A SPEECH ANALYSIS SYSTEM FOR USE IN EDUCATIONAL RESEARCH

JOHN ROBERT RUNCHMAN

Submitted for the degree of Doctor of Philosophy

The University of Aston in Birmingham

December 1981

THE UNIVERSITY OF ASTON IN BIRMINGHAM

A SPEECH ANALYSIS SYSTEM FOR USE IN EDUCATIONAL RESEARCH

JOHN ROBERT RUNCHMAN

PhD

1981

SUMMARY

This thesis describes work undertaken in order to fulfil a need experienced in the Department of Educational Enquiry at the University of Aston in Birmingham for speech analysis facilities suitable for use in teaching and research work within the Department. The hardware and software developed during the research project provides displays of speech fundamental frequency and intensity in real time. The system is suitable for the provision of visual feedback of these parameters of a subject's speech in a learning situation, and overcomes the inadequacies of equipment currently used for this task in that it provides a clear indication of fundamental frequency contours as the subject is speaking.

The thesis considers the use of such equipment in several related fields, and the approaches that have been reported to one of the major problems of speech analysis, namely pitch-period estimation. A number of different systems are described, and their suitability for the present purposes is discussed. Finally, a novel method of pitch-period estimation is developed, and a speech analysis system incorporating this method is described. Comparison is made between the results produced by this system and those produced by a conventional speech spectrograph.

Indexing key words:

SPEECH ANALYSIS

PITCH-PERIOD ESTIMATION

FUNDAMENTAL FREQUENCY

ACKNOWLEDGEMENTS

I wish to express my thanks to my Supervisor, Mr Ian Gould of the Computer Centre, and to my Associate Supervisor, Dr John Bradshaw of the Department of Educational Enquiry, for their wise counsel throughout the course of this research project.

I consider myself fortunate to have been supervised jointly in two Departments of the University, as I have been able to enjoy the benefits of discussion with and assistance from members of the academic, administrative and technical staffs of the Computer Centre and the Department of Educational Enquiry, as well as from my fellow students.

I am grateful to Brian Patten, a Producer with the BBC, for permission to use tape-recorded material from Lord Denning's "With Great Pleasure" radio programme, and to Jill Bacon and Paul Rogers for permission to use passages spoken by them in this programme.

Finally, my thanks go to my family for their encouragement, and especially to my wife Sarah for many sacrifices and much help.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND TO THE PROJECT

The research project which forms the subject of this thesis was motivated by a need experienced in the Department of Educational Enquiry at the University of Aston for improved speech analysis facilities to be used in teaching and research work within the Department. The foremost requirement was for a system which would enable trainee counsellors on the Diploma in Counselling in Educational Settings course to be aware of and to modify certain parameters of their speech with a view to improving their communication skills. Teaching requirements included the demonstration of some of the paraverbal aspects of human communication to undergraduate students on the degree course in Human Communication. Research interests included analysis of the roles played by certain parameters of speech in a dyadic communication situation, with particular reference to the counselling situation. Preliminary experiments suggested that the two most important parameters of speech for all these intended application areas were the contours of fundamental frequency and speech intensity against time [B101].

1.2 EXISTING SPEECH ANALYSIS FACILITIES

At the commencement of the present project the main tool used for teaching and research purposes was a Voice Identification Inc. Series 700 Speech Spectrograph [VI02]. This produces wide-band and narrow-band spectrograms, displays of frequency spectrum at a given point in time, and contours of average amplitude for tape-recorded utterances of up to 2.5 seconds duration. The time taken to analyse a 2.5 second portion of speech is approximately 80 seconds.

The speech spectrograph is a widely-used analytical tool which produces detailed results concerning the structure of the speech signal. It is unsatisfactory for speech correction and for teaching purposes for two main reasons. Firstly, the time delay involved in the production of results makes it difficult to relate the resulting spectrogram to the speech act which it represents. Secondly, the richness of detail in the spectrograms makes them very difficult for the inexperienced user to interpret. In cases where the fundamental frequency contour is of interest, students must be instructed to "look at the lowest narrow bar above the baseline" on a narrow-band spectrogram; the presence of a number of harmonics in addition to the fundamental is a constant source of confusion. The accuracy with which fundamental frequency can be measured depends upon the clarity of the spectrograms. Lehiste and Peterson [LI03] suggest that an accuracy of ± 1Hz can be achieved in cases where the 20th harmonic is visible on the spectrogram. The experience of the present author is that accuracy of frequency measurement is usually rather less than this.

1.3 REQUIREMENTS OF THE PROPOSED SYSTEM

A speech analysis system was required which would produce displays in real-time of fundamental frequency and speech intensity contours from a 'live' microphone or from a tape-recording, without the need for special anechoic conditions. While real-time operation was a necessity, a short time delay between input of speech and display of results would be acceptable. There were certain financial restrictions which made it desirable to make use of equipment normally found in a University psychology laboratory. The system must be suitable for use by non-technical students, and must be reasonably portable to allow use in the field. The displays produced must be suitable for small-scale class teaching, must have the ability to combine 'target' and 'trial' contours, and must have provision for hard-copy output. The intended user community would consist of adults free from any serious speech defects, although the ability to analyse defective speech and the speech of children would be an advantage. Experience suggested that the range of fundamental frequencies to be encountered would be between 50Hz and 500Hz. Finally, absolute accuracy was felt to be less important than the presentation of easily-followed contours: the overall pattern was more important than the instantaneous accuracy of fundamental frequency estimation.

1.4 VISUAL FEEDBACK IN A LEARNING SITUATION

1.4.1 Knowledge of results and information feedback

The provision of visual feedback of some parameter of a subject's performance in a learning situation is said to give the subject KNOWLEDGE OF RESULTS [A104,pp26-29]. The traditional reinforcement theory view is that knowledge of results can be regarded as rewarding or punishing the subject, with rewarding results preserving the behaviour which preceded it [A104,pp29-36] [H105,pp204-207]. The more recent CYBERNETIC theory of learning suggests that knowledge of results gives feedback information to the subject [A104,p12]. The role of information in cybernetic learning theory is discussed by Pask [P106,pp17-44].

The present author views the visual speech parameter display system proposed in Section 1.3 above as a tool to be used for the provision of information in a learning situation. Such visual feedback can supplement a subject's auditory perception of speech parameters, which may be inaccurate, providing visual-auditory 'cross-modal transfer' [F107].

1.4.2 Real-time considerations

As mentioned above, Section 1.3, the design objective of the proposed system is to have a short, and constant, time delay between input of the speech signal and display of speech parameter information. The intention behind this is to allow a subject to receive information regarding a complete section of speech within a short time of completing the utterance. Systems requiring greater than 'real-time' for processing suffer from a cumulative delay, after which time a speaker's memory of his articulatory intentions may have faded beyond recall [L108].

Much experimentation has been conducted into the effects of direct and delayed visual feedback upon a subject's performance in tracking tasks. Wargo [W109] found that delays in visual feedback degraded tracking performance, that degradation increased with increasing magnitude of delay, and that little or no adaptation to delayed sensory feedback was apparent after continued and active practice. Smith et al [S110] [S111] found that subjects could adapt to short (<=100mS) delays of visual feedback during target-directed movement. Christina [C112] suggests that the average minimum time required by subjects to process visual feedback ranges from 190mS to 260mS.

The above results have implications for any system in which a subject receives visual feedback while he is performing a task. Rostron and Welbourn [R113] have used a real-time visual display of fundamental frequency to examine the control systems active when a singer attempts to match a note. Rather than attempting to maintain a given frequency, the subject attempts to produce a straight line on the fundamental frequency display. The present author suspects that in such a situation the mechanism used to process visual feedback may play as significant a role as any mechanisms used to control voice fundamental frequency.

Consideration of the theoretical implications of delayed sensory feedback, and any investigations into the cybernetic theory of learning using feedback information are beyond the scope of the present project. The proposed analysis system will, however, produce displays of the sort which have been employed in several different learning situations (see Sections 1.7.1 to 1.7.3).

1.5 PROSODIC FEATURES AND LINGUISTICS

Linguists traditionally describe a spoken language in terms of a hierarchy of units [P114,p2], the smallest being the individual vowel and consonant sounds. These may be combined into syllables, syllables into words, and the words may be used to form sentences.

PHONEMES are the speech sounds that differentiate words. For example, the spoken words 'man', 'ban' and 'pan' differ only in their initial sound, the relevant phonemes being /m/, /b/ and /p/. A phoneme is defined within a particular language system, rather than being defined acoustically, the acoustic features of a phoneme being dependent upon precise context. Each phoneme has associated with it a set of acoustically distinct variants called ALLOPHONES. The term PHONE is used to refer to the general acoustical features of a phoneme, and is not concerned with the individual features of the set of allophones. The features of phonemes are sometimes referred to as SEGMENTAL FEATURES, the phonemes being the basic segments of speech [P114,p4]. The term PROSODIC FEATURES is a general name for the rhythmic and tonal features of speech [P114,p80]. Prosodic features are generally 'suprasegmental' - that is to say, they extend over more than one phoneme segment.

Crystal [C115,p128] defines prosodic features from a phonetic point of view as "vocal effects constituted by variations along the parameters of pitch, loudness, duration, and silence". Crystal further defines as PARALINGUISTIC those vocal features which are "primarily the result of physiological mechanisms other than the vocal cords, such as the direct result of the workings of the pharyngeal, oral or nasal cavities".

Pickett [P114,p80] defines prosodic features in a similar fashion, but includes effects due to the shaping of the vocal tract. There is thus some disagreement concerning the precise boundary between prosodic and paralinguistic features.

STRESS and INTONATION are normally considered to be the most significant of the prosodic features which convey linguistic information. Stress may be used in English to differentiate similar forms that have different meanings. For example, the two spoken phrases:

'That's just in <u>sight</u>'

and 'That's just <u>in</u>sight'

where stress is indicated by underlining, are differentiated by the different placing of stress [P114,p80].

There is a sharp divergence between 'tone languages' and 'intonation languages' [B116,p13]. In a tone language, such as Chinese, one word may differ from another only in the fundamental frequency used during its production. For example, the Chinese word 'li' pronounced with a rising pitch means 'pear'. The same word pronounced with a falling pitch means 'chestnut' [M117]. Intonation languages lack this systematic use of tone, although in most languages pitch can be used to distinguish between one phoneme and another.

The terms 'stress' and 'intonation' are widely used by linguists, although there is some disagreement concerning the precise definition of both terms.

Crystal [C115,Section 3.8] gives examples of the range of viewpoints regarding stress, from the physico-physiological to the psychological (the productive and receptive aspects respectively). Fonagy [F118] defines stress as a function of greater speaking effort; Potter [P119,p63] states that stress "may be measured by instruments precisely"; Trager and Smith

[T120,p36] define stress as "relative strength or loudness"; Bolinger [B121] states that stress is perceived prominence imposed within utterances.

The formerly prevalent view that stress is directly related to intensity of speech has given way more recently to a complex view of stress. This latter view reflects the definition of 'emphasis' by elecutionists:

> "Making a particular word or a sentence stand out prominently emphasises the meaning of the particular sentence. This can be effected in different ways by making changes in the intensity, in the pitch, or in the pace of vocal utterance, by introducing the pause before or after the emphatic word, or by modulating the voice".
>
> [H122,p89]

Ridley [R123,pp65-66] similarly discusses the introduction of emphasis "by stress or by giving extra force to a word", "by pause", "by change of tone or by inflection" and "by change of pace". Several linguists now agree that stress is a complex combination of duration, loudness, pitch and quality [F124] [L125].

There are many references to intonation in the literature, but there would seem to be no precise description of intonation and related effects for English or any other language, and no sound theoretical basis for studies of areas such as the semantics of intonation. Coulthard points out that, while many descriptive linguistic systems rely quite heavily upon intonation for the purpose of categorising utterances or parts of utterances,

> "...no one currently involved in discourse analysis marks intonation continuously in their transcriptions; appeal to intonation is apparently spasmodic, if not, haphazard, and usually occurs when differences are perceived for which there can be no other explanation."
> [C126,p116]

Crystal [C115] makes a major effort towards obtaining a definition of 'prosodic systems' and stating exhaustively the intonational options in English. He aims to describe the formal prosodic contrasts available in English for the expression of differences in meaning. Contrastive features are those whose omission from an utterance would cause a linguistically untrained group of native English speakers to state that the utterance was different in meaning from the original [C115,p127]. Crystal points out that intonation consists of a complex combination of features from different prosodic systems; it has a "very clear centre of pitch contrasts", but has in addition a "periphery of reinforcing (and occasionally contradicting) contrasts of a different order". He concludes that, in the current state of knowledge regarding prosodic features and intonation, it is impossible to make any valid generalisations about meaning.

Nevertheless, some attempts have been made to provide generalised attitudinal meanings for various major intonation patterns. O'Connor and Arnold [O127] specify three roles of intonation. As well as the grammatical roles of dividing longer utterances into grammatically relevant word groups and differentiating between different grammatical functions, they state that

> "...intonation expresses the speaker's attitude, at the moment of speaking, to the situation in which he is placed".
>
> [O127,p4]

O'Connor and Arnold readily admit that no 'tone group' (intonation pattern) is used exclusively with one particular sentence type, such as question, assertion, and so on. They maintain, however, that some sentence types are more likely to be said with one tone group than with another, and that one can meaningfully talk of a 'normal' tone group for a particular structure. They go so far as to provide detailed accounts of attitudinal meanings associated with various tone groups, e.g.

> "...such statements tend to sound soothing, reassuring; they offer the information as a means of setting the listener's mind at rest; no criticism is implied...but there is a hint of great self-confidence or self-reliance on the part of the speaker".
>
> [O127,p62]

The complexity surrounding intonation as mentioned by Crystal, and evidenced by the apparent confusion among many linguists regarding the precise nature of intonation, brings into question the connection between the perceived intonation contour and the physical characteristics of a spoken utterance. Lieberman [L128] investigated this relationship, defining intonation as the "entire ensemble of pitch contours, pitch levels, and stress levels that occur when a sentence is spoken". His experiment involved the use of the Trager-Smith [T120] notation by experienced linguists. The Trager-Smith notation is widely used by linguists, and consists of four pitch levels, three 'terminal junctures' - sustention of pitch, falling pitch and rising pitch -

and various vocal qualifiers to describe the pitch contour of an utterance.

Lieberman set out to discover whether linguists employ an 'objective procedure', considering the physically present acoustic signal, or whether they consider their own 'subjective' judgement of the structure of the sentence, and fill in the Trager-Smith pitch notation that is appropriate to the structure of the sentence (inferred from the words of the sentence and from their knowledge of the language).

Lieberman concluded that there is often no distinct physical basis for the phonemic pitch levels and terminal symbols of the Trager-Smith system. The linguist will frequently infer their presence from his knowledge of the transcriptions that the Trager-Smith system normally uses for certain combinations of words. Moreover, Lieberman found no basis for regarding Trager-Smith pitch levels as perceptual manifestations of either absolute or relative fundamental frequency ranges, with the exception of certain contours which recur quite frequently in normal discourse; it would appear that these contours are perceived as complete entities. When other intonation contours were transcribed, the Trager-Smith notation became inconsistent, and Lieberman found no reasonable relationship to those attributes of the physical signal which is supposedly being transcribed.

Lehiste and Peterson [L103] tried to determine experimentally some of the factors influencing the phonetic realisation of the intonation contour. Their results indicate a number of inconsistencies between the physical signal and the perceived intonation contour. A linguistically significant intonation level may have a wide range of phonetic manifestations. The phonetic quality of the syllabic sound influences the fundamental frequency at which the intonation level is produced. Finally, the initial consonant in a consonant-vowel sequence may influence the fundamental frequency of the vowel following the consonant. Lehiste and Peterson conclude that the "instrumental analysis of intonation emerges as a problem of great complexity".

On this question of instrumental analysis of intonation, John Firth is quoted thus:

> "How can I bring home to these linguists who come to me saying – 'you have a laboratory, why don't you give us an intonation meter?' – How can I tell them that it is impossible to build a physical instrument which measures a linguistic event?"
>
> [D129]

Denes [D129] argues that linguists often use labels with acoustic or articulatory connotations to describe the parameters of language, despite the fact there is no direct relationship between the linguistic and the acoustic/articulatory events of speech, and concludes than an instrument constructed to measure acoustic or articulatory events cannot measure linguistic events.

This conclusion obviously has profound implications for any attempts to relate a visual display of fundamental frequency against time to the intonation contour of that utterance, or to relate frequency and intensity contours to judgements of sentence stress. Further investigation of the relationship between the perceived and the physical characteristics of spoken utterances is beyond the scope of this project; the present author adopts the common assumptions that, for a spoken utterance, perceived intonation is closely related to fundamental frequency, and that sentence stress is often related to speech intensity, and therefore concludes that a visual display of fundamental frequency and speech intensity against time is a useful aid to the study of prosodic features. It should, however, be stressed that the proposed speech analysis system will provide information concerning the physical characteristics of the speech signal, and will not attempt to provide a display of the perceptual attributes of the speech signal.

1.6 PHYSICAL AND PERCEIVED CHARACTERISTICS OF SPEECH

The apparent confusion among some linguists between the physical and perceived characteristics of speech is reflected in the literature on speech analysis and signal processing. As Pickett [P114,p81] points out, in acoustics writings on speech the term 'pitch' means the fundamental frequency of the vocal cord action in producing a glottal sound source. In the hearing sciences, the sensation of pitch is found to depend largely on the fundamental frequency of the sound stimulus, and thus the two are closely related but are not the same thing. It has, for example, been demonstrated [D130,p28] that perceived pitch is a function of the intensity as well as of the frequency of a periodic sound stimulus. In the speech literature, however, 'pitch' and 'fundamental frequency' are used as synonyms.

There is a similar tendency to confuse the terms 'intensity' and 'loudness', the former being a physical attribute of the speech signal, the latter being a subjective perception related to, but distinct from, intensity.

While wishing to standardise a vocabulary for the description of speech production and perception which recognises the basic distinction between physical and perceived characteristics, the present author has nevertheless found that it is necessary to take into account common usage, especially in the case of the use of the term 'pitch' as a physical characteristic. Phrases such as 'pitch synchronous analysis', 'pitch detection' and 'pitchmeter' are undeniably part of the vocabulary of present day speech literature. Wherever possible, however, it is proposed to use the following terminology:

FUNDAMENTAL FREQUENCY - the lowest frequency component of the frequency spectrum of a voiced speech sound, corresponding to the frequency of vibration of the vocal cords.

PITCH-PERIOD - the time taken to complete one cycle of vibration at the fundamental frequency. The term 'pitch-period' is also used to refer to one such cycle.

PITCH - an attribute of the perception of speech in which sounds may be ordered on a scale running from 'low' to 'high' [C115,p108].

INTENSITY - a measure of the power contained in the speech signal.

AMPLITUDE - a measure of the magnitude of displacement of the speech waveform.

LOUDNESS - an attribute of the perception of speech in which sounds may be ordered on a scale running from 'soft to 'loud' [C115,p113].

1.7 FURTHER APPLICATIONS OF SPEECH DISPLAYS

1.7.1 Foreign language learning

The traditional approach to the teaching of foreign language pronunciation is dominated by articulatory phonetics, the teaching of exotic sound production being related specifically to articulatory postures [C131]. The MOTOR THEORY of speech perception is in accord with this traditional approach, stating that sounds are perceived chiefly in terms of their means of production [L132]. Kalikow and Swets [K133] have used an 'automated pronunciation instructor' to teach English pronunciation to Spanish students. This system produces visual displays of tongue position within the mouth.

Fourcin [F134] has proposed an alternative model of speech perception, in which speech perception is seen primarily as a process of auditory pattern perception. This auditory theory has implications for the teaching of foreign language pronunciation, suggesting that this should be based upon the acquisition of appropriate auditory patterns. Abberton and Fourcin [A135] put forward the view that naturalness in speaking a foreign language is heavily dependent upon effective control of its patterns of rhythm and intonation. Patterns of intonation are, however, in large measure different from one language to another; problems in learning a foreign language are perceptual as well as linguistic [F107]. The student must learn to perceive and produce intonation patterns which are alien to him, and then use those patterns appropriately in the new language. Several courses in intonation exist for English as a foreign language [H136] [O127]. These consist of textbooks accompanied by tape recordings of model intonation patterns. Some students, however, are unable to perceive the direction of pitch changes, and consequently have difficulty in producing appropriate intonation patterns, even when presented with a theoretical description or tape-recorded example of the required patterns. Fourcin and Abberton [F107] have used a real-time display of fundamental frequency against time to teach intonation control to such students using visual feedback. James [J137] has used a similar method in a second language teaching situation. Both systems allow the student to compare his own intonation patterns with 'target' patterns, and to monitor his attempts to improve the degree of match.

1.7.2 Analysis of defective speech

Traditional methods of treatment for speech defects require considerable skill on the part of the therapist, both in recognising the nature of the defect, and in devising suitable curative measures and communicating these to the patient:

> "When the teacher is able to recognise the wrong sound and to make it, and when he knows the exact position of the organs of speech for the right sound, he will be able to use his knowledge and practical ability to devise exercises for obtaining the particular sound he wants. Some of these exercises he may find in books, but he will not find all speech defects dealt with in books, and there are remarkably few which treat the subject at all. He may have to make up for himself exercises suitable for overcoming the particular defect he is treating. The methods he uses will depend largely on the intelligence and will power of his patient, for it would be of little use to tell a small child how his tongue should be placed, whereas a grown up pupil may find such information of value"
> [W138]

Fourcin and Abberton [F107] [F139] have used real-time visual display devices in speech therapy, in order to identify the cause of defective speech production, to demonstrate the nature of the problem to the patient, and to allow the patient to correct the defect in a pattern feedback learning situation. Such equipment has also been used in the analysis of pathological conditions of the vocal mechanism [W140], and in a study of laryngeal trauma associated with general anaesthesia [M141].

1.7.3 Speech aids for the deaf

For a subject born deaf or deafened prior to about five years of age, the absence of acoustic feedback of his or her own voice leads to the production of unnatural or even unintelligible speech [B142]. Nickerson and Stevens [N143] classify the deficiencies of the speech of deaf subjects as being due to lack of control of the vocal cords and musculature of the larynx, of breathing, of fundamental frequency, of loudness, of timing and rhythm, and of the velum, the latter deficiency leading to the production of 'nasal' speech. The lack of fundamental frequency control has been the subject of many studies. Observed deficiencies with respect to fundamental frequency include an abnormally high average fundamental [A144] [W145], and the production of unnatural intonation patterns, or in the extreme case a complete lack of intonation, giving the speech a monotonous quality [M146]

Hudgins [H147] reports on two early devices used to give visual feedback of certain properties of speech to the speaker: a stroboscope, which is used to provide an indication of fundamental frequency, and a pneumodeik, which gives a kymographic record of air pressure variations during speech, and which is used to correct nasality and to give an indication of the presence or absence of voicing during the attempted production of consonant sounds. Hudgins states three rules which must be observed in developing visual aids of practical value to the deaf:

(i) the visual pattern must be simple and clear-cut, so that the deaf subject will have no difficulty in understanding it,

(ii) the apparatus which presents these visual patterns must be easy to operate and adaptable to use in the classroom,

(iii) the apparatus must present the patterns while the child is speaking.

Some thirty years later, Pronovost [P148] reports that these three criteria have been met, and details various devices for the visual display of spectral properties and fundamental frequency contours of the subject's speech. Similar surveys of visual display devices are given by Pickett [P149] [P150] [P151].

Devices producing fundamental frequency-against-time contours on a display for the purpose of visual feedback to a deaf speaker are described in [A152] [D153] [F107] [F139].

Willemain [W145] and Stratton [S154] report on devices which provide tactile feedback of fundamental frequency. Advantages claimed for this arrangement are the avoidance of interference with simultaneous lipreading, and the possibility of producing compact, inconspicuous and wearable feedback devices [S154].

Boothroyd [B155] is critical of previous tactile, visual and auditory aids for the deaf, feeling that too much attention has been paid to the engineering aspects of the devices themselves, while insufficient attention has been paid to the application of the devices in teaching deaf subjects to improve their speech. Boothroyd reports on experiments conducted using a system which produces fundamental frequency-against-time contours on a storage oscilloscope, and concludes that such a display offers many advantages over noninstrumental techniques in the teaching of complex fundamental frequency control, such as the intonation of connected speech.

Boothroyd and Decker [B156] state three prerequisites for a deaf subject to learn the production of appropriate fundamental frequency patterns in his speech:

(i) knowledge by the teacher of the fundamental frequency patterns of normal speech

(ii) a means of providing the subject with information about desired fundamental frequency patterns

(iii) a means of providing the subject with information about his own fundamental frequency patterns (i.e. feedback).

CHAPTER TWO

SPEECH PRODUCTION

2.1 PHYSIOLOGY OF SPEECH PRODUCTION

The natural speech production process involves manipulation of organs originally evolved for breathing and eating purposes. Various methods of examination have been applied in order to determine the precise function of these organs, including x-ray [F201] and cinematographic [F202] analysis. The major components of the VOCAL TRACT are illustrated in Figure 2-1, which is a sagittal-plane view of the human head.

The vocal tract proper extends from the lips down to the vocal cords, with a length of approximately 17cm in an adult male [F203]. The maximum cross-sectional area is some 20cm^2 [F203]. An additional chamber, the NASAL CAVITY, extends from the velum to the nostrils, with an average length of 12cm and an average volume of 60cm^3 in an adult male [F203]. For the production of nasal sounds, such as /m/ (m ate), /n/ (n ice) and /ŋ/ (si ng), the nasal cavity is coupled to the vocal tract, allowing sound to radiate from the nostrils. In addition to the nasal cavity, the vocal tract contains two other major cavities, ORAL and PHARYNGEAL.

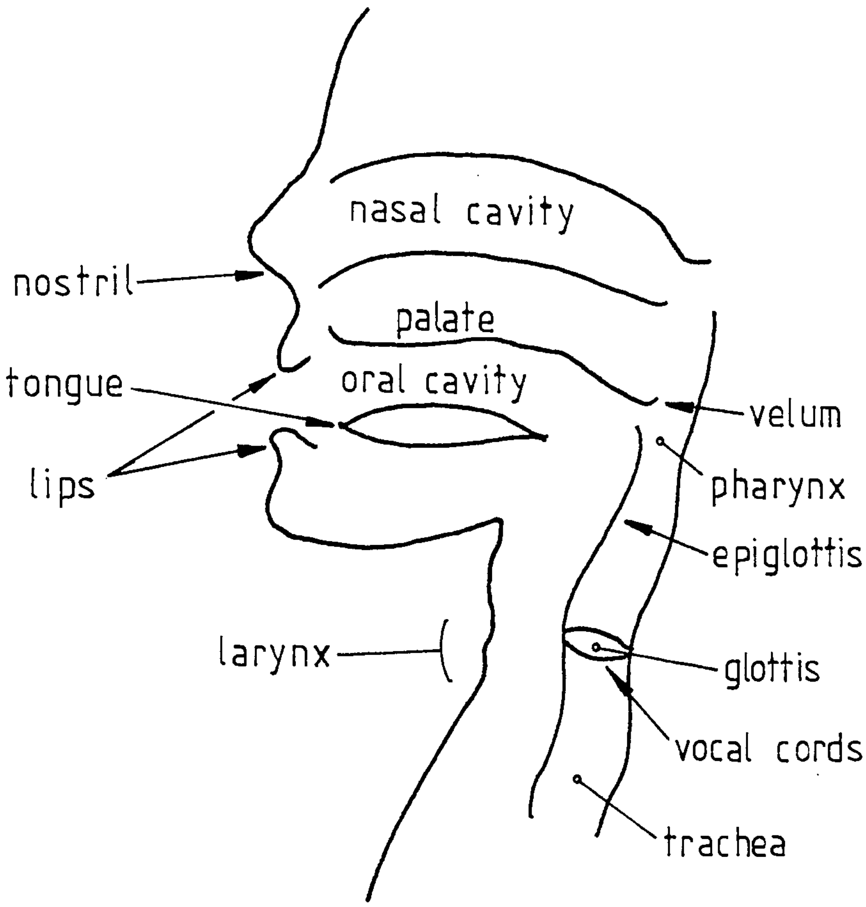


Figure 2-1: The vocal tract, showing major components

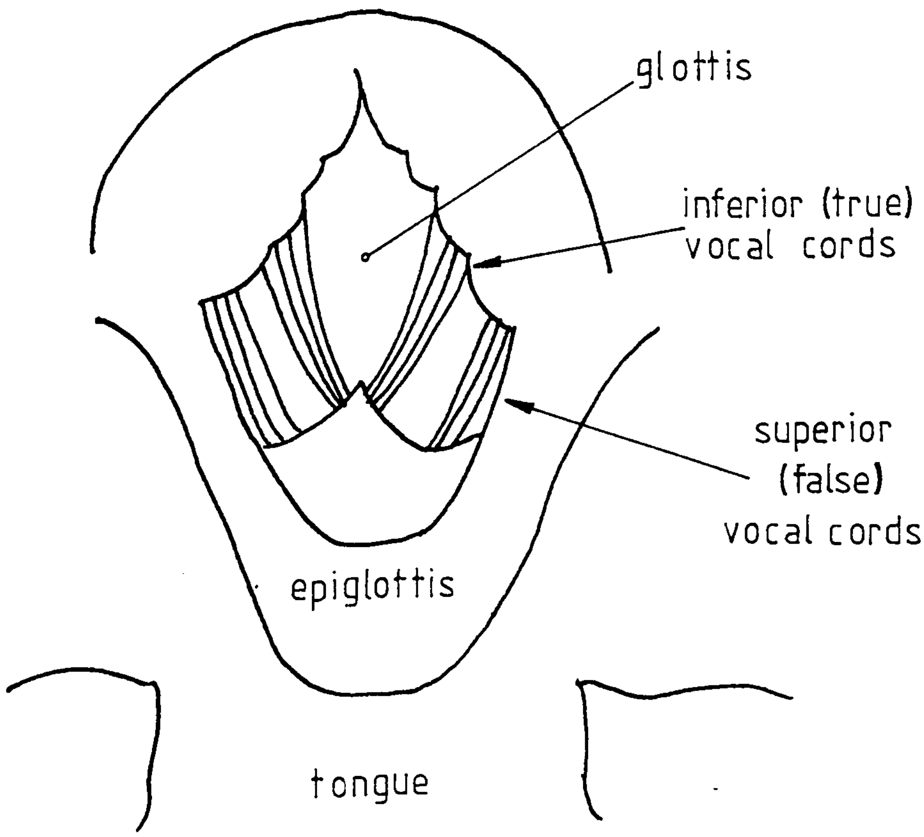


Figure 2-2: The larynx, viewed from above

The LARYNX, which is situated below the pharynx, is illustrated in greater detail in Figure 2-2, which shows the larynx viewed from above. The inferior (true) vocal cords, hereafter referred to simply as the VOCAL CORDS, are two strong bands of fleshy ligament, which may be held apart voluntarily, or allowed to lie closely together. The area between the vocal cords is termed the GLOTTIS. Until puberty there is no marked difference between the larynx of the male and that of the female. Thereafter, the male larynx increases greatly in size, the length of the glottis nearly doubling [D204].

During normal speech production, the lungs give rise to a flow of air which travels up the trachea, passes through the larynx and pharynx, and exits via either the nasal or the oral cavity. This is illustrated in block-diagram form in Figure 2-3. In the case of certain short-duration speech sounds, such as /p/ (p at) and /f/ (f un), the airflow may originate from a 'reservoir' of air held in the oral cavity, rather than from the lungs.

This inaudible flow of air may be rendered audible by modulation introduced in one of three main ways.

VOICED SOUNDS are produced by narrowing the glottis and allowing the vocal cords to vibrate. In this way, the steady airflow from the lungs is converted into a succession of short bursts of air. The frequency of repetition of these bursts of air depends upon the air pressure applied to the larynx and upon the physiological adjustment of the vocal cords, including their

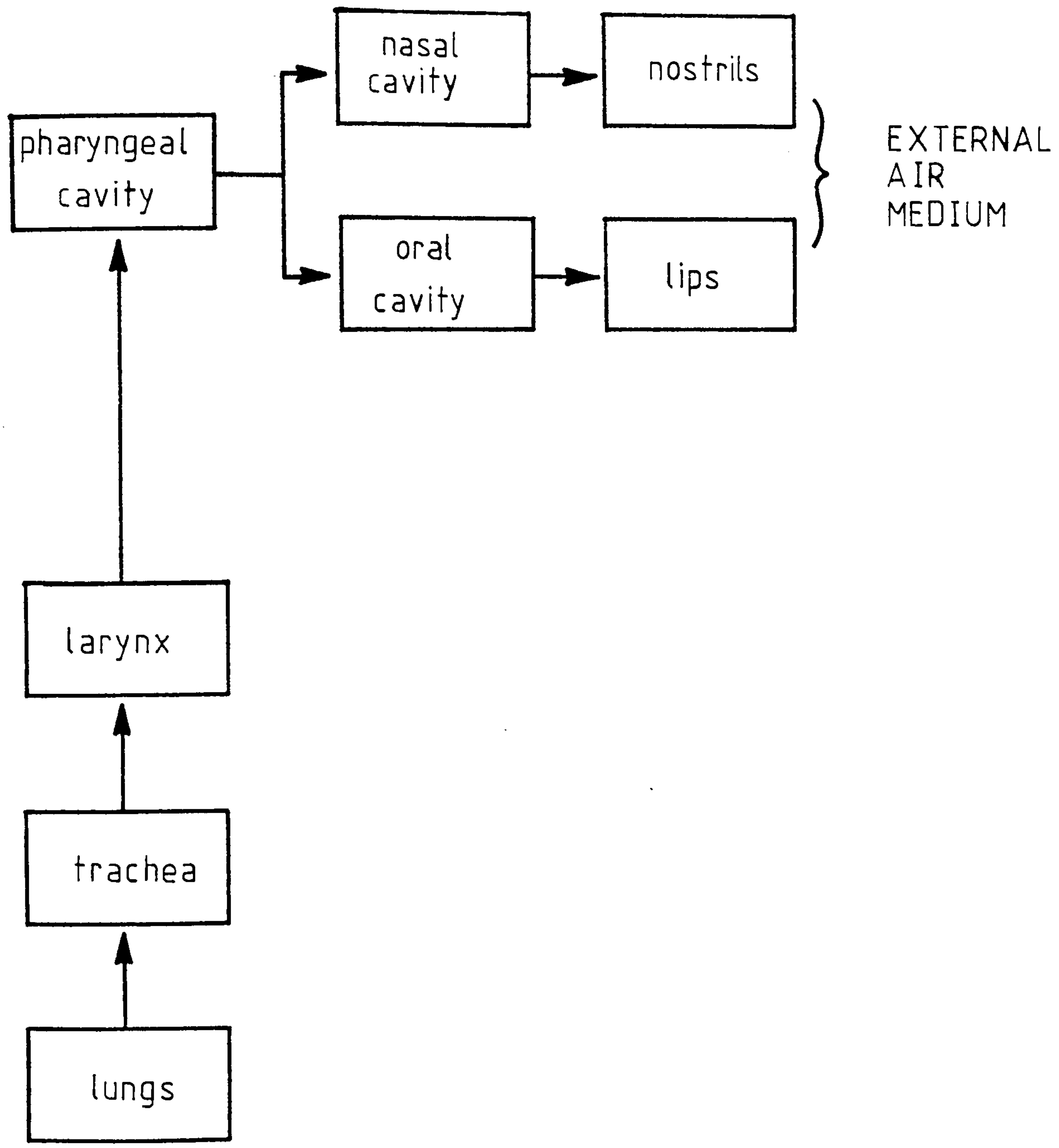


Figure 2-3: Airflow during normal speech production

length, thickness and tension.

FRICATIVE SOUNDS are produced when the steady airflow is forced through a constriction formed at some point in the vocal tract, and becomes turbulent as a result.

PLOSIVE SOUNDS occur when some part of the vocal tract is closed completely, creating a build-up of air pressure, and then abruptly released, causing a sudden burst of air, which is normally accompanied by fricative noise.

Voiced sound production results in quasi-periodic, broad-spectrum pulses; fricative and plosive vocal activities produce aperiodic, broad-spectrum acoustic noise.

All three types of airflow modulation act as excitation sources for the vocal tract which takes the form of a non-uniform acoustic tube, acting as a time-varying filter, which imposes its resonant characteristics upon the sound source. The vocal tract can be characterised by its FORMANTS or natural frequencies, which correspond to resonances in the vocal tract transmission characteristics. These resonances are due to the chain of cavities which form part of the vocal tract. The most significant is the oral cavity, the size and shape of which can be altered within wide limits by movements of the tongue and lower jaw. These movements also affect the size and shape of the pharyngeal cavity. The proportions of the nasal cavity remain reasonably constant.

As well as being affected by the geometry of these cavities, the vocal tract transmission characteristics are dependent upon the acoustic properties of the various surfaces of the vocal tract. Examples of this latter effect are the damping actions of the tongue, cheeks, hard palate and velum.

The major sources of sound radiation into the external air medium are the oral and nasal orifices. Additional radiation, especially at low frequencies, originates from the neck and chest of the speaker.

2.2 CLASSIFICATION OF SPEECH SOUNDS

The speech sounds produced by the quasi-periodic and aperiodic sound sources described above do not correspond directly to the commonly-used classes 'vowel' and 'consonant'. A speech sound may, for example, arise from a combination of quasi-periodic and aperiodic excitation of the vocal tract. A more thorough classification of speech sounds is as follows [F205,Chapter 1]:

Class	Examples
Pure vowels	/u/ (t oo l), /U/ (t oo k)
Dipthongs	/aI/ (t i me)
Transitionals	/w/ (w oo)
Semi-vowels	/l/ (l ate)
Fricative consonants	/z/ (z ero), /f/ (f un)
Stop consonants	/b/ (b at), /p/ (p at)

2.3 THE NATURE OF THE SPEECH WAVEFORM

The time-domain and frequency-domain characteristics of the speech waveform radiated into the external air medium depend upon the nature of the voiced sound source, the unvoiced sound sources, and the frequency response of the vocal tract.

2.3.1 Voiced sound source

The precise characteristics of the sound source at the glottis have been investigated in a variety of ways [F202] [M206] [H207] [F208] [F209] [F210] [F211] [R212] [T213]. Farnsworth [F202] used high-speed cinematography to produce a detailed account of the changes in glottal area with time during voiced speech activity. Fletcher [F208] used this method to determine the glottal area waveform in time. Flanagan [F209] proceeded from Fletcher's waveforms to plot the spectral characteristics of glottal pulses, and later developed a computer model of vocal cord action [F210]. Miller [M206], Holmes [H207] and Rothenberg [R212] have examined the glottal waveform using the inverse filtering technique, in which the resonant characteristics of the vocal tract are neutralised using inverse filter networks. The passage of voiced speech through such a system produces a waveform similar to that at the glottis.

The glottal airflow waveform and harmonic analysis of a typical voiced speech sound (after Miller [M206]) are shown in Figure 2-4. The voiced sound source is a quasi-periodic volume-velocity wave whose spectrum envelope decreases with increasing frequency at a rate of some 12dB/octave in the range 300-2500Hz [F203]. The precise spectral shape depends upon the exact shape of the glottal pulses. The spacing between harmonic components in the glottal frequency spectrum depends upon the repetition rate of glottal pulses and the corresponding fundamental frequency of the glottal waveform.

