQUAL TO ZERO
15	0	TEST FAILS
	1	TEST PASS FOR EXTERNAL BRANCH

The DACBUF, M1, and M2 are dedicated for use with interrupt service in the data acquisition routine. When other registers are used by this routine their contents are first saved on the stack in the program memory. After completion of the interrupt service the contents are restored. All other registers are available for use by the microprogram.

6.1.3 Instruction Description

The list of instructions including their OPCODEs (OPERATION CODES) is summarized in Table 7 under the major categories of data transfer, program control, arithmetic and logical and special purpose instructions.

The list of instructions including their OPCODEs (OPERATIONAL CODES) is summarized in Table 9 under the major categories of data transfer, program control, arithmetic, and special purpose instructions. The mapping PROM configuration allows up to 256 instructions to be defined. Memory space exists on the mapping PROM for additional instructions.

Data movement instructions use operands provided by implication or as part of the instruction. In the immediate mode, data is provided as part of the instruction. This group of instructions can be used to set different initial values for other indices than those provided in INIT, which is described as a special purpose instruction.

One unconditional and two conditional branch instructions exist for program control. The next instruction address is supplied following the OPCODE as part of the instruction. All the branching instructions are 2 words (32 bits) long. The condition is considered met if the MSB of the FLAG register is set. If this bit is 0 the condition fails. A STOP and NOP (no operation) are also provided.

TABLE 9

LPCMK Instructions

1. DATA TRANSFER		
OPCODE	MNEMONIC	FUNCTION
01	LDRI X,RB : DATA	REGISTER B IS LOADED WITH DATA
02	LDRQ X,RB	REGISTER B < QREG
03	LDQR RA,X	QREG < REGISTER B
04	LDQ : DATA	QREG < DATA
05	LDRR RA,RB	REGISTER B < REGISTER A
06	LDM RA,(RB)	(REGB MEMORY) < REGISTER A
07	LDRMI X,(RB):DATA	(REGB MEMORY) < DATA
08	LDMI:DATA:ADDRESS	(ADDRESS MEM) < DATA
09	INIT	INITIALIZES INDEXES FOR LPCMK
2. PROGRAM CONTROL		
OPCODE	MNEMONIC	FUNCTION
0A	BRA : ADDRESS	UNCONDITIONAL BRANCH
0B	BRF : ADDRESS	BRANCH IF TEST FAILS
0C	BRC : ADDRESS	BRANCH IF CONDITION MET
0D	STOP	HALT PROGRAM EXECUTION
0E	NOP	NO-OPERATION

3. ARITHMETIC AND LOGICAL

OPCODE	MNEMONIC	FUNCTION
0F	TEST BIT,X	FLAG BIT SPECIFIED IS DUPLICATED AS MSB OF FLAG
10	MULT	MCND * QREG = ACCA,QREG
11	DIVIDE	MCND,QREG/ACCB = QREG

4. SPECIAL

OPCODE	MNEMONIC	FUNCTION
12	FILL	FILLS CURRENT BUFFER WITH WINDOWED DATA
13	SOW	FLAG BIT 4 IS SET IF START OF WORD
14	KENDAL	AUTOCORRELATION CALC. AND SAVED FOR 1 FRAME
15	DURBIN	PREDICTOR COEFFICIENTS, PARCOR COEFFICIENTS, AND FRAME ERROR IS CALCULATED AND SAVED
16	MOVFL	FLAG BIT 1 SET IF MEMORY OVERFLOW
17	EOW	FLAG BIT 3 IS SET IF END OF WORD DETECTED
18	STAT1	SETS 'RUN' LED FROM 'WAIT'
19	STAT2	SETS 'END' LED FROM 'RUN'

The major arithmetic instruction is TEST. The FLAG bit referred to by the operand is duplicated in the MSB of the FLAG register for use with a conditional branch instruction. The MULT instruction executes a 16 by 16 bit two's complement multiply with implied operands in MCND and the QREG. instruction. DIVIDE has an implied dividend in the register pair MCND, QREG and divisor ACCB.

Special instructions are designed to simplify the programming of real time LPC of speech. All of these instructions have implied operands and are one word (16 bits) in length. Most involve some form of data movement and arithmetic or logical calculation. All the special instructions are described in detail below.

INIT sets all indexes and constants required by the LPCMK program described in Sec. 6.3. The memory map assumed by INIT is shown in Fig. 23 with the addresses given in both the LPCMK and EXORciser memory space. As previously mentioned, the indexes can be adjusted by use of the data transfer instructions. Also, all the memory locations above 200H (LPCMK address) are cleared with INIT.

FILL insures that the current buffer is filled with windowed data so another frame is ready for processing. This is a 'wait-and-fill' command whose execution time depends on the amount of data required to fill the current frame. FILL calls the data acquisition routine on a polling basis, checking the $\overline{\text{IRQ}}$ signal with a loop. No flag bits are set by this instruction.

SOW compares the previous frame energy R(0) with a threshold to test for the start of the utterance. All the autocorrelation values are normalized so that a 1 is in the second MSB position. The number of left

EXORCISER Mode

LPCMK Mode

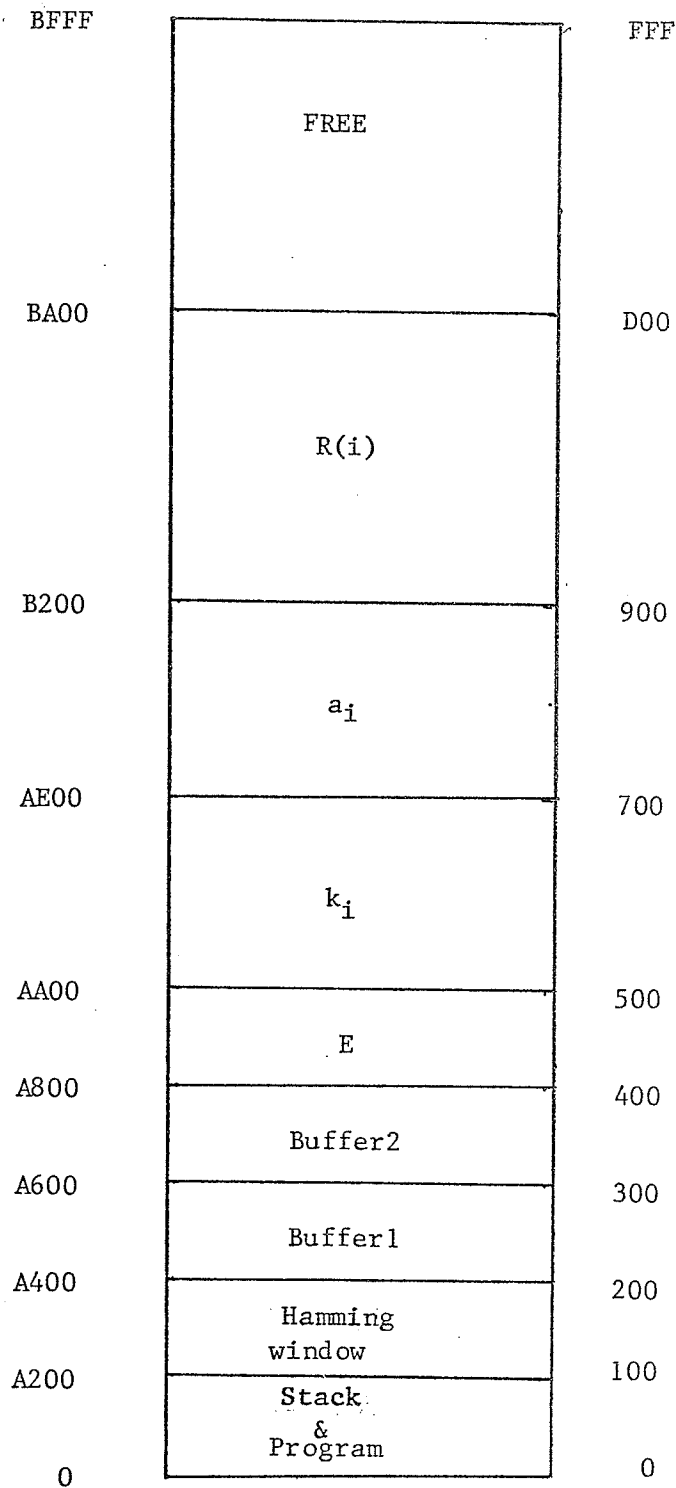


Figure 21: LPCMK Memory Map

double arithmetic shifts required to accomplish this is used to detect the start of word (SOW). If only two shifts are used the start of a word is assumed and FLAG bit 4 is set. SOW has an execution time of 1.4 usec.

MOVFL uses the present index of the error signal storage ERRX to determine whether a memory overflow will occur after the next data frame is processed. The present error index is compared to its expected value after 45 frames have been processed. When 45 frames have been processed, FLAG bit 1 is set and the status lights are sequentially lit to indicate the overflow condition. A limit of $45 \times 20 \text{ msec} = 0.9 \text{ sec}$ is set for the utterance length when all the data shown in Fig. 23 are saved. This can be increased to about 10 seconds if only the LPC coefficients are desired.

EOW uses the frame energy $R(0)$ to detect silent frames. When two silent frames are detected consecutively (40 msec) the end of word is assumed and FLAG bit 3 is set. The limit of two frames is necessary to account for silent periods within an utterance. The EOW execution time is 2 usec.

Program status is also monitored externally by three LEDs (Light Emitting Diodes) located on the processor board. They are labelled WAIT, RUN and STOP. A system reset lights only the WAIT LED indicating that the system is ready for an utterance to start. STAT1 lights RUN while extinguishing WAIT and STAT2 lights STOP, indicating the end of execution. The circuit schematic for external status is shown in Appendix 1.

KENDAL calculates the autocorrelation values and DURBIN calculates the predictor coefficients, reflection coefficients and framing error.

These are discussed in detail in Sec. 6.2.3. KENDAL execution time is 10.4 msec and DURBIN takes 1.5 msec for execution. These times were measured.

6.2 MICROPROGRAM SOFTWARE

6.2.1 Development

It was mentioned in Sec. 3.3 that a definition and assembly phase are required for the development of a microprogram with the Mace development system.

During the definition phase the bit patterns for the microinstruction mnemonics are assigned. The definition file LPCDEF for the LPCMK is listed in Appendix 2. These mnemonics are chosen to give maximum flexibility in the microprogramming. Each definition contains all 48-bits of the microinstruction. Each bit or group of bits (field) is set to one of the following values:

- a) Don't care bit;
- b) A specified pattern;
- c) A variable value defined in the assembly phase; or
- d) A variable value defined in the assembly phase with a default value.

After the definition phase, the microinstruction is built by overlaying the mnemonics which have been defined. The only restriction is that the assigned or defaulted fields are not defined twice in the same microinstruction. In other words, a defined field in a mnemonic can be overlain only on the don't care field of another. Any number of mnemonics can be used to define one microinstruction so long as the fields do

not conflict. After this, the definition and program files are assembled. The definition table created is referred to during microprogram assembly.

With use of the definition file, mnemonics are used to build microinstructions rather than bit patterns. Another major advantage to this approach is in the hardware changes that are made to the system after the microprograms have been written. For example, the I/O procedure was changed several times during the LPCMK development. The microprograms already written required only minor changes to accommodate this since the definition file could be changed and the microprograms reassembled with the new definition file.

6.2.2 Instruction/Microprogram Interface

All instructions are located in the program memory. Addresses to the mapping PROM are provided by the instructions. The FETCH microroutine is used to read the instruction indicated by the program counter PROG and increment PROG to set up the next instruction. If the instruction is longer than one word, maintenance of the program counter is handled by the microroutine involved.

After instruction execution is complete, an unconditional jump to FETCH acquires the next address.

6.2.3 Data Acquisition

The data acquisition unit issues an interrupt request $\overline{\text{IRQ}}$ to the LPCMK when data is ready for transmission. A conversion starts every 100 usec and takes 64 usec. The speech data is valid only for 36 usec

while $\overline{\text{IRQ}}$ is active and must be read in this interval or it will be lost.

A polling method is used to meet this requirement. It is unacceptable to poll the $\overline{\text{IRQ}}$ signal at every microinstruction because current microprogram status and the contents of the MAR, DOUT, and DIN registers cannot be saved. The 36 usec requirement cannot be met by polling after every instruction because of the long execution times of some special instructions. The $\overline{\text{IRQ}}$ is tested at points in the special instructions that are separated by not more than 30 usec in execution time. This guarantees that no data is lost. Polling is implemented by a conditional jump to the data acquisition microroutine ACQUIRE if $\overline{\text{IRQ}}$ is active (low). The polling always takes place when there is no useful information in the MAR, DIN or DOUT registers and the internal status is not required.

The flow chart for the ACQUIRE microroutine is shown in Fig. 24. Registers which are not dedicated to ACQUIRE and are used during the routine are stacked in memory and unstacked after completion. After this, data is read from the data acquisition routine and converted to 16 bit 2's complement form. The Hamming window is then used to multiply the signal. After every sample, the appropriate register HAMLOC, which points to the 200 point 16 bit Hamming window lookup table, is incremented. The register HAMLOC is dedicated to the current Hamming window location. The windowed value is stored in the appropriate program memory buffer which is referred to by DACBUF, another register dedicated to ACQUIRE.

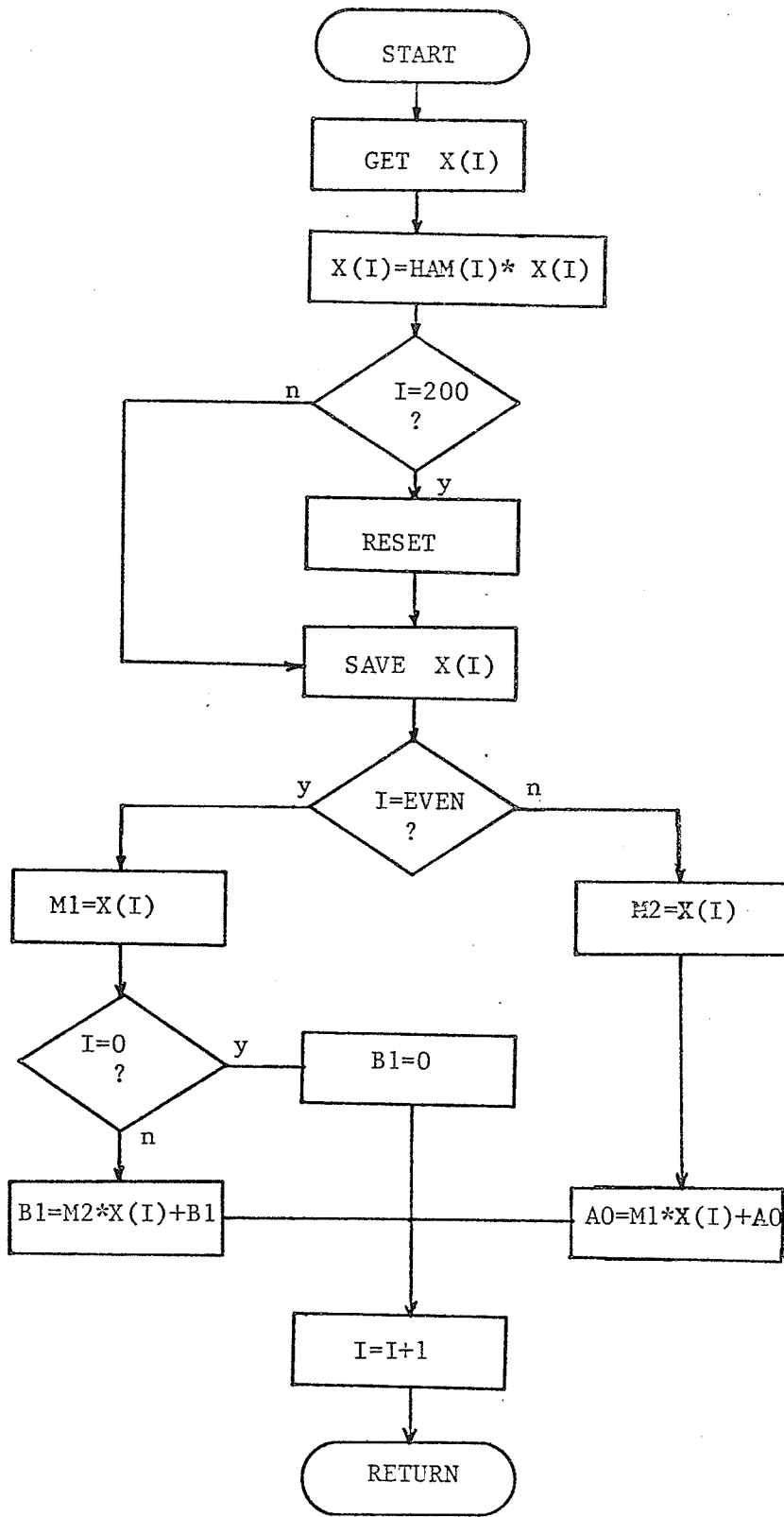


Figure 24: Data Acquisition Microroutine Flowchart

While data is in the LPCMK the initial values for A0 and B1 are calculated and saved in the program memory. These values are described in Eq. (42) and Eq. (47), respectively, and are used during solution of the autocorrelation matrix in Kendall's algorithm.

After an entire frame of data has been received, windowed, and saved, the starting address for the next buffer to be filled is loaded into the DACBUF register. The starting address of the frame just filled is loaded into the LPP register for use by KENDAL. Bit 0 of the FLAG register is complemented to indicate to the main program that a frame is ready and the other buffer is being filled. Also, the completed A0 and B1 are moved in memory for use by KENDAL and their former memory locations are cleared.

The measured execution time for ACQUIRE is 22 usec. The ACQUIRE micro-routine is called 200 times per frame for a total of 4.4 msec execution time per frame.

6.2.4 Major Microprogram Routines

The instructions of most importance to the LPC of speech are MULT, DIVIDE, KENDAL and DURBIN. These are described in detail with emphasis on computational efficiency. The flow charts are shown in Appendix 2.

Booth's algorithm was chosen for the multiplication routine since a 2's complement multiplicand and multiplier are used. On entry the multiplicand must be in the MCND register and the multiplier in the QREG. The product is 32 bits and occupies the ACCA,QREG register pair on exit. Since the number representation is assumed fractional, the final shift required by Booth's algorithm is not performed. Otherwise, an extra

sign bit would occupy the most significant position giving a result $1/2$ the actual value.

The three entry points provided into this microroutine are MULT, SQUARE and MULTAC. MULT is a normal 16 by 16 bit multiplication and SQUARE will square the value in the QREG. MULTAC is used for higher accuracy in ACQUIRE. If done with MULT the highest accuracy of Hamming-windowed value would be 8 bits, since only 8 bits are available from the data acquisition A/D converter. Fewer shifts are performed in MULTAC giving up to 15 bits accuracy in the ACCA result. This follows because the MSW of the product is used as the windowed speech sample. The speech value must be the multiplier QREG for this to be valid.

With a higher accuracy in the windowed sample the autocorrelation calculation becomes more accurate. Nine shifts are used to prevent overflow during KENDAL and give as high an accuracy as possible. A more desirable approach is the use of adaptive scaling to get the most accuracy possible for all signals. The execution time for MULT is approximately 7 usec and that for MULTAC about 4 usec showing that execution time actually improves if more accuracy is given in the most significant word of the product.

DIVIDE is a nonrestoring two quadrant division, allowing a positive or negative 32 bit 2's complement dividend in the MCND,QREG register pair. The divisor must be positive 2's complement and in ACCB on entry. The quotient is shifted into the QREG during the execution time of 7 usec. This division is used only in Durbin's algorithm.

Kendall's algorithm is used to evaluate the autocorrelation matrix for reasons discussed in Sec. 3.2. This is implemented in KENDAL which

has a 10.4 msec execution time. The initial values A0 and B1 are obtained from ACQUIRE.

The execution of KENDAL is separated into an even and odd section. An important concern in implementing any numerical algorithm is that of overflow. Since a fractional number representation is chosen the range is restricted to $-1 \leq n < 1$, where n is the number. For KENDAL, this problem is avoided by adjusting the number of shifts in MULTAC used in ACQUIRE. When the amplitude range of the signal to be correlated is sufficiently restricted, an overflow is prevented. Only the zero-th lag R(0) needs to be checked for overflow since it will be the largest autocorrelation value, as stated in Sec. 3.2.

The starting address of the buffer containing the speech data being autocorrelated is in the LPP register. This value is assigned by ACQUIRE at the end of a data frame acquisition.

After the first autocorrelation value is calculated, R(0) is shifted left until a 1 occupies the 2nd MSB position, normalizing R(0) to a value greater than or equal to 0.5. The other values are shifted the same amount, adjusting all autocorrelation values for use in Durbin's algorithm. The number of shifts during normalization is an indication of the energy in the original frame, is saved in the program memory for use in the EOW and SOW instructions. An overflow in R(0) will result in the external status lights being sequentially flashed as described in Sec. 6.1.3.

Since 10 predictor coefficients are required there will be 11 autocorrelation values calculated R(0), R(1)...R(10). These 32 bit values are saved in locations calculated from the index saved in the memory lo-

cation AUTOX. For each frame, 22 memory locations are required to save the autocorrelation values. AUTOX is not updated at this time as it is used in DURBIN.

Durbin's algorithm yields the predictor coefficients, reflection coefficients and frame error for each frame of data. The algorithm is implemented with DURBIN. This instruction uses the autocorrelation matrix starting at the AUTOX memory index location. The algorithm is described in Sec. 3.3 and, as in KENDAL, special attention must be paid to the problem of overflow.

During the recursion in DURBIN, the predictor coefficient may become greater than one. An adaptive scaling approach is taken where the scaler starts off being zero and is incremented every time there is an overflow of the predictor coefficient. When this occurs, DURBIN is started again and all predictors are shifted by the scaler amount. The time penalty of restarting DURBIN is not large since all overflows were found to occur in the first or second passes through the recursion formulae. The predictor values were found to be shifted down at most two times. This adaptive method is useful for retaining predictor accuracy.

After each pass is made the reflection coefficient k is saved at the location defined by the previous number of passes plus the starting address at memory location REFLDEX. The frame error E is saved after DURBIN is complete at the address stored in the ERRX memory location. During calculation, the intermediate predictor results are saved in the memory locations starting at the address contained in LPCDEX. During the final pass of the recursion, the values placed here are the desired predictor coefficients.

After Durbin's algorithm is complete, the ERRX, LPCDEX, AUTOX, and REFLDEX indexes are updated for use in the next frame. These are located in the program memory and are initialized with INIT. Execution time for DURBIN is measured as approximately 1.5 msec.

6.3 MAIN PROGRAM

The flowchart for the main LPCMK program is given in Fig. 25. The program listing is in Appendix 2 in addition to the Hamming window look-up table.

The execution time for processing one frame of speech data to get the autocorrelation matrix, reflection and predictor coefficients, and frame error is approximately 16.3 msec. The requirement for real time operation of less than 20 msec is met.

The entire program, not including the Hamming window look-up table occupies 16 words of program memory. This is only possible because of the specially defined instruction set. The program listing is given in Table 10.

TABLE 10
LPCMK Real Time LPC Generation Program

OPCODE	LABEL	MNEMONIC	OPERAND
09	LPC	INIT	
12	START	FILL	
14		KENDAL	
13		SOW	
0B 02		BRF	START
18		STAT1	
15	START1	DURBIN	
16		MOVFL	
12		FILL	
14		KENDAL	
17		EOW	
0C 07		BRC	START1
19		STAT2	
0D		STOP	

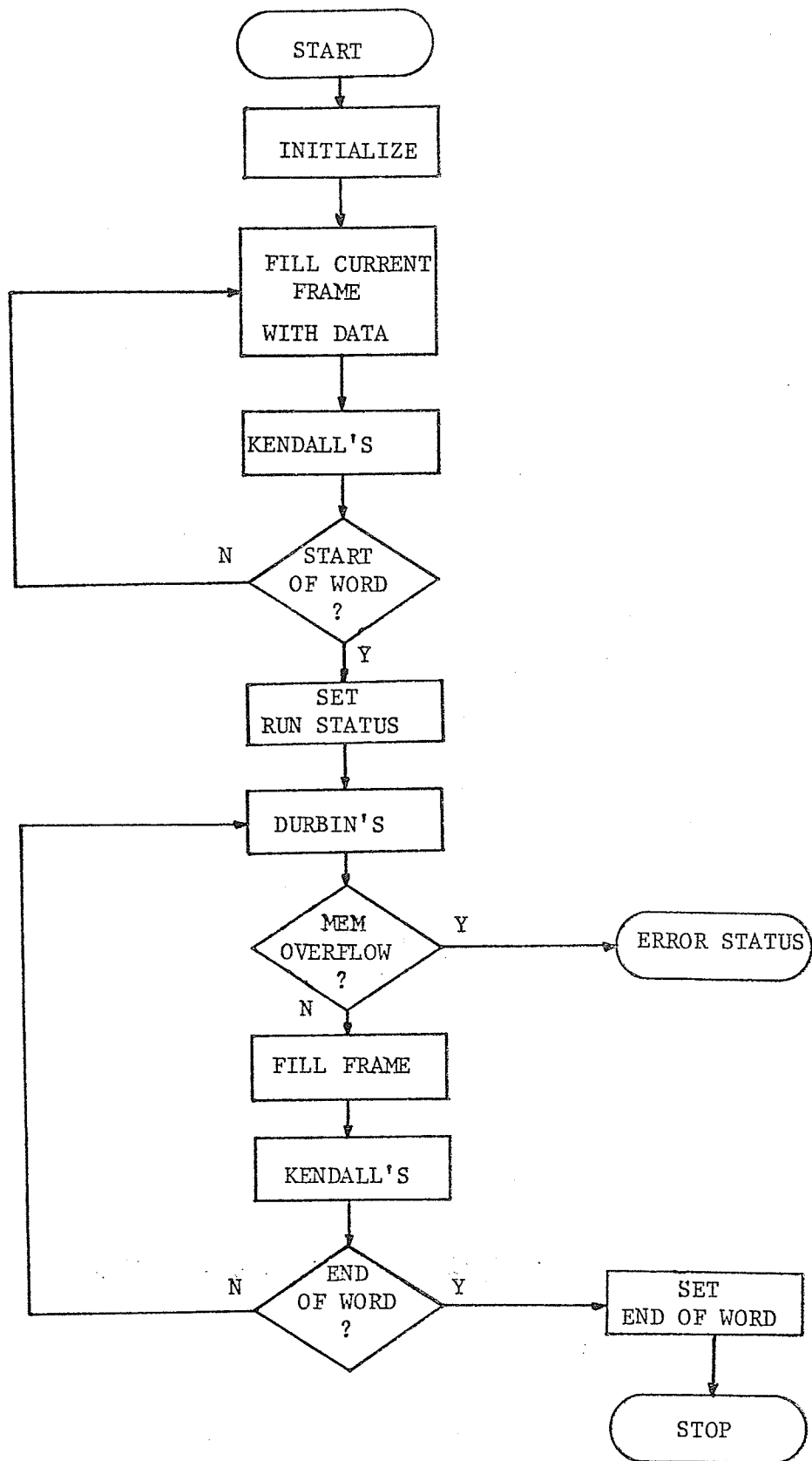


Figure 25: Main Program Flowchart

Chapter VII

RESULTS

Three frames of speech data are tested with the LPCMK system, one voiced and two unvoiced. The LPCMK results are compared to those obtained by simulation on an AMDAHL mainframe computer. Both sets of results are truncated. The LPCMK uses 16 bit fixed-point arithmetic while the AMDAHL uses 32 bit floating point arithmetic in its calculations. Therefore results obtained by the main frame AMDAHL computer are taken as the reference.

The autocorrelation coefficients, reflection coefficients, predictor coefficients and frame error are calculated for each of the three speech sounds. The results are presented in Table 11 in the form of truncated 16 bit 2's complement numbers.

All sets of predictor coefficients were found to yield stable filters according to the stability test found in [11].

TABLE 11

LPCMK Results

PARA- METER	VOICED		UNVOICED			
	'E-E-E-E'		'F-F-F-F'		'H-H-H-H'	
	AMDAHL	LPCMK	AMDAHL	LPCMK	AMDAHL	LPCMK
AUTOCORRELATION VALUES						
		2-shifts		2-shifts		5-shifts
R(0)	64BE	64BD	62CE	62CE	7FE2	7FE5
R(1)	487C	487E	3B90	3B8F	6950	6957
R(2)	22FE	22FD	F060	F05F	3D88	3D86
R(3)	27D8	27DC	C3D6	C3D6	180E	180E
R(4)	3960	395F	D392	D392	0B7B	0B7B
R(5)	2C74	2C77	09E8	09E7	2326	2324
R(6)	0FEA	0FEA	36CA	36C9	494E	4950
R(7)	01C8	01C6	36E0	36DF	63F4	63F9
R(8)	00B8	00B8	101C	101B	6712	6714
R(9)	FC82	FC82	EAEA	EAEA	50CC	50D3
R(10)	EF14	EF15	E618	E617	3322	3320
REFLECTION COEFFICIENTS						
k ₁	5C1A	5C1D	4D2A	4D29	696A	696F
k ₂	D2CE	D2BB	9728	9723	B1A4	B16F
k ₃	5E52	5EAD	238E	23BD	21D8	22AB
k ₄	CCF0	CB4B	0794	0759	25A0	252D
k ₅	1234	15AF	1B00	1B01	5772	581D
k ₆	D964	D49D	124E	125B	FA24	F841
k ₇	D88A	DD41	03CC	03D1	1896	1C15
k ₈	0830	075F	0ECC	0F13	13C8	1235
k ₉	E080	DF03	2260	22AF	1DC2	2185
k ₁₀	F192	F5FF	0688	0671	110C	0F8F

PREDICTOR COEFFICIENTS							
a ₁	3162	32BF	4DA8	4DAD	49EC	4AFD	
a ₂	C8B0	C595	BD94	BD4F	D500	D179	
a ₃	306A	34F5	1264	1281	14D4	1914	
a ₄	F1DE	EC1E	FB5C	FB4B	CD94	C918	
a ₅	F9C8	FD11	01F8	01DE	33C8	3896	
a ₆	1402	12FB	08C0	08CD	F784	F28A	
a ₇	EB92	EA20	0A30	0A41	0740	0B94	
a ₈	07E4	0A4E	F538	F517	FD98	FA23	
a ₉	FDCC	FBC4	0D30	0D63	04C8	0765	
a ₁₀	FC64	FD7F	0344	0338	0888	07C7	
FRAME ERROR							
E	031A	02F0	0834	0819	0538	04E1	

7.1 AUTOCORRELATION VALUES

The voiced 'E' and fricative 'F' autocorrelation values agree with the AMDAHL results to 15 bits in every case except for R(3) of the 'E' sound. The LPCMK results for the aspirated 'H' sound give autocorrelation results with only about 12 bits of accuracy. This difference in accuracy is explained by looking at the number of arithmetic shifts required in the normalization. Two shifts are used on the 'E' and 'F' sound autocorrelation values and five with the 'H' sound.

Adaptive scaling would avoid this inaccuracy by retaining more bits of information in the MULTAC routine. This would entail more computation time but, as shown earlier, 3.6 msec per frame are left after the present calculations are made.

The relationship between amplitudes of the successive autocorrelation values for each of the frames yields important information about each signal. As stated in Chapt. 2, it can be seen that the point with the largest magnitude is $R(0)$. The fricative 'F' has autocorrelation values that decrease rapidly and remain low in amplitude for lags greater than 0. The voiced 'E' and aspirated 'H' have a quickly decreasing amplitude after $R(0)$ and for the 'H' sound some autocorrelation values are almost as large as $R(0)$ after a minimum at $R(3)$. This agrees with the appearance of the respective signals when observed on an oscilloscope. The voiced 'E' and aspirated 'H' waveforms are much smoother than the fricative waveform, which has a very noisy appearance.

7.2 REFLECTION COEFFICIENTS

The error in the reflection coefficient calculations for the three examples after each successive recursion of Durbin's algorithm accumulates due to finite register length effects. The first coefficient has 15 bits of accuracy for the voiced 'E' and fricative frames. There are only 13 bits of accuracy for the first reflection coefficient of the unvoiced 'H' sound. The reflection coefficient accuracy for this sound is consistently worse because of the lower autocorrelation accuracy.

The final reflection coefficient value for the voiced 'E' has only 5 bits of accuracy and the fricative 'F' is accurate to 11 bits. The aspirated 'H' sound reflection coefficient accuracy, as expected, is much lower, only 3 bits.

7.3 PREDICTOR COEFFICIENTS

The prediction coefficients are re-evaluated during every recursion of Durbin's algorithm. Because of accumulated error the reflection coefficients are less accurate than are the reflection coefficients. Seven bit accuracy is observed for the first predictor coefficient of the voiced 'E' and the unvoiced 'H'; while the fricative 'F' has 13 bits of accuracy. The tenth predictor value has 7 bit accuracy for the voiced 'E' and unvoiced 'H' and an 11 bit accuracy for the fricative 'F'.

The fricative predictor coefficients were calculated with more accuracy than those of the voiced 'E' and aspirated 'H'. This is explained by examining Durbin's algorithm. The first reflection coefficient is defined as $k_1 = R(1)/R(0)$ which is closer to 1 for the smooth frames. This value is used to evaluate the error $E = (1 - k_1^2) * E(0)$, where $E(0) = R(0)$ and E is the first calculated frame error. For the case where k_1 is close to 1, E is close to zero. Since E is used as the divisor to calculate the k_i in Durbin's algorithm and fixed point arithmetic is used, the quotient accuracy is smaller than for large E . The first E is larger in the voiced and aspirated frame giving better results for successive k_i and a_i .

7.4 FRAMING ERROR

The calculated frame error signal is accurate to 11 bits for the voiced 'E', 10 bits for the fricative 'F' and 10 bits for the aspirated 'H'. The unvoiced frames have the largest errors since less periodicity exists in the signal and prediction based on past values is less accurate.

Chapter VIII

CONCLUSIONS

A speech research facility has been developed that is capable of acquiring raw speech data and performing linear predictive coding on the data in real time. The real time approach is implemented with bit-slice components with instructions expressly designed to aid the real time programming of the LPCMK processor for LPC. One major advantage of real time calculation is the relatively modest memory requirements for obtaining predictor coefficients, reflection coefficients, autocorrelation values and frame errors for complete utterances, such as words.

The LPCMK is programmed to save all of these parameters. Therefore, with the program memory provided, 1 sec of speech is transformed.

Chapter IX

RECOMMENDATIONS

The LPCMK with its associated system is a fast, versatile research facility capable of real time digital signal processing on the speech signal. The following improvements are suggested for the system hardware to decrease the computation time and ease further microprogram development:

- a) Latch the speech data using the A/D converters' EOC signal to trigger the latch. This allows the researcher to poll the IRQ at 100 usec intervals instead of 36 usec intervals;
- b) Use the R/\bar{W} microinstruction bit to generate the DOUT register trigger and enable signal. This frees a microinstruction bit for use elsewhere;
- c) Implement a complementary, as well as standard, output from the status register for greater microprogram conditional branch control;
- d) The facility to save the contents of the status, MAR, DIN, and DOUT registers would improve the IRQ handling facility considerably. This is accomplished by providing additional registers to save the contents. The microprogram would then poll the IRQ at every microinstruction where a branch instruction does not exist;
- e) Program execution time is improved if the first word of every instruction is copied to an alternate register that supplies the

OPCODE to the mapping PROM. If additional operands are required from memory, then this instruction is not written over and lost;

f) The LPCMK system clock speed can be increased without affecting the A/D conversion if the converter clock is independent of the system clock. This requires changes in the I/O software, but the performance would be improved;

Software modifications that would allow the dynamic modification of some system parameters such as the number of predictor coefficients generated and the values which are actually saved in memory. It is desirable to introduce adaptive scaling to KENDAL so maximum accuracy is obtained for the windowed speech frame. This also assures that the accuracy for the parameters calculated in DURBIN does not vary from frame to frame.

The LPCMK system, including software can be used to study LPC of the speech signal. The effect on the quality of LPC with varying spectral pre-emphasis is interesting since it has been shown that a spectrally flattened signal will yield LPC coefficients with a smaller frame error.

Another research application the LPCMK can be readily adapted to is speech recognition. A method called Linear Programming is used with the predictor coefficients to perform speech recognition experiments on isolated words. The LPCMK system can be used for this purpose with additional microprograms.

In summary, the LPCMK is the heart of a speech research facility that can be used for the study of LPC and associated speech research topics. Some examples are speech recognition, speaker recognition, speech analysis and speech synthesis since a D/A is provided.

BIBLIOGRAPHY

1. Hofstetter, et al., 'Microprocessor Realization of a Linear Predictive Vocoder.' IEEE Transactions on Acoustics, Speech, and Signal Processing. Vol. ASSP-25, No. 5, Oct. 1977, pp.379-387,
2. Denes, Peter B. and Pinson, Elliot N., The Speech Chain: The Physics and Biology of Spoken Language, Anchor Books, Anchor Press/Doubleday Garden City, New York, 1973,
3. Flanagan, J.L., Speech Analysis, Synthesis, and Perception, Springer-Verlag, New York, 1972,
4. Makhoul, J. et al., 'A Mixed Source Model for Speech Compression and Synthesis', Acoustical Society of America, Dec. 1978, pp.1577-1581
5. Rabiner, L.R. AND SHAFER, R.W., Digital Processing of Speech Signals, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1978,
6. Mairtra, S. and Davis C., 'Speech Digitizer at 2400 Bits/Second', IEEE Acoustics Speech and Signal Processing, Vol ASSP-27, No.6, Dec. 1979, pp.729
7. Motorola Inc., Mace 29/800 Development System Users Guide
8. Mick, J. and Brick, J., Bit Slice Microprocessor Design McGraw-Hill Book Company, New York, 1980
9. Advanced Micro Devices, The Am2900 Family Data Book, 1979
10. National Semiconductor, IDM2900 Family Microprocessor Databook, Santa Clara California, 1980
11. Atal, B.S. and Hanauer, S. , 'Speech Analysis and Synthesis by Linear Prediction of the Speech Wave,' The Acoustical Society of America, vol. 50, pp. 637-655 (1971),
12. Winckel, F., Music, Sound, and Sensation: A Modern Exposition, Dover Publications, Inc., New York, 1967,
13. Oppenheim, A.V. and Schafer, R.W., Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975,
14. Rabiner, L.R., and Gold, B., Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975,


15. Chen, Chi-Tsong, One-Dimensional Digital Signal Processing, Marcel Dekker, Inc., New York and Bessel, 1979
16. Knudsen, M.J., 'Real-Time Linear-Predictive Coding of Speech on the SPS-41 Triple-Microprocessor Machine, 'IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. Assp-23, No. 1, Feb. 1975, pp.140-145.
17. Makhoul, J., 'Linear Prediction: A Tutorial Review,' Proceedings of the IEEE., VOL. 63, No. 4, April 1975, pp. 561-578, 1978
18. Zuch, L.E., Data Acquisition and Conversion Handbook, Data Intersil, 1980, pp.109
19. Mahieu, L., 'A Real Time Linear Predictive Speech Processor', MScEE Thesis, University of Manitoba, Winnipeg, Manitoba, 1981


Appendix A

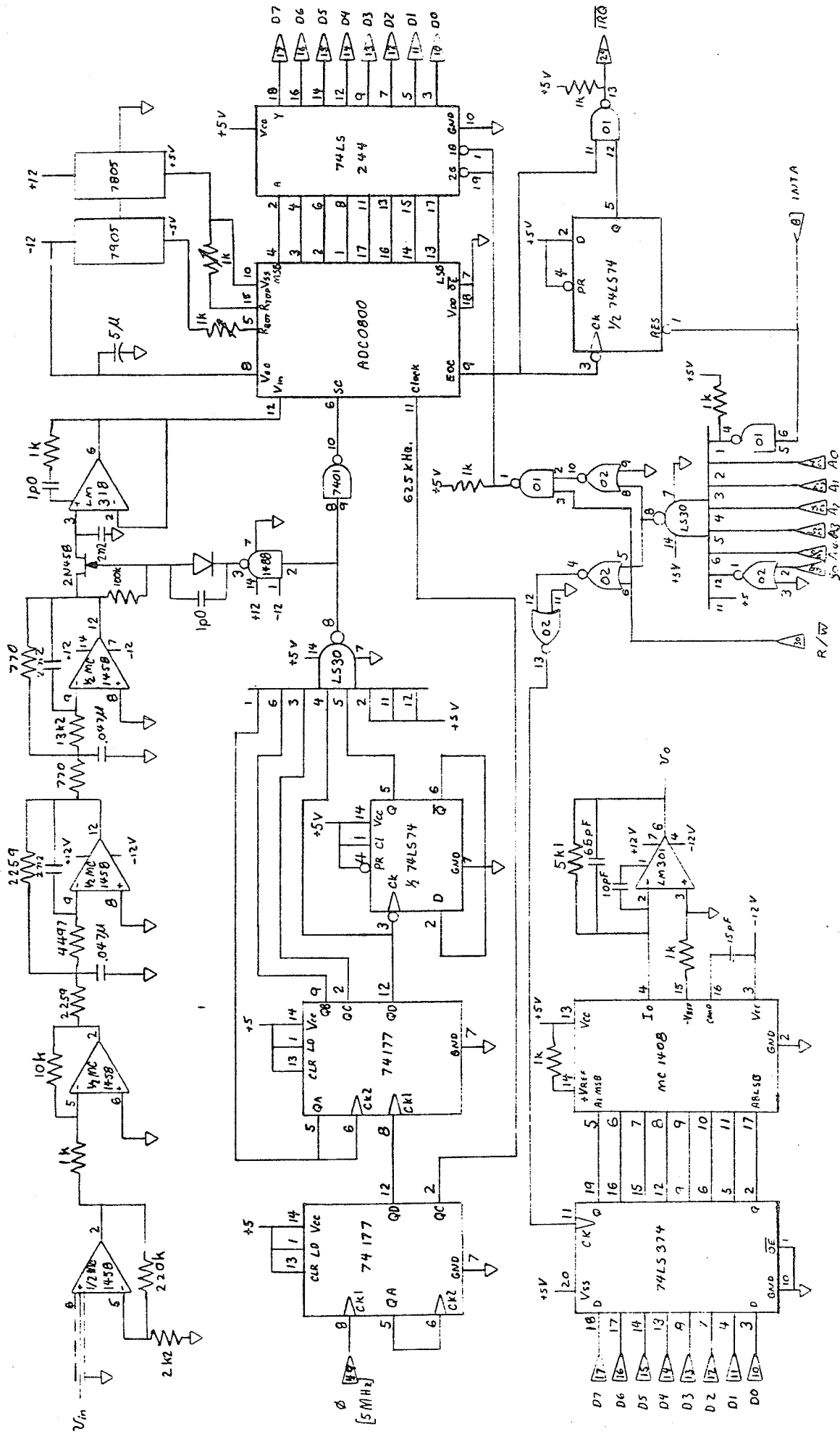
LPCM AND ASSOCIATED HARDWARE

LPCMK External Bus Assignment

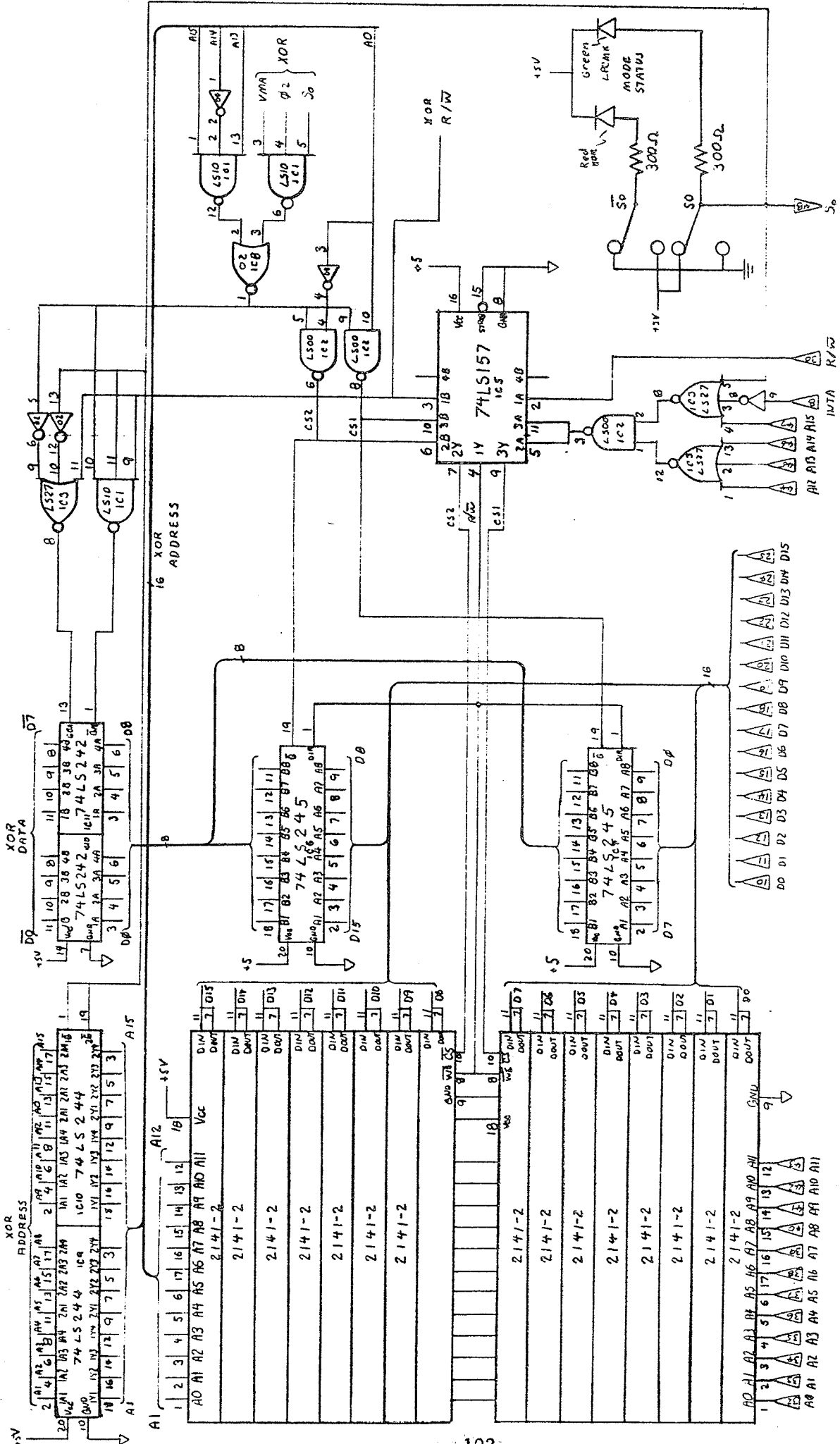
PIN FUNCTION	FUNCTION	PIN	FUNCTION	PIN	FUNCTION	PIN
1	+ 5V	51	+ 5V	26		76
2	+12V	52	-12V	27	GND	77
3		53	GND	28	GND	78
4		54	GND	29	$\overline{\text{IRQ}}$	79
5		55	GND	30	R/ $\overline{\text{W}}$	80
6		56	GND	31		81
7	$\overline{\text{INTA}}$	57	GND	32	A 0	82
8	$\overline{\text{RESET}}$	58	GND	33	A 1	83
9		59	GND	34	A 2	84
10	D 0	60	GND	35	A 3	85
11	D 1	61	GND	36	A 4	86
12	D 2	62	GND	37	A 5	87
13	D 3	63	GND	38	A 6	88
14	D 4	64	GND	39	A 7	89
15	D 5	65	GND	40	A 8	90
16	D 6	66	GND	41	A 9	91
17	D 7	67	GND	42	A10	92
18	D 8	68	GND	43	A11	93
19	D 9	69	GND	44	A12	94
20	D10	70	GND	45	A13	95
21	D11	71	GND	46	A14	96
22	D12	72	GND	47	A15	97
23	D13	73	GND	48	S0	98
24	D14	74	GND	49	∅	99
25	D15	75	GND	50	GND	100