2.3.2 Unvoiced sound source

Studies of the unvoiced sound source are complicated by several factors. Unvoiced sounds tend to have a much shorter duration than voiced sounds; plosives in particular consist of very short bursts of noise. The unvoiced sound source may be located at one of many different points along the vocal tract. Published studies of unvoiced sounds, and of voiced sounds containing an unvoiced component, have tended to concentrate upon a particular class of consonants, and have in general used the waveform radiated into the external air medium as the basis of analysis. Strevens [S214], Hughes and Halle [H215], and Heinz and Stevens [H216] have investigated fricative sounds; Halle, Hughes and Radley [H217] have detailed the acoustic properties of stop consonants; Fujimura [F218] has published an analysis of nasal consonants; Jassem [J129] has dealt with all classes of consonant occurring in the Polish language. In general, it

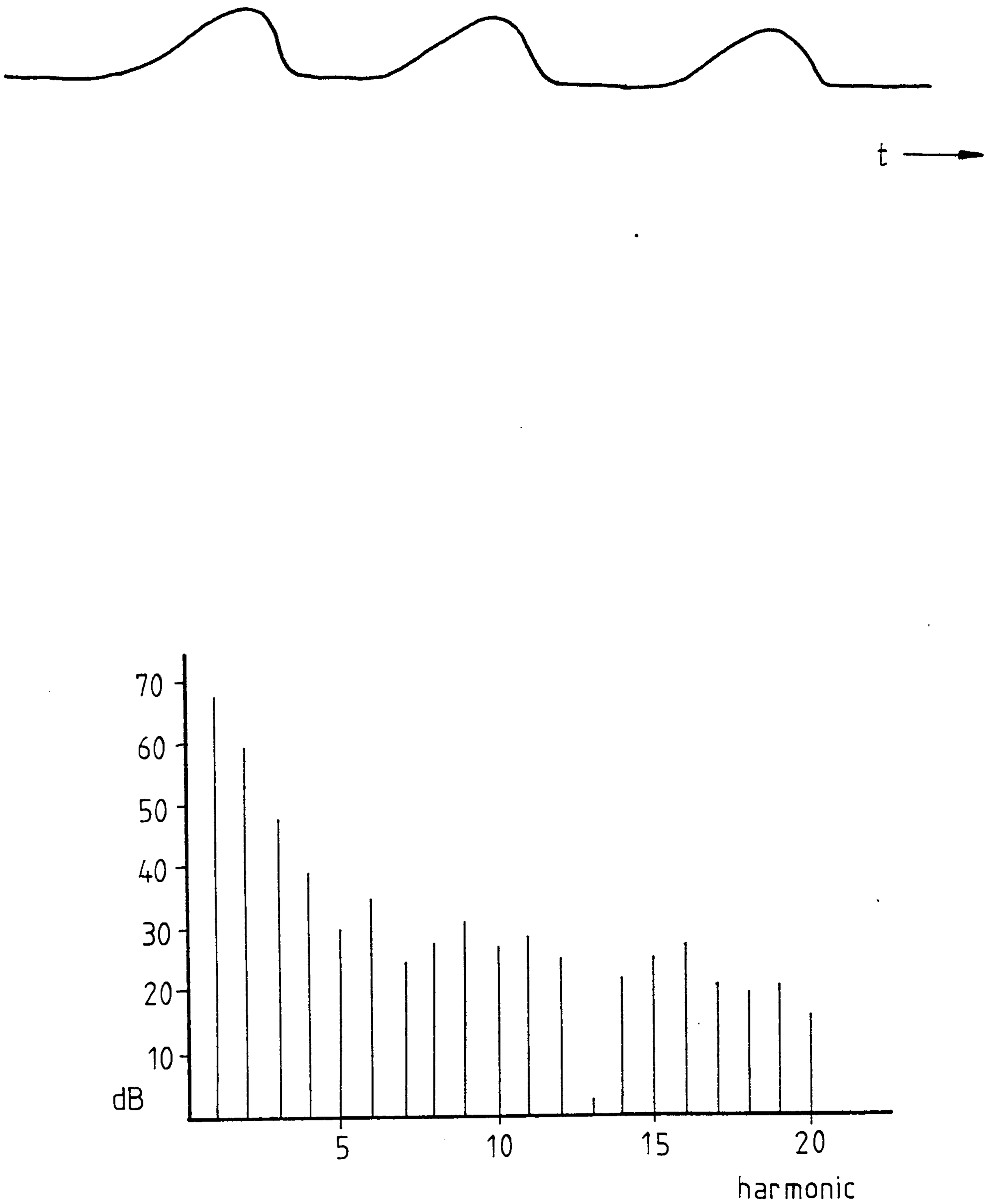


Figure 2-4: Typical glottal waveform and frequency spectrum

appears that the unvoiced sound source exhibits a noisy waveform and a correspondingly flat frequency spectrum.

2.3.3 Vocal tract transfer characteristics

The generally-accepted theory of speech production [F201] [D220] [C221] [F222] views the vocal tract as a filter which emphasises by contrast some of the components of the sound source, namely those close to the resonant frequencies of the vocal tract. This SOURCE-FILTER THEORY of speech production suggests that the spectral properties of the vocal tract transfer characteristics will be imposed upon the sound source, and will therefore be reflected in the spectral characteristics of the radiated speech sounds. The vocal tract transfer characteristics may be determined in a number of ways, including sine-wave excitation of the vocal tract, and inverse filtering [F222].

Figure 2-5 (after Fant [F201], Stevens and House [S223]), illustrates the imposition of vocal tract filter characteristics upon the glottal sound source for various vowel sounds. The glottal source spectrum has a slope of some -12dB/octave; however, voiced sound radiation from the oral orifice increases with increasing frequency by about 6dB/octave [S223], and the glottal spectrum has been modified to give an effective slope of -6dB/octave.

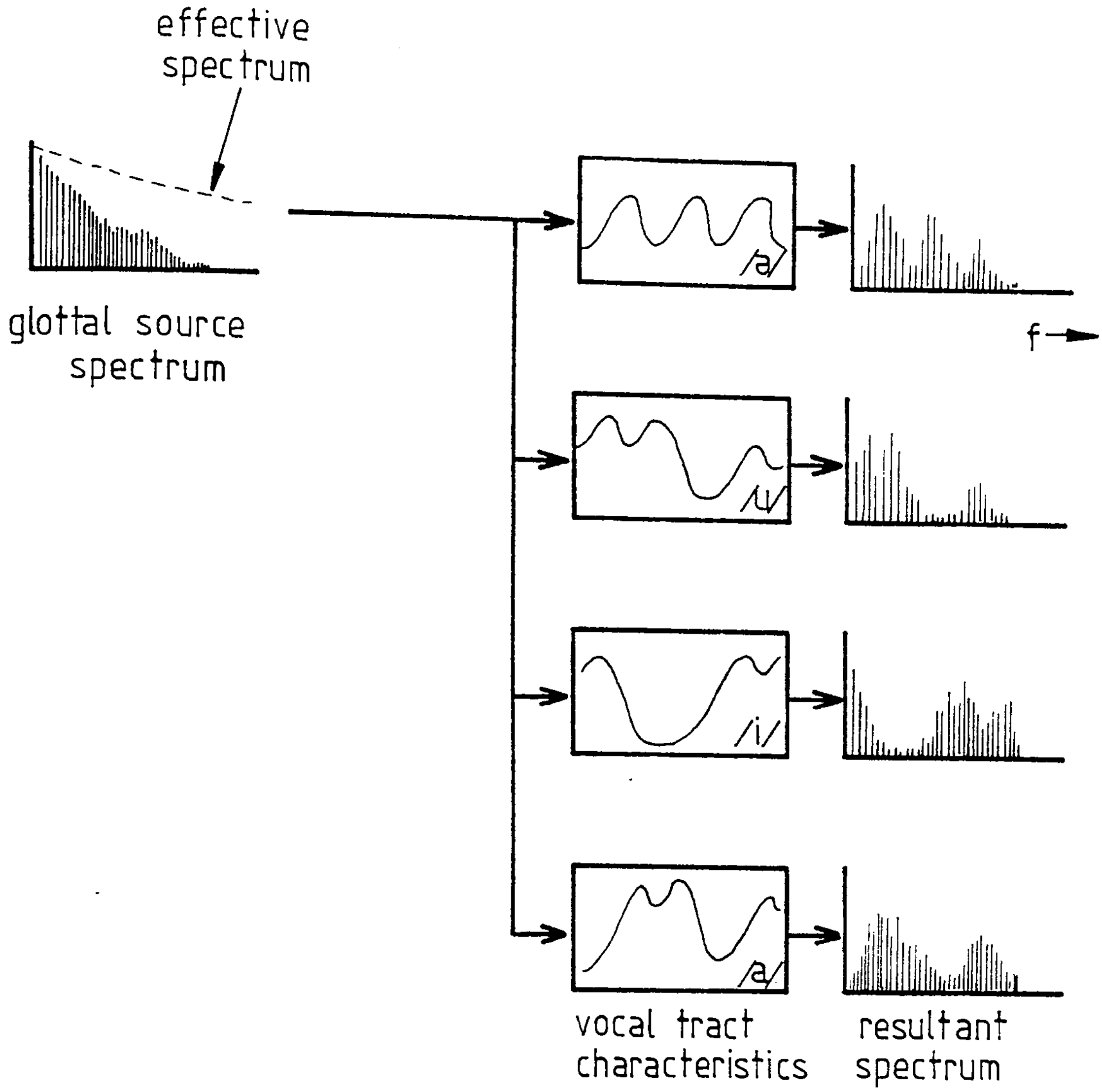


Figure 2-5: Spectral shaping in various vowel sounds

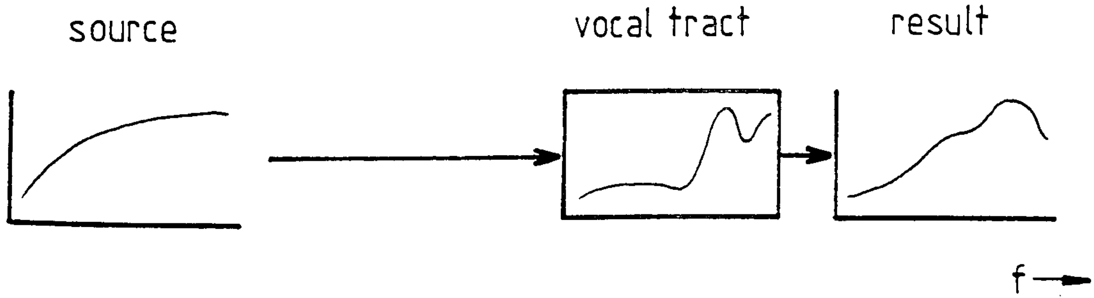


Figure 2-6: Spectral shaping in a typical unvoiced sound

Figure 2-6 (after Flanagan [F224]) illustrates the effect of vocal tract filter characteristics for a typical unvoiced sound.

2.4 MODELS OF THE SPEECH PRODUCTION PROCESS

Various attempts have been made to model the vocal tract using acoustical analogues [F201] [D225] and electrical analogues [F201] [D225] [S226]. The generally-accepted speech production model due to Fant [F201] models the unvoiced and voiced sound sources and the vocal tract shape separately; this is illustrated in Figure 2-7.

This linear model may be adapted so as to be based upon digital signal processing principles, as shown in Figure 2-8 (after Schafer and Rabiner [S227]).

2.5 EXAMPLES OF VOICED AND UNVOICED SPEECH SOUNDS

Figures 2-9 to 2-11 are oscillographic and spectrographic analyses of the word "speech" spoken by an adult male, illustrating various features of the speech sounds.

Figure 2-9 is an oscillogram of the utterance, showing amplitude variations against time. It will be noted that speech has a wide dynamic range, with voiced sounds having a much greater peak amplitude than unvoiced sounds.

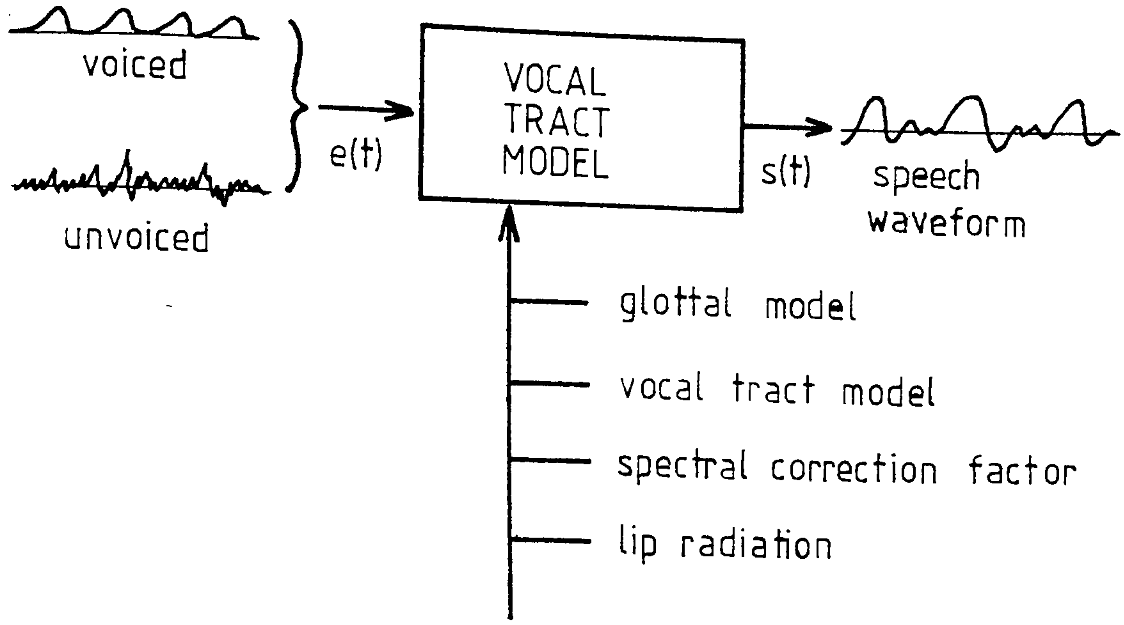


Figure 2-7: Linear speech production model

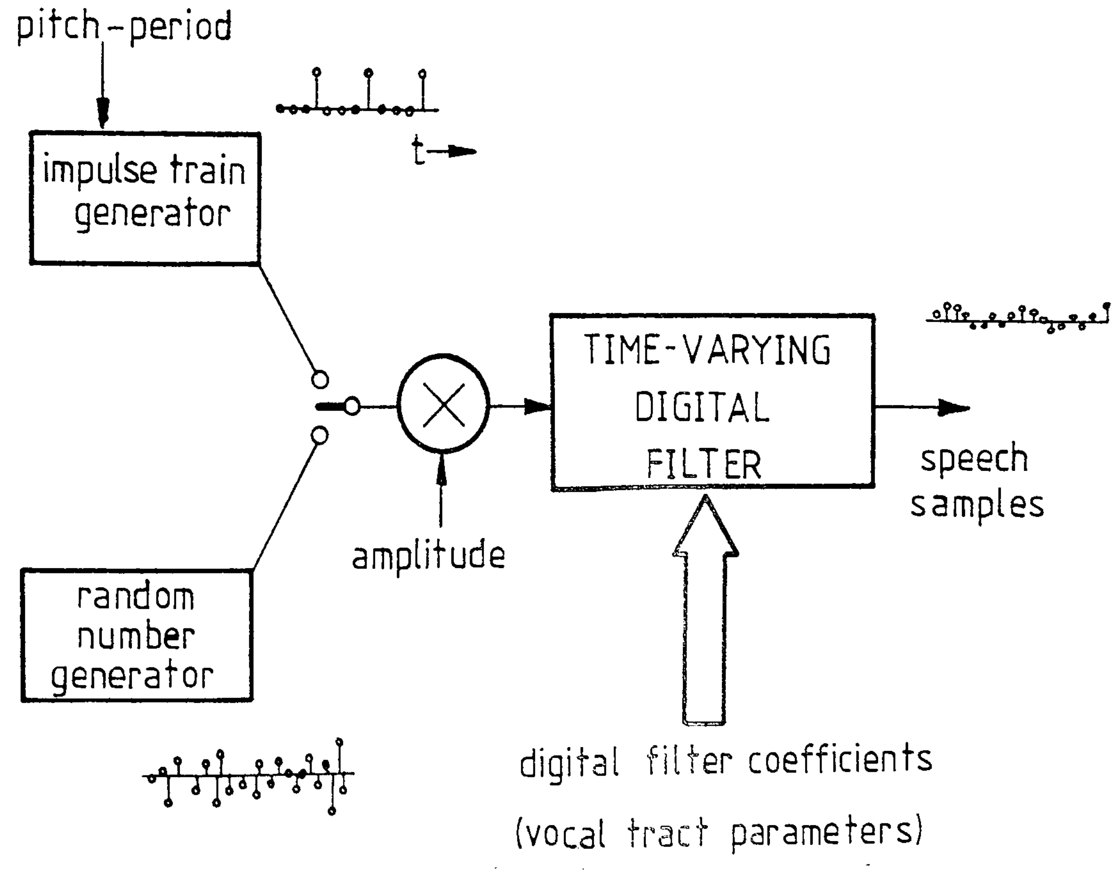


Figure 2-8: Digital speech production model

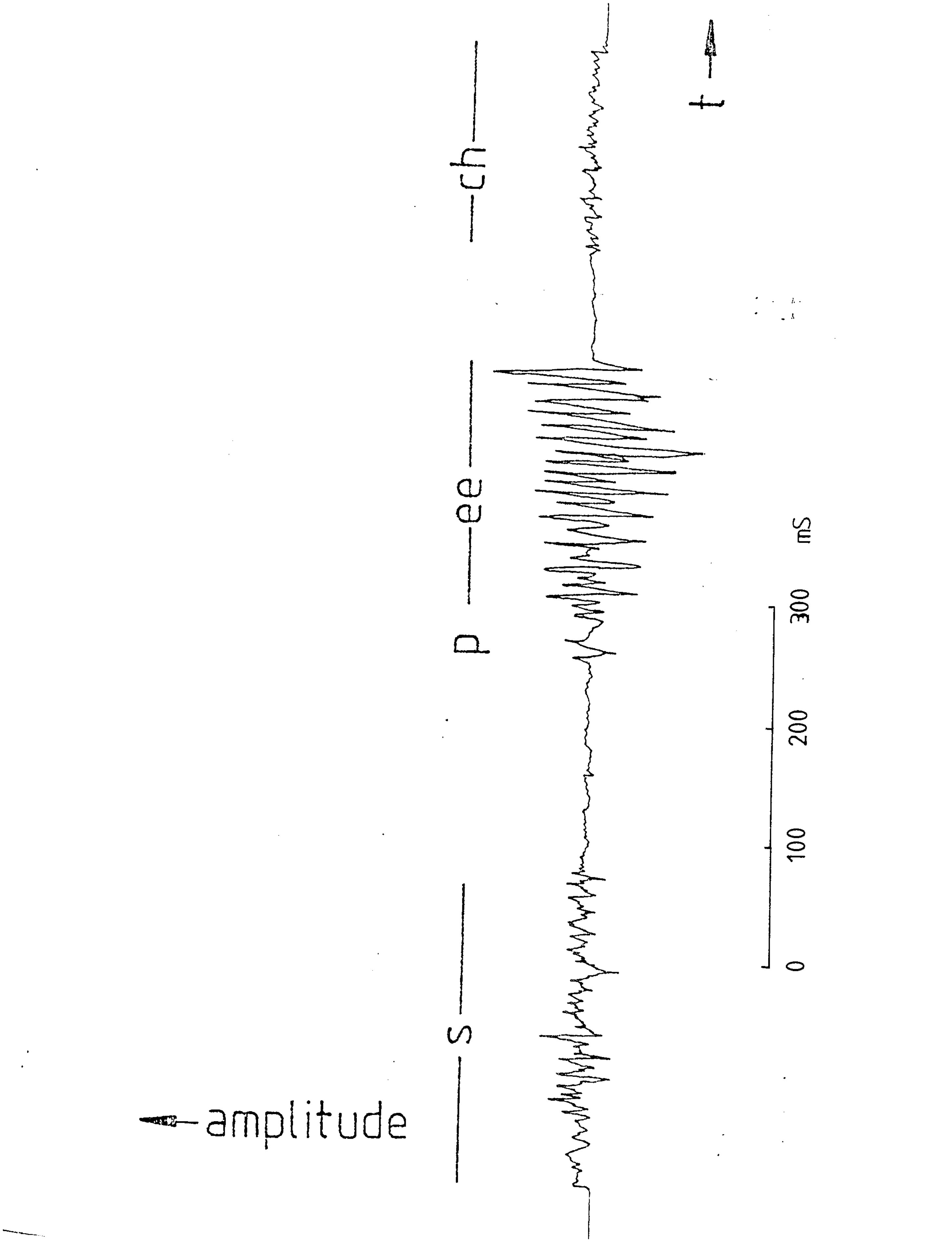


Figure 2-9: Oscillogram of the word "speech"

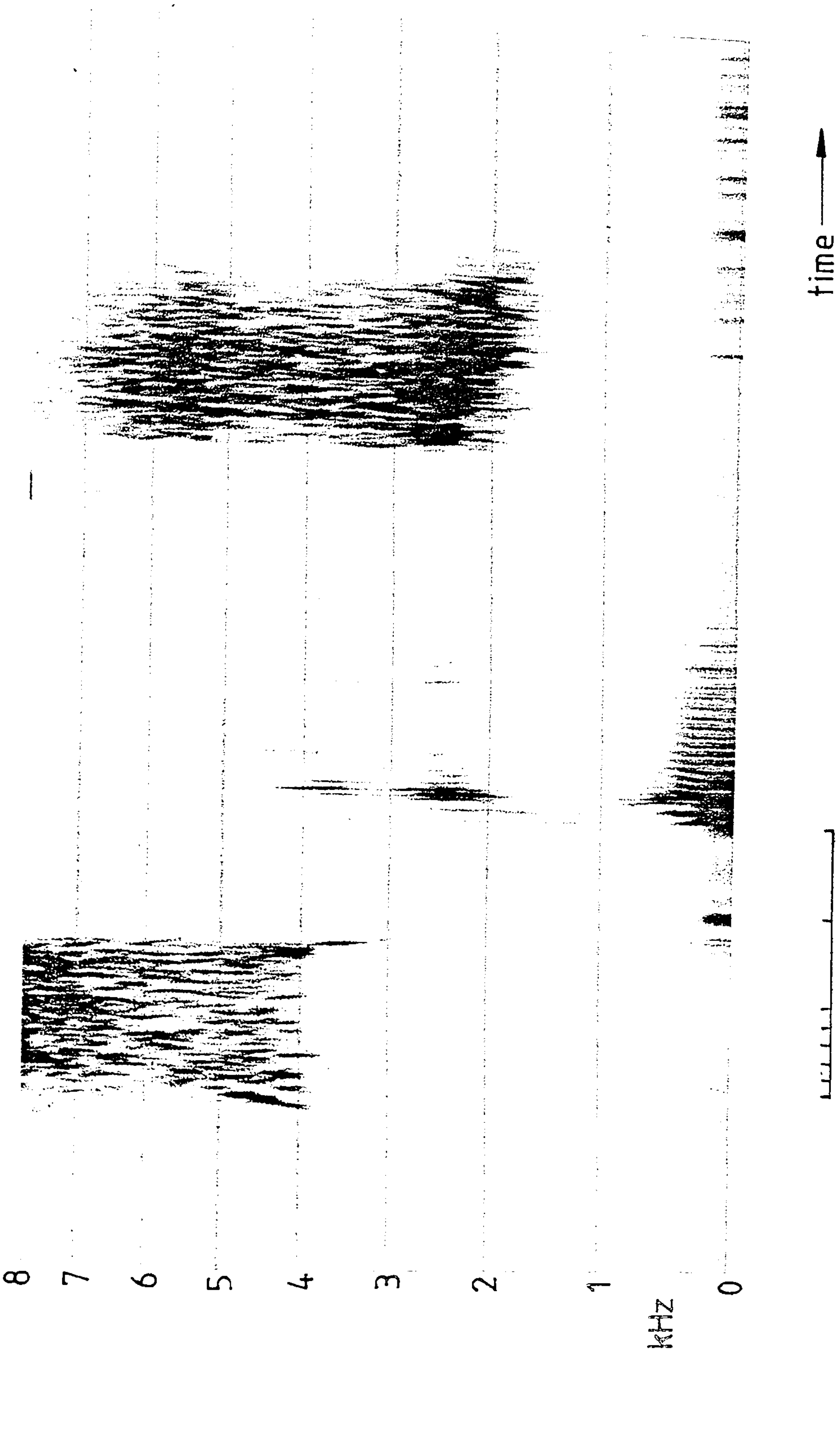


Figure 2-10: Wide-band spectrogram of the word "speech"

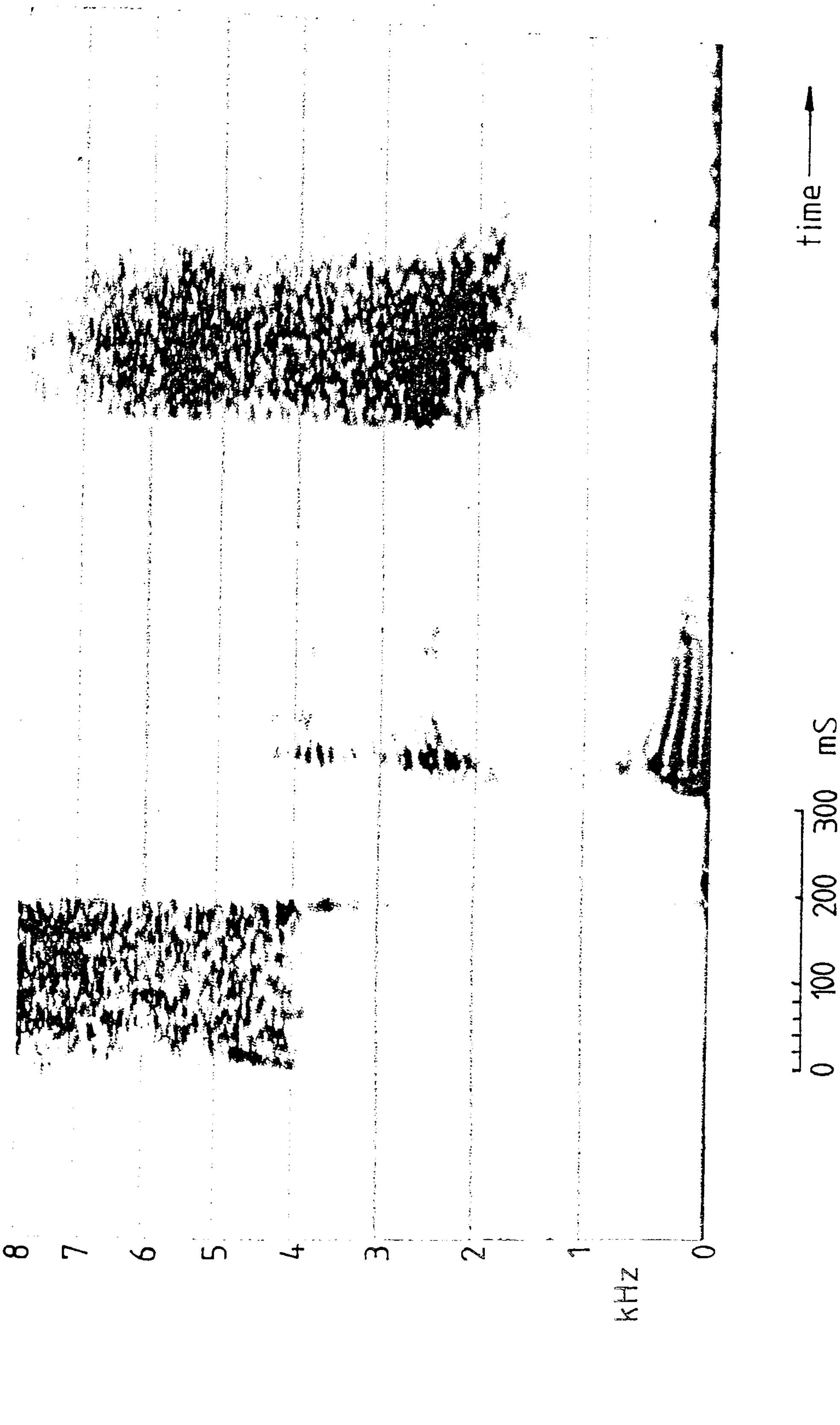


Figure 2-11: Narrow-band spectrogram of the word "speech"

Figure 2-10 is a wide-band spectrogram of the utterance; this shows the overall spectral characteristics, in particular the formants which appear during voiced portions of the utterance.

Figure 2-11 is a narrow-band spectrogram; the timescale and frequency range are the same as in Figure 2-10. The narrow bandwidth analysis filter is able to resolve individual harmonics of the voiced portions of the utterance.

CHAPTER THREE

EXISTING SPEECH ANALYSIS SYSTEMS

3.1 INTRODUCTION

Many methods for the analysis of speech signals into their fundamental properties and characteristics have been described. A basic dichotomy exists between TIME-DOMAIN and FREQUENCY-DOMAIN analysis. The former is based upon the speech waveform as a function of time; the latter is concerned with the spectral representation of the speech waveform.

3.2 BASIC TIME-DOMAIN MEASUREMENTS

3.2.1 Peak amplitude measurements

Examination of Figure 3-1, which is an oscillogram of the word "speech" spoken by an adult male, shows that voiced portions of the speech waveform are characterized by a series of pseudo-periodic amplitude peaks, the period of which corresponds to the fundamental period of the speech signal. Unvoiced portions of the waveform display aperiodic peaks of a generally lower amplitude. The nature of the amplitude peaks in the speech waveform has been used as the basis for voiced/unvoiced analysis, crude amplitude measurement, and pitch period estimation. The pseudo-periodicity of the speech waveform, together with the frequent presence of several minor amplitude peaks within one pitch period, necessitates some care in selecting the major peak

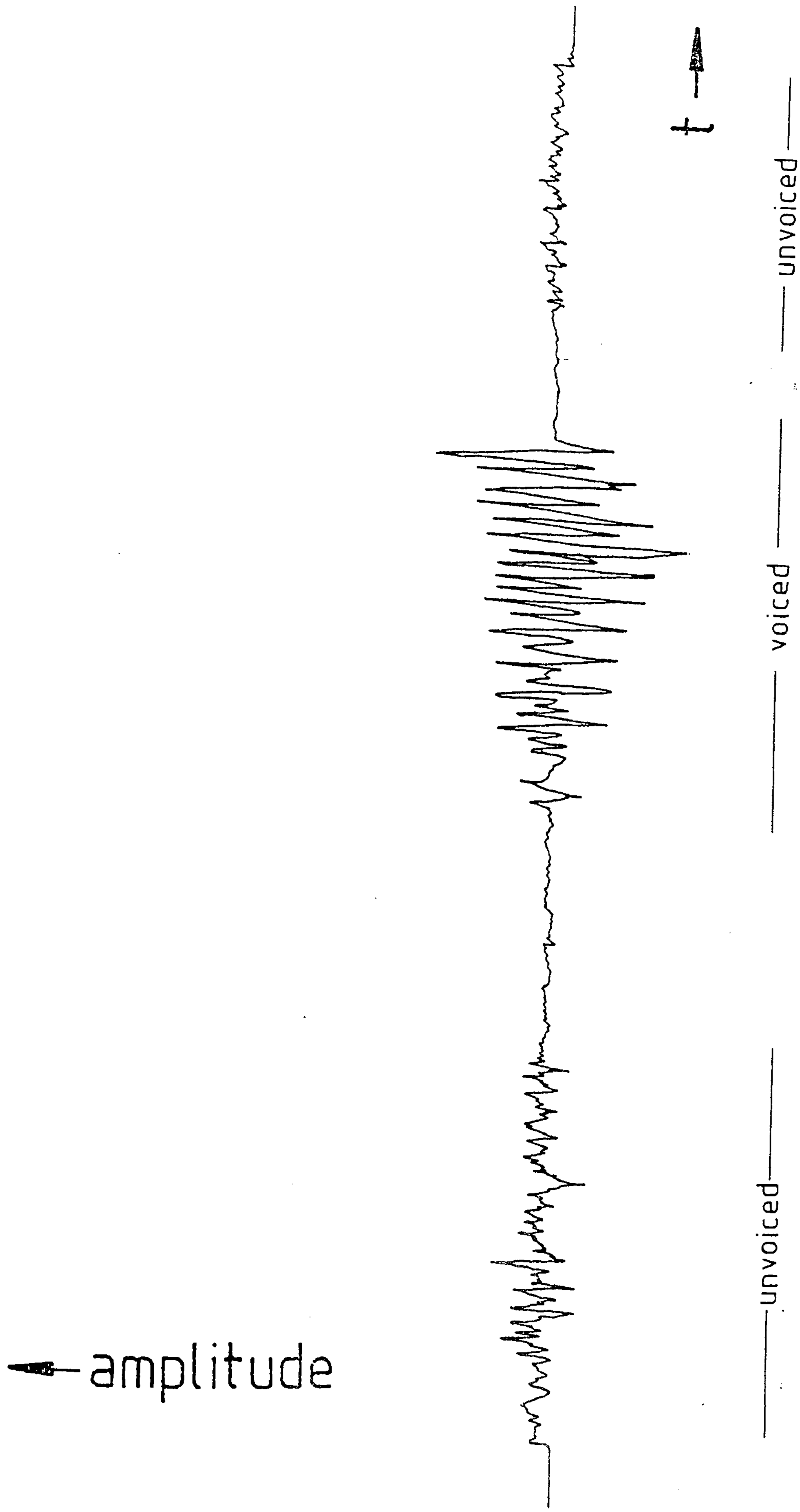


Figure 3-1: Oscillogram of the word "speech"

corresponding to each pitch period.

3.2.2 Energy measurements

Various measures of speech waveform energy have been used to provide voiced/unvoiced detection and to aid major amplitude peak selection. The general definition of the energy of a discrete speech signal, x(n), is:

$$E = \sum_{n=-\infty}^{\infty} x^2(n) \qquad \ldots(3.1)$$

In practice, it is normal to employ some form of time window:

$$E(n) = \sum_{m=0}^{N-1} [w(m)x(n-m)]^2 \qquad \ldots(3.2)$$

where w(m) is a N-sample weighting sequence.

In order to overcome the effect of the squaring operation, which tends to emphasize variations between successive estimates of E(n), the sum of absolute values may be computed:

$$\hat{E}(n) = \sum_{m=0}^{N-1} \left| w(m)x(n-m) \right| \qquad \ldots(3.3)$$

The fidelity with which E(n) will exhibit the time-varying amplitude properties of x(n) depends upon the shape and duration of the weighting function chosen, and upon the value of N related to the sampling period and the average pitch period of the speech signal being analysed.

3.2.3 Zero-crossing measurements

A continuous waveform exhibits a zero-crossing whenever the waveform crosses the zero axis, that is to say, whenever the polarity of the waveform changes. A simple waveform such as a sinusoid with frequency f_0Hz has $2f_0$ zero-crossings per second. For a sampled waveform, a zero-crossing is said to occur between two successive samples x(n-1) and x(n) if:

$$\mathrm{sign}[x(n-1)] \neq \mathrm{sign}[x(n)] \qquad \ldots(3.4)$$

In practice, speech waveforms exhibit more than two zero-crossings per pitch period, so that some sophistication is necessary if a pitch period estimation scheme is to be based upon zero-crossing measurements. Zero-crossing counts have been used to provide voiced/unvoiced detection and to give a crude indication of spectral properties.

3.2.4 Short-time autocorrelation analysis

The general autocorrelation function for a discrete-time signal x(n) is defined as:

$$\emptyset_x(m) = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} x(n)x(n+m)$$

...(3.5)

The autocorrelation function is useful for displaying structure in a waveform. For the purposes of pitch-period detection, if we assume that x(n) is exactly periodic with period P, so that x(n) = x(n+P) for all 'n', then it may be shown [R301] that:

$$\emptyset_x(m) = \emptyset_x(m+P)$$

...(3.6)

that is to say, the autocorrelation is also periodic, with the same period.

For a nonstationary signal, such as a speech signal, it is convenient to define a short-time autocorrelation function

$$\emptyset_l(m) = \frac{1}{N} \sum_{n=0}^{N'-1} [x(n+l)w(n)][x(n+l+m)w(n+m)],$$

$$0 <= m <= M_0 - 1$$

...(3.7)

where $w(n)$ is an analysis window, N is the length of the section under analysis, N' is the number of samples actually considered in the calculation of $\emptyset_l(m)$, M_0 is the number of autocorrelation points to be computed, and l is the index of the starting sample of the analysis 'frame'.

3.3 BASIC FREQUENCY-DOMAIN MEASUREMENTS

[H302],[S303],[T304],[S227]

The mathematical link between an aperiodic time function $x(t)$ and its complex amplitude-density spectrum $X(\omega)$ is the Fourier transform pair:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\, dt \qquad \ldots(3.8)$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\, d\omega, \qquad \omega = 2\pi f \qquad \ldots(3.9)$$

For the transform to exist,

$$\int_{-\infty}^{\infty} |x(t)| \quad dt$$

must be finite.

A continuous speech signal is not, generally, known over all time, nor does it satisfy the existence criterion above. It is, therefore, common practice to 'window' the speech waveform using some weighting function h(t), so that its transform exists for integration over known values and is limited in extent to quasi-steady segments of the waveform. The result is a 'running spectrum', with real time as an independent variable.

The weighting function h(t) is chosen to ensure that the product of window and waveform is Fourier transformable, and is usually the impulse response of a physically-realisable linear system, with $h(t) = 0$ for $t < 0$. The choice of h(t) will determine the resolution obtained in the time and frequency domains; a short time window gives spectral analysis with fine temporal resolution, whereas a relatively long time window yields spectral analysis with fine frequency resolution.

The resulting SHORT-TIME FOURIER TRANSFORM pair has the form:

$$X(\omega,t) = \int_{-\infty}^{t} x(l)h(t-l)e^{-j\omega l}\,dl \qquad \ldots(3.10)$$

$$[x(l)h(t-l)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega,t)e^{j\omega l}\,d\omega \qquad \ldots(3.11)$$

where 'l' indicates the start of the analysis window.

Equations 3.8 to 3.11 deal with a continuous time signal. In the domain of sampled data, the short-time DISCRETE FOURIER TRANSFORM (DFT) takes the form:

$$X(\omega,nT_s) = \sum_{r=-\infty}^{n} x(rT_s)h(nT_s-rT_s)e^{-j\omega rT_s}$$

...(3.12)

where T_s is the sampling period. For a given value of ω, $X(\omega,nT_s)$ is a sequence defined on the discrete time index nT_s; at a given time, $X(\omega,nT_s)$ is a continuous periodic function of ω, with period $2\pi/T_s$.

Much attention has been paid to the computational overheads involved in the calculation of the DFT for a particular segment of time-domain waveform, leading to the development of the FAST FOURIER TRANSFORM (FFT), a class of computational algorithms for the efficient evaluation of the DFT, defined as:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}nk},$$

for $k = 0,1,\ldots,N-1$

...(3.13)

3.4 SPEECH ANALYSIS METHODS

A brief description of a number of proposed speech analysis methods will be given in this section under four main categories:

- Time-domain analysis
- Short-time spectrum analysis
- Homomorphic analysis
- Linear prediction

3.4.1 TIME-DOMAIN ANALYSIS METHODS

As mentioned above, the presence of minor amplitude peaks and zero-crossings within one pitch period of a speech waveform complicates the estimation of pitch-period from peak/peak or inter-zero-crossing time interval measurements. Most time-domain pitch-period estimation schemes consist of two phases, a period calculation phase, which may be a simple time interval measurement, preceded by some form of preprocessor, the aim of which is, literally or effectively, to reduce the speech waveform to one amplitude peak of a given polarity per pitch-period, or, correspondingly, to two zero-crossings of opposite direction per pitch-period. The preprocessor may use schemes such as filtering and clipping to reduce waveform excursions within a pitch-period, or may employ heuristics to separate significant features from insignificant ones.

3.4.1.1 Methods based upon peak and energy measurements

A number of 'pitchmeters' [G305] [D306] [A152] [F307] [D153] have been constructed around the basic circuit shown in Figure 3-2.

The waveforms associated with this circuit are shown in Figure 3-3.

Diode D charges capacitor C1 as long as the input voltage (Figure 3-3a) exceeds that across C1. Beyond peak 'α', conduction in D ceases, and C1 discharges exponentially through R1, until a point is reached where the input voltage again exceeds that across C1, whereupon D will conduct and C1 will again be charged. The voltage across C1, shown in Figure 3-3b, is differentiated by C2 and R2, resulting in the waveform of Figure 3-3c, in which the major peaks of the input waveform are accentuated. The process may be repeated as many times as is required to remove all but the significant initial peaks of the waveform. The time constants R1C1 and R2C2 must be chosen carefully with due regard to the expected fundamental frequency range of the input speech samples. Once the initial peaks have been extracted, the pitch-period may be estimated by measuring the inter-peak time interval; alternatively, simple analogue electronic means may be employed to cause the period between peaks to control the vertical deflection of an oscilloscope trace, producing a pitch-period or frequency contour.

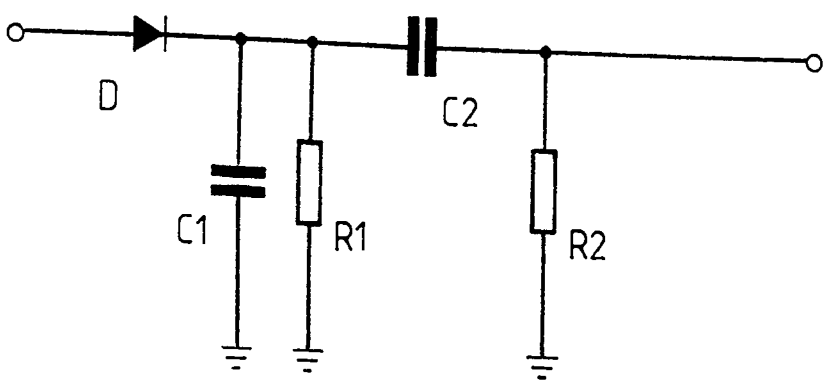


Figure 3-2: Basic 'pitchmeter' circuit

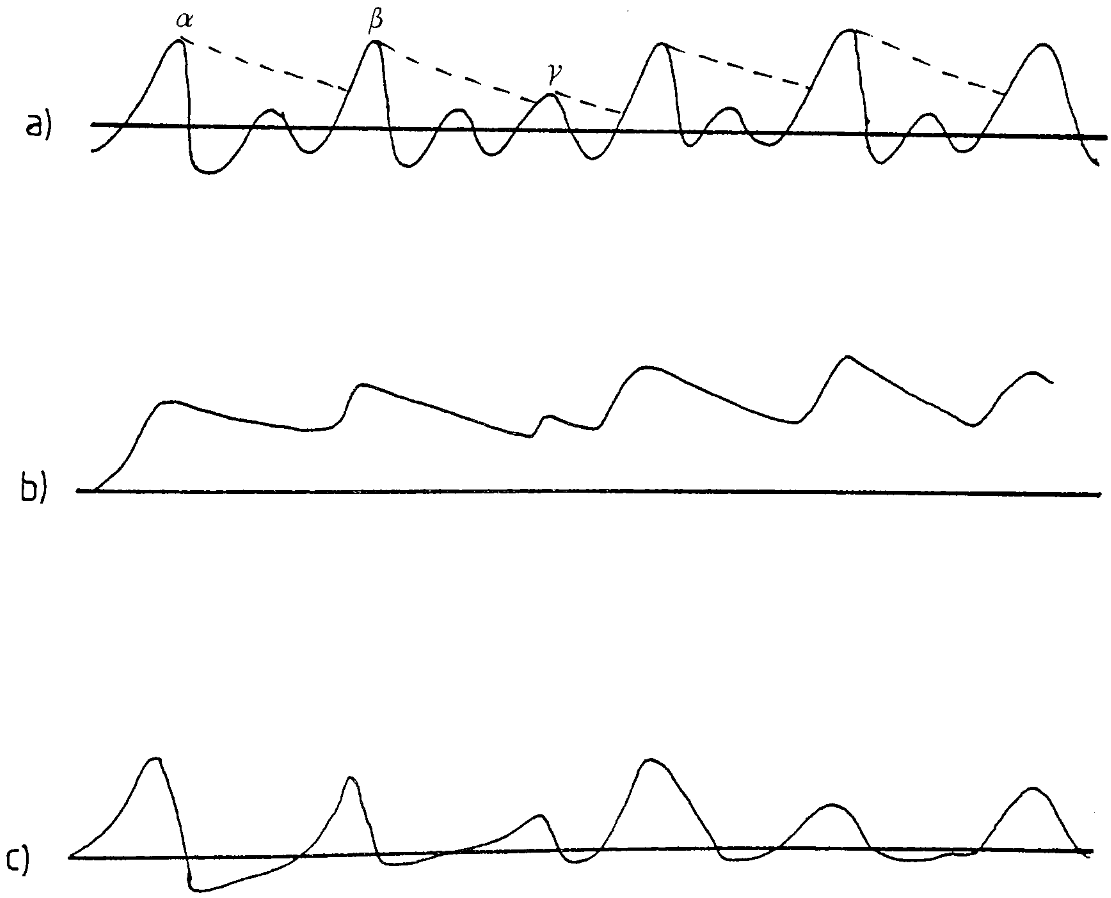
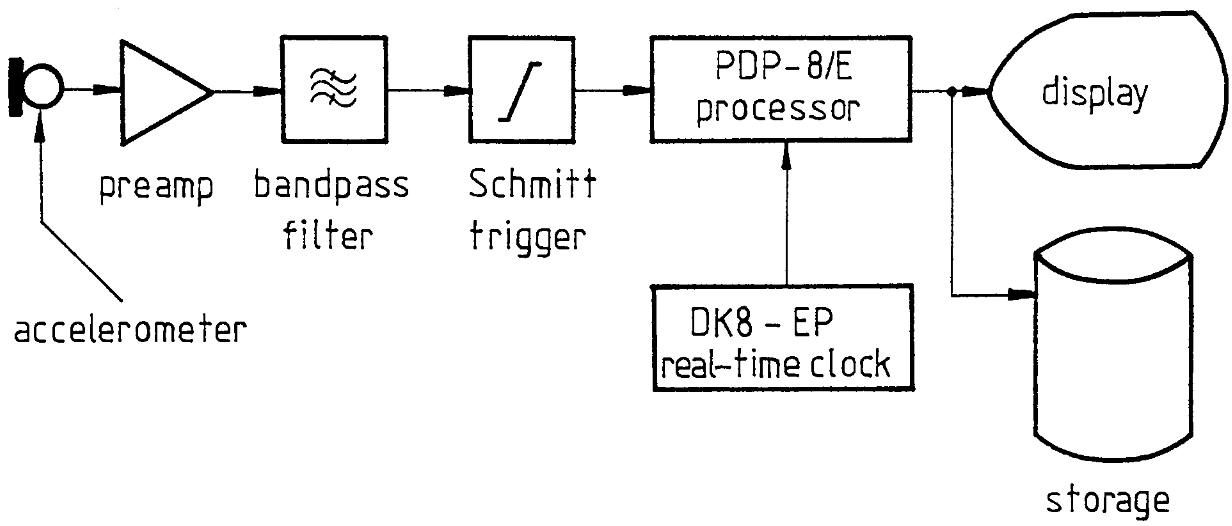


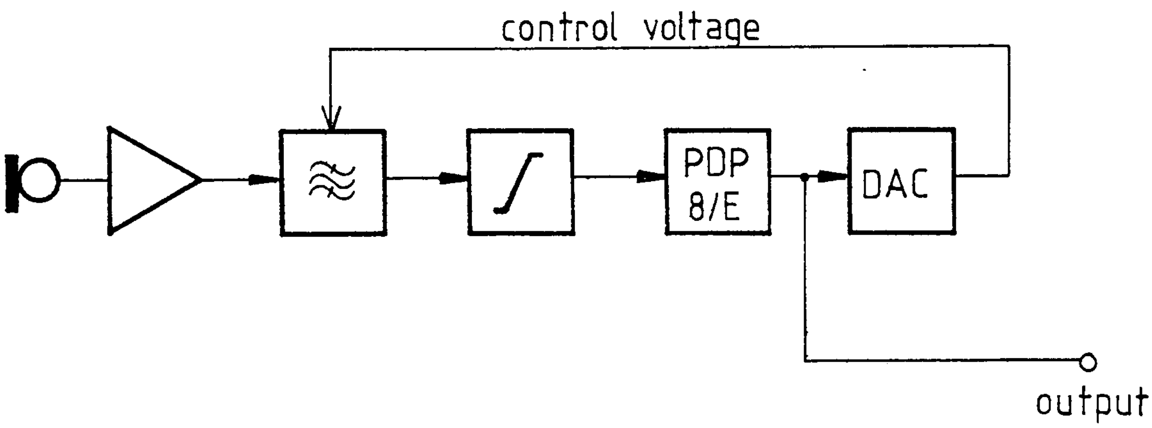
Figure 3-3: Waveforms associated with the basic 'pitchmeter'

The LARYNGOGRAPH and VOISCOPE system [F308] [F107] [W309] uses neck-mounted electrodes to measure the electrical impedance in the area of the larynx. It has been shown [F310] that this impedance value is affected by the movements of the vocal cords which occur during voiced speech production, and may be used to derive an electrical analogue of the glottal waveform (specifically, the impedance value is almost totally dependent upon the degree of vocal cord contact [F107]). The VOISCOPE display unit uses simple electronic circuitry to perform pitch-period estimation and to display an intonation contour of fundamental frequency against time.

Rostron and Welbourn [R113] have used a small accelerometer taped to the neck to provide a low-harmonic glottal waveform. The two processing schemes employed are illustrated in Figure 3-4. Both methods utilise a bandpass filter in an attempt to isolate the fundamental frequency of the glottal waveform. The output of the filter triggers a Schmitt trigger circuit, and a minicomputer is used to time the interval between interrupts thus produced. In Figure 3-4a the filter centre frequency is fixed; in Figure 3-4b a voltage-controlled filter is used, and the centre frequency is controlled dynamically by means of a feedback loop from the digital processor. The system aims to 'track' the fundamental frequency of the speech input by varying the filter centre frequency.



a)



b)

Figure 3-4: Processing schemes used by Rostron and Welbourn

Boothroyd and Decker [B155] [B156] use a 24dB/octave low pass filter to process the speech waveform from a microphone. The filter output, which is, ideally, approximately sinusoidal, is fed to a trigger circuit, and the measured pitch-period interval controls the vertical deflection of an oscilloscope display.

Reddy [R311] [R312] describes a method of pitch-period estimation in which local amplitude maxima and minima within a given segment of the waveform are identified, and are examined to determine the location of significant maximum and minimum amplitude peaks, according to certain heuristic procedures. Reddy notes that the initial amplitude peak within a pitch-period is usually in close proximity to a significant peak of the opposite polarity, and uses this information to allocate 'pitch markers' to significant maximum/minimum peak pairs, which are generally found to correspond to the beginning of a pitch-period. Further heuristic procedures are applied to cater for extraneous, missing and misplaced markers.

Miller [M313] describes a similar scheme, in which the excursion cycles of the speech waveform - the portions between consecutive zero-crossings - are integrated numerically to provide a coarse indication of excursion cycle energy. Miller notes that principle excursion cycles - those occurring at the beginning of a pitch-period - tend to have long durations as well as large amplitudes, and consequently contain considerable energy. On the basis of this energy analysis, a database of presumptive principal cycles is constructed. Heuristic procedures are then applied to isolate actual principal cycles, and a pitch-period estimation is made.

Tucker and Bates [T314] describe an algorithm in which a pair of adaptive thresholds, β+(t) and β−(t), are used to extract a train of pulses, pm(t), from a voiced speech segment, s(t) (shown in Figure 3-5, with the pulses shaded). The pitch-period estimation is based upon two sets of 'features' of the pulses train. The 'primary features' extracted from s(t) are:

the amplitude of the mth pulse ... Am

the energy in the mth pulse ... Em

the duration of the mth pulse ... τm

The 'secondary features', derived from the primary features and chosen to be relatively insensitive to envelope variations are:

$$Bm = \text{sign } [Am]$$

$$Cm = Am/Em^{1/2}$$

The primary and secondary features are combined to produce a 'vector signal', U(t), comprising the components:

$$Bm, \ |Cm|, \ \tau m \text{ and } Em$$

The pitch-period, T, of s(t) is estimated from U(t).

Gold [G315] [G316] [G317] examined two main properties of speech which have been used as the basis of pitch-period estimation schemes: firstly, the regularity of large sections of the speech wave, which may allow the fundamental frequency component to be extracted by low-pass filtering [G305], and, secondly, the 'peakedness' of the speech waveform, which may allow the pitch-period to be measured following peak detection [D306]

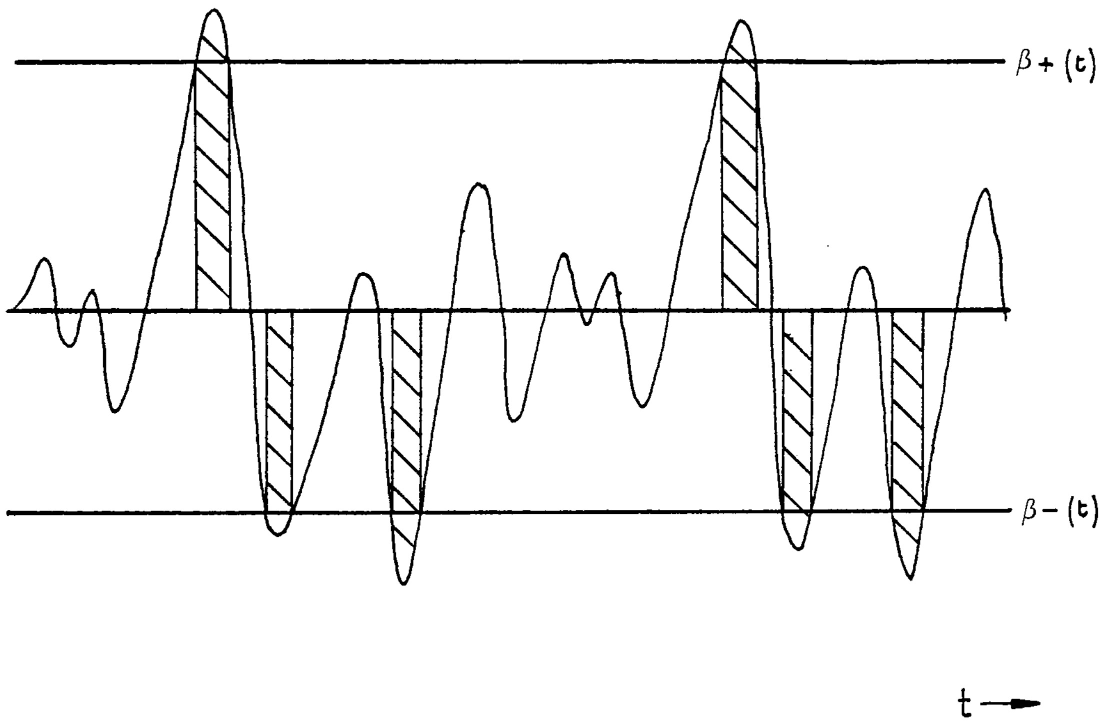


Figure 3-5: Waveform pre-processing using adaptive thresholds

[A152] [F307] [D153]. The former set of schemes tend to fail when the pitch-period or spectrum of the speech waveform change rapidly; the latter set tend to fail when the waveform is insufficiently peaked, despite its regularity. Based upon this observation, Gold constructed a PARALLEL PROCESSING technique, whereby the speech waveform is processed by six separate pitch-period estimators, three based upon regularity rules, and three based upon peakedness rules. The six individual pitch-period estimates are examined for common features, and from these a choice of estimated pitch-period is made.

3.4.1.2 Zero-crossing analysis

In a study of the effects of various types of distortion upon the intelligibility of speech, Licklider and Pollack [L318] showed that, after suitable preprocessing, clipped speech is highly intelligible. This suggests that much significant information necessary for speech recognition is contained in the zero-crossing sequence. Zero-crossing analysis has, consequently, been applied in many speech recognition systems, as well as in a number of speech analysis methods [P319] [M320] [B321]. Various zero-crossing analysis techniques are examined and classified by Niederjohn [N322].

Geckinli and Yavuz [G323] describe an algorithm which searches for repetitive portions in the zero-crossing interval sequence of a speech waveform, and thereby estimates the pitch-period of the waveform.

Zero-crossing interval sequences may be plotted against time to produce 'intervalgrams' [C324] [B325], which are similar to speech spectrograms (see Section 3.4.2.3).

3.4.1.3 Short-time autocorrelation analysis

Autocorrelation has been used as the basis of many pitch-period estimation schemes. The autocorrelation computation is made directly upon the waveform, the calculations required are fairly simple, and the results of autocorrelation analysis are largely phase-insensitive [R301]. A number of problems associated with autocorrelation analysis have been identified [R301]. The autocorrelation computation is highly time-consuming, despite the basic simplicity of the calculations involved. Furthermore, in addition to the pitch-period of the speech waveform, the autocorrelation function also exhibits peaks due to the detailed formant structure of the speech signal. The effect of windowing for short-time autocorrelation analysis causes the autocorrelation function to taper to zero with increasing autocorrelation index, so that formant peaks may be of greater magnitude than pitch-period peaks. Finally, the window size is best related to the pitch-period being measured, ideally containing between two and three complete pitch-periods, so that dynamic selection of window length is required.

Several techniques have been developed leading to reductions in the calculations needed to compute the autocorrelation function [R326] [M327] [P328] [B329] [L330] [R331]. Gill [G332] infinitely peak-clipped speech prior to autocorrelation. The resulting binary signal led to significant simplifications in the implementation of the correlator. However, the process of infinite peak-clipping does not remove the formant structure of the speech signal.

Partial elimination of the effects of the higher formant structure has been attempted using various spectral flattening techniques. Sondhi [S333] has used a filter-bank spectrum flattener and a centre-clipping circuit to process speech prior to autocorrelation. Dubnowski et al [D334] have used a combination of centre- and infinite peak-clipping prior to autocorrelation. Rabiner [R301] examined the effects of various combinations of simple centre-clipping, compressed centre-clipping, and combined centre- and peak-clipping, drawing the conclusion that almost all of the combinations of these non-linear preprocessors gave essentially the same performance.

Several variations of autocorrelation analysis have been implemented, including average magnitude difference function (AMDF) [R335] and optimum comb [M336].

3.4.2 SHORT-TIME SPECTRUM ANALYSIS

Two methods are commonly used for short-time spectrum analysis: filter banks and FFT algorithms.

3.4.2.1 Filter banks

Filter bank systems employ a bank of bandpass filters, with passbands chosen to cover the normal speech frequency range (Figure 3-6). Analogue filters have largely been superseded by digital filters [S337] [S338].

3.4.2.2 FFT analysis

The FFT [B339] [S340] [I341] (see Equation 3.13) can be interpreted as the Fourier transform of an N-sample speech segment, the frequency resolution of the spectral measurement being inversely proportional to N. Figure 3-7a shows a segment of speech waveform windowed using a 50mS Hamming window [B342] (N = 500 samples) at a 10kHz sampling rate; Figure 3-7b shows the corresponding short-time spectrum. Figures 3-8a and 3-8b show the windowed speech segment and corresponding short-time spectrum for N = 50; the former contains both fundamental frequency and vocal tract transfer information, while the latter shows only the general shape of the vocal tract transfer function.

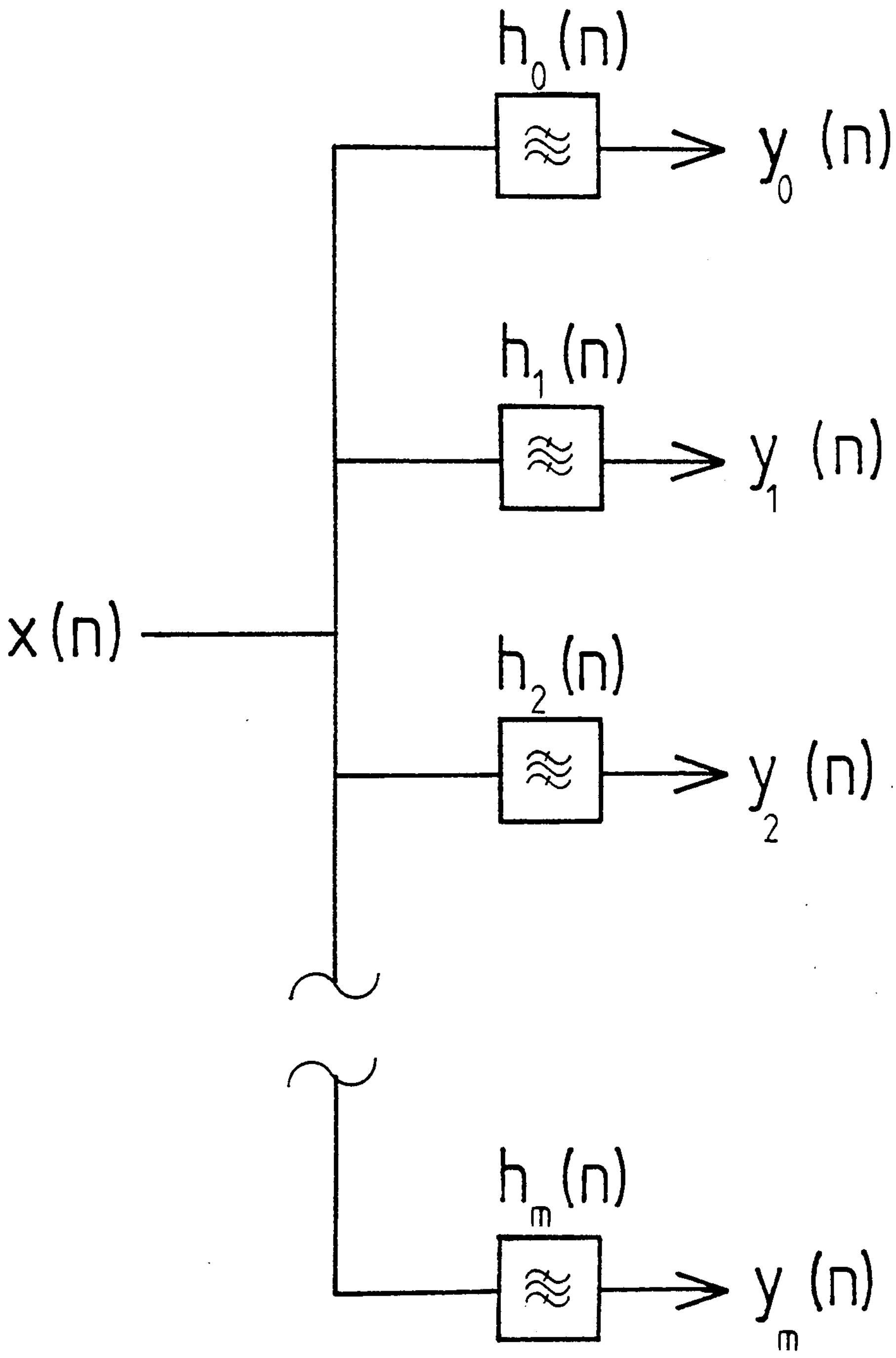


Figure 3-6: Analysis by bandpass filter bank

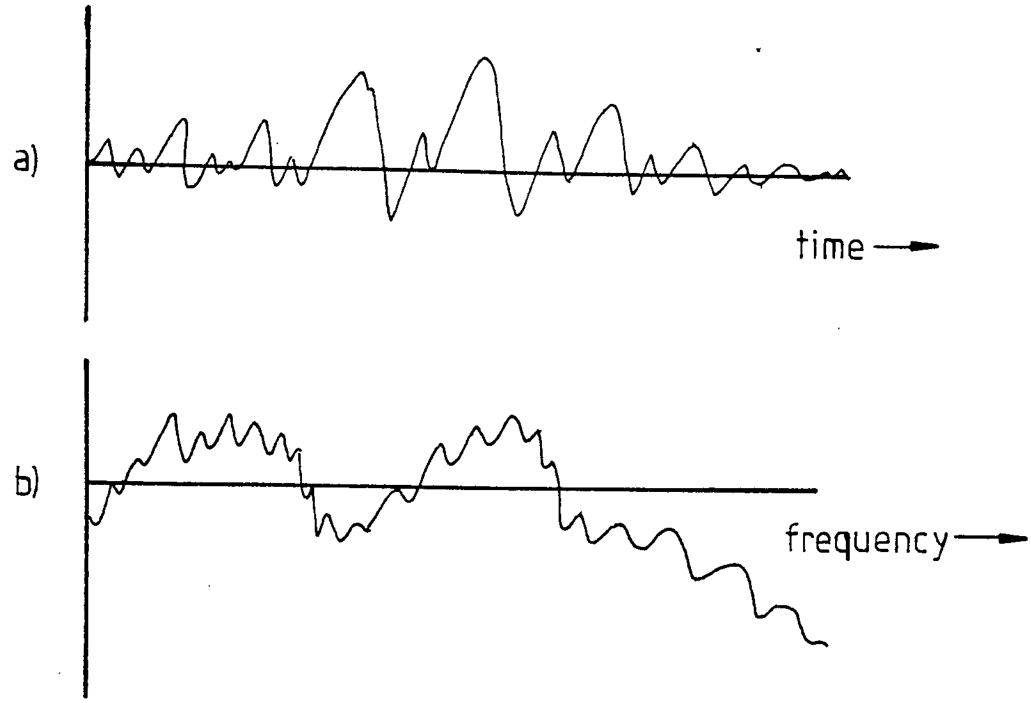


Figure 3-7: Speech waveform and corresponding FFT, N=500

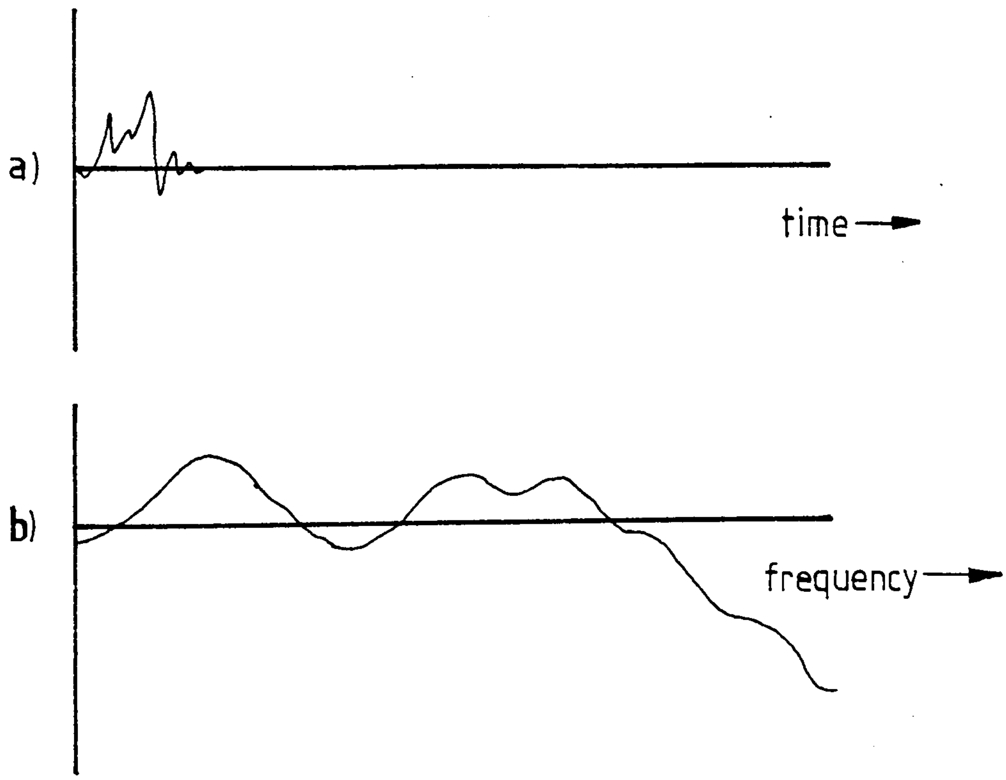


Figure 3-8: Speech waveform and corresponding FFT, N=50

Pitch synchronous Fourier analysis [M343], in which the speech waveform is first segmented into individual pitch-periods by some means such as visual inspection, and then subjected to Fourier analysis, gives a particularly detailed spectral representation of the speech signal, and allows poles (resonances) and zeroes (antiresonances) in the spectrum to be assigned uniquely to either the vocal tract or the vocal cord excitation source.

3.4.2.3 The sound spectrograph

The SOUND SPECTROGRAPH [K344] [P345] provides a means of displaying the short-time spectrum of an utterance several seconds in length. The basic operation of the spectrograph is illustrated in Figure 3-9.

The device contains a loop of magnetic material, which is used as the recording medium for the speech sample. During the production of a spectrogram, the loop is replayed repeatedly at high speed, and the signal is passed through a bandpass filter, which is, effectively, scanned slowly across the frequency band of the signal. In practice, the filter centre frequency is fixed, and the spectrum of the speech signal is caused to 'move' past the filter by modulating the signal onto a high fequency carrier, so that one sideband of the resultant signal is swept past the fixed bandpass filter. The carrier frequency control is mechanically coupled to an electrical stylus bearing upon a conductive drum covered with electrically-sensitive paper, so that the stylus is drawn across the paper as the sideband sweeps

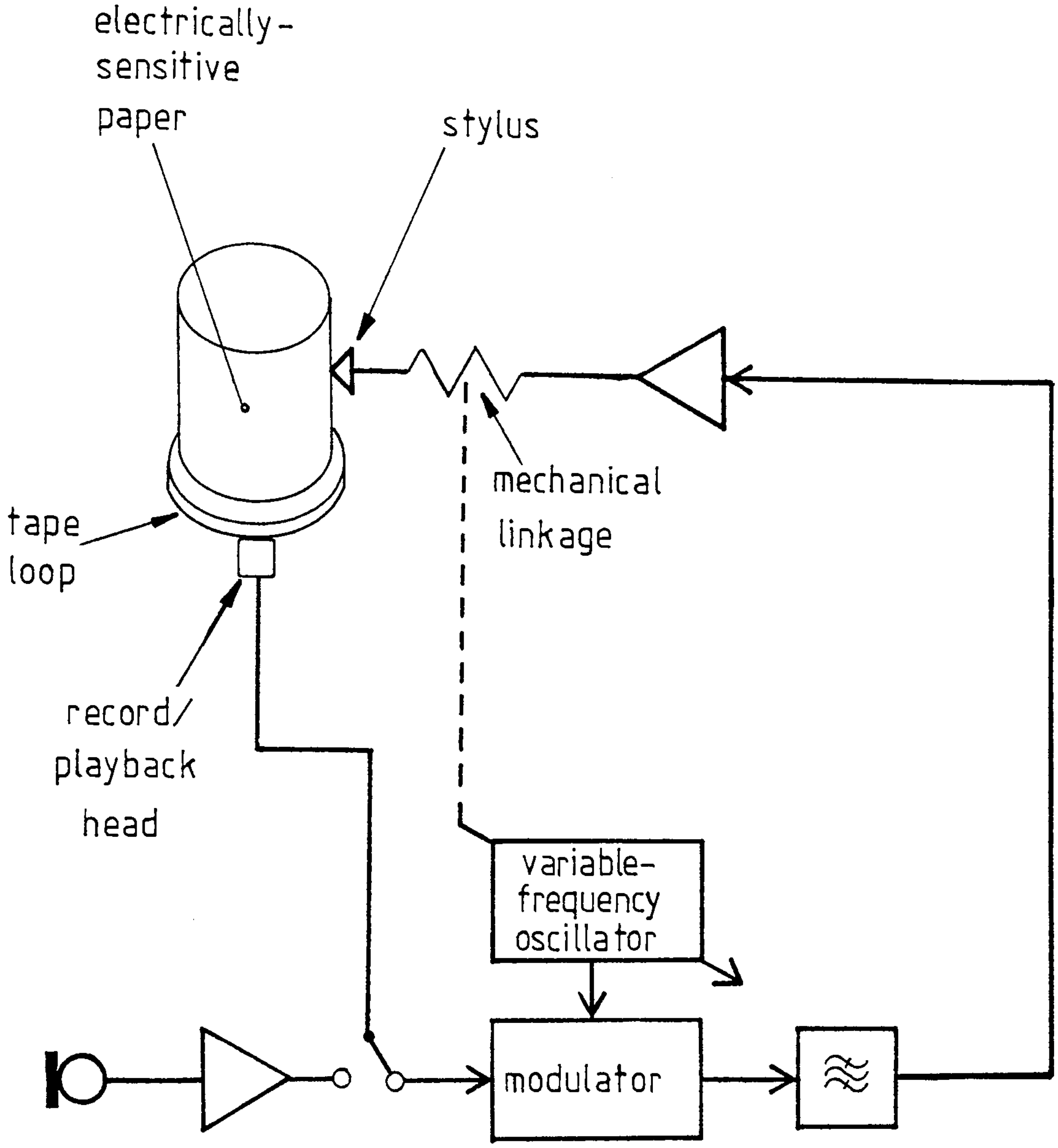


Figure 3-9: The sound spectrograph

past the bandpass filter. The output of the filter is amplified and fed to the stylus, so that the paper is burned in proportion to the magnitude of the stylus current. The result is a plot of frequency against time, with intensity shown by the degree of blackness of the trace. In practice, the intensity range of the paper is limited to about 12dB [P346].

While sound spectrographs based upon the above description are still widely used, a number of systems have been developed which use advanced short-time spectrum analysis techniques to produce spectrogram-like displays [G347] [R348] [M349] [M350] [O351] [S352]. The application of the FFT to spectrogram production leads to versatility in choice of factors such as frequency range and frequency resolution, and makes possible real-time spectrogram displays.

3.4.2.4 Applications of short-time spectrum analysis

Longuet-Higgins [L108] has used an analogue filter bank approach to construct a 'speech intonation spectrometer', which displays the speech spectrum in real time as a series of curves on an oscilloscope. The system is intended to render visible the intonational features of spoken utterances.

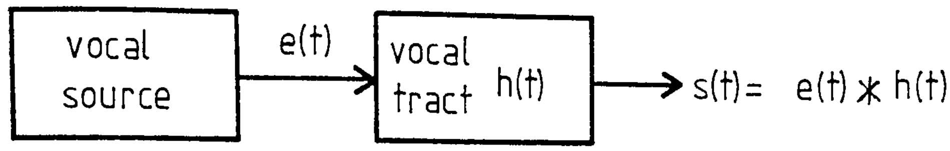
James [J137] describes a speech analysis system in which the speech signal is fed to a bank of four bandpass filters and is then processed by a computer program which attempts to identify the channel in which the fundamental frequency is located.

Flanagan and Golden [F353] found that instantaneous frequencies measured at the outputs of a bank of bandpass filters fed with a voiced speech sample differ little from integer multiples of the fundamental frequency up to at least 2500Hz. The fundamental frequency can be estimated from this set of instantaneous frequencies, representing the frequencies of harmonic components of the speech signal, by dividing each instantaneous frequency by the corresponding harmonic number. This basic technique has been applied in several pitch-period detection systems [S354] [M355] [N356].

3.4.3 HOMOMORPHIC ANALYSIS

Homomorphic analysis techniques were first applied to seismic signals [B357], and have since found application in various fields where it is required to separate signals that have been combined by multiplication and convolution [O358]. The applicability of homomorphic techniques to speech depends upon the model of speech production (ref Section 2.4) in which the speech signal is viewed as the convolution of an excitation source - either a quasi-periodic pulse train or random noise - with the vocal tract impulse response (see Figure 3-10).

A homomorphic speech analysis system [N359] [N360] [O361] is illustrated in Figure 3-11. The signal at point A is the convolution of the glottal excitation source with the vocal tract impulse resonse. The short-time Fourier transform $\mathfrak{F}[s(t)]$ of s(t), point B in Figure 3-11, is the product of the Fourier



'*' denotes convolution

h(t) is the impulse response of the vocal tract

e(t) is the source signal at the glottis

s(t) is the output signal for voiced speech

Figure 3-10: Convolution of the vocal source with the vocal tract response

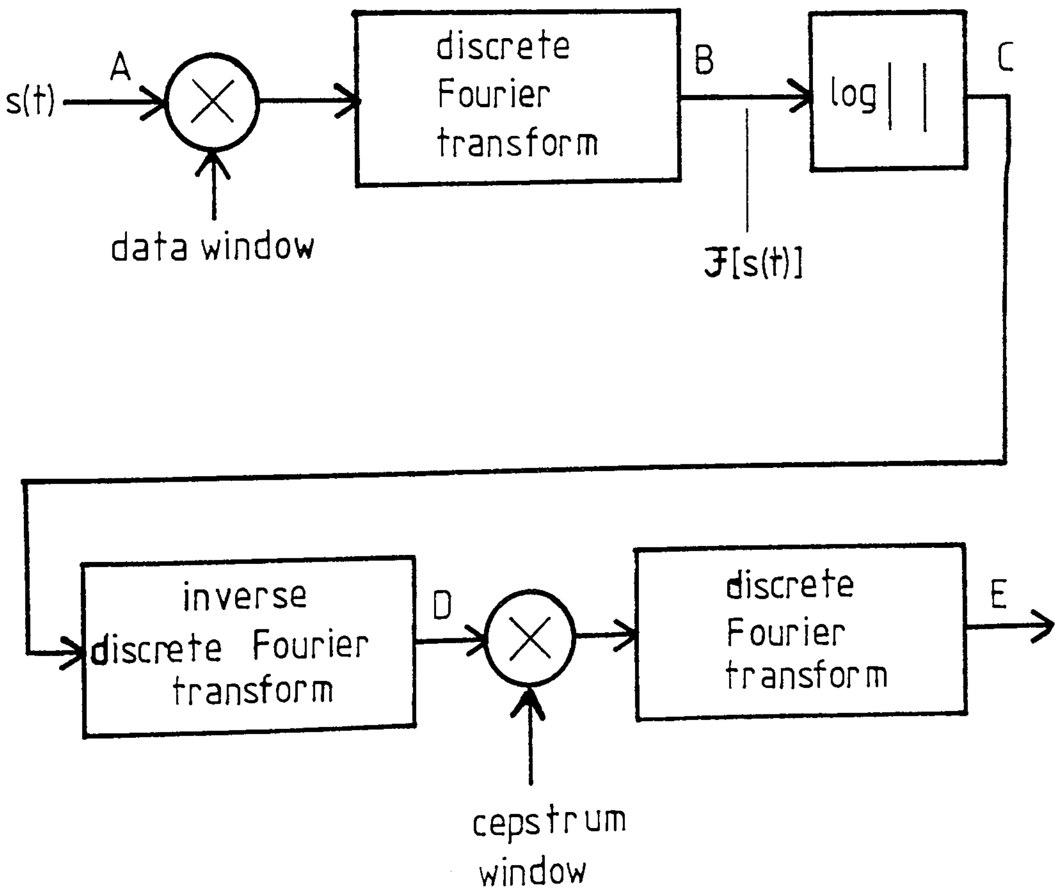


Figure 3-11: Homomorphic speech analysis

transforms of the glottal source and the vocal tract impulse response, i.e.

$$\mathfrak{F}[s(t)] = \mathfrak{F}[e(t)].\mathfrak{F}[h(t)],$$

where $\mathfrak{F}[x]$ indicates the Fourier transform of x ...(3.14)

The underlying principle of homomorphic processing is that the spectrum itself can be regarded as a signal, and can be processed by standard signal analysis techniques [N360]. The act of taking the logarithm of the magnitude of $\mathfrak{F}[s(t)]$ yields, at point C, the sum of the logarithms of $\mathfrak{F}[e(t)]$ and $\mathfrak{F}[h(t)]$, i.e. it separates the effects of the glottal source and the vocal tract. The inverse DFT of C, a linear operation, preserves the addition, giving at point D the CEPSTRUM of s(t), which is a combination of the cepstra of e(t) and h(t).