 EXTERNAL BUS

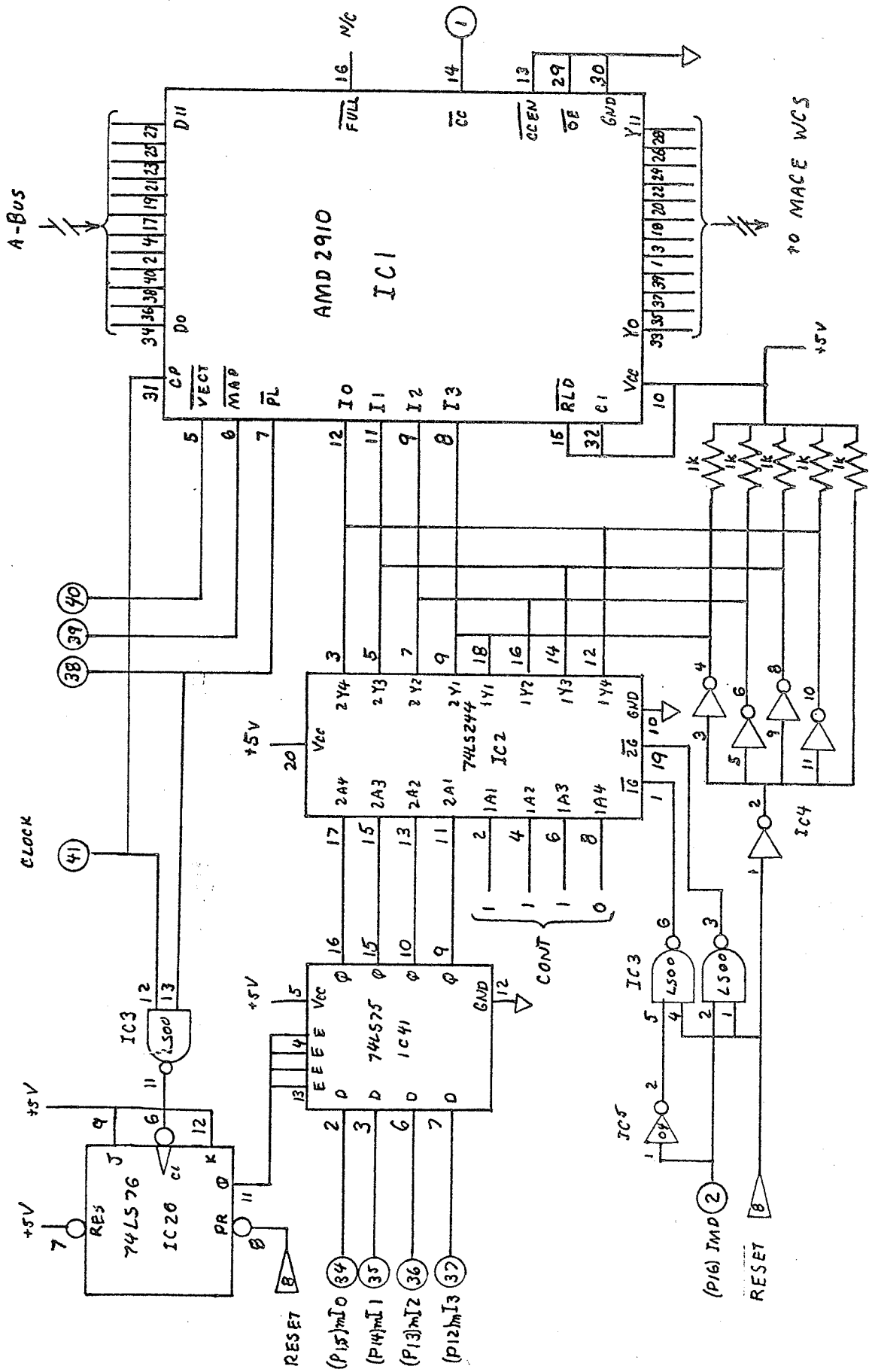
 INTERNAL LPCMK CONNECTION



DATA ACQUISITION CIRCUIT SCHEMATIC DIAGRAM

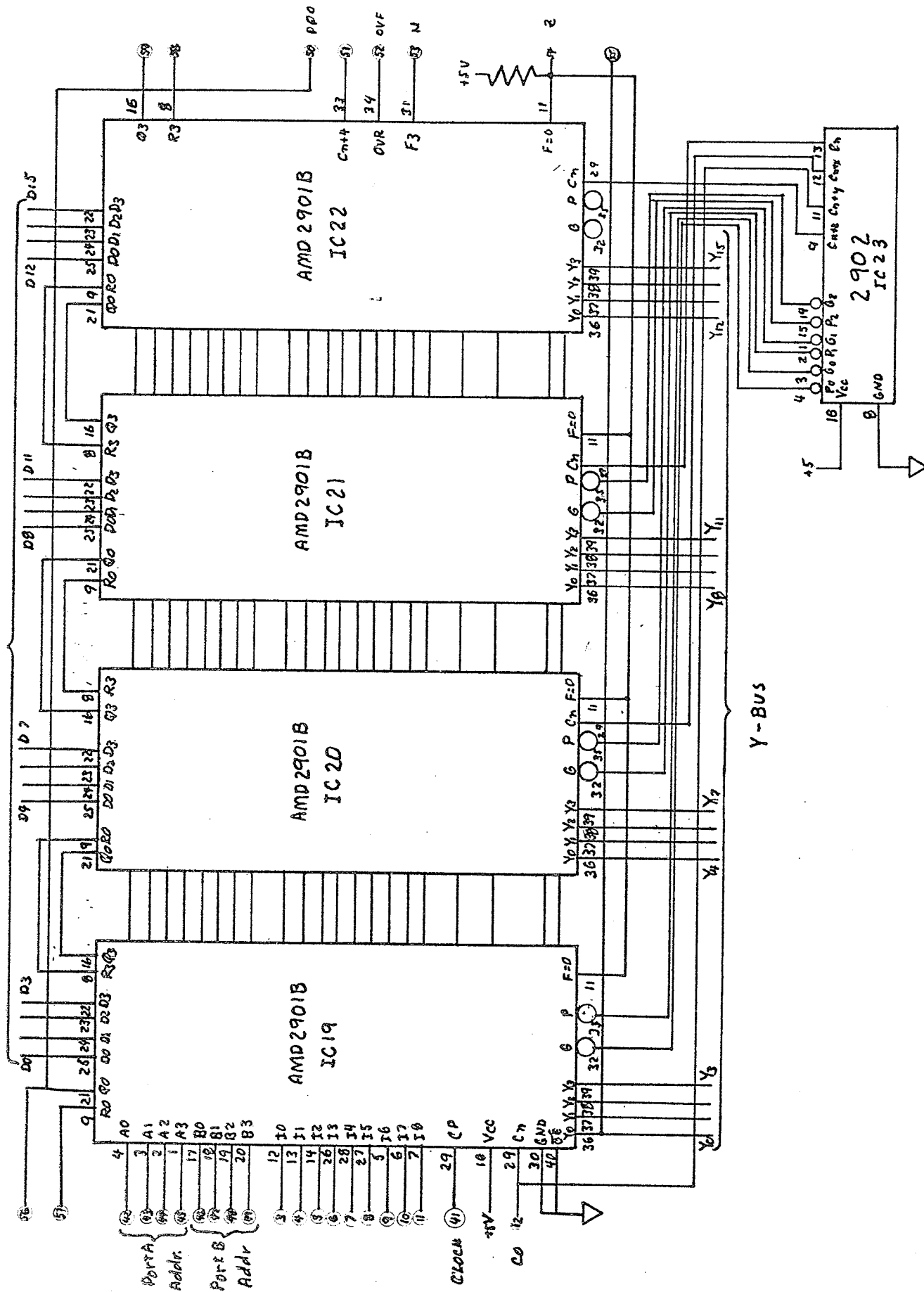


PROGRAM MEMORY CIRCUIT SCHEMATIC DIAGRAM



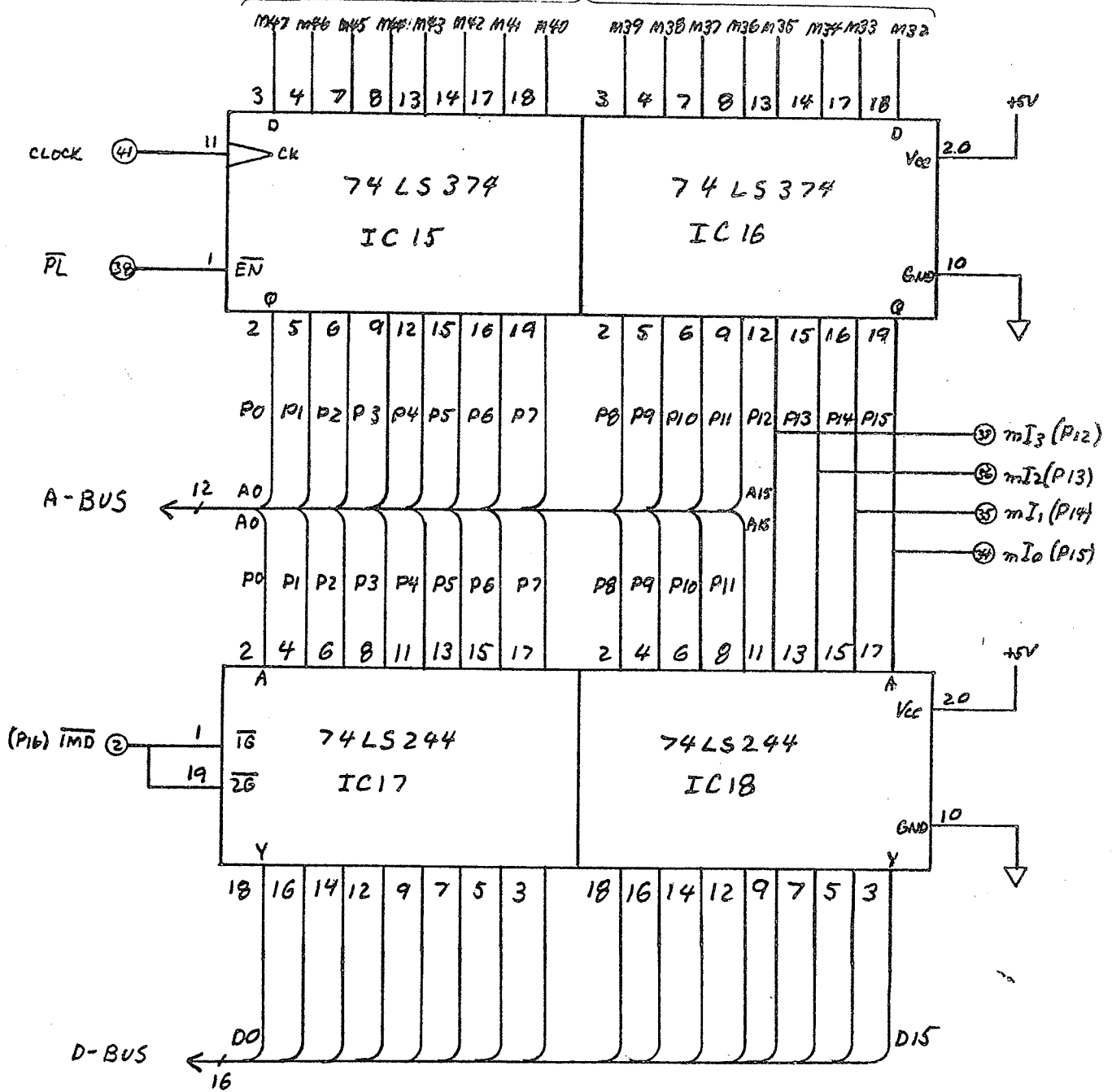
MICROINSTRUCTION SEQUENCER

D-BUS

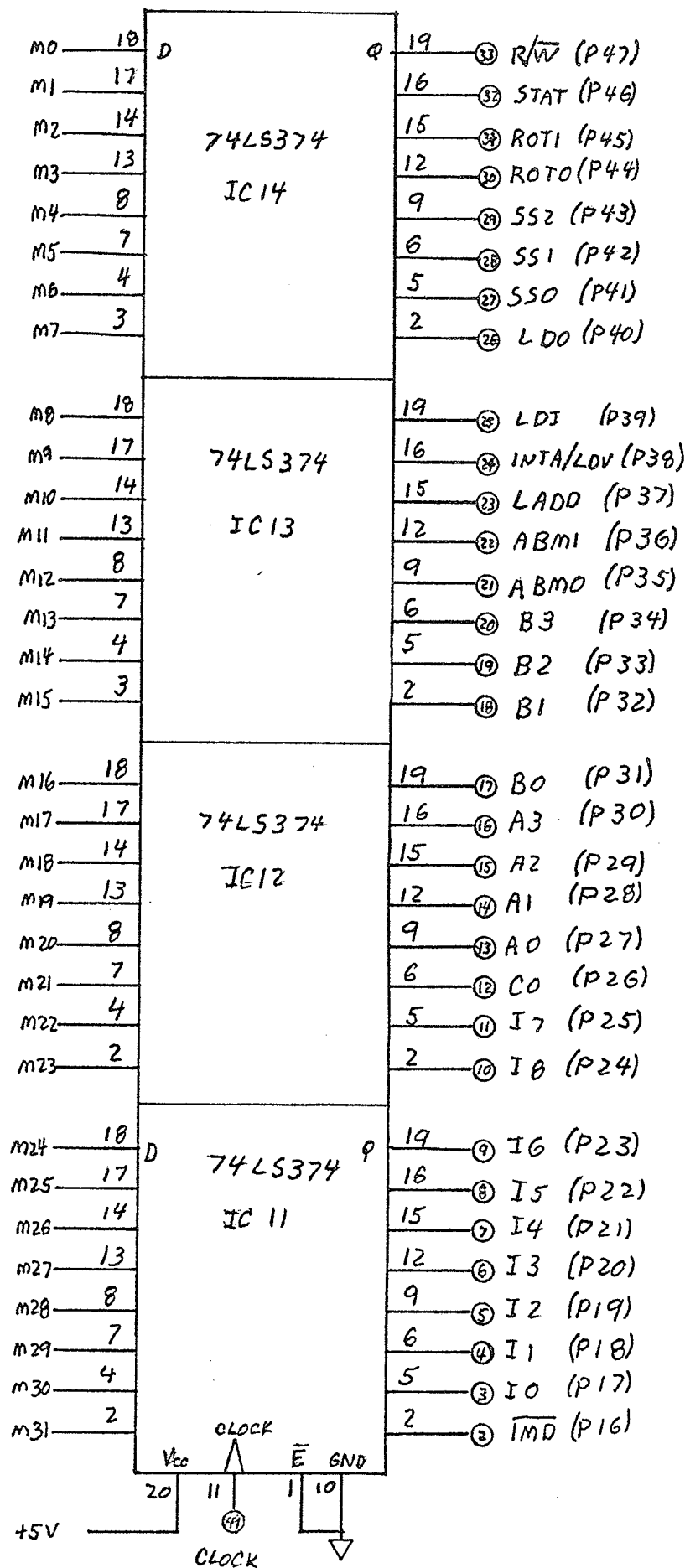


BIT SLICE REGISTER SET & ALU

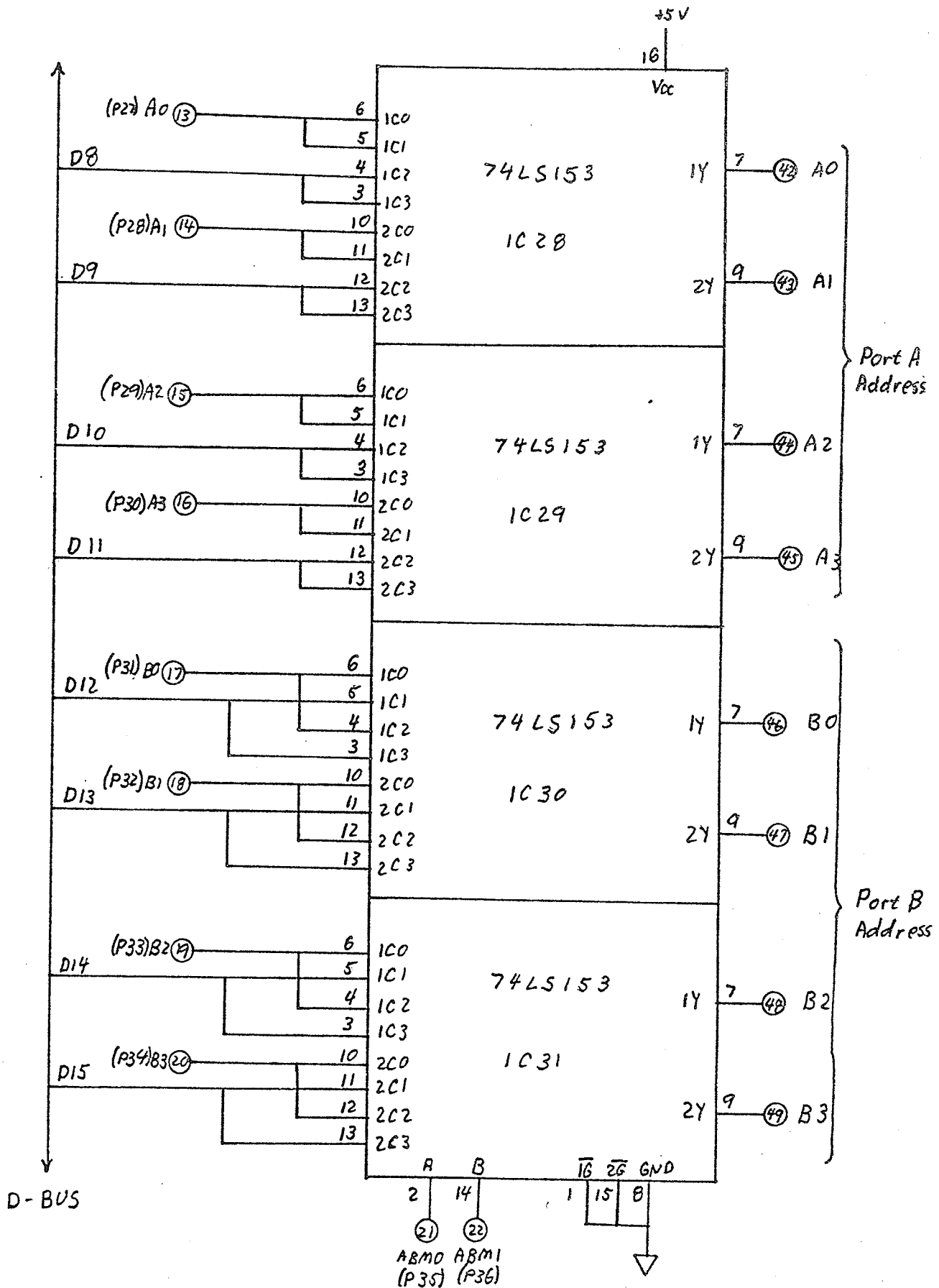
from MACE WCS



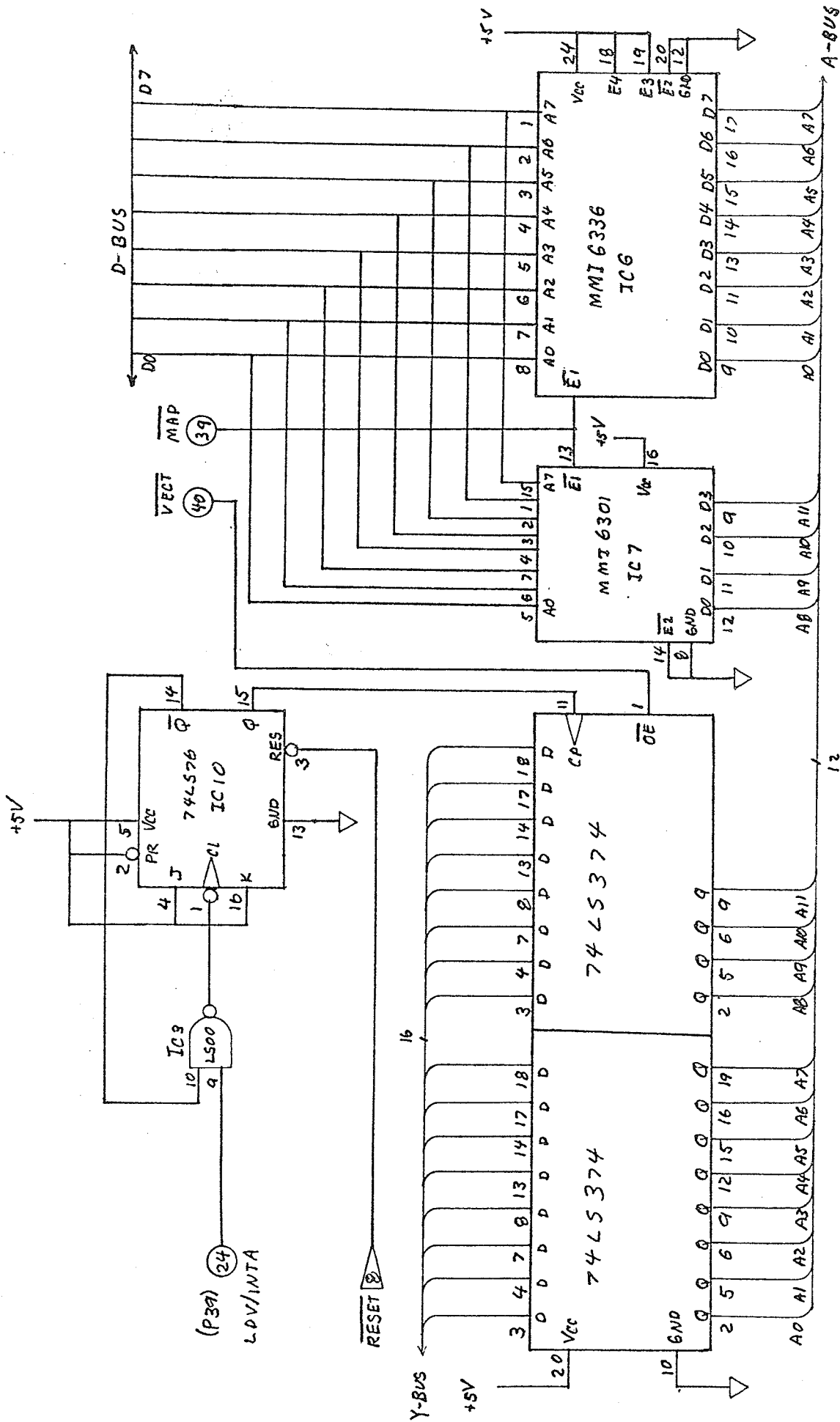
PIPELINE REGISTER/IMMEDIATE DATA



PIPELINE REGISTERS

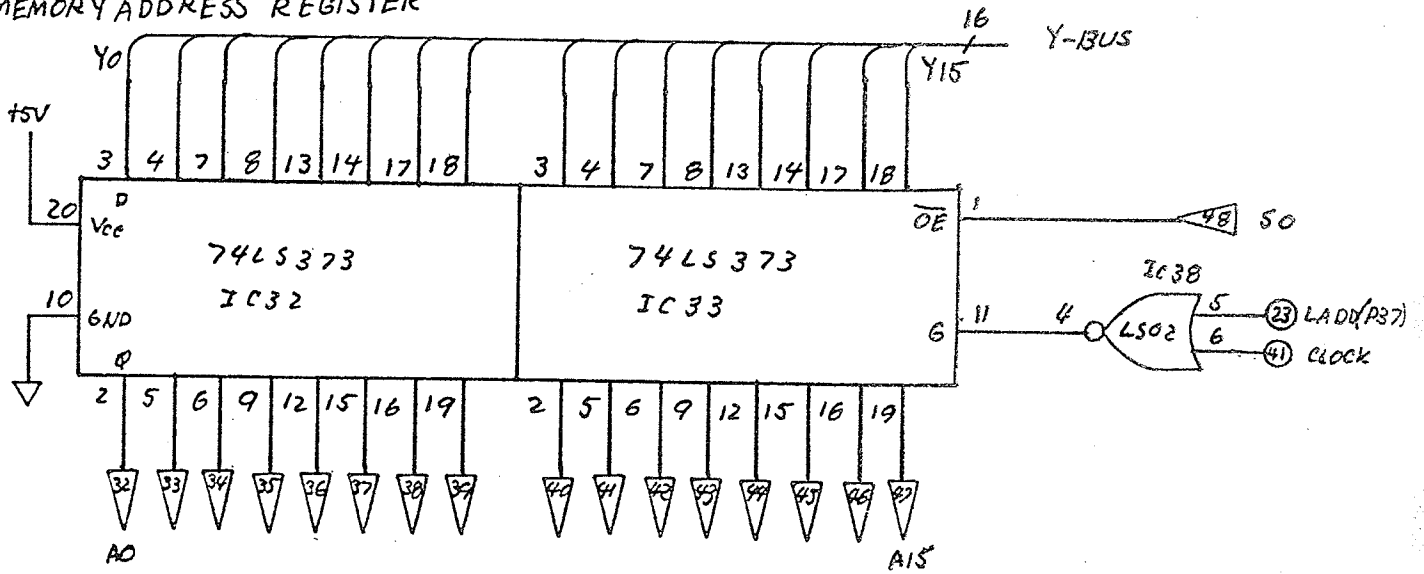


REGISTER ADDRESS MVX

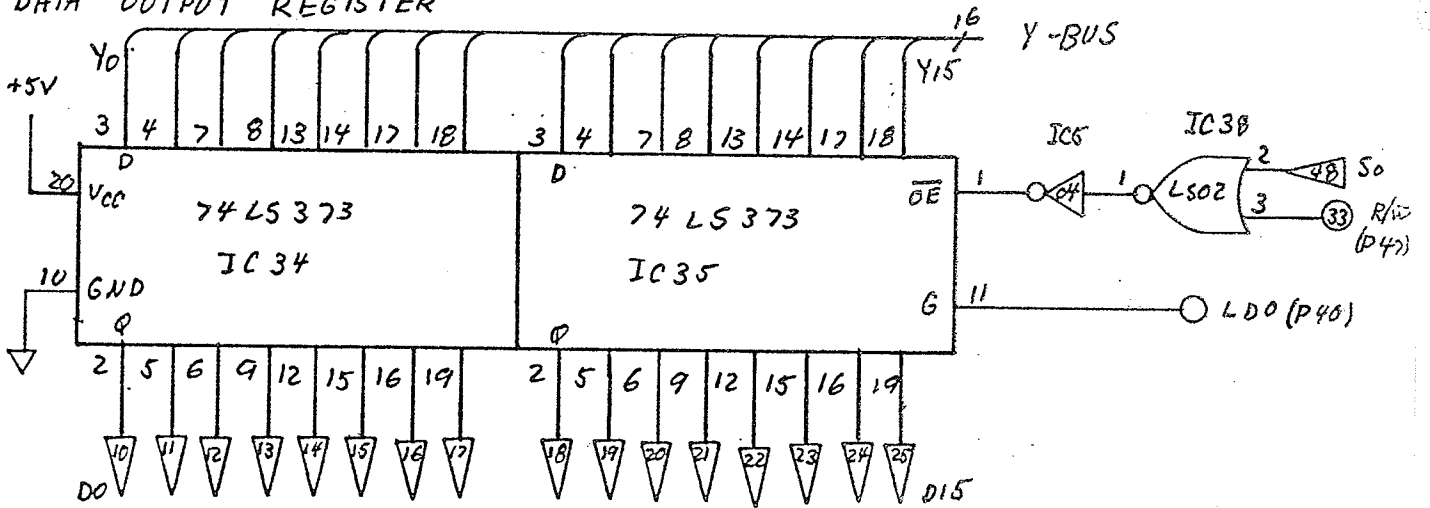


VECTOR ADDRESS AND MAPPING PROM

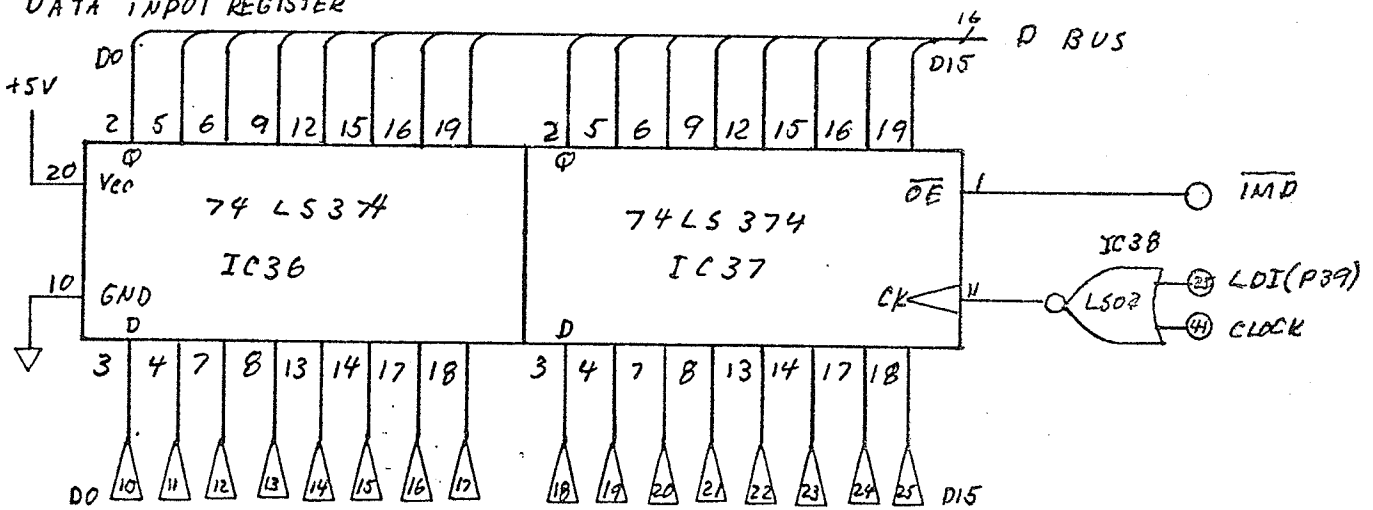
MEMORY ADDRESS REGISTER



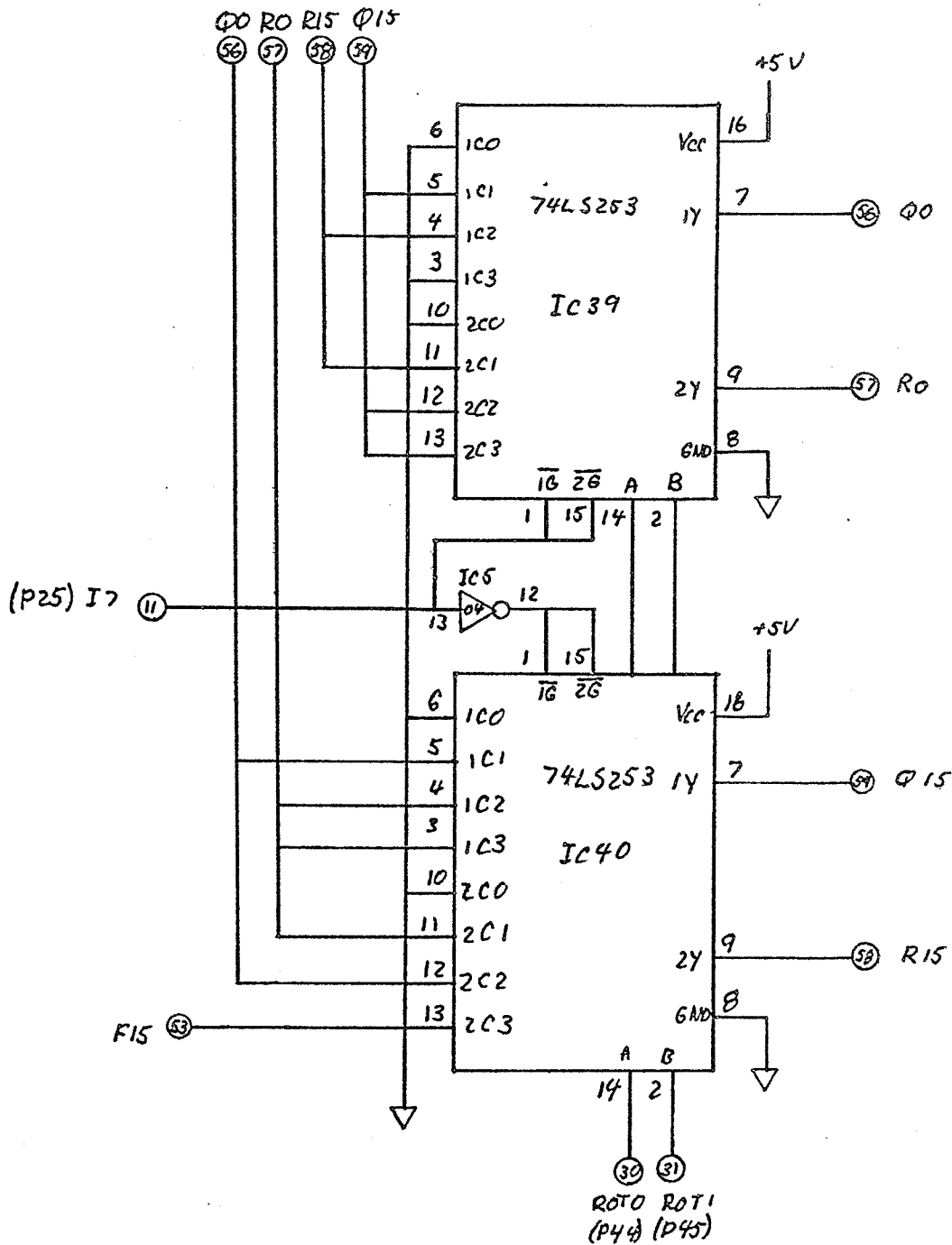
DATA OUTPUT REGISTER



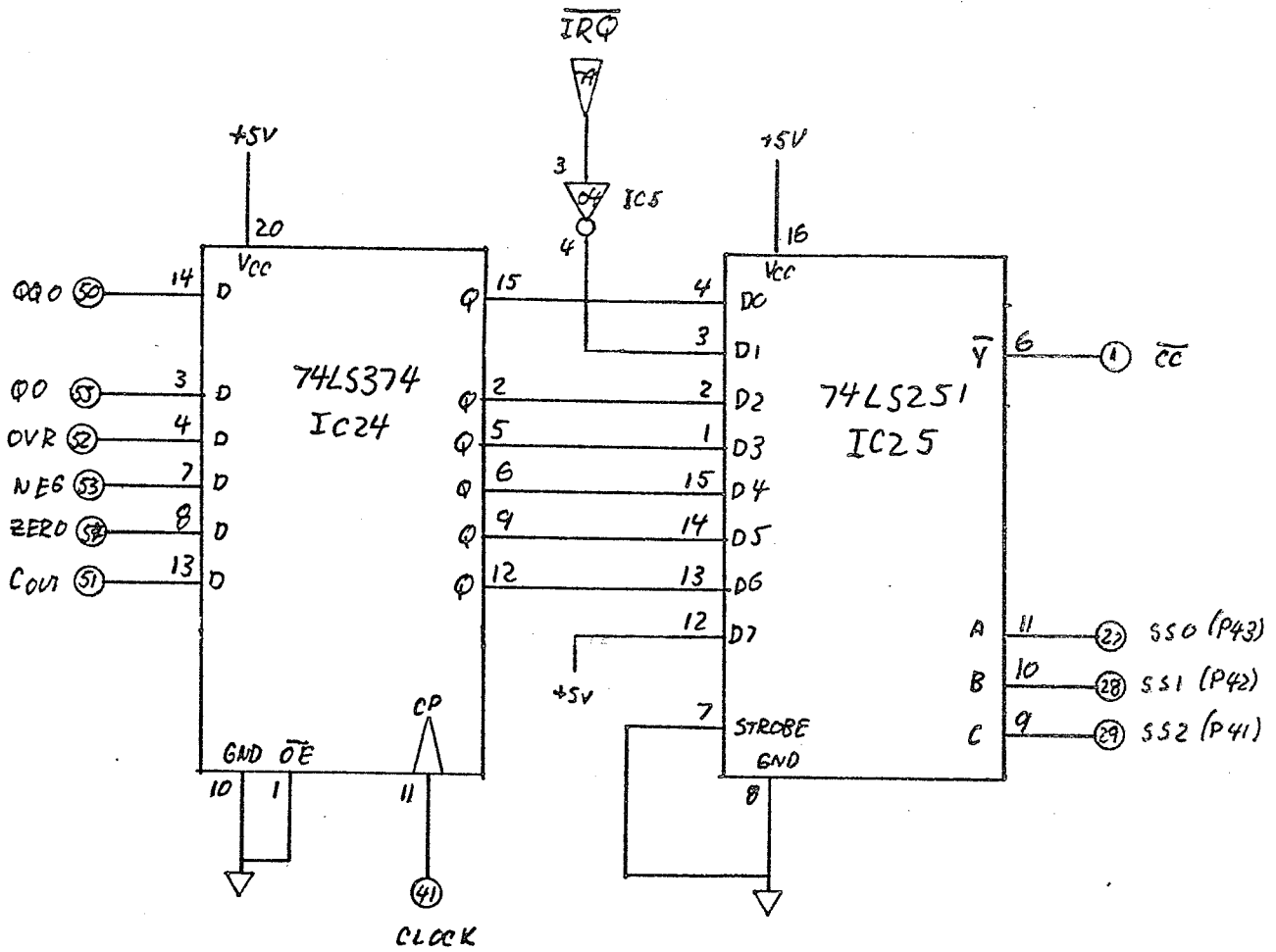
DATA INPUT REGISTER



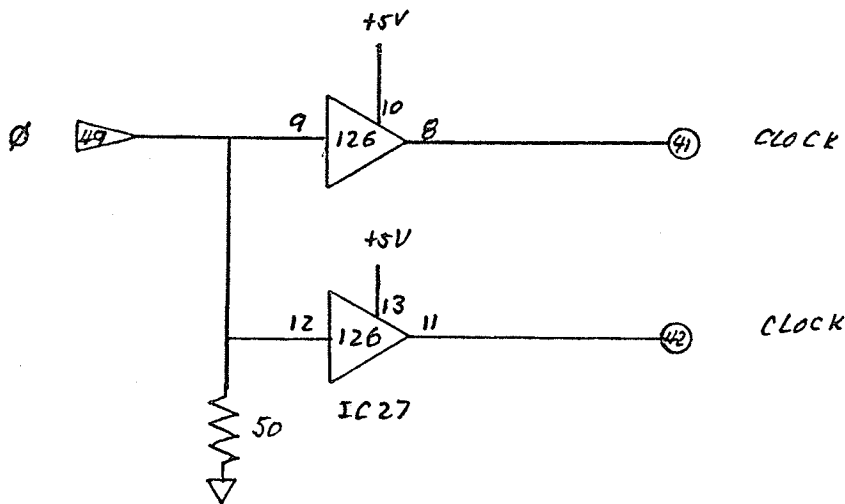
I/O REGISTERS



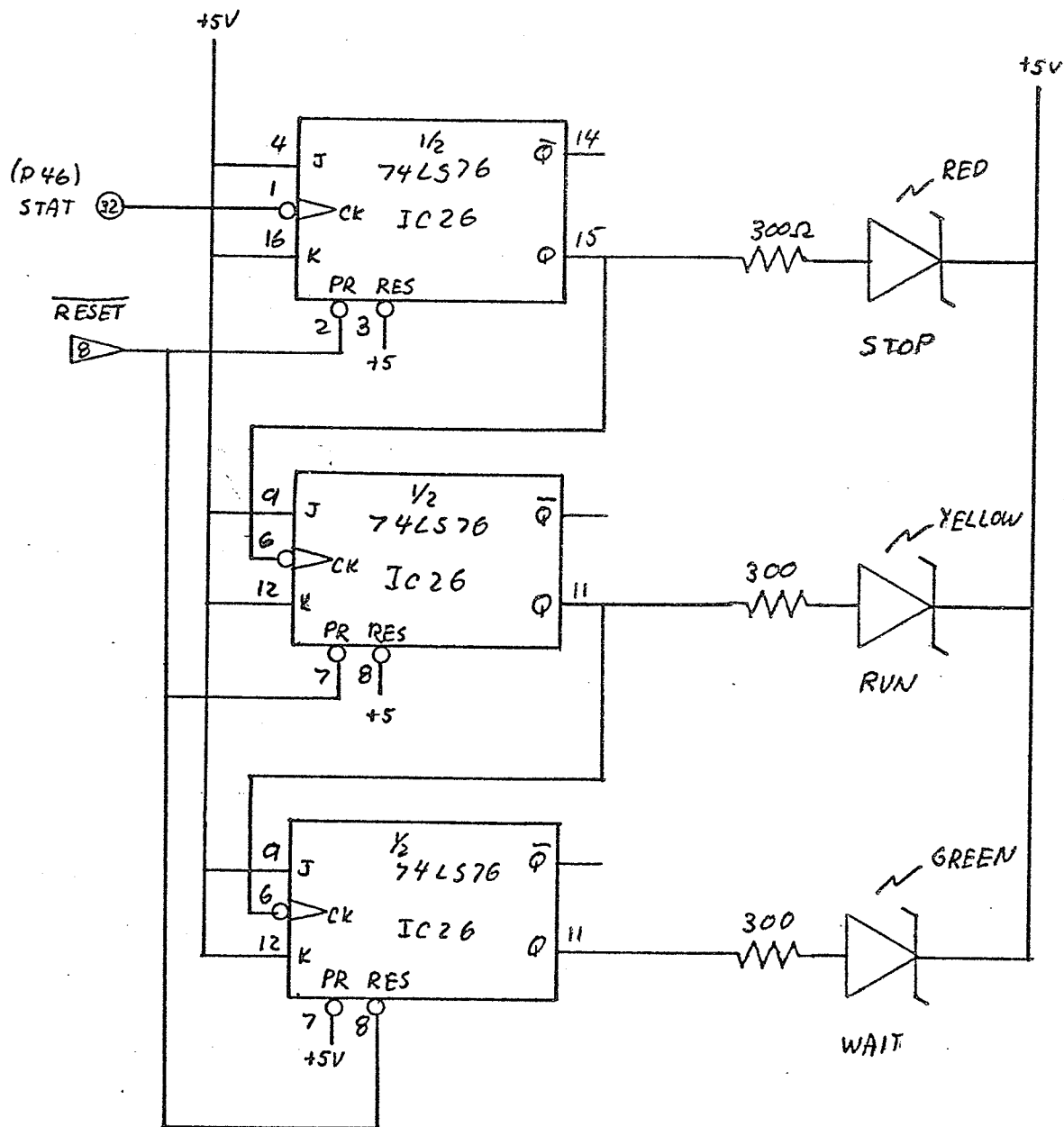
SHIFT CONTROL



Status Register and Selector



SYSTEM CLOCK INTERFACE



PROGRAM STATUS DISPLAY

Appendix B

LPCMK MICROPROGRAM LISTINGS

TITLE LPCMK DEFINITION FILE

```

0020      SIZE 48
0030      OPT  -L,D=DEFN:1
0040 PR    EQU  0AH
0050 ;
0060 ; NEXT INSTRUCTION
0070 ;
0080 ALOAD SUB  1B1,24X,3VQ7,4X      ; DEFAULT CONDITION MET
0090 JZ     FORM 4H0,12X,1B1,31X    ; RESTART
0100 CJS    FORM 4H1,12X,ALOAD      ; CONDITIONAL JUMP TO SUBROUTINE
0110 JMAB   FORM 4H2,12X,1B1,31X    ; JUMP TO MAPPING PROM ADDRESS
0120 CJP    FORM 4H3,12X,ALOAD      ; CONDITIONAL JUMP
0130 PUSH   FORM 4H4,12X,ALOAD      ; PUSH / CONDITIONAL LOAD COUNTER
0140 JSRP   FORM 4H5,12X,ALOAD      ; CONDITIONAL JUMP TO SUB. R/PL
0150 CJV    FORM 4H6,12X,ALOAD      ; CONDITIONAL JUMP TO VECTOR
0160 JRP    FORM 4H7,12X,ALOAD      ; COND. JUMP R/PL
0170 RFCT   FORM 4H8,12X,1B1,31X    ; REPEAT LOOP COUNTER NOT ZERO
0180 RPCT   FORM 4H9,12X,1B1,31X    ; REPEAT PL COUNTER NOT ZERO
0190 CRTN   FORM 4HA,12X,ALOAD      ; COND. RETURN FROM SUBROUTINE
0200 CJPP   FORM 4HB,12X,ALOAD      ; COND. JUMP PL AND POP
0210 LDCT   FORM 4HC,12X,1B1,31X    ; LOAD COUNTER
0220 CLOOP  FORM 4HD,12X,ALOAD      ; COND. LOOP
0230 CONT   FORM 4HE,12X,1B1,31X    ; CONTINUE
0240 TWB    FORM 4HF,12X,ALOAD      ; THREE WAY BRANCH
0250 ;
0260 ; THE DEFAULT CONDITION FOR ALL BRANCHES IS COND. MET
0270 ;
0280 ; NEXT ADDRESS/DATA FIELD
0290 ;
0300 NA     FORM 4X,12VH0,32X      ; NEXT ADDRESS DEFAULT=RESTART
0310 DATA  FORM 16VH0,1B0,31X    ; IMMEDIATE DATA DEFAULT=ZERO
0320 CYC    FORM 4X,12VH8,32X     ; CYCLE COUNTER LOAD
0330 ;
0340 ; CONDITION CODE FIELD
0350 ;
0360 QQ0    EQU  0Q              ; HALF NEGATIVE
0370 IRQ    EQU  4Q
0380 Q0     EQU  2Q
0390 OVR    EQU  6Q
0400 N      EQU  1Q
0410 Z      EQU  5Q
0420 C      EQU  3Q
0430 MET    EQU  7Q
0440 ;
0450 ; ALU REGISTER ASSIGNMENTS
0460 ;
0470 PROG   EQU  4H0              ; PROGRAM COUNTER
0480 ACCA   EQU  4H1
0490 ACCB   EQU  4H2
0500 ACCC   EQU  4H3
0510 ACCD   EQU  4H4
0520 ACCE   EQU  4H5
0530 AUTOHIGH EQU  4H6
0540 AUTOLOW EQU  4H7

```

```

0550 REFL EQU 4H8 ; REFLECTION COEFFICIENT
0560 MCND EQU 4H9 ; FOR MULTIPLY ROUTINE
0570 HAMLOC EQU 4HA ; CURRENT HAMMING WINDOW LOC.
0580 M1 EQU 4HB
0590 M2 EQU 4HC
0600 LPP EQU 4HD ; AUTOCORRELATION BUFFER
0610 DACBUF EQU 4HE ; DATA ACQUISITION BUFFER
0620 FLAG EQU 4HF ; PROGRAM CONTROL FLAGS
0630 ;
0640 ; STACK POINTER IS DEFINED IN MICROCODE AND THEREFORE
0650 ; NOT EXTERNALLY ALTERABLE.
0660 ;
0670 ; STACK VALUE = AD MSW LOCATION
0680 ;
0690 A1 EQU 16HFF ; MSW
0700 A0 EQU 16H#A1-1 ; LSU
0710 B1 EQU 16H#A1-2 ; MSW
0720 B0 EQU 16H#A1-3 ; LSU
0730 SRTBUF1 EQU 16H#A1-4 ; STARTING ADDRESS OF BUFFER 1
0740 SRTBUF2 EQU 16H#A1-5 ; STARTING ADDRESS OF BUFFER 2
0750 ENDHAM EQU 16H#A1-6 ; ST. ADDR. OF HAMMING WINDOW
0760 A1BUF EQU 16H#A1-7
0770 A0BUF EQU 16H#A1-8
0780 B1BUF EQU 16H#A1-9
0790 B0BUF EQU 16H#A1-A
0800 AUTOX EQU 16H#A1-B ; AUTOCORRELATION STORAGE INDEX
0810 LPCDEX EQU 16H#A1-C ; LPC COEFF. STORAGE INDEX
0820 REFLDEX EQU 16H#A1-D ; REFLECTION COEFF. STORAGE INDEX
0830 AX1 EQU 16H#A1-E ; USED IN AUTOCORRELATION
0840 AX0 EQU 16H#A1-F
0850 BX1EVEN EQU 16H#A1-10
0860 BX0EVEN EQU 16H#A1-11
0870 BX1ODD EQU 16H#A1-12
0880 BX0ODD EQU 16H#A1-13
0890 ERRX EQU 16H#A1-14 ; PREDICTOR ERROR INDEX
0900 ERSTAK EQU 16H#A1-15 ; PREDICTOR ERROR BUFFER
0910 PROGSTK EQU 16H#A1-16 ; PROGRAM COUNTER STACK