A sample spectrum and cepstrum for a voiced speech segment are shown in Figure 3-12 (after Noll [N360]). The logarithm power spectrum in Figure 3-12a displays a spectral periodicity resulting from the pitch-periodicity of the speech sample. The cepstrum (Figure 3-12b) has a sharp peak corresponding to this periodicity. The presence of a sharp peak in the cepstrum of a speech sample is an indication that the segment under analysis is voiced, and the location of the peak is a good indicator of the pitch-period.

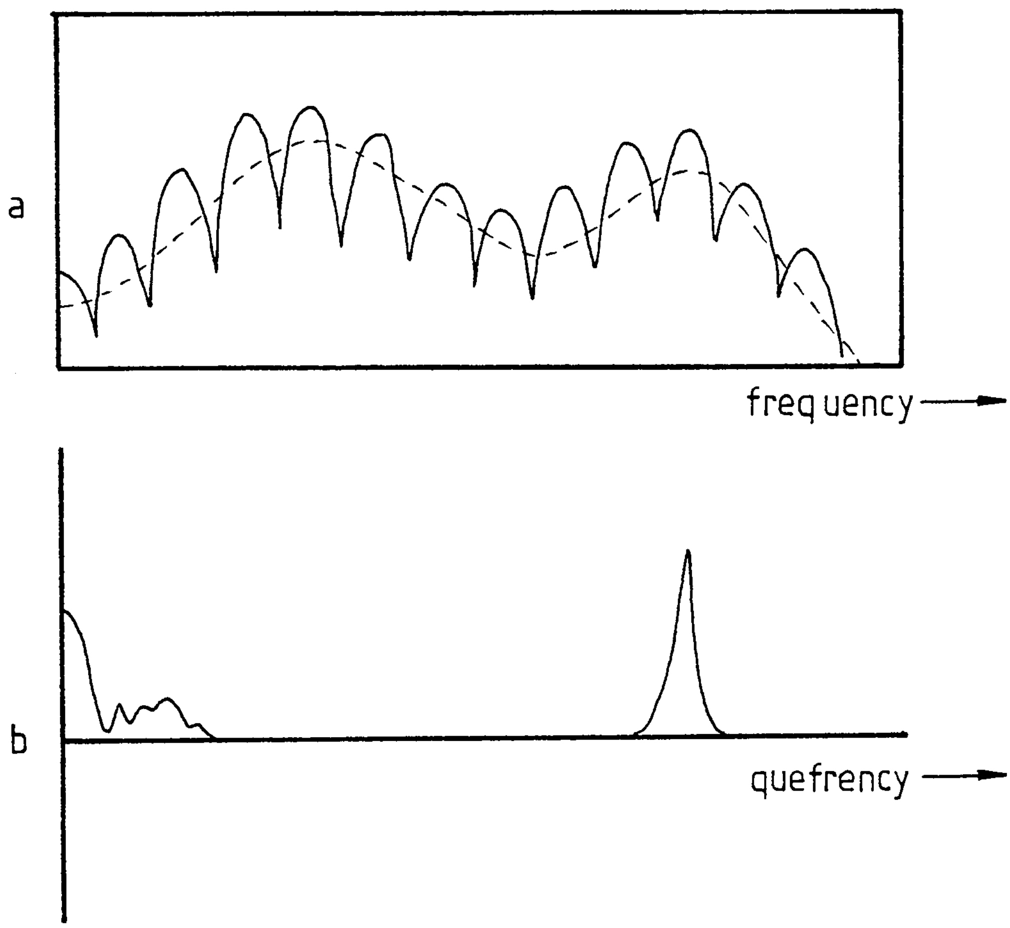


Figure 3-12: Spectrum (a) and cepstrum (b) for a voiced speech segment

Homomorphic analysis has been used as the basis of a number of pitch-period estimation systems [W362] [D363] [K364] and for the production of spectrogram-like displays [W365]. Programming examples are given in [I366].

3.4.4 LINEAR PREDICTIVE ANALYSIS

The linear predictive method of speech analysis [M367] [M368] is based upon the theory that a sample of speech can be approximated as a linear combination of the past 'p' speech samples, the present speech sample s(n) being predicted by:

$$s(n) = \sum_{K=1}^{p} a_k s(n-k) \qquad \ldots(3.15)$$

The predictor coefficients a_k in Equation 3.15 - the weighting coefficients of the linear combination - can be determined by minimising the square difference between the actual speech samples and the linearly predicted ones. Various methods of achieving this have been proposed, the two main ones being the COVARIANCE method [A369] and the AUTOCORRELATION method [I370].

The coefficients a_k are the parameters of the time-varying digital filter of Figure 2-8. In this case, the filter is a pth-order recursive digital filter with the transfer function:

$$H(z) = \frac{1}{1 - \sum_{K=1}^{P} a_k z^{-k}} \qquad \ldots(3.16)$$

Once obtained, the predictor coefficients a_k can be used in various ways to investigate the properties of the speech signal. These include spectrum analysis [A369] and formant frequency estimation [M371]. Linear predictive coding has been successfully used as an efficient method of speech coding for small speech synthesis systems [T372]. Linear predictive analysis has been used as the basis of several pitch-period detection schemes [M371] [M373]. Practical examples of linear prediction computation are given in [I374] [W375].

3.5 VOICED/UNVOICED/SILENCE DISCRIMINATION

Some means of discriminating between voiced and unvoiced sections of a spoken utterance, and between intervals containing speech and silent intervals, is necessary for most pitch period estimation schemes. In some cases the voiced/unvoiced/silence discrimination is an integral part of the analysis algorithm; in other cases, the voiced/unvoiced/silence discrimination is an independent process.

Various independent voiced/unvoiced/silence discriminators have been described, including those based upon pattern recognition [A376] and pattern classification [S377], methods using certain characteristics of the speech waveshape [C378], systems based upon delta modulation [U379], and analogue circuitry used to compare high- and low-frequency content of the speech signal [K380].

3.6 PERFORMANCE OF EXISTING SYSTEMS

An exhaustive comparative performance study of seven pitch-period detection systems has been carried out by Rabiner et al [R381]. The algorithms investigated fall into four groups:

(i) time-domain waveform methods -

data reduction method (DARD) [M313]

parallel processing method (PPROC) [G317]

(ii) autocorrelation methods -

modified autocorrelation method using clipping (AUTOC) [D334]

average magnitude difference function method (AMDF) [R335]

(iii) frequency-domain method -

cepstrum method (CEP) [S382]

(iv) <u>linear predictive coding (LPC) hybrid methods</u> -

simplified inverse filtering technique (SIFT) [M371]

spectral equalisation LPC method using Newton's transformation (LPC) [A383]

(the abbreviations in parentheses are used in the discussion of results)

Rabiner et al identify seven criteria which may be used as the basis for evaluation of pitch-period detection algorithms:

(i) the accuracy in estimating pitch-period

(ii) the accuracy in making a voiced/unvoiced decision

(iii) the robustness of the measurements (with respect to different speakers, transmission conditions and so on)

(iv) the speed of operation

(v) the complexity of the algorithm

(vi) the suitability for hardware implementation

(vii) the cost of a hardware implementation

The results presented in [R381] are based upon the first two criteria in this list, which are those most easily quantified and tabulated. Rabiner et al do, however, give some consideration to the remaining five criteria. The authors note that their evaluation is specifically related to speaker verification and digit recognition schemes.

A database of recorded utterances was used in the performance study, spoken by three male adults, two female adults and a child, representing an overall fundamental frequency range from some 50Hz to over 400Hz. The test utterances used were four monosyllabic nonsense words and four sentences. Three different recording conditions were used: a close-talking microphone recording, a standard telephone transmission recording, and a wideband high-quality microphone recording.

For the purposes of the study, a 'standard' pitch-period contour was produced for each of the test recordings using a sophisticated semiautomatic pitch-period detection scheme [M384]. During error analysis a nonlinear smoothing algorithm was applied in order to detect and correct several types of pitch-period detection error [R385]. Data were recorded with and without the inclusion of this additional error correction mechanism.

For every utterance in the database, the standard pitch-period contour, $p_s(m)$ was compared with the pitch-period contours from each pitch-period detector, $p_j(m)$ $(1 <= j <= 7)$. For a given pitch-period detector, a comparison between $p_s(m)$ and $p_j(m)$ yields one of four possible states for each m:

(i) $p_s(m) = 0$, $p_j(m) = 0$ - no error

(ii) $p_s(m) = 0,\ p_j(m) \neq 0$ - an unvoiced-to-voiced error

(iii) $p_s(m) \neq 0,\ p_j(m) = 0$ - a voiced-to-unvoiced error

(iv) $p_s(m) = P_1 \neq 0,\ p_j(m) = P_2 \neq 0$ -
two types of error can exist in this case, if $P_1 \neq P_2$.
Defining voiced error e(m) as:

$$e(m) = P_1 - P_2 \qquad ...(3.17)$$

if $|e(m)|$ >= 1mS then this is classified as a GROSS pitch-period error;
if $|e(m)|$ < 1mS then this is classified as a FINE pitch-period error.

On the basis of the above four possible states, five distinct measurements of the performance of each pitch detector were derived:

(i) GROSS ERROR COUNT - the number of gross pitch-period errors per utterance
(ii) MEAN OF FINE PITCH ERRORS - a measure of the bias of the pitch-period measurement during voiced intervals
(iii) STANDARD DEVIATION OF FINE PITCH ERRORS - a measure of the accuracy of the pitch-period detector in measuring pitch-period during voiced intervals
(iv) VOICED-TO-UNVOICED ERROR RATE - the accuracy of the pitch-period detector in correctly classifying voiced intervals
(v) UNVOICED-TO-VOICED ERROR RATE - the accuracy of the

pitch-period detector in correctly classifying unvoiced intervals.

Performance ratings were produced for each of the pitch-period detectors under test, based upon empirical evaluation of the average error ratings for the pitch-period detector, summed across speakers. The results are reproduced graphically in Figures 3-13a..3-13l (based upon results quoted in [R381]). It should be noted that a high error rating indicates low performance. Figure 3-14 displays graphically the overall error ratings for each of the pitch-period detectors, derived by summing the individual error ratings shown in Figure 3-13. Finally, Figure 3-15 displays the processing time required by each of the pitch-period detectors to process one second of speech.

The individual error ratings of Figure 3-13, which are summed across the range of speakers, may be influenced by a particular pitch-period detector performing poorly with just one speaker; several detectors produced poor results with very low and very high fundamental frequencies. Furthermore, the overall error ratings of Figure 3-14 may be influenced by poor performance in certain tests; the relatively high error rating of the cepstrum pitch-period detector is largely attributable to poor performance in detecting voiced-to-unvoiced transitions [R381,p417].

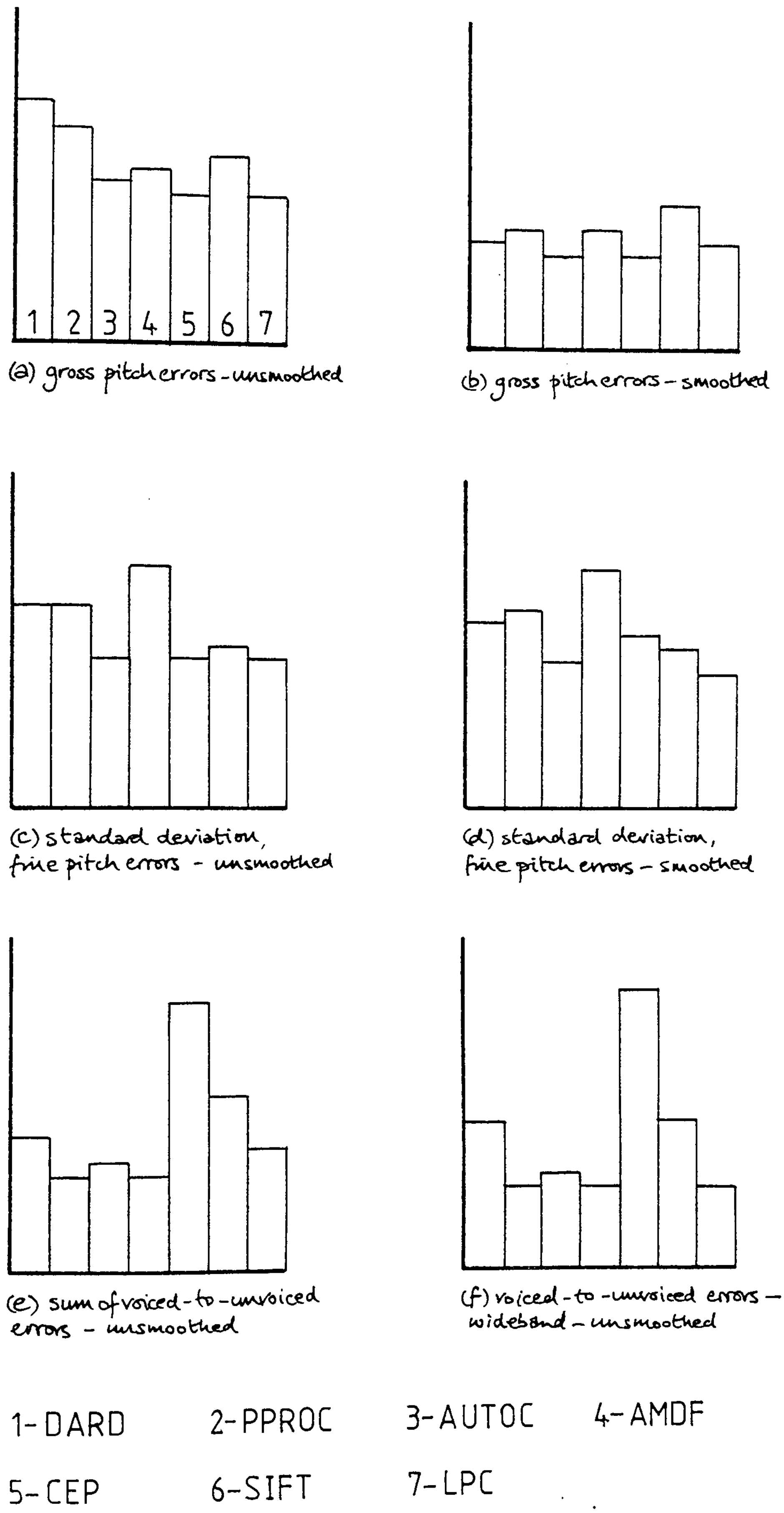
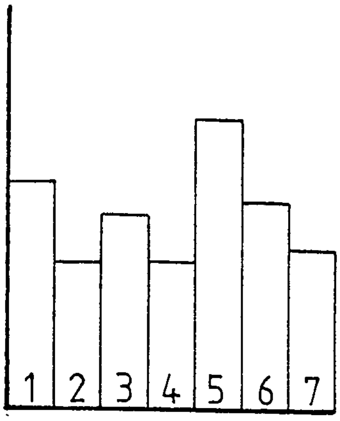
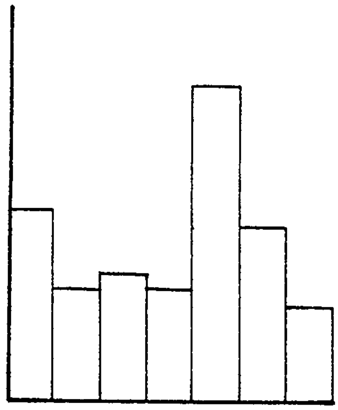


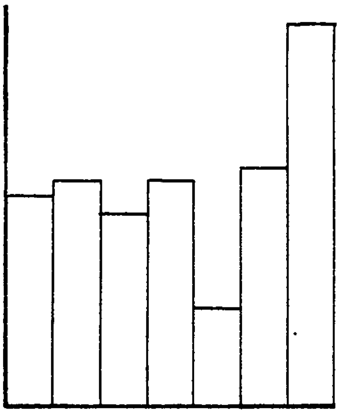
Figure 3-13: Individual performance scores of seven pitch-period detection algorithms



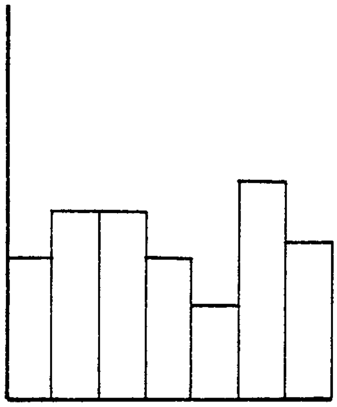
(g) sum of voiced-to unvoiced errors - smoothed



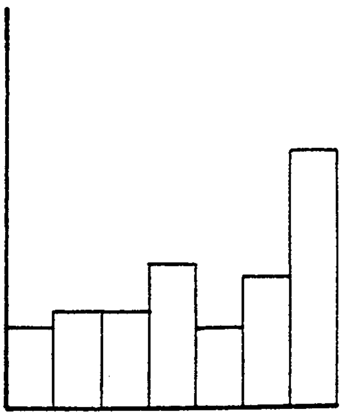
(h) voiced-to-unvoiced errors - wideband - smoothed



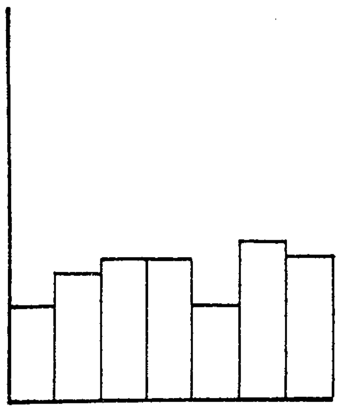
(i) sum of unvoiced-to-voiced errors - unsmoothed



(j) unvoiced-to-voiced errors - wideband - unsmoothed



(k) sum of unvoiced-to-voiced errors - smoothed



(l) unvoiced-to-voiced errors - wideband - smoothed

[for key see previous page]

Figure 3-13 (contd.)

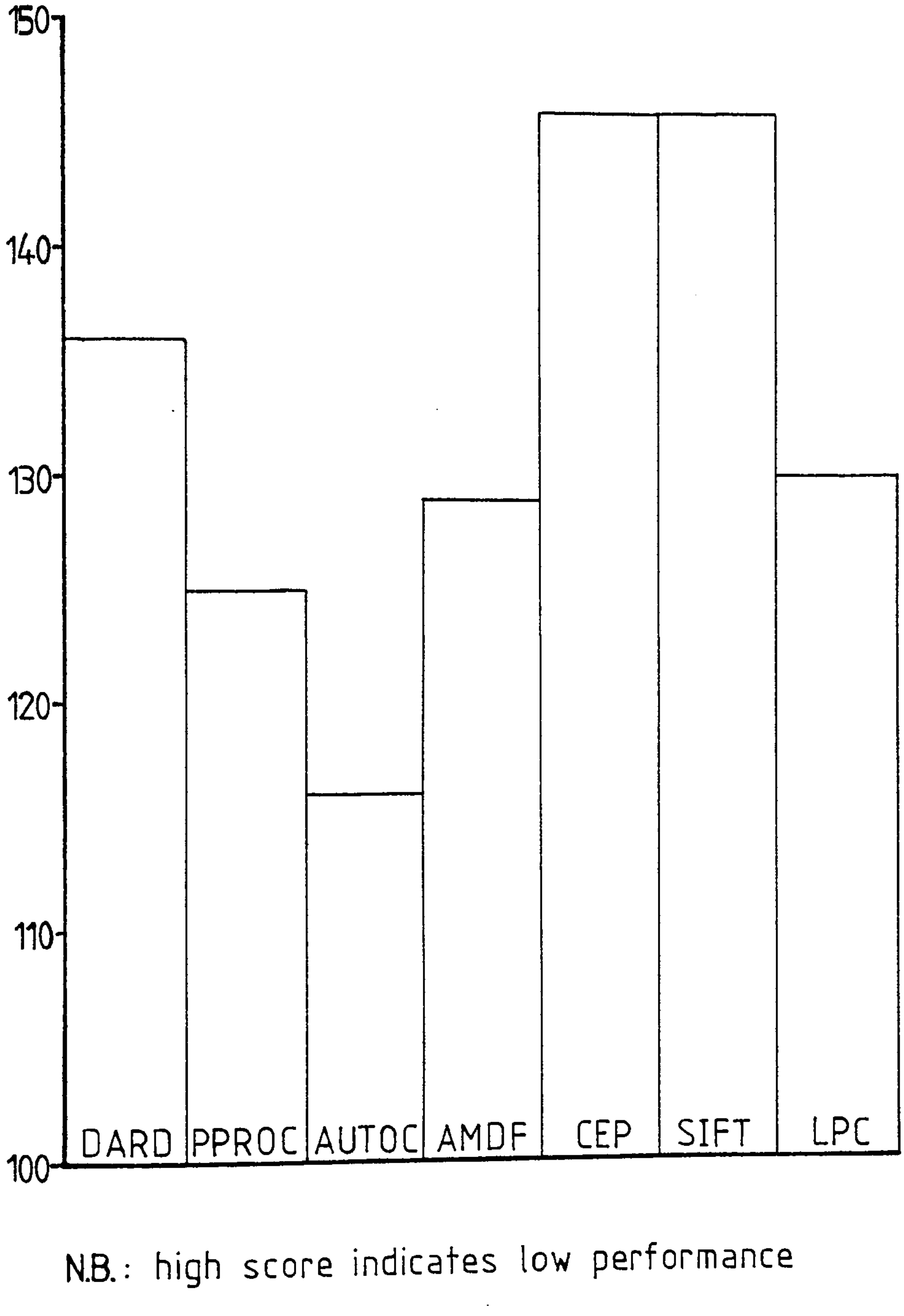


Figure 3-14: Overall performance scores of seven pitch-period detection algorithms

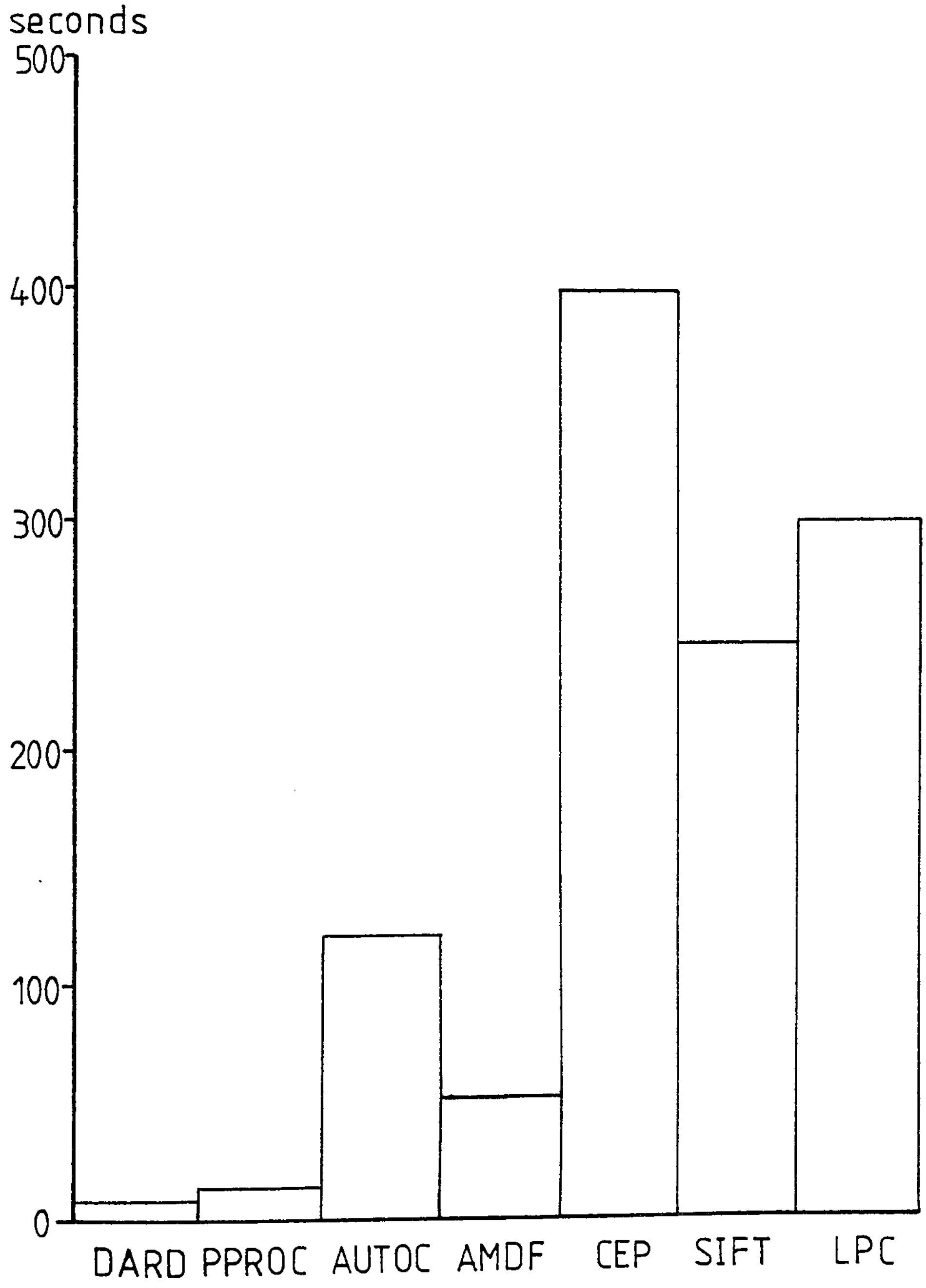


Figure 3-15: Processing time for one second of speech for seven pitch-period detection algorithms

The processing time requirements illustrated in Figure 3-15 give an indication of the processing complexity of the various pitch-period detectors. For the purpose of this study, each detector was implemented in Fortran on a NOVA 800 minicomputer (cycle time 800nS, add time 1.6μS, multiply time 3.6μS).
Processing time requirements of this particular implementation of any of the pitch-period detectors do not necessarily reflect the performance of the implementation chosen by the original author. Table 3-1 gives details of the primary implementations of the pitch-period detectors, where these are provided by the authors. It will be seen that the implementation for the comparative performance study is generally much slower than the original implementation. Nevertheless, the processing times recorded during the study give an indication of the computational complexity of the various detectors, and it is noted that there is no general inverse relationship between the overall error ratings and the processing times.

Name	Primary implementation
DARD	Real-time performance on a general-purpose medium-speed machine
PPROC	50 times real-time on 'moderately fast' Lincoln Laboratory TX-2 machine
AUTOC	Real-time operation in a dedicated hardware implementation
AMDF	Real-time operation on the GTE Sylvania Programmable Signal Processor
SIFT	10 - 20 times real-time when implemented on a general-purpose minicomputer. Prediction of real-time operation when implemented in dedicated hardware
CEP	120 times real-time to estimate three formants, predict the pitch-period and plot the results on microfilm. High-quality results are limited to male speakers

Table 3-1
Primary implementations of pitch-period detectors

The lack of any absolute yardstick for use in a comparative study of this type may give some cause for doubt concerning the validity of the results. The 'semiautomatic pitch detector' used to provide the standard pitch-period contours probably represents the greatest accuracy obtainable at the time of the study. It is interesting to note that the interactive evaluation of autocorrelation, cepstrum and waveform displays provided by this system requires some 1800 times real-time.

3.7 EVALUATION OF EXISTING SYSTEMS WITH REGARD TO THE PROPOSED SPEECH ANALYSIS SYSTEM

Rabiner et al stress that interpretation of performance ratings of pitch-period detection algorithms must depend upon the intended application of the analysis. For some applications, extreme accuracy is a major requirement, and high performance with regard to gross errors is necessary. For many other applications, the dominant requirement is the perceptual accuracy of pitch-period detection - how faithfully the pitch-period contour measured by the pitch-period detector matches the natural excitation pitch-period contour in terms of synthetic speech quality [R381,p401]. For these applications, a subjective assessment by means of perceptual tests is more relevant in evaluating pitch-period detectors (such a study has been carried out by McGonegal et al [M386]). Rostron and Welbourn point out that in a training aid for helping the deaf to acquire appropriate intonation patterns:

> "...any lack of following the exact pitch contours, which is liable to occur in aperiodic bursts of voicing, may be of advantage, since it is the overall general pitch which has to be matched rather than any small and sudden changes."
>
> [R113]

This suggests that for visual feedback teaching applications a high priority might be placed on the ability of a pitch-period detector to produce a smoothed display of overall general pitch-period, rather than a detailed display including cycle-to-cycle variations. One of the major requirements of the proposed speech analysis system is the provision of a real-time display, without which much of the effectiveness of its use as a training aid would be lost.

The simplest course of action for the present project would have been to have implemented an existing speech analysis system. From the point of view of real-time operation, the simple 'pitchmeter' systems described in Section 3.4.1.1 present few problems, and have been used in a number of attempts to provide speech aids for the deaf (see Section 1.7.3). However, the present author felt that the simplicity of these systems would almost certainly imply a high error rate, especially when analysing speech via a microphone, with the consequent complexity of the waveform. A low-pass filter can be used to reduce the complexity of the waveform [B155] [B156], but this involves the problem of choosing a suitable filter cutoff frequency, which should, ideally, be tailored dynamically according to the instantaneous fundamental frequency of the speech signal. Rostron and Welbourn [R113] attempt to achieve this, but their system involves the use of an accelerometer taped to the neck of the speaker. This approach was considered to be unsuitable for the present project as it is relatively intrusive, and cannot be used to analyse conventional tape-recorded speech. The same objection ruled out the use of the Laryngograph/Voiscope system [F107], although the present author was most impressed with the performance of this particular system, which has found widespread use in applications where the need to use neck electrodes does not present a problem.

The real-time operation requirement proved to be a major obstacle to the implementation of systems based upon homomorphic analysis, linear predictive analysis and spectral analysis by Fourier transformation, which appeared to be ruled out for the purposes of the present project due to their high computational demands. Examination of Figure 3-15 suggests that the time-domain waveform methods of analysis have the least computational complexity, and are therefore the most suited to real-time implementations. The parallel processing method performed well in the comparative performance study, both in terms of processing time and in terms of overall error rating. It will, however, be noted that the original implementation of this pitch-period detector [G317] required approximately 50 times real-time on the 'moderately fast' Lincoln Laboratories TX-2 computer.

It seemed clear that, for the purposes of the present project, some adaptation of one or more existing methods of speech analysis would be necessary. Processing time appeared to be the most critical aspect affecting the choice of analysis system; methods of reducing processing time are examined in Chapter Four.

CHAPTER FOUR

DEVELOPMENT OF THE PRESENT PROJECT

4.1 METHODS OF ACHIEVING REDUCED PROCESSING TIME

The pitch-period detection algorithms compared in the study by Rabiner et al [R381] were all required to be coded in Fortran and run on a general-purpose minicomputer. For production systems, several different approaches have been employed to reduce processing time and thereby improve system performance.

4.1.1 Specialised computer architecture for signal processing

Allen [A401] notes that processing time may be speeded up by a factor of up to 100 when a signal-processing algorithm is implemented on a special-purpose signal-processing computer.

4.1.1.1 Factors affecting the design of signal-processing computers

Allen identifies five structural factors which affect the design of signal-processing computers.

1) Technology

Small machines aiming at low cost generally employ standard transistor-transistor logic (TTL) or Schottky TTL logic circuits. Emitter-coupled logic (ECL) has a better speed-power product, but is expensive in terms of logic cost, power dissipation, and the necessity for special constructional techniques. Random logic may be replaced by a programmed logic array (PLA), which offers a saving in wiring and space and enhances the speed of operation.

2) <u>Algorithm structure</u>

The 'sum-of-products' structure is basic to the implementation of digital filters, fast Fourier transform algorithms, and specialised multiplication units. This structure is ideally suited to 'pipelining', in which a process is divided into a number of sequential tasks and is executed in 'assembly-line' fashion, with a number of separate processes in operation at a given time.

3) <u>Data structures</u>

Memory structures may be optimised, with separate memory for instructions and data, and partitioned memories for the real and imaginary parts of complex numbers.

4) <u>Programming languages</u>

While Fortran is usually available on the larger signal-processing computers, the smaller machines tend to be limited to assembly language coding. In the absence of widespread high-level signal-processing languages, highly optimised routines have been developed for frequently-used algorithms such as the FFT, and these may be linked into the user's code. High-level language programming has the advantage of easy and rapid implementation. Optimising Fortran compilers are available, which detect parallelism and possible vector calculations (suitable for a pipelined approach) in the source code.

5) Hardware architecture

Parallelism occurs in many forms, including pipelining. Several independent arithmetic elements may be made available, with duplicated memories to facilitate high-speed access.

In microprogrammed designs, the memory for the relatively long instructions may be kept separate from the data memory, with memory accesses overlapped in time to speed execution. Registers available for external manipulation may be mapped into the signal processor's memory space for direct addressing.

Specialised arithmetic units may be provided to compute frequently-used expressions such as (X.Y)+Z, which is needed for the sum-of-products calculation. On some machines, the arithmetic units can be reconfigured dynamically under microprogram control, allowing the user to tailor the structure of the arithmetic units to his particular needs.

Multiple processors may be provided, each distinct processor having its own program dedicated to one particular task. Such systems require careful programming to ensure a balanced distribution of programming effort, and to reduce the risk of conflict when processors contend for shared resources.

High-speed combinational circuits may be provided to cope with tasks such as array multiplication.

There may be provision for automatic trapping of certain error conditions - such as overflow - together with appropriate recovery procedures. This obviates the need for explicit test instructions in the source code.

A smaller signal-processing computer may be connected to a general-purpose mainframe machine to provide bulk memory, access to system peripherals, compilation facilities, and so on.

4.1.1.2 Specific special-purpose signal-processing computers

A large number of signal-processing computers have been constructed, each making use of one or more of the structural factors outlined above. Direct performance comparisons are difficult, since specific machines may be tailored towards a particular type of processing. Comparisons of various features of a number of signal-processing computers are given in [A401] and [Z402].

Specific examples of special-purpose processors are the Small Signal Processor (SSP) [A401,p629], which features a highly combinational design, the Fast Digital Processor (FDP) [G403], which makes use of four identical arithmetic units, each containing an adder and a multiplier, and the Signal Processor with Distributed Arithmetic (SDA) [Z402], which is constructed from microcoded bit-slice microprocessors, and features distributed arithmetic capabilities.

4.1.2 Dedicated hardware systems

The special-purpose signal-processing computers mentioned in the above Section are programmable processors which may be used to implement a specific algorithm. Speech analysis methods may also be implemented in a dedicated hardware system. An example of such a system is the real-time digital hardware pitch-period detector of Dubnowski et al [D334], a software version of which was included in the comparative performance study outlined above (Section 3.6). This system achieves real-time autocorrelation analysis by the use of some 150 integrated circuits, including three 100-word by 8-bit metal-oxide semiconductor (MOS) shift registers, a fast 512-word by 2-bit bipolar memory and various standard TTL circuits.

Markel [M371] predicts that a dedicated hardware implementation of the SIFT algorithm should run in real-time, compared with ten to twenty times real-time when implemented on a general-purpose minicomputer.

4.1.3 Hardware/software hybrid systems

An alternative to complete dedicated hardware systems is a hardware/software hybrid approach, in which part of the overall system function is implemented in a dedicated hardware system, information from the latter being fed to a programmable processor for further processing. In this way it may be possible to achieve real-time performance by performing some time-consuming task such as a highly iterative calculation externally to the main processor.

Such an approach is proposed by Ross et al [R335] for realising a real-time average magnitude difference function (AMDF) analysis system. They suggest that the AMDF itself be generated in an external hardwired device, with pitch-period extraction logic being implemented in the programmable processor. A real-time speech analysis operation using the AMDF forms part of the TSP-100 specch digitizer [M404]; AMDF values are generated using serial hardware, and are read into an Intel 8080 microprocessor via interrupts.

4.1.4 Data reduction techniques

The methods of achieving improved performance considered so far have concentrated upon means of speeding up the processing of data presented to the analysis system. An alternative approach is to examine ways in which the processing burden may be lessened by reducing the amount of data presented to the system. This is the essence of the 'data reduction' approach to speech analysis, which is embodied in several published speech analysis systems, but forms the basis of one particular system described by Miller [M313]. Basically, the data reduction strategy attempts to overcome the main obstacle to computational efficiency which hinders most speech analysis systems, namely the need to perform hundreds of computations on each sample of the speech signal. In Miller's system, each sample undergoes a small number of computations, sufficient to allow the construction of a compact data structure representing the speech signal, and containing approximately two orders of magnitude less information than the original sampled signal. Within this data structure the signal is represented in terms of its significant features, which for the purposes of Miller's algorithm are the excursion cycles – those portions of the waveform between consecutive zero-crossings.

Tucker and Bates [T314], who describe a superficially similar approach to pitch-period estimation, note that "...a representation of the signal by its pulse parameters (i.e. 'features') is significantly more compact than the sampled signal representation".

4.1.5 Analogue signal-processing techniques

4.1.5.1 Analogue techniques compared with digital techniques

The theory of digital signal-processing is well established [G405] [R406] [O407], and pitch-period detection systems described during the past decade employ almost exclusively digital signal-processing techniques. Bruce [B408] contrasts hardware implementations of digital filters, based upon techniques described by Jackson et al [J409], with their analogue equivalents, and finds that the digital circuits are favourable, both in terms of physical size, and with regard to flexibility in changing the processing algorithm parameters. The advent of relatively cheap microelectronic circuitry, and in particular the microprocessor, has strengthened still further that attraction of digital approaches to signal-processing tasks. From the point of view of commercial production runs, digital systems do not, in general, suffer from the problems of tolerance of discrete analogue components, which lead to spreads in realised performance; a carefully designed digital processor may be rigidly specified while being amenable to change.

4.1.5.2 Specific analogue circuits

While recent progress in microelectronics has been dominated by advances in digital techniques and circuitry, there have also been advances in analogue component design.

4.1.5.2.1 The phase-locked loop

The basic phase-locked loop (PLL) system [M410] [T304,Section 10.7] is illustrated in Figure 4-1. The closed-loop frequency-feedback system comprises a phase comparator, a low-pass filter and a voltage-controlled oscillator (VCO). When an input signal Vs(t) is applied to the PLL, the phase comparator compares the phase and frequency of this signal with the VCO output signal Vo(t), and generates an error voltage Ve(t) proportional to the phase and frequency difference. Ve(t) is low-pass filtered, and the resulting voltage Vd(t) is applied to the VCO control input, causing Vo(t) to vary in such a way as to reduce the frequency difference between Vo(t) and Vs(t). Once the VCO output frequency is the same as the frequency of the input signal, with a finite phase difference between the two signals, the PLL is said to be 'locked' on to the input signal, and features the ability to track the input signal over a certain frequency range. The use of an active low-pass filter improves the capture and tracking abilities of the PLL [C411].

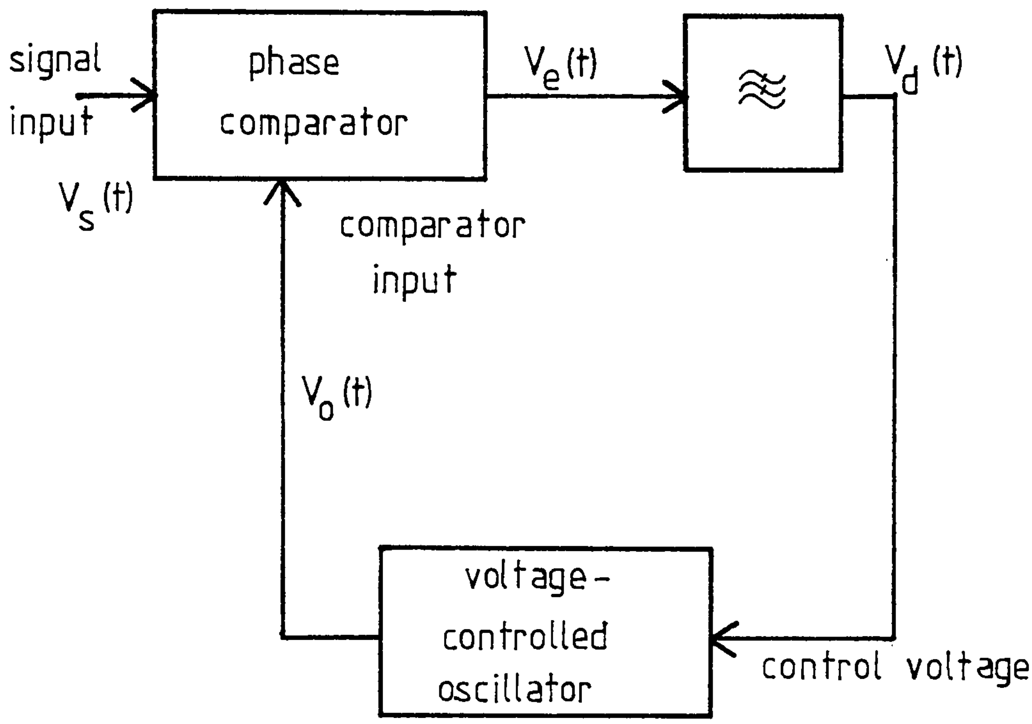


Figure 4-1: Basic phase-locked loop system

An integrated CMOS PLL [R412] has been used as the basis of a fundamental frequency extraction filter operating at audio frequencies [R413]. The tracking abilities of the PLL are potentially relevant to a speech pitch-period detection system.

4.1.5.2.2 Analogue filter implementation

Integrated operational amplifiers may be used to construct efficient active filters [M414,Section 16-6] [T415,Chapter 8] [G416,Section 3.5]. The physical size of filters may be reduced, with an increase in performance, by the use of gyrators, charge-coupled device filters and switched-capacitor filters [B417]

4.1.5.2.3 Other analogue signal-processing functions

Analogue circuitry, based upon integrated operational amplifiers, may be used to perform various measurements and conditioning functions upon signals, for example waveform integration and differentiation, waveform peak detection and waveform clipping, absolute value and RMS value computation and zero-crossing detection. Various examples are given in [M414] [T415] [G416].

4.1.5.3 Practical examples of analogue speech analysis

Early 'pitchmeters' (Section 3.4.1.1) employed exclusively analogue techniques in determining the pitch-period of speech signals. As digital computers became available, there was a general tendency to turn to digital techniques to the exclusion of analogue techniques, and relatively few hybrid analogue/digital pitch-period detection systems are to be found in the literature. Bruce [B408] predicts that the application of analogue techniques will be limited to three main areas: systems with very simple processing needs, the interfacing of transducers to signal processors, and high-frequency systems.

A recent editorial comment suggests that there may still be a role for analogue circuitry in hybrid systems for real-time applications:

> "Digital computers can be a 'square peg in a round hole' when faced with such blazing real-time systems as aircraft flight control, fluid flow dynamics, or predictive/adaptive control loops - and so engineers should know when the better bet is a hybrid computer: several analog processors combined with a digital computer geared for real-time work with interrupts."
> [I418]

4.2 SELECTION OF HARDWARE FOR THE PRESENT PROJECT

4.2.1 Signal-processing computers

The use of a special-purpose signal-processing computer was rejected, as no machine was already available, and the purchase of new hardware was prohibitively expensive. One possible approach which was considered, but not pursued, was the construction of a programmable signal-processor using microprogrammable bit-slice microprocessors. The use of such a system allows the designer to develop a microcoded instruction set specifically suited to the intended application, allowing efficient operation and increased processing speed compared with non-microprogrammable processors. However, at the time of choosing a system, such high-performance microprocessors were relatively new, and little development support was available.

4.2.2 Analogue systems

Wholly analogue processing was considered, and some attention was given to possible systems using phase-locked loops and analogue filter banks. Tentative experiments were conducted using the NE565A and CD4046A PLL circuits as fundamental frequency trackers. While some success was achieved, particularly with the CD4046A device, it was concluded that stability was insufficient for the intended purposes of the project, and there was a tendency for the phase-locked loop to lock on to harmonics of the fundamental frequency. Experiments were also carried out on a system similar to (but developed in ignorance of) the harmonic identification (HIPEX) system of Miller [M355], in which various harmonics of the voice fundamental frequency are isolated by means of a bank of bandpass filters, and an attempt is made to recognise coincidences in the occurrence of pulse trains derived from the harmonic components. These coincidences tend to correspond to the fundamental periodicity of the waveform [S354]. Some practical difficulties were experienced in achieving bandpass filters of sufficient frequency resolution, and in devising a logic system for the coincidence detection process without resort to a programmable digital processor.

4.2.3 Dedicated hardware and hardware/software systems

Dedicated hardware systems, similar to the real-time hardware clipped autocorrelation system of Dubnowski et al [D334] were felt to be more suitable for the implementation of proven techniques such as autocorrelation, where the underlying arithmetic and logical operations are basically trivial. The present author was interested in the development of some heuristic or adaptive procedures to improve the quality of analysis by taking into account the characteristics of the particular speech signal being analysed. This suggested that a programmable processor would have to be included in the system. A scheme such as that shown in Figure 4-2 was considered, in which a dedicated hardware processor is used to implement an autocorrelation algorithm, and a programmable digital processor provides feedback control signals to adapt the autocorrelation

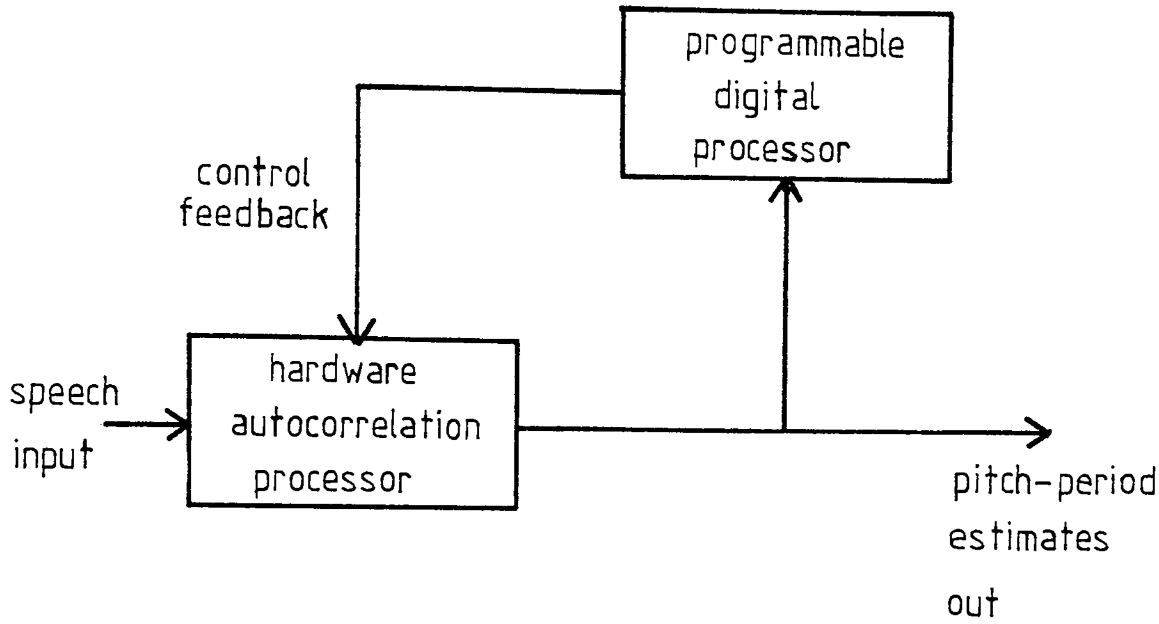


Figure 4-2: Hybrid hardware/software system

process in some way to the particular speech signal being analysed.

In his study of the use of autocorrelation analysis for pitch-period detection, Rabiner [R301] isolates as one of the major remaining problems the provision of a variable analysis frame size for the autocorrelation process. The analysis frame should be large enough to encompass at least two pitch-periods, but not so large that it covers many pitch-periods. Assuming that, for a given speaker, the range of fundamental frequency variation within an utterance will be within one octave [R301] - a factor of 2 to 1 - it follows that the variability of frame size for a given speaker is less important than the variability from speaker to speaker. A speaker with high average fundamental frequency may require an average frame size of 4mS, while a low-pitched speaker may require a frame size as large as 40mS. The system described by Dubnowski et al [D334] seeks a compromise with a fixed frame of 30mS length. Rabiner [R301] suggests that the frame size should be based upon the average pitch-period $\bar{p}(m)$ of the speaker:

$$\bar{p}(m) = \frac{1}{N_m} \sum_{i=1}^{N_m} p(i), \quad \text{for } N_m >= 10$$
$$= 100, \quad \text{for } N_m < 10$$

...(4.1)

where p(i) is the pitch-period of the ith voiced frame, N_m is the number of voiced frames up to the mth frame, and $\bar{p}(m)$ is expressed as the number of samples at a 10kHz sampling rate (10 samples per millisecond). For values of N_m less than 10, the condition $\bar{p}(m) = 100$ is used to ensure a reasonable initial frame size. The frame length $L(m)$ is related to $\bar{p}(m)$ by the simple rule:

$$L(m) = 3.\bar{p}(m) \quad ...(4.2)$$

the factor of three allowing for up to 50% variation in pitch-period from the average value. Bounds are set on the range of permissible values for $L(m)$:

$$100 <= L(m) <= 600, \text{ for all } m \quad ...(4.3)$$

Rabiner suggests that the application of such an adaptive frame length scheme to an autocorrelation system enhances the accuracy of the results obtained, and, in cases where a small frame length is adopted, reduces the number of calculations required to compute the autocorrelation function. The converse, however, also applies, and the overall processing speed is reduced when long frames are used for speakers with low average fundamental frequencies.

At first sight, such a scheme would appear to be ideal for implementation in the system shown in Figure 4-2, with the programmable processor providing feedback information controlling the length of the analysis frame. However, the variation of parameters such as frame length in a dedicated hardware system is difficult to achieve, since frame length is governed by the physical devices used: in the case of the system used by Dubnowski et al [D334], the 300-sample analysis frame processor is constructed from three 100-word shift registers, and variation of the length of the frame would necessitate the inclusion of an extremely complex array of logic circuits. It would seem that adaptive schemes are far better suited to software implementations, where variation in parameters such as frame length may involve simply altering the value of a pointer variable.

4.2.4 Microprocessor-based systems

In view of the fact that special-purpose signal-processing computers, wholly analogue systems, and dedicated hardware systems had been rejected for the current project, attention was turned to the selection of a suitable programmable digital processor to form the basis of a speech analysis system. The Department of Educational Enquiry at the University of Aston has no minicomputer facilities, and the University's timeshared mainframe computers, to which the Department has access, are not suitable for interactive real-time processing. From the point of view of cost, and bearing in mind the requirement that the system should be portable, a microprocessor-based system was decided upon.

At the time when consideration was being given to the purchase of equipment - Summer 1978 - the microprocessor market was dominated by a small group of popular 8-bit general-purpose devices: the Intel 8080 [I419], the Zilog Z-80 [Z420], the MOS Technology 6502 [M421] and the Motorola 6800 [M422]. The 6800 device was initially chosen as the most suitable, mainly in view of the fact that the University of Aston Computer Centre was equipped with a Motorola EXORciser software/hardware development system for the 6800 microprocessor [M423]. The EXORciser provides a suitable environment for the user to develop and test system hardware and software before building a dedicated microprocessor version. After some initial work had been carried out, based upon the 6800, the Department of Educational Enquiry purchased two CBM PET 2001-32N microcomputers [C424], and it was decided to base the speech analysis system upon one of these machines. The PET uses the MOS Technology 6502 microprocessor, which is an enhanced version of the 6800, but with a different instruction set. The PET lacks the software and hardware debugging tools which are provided on the EXORciser, but has the advantage of a built-in BASIC interpreter, which simplifies the writing of interactive programs. Use of the PET was attractive for several further reasons. When not being used as the basis of a speech analysis system, the machine could be used for other computing purposes in the Department. The cost of the PET was, however, low enough to make feasible the purchase of a machine specifically for use with the speech analysis system at a later date. According to a survey published in April 1981, the PET was at that time considered to be the most popular microcomputer for use in educational establishments in this country [C425], and a speech analysis system based upon a PET would be ideally suited to implementation in other institutions, if this were felt to be desirable. At the time of choosing the machine it was felt that the 6502 microprocessor would be adequate for the task in hand; more recent tests [K426] suggest that the 6502 is, in fact, one of the most suitable microprocessors for real-time signal-processing applications, in terms of processing speed, instruction set, and addressing modes.

4.3 SELECTION OF DISPLAY DEVICE

Some form of computer-controlled display device is an essential component of a visual feedback pitch-period analysis system. It was initially felt that a standard visual display device (VDU) might be employed for this task. The built-in character display capabilities of a VDU can be used to form a low-resolution graphical display, and it was intended to investigate the possibility of implementing high-resolution graphics on a VDU. However, it was eventually decided to purchase a storage oscilloscope for use as a display device. The Gould Advance OS4000 digital storage oscilloscope chosen for this task [G427] has the advantage of being an excellent tool for the examination of speech waveforms, and a useful source of digital waveform samples with the addition of the OS4002 data output option.

4.4 SELECTION OF ANALYSIS METHOD

A number of factors had to be taken into account before a final selection of analysis method could be made.

4.4.1 Microprocessor implementation of algorithms

There are a number of references in the literature to speech analysis algorithm implementation on microprocessor-based systems, two of which specifically refer to the implementation of FFT algorithms on the PET [H428] [R429]. The FFT generally requires approximately $4.N.\log_2 N$ real multiply-add operations for an N-point sequence (analysis of N sample points) [S303] [A401]. Assuming that a multiply-add operation in a loop takes 6mS when implemented in BASIC on a PET [R429], a 1024-point FFT requires

$$4.1024.10.6.10^{-3} \simeq 246 \text{ seconds}$$

When implemented in machine code on a PET (multiply-add operation in a loop taking approximately 250µS), a 1024-point FFT requires

$$4.1024.10.250.10^{-6} \simeq 10 \text{ seconds}$$

No doubt the latter time could be reduced to a certain extent by optimisation of the machine code. Nevertheless, this figure must be compared with the performance of special-purpose signal-processing computers and hardware FFT implementations, which typically require less than 10mS to compute a 1024-point FFT [B339] [A401] [Z402].

The above figures based upon the PET microcomputer illustrate two points: firstly, that programs written in BASIC run far too slowly to be considered for real-time signal-processing, and secondly, that even programs written directly in machine-code on the 6502 microprocessor may require considerable time for processing. A further illustration of these points came when the present author wrote BASIC and machine-code routines to perform autocorrelation on 1024-sample segments of speech, representing approximately 100mS sampled at approximately 10kHz. The BASIC program, which used multiplication in the calculation of the autocorrelation function, ran for approximately 29 minutes; the machine-code routine, which was a software simulation of the real-time hardware autocorrelation system of Dubnowski et al [D334] ran for approximately 10 seconds.

4.4.2 Reducing computation time in a microprocessor-based system

Miller's data reduction approach to pitch-period detection [M313] specifically aims at a reduction in computational complexity, and the results of the comparative performance study by Rabiner et al [R381] suggest that time-domain waveform algorithms such as that employed by Miller are the most suitable for real-time operation.

The structure of Miller's system is illustrated in Figure 4-3. The speech signal, after combined low- and high-pass filtering, is sampled using a 20kHz analogue-to-digital converter (ADC). For the purpose of later stages of the algorithm, the sampled waveform is first examined numerically to isolate the excursion cycles; Figure 4-4 illustrates a section of voiced waveform with positive excursion cycles shaded. The 'area' of each positive excursion cycle - a coarse approximation to the energy in the excursion cycle - is calculated numerically by summation of the samples within the excursion cycle, a form of numerical integration. The bounds of each positive excursion cycle are

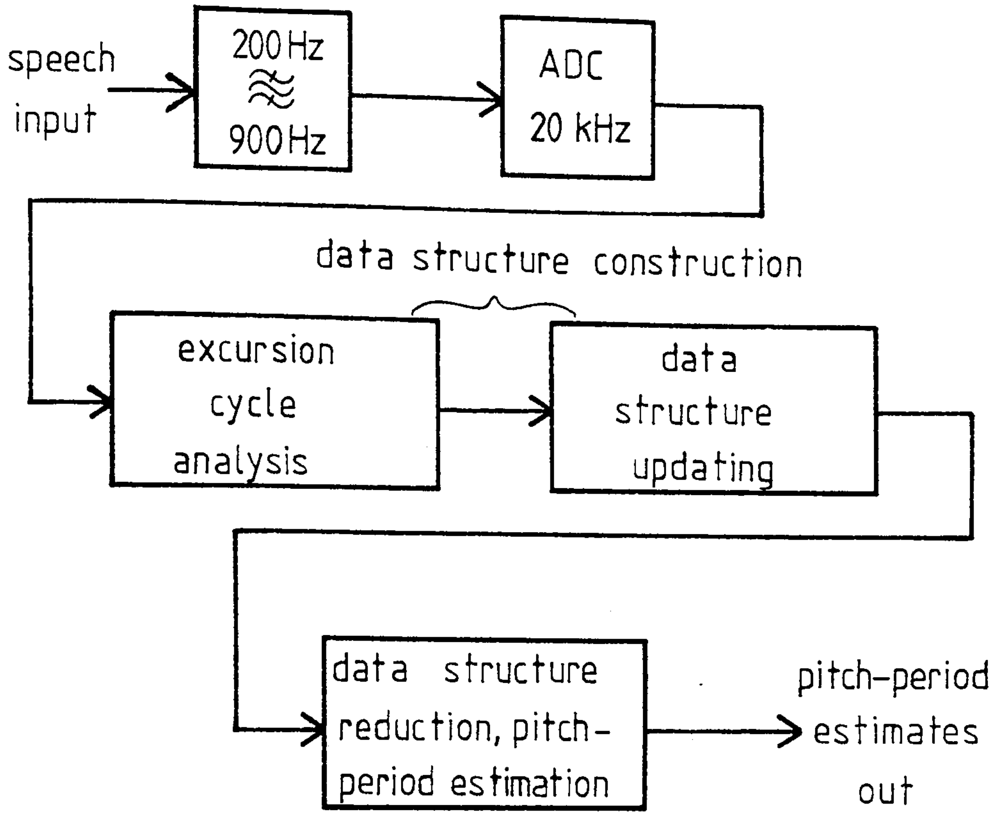


Figure 4-3: Structure of Miller's data reduction scheme

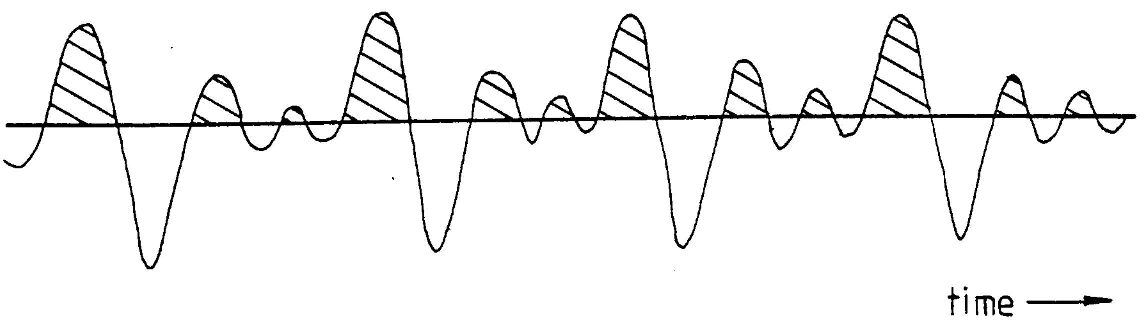


Figure 4-4: Section of voiced waveform showing positive excursion cycles

determined by shifts in polarity between adjacent samples, which indicate zero-crossings of the waveform. Insignificant excursion cycles are rejected by the application of procedures which compare the energy in a positive excursion cycle with a voiced-unvoiced threshold value, and compare the time interval between consecutive excursion cycles with a lower limit. Numerical details of the remaining positive excursion cycles, comprising the number of the initial non-zero sample, the maximum amplitude value of the excursion cycle, and the sample number of this maximum value, are entered into the excursion cycle data structure. More complex procedures are then applied to the entries in this data structure in order to isolate the significant excursion cycles, which are those occurring near to the beginning of each pitch-period, and to predict a series of pitch-period values for the speech signal. Once the waveform data have been reduced, the speech signal is represented by, at most, 500 entries per second of sampled speech in the data structure [M313,p74]. The processing required to construct this reduced data structure accounts for approximately half of the total computation time at a sampling rate of 20kHz [M313,p74]. Thus it can be seen that the data reduction strategy, while reducing the computational complexity of the main pitch-period detection algorithm, nevertheless constitutes a considerable portion of the overall processing load.

The first stage in most digital pitch-period detection schemes, following initial low-pass filtering, is analogue-to-digital conversion of the waveform. The sampling rate is chosen to provide an adequate representation of the speech signal, and rates of between 10kHz and 20kHz are normally selected. In Miller's data reduction system the sampling rate of 20kHz is necessary to preserve the structure of the speech waveform so that zero-crossing and energy measurements can be carried out numerically. Most ADC systems are interrupt-driven; that is to say, the converter periodically generates an interrupt signal to the main processor, which interrupts the main program flow and executes an interrupt service routine before returning to the point of interruption and resuming execution of the main program. In an interrupt-driven ADC system both the total conversion time of the ADC and the execution time of the interrupt service routine must be less than the interval between successive samples generated at the required sampling rate. At a sampling rate of 20kHz, this critical time is

$$1/20000 = 50\mu S$$

Fullagar et al [F430] quote the following fastest conversion speed ranges for two common types of 8-bit ADC:

integrating types: 0.3mS - 20mS

successive approximation types: 0.8µS - 30µS

Integrating types [M414,Section 16-13] have greater conversion accuracy, but are relatively slow; successive approximation types [M414,Section 16-13] have more error sources, resulting in lower conversion accuracy, but their speed of conversion is clearly sufficient to meet the 50µS conversion time requirement for a 20kHz sampling rate. A practical example of an analogue-to-digital conversion system is tha Analog Devices RTI-1230-8R [A431] [A432], which is based upon an 8-bit successive approximation ADC with a total conversion time of 8μS.

The main function of the interrupt service routine is to load a sample from the ADC and to store it in memory, but, in addition to this, the routine may be required to perform tasks such as zero correction or scaling of the sample value, comparison of the present sample with threshold values or with previous samples, and adjustment of system parameters or setting of flags based upon the value of the sample; in practice, interrupt service routines for microprocessors may require several hundred machine cycles, with total execution times as long as 1mS [F430].

4.4.3 Interrupt servicing on the 6502 microprocessor

Let us consider three interrupt service routines for a 6502 microprocessor, with a 1MHz clock input and 1μS machine cycle time. Assembly language listings of the three routines, together with instruction execution times, are given in Appendix One. All three routines are initiated by the non-maskable interrupt ($\overline{NMI}$) input and perform the minimum function of loading a sample from the ADC and storing it in the next available memory location.

With reference to interrupt service routine number one, it will be seen that the preprogrammed automatic response to the $\overline{NMI}$ input carries a processing overhead of 7µS. The first task of the interrupt service routine is to save the current values of all registers to be used during the routine; these are pushed on to the stack at the start, and pulled from the stack just before returning to the main program. It will be noticed that the interrupt service routine accesses the ADC sample as if it were a normal memory location with the label 'ADCIN'. This method of treating peripheral device registers as if they were memory locations is common in such minicomputers as the DEC PDP-11 [F430] and is supported by the 6800 and 6502 microprocessors. Interrupt service routine number one uses a two-byte pointer value (locations 'POINTER' and 'POINTER'+1) to store the address of the next available memory location. This is located in 'page zero' of the system's memory map - the bottom 256 bytes - which leads to slight reductions in the execution times of certain operations. The total execution time for interrupt service routine number one is 59µS.

Interrupt service routine number two employs a slightly different method of incrementing the pointer value, resulting in a total execution time of 51µS for most interrupts, and 56µS every 256 interrupts, when the more significant byte of the pointer is also incremented.

Interrupt service routine number three makes use of a pointer to the beginning of the current 256-byte memory 'page', and maintains an offset value from that base address. This gives a total time of 51μS for most interrupts, and 57μS every 256 interrupts.

These three examples illustrate the fact that even a simple interrupt service routine for the 6502 microprocessor requires more than the total execution time available between samples at a sampling rate of 20 kHz.

There exists a technique known as direct memory access (DMA), in which a device external to the main microprocessor system may transfer data directly into the system memory by 'stealing' one or more machine cycles and controlling the address and data buses during this time. It is possible for an ADC to place samples directly into system memory by making use of DMA techniques, and the 6502 microprocessor has provision for this operation [M421,Section 2.3.4.2] in the form of the 'RDY' input which, when taken to a low level, causes the processor to halt in the first non-write cycle it encounters, with the data bus in a high-impedance state. The sytem is then accessible to an external DMA controller. However, a major disadvantage of DMA is that it generally requires a larger amount of external circuitry [F430].

Assuming that DMA is not to be used, and bearing in mind the considerable processing overhead represented by the interrupt service routine, the only course of action open for the implementation of a scheme such as Miller's data reduction system on a 6502 microprocessor would seem to be to reduce the sampling rate, thereby increasing the interval between successive samples. Even at a reduced sampling rate of 10kHz, the process of loading samples from the ADC and storing them in successive memory locations will take up over half of the available processing time (100μS between samples).

In Miller's system, as stated above, the sampling rate is required to be fairly high in order to give sufficiently fine quantization of the input waveform in the time-domain to allow accurate zero-crossing measurement. This requires the processing of a relatively large number of input samples: rather the reverse of the intention of 'data reduction'. Niederjohn and Stick [N433] have examined this situation, in which it is required to extract accurate zero-crossing information from the input waveform without incurring a large processing overhead, and suggest the use of a special-purpose interface unit which detects zero-crossings in the input waveform, counts the intervals between successive zero-crossings, and makes this information available to the main processor via an interrupt-driven service routine. In this way, the processor is only interrupted when necessary, that is, at the end of each zero-crossing interval.

4.5 CHOICE OF BASIC SYSTEM STRUCTURE FOR THE PRESENT PROJECT

The present author felt that an external zero-crossing interface such as that described above might usefully be employed in a microprocessor-based implementation of a data-reduction pitch-period detection scheme, and, furthermore, that additional operations, such as integration of input waveform excursion cycles, might be implemented in an external hybrid analogue/digital hardware preprocessor. This would reduce considerably the number of interrupts generated per second of speech input, and would contribute to a significant reduction in the computational load on the microprocessor. It was decided to proceed with the design and construction of a speech analysis system on this basis; an outline illustration is given in Figure 4-5.

4.6 INITIAL HARDWARE DESIGN CONSIDERATIONS

4.6.1 Intensity calculation and display

One of the requirements of the proposed system is a real-time display of speech intensity. Several characteristics of a speech signal can be used as indicators of its magnitude, including the peak value, the average value, and the root mean square (rms) value of the signal. A display of rms value was chosen for the present project as it is an indicator of the energy content of the signal without regard to its waveform, being equal to the DC voltage causing the same degree of heating in a resistive element

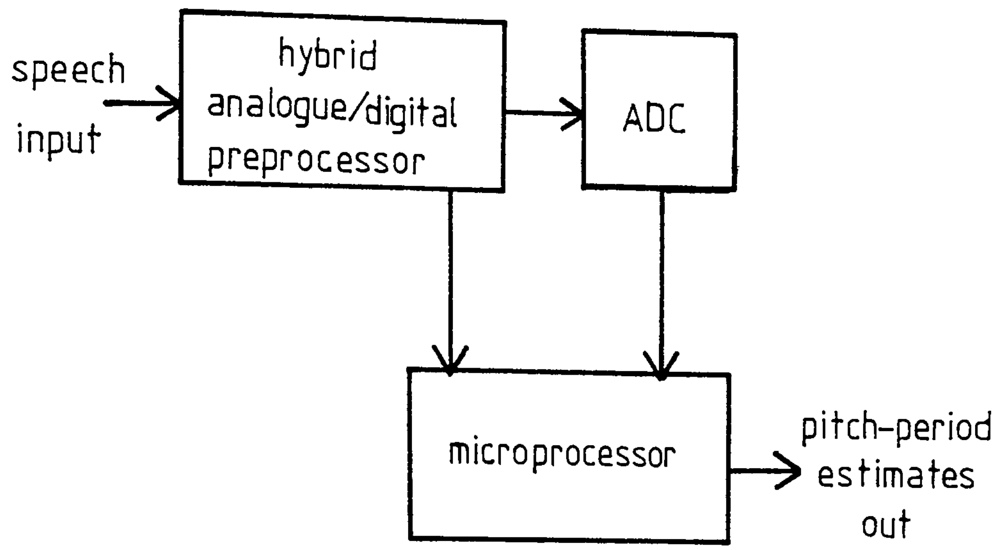


Figure 4-5: Outline illustration of proposed speech analysis system

as the signal itself [M434,p493].

Digital calculation of the rms value of input speech was considered, as was the construction of an analogue processor to perform this calculation [G416,Section 4.5]. However, several integrated rms-to-DC converters were readily available, and it was decided to employ a ready-built module.

4.6.2 Zero-crossing detection

It was decided to follow the basic idea described by Niederjohn and Stick [N433], involving analogue detection of zero-crossings combined with a digital measurement of zero-crossing interval by means of a counter driven by a constant-frequency clock signal. Their system is shown in outline form in Figure 4-6. Niederjohn and Stick point out that there exists an inherent trade-off between the frequency of the clock, the number of bits to be accumulated by the interface and the maximum length of zero-crossing interval to be measured. If T is the longest zero-crossing interval to be measured and n is the number of bits in the clock register, then the highest frequency to which the clock may be set is given by:

$$f_{clock} = (2^n - 1)/T \qquad \text{...(4.4)}$$

Bearing in mind that an 8-bit microprocessor was to be used as the main digital processor, it was most convenient to employ an 8-bit counter as the clock register. Assuming that the lowest fundamental frequency to be encountered is 50Hz, this represents

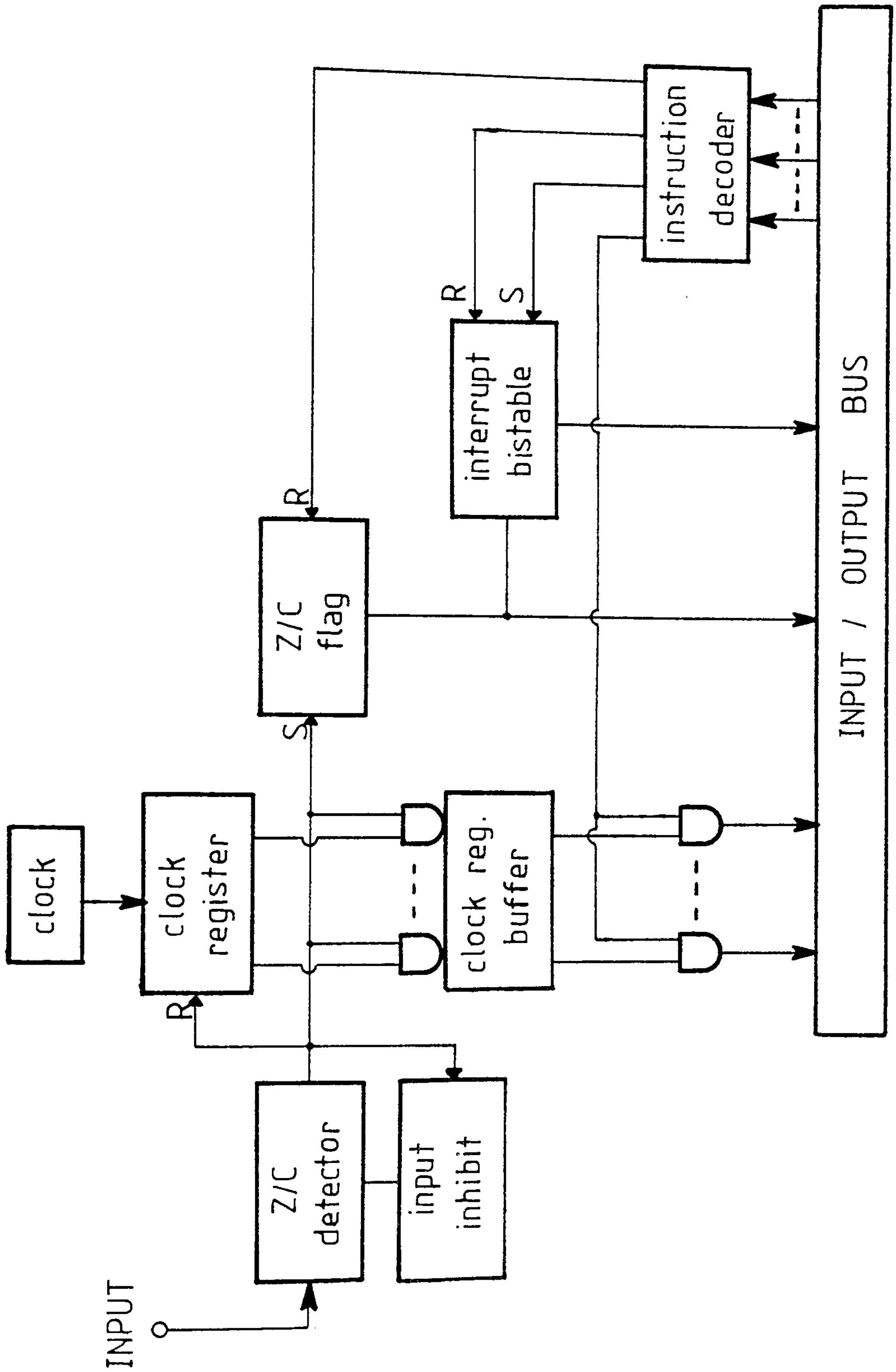


Figure 4-6: Zero-crossing interval interface (Niederjohn and Stick)

a maximum zero-crossing interval of 20mS. The maximum clock frequency, according to Equation 4.4, is 12750Hz. The resolution of the zero-crossing measurements - the time quantization level - is the reciprocal of the clock frequency used, giving an optimum resolution of approximately 80μS, which was considered to be more than adequate for the present project. In order to allow some latitude in zero-crossing interval measurement, it was decided to use a clock frequency of 10kHz, giving a maximum zero-crossing interval of 25.5mS and a resolution of 100μS.

It should be noted that the interval between zero-crossings is being measured, i.e. the position in time of a zero-crossing is relative to the occurrence of the previous zero-crossing, rather than being related to some absolute timescale, such as a real-time clock indicating the elapsed time since system initialisation. A consequence of this is that no meaningful measurement can be made of intervals longer than the anticipated maximum zero-crossing interval. The use of a 16-bit counter as the clock register would allow much finer time resolution, or, alternatively, a much longer maximum zero-crossing interval - over 6.5 seconds with a 10kHz clock frequency. The handling of 16-bit data was, however, felt to constitute too great a computational burden for the 8-bit microprocessor, as each access to a 16-bit value would require two 8-bit operations.

4.6.3 Excursion cycle integration

The integration of speech signal excursion cycles seemed ideally suited to analogue implementation, and a number of specialised basic designs were available, based upon operational amplifiers [G416,Section 3.1] [T415,Section 6.3]. These would produce a DC voltage equivalent to the integral of an excursion cycle in the input speech waveform. For processing by the digital processor some form of analogue to digital conversion would be necessary. It was envisaged that excursion cycle integrals would be used mainly in simple comparisons, and consequently a fine degree of quantization was not required. Again, it was decided that an 8-bit sample length would be most suitable for processing by an 8-bit microprocessor.

4.6.4 Voiced/Unvoiced threshold detection

Miller's data reduction system compares each excursion cycle integral estimate with a threshold value which exceeds the excursion cycle integral values of unvoiced sections of speech. In this way the voiced/unvoiced discrimination is performed within the speech analysis algorithm, and does not require a separate discriminator. Miller also points out that the principal excursion cycle in each pitch-period tends to have a greater integral value than other (insignificant) excursion cycles within the pitch-period. It was felt that the threshold comparison process which gives a voiced/unvoiced discrimination might also be employed to exclude insignificant excursion cycles from the voiced portions of the waveform. The threshold would therefore have to be based on some instantaneous estimate of the intensity of the speech signal, which is available in the form of the rms value of the signal. It was therefore decided that an analogue threshold comparison process should be included in the proposed system, in which the excursion cycle integral values are compared with a threshold value based upon the rms value of the signal. Initial experiments suggested that this process could give an effective discrimination between voiced and unvoiced portions of the waveform, and could also exclude insignificant excursion cycles from the waveform. The process is discussed in greater detail in Chapter 5.