0920 AUTSCA EQU 16H#A1-17 ; AUTO.NORMALIZATION BUFFER
0930 WBOUND EQU 16H#A1-18 ; WORD BOUNDARY COUNTER
0940 PRI EQU 16H#A1-19 ; NUMBER OF COEFFICIENTS
0950 REGSTK EQU 16H#A1-1A ; TOP OF INTERRUPT STACK
0960 ;
0970 ; ALU
0980 ;
0990 ; SCOURCE
1000 ;
1010 AQ EQU 0Q ; R=PORT A S=QREG
1020 AB EQU 1Q ;
1030 ZQ EQU 2Q ;
1040 ZB EQU 3Q ;
1050 ZA EQU 4Q ;
1060 DA EQU 5Q ;
1070 DQ EQU 6Q ;
1080 DZ EQU 7Q ; R=DATA S= 0

```

```

1090 ;
1100 S      FORM 17X,3VQ#DZ,28X
1110 ;
1120 ; DESTINATION
1130 ;
1140 ;
1150 QREG   EQU 3Q0      ; Y = F F ==- Q
1160 NOP    EQU 3Q1      ; Y = F TEST
1170 RAMA   EQU 3Q2      ; Y = A F ==- B
1180 RAMF   EQU 3Q3      ; Y = F F ==- B
1190 ;
1200 D      FORM 23X,3VQ#RAMF,22X
1210 ;
1220 ; ROTATE AND SHIFT FIELDS
1230 ;
1240 BLOAD  SUB 18X,2Q0,2X ; ZERO SHIFTS UP AND DOWN
1250 RZD    FORM 23X,3Q5,BLOAD ; 0 > RAM3 DOWNSHIFT
1260 RQZD   FORM 23X,3Q4,BLOAD ; 0 > RAM3 0 > Q3
1270 RZU    FORM 23X,3Q7,BLOAD ; RAM0 < 0 SHIFT UP
1280 RQZU   FORM 23X,3Q6,BLOAD ; RAM0 < 0 Q0 < 0
1290 ;
1300 CLOAD  SUB 18X,2Q2,2X ; ROTATE UP AND DOWN
1310 RSRD   FORM 23X,3Q5,CLOAD ; RAM SINGLE ROTATE DOWN
1320 RQSRD  FORM 23X,3Q4,CLOAD ; RAM + QREG SINGLE ROTATE DOWN
1330 RSRU   FORM 23X,3Q7,CLOAD ; RAM SINGLE ROTATE UP
1340 RQSU   FORM 23X,3Q6,CLOAD ; RAM + QREG SINGLE ROTATE UP
1350 ;
1360 DLOAD  SUB 18X,2Q1,2X ; DOUBLE ROTATE UP AND DOWN
1370 RDRD   FORM 23X,3Q5,DLOAD ; Q0 > RAM3
1380 RQDRD  FORM 23X,3Q4,DLOAD ; Q0 > RAM3 RAM0 > Q3
1390 RDRU   FORM 23X,3Q7,DLOAD ; Q3 > RAM0
1400 RQDRU  FORM 23X,3Q6,DLOAD ; Q3 > RAM0 RAM3 > Q0
1410 ;
1420 ELOAD  SUB 18X,2Q3,2X ; ARITHMETIC SHIFT UP AND DOWN
1430 SRA    FORM 23X,3Q5,ELOAD ; F3 > RAM3 ARITH. SHIFT RIGHT.
1440 DSRA   FORM 23X,3Q4,ELOAD ; F3 > RAM3 RAM0 > Q3
1450 SLA    FORM 23X,3Q7,ELOAD ; Q3 > RAM0 ARITH. SHIFT LEFT
1460 DSLA   FORM 23X,3Q6,ELOAD ; Q3 > RAM0 0 > Q0
1470 ;
1480 ; PORT ADDRESSING SOURCE
1490 ;
1500 PM     EQU 2Q0      ; ADDRESS FROM MICROINSTRUCTION
1510 PHAOPB EQU 2Q1      ; PORT A ADDR.FROM MICRO.& PORT B ADDR.FROM OPRAND
1520 PHBOPA EQU 2Q2      ; PORT A ADDR. FROM OPRAND & PORT B ADDR.FROM MICRO.
1530 POP    EQU 2Q3      ; ADDRESS FROM OPERAND
1540 ;
1550 P      FORM 27X,4VH#ACCA,4VH#ACCA,2VQ#PM,11X ; PORT SELECT
1560 ;
1570 ; ALU FUNCTION
1580 ;
1590 ADD    EQU 0Q      ; R+S
1600 SUB    EQU 1Q      ; S-R
1610 MUL    EQU 2Q      ; R-S
1620 OR     EQU 3Q      ; R OR S

```

```

.630 AND EQU 4Q ; R AND S
.640 NRS EQU 5Q ; (NOT R) AND S
.650 EOR EQU 6Q ; R EXOR S
.660 ENOR EQU 7Q ; R EXNOR S
.670 ;
.680 CO EQU 0B ; CARRY=0
.690 C1 EQU 1B ; CARRY = 1
.700 ;
.710 F FORM 20X,3VQ#OR,3X,1VR#CO,21X
.720 ;
.730 ;
.740 ; PROGRAM LEVEL STATUS FLAGS ARE TESTED
.750 ; REGISTER 'FLAG' IS USED IN A MAILBOX FASHION TO PASS STATUS.
.760 ; THE ZERO FLAG IS SET IF CONDITIONS ARE MET.
.770 ;
.780 TST SUB 1B0,DA,AND,NOP,1X,FLAG,4X,PH,11X ; ZERO FLAG IS USED
.790 TSTBUF2 FORM 16H1,TST ; BUFFER2 USED BY DATA ACQ.ROUTINE
.800 TSTNOVR FORM 16H2,TST ; MEMORY OVERFLOW WILL OCCUR
.810 TSTEOW FORM 16H4,TST ; END OF WORD.
.820 TSTZERO FORM 16H8,TST ; RESULTS ARE ZERO.
.830 ;
.840 EN EQU 1 ; ENDABLE PROGRAM STATUS ON I/O FORMS
.850 ;
.860 ; INPUT/OUTPUT
.870 ;
.880 MAR FORM 37X,0B,1VB1,1B,0B,5X,1VB1,1VB1 ; MEMORY ADDRESS REG IS ENABLED
.890 DOUT FORM 37X,1B,1VB1,1B,1B,5X,1VB1,0 ; DATA OUT REG. ENABLE
.900 DIN FORM 37X,1B,1VB1,0B,0B,5X,1VB1,1VB1 ; DATA INPUT ENABLE
.910 NOIO FORM 37X,1B,1VB1,1B,0B,5X,1VB1,1VB1 ; NO I/O OPERATIONS
.920 MDIN FORM 37X,0B,1VB1,0B,0B,5X,1VB1,1VB1 ; MAR AND DIN BOTH LOADED.
.930 ;
.940 ; INTERRUPT VECTOR /INTERRUPT ACKNOWLEDGE
.950 ; - DATA IS PASSED TO INTERRUPT VECTOR THROUGH D-BUS
.960 ; - ALSO AN INTERRUPT ACKNOWLEDGE FOR THE DATA ACQUISITION ROUTINE
.970 ;
.980 INTA EQU 0 ; INT. ACKN. / VECTOR LOAD
.990 ;
2000 ; MAJOR ALU FUNCTION DEFINITIONS
2010 ;
2020 ZREG FORM 17X,ZB,AND,RAMF,5X,4VH#ACCA,PH,11X ; REG (ACCA) = 0
2030 INCR FORM 17X,ZB,ADD,3VH#RAMF,C1,4X,4VH#ACCA,PH,11X ; REG(ACCA)+1=REG
2040 TFST FORM 17X,ZA,OR,NOP,1X,4VH#ACCA,4X,PH,11X ; TEST REG(ACCA)
2050 TESTQ FORM 17X,ZQ,OR,NOP,22X ; TEST REGISTER Q
2060 PASSE FORM 17X,ZB,EOR,8X,4VH#ACCA,PH,11X
2070 ;
2080 ; ADDA - PORTB=PORTB+PORTA DEFAULT PORTB=ACCA
2090 ; PORTA=MCND
2100 ;
2110 ADDA FORM 17X,AB,ADD,RAMF,C0,4VH#MCND,4VH#ACCA,PH,11X
2120 ;
2130 ; ADA - PORTB + PORTA DEFAULT PORTB=ACCA
2140 ; PORTA=MCND
2150 ; NO DESTINATION SPECIFIED.
2160 ;