4.6.5 Major differences from Miller's data reduction system

In Miller's system, the interval between excursion cycles is measured from the start of one excursion cycle to the start of the next excursion cycle of the same polarity (see Figure 4-7a). In the present system it was hoped to exclude some insignificant excursion cycles from the waveform before the data structure is constructed. It may be, for example, that the threshold comparison process outlined in Section 4.6.4 excludes excursion cycle β from the waveform in Figure 4-7a. In this case it will be necessary to compute the interval between excursion cycles α and γ. Since excursion cycle β cannot be excluded until its integral has been calculated, i.e. until the end of the excursion cycle, it will be necessary to save interval T1 in temporary storage, compute the integral of excursion cycle β,

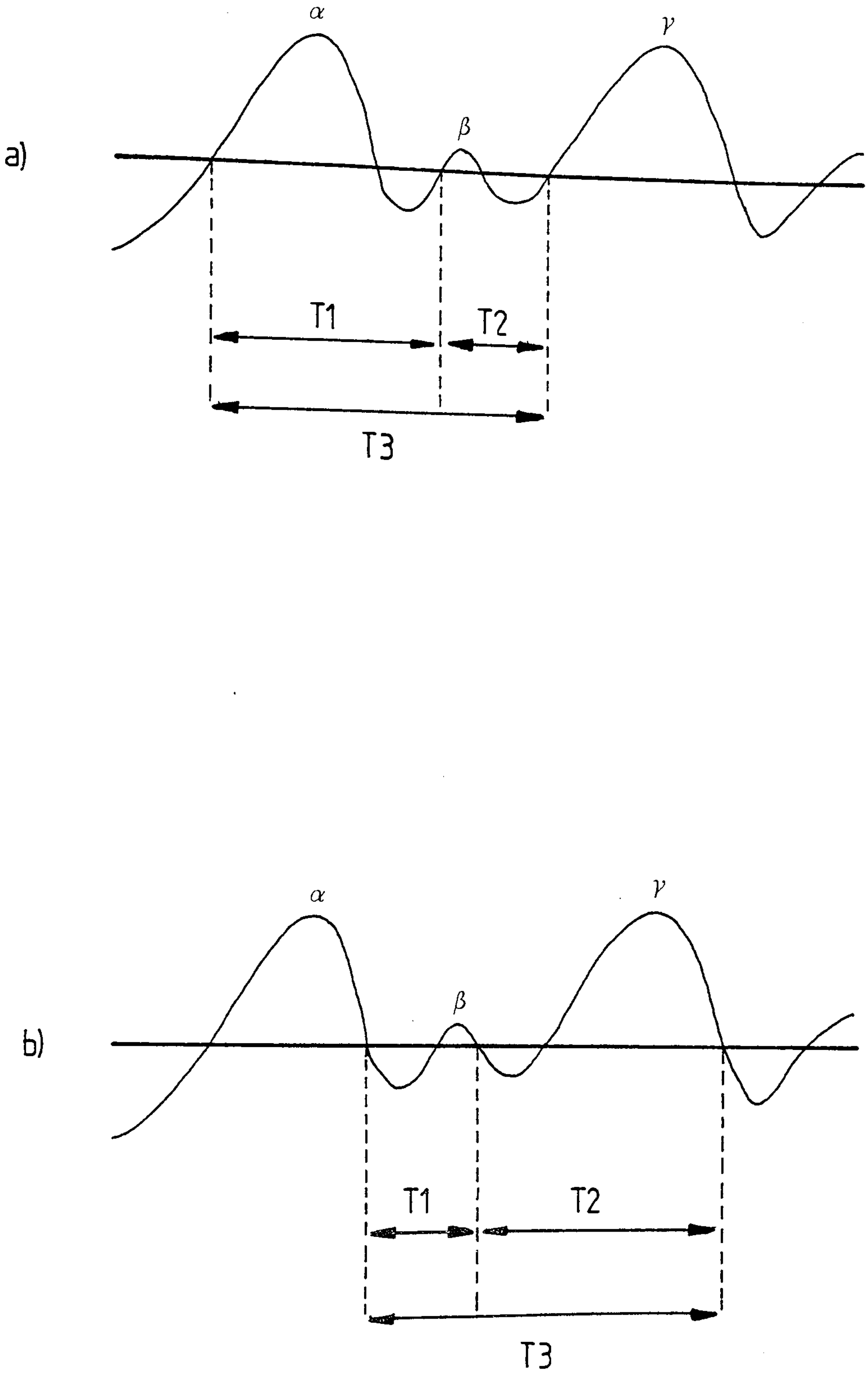


Figure 4-7: Excursion cycle interval measurements

compare this value with the threshold and then, on the basis of the result of this comparison, either pass time interval T1 across to the main processor or else, if excursion cycle β is to be excluded, indicate in some fashion that interval T1 is to be added to interval T2 to form the interval shown as T3. It may be, of course, that excursion cycle γ is also excluded from the waveform, in which case intervals T1 and T2 must be held in temporary storage until an excursion cycle that exceeds the threshold is encountered.

The situation is simplified considerably if zero-crossing intervals are measured from the end of one excursion cycle to the end of the next (see Figure 4-7b). In this case the counter need only be read and reset to zero if the integral of the excursion cycle that has just terminated exceeds the threshold value. Thus, in Figure 4-7b, if excursion cycle β contains sufficient energy to exceed the threshold, time interval T1 can be saved to pass across to the main processor, and the counter can then be reset to commence timing interval T2. If, however, excursion cycle β is excluded from the waveform by the threshold comparison test, the counter need not be reset, in which case it continues counting to measure time interval T3. Time interval measurement relative to the ends of excursion cycles simplifies considerably the design of the preprocessor, and reduces the number of samples that must be passed to the main processor. Measurements made by the present author on oscillograms of voiced speech waveforms suggest that the accuracy of pitch-period measurement is not significantly affected by this deviation.

In Miller's system, the data structure entry for each excursion cycle includes the maximum amplitude value for each excursion cycle and the position within the excursion cycle of that maximum value. This information is used by the phase of the algorithm responsible for excluding insignificant excursion cycles from the waveform. It was decided not to include these two values in the proposed speech analysis system for two main reasons. Firstly, while it would be perfectly feasible to include a simple analogue peak value detector in the preprocessor [T415,Section 9.5], the passing of the relevant information would require an additional ADC to convert the maximum amplitude value and some means of storing the counter value at the time of occurrence of the amplitude maximum. Secondly, the information is used in Miller's system to perform data structure reduction by exclusion of insignificant excursion cycles, and in the proposed system this reduction will hopefully be achieved by the use of the threshold comparison process.

4.6.6 Communication with the main processor

It was felt that the most appropriate means of communicating digital values from the preprocessor to the main processor would be via peripheral interface adapter (PIA) circuits. The Motorola 6821 PIA includes all the circuitry to allow an 8-bit microprocessor such as the 6502 to access two 8-bit words individually in a memory-mapped fashion.

4.6.7 Other hardware-related considerations

It was decided to follow Miller in taking into account only excursion cycles of one polarity, this being chosen to correspond to the more markedly peaked side of the waveform. As Miller points out, within a given utterance the major peaks tend to be mostly positive or mostly negative. Filip [F307] attributes this to the fact that phonation normally occurs exclusively during exhalation, so that for each pitch-period the initial major peak is always of the same polarity, and finds that unipolar processors normally perform satisfactorily in pitch-period detection tasks.

The question of phase distortion has to be considered in a pitch-period detection system. Miller suggests that the processing of phase distorted speech waveforms makes it impossible precisely to determine the exact start of each pitch-period; however, he finds that it is still possible to estimate the overall pitch-period value of the waveform when phase distortion is present. As the latter represents the requirements of the present system, it was felt that no particular attention need be paid to phase distortion within the preprocessor.

4.7 INITIAL SOFTWARE DESIGN CONSIDERATIONS

4.7.1 Data structure design

The entries in the data structure to be used in the proposed speech analysis system will comprise 'integral' and 'count' values from the integrator and the zero-crossing interval counter in the preprocessor. With a maximum permissible fundamental frequency of 500Hz - a design requirement of the proposed system and the value chosen by Miller for his system - there is a minimum time interval of 2mS between entries in the data structure, giving a maximum of 500 entries per second of speech, or a maximum data transfer rate of 1000 bytes per second, a reduction of an order of magnitude compared with a standard 10kHz sampling ADC system. Storage of up to 1000 bytes per second does not, as such, represent a problem, and it would be perfectly feasible to store samples in adjacent memory locations in system RAM. From the point of view of efficiency of the analysis algorithm, however, it was felt that an organised data structure should be designed which would allow efficient access by the main processor to the entries therein.

4.7.2 Pitch-period output estimates

The design requirement was for a system producing a display of fundamental frequency against time, and consequently some means of converting pitch-period estimates into the corresponding fundamental frequency values would have to be employed. A decision would also have to be made concerning the number of outputs to be generated per second of input speech. A device such as the Laryngograph/Voiscope [F107] produces one output sample per pitch-period, and is therefore capable of displaying variations between individual pitch-periods. It was felt that such detail in output was unnecessary for the purposes of the present project, and might be impractical within the confines of processing time available. It was, therefore, concluded that outputs should be produced at regular intervals, and that the exact number of outputs per second should be chosen with regard to ease of implementation, but should be large enough to provide an adequately detailed output contour. A survey of the speech analysis systems detailed in Chapter 3 suggested that between 50 and 100 output values per second would be adequate.

In Miller's data reduction system, data structure entries representing several hundreds of milliseconds of speech are examined with a view to locating 'syllabic boundaries' and 'syllabic nuclei', and data structure reduction and pitch-period estimation are based upon the characteristics of these major portions of the speech signal. This approach is impractical for the present project, as it would be necessary to delay by an unacceptably long interval in order to gather data over one or more syllables before producing a series of pitch-period estimates. It was felt that, for the proposed system, the data structure should be limited to contain entries relating to a certain fixed length of speech signal, and that pitch-period estimates should be generated regularly, based upon the contents of the data structure.

4.7.3 Error detection and correction

In a system such as the Laryngograph/Voiscope [F107] there is little need for error correction, as the process of pitch-period estimation is based upon an extremely simple signal providing a highly accurate analogue of glottal activity. In a system such as the proposed one, which analyses speech via a conventional microphone, it is inevitable that erroneous entries will appear in the data structure.

In Miller's data reduction technique, attempts are made to detect and correct pitch-period halving, pitch-period doubling, and the incorrect selection of principal excursion cycle within a pitch-period. Miller notes that if increased processing time were available it could be used to examine the pitch synchronous amplitude envelope provided by the maximum amplitude values of presumptive principal excursion cycles; any substantial lack of smoothness in this envelope suggests the presence of erroneous entries in the data structure.

Reddy [R312] describes an analysis algorithm which attempts to assign 'pitch markers' to a speech waveform in order to indicate the commencement of pitch-periods. He provides a detailed analysis of possible error sources, together with algorithms for detecting and, where possible, correcting these errors.

The three major error types are:

(i) <u>extra marker</u>: an unwanted pitch marker

(ii) <u>hop</u>: the marker changes peak from one pitch-period to the next

(iii) <u>hole</u>: the program neglects to insert a pitch marker where one should occur

Detection of these three error types is based upon an analysis of the 'relative error' between the 'indicated pitch' and the 'expected pitch'. These terms are defined as follows:

<u>Indicated Pitch</u> (IP): the period between the marker being tested and the previously accepted marker.

<u>Expected Pitch</u> (EP): the value of the adjacent pitch-period if the latter required no correction; otherwise, the 'most likely pitch' of the utterance, which is derived from a statistical analysis of the set of pitch-period values for the whole utterance.

<u>Relative Error</u> (RE) = (IP-EP)/EP

If the IP falls within the 'expected region', which is a small neighbourhood ($\pm$1/8 of EP) around the point where a pitch marker is expected to appear, then the pitch-period estimate is accepted; otherwise a corrective procedure is applied depending upon the value of RE. This operation is shown in flowchart form in Figure 4-8.

It was felt that the inclusion of an error detection/correction process similar to that described by Reddy would be helpful for the proposed speech analysis algorithm, and might compensate for inadequacies in the pitch-period estimation phase.

The maximum inter-entry count value of 25.5 mS, which is a consequence of the choice of counter clock frequency and word length, means that, as mentioned in Section 4.6.2, in cases where one or more principal excursion cycles are missing for some reason, it may be impossible to determine the precise interval between adjacent entries in the data structure. This had to be taken into consideration when designing error correction mechanisms for the proposed system.

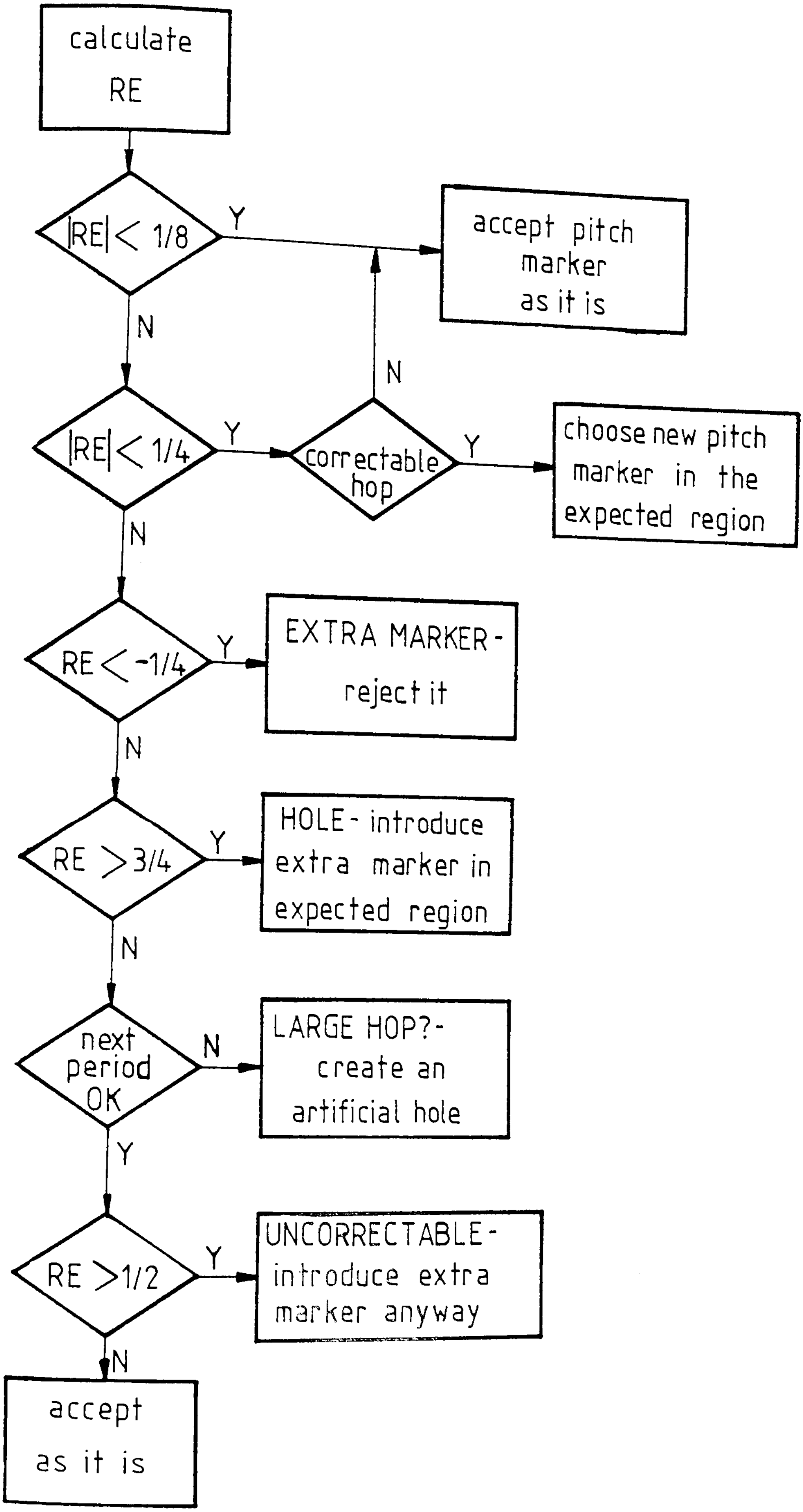


Figure 4-8: Reddy's pitch-period error correction scheme

4.8 INITIAL SYSTEM DESIGN EXPERIMENTS

4.8.1 Use of the storage oscilloscope as a pseudo-ADC

The Gould Advance OS4000 Digital Storage Oscilloscope, which was purchased as a display device for the proposed system, together with the OS4002 Data Output Option, can be used to output a stored waveform as a series of 8-bit digital samples. In this way it is possible to simulate, for a limited time, an analogue-to-digital converter, and, by appropriate adjustment of the oscilloscope timebase during data capture and data readout, to vary the effective sampling rate.

A simple buffer board was constructed, which, together with an interrupt service routine similar to those shown in Appendix 1, allowed the storage oscilloscope to be connected to the PET microcomputer via the latter's User Port, so that sections of speech waveform could be captured on the oscilloscope and transferred into RAM on the PET. The storage oscilloscope has a 1024-word store, so that, when the timebase is adjusted to give an effective sampling rate of approximately 10kHz, the 1024 samples represent approximately 100mS of speech waveform.

Once samples representing a section of speech waveform have been stored in the PET, they are accessible to high- and low-level programs running on the PET; in this way, it was possible to simulate some of the functions of the proposed speech analysis system, and of other systems, without the need for an actual ADC. The use of the storage oscilloscope allows one to search for an interesting section of speech waveform at random, and, once such a section is found, to transfer the samples into the PET, and subject them to as much analysis as required. At the same time, the data output option has the facility for hard copy output of the stored waveform on a strip recorder.

One of the first tasks completed by the present author was the development of a clipped autocorrelation routine based upon the system described by Dubnowski et al [D334], which accepts a 100mS section from the storage oscilloscope and provides seven pitch-period estimates for that section. The software is described in Appendix 2. This proved to be an accurate pitch-period detector, and was used as a yardstick with which to compare various trial algorithms.

4.8.2 Extraneous entry prediction using 'modified autocorrelation'

Figure 4-9a shows a section of voiced speech waveform in which each pitch-period contains, in addition to the principal positive excursion cycle, one other prominent positive excursion cycle. Other, minor, positive excursion cycles are ignored for the purposes of the present example. Figure 4-9b shows the result of positive excursion cycle integration by an analogue integrator. The integral value is shown by the highest level reached by the integrator output prior to reset at the end of the excursion

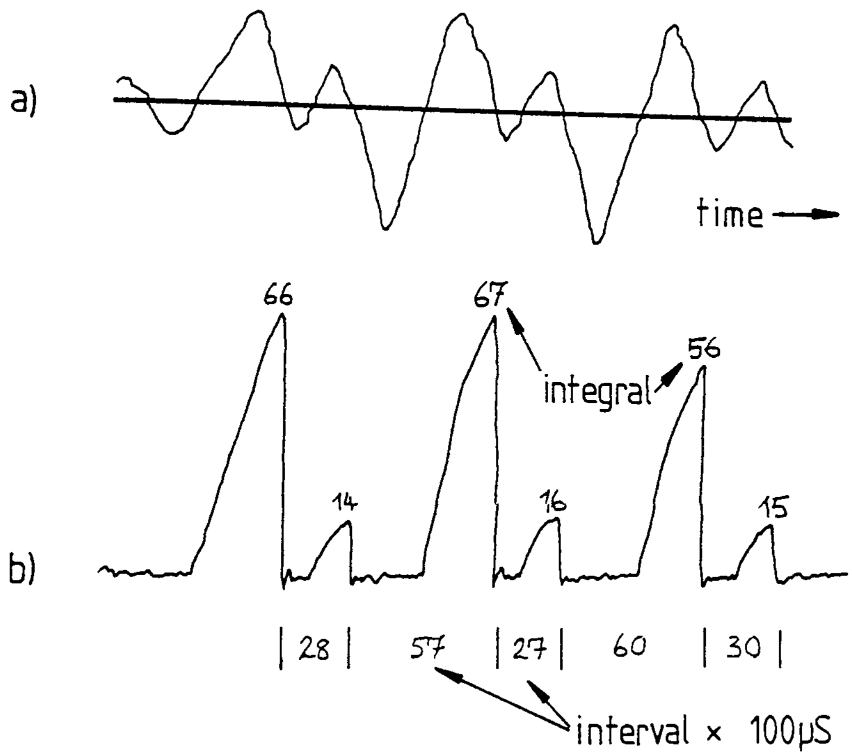


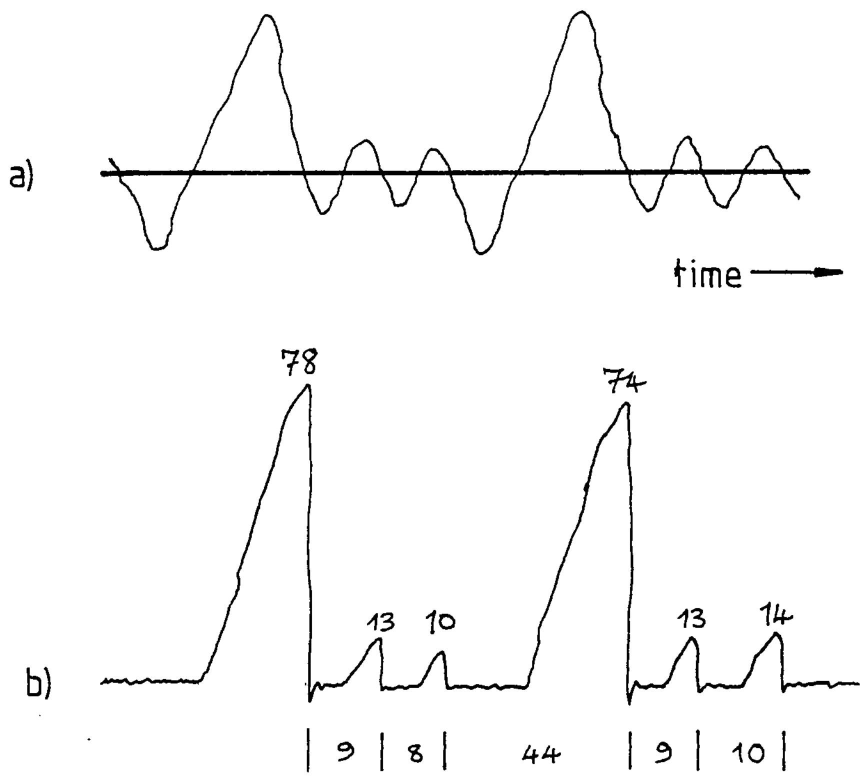
Figure 4-9: Waveform segment with one extraneous excursion cycle per pitch-period

cycle. Marked on Figure 4-9b are the integral values (on a scale from 0 to 100) and the intervals between excursion cycles (measured from the end of one positive excursion cycle to the end of the next, in units of 100μS).

It was noticed that, in waveforms such as that shown in Figure 4-9a, in which there is one prominent insignificant excursion cycle in each pitch-period, there is a pronounced regularity in the patterns of the excursion cycle integral values and/or the intervals between excursion cycles. Examination of the values marked on Figure 4-9b shows that both sets of values have a pattern of alternating high and low values.

Figures 4-10a and 4-10b show the results for a waveform in which there are two prominent insignificant excursion cycles in each pitch-period (as before, minor peaks have been ignored). Examination of the sets of values shows that there is again a regularity, in this case with a repetition at every third value.

It has been noted elsewhere that the process of autocorrelation is sensitive to repetitive patterns in a set of values, and it was felt that some form of modified autocorrelation function might be developed which would detect any regularities in the sets of excursion cycle integral and excursion cycle interval values, and which could be used as a basis for the prediction of the probability of extraneous excursion cycles being present in a section of voiced speech waveform.



<u>Figure 4-10</u>: Waveform segment with two extraneous excursion cycles per pitch-period

The first stage in the algorithm which was developed is to construct two lists for the section of waveform being examined: the first relating to excursion cycle integral values and the second relating to excursion cycle interval values. In both cases each value calculated from the waveform is compared with the preceding value, and a decision is made as to whether the second value is greater than ($\bigcirc\!\!\!\!\!>$), less than ($\bigcirc\!\!\!\!\!<$), or equal to ($\bigcirc\!\!\!\!\!=$) the first value. The decision is made on the following basis:

Consider two values, v1 and v2, where v1 immediately precedes v2. The result of comparison of v2 with v1 is:

$$v2 \ \bigcirc\!\!\!\!\!> \ v1 \text{ if } v2 > (v1 + (v1/8)),$$
$$v2 \ \bigcirc\!\!\!\!\!< \ v1 \text{ if } v2 < (v1 - (v1/8)),$$
$$v2 \ \bigcirc\!\!\!\!\!= \ v1 \text{ otherwise.}$$

...(4.5)

It will be seen that a small region around v1 ($\pm 1/8$) is defined, and that it is said that v2 $\bigcirc\!\!\!\!\!=$ v1 if v2 falls within that region. The lists constructed by this process, referred to as 'inter-entry comparison' lists, consist of a set of tri-state values ($\bigcirc\!\!\!\!\!<$, $\bigcirc\!\!\!\!\!>$, $\bigcirc\!\!\!\!\!=$). Lists for the two example considered so far are shown in Figure 4-11.

The next stage is the modified autocorrelation computation. This is defined as:

Results for Figure 4-9:

Integrals:

$<$	$>$	$<$	$>$	$<$	
$<$	$>$	$<$	$>$	$<$	$\hat{\phi}_x(0) = 5$
	$<$	$>$	$<$	$>$	$\hat{\phi}_x(1) = -4$
		$<$	$>$	$<$	$\hat{\phi}_x(2) = 3$
			$<$	$>$	$\hat{\phi}_x(3) = -2$

Integral prediction: 1 extraneous entry

Counts:

$>$	$<$	$>$	$<$	
$>$	$<$	$>$	$<$	$\hat{\phi}_x(0) = 4$
	$>$	$<$	$>$	$\hat{\phi}_x(1) = -3$
		$>$	$<$	$\hat{\phi}_x(2) = 2$
			$>$	$\hat{\phi}_x(3) = -1$

Count prediction: 1 extraneous entry
Combined prediction: 1 extraneous entry

Results for Figure 4-10:

Integrals:

$<$	$<$	$>$	$<$	$=$	
$<$	$<$	$>$	$<$	$=$	$\hat{\phi}_x(0) = 4$
	$<$	$<$	$>$	$<$	$\hat{\phi}_x(1) = -1$
		$<$	$<$	$>$	$\hat{\phi}_x(2) = 0$
			$<$	$<$	$\hat{\phi}_x(3) = 1$

Integral prediction: 2 extraneous entries

Counts:

$=$	$>$	$<$	$=$	
$=$	$>$	$<$	$=$	$\hat{\phi}_x(0) = 2$
	$=$	$>$	$<$	$\hat{\phi}_x(1) = 0$
		$=$	$>$	$\hat{\phi}_x(2) = 0$
			$=$	$\hat{\phi}_x(3) = 0$

Count prediction: possible extraneous entries
Combined prediction: 2 extraneous entries

Figure 4-11: Extraneous entry predictions for Figures 4-9 and 4-10

$$\hat{\emptyset}_x(m) = \sum_{n=1}^{N-m} x(n)x(n+m), \qquad m = 0,1,2,3$$

...(4.6)

where $x(n)$ represents one of the inter-entry comparison lists, $\hat{\emptyset}_x(m)$ is the modified autocorrelation value for a lag value of m, N is the number of entries in $x(n)$, and $x(n)x(n+m)$ takes one of the following values:

$$x(n)x(n+m) = 0 \text{ if } x(n) = (=)$$
$$\text{or if } x(n+m) = (=)$$

otherwise

$$x(n)x(n+m) = +1 \text{ if } x(n) = x(n+m)$$
$$= -1 \text{ if } x(n) \neq x(n+m)$$

...(4.7)

It should be stressed that, despite the superficial similarity between the above process and the clipped autocorrelation routine described in Appendix 2, the two are totally distinct, and serve very different purposes.

On the basis of the results of the modified autocorrelation computation, a prediction is made of the probability of 'extraneous entries' – i.e. insignificant excursion cycles – appearing in the data derived from the waveform, according to the following algorithm:

compute $\hat{\emptyset}_x(0)$;

IF $\hat{\emptyset}_x(0) < \text{lim1}$ {test one}

THEN predict no extraneous entries

ELSE

compute $\hat{\emptyset}_x(1)$, $\hat{\emptyset}_x(2)$, $\hat{\emptyset}_x(3)$;

IF $\hat{\emptyset}_x(1) >= \text{lim2}$ {test two}

THEN

IF $(\hat{\emptyset}_x(1) > \hat{\emptyset}_x(2))$

AND $(\hat{\emptyset}_x(1) > \hat{\emptyset}_x(3))$ {test three}

THEN predict no extraneous entries

ELSE predict possibility of extraneous entries

ENDIF

ELSE

IF $\hat{\emptyset}_x(2) = \hat{\emptyset}_x(3)$ {test four}

THEN predict possibility of extraneous entries

ELSE

$\hat{\emptyset}_x(p)$:= higher of $(\hat{\emptyset}_x(2), \hat{\emptyset}_x(3))$;

IF $\hat{\emptyset}_x(p) < \text{lim3}$ {test five}

THEN predict no extraneous entries

ELSE predict 'p-1' extraneous entries

per pitch-period

ENDIF

ENDIF

ENDIF

ENDIF

Limits for each of the threshold tests were derived by experiment, and optimal results were found to be achieved with the following values:

$$\text{lim1} = 2$$

$$\text{lim2} = 0$$

$$\text{lim3} = 1$$

The tests in the algorithm operate as follows:

TEST ONE ensures that at least two of the entries in the inter-entry comparison list are not $=$. If this condition is not met, the prediction is that no extraneous entries are present in the data.

TEST TWO examines $\hat{\emptyset}_x(1)$. It has been found that a negative value for $\hat{\emptyset}_x(1)$ indicates that there are probably extraneous entries present. A non-negative value for $\hat{\emptyset}_x(1)$ indicates that there are probably no extraneous entries, although sequences such as:

$$= \quad < \quad < \quad = \quad < \quad <$$

may give this result; such sequences are taken into account by TEST THREE.

The rest of the algorithm is concerned with comparing $\hat{\emptyset}_x(2)$ with $\hat{\emptyset}_x(3)$. If they are equal (TEST FOUR), then there is a possibility of extraneous entries, although it is not possible to predict whether there are one or two extraneous entries in each pitch-period. If $\hat{\emptyset}_x(2) \neq \hat{\emptyset}_x(3)$, then the higher of the two values is chosen. If this higher value (denoted $\hat{\emptyset}_x(p)$ in the algorithm) is less than 1 then there is no sound basis for an extraneous entry prediction; otherwise, the prediction is that there are 'p-1' extraneous entries per pitch-period ('p' has the value 2 or 3).

It was found that at least four entries had to be present in each of the inter-entry comparison lists for the results to be reliable. No attempt was made to detect the regular presence of more than two extraneous entries per pitch-period; this allows the modified autocorrelation computation to be limited to the calculation of 4 'lag' values (including the zero lag calculation).

The final step is to combine the results from the excursion cycle integral list and the excursion cycle interval list into a single prediction. The combination is achieved as follows:

- if both results are the same, then this is chosen as the prediction
- if one result is a firm prediction of extraneous entries and the other is 'no extraneous entries' or 'possible extraneous entries', then the firm prediction is selected
- if both results are firm predictions, but of different numbers of extraneous entries, then a prediction of 'possible extraneous entries' is adopted.

The prediction selected becomes the extraneous entry prediction for the portion of waveform under analysis, and takes one of four forms:

(i) no extraneous entries

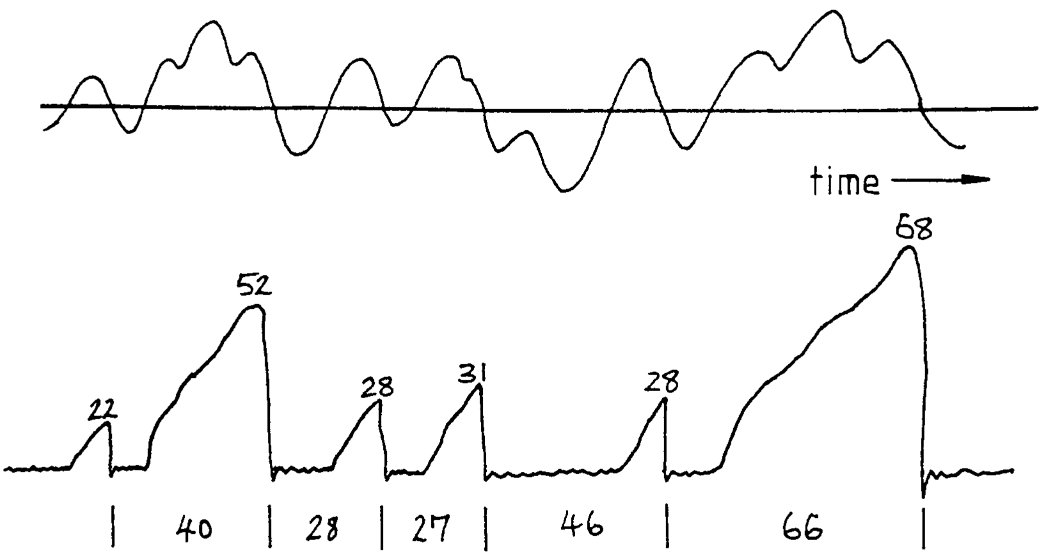
(ii) possible extraneous entries

(iii) ONE extraneous entry per pitch-period

(iv) TWO extraneous entries per pitch-period

The results for each of the two examples already considered in this section are given in Figure 4-11. Figure 4-12 shows the results in a case where there is no consistent pattern of extraneous entries.

The algorithm was developed using the storage oscilloscope as a sampled waveform source for the PET microcomputer (as described in Section 4.8.1), and was tested on a large number of waveform sections. Rather than identifying individual extraneous entries in the data relating to a section of waveform, it aims to give a prediction of the overall trend in the section, and it was felt that such a prediction would be of value in the pitch-period detection system to be developed. The use of the extraneous entry prediction process within that system is described in Chapter 6.



Integrals: $>$ $<$ $=$ $=$ $>$

$>$ $<$ $=$ $=$ $>$ $\hat{\phi}_x(0) = 3$

$>$ $<$ $=$ $=$ $\hat{\phi}_x(1) = -1$

$>$ $<$ $=$ $\hat{\phi}_x(2) = 0$

$>$ $<$ $\hat{\phi}_x(3) = -1$

Integral prediction: no extraneous entries

Counts: $<$ $=$ $>$ $>$

$<$ $=$ $>$ $>$ $\hat{\phi}_x(0) = 2$

$<$ $=$ $>$ $\hat{\phi}_x(1) = 0$

$<$ $=$ $\hat{\phi}_x(2) = 0$

$<$ $\hat{\phi}_x(3) = 0$

Count prediction: possible extraneous entries

Combined prediction: no extraneous entries

Figure 4-12: Extraneous entry prediction for a section of waveform with no marked regularity of extraneous entries

CHAPTER FIVE

DESCRIPTION OF THE PROJECT HARDWARE

5.1 PREPROCESSOR OPERATION

Figure 5-1 displays the overall structure of the preprocessor, split into basic modular units. The diagram shows the inputs to the preprocessor, the main outputs of the preprocessor - the rms analogue output, the excursion cycle integral ('INTEG') and the excursion cycle interval ('COUNT') - as well as the intermediate control signals: ZC (zero-crossing), NGT (not greater than threshold) and NLT (not less than threshold).

The main function of the preprocessor is to calculate the INTEG and COUNT values for all significant positive excursion cycles, where significance is indicated by the excursion cycle integral value exceeding a threshold level which is based upon the rms value of the signal. Figure 5-2 shows the two major points in the waveform which are of relevance to the preprocessor - a positive-going zero-crossing, point A, and the immediately following negative-going zero-crossing, point B. At point A in the waveform, which is detected by the zero-crossing detector, the integrator is started, and integrates the positive excursion cycle. When point B is reached, again detected by the zero-crossing detector, the integral value calculated for this excursion cycle is compared with a 'threshold' value, which is based upon the present output of the rms-to-dc converter. If the

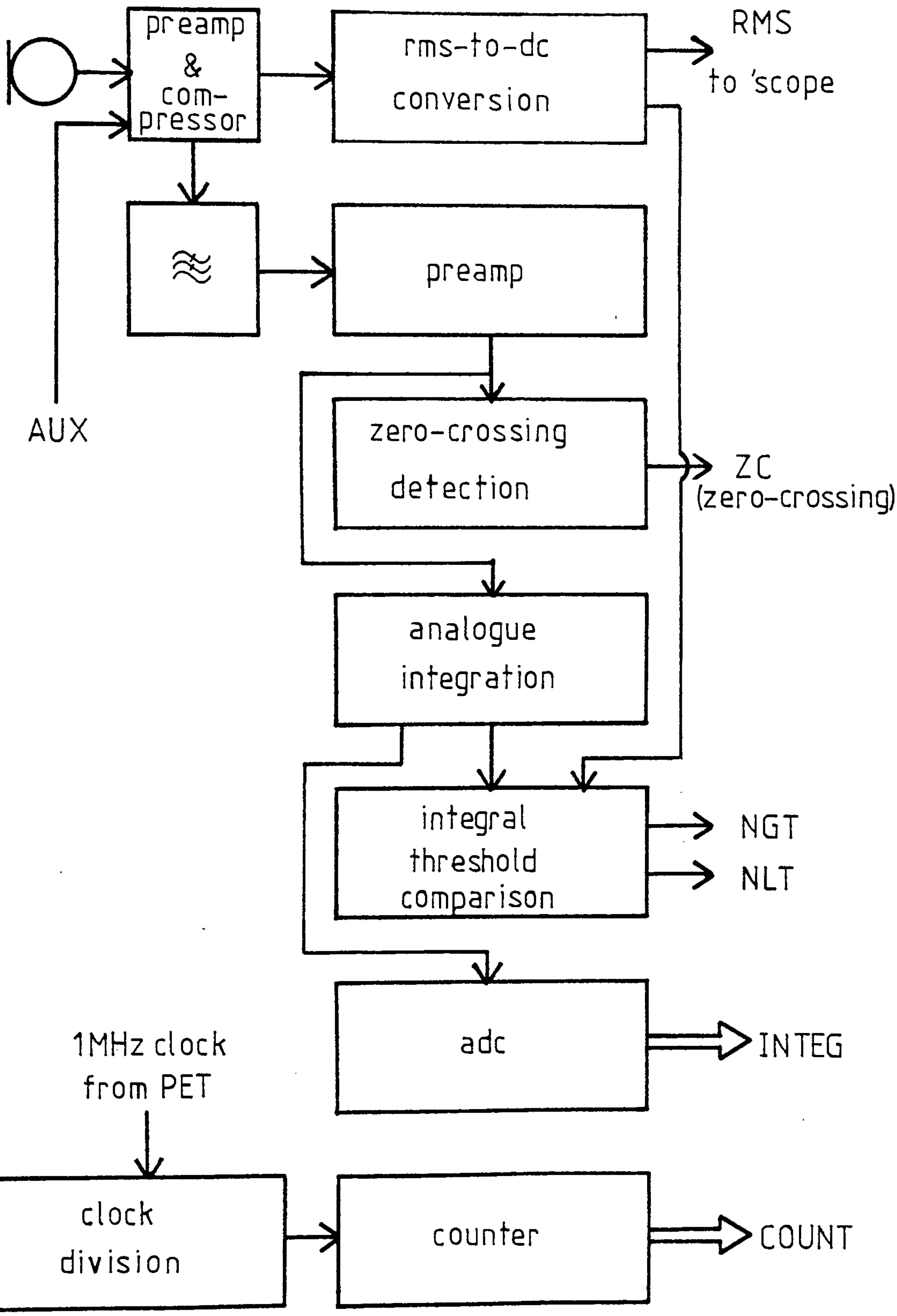


Figure 5-1: Block diagram of preprocessor

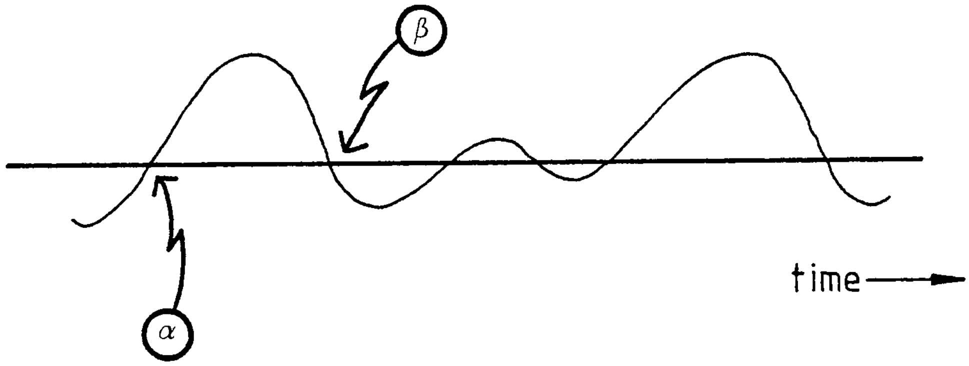


Figure 5-2: Positive- and negative-going zero-crossings

integral value is less than the threshold, then the excursion cycle is said to be insignificant; the integrator is reset, and no further action is taken until the next positive-going zero-crossing is detected. If the integral value exceeds the threshold value - a significant excursion cycle - then two main actions are taken. Firstly, the counter value, which represents the elapsed time since the end of the last significant excursion cycle, is latched, and the counter is reset to zero and recommences counting. Secondly, the integral value is converted to digital form by an ADC. These two digital values, the excursion cycle integral 'INTEG' and the excursion cycle interval 'COUNT', are presented to a peripheral interface adapter, ready for communication to the PET microcomputer. The integrator and ADC are then both reset, and await the next positive-going zero-crossing.

It will be seen from the foregoing description that all positive excursion cycles are integrated, but INTEG and COUNT values are only saved when the excursion cycle proves to be significant. Insignificant excursion cycles are therefore effectively ignored.

A compressor is included in the signal preamplification stage in order to limit the maximum amplitude value of the waveform; this ensures that the integrator is not overloaded. The low-pass filter limits high-frequency excursions of the waveform, and thus reduces the number of zero-crossings per second; in the present implementation, a 6th-order low-pass filter with a cutoff frequency of 1kHz is used, although reliable results were achieved with other cutoff frequencies. The rms-to-dc converter is fed with an unfiltered signal.

5.2 DESCRIPTION OF CIRCUITRY

The electronic circuitry was developed in the modular fashion indicated in Figure 5-1. The preprocessor contains a mixture of analogue and digital integrated circuits, most of the latter operating asynchronously. It is most convenient to describe the various modules individually.

5.2.1 Preamplifier

Figure 5-3 shows the preamplifier, which is designed to accept signals from a microphone or tape-recorder. The common-emitter transistor provides initial amplification of the low-level microphone signal; the operational amplifier acts as an inverting buffer stage.

5.2.2 Compressor

The compressor circuit illustrated in Figure 5-4 is based upon a recommended circuit for the RS306-803 electronic attenuator [R501]. This circuit is designed to compress all inputs in excess of a predetermined amplitude. For the present purposes, the preset potentiometers are set to give a compression threshold of 500mV peak-to-peak, with an overall circuit gain of 4. The maximum gain of the circuit is therefore limited to 2V

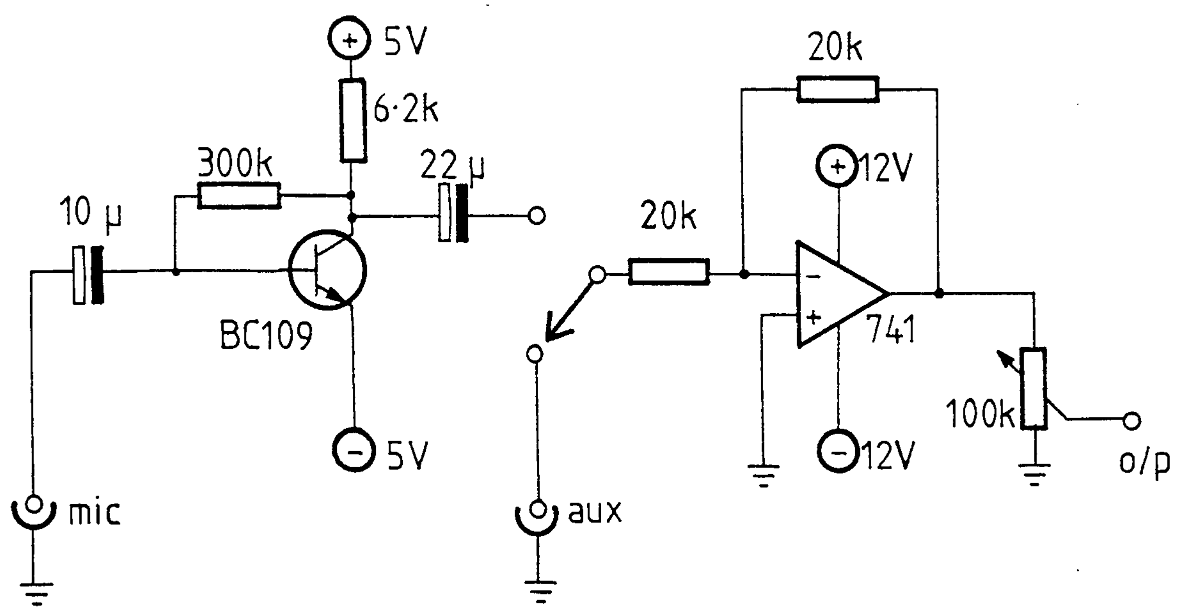


Figure 5-3: Preamplifier

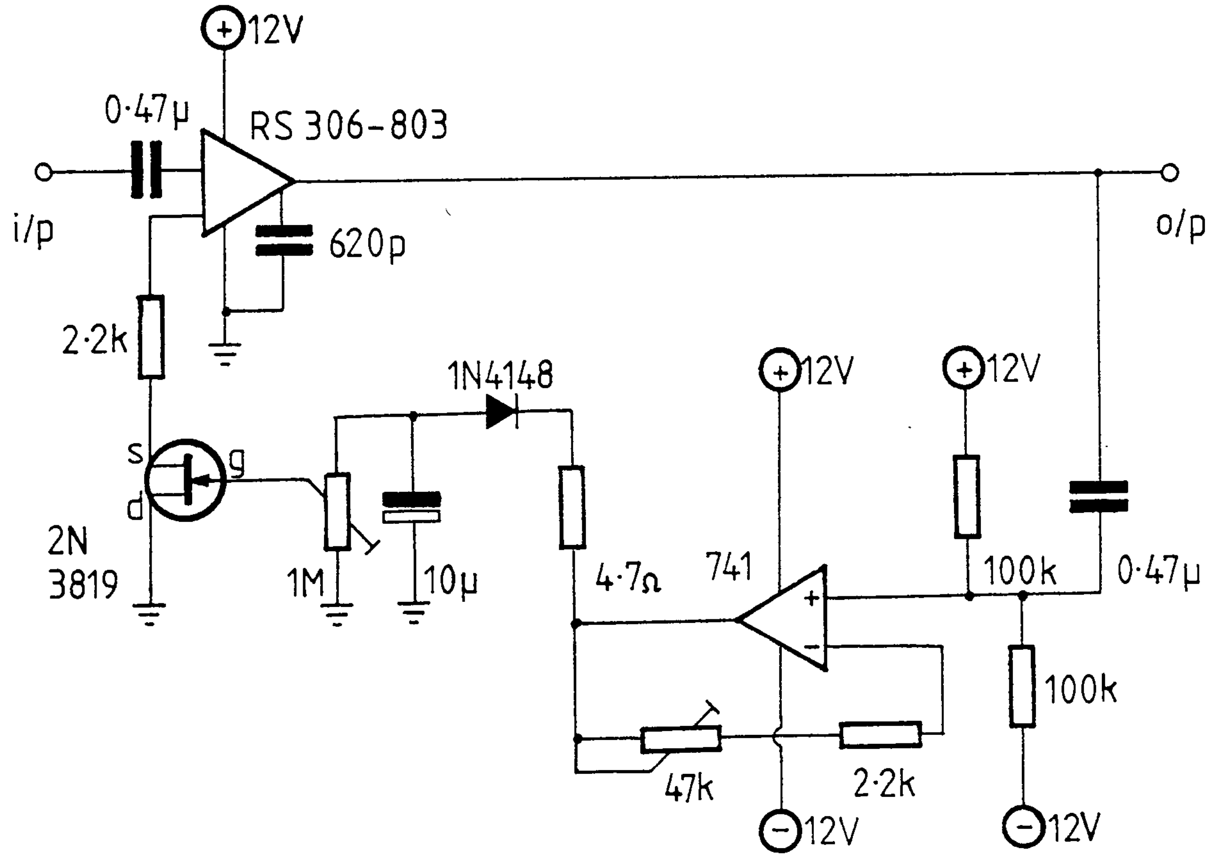


Figure 5-4: Compressor

peak-to-peak.

5.2.3 Rms-to-dc conversion

Figure 5-5 shows the rms-to-dc conversion circuit, which makes use of the Analog Devices AD536A true rms-to-dc converter [A502]. This integrated circuit contains all the circuitry necessary to convert an input waveform into a dc voltage corresponding to the rms value of the signal. The converter is preceded by an inverting stage with variable gain, and the output is split into two channels, each buffered by a non-inverting voltage follower.

Some experimentation was conducted in order to find the most suitable value for the rms-to-dc converter smoothing capacitor. There exists a tradeoff between eradication of ripple in the converter output and response time with respect to changes in the input signal level. It was required that the dc output level should not alter appreciably during one pitch-period, but that the dc output level should follow variations in rms level from one pitch-period to the next. After much trial-and-error testing, it was found that a value of 0.47μF gave a good compromise over a wide range of fundamental frequencies.

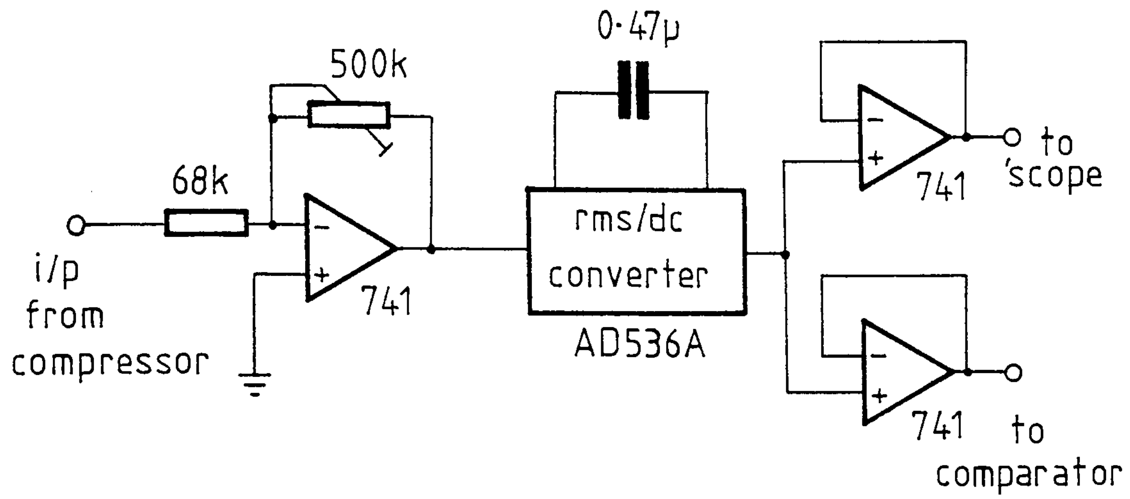


Figure 5-5: Rms-to-dc converter

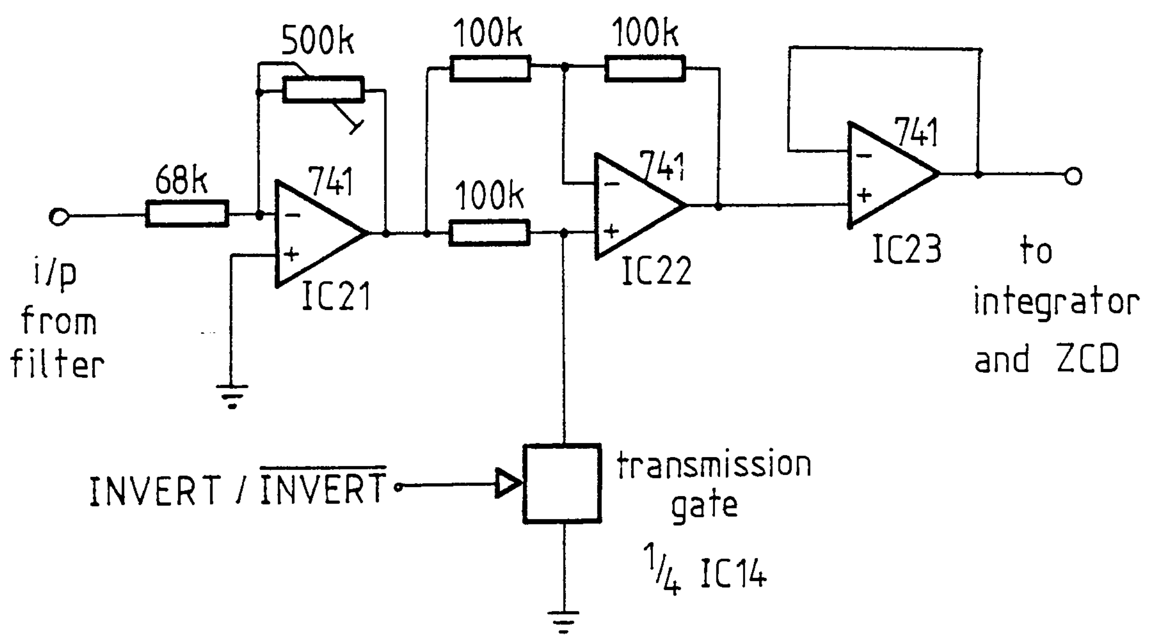


Figure 5-6: Amplifier/inverter stage

5.2.4 Amplifier/Inverter stage

This circuit, Figure 5-6, amplifies and conditionally inverts the output of the low-pass filter. IC21 provides variable gain for this channel. IC22 is configured as a digitally-controlled inverter/non-inverter. When the control input to CMOS transmission gate 1 is high, the gate shorts the non-inverting input of the operational amplifier to ground, and the circuit acts as an inverting amplifier with a gain of -1. When the control input is low, the transmission gate presents a very high resistance path to earth, and the circuit acts as a non-inverting voltage follower, with a gain of 1. The inverter is included in order to ensure that the more markedly peaked side of the waveform (ref. Section 4.6.7) is always the positive side, by inverting the signal if necessary. Once set for a particular microphone or tape-recorder, the INVERT/$\overline{\text{INVERT}}$ control does not have to be altered.

The digitally-controlled configuration was adopted so that the inverting/non-inverting selection could be controlled automatically. In the present design, however, the control input to the transmission gate is altered manually.

5.2.5 Zero-crossing detector

Figure 5-7a illustrates the zero-crossing detector, which is based upon the 710 voltage comparator. This has a TTL-compatible output, which is buffered and squared-up by two TTL NAND gates. The 60pF capacitor was found to be necessary to ensure stability of operation. The operation of the circuit is illustrated in Figure 5-7b, which shows input and output waveforms for a voiced section of speech signal.

5.2.6 Integrator

Figure 5-8a illustrates the analogue integrator circuit. The integrator itself is built around IC15 and IC16, which are metal-oxide semiconductor (MOS) operational amplifiers with very high input impedances. This reduces leakage from the integration capacitor, which is a high-quality polypropylene component. The output is inverted by IC17, which provides a positive INTEG output for the positive excursion cycles. The two transmission gates and their associated driving components are included to start, stop and reset the integrator. Integration starts when the ZC control signal from the zero-crossing detector goes low on a positive-going zero-crossing. This closes transmission gate 2, allowing the signal from the filtered speech channel to reach the input of the integrator. At the same time, transmission gate 3 is opened via the NAND gate bistable (gates 3 and 4), so that it represents a very high resistance in parallel with the integration capacitor. Integration continues until a

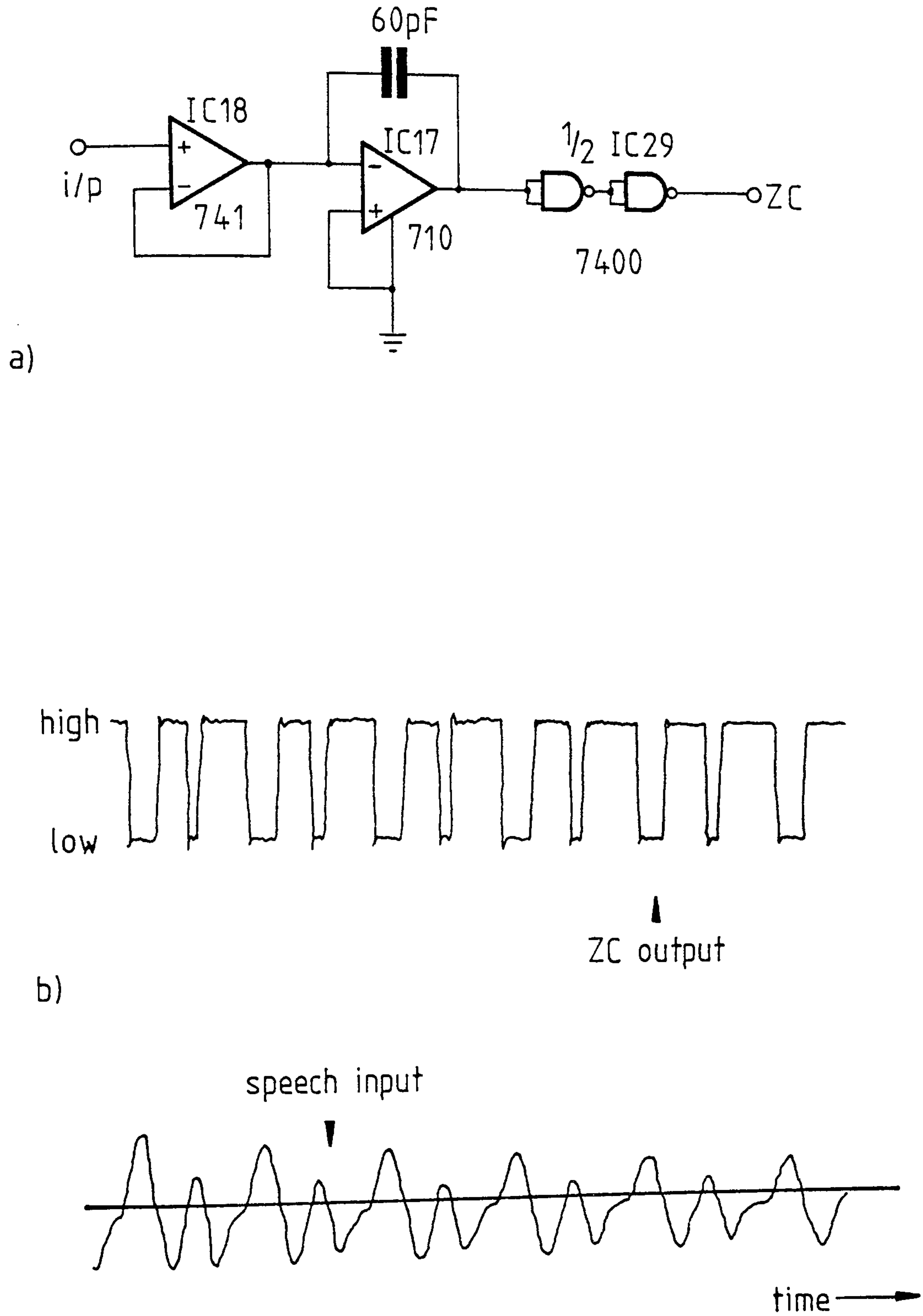


Figure 5-7: Zero-crossing detector

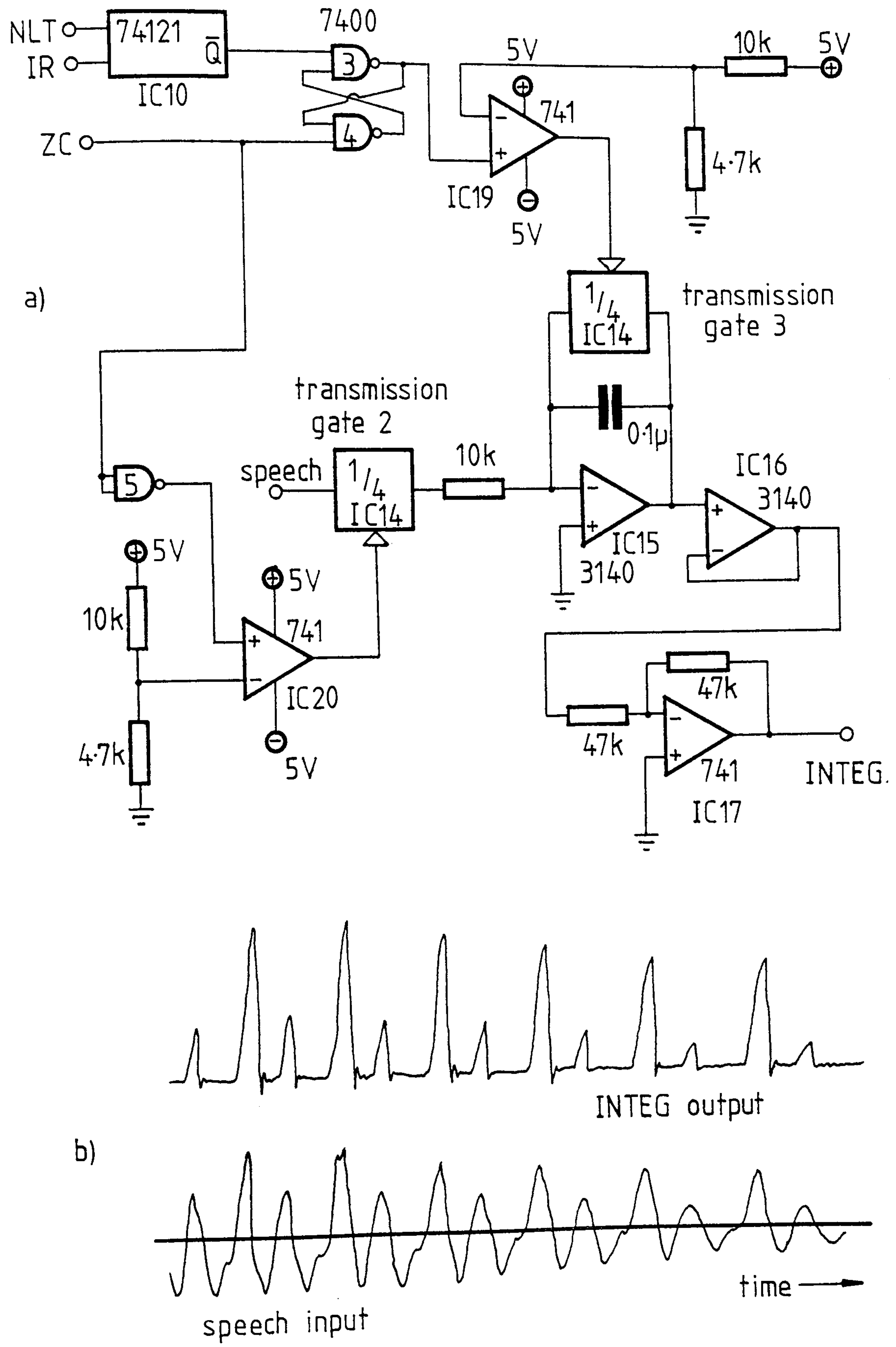


Figure 5-8: Integrator

negative-going zero-crossing is detected, whereupon ZC goes high, opening transmission gate 2 and thus isolating the integrator from the input signal. At this stage the INTEG output of the circuit represents the integral value of the positive excursion cycle that has just ended. This voltage is held until either the NLT control signal or the IR control signal goes low. The generation of these two control signals, which indicate respectively that the excursion cycle integral value is less than the threshold, or that data values have been passed to the microcomputer, is dealt with in Sections 5.2.7 and 5.2.8. When one of these two control signals makes a negative transition, monostable IC10 is triggered, setting the NAND gate bistable, and this causes transmission gate 3 to close, so that the integration capacitor is discharged through this low parallel resistance. The integrator is thus reset, ready for the next positive excursion cycle.

Operational amplifiers IC19 and IC20 were found to be necessary in order to drive the CMOS transmission gates from TTL signals.

Figure 5-8b shows speech input and INTEG output waveforms for a voiced portion of filtered speech signal.

5.2.7 Integral threshold comparison

This circuit, illustrated in Figure 5-9, compares the INTEG value with a threshold value derived from the rms analogue voltage, and, at the end of a positive excursion cycle, causes one of the two normally-high control signals, NLT and NGT, to go low. NLT going low indicates that the INTEG value is less than the threshold; NGT going low indicates that the INTEG value exceeds the threshold.

The threshold voltage is derived from the rms analogue voltage via potentiometer VR1. A dc offset voltage is added to this via potentiometer VR2. The resulting threshold voltage is compared with the INTEG voltage by IC6, a 710 comparator. The buffered COMP output goes low when INTEG exceeds the threshold; otherwise it is high.

Bistables IC3a and IC3b generate the NLT and NGT control signals. At the start of a positive excursion cycle, both bistables are cleared by the ZC control signal, so that NLT and NGT are both high. When ZC goes high at the end of a positive excursion cycle, a clock pulse is generated by monostables IC2 and IC8. This causes either NLT or NGT to go low, depending upon the J and K inputs of the relevant bistable. The J and K inputs of IC3a are both connected to the COMP control signal; the J and K inputs of IC3b are connected to an inverted version of COMP. If COMP is low when the clock pulse is generated, the J and K inputs of IC3b are high, causing NGT to go low to indicate that the

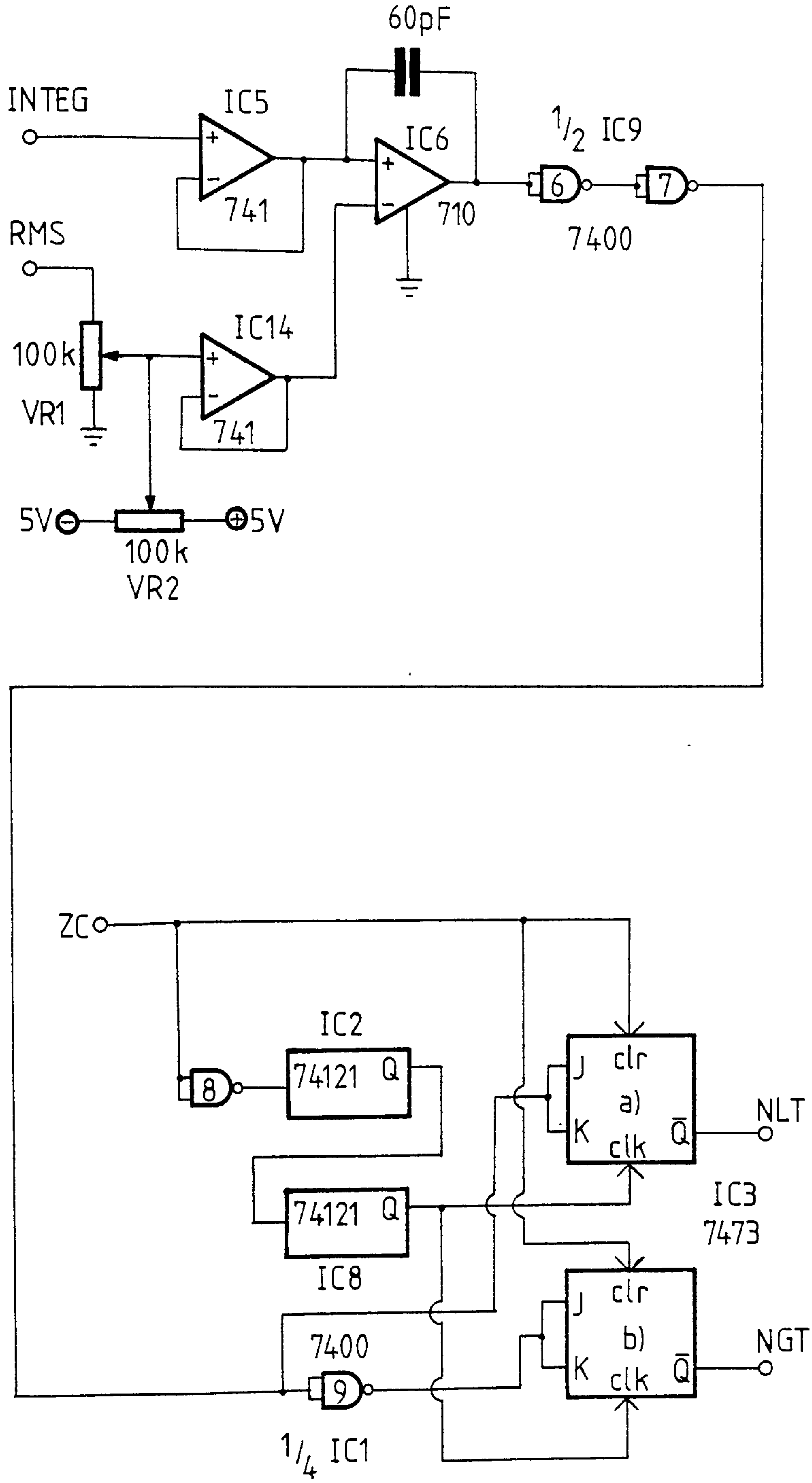


Figure 5-9: Integral threshold comparison

threshold has been exceeded. Otherwise, with COMP high when the clock pulse is generated, the J and K inputs of IC3a are high, causing NLT to go low. The circuit thus provides an indication of whether the immediately preceding positive excursion cycle is to be considered significant or not.

5.2.8 Analogue-to-digital converter

The analogue-to-digital converter (ADC) circuit illustrated in Figure 5-10 is designed to convert the INTEG voltage from the integrator to an 8-bit digital value. The circuit is based upon the ZN425E digital-to analogue converter (DAC) integrated circuit. This device is normally employed in a counter-ramp ADC configuration, which generates a digital value equivalent to the analogue input voltage. With a clock signal input of 256kHz, the maximum conversion time is 1mS [R503]. In a conventional configuration, conversion is initiated by a 'request conversion' control signal. The conversion process then takes place, and a 'conversion complete' signal is generated when a digital value has been generated [R503]. The basic operation of the counter-ramp type of ADC is described in Appendix 3.

For the present application, the basic counter-ramp ADC configuration has been extended, so that conversion takes place DURING the integration phase, rather than waiting for the integration to be completed before commencing conversion. The approach adopted improves overall conversion time by reducing the delay between the end of integration and the completion of

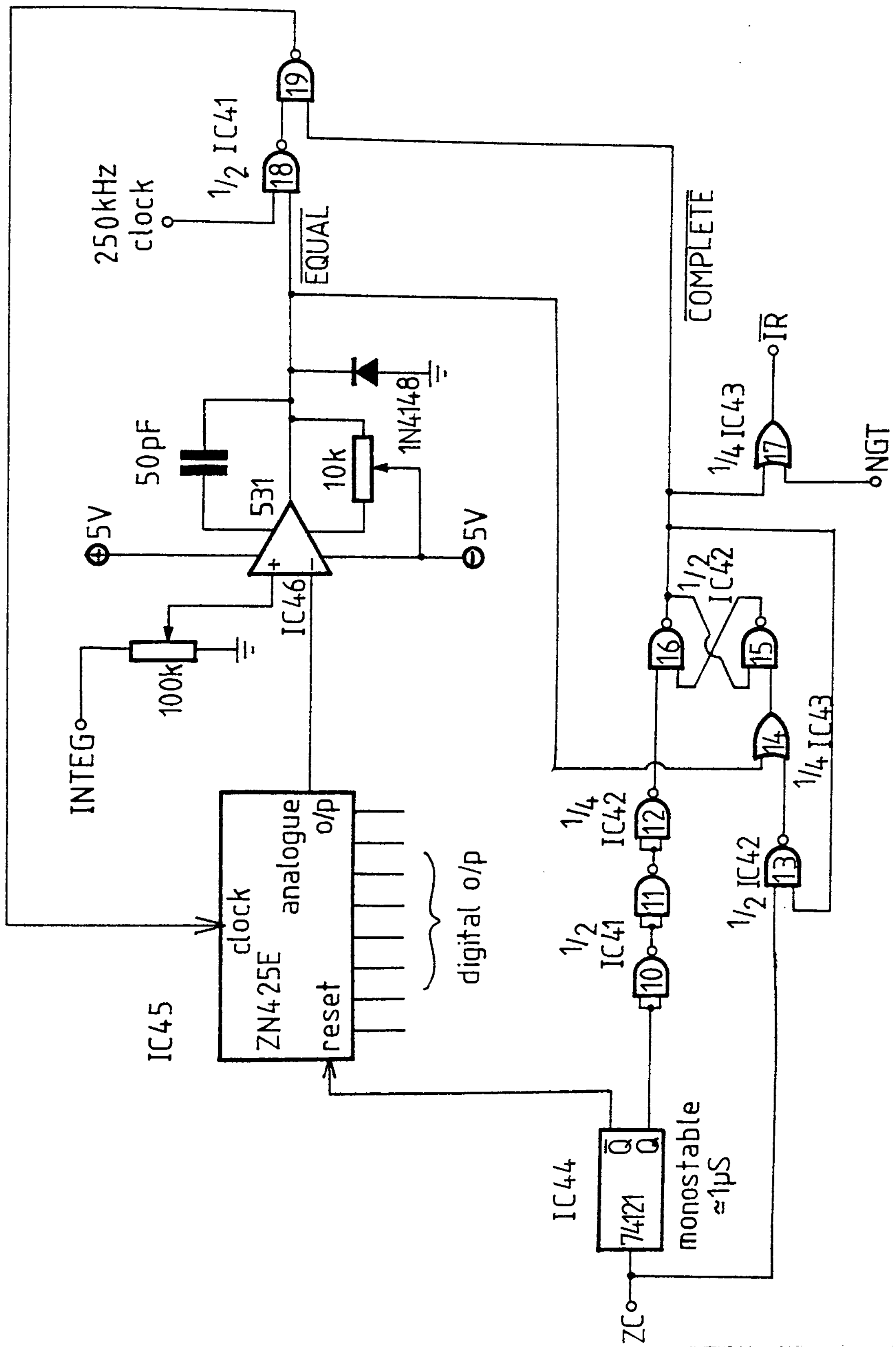


Figure 5-10: Analogue-to-digital converter

conversion.

Conversion commences when ZC goes low at the start of a positive excursion cycle. This triggers monostable IC44, which resets the ZN425E, and sets the bistable formed from NAND gates 15 and 16, so that $\overline{\text{COMPLETE}}$ goes high. Gates 10, 11 and 12 are included to introduce a time delay, in order to ensure that the ZN425E is reset before the bistable is set. The 531 operational amplifier is used as a comparator for the ADC circuit, and the output of this device, $\overline{\text{EQUAL}}$, goes low when the output of the ZN425E just exceeds the INTEG voltage. As long as $\overline{\text{EQUAL}}$ and $\overline{\text{COMPLETE}}$ are high, clock pulses from the 250kHz clock source are passed through gates 18 and 19 to the ZN425E.

In the conventional counter-ramp ADC circuit, clock pulses are permanently inhibited directly the DAC output exceeds the analogue input. In the present circuit, clock pulses are temporarily inhibited by gate 18 when $\overline{\text{EQUAL}}$ goes low; however, transmission of clock pulses continues when $\overline{\text{EQUAL}}$ goes high again. Clock pulses are only permanently inhibited by gate 19 when $\overline{\text{COMPLETE}}$ goes low; gates 13 and 14 ensure that this only happens when ZC is high, indicating that the positive excursion cycle has terminated, and $\overline{\text{EQUAL}}$ is low, indicating that the DAC output voltage has just exceeded the value of INTEG. Thus, the clock pulses are permanently inhibited only when the conversion is complete following the termination of the positive excursion cycle. Gate 17 ensures that $\overline{\text{IR}}$ goes low only if NGT is low when the analogue-to-digital conversion process is complete. $\overline{\text{IR}}$ is the active-low non-maskable interrupt signal ($\overline{NMI}$) to the microcomputer, and so an interrupt is only generated if the excursion cycle integral exceeds the threshold level. If this is not so (in which case NGT remains high), no interrupt occurs.

The ADC circuit remains in a non-active state until the next positive-going zero-crossing, whereupon the conversion cycle is repeated.

The overall design of the preprocessor has made it possible to employ a cheap and simple ADC circuit, since the time between conversions is relatively long. However, the specific circuit developed for the ADC results in efficient operation, and in practice it is found that conversion is complete within, at most, a few hundreds of microseconds of the end of a positive excursion cycle.

5.2.9 Interval counter

The circuit illustrated in Figure 5-11 is basically an 8-bit counter driven by a 10kHz clock signal. When the NGT control signal goes low, indicating that a significant excursion cycle has just terminated, the current counter reading, 'COUNT', is latched into IC27 and IC28, two 4-bit latches. The counter is then reset via monostable IC31, and recommences a count up from zero. The interval count remains latched until the microcomputer is ready to accept the value.