```

```
2170 ADA    FORM 17X,AB,ADD,3X,C0,4VH#MCND,4VH#ACCA,PM,11X
2180 ;
2190 ;SUBA  - PORTB=PORTB-A      DEFAULT  PORTE=ACCA
2200 ;                                PORTA = MCND
2210 SUBA    FORM 17X,AB,SUR,RAF,C1,4VH#MCND,4VH#ACCA,PM,11X
2220 ;
2230 ;SBA   - PORTB - PORTA      DEFAULT  PORTB = ACCA
2240 ;                                PORTA = MCND
2250        END
```

```
TOTAL ERRORS    0
```


0010
0020
0030

TITLE LPCMK SOFTWARE
OPT 0=LPC:1,T=DEFN:1

```
0040 ; THE PROGRAM INSTRUCTIONS ARE DEFINED WHICH
0050 ; ARE USED IN CALCULATING THE LPC, ERROR, AND
0060 ; REFLECTION COEFFICIENTS FOR A SPEECH SIGNAL
0070 ; IN REAL TIME.
0080
0090 ; FETCH - THIS ROUTINE WILL FETCH INSTRUCTION OR
0100 ; . DATA FROM PROGRAM MEMORY AT THE PRESENT
0110 ; . PROGRAM COUNTER LOCATION AND INCREMENT THE
0120 ; . P.C. FOR THE NEXT FETCH.
0130 ;
0140 0000 E000B0A00303 FETCH CONT S,ZB F,ADD,C1 D,RAHA P,PROG,PROG MAR ;
0150 0001 E00080400603 CONT D,NOP DIN ; ALLOW NEXT MI ADDRESS TO SETTLE
0160 0002 200080400703 JMAP D,NOP NOIO ; MAPPING PROM
0170 ;
0180 ;
0190 0003 E00080000000 CONT
0200 0004 E00080000000 CONT
0210 0005 E00080000000 CONT
0220 0006 E00080000000 CONT
0230 ;
0240 ;
0250 ; 1. DATA MOVEMENT INSTRUCTIONS ALLOW
0260 ; . - MEMORY-REGISTER
0270 ; . - REGISTER-REGISTER
0280 ; . - REGISTER-MEMORY
0290 ; . TRANSFERS.
0300 ;
0310 ; LDRI XX,RB ; DATA - DATA IS LOADED INTO RB THROUGH
0320 ; . -ACCA.
0330 ;
0340 0007 E000C6402303 LDRI CONT S,ZA F D,NOP P,PROG MAR ; FETCH DATA
0350 0008 E000C2C02203 CONT S,ZA F,SUR D P,PROG MDIN
0360 0009 E000F6C22703 CONT S F D P NOIO ; ACCA=DATA
0370 000A E000C6C22E03 CONT S,ZA F D P,,,PHAOPB DIN
0380 000B 3000B0E20773 MA,FETCH CJP S,ZB F,ADD,C1 D P,,PROG NOIO
0390 ;
0400 ;
0410 000C E00080000000 CONT
0420 000D E00080000000 CONT
0430 000E E00080000000 CONT
0440 000F E00080000000 CONT
0450 ;
0460 ; LDRQ XX,RB - RB IS LOADED WITH DATA FROM REGQ
0470 ;
0480 ;
0490 0010 E00080000000 CONT
0500 0011 E00080000000 CONT
0510 0012 E00080000000 CONT
0520 0013 E00080000000 CONT
```

```

0530 ;
0540 0014 3000A6C23F73 LDRQ NA,FETCH CJP S,ZQ F D P,,,POP NOIO ; RB=QREG
0550
0560 ; LDQR RA,XX - QREG IS LOADED WITH DATA FROM RB
0570 ;
0580 0015 3000C6023F73 LDQR NA,FETCH CJP S,ZA F D,QREG P,,,POP NOIO ; QREG=RA
0590 ;
0600 ;
0610 0016 E00080000000 CONT
0620 0017 E00080000000 CONT
0630 0018 E00080000000 CONT
0640 0019 E00080000000 CONT
0650
0660 ; LDQ XX,XX : DATA - QREG LOADED FROM IMMEDIATE DATA
0670 ;
0680 001A E000B0A00303 LDQR CONT S,ZB F,ADD,C1 D,RAMA P,PROG,PROG MAR
0690 ;
0700 ;
0710 001B E00080000000 CONT
0720 001C E00080000000 CONT
0730 001D E00080000000 CONT
0740 001E E00080000000 CONT
0750 001F 3000F6000673 NA,FETCH CJP S F D,QREG DIN ; QREG=DATA
0760
0770 ; LDRR RA,RB - RA TRANSFER TO RB
0780 0020 3000C6C23F73 LDRR NA,FETCH CJP S,ZA F D P,,,POP NOIO ; RB=RA
0790 ;
0800 ;
0810 0021 E00080000000 CONT
0820 0022 E00080000000 CONT
0830 0023 E00080000000 CONT
0840 0024 E00080000000 CONT
0850
0860 ; LDM RA,(RB) - RA CONTENTS TO (RB) LOCATION
0870 ;
0880 0025 000036423B03 LDM S,ZB F D,NOP P,,,POP MAR
0890 0026 3000C6423FF2 NA,FETCH CJP S,ZA F D,NOP P,,,POP DOUT
0900 ;
0910 ;
0920 0027 E00080000000 CONT
0930 0028 E00080000000 CONT
0940 0029 E00080000000 CONT
0950 002A E00080000000 CONT
0960
0970 ; LDRMI XX,(RB) : DATA - DATA LOADED INTO (RB) LOCATION
0980 ; THROUGH ACCA.
0990 ;
1000 002B E000C6402303 LDRMI CONT S,ZA F D,NOP P,PROG MAR
1010 002C E000C2402203 CONT S,ZA F,SUR D,NOP P,PROG MDIN
1020 002D E000F6C22703 CONT S F D P NOIO ; ACCA = DATA
1030 002E E000B6423B03 CONT S,ZB F D,NOP P,,,POP MAR ; MAR = RB
1040 002F 3000B0E20773 NA,FETCH CJP S,ZB F,ADD,C1 D P,,,PROG NOIO
1050 ;
1060 ;

```

1070	0030	E00080000000	CONT	
1080	0031	E00080000000	CONT	
1090	0032	E00080000000	CONT	
1100	0033	E00080000000	CONT	
1110				
1120			LDR (RA),RB	- (RA) MEMLOCATION TO RB
1130				MEMORY TO REGISTER.
1140				
1150	0034	E000C6423B03	LDR	CONT S,ZA F D,NOP P,,,POP MAR ; GET MEM.CONTENTS
1160	0035	3000F6C23E73		NA,FETCH CJP S F D P,,,POP DIN ; RB=MEM.CONTENTS
1170				
1180				
1190	0036	E00080000000	CONT	
1200	0037	E00080000000	CONT	
1210	0038	E00080000000	CONT	
1220	0039	E00080000000	CONT	
1230				
1240			LDHI XXXX : DATA : ADDRESS	- DATA IS LOADED INTO THE
1250				MEMORY ADDRESS PROVIDED
1260				THROUGH ACCA.
1270				
1280	003A	E00080A00303	LDHI	CONT S,ZB F,ADD,C1 D,RAMA P,PROG,PROG MAR
1290	003B	E000F6802203		CONT S F D,RAMA P,PROG MDIN ; ACCA=DATA
1300	003C	E000C6422782		CONT S,ZA F D,NOP P DOUT ; MEM=DATA
1310	003D	3000B0E20773		NA,FETCH CJP S,ZB F,ADD,C1 D P,,PROG MDIO
1320				
1330				
1340	003E	E00080000000	CONT	
1350	003F	E00080000000	CONT	
1360	0040	E00080000000	CONT	
1370	0041	E00080000000	CONT	
1380				
1390				2. PROGRAM CONTROL INSTRUCTIONS
1400				
1410			BRA XXXX : ADDRESS	- UNCONDITIONAL BRANCH
1420				
1430	0042	E000C6402303	BRA	CONT S,ZA F D,NOP P,PROG MAR ; MAR=PROG
1440	0043	E000F0E20603		CONT S F,ADD,C1 D P,,PROG DIN ; PROG=ADDRESS+1
1450	0044	E000C2402303		CONT S,ZA F,SUR D,NOP P,PROG MAR
1460	0045	E00080400603		CONT D,NOP DIN
1470	0046	200080400703		JMAP D,NOP MDIO
1480				
1490				
1500	0047	E00080000000	CONT	
1510	0048	E00080000000	CONT	
1520	0049	E00080000000	CONT	
1530	004A	E00080000000	CONT	
1540				
1550			BRC : ADDRESS	- BRANCH IF MSB OF FLAG REGISTER SET
1560				
1570	004B	E000B681E303	BRC	CONT S,ZB F D,RAMA P,PROG,FLAG MAR
1580	004C	304EB0E20613		NA,BRC1 CJP,N S,ZB F,ADD,C1 D P,,PROG DIN

1590	004D	300080400773	NA,FETCH CJP D,NOP NOIO
1600	004E	3000F6C20773	BRC1 NA,FETCH CJP S F D P,,PROG NOIO
1610		;	
1620		;	
1630	004F	E00080000000	CONT
1640	0050	E00080000000	CONT
1650	0051	E00080000000	CONT
1660	0052	E00080000000	CONT
1670			
1680		;	BRF : ADDRESS - BRANCH IF MSB OF FLAG IS ZERO
1690		;	
1700	0053	E00086800303	BRF CONT S,ZB F D,RAHA P,PROG,PROG MAR
1710	0054	3000B0E20613	NA,FETCH CJP,N S,ZB F,ADD,C1 D P,,PROG DIN
1720	0055	3000F6C20773	NA,FETCH CJP S F D P,,PROG NOIO
1730		;	
1740		;	
1750	0056	E00080000000	CONT
1760	0057	E00080000000	CONT
1770	0058	E00080000000	CONT
1780	0059	E00080000000	CONT
1790			
1800		;	STOP XXXX - A SELF LOOP
1810		;	
1820		;	
1830	005A	E00080000000	CONT
1840	005B	E00080000000	CONT
1850	005C	E00080000000	CONT
1860	005D	E00080000000	CONT
1870		;	
1880	005E	305E80400773	STP1 NA,STP1 CJP D,NOP NOIO
1890			
1900		;	NOP XXXX - CONTINUE
1910		;	
1920	005F	300080400773	NOOP NA,FETCH CJP D,NOP NOIO
1930		;	
1940		;	
1950	0060	E00080000000	CONT
1960	0061	E00080000000	CONT
1970	0062	E00080000000	CONT
1980	0063	E00080000000	CONT
1990			
2000		;	3. ARITHMETIC AND LOGICAL OPERATIONS WILL CONSIST ONLY
2010		;	
2020		;	
2030		;	TEST BIT,X BIT FROM 1-16 OF FLAG REGISTER IS MOVED INTO MSB
2040		;	POSITION.
2050		;	
2060	0064	E000F6C22703	TEST CONT S F D P NOIO
2070	0065	0F0059C22703	DATA,0F00H S,DA F,AND D P NOIO
2080	0066	000176C24703	DATA,1 S F D P,,ACCB NOIO ; SET PATTERN
2090	0067	010052E22703	TEST2 DATA,0100H S,DA F,SUR,C1 D P NOIO
2100	0068	306A80400713	NA,TEST1 CJP,N D,NOP NOIO

```

2110 0069 306787C42773      NA,TEST2 CJP S,ZB F RZU P,ACCB NOIO
2120 006A E00098C24703 TEST1  CONT S,AB F,AND D P,,ACCB NOIO
2130 006B 306E80400753      NA,TEST3 CJP,Z D,NOP NOIO
2140 006C 800056DFE703      DATA,8000H S,DA F,OR D P,FLAG,FLAG NOIO
2150 006D 300080400773      NA,FETCH CJP D,NOP NOIO
2160 006E 80005ADFE703 TEST3  DATA,8000H S,DA F,NRS D P,FLAG,FLAG NOIO
2170 006F 300080400773      NA,FETCH CJP D,NOP NOIO
2180                          ;
2190                          ;
2200 0070 E00080400703      CONT D,NOP NOIO
2210 0071 E00080400703      CONT D,NOP NOIO
2220 0072 E00080400703      CONT D,NOP NOIO
2230
2240                          ;
2250                          ;
2260                          ;
2270                          ;
2280 0073 124A80400773 MULTIP  NA,MULT CJS D,NOP NOIO
2290 0074 300080400773      NA,FETCH CJP D,NOP NOIO
2300 0075 E00080400703      CONT D,NOP NOIO
2310 0076 E00080400703      CONT D,NOP NOIO
2320 0077 E00080400703      CONT D,NOP NOIO
2330
2340
2350                          ;
2360                          ;
2370 0078 125E80400773 DIVI    NA,DIVIDE CJS D,NOP NOIO
2380 0079 300080400773      NA,FETCH CJP D,NOP NOIO
2390 007A E00080400703      CONT D,NOP NOIO
2400 007B E00080400703      CONT D,NOP NOIO
2410 007C E00080400703      CONT D,NOP NOIO
2420
2430                          ;
2440                          ;
2450                          ;
2460                          ;
2470                          ;
2480                          ;
2490                          ;
2500                          ;
2510                          ;
2520                          ;
2530                          ;
2540                          ;
2550                          ;
2560                          ;
2570                          ;
2580                          ;
2590                          ;
2600                          ;
2610                          ;
2620                          ;
2630                          ;
2640                          ;
2650                          ;
2660                          ;
2670                          ;
2680                          ;
2690                          ;
2700                          ;
2710                          ;
2720                          ;
2730                          ;
2740                          ;
2750                          ;
2760                          ;
2770                          ;
2780                          ;
2790                          ;
2800                          ;
2810                          ;
2820                          ;
2830                          ;
2840                          ;
2850                          ;
2860                          ;
2870                          ;
2880                          ;
2890                          ;
2900                          ;
2910                          ;
2920                          ;
2930                          ;
2940                          ;
2950                          ;
2960                          ;
2970                          ;
2980                          ;
2990                          ;
3000                          ;
3010                          ;
3020                          ;
3030                          ;
3040                          ;
3050                          ;
3060                          ;
3070                          ;
3080                          ;
3090                          ;
3100                          ;
3110                          ;
3120                          ;
3130                          ;
3140                          ;
3150                          ;
3160 007D 00F876400303 INIT    DATA,A1BUF S F D,NOP MAR
3170 007E E000F8C36732      CONT S F,AND D P,,M1 DOUT

```

4. SPECIAL INSTRUCTIONS

```

0060 SRTHAM EQU 100H
0070 SRTREF EQU 500H
0080 SRTLPC EQU 700H
0090 BUF1 EQU 200H
0100 BUF2 EQU 300H
0110 SRTAUTO EQU 900H
0120 SRTERR EQU 400H
0130 HAMEND EQU 1C7H
0140 COEF EQU 0AH
0150

```

0180 007F 09F776400303	DATA,A0BUF S F D,NOP MAR
0190 0080 E000F8C38782	CONT S F,AND D P,,M2 DOUT
0200 0081 00F676400303	DATA,B1BUF S F D,NOP MAR
0210 0082 E000F8C20782	CONT S F,AND D P,,PRG DOUT
0220 0083 00F676400303	DATA,B1BUF S F D,NOP MAR
0230 0084 E000F8C3E782	CONT S F,AND D P,,FLAG DOUT
0240 0085 010076C34703	DATA,SRTHAM S F D P,,HAMLOC NOIO
0250 0086 00F976400303	DATA,ENDHAM S F D,NOP MAR
0260 0087 01C776400782	DATA,HAMEND S F D,NOP DOUT
0270 0088 00FR76400303	DATA,SRTBUF1 S F D,NOP MAR
0280 0089 020076C3C782	DATA,BUF1 S F D P,,DACBUF DOUT
0290 008A 00FA76400303	DATA,SRTBUF2 S F D,NOP MAR
0300 008B 030076400782	DATA,BUF2 S F D,NOP DOUT
0310 008C 00EB76400303	DATA,ERRX S F D,NOP MAR
0320 008D 040076400782	DATA,SRTERR S F D,NOP DOUT
0330 008E 00F276400303	DATA,REFLDEX S F D,NOP MAR
0340 008F 050076400782	DATA,SRTREF S F D,NOP DOUT
0350 0090 00F376400303	DATA,LPCDEX S F D,NOP MAR
0360 0091 070076400782	DATA,SRTLPC S F D,NOP DOUT
0370 0092 00F476400303	DATA,AUTOX S F D,NOP MAR
0380 0093 090076400782	DATA,SRTAUTO S F D,NOP DOUT
0390 0094 00E676400303	DATA,PR1 S F D,NOP MAR
0400 0095 000A76400782	DATA,COEF S F D,NOP DOUT
0410 0096 00E776400303	DATA,WROUND S F D,NOP MAR
0420 0097 E000F8400782	CONT S F,AND D,NOP DOUT
0430	;
0440	;
0450	;
0460 0098 CDFE80400783 ZERO	MEMORY IS ZEROED ABOVE 200H
0470 0099 020076C22303	NA,0DFEH LDCT D,NOP NOIO
0480 009A E000F8400782 ZE1	DATA,200H S F D P MAR
0490 009B 909A80E22303	CONT S F,AND D,NOP DOUT
0500 009C A00080400773	NA,ZE1 RPCT S,ZB F,ADD,C1 D P MAR
0510	CRTN D,NOP NOIO
0520	;
0530	;
0540 009D E00080000000	LEAVE SPACE FOR MODIFICATION.
0550 009E E00080000000	CONT
0560 009F E00080000000	CONT
0570 00A0 E00080000000	CONT
0580	;
0590	;
0600	;
0610	;
0620	;
0630	;
0640	;
0650 00A1 00085ADFE703 SOW	SOW - BIT 4 OF FLAG=1 IF START OF WORD DETECTED &
0660 00A2 00E876400303	ZERO SIGN OF STSUS REGISTER CLEARED.
0670 00A3 E000F6C22603	DATA,0008H S,DA F,NRS D P,FLAG,FLAG NOIO ; BIT 4 = 0
0680 00A4 000254622703	DATA,AUTSCA S F D,NOP MAR
0690 00A5 A000F8400713	CONT S F D P DTN
0700	DATA,2 S,DA F,SUS,C1 D,NOP P NOIO
0710 00A6 000256DFE703	CRTN,N S F,AND D,NOP NOIO ; SET ZERO STATUS
	DATA,0008H S,DA F,OR D P,FLAG,FLAG NOIO ; BIT 4 = 1

```

0720 00A7 A000C65E2773 CRTN S,ZA F D,NOP P,FLAG NOIO ; CLEAR ZERO FLAG
0730 ;
0740 ;
0750 00A8 E00080000000 CONT
0760 00A9 E00080000000 CONT
0770 00AA E00080000000 CONT
0780 00AB E00080000000 CONT
0790 ;
0800 ;
0810 ;
0820 ;
0830 ;
0840 ;
0850 ;
0860 00AC 00EB76400303 MOVFL DATA,ERRX S F D,NOP MAR
0870 00AD E000F6C22603 CONT S F D P DIN
0880 00AE 043252622703 DATA,432H S,DA F,SUR,C1 D,NOP P NOIO
0890 00AF 30B280400713 NA,MOVFL1 CJP,N D,NOP NOIO
0900 00B0 000256DFE703 DATA,2H S,DA F,OR D P,FLAG,FLAG NOIO
0910 00B1 A000F8400773 CRTN S F,AND D,NOP NOIO ; SET ZERO FLAG
0920 ;
0930 00B2 00025ADFE703 MOVFL1 DATA,2H S,DA F,NRS D P,FLAG,FLAG NOIO ; CLEAR BIT
0940 00B3 A000F8400773 CRTN S F D,NOP NOIO ; Z=0
0950 ;
0960 ;
0970 00B4 E00080000000 CONT
0980 00B5 E00080000000 CONT
0990 00B6 E00080000000 CONT
1000 00B7 E00080000000 CONT
1010 ;
1020 ;
1030 ;
1040 ;
1050 ;
1060 ;
1070 00B8 00045ADFE703 EOW DATA,04 S,DA F,NRS D P,FLAG,FLAG NOIO ; BIT 3 = 0
1080 00B9 00E876400303 DATA,AUTSCA S F D,NOP MAR
1090 00BA E000F6C22603 CONT S F D P DIN
1100 00BB 00E776400303 DATA,WBOUND S F D,NOP MAR
1110 00BC 000852622703 DATA,8 S,DA F,SUR,C1 D,NOP P NOIO ; TEST FOR SILENT FRAME
1120 00BD 30C2F6C24613 NA,EDW1 CJP,N S F D P,,ACCB DIN
1130 ;
1140 00BE 000254642703 DATA,2 S,DA F,SUS.C1 D,NOP P,ACCB NOIO
1150 00BF 30C3C0642792 NA,EDW3 CJP,N S,ZA F,ADD,C1 D,NOP P,ACCB DOUT ; SILENT FRAME COUNTER
1160 00C0 000456DFE703 DATA,04H S,DA F,OR D P,FLAG,FLAG NOIO
1170 00C1 A000C65E2773 EOW2 CRTN S,ZA F D,NOP P,FLAG NOIO ; CLEAR ZERO STATUS
1180 ;
1190 00C2 E000F8400782 EDW1 CONT S F,AND D,NOP DOUT ; RESET WBOUND COUNT
1200 00C3 A000F8400773 EOW3 CRTN S F,AND D,NOP NOIO ; SET ZERO FLAG
1210 00C4 E00080000000 CONT
1220 00C5 E00080000000 CONT
1230 00C6 E00080000000 CONT
1240 00C7 E00080000000 CONT
1250 ;

```

```

MOVFL - MEMORY OVERFLOW IF SAMPLE IF LONGER THAN 2SEC.
. - 1 INTO BIT 1 OF FLAG IF OVFL.
. - ALSO ZERO STATUS IS SET IF OVFL.

```

```

EOW - BIT 3 OF FLAG=1 IF END OF WORD
. - ZERO FLAG IS SET ON RETURN IF NOT END OF WORD

```

```

1260 ;-----
1270 ;
1280 ; STAT1 - SETS LPC CALC STATUS LIGHT
1290 ; STAT2 - SETS EQW STATUS LIGHT
1300 ;
1310 00C8 A00080400773 STAT2 CRTN D,NOP NOID,,1 ; UPDATE STATUS
1320 ;
1330 00C9 E00080400703 STAT1 CONT D,NOP NOID,,1
1340 00CA A00080400773 CRTN D,NOP NOID,,1
1350 00CB E00080000000 CONT
1360 00CC E00080000000 CONT
1370 00CD E00080000000 CONT
1380 00CE E00080000000 CONT
1390 ;
1400 ;-----
1410 ;
1420 ; FILL CURRENT FRAME WITH DATA
1430 ;
1440 00CF 000158DE2703 FILL DATA,01 S,DA F,AND D P,FLAG,ACCA NOID
1450 00D0 11F480400743 FILL1 NA,ACQUIRE CJS,IRQ D,NOP NOID
1460 00D1 000158DE4703 DATA,1 S,DA F,AND D P,FLAG,ACCB NOID
1470 00D2 E0009C424703 CONT S,AB F,EDR D,NOP P,ACCA,ACCB NOID
1480 00D3 30D080400753 NA,FILL1 CJP,Z D,NOP NOID
1490 00D4 A00080400773 CRTN D,NOP NOID
1500 ;
1510 ;
1520 00D5 E00080000000 CONT
1530 00D6 E00080000000 CONT
1540 00D7 E00080000000 CONT
1550 00D8 E00080000000 CONT
1560 ;
1570 ;-----
1580 ;
0010 ;-----
0020 ;
0030 ;
0040 ;THE FOLLOWING REGISTERS ARE USED IN THIS SUBROUTINE:
0050 ;---MCNDD, AUTOHIGH (S), AUTOLOW(S), QREG, ACCA, PROG(ADAPTIVE SCALER)
0060 ;---ACCB, ACCE (K), ACCD (J), ACCE (I)
0070 ;
0080 ;MEMORY LOCATIONS:
0090 ;-----AUTOX, (AUTOX) (AUTOX) + PR
0100 ;----- REFLDEX, (REFLDEX) (REFLDEX) + PR
0110 ;----- ERRX, ERSTAK, (ERRX), PROGSTK
0120 ;
0130 ; EXIT UPDATES - (AUTOX) TO (AUTOX)+PR
0140 ; .....- (LPCDEX) TO (LPCDEX)+PR
0150 ; .....- (REFLDEX) TO (REFLDEX)+PR
0160 ; .....- (ERRX) TO (ERRX)+1
0170 ; .....- SAVES PREDICTOR COEFFICIENTS, REFLECTION COEFFICIENTS,
0180 ; ..... AND PREDICTION ERROR,
0190 ;
0200 00D9 00E976400303 DURBIN DATA,PROGSTK S F D,NOP MAR ; STORE PROG COUNTER

```



```

0210 00DA FFFF76800782 DATA,0FFFFH S F D,ZAMA P,PROG,PROG DOUT ; INITIALIZE ADAPTIVE SCALER
0220 00DR 000176C2A703 DURBB DATA,1 S F D P,,ACCF NOIO ; I=1
0230 00DC 1113B0E20773 NA,FIRSTK CJS S,ZB F,ADD,C1 D P,,PROG NOIO ; INCREASE SCALE
0240 00DD 11F480400743 NA,ACQUIRE CJS,IRQ D,NOP NOIO ;*-*-*-*-*-*-*-*-*-*-
0250 00DE 110F80400773 NA,REFSAVE CJS D,NOP NOIO
0260 ;
0270 ; START RECURSION
0280 ;
0290 00DF 10ECC6062773 DURB1 NA,ERRCALC CJS S,ZA F D,QREG P,ACCC NOIO ; E=(1-K*K)*F, GET K
0300 00E0 11F480E2A743 NA,ACQUIRE CJS,IRQ S,ZB F,ADD,C1 D P,,ACCE NOIO ; I=I+1 *-*-*-*-*-*-*-*-*-*-
0310 00E1 000A546A2703 DATA,PR S,DA F,SUS,C1 D,NOP P,ACCE NOIO ; I<=PR ?
0320 00E2 30E8F8C2C713 NA,DURBEND CJP,N S F,AND D P,,AUTOHIGH NOIO ; FINISHED ?, S(MSW) = 0
0330 00E3 10F3F8C2E773 NA,REFCALC CJS S F,AND D P,,AUTLOW NOIO ; S(LSW) = 0
0340 00E4 110F80400773 NA,REFSAVE CJS D,NOP NOIO ; K=ACCC
0350 00E5 112D80400773 NA,AJCALC CJS D,NOP NOIO ; AJ=AJ-K*(I-J) J=1,I-1
0360 00E6 11F480400743 NA,ACQUIRE CJS,IRQ D,NOP NOIO ;*-*-*-*-*-*-*-*-*-*-
0370 00E7 30DF80400773 NA,DURB1 CJP D,NOP NOIO
0380 ;
0390 ; END OF DURBINS ALGORITHM
0400 ;
0410 00E8 112880400773 DURBEND NA,ERSAVE CJS D,NOP NOIO
0420 00E9 114E80400773 NA,REVISEX CJS D,NOP NOIO ; INDEX UPDATE
0430 00EA 00E976400303 DATA,PROGSTK S F D,NOP MAR
0440 00EB A000F6C20673 CRTN S F D P,,PROG DIN ; RETURN AND RESTORE PROGRAM COUNTER
0450 ;
0460 ; -----
0470 ;
0480 ; E=(1-K*K)*E ERROR CALCULATION
0490 ;
0500 00EC 124980400773 ERRCALC NA,SQUARE CJS D,NOP NOIO ; K=K*K
0510 00ED 121EA6400773 NA,ROUND CJS S,ZQ F D,NOP NOIO ; ROUND TO 16 BITS
0520 00EE E000C6C32703 CONT S,ZA F D P,,MCND NOIO ; MCND=K*K
0530 00EF 00EA76400303 DATA,ERSTAK S F D,NOP MAR
0540 00F0 124AF6000673 NA,MULT CJS S F D,QREG DIN ; QREG = ERROR, K*K * E
0550 00F1 121EA6400773 NA,ROUND CJS S,ZQ F D,NOP NOIO ; ROUND TO 16 BITS
0560 00F2 A000D4E227F2 CRTN S,DA F,SUS,C1 D P DOUT ; (ERSTAK)=E, E=E-E*K*K
0570 ;
0580 ; -----
0590 ; SUBROUTINE REFLCALC - THE REFLECTION COEFFICIENTS ARE CALCULATED HERE
0600 ; .....K=(R(I)-S)/E
0610 ; ENTER - ACCE (I) ACCD (J) ERSTAK (E)
0620 ; .....- S = 0 = AUTOHIGH,AUTLOW
0630 ; .....- ((LPCDEX)), ((LPCDEX)+1), . . . , ((LPCDEX)+I-1)
0640 ;
0650 ; EXIT - ACCC = K
0660 ;
0670 ; USE - ACCB (LPCINDEX), ACCC (AUTODEX)
0680 ;
0690 00F3 000176C28703 REFCALC DATA,1 S F D P,,ACCD NOIO ; J=1
0700 00F4 11F4C3CA6743 NA,ACQUIRE CJS,IRQ S,ZA F,SUR RZU P,ACCE,ACCC NOIO ; ACCC = (I-1) * 2 *-*-*-*-*-
0710 00F5 00F476400303 DATA,AUTOX S F D,NOP MAR ; GET AUTO INDEX
0720 00F6 E0000CC66603 CONT S,DA F,ADD D P,ACCC,ACCC DIN ; ACCC = R(I-1) ADDRESS
0730 00F7 00F376400303 DATA,LPCDEX S F D,NOP MAR ;

```

```

0740 00F8 E000F6C24603      CONT S F D P,,ACCB DIN ; LPCINDEX = ACCB
0750 00F9 E000B0A44303      CONT S,ZB F,ADD,C1 D,RAMA P,ACCB,ACCB MAR ; MAR = LPCINDEX (A(I) BUFF)
0760
0770                          ;
0780                          S = S + A(J) * (-R(I-J)) FOR J= 1 TO I-1

0790 00FA 000252A66203 DURB2 DATA,2 S,DA F,SUR,C1 D,RAMA P,ACCC,ACCC MDIN ; GET A(J), MAR=R(I-1) ADDRESS
0800 00FB E000F6000703      CONT S F D,QREG NOIO ; A(J)= QREG
0810 00FC 124AF2E32673      NA,MULT CJS S F,SUR,C1 D P,,MCND DIN ; MCND=-R(I)
0820 00FD 11F480400743      NA,ACQUIRE CJS,IRQ D,NOP NOIO ;*-*-*-*-*-*-*-*-*-*-*-
0830 00FE 11AFB0E28773      NA,RXADD CJS S,ZB F,ADD,C1 D P,,ACCD NOIO ; J=J+1
0840 00FF E0009468A703      CONT S,AB F,SUS,C1 D,NOP P,ACCD,ACCE NOIO ; J-I
0850 0100 30FAB0A44313      NA,DURB2 CJP,N S,ZB F,ADD,C1 D,RAMA P,ACCB,ACCB MAR ; MAR = A(J) , J=J+1
0860
0870                          ;
0880                          K = [ R(I) + (-S) ] / E

0890 0101 E000C7CA2703      CONT S,ZA F,RZU P,ACCE NOIO ; ACCC=R(1) ADDRESS, ACCA=2*(I-1)
0900 0102 E00090C28703      CONT S,AB F,ADD D P,,ACCC NOIO ; ACCC=R(I) ADDRESS
0910 0103 11B6C6C64373      NA,FILLMQ CJS S,ZA F D P,ACCC,ACCB MAR ; ACCB = R(I) MSW ADDRESS
0920                          ;*****
0930                          ;
0940                          ;
0950                          ;
0960 0104 E000C2C04703      CONT S,ZA F,SUR D P,PROG,ACCB NOIO
0970 0105 3108B0400713 DURB11 NA,DURB10 CJP,N D,NOP NOIO
0980 0106 E000B703270F      CONT S,ZB F DSRA P,,MCND NOIO
0990 0107 3105B2C24773      NA,DURB11 CJP S,ZB F,SUR D P,,ACCB NOIO
1000
1010                          ;*****
1020                          ;

1030 0108 E000B00E2703 DURB10 CONT S,AQ F,ADD D,QREG P,AUTOLOW NOIO ; DINKE
1040 0109 310B90ED2733      NA,DURB3 CJP,C S,AB F,ADD,C1 D P,AUTOHIGH,MCND NOIO ; X=R(I)+ (-S)
1050 010A E000B2C32703      CONT S,ZB F,SUR D P,,MCND NOIO
1060 010B 00EA76400303 DURB3  DATA,ERSTAK S F D,NOP MAR ; MAR(ERSTAK
1070 010C 11F480400743      NA,ACQUIRE CJS,IRQ D,NOP NOIO ;*-*-*-*-*-*-*-*-*-*-*-
1080 010D 125EF6C24673      NA,DIVIDE CJS S F D P,,ACCB DIN ; X/ERROR QREG = (MCND,QREG)/ACCB
1090 010E A000A6C26773      CRTN S,ZQ F D P,,ACCC NOIO ; ACCC(K END ACCC=K
1100

1110                          ;-----
1120                          ;
1130                          ;
1140                          ;
1150                          ;
1160                          ;
1170                          ;
1180                          ;
1190 010F 00F276400303 REFSAVE DATA,REFLDEX S F D,NOP MAR ; MAR<REFLDEX
1200 0110 E000C2CA2703      CONT S,ZA F,SUR D P,ACCE NOIO ; I-1 = ACCA
1210 0111 E000D0422203      CONT S,DA F,ADD D,NOP P MDIN ; KI LOCATION
1220 0112 A000C64627F2      CRTN S,ZA F D,NOP P,ACCC DOUT ; KI=K
1230

1240                          ;-----
1250                          ;
SUBROUTINE FIRSTK - R(1) / R(0) = K > ACCC , ERSTAK < E0

```

```

1260 ;
1270 ;      USES - REG - ACCA, ACCB, QREG, MCND, ACCC (K)
1280 ;.... - HEM - ERSTAK (LOCATION OF ERROR STACK) E0(ERROR #0)
1290 ;      EXIT - E0=R(0)/4 MSW IN ERSTAK AND TO ACCA
1300 ;      .... - K1 IN ACCC = R(1)/R(0)
1310 ;
1320 ;
1330 0113 00F476400303 FIRSTK DATA,AUTOX S F D,NOP MAR ;
1340 0114 E000F6C22203      CONT S F D P MDIN ; ACCA = INDEX
1350 0115 E000F6C24603      CONT S F D P,,ACCB DIN ; ACCB = R(0) MSW
1360 0116 000250C22303      DATA,2 S,DA F,ADD D P MAR
1370 0117 E000F6C32603      CONT S F D P,,MCND DIN ; R(1)MSW = MCND
1380 0118 E000C0622303      CONT S,ZA F,ADD,C1 D,NOP P MAR
1390 0119 125EF6000673      NA,DIVIDE CJS S F D,QREG DIN ; R(1)LSW = QREG
1400 ;
1410 ;*****
1420 ;
1430 ;      E=R(0)/(SCALER) TO PREVENT OVERFLOW
1440 ;
1450 011A 00EA76400303      DATA,ERSTAK S F D,NOP MAR ; MAR=ERSTAK E0=R(0)/4 MSW
1460 011B E000C2C02703      CONT S,ZA F,SUR D P,PROG NOIO ; GET SCALER
1470 011C 311F80400713 DURB13 NA,DURB12 CJP,N D,NOP NOIO
1480 011D E000B742470F      CONT S,ZB F SRA P,,ACCB NOIO
1490 011E 311CB2C22773      NA,DURB13 CJP S,ZB F,SUR D P NOIO
1500 011F E000C6442782 DURB12 CONT S,ZA F D,NOP P,ACCB DOUT
1510 ;
1520 ;*****
1530 ;
1540 0120 00F376400303      DATA,LPCDEX S F D,NOP MAR ; A(1)=K/(SCALER)
1550 0121 E000F6400203      CONT S F D,NOP MDIN ;
1560 ;
1570 ;*****
1580 ;
1590 ;      A(1)=K/(SCALER)
1600 ;
1610 0122 E000A6C24703      CONT S,ZQ F D P,,ACCB NOIO ; GET K
1620 0123 E000C2C02703      CONT S,ZA F,SUR D P,PROG NOIO
1630 0124 312780400713 DURB18 NA,DURB15 CJP,N D,NOP NOIO
1640 0125 E000B742470F      CONT S,ZB F SRA P,,ACCB NOIO
1650 0126 3124B2C22773      NA,DURB18 CJP S,ZB F,SUR D P NOIO
1660 0127 A000A68467F2 DURB15 CRTN S,ZQ F D,RAHA P,ACCB,ACCC DOUT ; ACCC=K, OUTPUT A(J)=K/(SCALER)
1670 ;
1680 ;-----
1690 ;      SUBROUTINE ERSAVE -
1700 ;
1710 ;      ENTER - ERROR IS LOCATED IN ACCA
1720 ;
1730 ;      EXIT - ERROR IN (ERRX) LOCATION
1740 ;      ... -(ERRX)=(ERRX)+1
1750 ;
1760 ;      USES - ACCA, ACCB, (ERRX), ((ERRX)+1)
1770 ;
1780 0128 00EB76400303 ERSAVE DATA,ERRX S F D,NOP MAR ; MAR(ERRX
1790 0129 E00080430603      CONT D,NOP DIN ; DIN( ERRX)

```

```

1800 012A E000F0690782      CONT S F,ADD,C1 D,NOP DOUT ; (ERRX)<(ERRX)+1
1810 012B E000F6400303      CONT S F D,NOP MAR ; (ERRX)=(ERRX)+1
1820 012C A000C64227F2      CRTN S,ZA F D,NOP P DOUT ; SAVE ERROR
1830                          ;
1840                          ;
1850                          ;      SUBROUTINE AJCALC -----
1860                          ;
1870                          ;      ENTER - USES AJ 0<J(I-1 LOCATED IN LPCINDEX BUFFER
1880                          ;      ..... FROM (LPCINDEX))(LPCINDEX)+1
1890                          ;      .      K=ACCC ON ENTRY
1900                          ;
1910                          ;      EXIT - LEAVES NEW AJ FOR J = 1 TO I-1, AND K=A(I)
1920                          ;
1930                          ;      USES - ACCE (I), ACCD (J), ACCC (K), QREG, ACCA
1940                          ;      ..... MCND, AUTOHIGH (AJ), AUTLOW (A (I-J) ), ACCE (I), ACCB (LPCDEX)
1950                          ;
1960 012D 00F376400303 AJCALC DATA,LPCDEX S F D,NOP MAR
1970 012E 11F4F4C24643      NA,ACQUIRE CJS,IRQ S F,SUS D P,,ACCB DIN ; ACCB=LPCINDEX BASE - 1
1980 012F 000176C28703      DATA,1 S F D P,,ACCD NOID ; J=1
1990                          ;
2000 0130 E0009044A303      CONT S,AB F,ADD D,NOP P,ACCB,ACCE MAR ; A(I) ADDRESS TO MAR
2010                          ;
2020                          ;*****
2030                          ;
2040                          ;      OUTPUT A(I)=K/(SCALER)
2050                          ;
2060 0131 E000C6C72703      CONT S,ZA F D P,ACCC,MCND NOID ; MCND=A(I)
2070 0132 E000C2C02703      CONT S,ZA F,SUR D P,PROG NOID ; GET SCALER TO ACCA
2080 0133 313680400713 DURB16 NA,DURB17 CJP,N D,NOP NOID
2090 0134 E000R743270F      CONT S,ZB F SRA P,,MCND NOID
2100 0135 3133B2C22773      NA,DURB16 CJP S,ZB F,SUR D P NOID
2110 0136 E000C6522782 DURB17 CONT S,ZA F D,NOP P,MCND DOUT ; A(I)=K/4
2120                          ;*****
2130 0137 E000C6C72703      CONT S,ZA F D P,ACCC,MCND NOID ; MCND=K
2140                          ;
2150                          ;      AJ=AJ-K*A(I-J)  J=1 TO I-1
2160 0138 E00090448303 DURB4  CONT S,AB F,ADD D,NOP P,ACCB,ACCD MAR ; MAR=A(J) ADDRESS
2170 0139 E000F6C2C603      CONT S F D P,,AUTOHIGH DIN ; AUTOHIGH = A (J)
2180 013A E000C6CA2703      CONT S,ZA F D P,ACCE NOID ; ACCA = I
2190 013B E00092E82703      CONT S,AB F,SUR,C1 D P,ACCD NOID ; ACCA = I-J
2200                          ;
2210 013C E000C2422703      CONT S,ZA F,SUR D,NOP P NOID ; TEST FOR I=2
2220 013D 314890442353      NA,DURB7 CJP,Z S,AB F,ADD D,NOP P,ACCB MAR ; MAR = A(I-J) ADDRESS, BRA IF I=2
2230 013E E000F6C2E603      CONT S F D P,,AUTLOW DIN ; AUTLOW = A (I-J)
2240 013F 124AC60C2773      NA,MULT CJS S,ZA F D,QREG P,AUTOHIGH NOID ; QREG(AUTOHIGH
2250 0140 121EA6400773      NA,ROUND CJS S,ZQ F D,NOP NOID
2260 0141 E000946E2782      CONT S,AB F,SUS,C1 D,NOP P,AUTLOW DOUT ; A(I-J)=A(I-J)*K*(J)
2270 0142 124AC60E2773      NA,MULT CJS S,ZA F D,QREG P,AUTLOW NOID ; A(I-J)*K=X
2280 0143 121EA6400773      NA,ROUND CJS S,ZQ F D,NOP NOID
2290 0144 E00090448303      CONT S,AB F,ADD D,NOP P,ACCB,ACCD MAR ; MAR = A(J) ADDRESS
2300 0145 E000946C2782      CONT S,AB F,SUS,C1 D,NOP P,AUTOHIGH DOUT ; A(J) = A(J) - A(I-J)*K
2310                          ;
2320                          ;      IF J=J+1 AND THEN 2J=I THEN CENTER, 2J>I THEN END

```

```

2330
2340 0146 B0DBR0E28763 NA,DURBB CJPP,OVR S,ZB F,ADD,C1 D P,,ACCD NOIO ; J=J+1, START AGAIN IF OVERFLOW
2350 0147 E000C7C82703 CONT S,ZA F RZU P,ACCD NOIO ; ACCA = 2 * J
2360 0148 E00092EA2703 CONT S,AB F,SUR,C1 D P,ACCE NOIO ; 2*J - I = ACCA
2370 0149 3138B2C22713 NA,DURB4 CJP,N S,ZB F,SUR D P NOIO ; TEST FOR I=ODD
2380
2390 014A A00090484353 CRTN,Z S,AB F,ADD D,NOP P,ACCD,ACCB MAR ; MAR = J + LPCDEX BASE
2400 014B 124AF6000673 DURB7 NA,MULT CJS S F D,GREG DIN ; GREG = A(J) = A(I-J)
2410 014C 121EA6400773 NA,ROUND CJS S,ZQ F D,NOP NOIO
2420 014D A000D46227F2 CRTN S,DA F,SUS,C1 D,NOP P DOUT ; A(J) = A(J) - A(J)*K
2430 ;
2440 ;
2450 ; SUBROUTINE REVISEX - (LPCDEX)+PR, AUTOX+PR+1, (REFLDEX)+PR
2460
2470 014E 00F376400303 REVISEX DATA,LPCDEX S F D,NOP MAR ; (LPCDEX)+PR
2480 014F 000B76C22603 DATA,PR+1 S F D P DIN
2490 0150 E000D0422782 CONT S,DA F,ADD D,NOP P DOUT
2500 0151 00F276400303 DATA,REFLDEX S F D,NOP MAR ; (REFLDEX)+PR
2510 0152 E00080400603 CONT D,NOP DIN
2520 0153 E000D0422782 CONT S,DA F,ADD D,NOP P DOUT
2530 0154 00F476400303 DATA,AUTOX S F D,NOP MAR ; (AUTOX)+2PR+2
2540 0155 E00090E22603 CONT S,AB F,ADD,C1 D P DIN
2550 0156 A000D06227F2 CRTN S,DA F,ADD,C1 D,NOP P DOUT
2560 ;
2570 ;
0010 ;
0020 ;
0030 ; THIS ALGORITHM MAKES USE OF A0 AND B1 WHICH ARE AVAILABLE
0040 ; ON THE STACK AT A1, A0, B1, B0 RESPECTIVELY.
0050 ; THESE VALUES WERE CALCULATED IN THE DATA ACQUISITION ROUTINE.
0060 ;
0070 ; REGISTERS USED ARE
0080 ; LPP(START OF SPEECH BUFFER)
0090 ; NCND AUTOHIGH AUTOLOW QREG ACCA ACCB ACCC
0100 ; ACCD(K) ACCE (M)
0110 ;
0120 ; MEMORY LOCATIONS USED ARE:
0130 ; A0 A1 B0 B1 AX0 AX1 BX00DD BX10DD BX0EVEN BX1EVEN AUTOX
0140 ;
0150 ; RETURNS - R(0), R(1),...,R(PR)
0160 ;
0170 FRAME EQU 200 ;NUMBER OF SAMPLES
0180 PR EQU 100 ; NUMBER OF PREDICTOR COEFFICIENTS
0190 ;
0200 ; THE FOLLOWING PORTION - AX = A0 BXEVEN = A0 BXODD = B1
0210 ; - AUTOLOW,AUTOHIGH = A1, A0
0220 ;
0230 0157 00FE76400303 KENDAL DATA,A0 S F D,NOP MAR ; MAR=A0
0240 0158 00F076400203 DATA,AX0 S F D,NOP MDIN ; DIN=(A0) MAR=AX0
0250 0159 E000F6C2E782 CONT S F D P,,AUTOLOW DOUT ; DOUT=DIN AUTOLOW=DIN
0260 015A 00EE76400303 DATA,BXEVEN S F D,NOP MAR ; MAR=BXEVEN
0270 015B E000F6400782 CONT S F D,NOP DOUT ; DOUT=DIN
0280 015C 00FF76400303 DATA,A1 S F D,NOP MAR ; MAR=A1
0290 015D 00F176400203 DATA,AX1 S F D,NOP MDIN ; DIN=(A1) MAR=AX1

```

```

0300 015E E000F6C2C782      CONT S F D P,,AUTOHIGH DOUT ; DOUT=DIN AUTOHIGH=DIN
0310 015F 00EF76400303      DATA,RX1EVEN S F D,NOP MAR ; MAR=RX1FVEN
0320 0160 E000F6400782      CONT S F D,NOP DOUT ; DOUT=DIN
0330
0340                          ;
0350                          ;
0360 0161 00FC76400303      DATA,B0 S F D,NOP MAR ; MAR=B0
0370 0162 00EC76400203      DATA,BX00DD S F D,NOP MDIN ; MAR=BX00DD DIN=(B0)
0380 0163 E000F6400782      CONT S F D,NOP DOUT ; B0 >BX00DD
0390 0164 00FD76400303      DATA,B1 S F D,NOP MAR ; MAR=B1
0400 0165 00ED76C22203      DATA,RX10DD S F D P MDIN ; B1 > EX10DD
0410 0166 E000F6400782      CONT S F D,NOP DOUT ; MSW B1
0420 0167 E000B8C28703      CONT S,ZB F,AND D P,,ACCD NOIO ; K=0
0430
0440                          ;*****
0450                          ;
0460                          ;
0470                          ;
0480 0168 11B9FBC2A773 AUT1  NA,RXZERO CJS S F,AND D P,,ACCE NOIO ; M = 0 RX = 0
0490 0169 11BD78C24773      NA,LEVEN CJS S F,AND D P,,ACCE NOIO ; LOAD AUTOHIGH,AUTOLOW=-(AX+BXEVEN), ACCR=0=2M
0500 016A E00090DA4303 AUT2  CONT S,AB F,ADD D P,LPP,ACCB MAR ; ACCB, MAR = ACCB+LPP (2M) = MAR
0510 016B E00090648203      CONT S,AB F,ADD,C1 D,NOP P,ACCB,ACCD MDIN ; DIN = X(2M) MAR = ACCB+ACCD+1 MAR= 2M+K
0520 016C E000F6000703      CONT S F D,QREG NOIO ; QREG = DIN DIN = X(2M+K+1)
0530 016D 11FAE0000643      NA,ACQUIRE CJS,IRQ S,DR F,ADD D,QREG DIN ; X1=X(2M)+X(2M+K+1)
0540 016E E000C0642303      CONT S,ZA F,ADD,C1 D,NOP P,ACCB MAR ; MAR = ACCB+1 MAR=2M+1
0550 016F E000F6C32603      CONT S F D P,,MCND DIN ; MCND = DIN DIN = X(2M+K)
0560 0170 E00090448303      CONT S,AB F,ADD D,NOP P,ACCB,ACCD MAR ; MAR = ACCB+ACCD DIN = X(2M+1)
0570 0171 124A00032673      NA,MULT CJS S,DA F,ADD D P,MCND,MCND DIN ; X2=X(2M+1) + X(2M+K)
0580
0590 0172 11AFB0E2A773      NA,RXADD CJS S,ZB F,ADD,C1 D P,,ACCE NOIO ; ACCE = ACCE+1 M=M+1
0600 0173 00C855684703      DATA,FRAME S,DA F,SUS,C1 RZD P,ACCD,ACCB NOIO ; ACCB = (M-K)/2
0610 0174 E000946A4703      CONT S,AB F,SUS,C1 D,NOP P,ACCE,ACCB NOIO ; ACCE-ACCB
0620 0175 316AC7CA4713      NA,AUT2 CJP,N S,ZA F RZU P,ACCE,ACCB NOIO ; ACCB = M *2
0630 0176 11BB80400773      NA,RXSAVE CJS D,NOP NOIO
0640
0650                          ;*****
0660                          ;
0670                          ;
0680                          ;
0690 0177 000A54682703      DATA,PR S,DA F,SUS,C1 D,NOP P,ACCD NOIO ; PR - K
0700 0178 316C80400713      NA,AUTDONE CJP,N D,NOP NOIO
0710
0720                          ;
0730                          ;
0740 0179 11E2C6482773      NA,BX0DDN CJS S,ZA F D,NOP P,ACCD NOIO ; LOOK AT K
0750
0760                          ;*****
0770                          ;
0780                          ;
0790                          ;
0800                          ;
0810                          ;
0820 017A 11B9F8C2A773      NA,RXZERO CJS S F,AND D P,,ACCE NOIO ; RX=0, M=0
0830

```

```

0840 ; LOAD AUTOHIGH,AUTCLOW= -BXODD -AX + X(N-1-K)*X(N-1)
0850 ;
0860 0178 119BF8C24773 ; NA,IODD CJS S F,AND D P,,ACCB NOIO ; ACCB=2M=0
0870 ;
0880 017C E00090DA4303 AUT6 CONT S,AR F,ADD D P,LPP,ACCB MAR ; ACCB = ACCB+LPP MAR = ACCB
0890 017D E00090648203 CONT S,AB F,ADD,C1 D,NOP P,ACCB,ACCD MDIN ; DIN = X(2M) MAR = ACCB+ACCD+1
0900 017E E000F6000703 CONT S F D,RREG NOIO ; QREG = X(2M)
0910 017F E000E0000603 CONT S,DQ F,ADD D,QREG DIN ; QREG = DIN+QREG
0920 0180 E000C0642303 CONT S,ZA F,ADD,C1 D,NOP P,ACCB MAR ; MAR = ACCB+1
0930 0181 11F4F6C32643 NA,ACQUIRE CJS,IRQ S F D P,,MCND DIN ; X(2M+1)
0940 0182 E00090448303 CONT S,AB F,ADD D,NOP P,ACCB,ACCD MAR ; DIN = X(2M+K) MAR = ACCB+ACCD
0950 0183 124AD0D32673 NA,MULT CJS S,DA F,ADD D P,MCND,MCND DIN ; MCND = MCND+DIN
0960 ;
0970 0184 11AFB0E2A773 ; NA,RXADD CJS S,ZB F,ADD,C1 D P,,ACCE NOIO ; ACCE = ACCE+1 M=N+1
0980 ;
0990 ; REPEAT IF M+1 .GE.(N-K-1)/2
1000 ;
1010 0185 00C855484703 ; DATA,FRAME S,DA F,SUS R7D P,ACCD,ACCB NOIO ; ACCB = (N-ACCD-1)/2
1020 0186 E000944A4703 ; CONT S,AB F,SUS D,NOP P,ACCE,ACCB NOIO ; ACCE-ACCB
1030 0187 317CC7CA4713 ; NA,AUT6 CJP,N S,ZA F RZU P,ACCE,ACCB NOIO ; ACCB = ACCF*2
1040 ;
1050 ; *****
1060 ;
1070 0188 11BR80400773 ; NA,RXSAVE CJS D,NOP NOIO
1080 ;
1090 ; UPDATE AX & BXEVEN - A(K)=A(K+1)
1100 ;
1110 0189 11D6C6482773 ; NA,AXNEW CJS S,ZA F D,NOP P,ACCD NOIO ; LOOK AT K
1120 018A 11E4C6482773 ; NA,BXEVENN CJS S,ZA F D,NOP P,ACCD NOIO ; LOOK AT K
1130 ;
1140 018B 3168C6482773 ; NA,AUT1 CJP S,ZA F D,NOP P,ACCD NOIO ; START EVEN PORTION AGAIN WITH NEW K
1150 ;
1160 ; *****
1170 ;
1180 018C A00080400773 AUTDONE CRTN D,NOP NOIO ; END OF AUTOCORRELATION
1190 ; SUBROUTINE LEVEN-
1200 ; LOADS AUTOIGH,AUTOLOW = -(BXEVEN+AX)
1210 ;
1220 018D 00EE76400303 LEVEN NA,BXOEVEN S F D,NOP MAR
1230 018E E000F6C2E603 CONT S F D P,,AUTOLOW DIN
1240 018F 00EF76400303 DATA,BXIEVEN S F D,NOP MAR
1250 0190 E000F6C2C603 CONT S F D P,,AUTOHIGH DIN
1260 0191 00F076400303 DATA,AX0 S F D,NOP MAR
1270 0192 00F176400203 DATA,AX1 S F D,NOP MDIN
1280 0193 E000D0CEE703 CONT S,DA F,ADD D P,AUTOLOW,AUTOLOW NOIO
1290 0194 3196D0ECC633 NA,LEV1 CJP,C S,DA F,ADD,C1 D P,AUTOHIGH,AUTOHIGH DIN
1300 0195 E000R2C2C703 CONT S,ZR F,SUR D P,,AUTOHIGH NOIO
1310 0196 E000BEC2E703 LEV1 CONT S,ZB F,ENOR D P,,AUTOLOW NOIO
1320 0197 E000P0E2E703 CONT S,ZR F,ADD,C1 D P,,AUTOLOW NOIO
1330 0198 319ABCC2C733 NA,LEV2 CJP,C S,ZB F,ENOR D P,,AUTOHIGH NOIO
1340 0199 A000R0400773 CRTN D,NOP NOIO
1350 019A A000R0E2C773 LEV2 CRTN S,ZB F,ADD,C1 D P,,AUTOHIGH NOIO
1360 ;
1370 ; LODD - LOADS AUTOHTGH,AUTOLOW = X(N-1-K) * X(N-1)-AX-BXODD

```

```

1380 ;
1390 019B 00EC76400303 LODD MA,BXGDD S F D,NOP MAR
1400 019C E000F6C2E603 CONT S F D P,,AUTOLOW DIN
1410 019D 00ED76400303 DATA,BX10DD S F D,NOP MAR
1420 019E E000F6C2C603 CONT S F D P,,AUTOHIGH DIN
1430 019F 00F076400303 DATA,AXG S F D,NOP MAR
1440 01A0 00F176400203 DATA,AX1 S F D,NOP MDIN
1450 01A1 E000D0CEE703 CONT S,DA F,ADD D P,AUTOLOW,AUTOLOW NOIO
1460 01A2 31A4D0ECC633 MA,L0DD1 CJP,C S,DA F,ADD,C1 D P,AUTOHIGH,AUTOHIGH DIN
1470 01A3 E000B2C2C703 CONT S,ZB F,SUR D P,,AUTOHIGH NOIO
1480 01A4 E000BEC2E703 LODD1 CONT S,ZB F,ENCR D P,,AUTOLOW NOIO
1490 01A5 E000R0E2E703 CONT S,ZB F,ADD,C1 D P,,AUTOLOW NOIO
1500 01A6 31A8BEC2C733 MA,L0DD2 CJP,C S,ZB F,ENCR D P,,AUTOHIGH NOIO
1510 01A7 31A980400773 MA,L0DD3 CJP D,NOP NOIO
1520 01A8 E000B0E2C703 LODD2 CONT S,ZB F,ADD,C1 D P,,AUTOHIGH NOIO
1530 ;
1540 ;
1550 01A9 00C750DA2303 LODD3 DATA,FRAME-1 S,DA F,ADD D P,LPP MAR
1560 01AA E000F6C32603 CONT S F D P,,MCND DIN ; GET X(N-1)
1570 01AB E00092E82303 CONT S,AB F,SUR,C1 D P,ACCD MAR ; GET X(N-1-K)
1580 01AC 124AF6000673 MA,MULT CJS S F D,GREG DIN
1590 01AD 11AF80400773 MA,RXADD CJS D,NOP NOIO
1600 01AE A00080400773 CRTN D,NOP NOIO ;
1610 ;
1620 ;
1630 ; AUTOHIGH AUTOLOW
1640 ; . + ACCA QREG
1650 ; -----
1660 ; AUTOHIGH AUTOLOW
1670 ;
1680 01AF E000B0CEE703 RXADD CONT S,AQ F,ADD D P,AUTOLOW,AUTOLOW NOIO
1690 01B0 A00090E2C733 CRTN,C S,AB F,ADD,C1 D P,,AUTOHIGH NOIO
1700 01B1 A000B2C2C773 CRTN S,ZB F,SUR D P,,AUTOHIGH NOIO ; RESTORE VALUE
1710 ;
1720 ;
1730 ; ( ACCB ) > QREG LSW
1740 ; ( ACCB-1 ) > ACCA MSW
1750 ;
1760 01B2 E000C2442303 FILLAQ CONT S,ZA F,SUR D,NOP P,ACCB MAR
1770 01B3 E000C6442203 CONT S,ZA F D,NOP P,ACCB MDIN
1780 01B4 E000B0400603 CONT D,NOP DIN
1790 01B5 A000F6000773 CRTN S F D,QREG NOIO
1800 ;
1810 ; MSW ADDRESS IN MAR & ACCB
1820 ; (ACCB)QREG LSW
1830 ; (ACCB+1) MCND MSW
1840 ;
1850 01B6 E000F6C32603 FILLMQ CONT S F D P,,MCND DIN
1860 01B7 E00080624303 CONT S,ZB F,ADD,C1 D,NOP P,,ACCB MAR ; CONTENTS OF NEXT LOC.
1870 01B8 A000F6000673 CRTN S F D,QREG DIN ; MULTIPLIER
1880 ;
1890 ;
1900 ; RX = 0
1910 ;

```



```

1920 01B9 E00086C2C703 RXZERO  CONT S,ZB F,AND D P,,AUTOHIGH NOIO
1930 01BA A000B8C2E773  CRTN S,ZB F,AND D P,,AUTOLOW NOIO
1940
1950 ; RXSAVE - 32 BITS OF RX ARE SAVED
1960 ; . - AUTOHIGH IS SAVED AT CURRENT 2K AND
1970 ; ..... AUTOLOW IS SAVED AT 2K+1 LOCATION IN THE
1980 ; ..... AUTOX BUFFER.
1990 ; . - AUTOX INDEX IS OBTAINED FROM STACK AND NOT
2000 ; . ALTERED.
2010 ; . - PREDICTOR INDEX K(ACCD) IS INCREMENTED AT THE END OF THIS
2020 ; . ROUTINE.
2030 ;
2040 ;SCALING
2050 01BB E000C6482703 RXSAVE  CONT S,ZA F D,NOP P,ACCD NOIO ; CHECK K=0?
2060 01BC 31C480400753  NA,RXSV1 CJP,Z D,NOP NOIO ; SET NORM IF ZERO
2070 ;
2080 01BD 00E876400303  DATA,AUTSCA S F D,NOP MAR
2090 01BE E000F6C22603  CONT S F D P DIN
2100 01BF 31D0C60E2753  NA,RXSV2 CJP,Z S,ZA F D,REG P,AUTOLOW NOIO
2110 01C0 E00082C22703 RXSV5  CONT S,ZB F,SUR D P NOIO ; SCALE=SCALE-1
2120 01C1 31C38782C75F  NA,RXSV4 CJP,Z S,ZB F D,SLA P,,AUTOHIGH NOIO
2130 01C2 31C080400773  NA,RXSV5 CJP D,NOP NOIO
2140 01C3 31D0A6C2E773 RXSV4  NA,RXSV2 CJP S,ZQ F D P,,AUTOLOW NOIO
2150 ;
2160 ;SET  SCALE
2170 ;
2180 01C4 E00088C22703 RXSV1  CONT S,ZB F,AND D P NOIO ; NORMREG=0
2190 01C5 E000C60E2703  CONT S,ZA F D,REG P,AUTOLOW NOIO
2200 01C6 E0008782C70F  CONT S,ZB F D,SLA P,,AUTOHIGH NOIO
2210 01C7 31F1C64C2713  NA,OVFL1 CJP,N S,ZA F D,NOP P,AUTOHIGH NOIO
2220 01C8 31CC80400713 NSET2  NA,NSET1 CJP,N D,NOP NOIO
2230 01C9 E0008782C70F  CONT S,ZB F D,SLA P,,AUTOHIGH NOIO
2240 01CA E00080E22703  CONT S,ZB F,ADD,C1 D P NOIO ; SHIFT COUNT + 1
2250 01CB 31C8C64C2773  NA,NSET2 CJP S,ZA F D,NOP P,AUTOHIGH NOIO
2260 ;
2270 01CC E0008702C70F NSET1  CONT S,ZB F DSRA P,,AUTOHIGH NOIO
2280 01CD 7FFF56CC703  DATA,7FFFH S,DA F,AND D P,AUTOHIGH,AUTOHIGH NOIO ; RESTORE SIGN
2290 ;
2300 01CE 00E876400303  DATA,AUTSCA S F D,NOP MAR
2310 01CF E000A682E782  CONT S,ZQ F D,RAMA P,,AUTOLOW DOUT ; DEPOSIT SCALE FACTOR
2320 ;
2330 01D0 00F476400303 RXSV2  DATA,AUTOX S F D,NOP MAR ; MAR = AUTOX
2340 01D1 E000C70284703  CONT S,ZA F RZU P,ACCD,ACCB NOIO ; ACCB = K *2 DIN = (AUTOX)
2350 01D2 E000D0C44263  CONT S,DA F,ADD D P,ACCB,ACCB MDIN ; INDEX+2K
2360 01D3 E00080AC4722  CONT S,ZB F,ADD,C1 D,RAMA P,AUTOHIGH,ACCB DOUT
2370 01D4 E000C6442303  CONT S,ZA F D,NOP P,ACCB MAR ; R(K) NSW ADDRESS
2380 01D5 A000B0AE87F2  CRTN S,ZB F,ADD,C1 D,RAMA P,AUTOLOW,ACCD DOUT ; K=K+1, AUTOLOW OUT
2390 ;
2400 ;
2410 ; CALCULATE NEW AXEVFN USING K-1 INSTEAD OF K
2420 ;
2430 ; X1=X(N-K+1) * X(N-K)
2440 ;
2450 01D6 00C850DA4703 AXNEW  DATA,FRAME S,DA F,ADD D P,LPP,ACCB NOIO ; ACCB = N+LPP

```

```

2450 0107 11B692EB4373      NA,FILLHQ CJS S,AB F,SUR,C1 D P,ACCD,ACCB MAR ; ACCB = ACCB-ACCD
2470 0108 124AB4E32773      NA,MULT CJS S,ZB F,SUS,C1 D P,,MCND NOIO ; NEGATE VALUE
2480      ;
2490      ;      A(K)=A(K-2)-X1
2500      ;
2510 0109 00F076C24303      DATA,AX0 S F D P,,ACCB MAR ; MAR = AX0 ACCB = AX0
2520 010A E003C0642203      CONT S,ZA F,ADD,C1 D,NOP P,ACCB MDIN ; DIN = (AX0) MAR = AX1
2530 010R E000E0000703      CONT S,DQ F,ADD D,QREG NOIO ; QREG = DIN+QREG DIN = (AX1)
2540 010C 31DED0E22633      NA,AUT3 CJP,C S,DA F,ADD,C1 D P DIN ; ACCA = DIN+ACCA+1
2550 01DD E000R2C22703      CONT S,ZB F,SUR D P NOIO ; ACCA = ACCA-1
2560 01DE E003C6422782 AUT3  CONT S,ZA F D,NOP P DOUT ; DOUT = ACCA
2570 01DF E000C6442303      CONT S,ZA F D,NOP P,ACCB MAR ; MAR = ACCB
2580 01E0 A000A64007F2      CRTN S,ZQ F D,NOP DOUT ; DOUT = QREG
2590
2600      ;
2610 01E1 F00058400000      DATA,0F000H S,DA F,AND D,NOP
2620      ;      CALCULATE NEW BXEVEN & BXODD
2630      ;      TWO ENTRY POINTS ARE PROVIDED.
2640      ;
2650 01E2 00EC76C2E703 BXODD  DATA,BXODD S F D P,,AUTOLW NOIO ; PREPARE ADDRESS
2660 01E3 31E6C2C84773      NA,BX1 CJP S,ZA F,SUR D P,ACCD,ACCB NOIO ; K=K-1
2670      ;
2680 01E4 00EE76C2E703 BXEVEN  DATA,BXEVEN S F D P,,AUTOLW NOIO
2690 01E5 E003C2C84703      CONT S,ZA F,SUR D P,ACCD,ACCB NOIO ; K=K-1
2700      ;
2710      ;      X1=X(K-1)*X(K-2)
2720      ;
2730 01E6 A00090DA4753 BX1    CRTN,Z S,AB F,ADD D P,LPP,ACCB NOIO ; REAL LOCATION K-1
2740 01E7 11B6B2C24373      NA,FILLHQ CJS S,ZB F,SUR D P,,ACCB MAR ; K=K-1 LOCATION
2750 01E8 124AB4E32773      NA,MULT CJS S,ZB F,SUS,C1 D P,,MCND NOIO ; NEGATE VALUE
2760 01E9 E003C64E2303      CONT S,ZA F D,NOP P,AUTOLW MAR ; BX0 ADDRESS
2770      ;
2780      ;      B(K)ODD = B(K-2) - X1
2790      ;
2800 01EA E000C06E2203      CONT S,ZA F,ADD,C1 D,NOP P,AUTOLW MDIN ; DIN = (BXODD) MAR = AUTOLW-1
2810 01EB E000E0000703      CONT S,DQ F,ADD D,QREG NOIO ; QREG = DIN+QREG DIN = (BX1ODD)
2820 01EC 31EED0E22633      NA,AUT3 CJP,C S,DA F,ADD,C1 D P DIN ; ACCA = DIN+ACCA+1
2830 01ED E000R2C22703      CONT S,ZB F,SUR D P NOIO ; ACCA = ACCA+1
2840 01EE E000C6422782 AUT3  CONT S,ZA F D,NOP P DOUT ; DOUT = ACCA
2850 01EF E000C64E2303      CONT S,ZA F D,NOP P,AUTOLW MAR ; MAR = AUTOLW
2860 01F0 A000A64007F2      CRTN S,ZQ F D,NOP DOUT ; DOUT = QREG
2870
2880      ;      THE OVERFLOW ROUTINE FLASHES THE STATUS LIGHTS
2890      ;
2900 01F1 CFFF80400703 OVFL1  NA,OFFFH LDCT D,NOP NOIO ; LOAD DELAY
2910 01F2 91F280400703 OVFL1  NA,OVFL1 RPCT D,NOP NOIO
2920 01F3 31F180400773      NA,OVFL1 CJP D,NOP NOIO,,1 ; FLASH LIGHTS
-----
0310      ;
0320      ;
0330      ;      DATA ACQUISITION INTERRUPT ROUTINE
0340      ;      -- DAQ IS DONE ON AN INTERRUPT BASIS INTERNAL TO THE LPCMK.
0350      ;      -- INPUT                                --OUTPUT
0360      ;      SPEECH SAMPLE,                                WINDOWED SPEECH SAMPLE, A0, B1
0370      ;      HANNING WINDOW                                LPP REG.= NEWBUFFER

```

```

0080 ;
0090 SPEECH EQU OFFH
0100 01F4 123380400773 ACQUIRE NA,STACK CJS D,NOP NOIO ; SAVE MCND, ACCA, QREG, ACCB, ACCC
0110
0120 01F5 00FF76400303 DATA,SPEECH S F D,NOP MAR ; GET SAMPLE
0130 01F6 E000F6C22403 CONT S F D P DIN,INTA ; LOAD SAMPLE
0140 01F7 007F5CC22703 DATA,7FH S,DA F,EOR D P,ACCA,ACCA NOIO
0150 01F8 008058422703 DATA,80H S,DA F,AND D,NOP P,ACCA NOIO
0160 01F9 31FC80400753 NA,POS CJP,Z D,NOP NOIO
0170 01FA FF0056C22703 DATA,0FF00H S,DA F D P NOIO ;
0180 01FB 31FD80400773 NA,ACQ6 CJP D,NOP NOIO
0190 01FC 00FF58C22703 POS DATA,0FFH S,DA F,AND D P NOIO ;
0200
0210 ; THE HANNING WINDOW IS OBTAINED FROM MEMORY
0220 ; AND SAVED IN THE Q-REG. IF NECESSARY, THE
0230 ; RESET IS DONE.
0240
0250 01FD 00F976400303 ACQ6 DATA,ENDHAM S F D,NOP MAR ; MAR=ENDHAM DIN((SPEECH)
0260 01FE E000D2952203 CONT S,DA F,SUR D,RAMA P,HAMLOC,MCND MDIN ; END OF BUFFER ?
0270 01FF 1220B0E34753 NA,RESET CJS,Z S,ZB F,ADD,C1 D P,,HAMLOC NOIO ; RESET IF END.
0280 0200 E000F6C32603 CONT S F D P,,MCND DIN ; GET HANNING WINDOW
0290
0300 ;*****
0310 ; DO NOT OUTPUT UNWINDOWED SAMPLE
0320 ;
0330 ; CONT S,ZB F,ADD,C1 D,RAMA P,PROG,PROG MAR ; UNWINDOWED SAMPLE BUFFER
0340
0350 0201 1247C6022773 NA,MULTAC CJS S,ZA F D,QREG P NOIO ; SPEECH SAMPLE, DO NOT OUTPUT SAMPLE
0360
0370 0202 121EA6400773 NA,ROUND CJS S,ZQ F D,NOP NOIO ; ROUND OFF ANSWER
0380 0203 E000B0B0C303 CONT S,ZB F,ADD,C1 D,RAMA P,DACBUF,DACBUF MAR ;PREPARE ADDRESS
0390 0204 E000C6C32782 CONT S,ZA F D P,ACCA,MCND DOUT ; MCND=ACCA DOUT(ACCA
0400 0205 C20AC6542703 NA,ODD LDCT S,ZA F D,NOP P,HAMLOC NOIO ; SEQREG=ODD TST(DACBUFF AUTSCA=ACCC
0410 0206 721480400723 NA,EVEN JRP,Q0 D,NOP NOIO ; ODD OR EVEN INDEX
0420 0207 81F180400763 ACQ8 NA,OVFL1 CJPP,OV R D,NOP NOIO ; CHECK FOR OVERFLOW
0430 0208 123080400773 NA,UNSTACK CJS D,NOP NOIO ; UNSTACK QREG,ACCA,ACCB,MCND, ACCC
0440 0209 A00080400773 CRTN D,NOP NOIO
0450
-----
0460 ;
0470 ; DDD - M2=X(I) , A(I)=M1*X(I)+A(I)
0480 ;
0490 020A E000C6D38703 DDD CONT S,ZA F D P,MCND,M2 NOIO ; M2=MCND
0500 020B 124AC6162773 NA,MULT CJS S,ZA F D,QREG P,M1 NOIO ; QREG=M1 )SUB (MULT)
0510 ;
0520 020C 00F776400303 DATA,A0BUF S F D,NOP MAR ; MAR=A0BU
0530 020D 00F876C24603 DATA,A1BUF S F D P,,ACCB DIN
0540 020E E000E0000782 CONT S,DQ F,ADD D,QREG DOUT
0550 020F 3212C6442333 NA,ACQ4 CJP,C S,ZA F D,NOP P,ACCB MAR
0560 0210 E00080400603 CONT D,NOP DIN ; GET A1BUFFER
0570 0211 3207D0C227F2 NA,ACQ8 CJP S,DA F,ADD D P DOUT ; A1BUF = A1BUF+MSW
0580 ;
0590 0212 E00080400603 ACQ4 CONT D,NOP DIN
0600 0213 3207D0E227F2 NA,ACQ8 CJP S,DA F,ADD,C1 D P DOUT ; A1BUF =A1BUF+MSW+1
0610 ;
-----

```

```

0620 ; EVEN - M1=X(I) , B(1)=M2*X(I)+B(1)
0630 ;
0640 0214 E000C6182703 EVEN CONT S,ZA F D,QREG P,M2 NOIO ; QREG=M2
0650 0215 124AC6D36773 NA,MULT CJS S,ZA F D P,MCND,M1 NOIO ; M1=MCND
0660 ;
0670 0216 00F576400303 DATA,B0BUF S F D,NOP MAR ; GET A0BUF
0680 0217 00F676C24603 DATA,B1BUF S F D P,,ACCB DIN ; ACCB=B1BUF
0690 0218 E000E0000782 CONT S,DQ F,ADD D,QREG DOUT
0700 0219 321CC6442333 NA,ACG5 CJP,C S,ZA F D,NOP P,ACCB MAR
0710 021A E00080400603 CONT D,NOP DIN ; GET B1BUF VALUE
0720 021B 3207D0C227F2 NA,ACG8 CJP S,DA F,ADD D P DOUT ; B1BUF=B1BUF+MSW
0730 ;
0740 021C E00080400603 ACQ5 CONT D,NOP DIN
0750 021D 3207D0C227F2 NA,ACG8 CJP S,DA F,ADD,C1 D P DOUT ; B1BUF=B1BUF+MSW+1
0760 ;
0770 ; ROUNDING OFF SUBROUTINE
0780 ; ACCA IS INCREMENTED IF QREG.GE.1/2
0790 ;
0800 021E A00080E22713 ROUND CRTN,N S,ZB F,ADD,C1 D P NOIO ; ROUND UP
0810 021F A00082C22773 CRTN S,ZB F,SUR D P NOIO ; RESTORE VALUE
0820 ;
0830 ; RESETS
0840 ; ----- M1, M2 = 0
0850 ; ----- DACBUF = OTHER BUFFER START
0860 ; ----- FLAG INDICATOR
0870 ; ----- HAMLOC = HAMSTART
0880 ;
0890 0220 00015C0FF703 RESET DATA,I S,DA F,EOR D P,FLAG,FLAG NOIO ; Y0 SET IF WAS ON BUFF1
0900 0221 3225F6C38723 NA,ACQ2 CJP,Q0 S F,AND D P,,M2 NOIO ; >ACQ2 IF ON BUFF2.
0910 0222 00F876400303 DATA,SRTBUF1 S F D,NOP MAR ; MAR=SRTBUF1
0920 0223 00FA76400203 DATA,SRTBUF2 S F D,NOP MDIN ; GET BUFFER START FOR AUTOCORR.CALC.
0930 0224 3228F6C3C773 NA,ACQ3 CJP S F D P,,DACBUF NOIO ; DACBUF = SRTBUF1
0940 0225 00FA76400303 ACQ2 DATA,SRTBUF2 S F D,NOP MAR ; MAR=SRTBUF2
0950 0226 00F876400203 DATA,SRTBUF1 S F D,NOP MDIN ;
0960 0227 E000F6C3C703 CONT S F D P,,DACBUF NOIO
0970 0228 E000F6C3A603 ACQ3 CONT S F D P,,LPP DIN ; LPP=NEWBUFFER
0980 ;
0990 ; TRANSFER A1BUF,A0BUF TO A1,A0
1000 ; B1BUF,B0BUF TO B1,B0
1010 ;
1020 0229 00F876C26703 DATA,A1BUF S F D P,,ACCC NOIO ; ACCC = A1BUF
1030 022A 00FF76C24703 DATA,A1 S F D P,,ACCB NOIO ; ACCB = A1
1040 022B C003F8C36703 NA,3 LDCT S F,AND D P,,M1 NOIO ; SET UP LOOP
1050 022C E000C2866303 ACQ7 CONT S,ZA F,SUR D,RANA P,ACCC,ACCC MAR ; START OF LOOP MAR=ACCC
1060 ;CLEAR A0BUF & B0BUF
1070 022D E00080400603 CONT D,NOP DIN
1080 022E E000F8400782 CONT S F,AND D,NOP DOUT
1090 ;
1100 022F E000C2844303 CONT S,ZA F,SUR D,RANA P,ACCB,ACCB MAR
1110 0230 922CF6400782 NA,ACQ7 RPCT S F D,NOP DOUT ; DEPOSIT A0 & B1
1120 ;
1130 0231 00C852D54703 DATA,FRAME S,DA F,SUR D P,HAMLOC,HAMLOC NOIO ; RESTORE HAMMING LOC START
1140 0232 A000C9554373 CRTN S,ZA F,ADD,C1 D,RANA P,HAMLOC,HAMLOC MAR
1150 ;

```

```

1160 ; STACK - PUSH ACCA, MCND, ACCB, QREG, ACCC ONTO STACK
1170 ;
1180 0233 00E576400303 STACK DATA,REGSTK S F D,NOP MAR ; MAR=REGSTK
1190 0234 E000C6422782 CCNT S,ZA F D,NOP P DCUT ; DOUT=ACCA
1200 0235 00E476400303 DATA,REGSTK-1 S F D,NOP MAR
1210 0236 E000C6422782 CCNT S,ZA F D,NOP P,MCND DOUT
1220 0237 00E376400303 DATA,REGSTK-2 S F D,NOP MAR
1230 0238 E000A6400782 CCNT S,ZQ F D,NOP DCUT
1240 0239 00E276400303 DATA,REGSTK-3 S F D,NOP MAR
1250 023A E000C6442782 CCNT S,ZA F D,NOP P,ACCB DOUT
1260 023B 00E176400303 DATA,REGSTK-4 S F D,NOP MAR
1270 023C A000C64627F2 CRTN S,ZA F D,NOP P,ACCC DOUT
1280 ;
1290 ; UNSTACK - GETS ACCA, MCND, ACCB, QREG FROM STACK
1300 ;
1310 023D 00E576400303 UNSTACK DATA,REGSTK S F D,NOP MAR ;MAR=REGSTK ACCB<REGSTK-1
1320 023E E000F6C22603 CCNT S F D P DIN ; RESTORE ACCA
1330 023F 00E476400303 DATA,REGSTK-1 S F D,NOP MAR
1340 0240 E000F6C32603 CCNT S F D P, MCND DIN ; RESTORE MCND
1350 0241 00E376400303 DATA,REGSTK-2 S F D,NOP MAR
1360 0242 E000F6000603 CCNT S F D,QREG DIN ; RESTORE Q-REG
1370 0243 00E276400303 DATA,REGSTK-3 S F D,NOP MAR
1380 0244 E000F6C24603 CCNT S F D P,,ACCB DIN ; RESTORE ACCB.
1390 0245 00E176400303 DATA,REGSTK-4 S F D,NOP MAR
1400 0246 A000F6C26673 CRTN S F D P,,ACCC DIN ; RESTORE ACCC
1410 ;
1420 ;
1430 ;
0010 ;
0020 ;
0030 ; VERSION 1.0 JUNE15/1981
0040 ; INPUT OUTPUT
0050 ; MULTIPLIER-QREG - ACCA MOST SIG. WORD
0060 ; MULTPLICAND-R7 - QREG LEAST SIG. WORD
0070 ;
0080 ; MULTAC IS IS MULTIPLY USED BY THE DATA
0090 ; ACQUISITION ROUTINE. THE UPPER 8 BITS OF THE
0100 ; SPEECH SAMPLE CONTAIN NO INFO
0110 ; THE SAMPLE MUST BE IN Q-REG AS MULTIPLIER.
0120 ;
0130 0247 C009F8C22703 MULTAC NA,9 LDCT S F,AND D P NOIO ; DATA ACQ.MULTIPLY
0140 0248 3248C702277F NA,QZER CJP S,ZA F DSRA P NOIO ; TEST Q0 VALUE
0150 ;
0160 ; SQUARE IS AN OPTIONAL ENTRY POINT THAT LETS
0170 ; THE QREG BE SQUARED.
0180 ;
0190 0249 E000A6C32703 SQUARE CONT S,ZQ F D P,,MCND NOIO ; MCND = QREG
0200 ;
0210 024A C00EF902270F MULT NA,0EH LDCT S F,AND DSRA P NOIO ;ACCA = 0 COUNT = 16
0220 024B 3259C6422703 QZER NA,SUBT CJP,QQ0 S,ZA F D,NOP P NOIO ; ERA.IF Q0=1
0230 024C 9248B702270F NA,QZER RPCT S,ZB F DSRA P NOIO ; DOUBLE ARITH.SHIFT LEFT ACCA QREG
0240 024D 324F90400703 NA,SKP1 CJP,QQ0 D,NOP NOIO ;FINAL
0250 024E 3250B6C22773 NA,DONE CJP S,ZB F D P NOIO
0260 024F 325092F22773 SKP1 NA,DONE CJP S,AB F,SUR,C1 D P,MCND NOIO

```

```

0270 0250 3255C6422773 QONE NA,CNT CJP,QQO S,ZA F D,NOP P NOIO ; QO = 1 ?
0280 0251 92489112270F NA,QZER RPCT S,AB F,ADD DSRA P,MCND NOIO ; ACCA = ACCA + MCND
0290 0252 325480400703 NA,SKP2 CJP,QQO D,NOP NOIO ;FTNAL
0300 0253 325DB6C22773 NA,DONE CJP S,ZB F D P NOIO
0310 0254 325D92F22773 SKP2 NA,DONE CJP S,AB F,SUR,C1 D P,MCND NOIO
0320 0255 9250A702270F CNT NA,QONE RPCT S,ZB F DSRA P NOIO ; DOUBLE SHIFT LEFT ACCA
0330 0256 325580400703 NA,SKP3 CJP,QQO D,NOP NOIO
0340 0257 325D90D22773 NA,DONE CJP S,AB F,ADD D P,MCND NOIO
0350 0258 325DB6C22773 SKP3 NA,DONE CJP S,ZB F D P NOIO
0360 0259 92509332270F SUBT NA,QONE RPCT S,AB F,SUR,C1 DSRA P,MCND NOIO ; ACCA = ACCA - MCND ( 2'S COMP )
0370 025A 325C80400703 NA,SKP4 CJP,QQO D,NOP NOIO
0380 025B 325D90D22773 NA,DONE CJP S,AB F,ADD D P,MCND NOIO
0390 025C 325DB6C22773 SKP4 NA,DONE CJP S,ZB F D P NOIO
0400
0410 025D A000C6422773 DONE CRTN S,ZA F D,NOP P NOIO ; RETURNS SIGN OF PRODUCT.
0420 ;
0430 ; TWO QUADRANT
0440 ;
0450 ; NONRESTORING TWO'S COMPLEMENT DIVISION
0460 ; INPUT - DEN>ACCB NUM>MCND,QREG
0470 ; OUTPUT - QUOTIENT>QREG NO REMAINDER
0480 ;
0490 ; DENOMINATOR ALWAYS POSITIVE
0500 ; NUMERATOR AND DENOMINATOR ARE DESTROYED AFTER
0510 ; USE.
0520 025E C00FC6D22703 DIVIDE NA,OFH LDCT S,ZA F D P,MCND NOIO ; COUNT=15 ACCA(MCND PUSH
0530 025F 326380400713 NRS3 NA,NRS1 CJP,N D,NOP NOIO ; JUMP IF ACCA=0
0540 0260 925F93A42707 NA,NRS3 RPCT S,AB F,SUR,C1 RQRU P,ACCB NOIO ; ACCA=ACCA-ACCB
0550 0261 3266AE000713 NA,NRS4 CJP,N S,ZQ F,ENOR D,QREG NOIO ; COM(QREG)
0560 0262 326780400773 NA,NRS2 CJP D,NOP NOIO
0570 0263 925F91842707 NRS1 NA,NRS3 RPCT S,AB F,ADD RQRU P,ACCB NOIO ; ACCA=ACCA+ACCB
0580 0264 3266AE000713 NA,NRS4 CJP,N S,ZQ F,ENOR D,QREG NOIO ; COM(QREG)
0590 0265 326780400773 NA,NRS2 CJP D,NOP NOIO
0600 0266 E000A0200703 NRS4 CONT S,ZQ F,ADD,C1 D,QREG NOIO ;SIGN
0610 0267 800058D32703 NRS2 DATA,8000H S,DA F,AND D P,MCND,MCND NOIO ; SIGN
0620 0268 A000B6122773 CRTN S,AQ F,OR D,QREG P,MCND NOIO
0630 ;
0640 ; END

```

TOTAL ERRORS 0

Appendix C

PROGRAM MEMORY TESTING LISTINGS

```

00010 00001          NAM MEMORY TESTING PROGRAM
00020 0002A 7000          ORG 7000H
00030 00003          1000 A MESSRT EQU $1000 MESSAGE BUFFER
00040 00004          1FFF A MESEND EQU $1FFF
00050 00005          A000 A START EQU $A000 CONTIGUOUS MEMORY TEST AREA
00060 00006          BFFF A END EQU $BFFF
00070 00007          *
00080 00008A 7000 CE 1000 A BEGIN LDX $MESSRT ZERO MESSAGE BUFFER
00090 00009A 7003 FF 70BC A          STX MESS
00100 00010A 7006 4F          CLRA
00110 00011A 7007 A7 00      A LOOP1 STAA 0,X
00120 00012A 7009 08          INX
00130 00013A 700A 8C 2000 A          CPX $MESEND+1
00140 00014A 700D 26 FB 7007          BNE LOOP1
00150 00015A 700F 5F          CLR B
00160 00016A 7010 F7 70B6 A          STAB COUNT
00170 00017A 7013 F7 70B7 A          STAB PAGE
00180 00018A 7016 CE A000 A          LDX $START
00190 00019A 7019 FF 70BA A          STX CURTST
00200 00020A 701C C6 01      A AGAIN LDAB $01 TESTING PATTERN
00210 00021A 701E BD 705A A ROT      JSR CLEAR
00220 00022A 7021 FE 70BA A          LDX CURTST
00230 00023A 7024 E7 00      A          STAB 0,X
00240 00024A 7026 E1 00      A          CMPB 0,X
00250 00025A 7028 27 03 702D          BEQ LOOP2
00260 00026A 702A 7E 706E A          JMP ERR1
00270 00027A 702D CE A000 A LOOP2 LDX $START
00280 00028A 7030 BC 70BA A LOOP3 CPX CURTST
00290 00029A 7033 27 07 703C          BEQ LOOP4
00300 00030A 7035 A6 00      A          LDAA 0,X MEMORY TEST
00310 00031A 7037 27 03 703C          BEQ LOOP4
00320 00032A 7039 BD 70B6 A          JSR ZERR ERROR #3
00330 00033A 703C 08          LOOP4 INX
00340 00034A 703D 8C C000 A          CPX $END+1
00350 00035A 7040 26 EE 7030          BNE LOOP3
00360 00036A 7042 58          ASLB
00370 00037A 7043 24 09 701E          BCC ROT
00380 00038A 7045 7C 70B6 A          INC COUNT
00390 00039A 7048 26 03 704D          BNE LOOP5
00400 00040A 704A BD 70A6 A          JSR DONEPA
00410 00041A 704D FE 70BA A LOOP5 LDX CURTST
00420 00042A 7050 08          INX
00430 00043A 7051 FF 70BA A          STX CURTST
00440 00044A 7054 8C C000 A          CPX $END+1
00450 00045A 7057 26 C3 701C          BNE AGAIN
00460 00046A 7059 3E          WAI
00470 00047          *
00480 00048          * SUBROUTINE CLEAR - CLEARS MEMORY LOCATIONS FROM
00490 00049          *                                A000 TO BFFF AND ABORTS IF CLEAR DOESN'T WORK.
00510 00050A 705A CE A000 A CLEAR LDX $START
00520 00051A 705D 4F          CLRA
00530 00052A 705E A7 00      A CLR1 STAA 0,X
00540 00053A 7060 A6 00      A          LDAA 0,X
00550 00054A 7062 27 03 7067          BEQ CLR2
00560 00055A 7064 7E 70BD A          JMP CLERR CLEARING ERROR
00570 00056A 7067 08          CLR2 INX
00580 00057A 7068 8C C000 A          CPX $END+1
00590 00058A 706B 26 F1 705E          BNE CLR1

```



```

00600 00059A 706D 39          RTS
00610 00060                  *
00620 00061                  * ERR1 SUBROUTINE ERROR IN LOADING TEST PATTERN.
00630 00062                  *
00650 00063A 706E FE 706C A ERR1 LDX MESS
00660 00064A 7071 86 FF A LDAA *$0FF
00670 00065A 7073 A7 00 A STAA 0,X
00680 00066A 7075 86 01 A LDAA $01 ERROR#1
00690 00067A 7077 A7 01 A STAA 01,X
00700 00068A 7079 86 70BA A LDAA CURTST FAULTY LOCATION
00710 00069A 707C A7 02 A STAA 02,X
00720 00070A 707E 86 70BB A LDAA CURTST+1
00730 00071A 7081 A7 03 A STAA 3,X
00740 00072A 7083 E7 04 A STAB 4,X NUMBER STORED BY MISTAKE
00750 00073A 7085 3E WAI
00760 00074                  *
00770 00075                  * ZERR - ERROR NUMBER 2 MEMORY LOCATION AFFECTED
00780 00076                  * BY A WRITE INT THE TEST LOCATION
00800 00077A 7086 C6 02 A ZERR LDAB $2 WRITE ERROR ON DIFFERENT LOCATION
00810 00078A 7088 FF 70BB A STX CURBUF SAVE FAULTY LOCATION POINTER
00820 00079A 708B 20 05 7092 BRA Z1
00830 00080                  *
00840 00081                  * CLERR-THIS MEMORY LOCATION COULD NOT BE CLEARED
00850 00082                  *
00870 00083A 708D C6 03 A CLERR LDAB $3 ERROR#3
00880 00084A 708F FF 70BB A STX CURBUF SAVE ERROR LOCATION
00890 00085A 7092 FE 70BC A Z1 LDX MESS
00900 00086A 7095 63 00 A COM 0,X FLAG BUFFER LOCATION
00910 00087A 7097 E7 01 A STAB 1,X
00920 00088A 7099 A7 02 A STAA 2,X ERRONEOUS NUMBER
00930 00089A 709B 86 70BB A LDAA CURBUF
00940 00090A 709E A7 03 A STAA 3,X
00950 00091A 70A0 86 70B9 A LDAA CURBUF+1
00960 00092A 70A3 A7 04 A STAA 4,X
00970 00093A 70A5 3E WAI
00980 00094                  *
00990 00095                  * DONEPA = INDICATES A SUCCESSFUL PAGE OF TESTING
01000 00096                  *
01020 00097A 70A6 FE 70BC A DONEPA LDX MESS
01030 00098A 70A9 86 70B7 A LDAA PAGE
01040 00099A 70AC A7 00 A STAA 0,X
01050 00100A 70AE 7C 70B7 A INC PAGE PAGE=PAGE+1
01060 00101A 70B1 08 INX
01070 00102A 70B2 FF 70BC A STX MESS
01080 00103A 70B5 39 RTS

01100 00105A 70B6 0001 A COUNT RMB 1
01110 00106A 70B7 0001 A PAGE RMB 1
01120 00107A 70B8 0002 A CURBUF RMB 2
01130 00108A 70B9 0002 A CURTST RMB 2
01140 00109A 70BC 0002 A MESS RMB 2
00110 END
TOTAL ERRORS 00000

```

PAGE 1 MTSTBIT MASM

```
0001 ; MEMORY TEST PROGRAM FOR OPERATION OF THE
0002 ; PROGRAM MEMORY WITH THE LPCMK.
0003 ; THE LPCMK IS OPERATING AT 200NS.
0004 ;
0005 ; ACCA - PRESENT TESTING PATTERN
0006 ; ACCB - PRESENT TESTING LOCATIO
0007 ; MCND - MEMORY LOC. UNDER TEST
0008
0009 0000 FFFF76C24703 START DATA,0FFFFH S F D P,,ACCB NOIO ; TEST LOC.
0010 0001 4FFF88C22373 ZERO1 NA,0FFFH PUSH S,ZB F,AND D P MAR ; ZERO MEM.
0011 0002 E000F8400782 ZERO CONT S F,AND D,NOP DOUT
0012 0003 E00080400603 CONT D,NOP DIN
0013 0004 E000F6400703 CONT S F D,NOP NOIO ;TEST RESULT
0014 0005 300780400753 NA,LOOP CJP,Z D,NOP NOIO
0015 0006 300680400773 ERR1 NA,ERR1 CJP D,NOP NOIO
0016 0007 800080E22303 LOOP RFCT S,ZB F,ADD,C1 D P MAR
0017
0018 ; INITIALIZING
0019
0020 0008 E000F8C32703 CONT S F,AND D P,,MCND NOIO
0021 0009 000176C22703 DATA,1 S F D P NOIO ; PATTERN
0022
0023 ; LOAD PATTERN
0024
0025 000A E000B0E24303 CONT S,ZB F,ADD,C1 D P,,ACCB MAR
0026 000B 100052642703 DATA,1000H S,DA F,SUR,C1 D,NOP P,ACCB NOIO
0027 000C 301980400753 NA,STOP CJP,Z D,NOP NOIO ; END OF TEST?
0028 000D E000C6422782 LOOP2 CONT S,ZA F D,NOP P DOUT ; SAVE PATTERN
0029 000E 300180400653 NA,ZERO1 CJP,Z D,NOP DIN ; END OF PATTERN
0030 000F E000D2622703 CONT S,DA F,SUR,C1 D,NOP P NOIO
0031 0010 3012B7C22753 NA,LOOP1 CJP,Z S,ZB F RZU P NOIO
0032 0011 301180400773 ERR2 NA,ERR2 CJP D,NOP NOIO
0033
0034 ; CHECK ALL OTHER LOCATIONS FOR ZERO
0035
0036 0012 E00094724703 LOOP1 CONT S,AR F,SUS,C1 D,NOP P,MCND,ACCB NOIO
0037 0013 3012B0832353 NA,LOOP1 CJP,Z S,ZB F,ADD,C1 D,RAMA P,MCND,MCND MAR
0038 0014 0FFF54722603 DATA,0FFFH S,DA F,SUS,C1 D,NOP P,MCND DIN
0039 0015 3018F6400713 NA,END2 CJP,N S F D,NOP NOIO
0040 0016 301280400753 NA,LOOP1 CJP,Z D,NOP NOIO
0041 ;
0042 0017 301780400773 ERR3 NA,ERR3 CJP D,NOP NOIO ; MEMORY FAULT
0043 ;
0044 0018 300DF8852373 END2 NA,LOOP2 CJP S F,AND D,RAMA P,ACCB,MCND MAR
0045
0046 0019 301980400773 STOP NA,STOP CJP D,NOP NOIO
0047 END
```

TOTAL ERRORS 0

Appendix D

AMDAHL SIMULATION PROGRAM

AMDAHL FORTRAN LPC TEST PROGRAM - WRITTEN BY Luc Mahieu August, 1981

```
C$JOB WATFIV KEN_MIKOL,NOEXT
CHARACTER CHAR*2(8)
DIMENSION S(200),IS(8),SS(80),ER(128),A(10),RR(128),T(200)
DO 1 I=1,25
READ 100,CHAR;PRINT 100,CHAR
DO 1 J=1,8
CALL CHARIN(CHAR(J),IS(J),2,16)
R=IS(J)/128.;IF(R.GE.1) R=R-2;S(8*I-8+J)=R I CONTINUE
CALL HAMMIN(S,200,0)
PRINT,'THE FRAME IS WINDOWED (HAMMING WINDOW)'
CALL DURBIN(S,200,10,A) 100
FORMAT(1X,A2,1X,A2,1X,A2,1X,A2,1X,A2,1X,A2,1X,A2,1X,A2)
STOP;END
C*****
SUBROUTINE DURBIN(S,N,P,A)
C*****
INTEGER P,IQ(16)
REAL S(N),R(11),K(10),E(10),A(P),B(10)
C***** C
CALCULATION OF CORRELATION COEFFICIENTS C
*****
PRINT,'AUTOCORRELATION METHOD BY DURBIN S ALGORITHM'
PRINT 200;IP1=P+1;IP2=P-1
DO 6 I=1,IP1
M=N+1-I;R(I)=0
DO 6 J=1,M 6 R(I)=R(I)+S(J)*S(J+I-1)
DO 7 I=1,IP1
R(I)=R(I)/2
CALL DEBI(R(I),IQ,1);PRINT,' ' 7 PRINT 300,I-1,R(I),IQ
PRINT 200 C
***** C
DURBIN'S ALGORITHM C
*****
E(1)=R(1)/4;K(1)=R(2)/R(1);A(1)=K(1)/4
CALL DEBI(K(1),IQ,1);PRINT,' ';PRINT 400,1,K(1),IQ
DO 8 I=1,IP2
PRINT 200
E(I+1)=(1-K(I)*K(I))*E(I);K(I+1)=R(I+2)/4
CALL DEBI(E(I+1),IQ,1);PRINT,' ';PRINT 600,E(I+1),IQ
DO 9 J=1,I
B(J)=A(J) 9 K(I+1)=K(I+1)-A(J)*R(I-J+2)
K(I+1)=K(I+1)/E(I+1)
CALL DEBI(K(I+1),IQ,1);PRINT,' ';PRINT 400,I+1,K(I+1),IQ
A(I+1)=K(I+1)/4;CALL DEBI(A(I+1),IQ,1)
PRINT,' ';PRINT 500,I+1,A(I+1),IQ
DO 10 J=1,I
A(I-J+1)=B(I-J+1)-K(I+1)*B(J)
CALL DEBI(A(I-J+1),IQ,1);PRINT,' ' 10 PRINT
500,I-J+1,A(I-J+1),IQ 8 CONTINUE;PRINT 200
RMS=E(P)*(1-K(P));PRINT,'RMS=',RMS 200 FORMAT
('*****') 300
FORMAT(1X,'R(',I2,')= ',F12.7,' = ',4I1,' ',4I1,' ',4I1,' ',4I1) 400
```

```

FORMAT (1X,'K(',I2,')=',F10.7,' = ',
      *4I1,' ',4I1,' ',4I1,' ',4I1) 500  FORMAT(1X,'A(',I2,')= ',F10.7,'
= ',4I1,' ',4I1,' ',4I1,' ',4I1
      *1) 600  FORMAT (1X,'E=',F10.7,' = ',
      ',4I1,' ',4I1,'
      RETURN;END
C*****
SUBROUTINE                                HAMMIN                                (S,N,IMUL)
C*****
DIMENSION S(N)
PI=ARCOS(-1.)
IF (IMUL.EQ.0) THEN
DO 1 I=1,N 1      S(I)=S(I)*(.54-.46*COS(2*PI*(I-1)/(N-1)))
ELSE
DO 2 I=1,N 2      S(I)=S(I)/(.54-.46*COS(2*PI*(I-1)/(N-1)))
ENDIF
RETURN;END
C*****
SUBROUTINE                                DEBI(R,IR,IT)
C*****
INTEGER IR(16)
R=R+IT*2.**(-15);IR(16)=0
IF(R.GE.1.OR.R.LT.-1) THEN
PRINT,'OVERFLOWN TWO S COMPLEMENT NUMBER'
IF (R.GE.1) THEN;IR(1)=0;DO 2 I=2,15 2      IR(I)=1
ELSE;IR(1)=1;DO 3 I=2,15 3      IR(I)=0;ENDIF
RETURN;ENDIF
IR(1)=IFIX(.5-SIGN(.5,R)+.1)
R2=R;IF (IR(1).EQ.1) R2=1+R
DO 1 I=1,14
R2=R2-2.**(-I);IR(I+1)=1
IF(R2.LT.0)THEN;R2=R2+2.**(-I);IR(I+1)=0;ENDIF 1      CONTINUE
RETURN;END
C*****
FUNCTION                                RB(IR)
C*****
INTEGER IR(16)
RB=0.;IF (IR(1).EQ.1) RB=-1.
DO 1 I=2,15 1      IF (IR(I).EQ.1) RB=RB+2.**(-I+1)
RETURN;END
C*****
SUBROUTINE                                CHARIN(CHAR,INT,N,IB)
C*****
CHARACTER CHAR*1(N)
CHARACTER H*1(16)/'0','1','2','3','4','5','6','7','8','9','A','B',
*'C','D','E','F'/
INT=0
DO 2 I=1,N
DO 1 J=1,IB
IF (CHAR(I).EQ.H(J)) THEN;INT=INT+(J-1)*IB**(N-I);GOTO 2;ENDIF 1
CONTINUE 2      CONTINUE
RETURN;END
C*****
SUBROUTINE                                HEXA(I,HEXNUM)

```

```
C*****
  INTEGER I(4)
  CHARACTER HEXNUM*1
  CHARACTER H*1(16)/'0','1','2','3','4','5','6','7','8','9','A','B',
* 'C','D','E','F'/
  J=I(4)+2*I(3)+4*I(2)+8*I(1)
  HEXNUM=H(J+1)
  RETURN;END
```