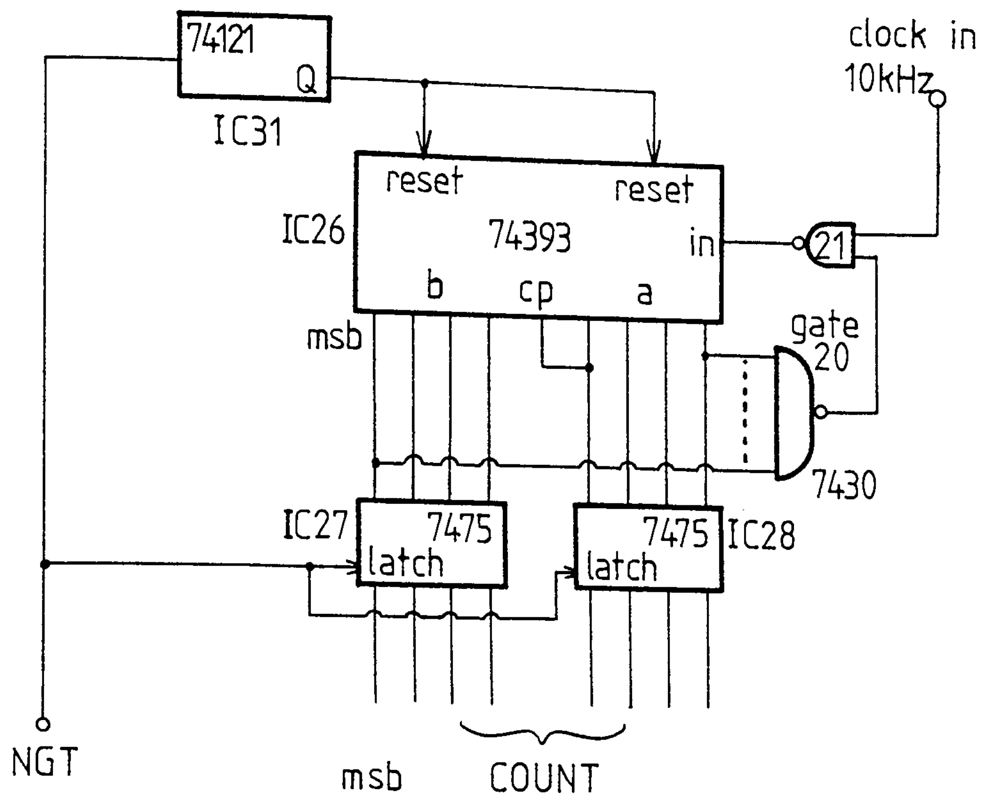


Figure 5-11: Interval counter

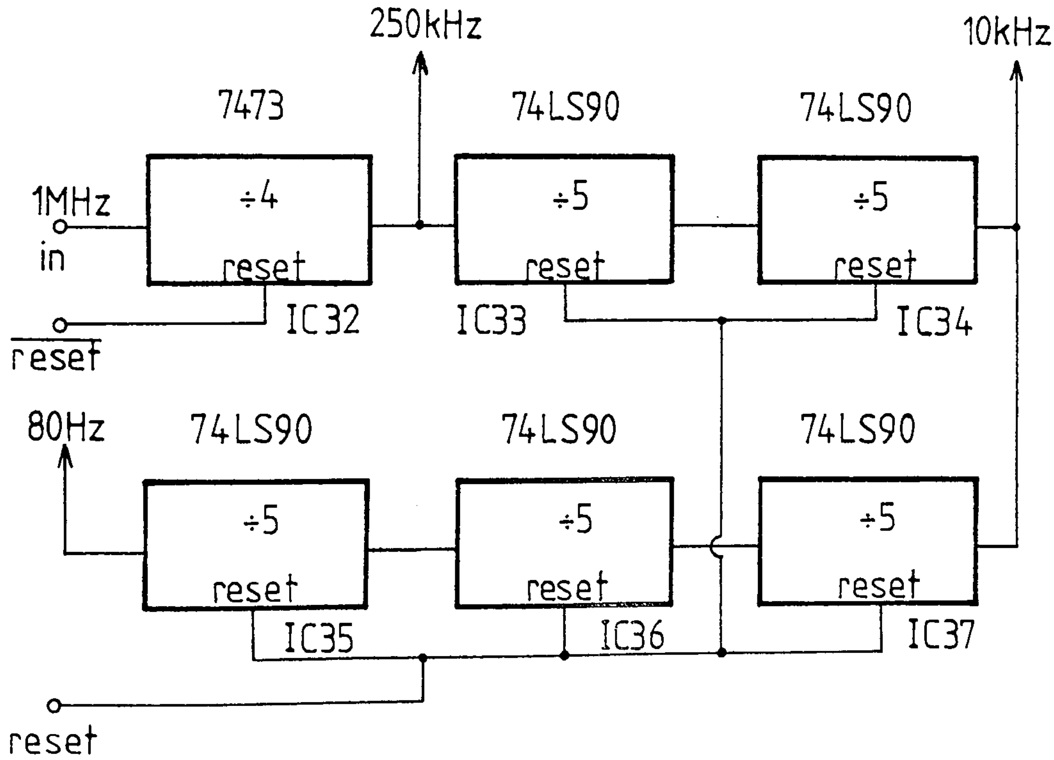


Figure 5-12: Clock division

If the count reaches a binary value 11111111, i.e. all bits set, the output of gate 20 goes low, inhibiting transmision of further clock pulses via gate 21. In this way, if the counter reaches its maximum count, there is no possibility of the counter 'wrapping around' to start counting up from zero again. The 10kHz clock gives a time quantization level of 100μS, and the 8-bit counter has a maximum count equivalent to 25.5mS.

5.2.10 Clock division

All timing signals in the preprocessor are derived from the 1MHz clock signal from the PET microcomputer. This is fed to a chain of dividers, shown in Figure 5-12, which provides a 250kHz clock for the ADC circuit, a 10kHz clock for the interval counter, and an 80Hz clock which is used to generate 80 output estimates per second via the IRQ interrupt input of the microcomputer.

5.2.11 Main preprocessor/PET interface

The main interface between the preprocessor and the PET microcomputer is a 6821 peripheral interface adapter (PIA) device, IC47, which communicates two 8-bit bytes, namely the COUNT and INTEG values (see Figure 5-13). The PIA chip select inputs are so arranged that the device appears at memory locations \$A810 - \$A813 in the PET memory map (the dollar sign indicates hexadecimal numbers). The PIA is selected using address lines A0, A1, A4 and A11, together with the SELA output from the PET. Additional PET signals used by the preprocessor

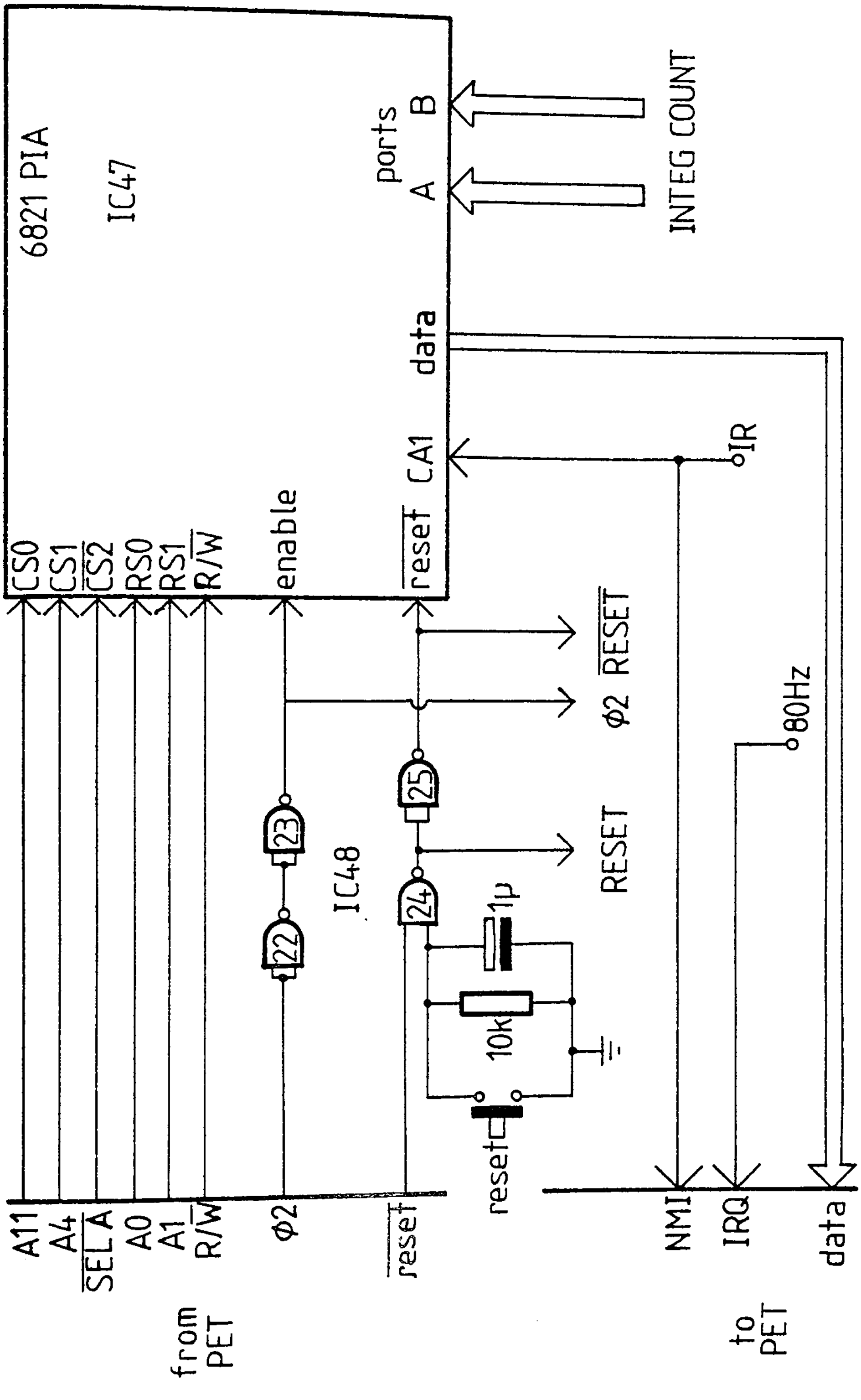


Figure 5-13: Main preprocessor/PET interface

are the 1MHz Ø2 clock, which is buffered by gates 22 and 23, and the RESET line, which is combined with a manual reset switch using gate 24, and is inverted by gate 25 to form an additional $\overline{\text{RESET}}$ line.

There are three main communication lines to the PET. The 8-bit data bus carries INTEG and COUNT values under control of the microcomputer. The $\overline{\text{NMI}}$ input to the PET is used to indicate that values are in the PIA ready for communication, and the $\overline{\text{IRQ}}$ input to the PET is driven by an 80Hz clock signal; this interrupt signal requests a new output sample from the PET.

5.2.12 PET output DAC

Although not part of the preprocessor circuit itself, the PET output DAC is contained on the same circuit boards and shares some of the control signals with the main preprocessor/PET interface.

The PET output DAC, illustrated in Figure 5-14, uses a ZN425E integrated circuit DAC, IC39, to generate an analogue voltage equivalent to a digital value from the PET microcomputer. This is buffered by IC39, which includes preset offset and gain adjustments [R503], and then fed to the storage oscilloscope. Digital output values are communicated from the PET via a second PIA, IC40. This uses address line A5 instead of A4, so that it occupies addresses $A820 - $A823 in the PET memory map. Only one 8-bit port is used on the PIA.

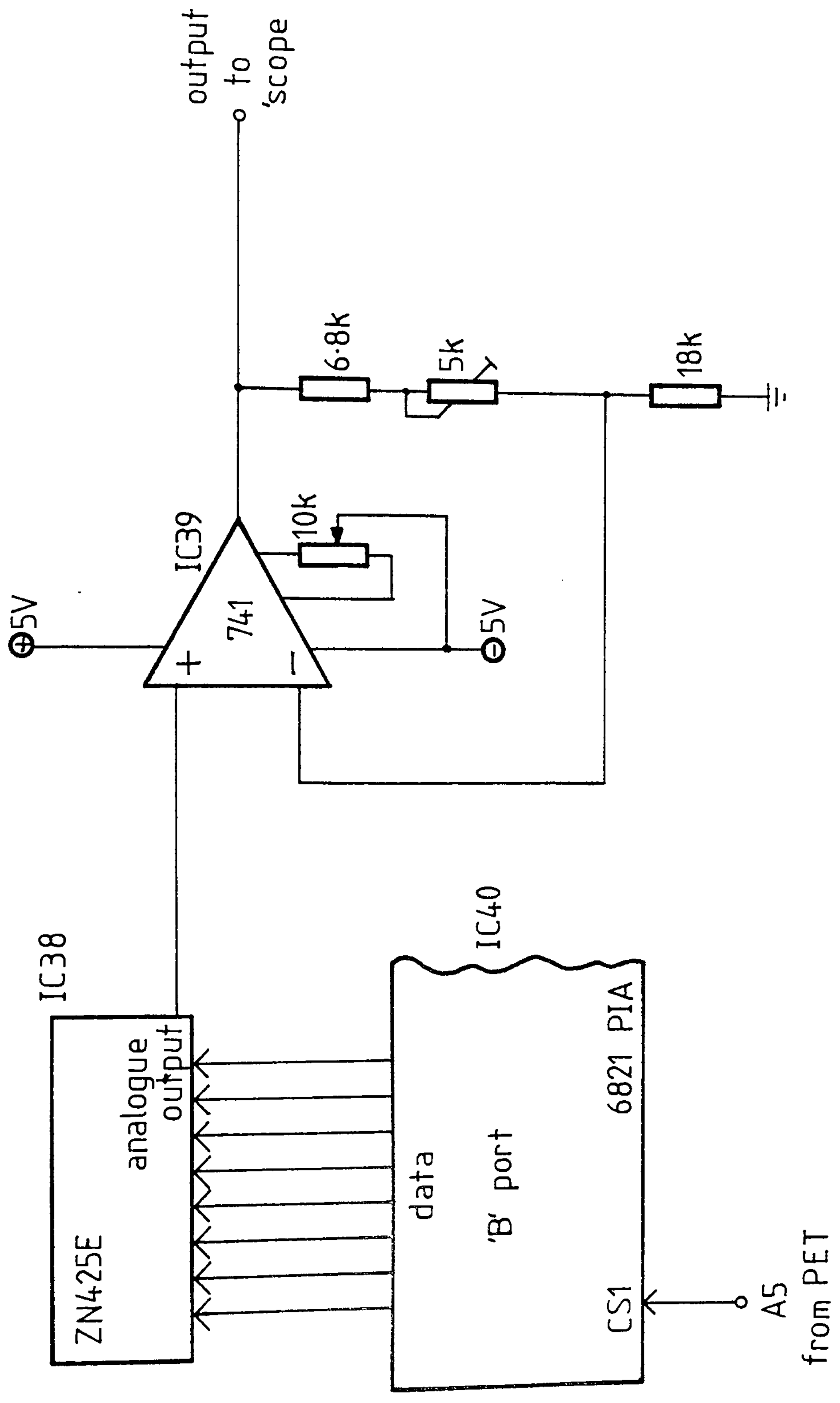


Figure 5-14: PET output DAC

5.3 CONSTRUCTIONAL DETAILS OF THE PREPROCESSOR

The microphone preamplifier and compressor circuits are contained in a small control unit. The rest of the circuitry is constructed on three wire-wrapped circuit boards, which fit into a common motherboard. The low-pass filter was not constructed specifically for this project, and is contained in a separate unit.

Some of the PET control signals are taken from the memory expansion connector on the PET 2001-32N. The data lines, however, are taken directly from the 6502 microprocessor, in order to bypass internal buffering circuitry in the PET.

5.4 THE THRESHOLD COMPARISON PROCESS

The use of the excursion cycle integral threshold comparison within the preprocessor is intended to serve two main purposes. Firstly, the process causes insignificant excursion cycles to be ignored. Secondly, the use of threshold comparison makes a separate voiced-unvoiced-silence discriminator unnecessary.

An example of the first role of the threshold comparison process is given in Figure 5-15, which shows a portion of voiced waveform and the corresponding transitions of the NGT (not greater than threshold) control signal. It will be seen that NGT only goes low during the major positive excursion cycles, with the

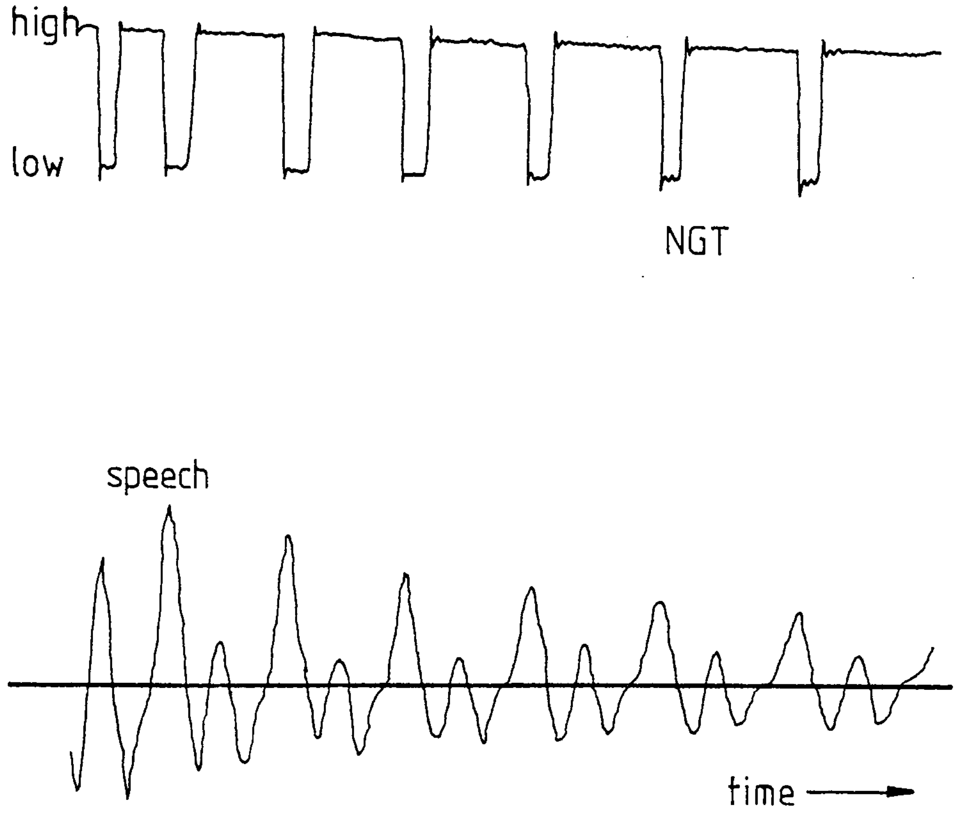


Figure 5-15: Elimination of insignificant excursion cycles

insignificant excursion cycles being ignored.

The voiced-unvoiced-silence discrimination role of the threshold detection process is illustrated in Figure 5-16. This shows (a) a portion of waveform at the start of the word "see", spoken by an adult male, (b) the output of the integrator, with, superimposed upon it, the rms-based threshold level, and (c) the NGT signal. It will be seen that the threshold voltage exceeds the peak integral levels during silent and unvoiced intervals, and is only exceeded by the integrator output during significant positive excursion cycles. The positive offset voltage, V_{offset}, shown in Figure 5-16b, is varied using potentiometer VR2 in the integral threshold comparison circuit (Figure 5-9). This control interacts with the threshold level potentiometer (VR1 in Figure 5-9), and they are both set so that the offset during silent and unvoiced intervals is sufficient to exceed the peak integral values, while the threshold is only exceeded during voiced intervals by significant excursion cycle integrals. Once a suitable combination of settings for VR1 and VR2 is arrived at, it is found that this remains satisfactory over a range of speakers.

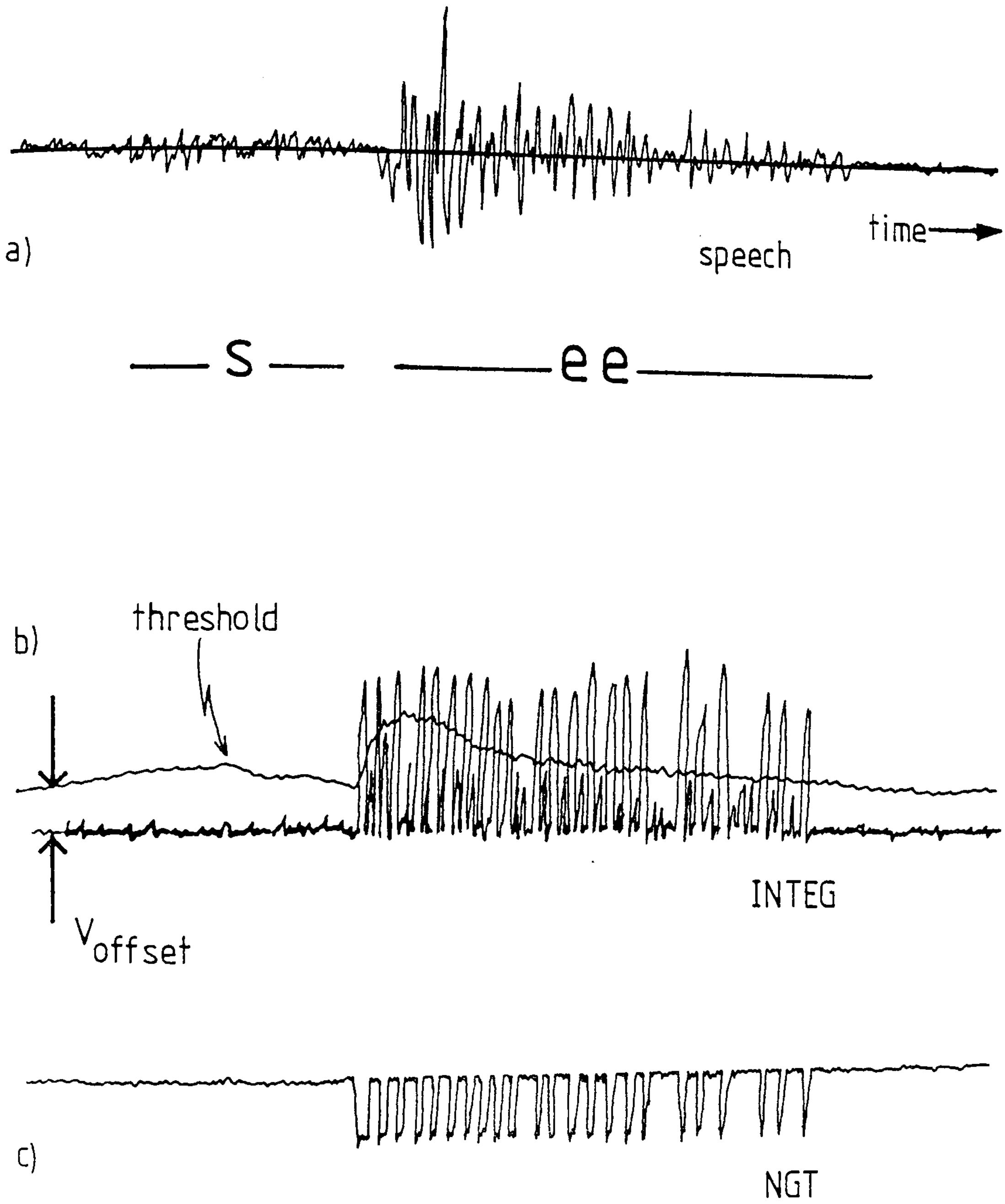


Figure 5-16: Voiced-unvoiced-silence discrimination

CHAPTER SIX

DESCRIPTION OF THE PROJECT SOFTWARE

6.1 SOFTWARE DEVELOPMENT

6.1.1 Handling of assembly language

It was clear at an early stage in the project that the slow execution on the PET of programs written in BASIC made it essential to write the speech analysis software at a lower level (see Section 4.4.1). During early experiments, the present author hand-coded routines directly into hexadecimal machine-code. This is an extremely tedious, time-consuming and error-prone process, and is impractical for large programs. A commercial cassette-based assembler was available for the PET, but suffered from two main disadvantages. Firstly, the assembler was not suitable for assembling disk-based programs, and secondly, the assembly time was over five seconds per instruction on average, the assembler itself being written in BASIC. The author consequently devoted some time to developing a disk-based assembler for the PET, called HAP (Hybrid Assembler for the PET). This uses low-level routines to carry out the normally time-consuming sections of the assembly process such as operation code lookup and symbol table maintenance. This, coupled with the use of efficient hash-coding methods, results in an average assembly time of 1.8 seconds per instruction. HAP source-code employs the standard 6502 operation code mnemonics, pseudo-operation names and labelling conventions, and is used for the low-level software listings in this thesis.

6.1.2 Software design methodology

Some consideration was given to the adoption of a specific software design methodology for the low-level code. The author concluded that a methodology such as that advocated by Jackson [J601] was unsuitable for the present purposes on the grounds that it is not designed to support code optimisation, and would not be suitable for interrupt handling. It was felt that optimisation of code was essential in a real-time system. On the question of interrupt-handling, Palmer [P602] does advocate the use of the Jackson design methodology in interrupt-driven systems. Palmer's particular application, however, has requirements that are different from those of the present project. It was consequently decided that the software should be designed in a 'bottom-up' fashion, as opposed to 'top-down', in order to give maximum emphasis to code optimisation with respect to efficiency of execution.

6.1.3 Description of low-level code

A popular method of describing and documenting low-level software is the use of standard flowchart symbols. These provide a notation for basic action and decision processes. The author feels that a detailed flowchart relating to a lengthy section of low-level code can be almost as difficult to follow as the code itself, and prefers, for the present purposes, to use an

ALGOL-like form of high-level algorithmic notation to describe the underlying structure of the routines that make up the speech analysis software. An attempt has been made to include adequate comment in the software listings themselves; where appropriate, specific instructions and sections of code are referred to in the text by line-number.

6.2 PROGRAMMING THE PET

6.2.1 Monitor facilities

The PET 2001-32N has a built-in monitor, TIM, which provides limited facilities such as memory examine and change. The PET lacks low-level development facilities such as the provision of breakpoints, single-step execution and program trace. In this respect, the development of a fairly complex software system is considerably more difficult on a PET than on a development system such as the Motorola EXORciser [M423]. However, provided that one is prepared to insert software 'patches' in the code, it is possible to simulate such run-time debugging facilities to a limited extent.

6.2.2 PET documentation

The PET 2001-32N is clearly intended to be used as a BASIC microcomputer, and the documentation provided by the manufacturer [C424] [C603] is not sufficiently detailed for serious low-level work. Moreover, the manuals contain a number of errors. The present author has found that much useful information concerning the PET is circulated in popular form for the hobbyist; the publications of the British Independent PET Users' Group (IPUG) have been particularly informative. There are also two privately-published books containing detailed descriptions of the PET operating system [H604] [P605].

Programming details for the 6502 microprocessor and its associated peripheral devices were gleaned from the manufacturer's documentation [M421] [M606], general works on 8-bit microprocessors [O607] [K608], and programming manuals for the 6800 microprocessor, which has much in common with the 6502 [L609] [O610].

6.2.3 Interrupts on the PET

The PET does not use the $\overline{\text{NMI}}$ non-maskable interrupt input to the 6502 microprocessor: this is held high via a resistor, and is available for use by external circuits.

The $\overline{\text{IRQ}}$ interrupt request input to the PET is used by an operating system routine which performs housekeeping functions such as updating the screen, scanning the keyboard and incrementing a real-time clock; interrupts for this routine are generated internally 60 times per second. It was found to be desirable to bypass this routine for two reasons: the execution of this additional code on a regular basis would add to the computational burden during the running of real-time analysis software, and the $\overline{\text{IRQ}}$ input would be required for other purposes during analysis. The latter requirement means that masking of interrupts by setting the $\overline{\text{IRQ}}$ mask bit in the microprocessor's processor status register is insufficient: the interrupt source must be disabled. This is achieved by reconfiguring one of the internal interface adapter registers [H604, p141].

6.2.4 Use of zero-page memory

As mentioned previously (Section 4.4.3), many of the 6502 microprocessor's instructions have shorter execution times when they operate upon the bottom 256 bytes of memory (known as 'zero-page addressing'). It is, therefore, desirable to locate frequently-accessed locations, such as flags and pointers, in this address space. The PET operating system uses almost all of the available locations in page zero of memory, and the values contained therein cannot be overwritten if normal PET operation is required. It is, however, possible to make use of zero-page addresses if the PET housekeeping interrupt routine is disabled. If this is done, then it is vital that the original contents of page zero be copied into a buffer area elsewhere in the memory map, and then restored prior to resumption of normal PET operation.

6.2.5 The use of the PET in the present project

Once one has taken the trouble to learn about the inner workings of the PET hardware and software, and provided that one is careful to disable the 60Hz interrupt source and to maintain page zero values, it is possible to access and use the PET's 6502 microprocessor almost as though it were dedicated to the low-level application.

6.3 OVERALL STRUCTURE OF THE SYSTEM SOFTWARE

The speech analysis system developed during the research project was given the acronym ERSA (Educational Research Speech Analyser), and the software was developed in three 'levels': ERSA level 1, ERSA level 2 and ERSA level 3. Each level consists of a set of software routines, levels 2 and 3 containing routines which enhance the operation of level 1. The levels have the nested structure illustrated in Figure 6-1, so that, for example, ERSA level 2 consists of the level 1 routines plus additional level 2 routines. This structure allowed a set of basic routines to be developed in level 1, before enhancements were made, resulting in level 2. Similarly, ERSA level 3 was built upon the foundation of level 2 routines. Each of the three levels is a functioning pitch-period estimation system in its own right.

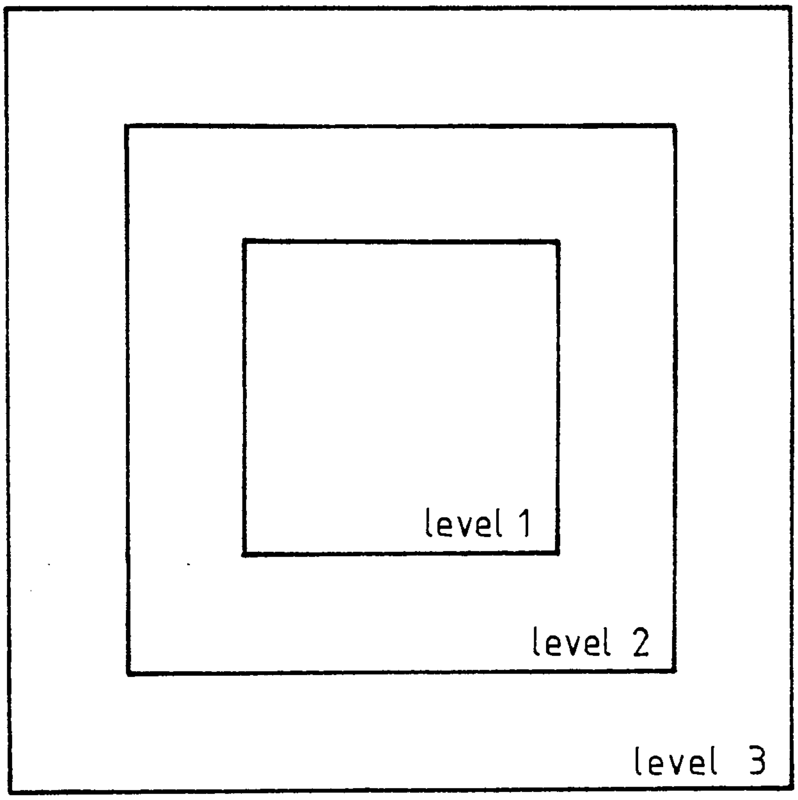


Figure 6-1: ERSA 'level' structure

Level 3 represents the final, most sophisticated version of ERSA.

6.3.1 ERSA level 1

Level 1 routines perform the following tasks:

- handling of input values from the preprocessor, and maintenance of a data structure containing these values;
- estimation of pitch-period values for the speech signal under analysis;
- generation of output values for display on a storage oscilloscope;
- bypassing the PET operating system at the start of analysis, and restoring the operating system when analysis is complete;
- provision of error messages and trace facilities.

6.3.2 ERSA level 2

Level 2 contains routines which perform a 'prediction continuity check' operation, aiming to detect and, if possible, correct, errors in the estimation of pitch-period value. These routines operate in a similar fashion to the error detection/correction algorithms described by Reddy [R312].

6.3.3 ERSA level 3

Level 3 routines perform a 'modified autocorrelation' operation, as described in Section 4.8.2, with the aim of predicting the likelihood of extraneous entries appearing in the data structure. This prediction is used to decide whether level 2 corrective procedures should or should not be applied.

6.4 INTERRUPT HANDLING

As stated previously, ERSA uses both the $\overline{\mathrm{IRQ}}$ and the $\overline{\mathrm{NMI}}$ interrupt inputs to the 6502 microprocessor.

The $\overline{\mathrm{IRQ}}$ input is connected to the 80Hz clock source, generating 80 interrupt requests per second. This is used as a 'request next output value' signal, and the $\overline{\mathrm{IRQ}}$ service routine in ERSA level 1 delivers the current pitch-period estimate to the PET output DAC contained in the preprocessor. This causes an appropriate deflection of the storage oscilloscope trace.

The $\overline{\mathrm{NMI}}$ input is driven by the $\overline{\mathrm{IR}}$ (interrupt request) line from the preprocessor; this indicates that INTEG and COUNT data values are available in the main preprocessor PIA. The $\overline{\mathrm{NMI}}$ service routine loads these values from the preprocessor and stores them in the data structure.

6.5 THE DATA STRUCTURE

The data structure holds COUNT and INTEG values from the preprocessor. Some initial considerations concerning the design of the data structure were discussed in Section 4.7. It was felt that most attention should be given to the efficiency of access to entries in the data structure, as the execution of the analysis software was likely to require many such accesses. The data structure should be large enough to contain entries relating to several tens of milliseconds of speech signal, so that, even in the case of a speaker with a very low average fundamental frequency (say 50Hz, equivalent to a pitch-period value of 20mS), several entries would be present during a voiced portion of speech signal. Finally, the design of the data structure should reflect the requirement to produce regular pitch-period estimates.

The final choice of data structure is shown in Figure 6-2. The data structure occupies 256 bytes of RAM, i.e. one 'page' of memory. This allows efficient access to entries in the data structure by means of the 8-bit index registers in the 6502 microprocessor, which will address a 256-byte range of memory locations offset from a given base address.

The 256-byte data structure is divided into 8 equal 'segments', which are given one- or two-character labels in Figure 6-2. The segment labelled 'C' is the 'core' segment - the segment which will be used as the basis for the pitch-period estimation.

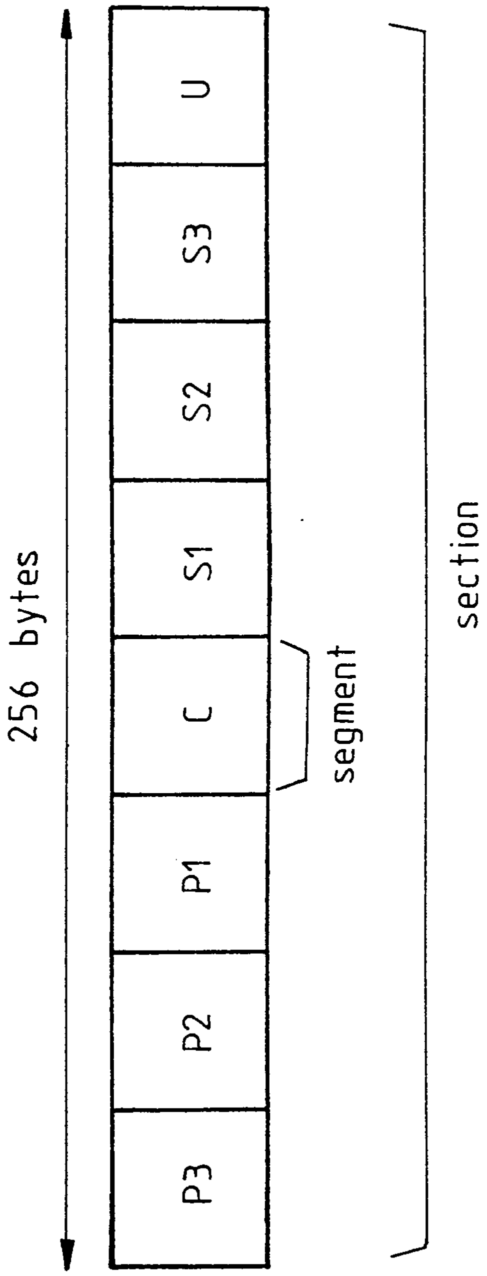


Figure 6-2: ERSA data structure

Segments P1, P2 and P3 are the three 'preceding' segments, i.e. those which contain values which were entered before those in the core segment. Segments S1, S2 and S3 are the 'succeeding' segments, i.e. those which were entered after those in the core segment. Segment U is the 'update' segment, into which new entries are placed.

The 256-byte data structure represents 100mS of speech, each segment being equivalent to 12.5mS. The three preceding segments, the three succeeding segments and the core segment together make up an 87.5mS 'section'. Eighty pitch-period estimates are generated per second of speech, therefore one estimate is made for each 12.5mS segment. At any one time, the current estimate is based upon the current core segment.

Reference to segments within the data structure is made by means of offset pointers from the base address of the data structure. This is illustrated in Figure 6-3. The base address of the data structure is DSSTRT. The offset pointer for the start of the core segment is COFFS. Thus, the starting address of the core segment is the base address offset by COFFS, i.e.

DSSTRT + COFFS.

Each of the eight segments can be accessed in this way.

Once a pitch-period estimate has been made for the core segment, the system must proceed to make an estimate based upon the next segment. That is to say, the segment that was S1 becomes the new C, C becomes the new P1, P1 becomes the new P2, and so on. This

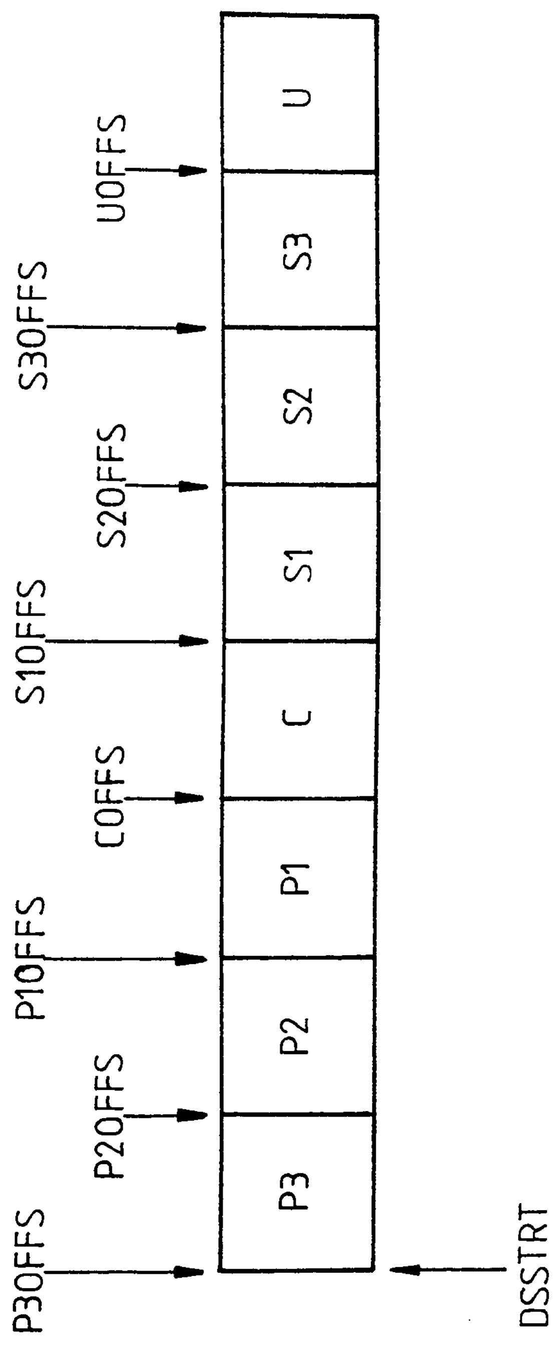


Figure 6-3: Data structure segment offsets

is accomplished by incrementing the values of the segment offset pointers. For example, COFFS is incremented so that it points to the start of what was S1. By simply incrementing the pointer values, the former S1 entries become the core segment entries. The state of the data structure after the segment offset pointers have been incremented is shown in Figure 6-4. This illustrates the 'wrap-around' effect of incrementing the pointers: the data structure is, in effect, a circular buffer, with new values overwriting the former P3 segment. It is important to note that the entries within the data structure do not themselves move; the apparent motion is effected by incrementing the pointers. The progression of segments around the data structure as the pointers are incremented every 12.5mS is illustrated in Figure 6-5.

At any one time, newly-arriving COUNT and INTEG values from the preprocessor are placed into the update segment. When the pointers are next incremented this becomes segment S3, then S2, then S1, and finally the core segment. There is consequently a delay equivalent to four segments between values being entered into the update segment, and the same values appearing in the core segment. As pitch-period estimates are based upon the entries in the core segment, rather than the update segment, this leads to a constant delay of 50mS between the time when values are entered into the data structure, and the time when a pitch-period estimate is made.

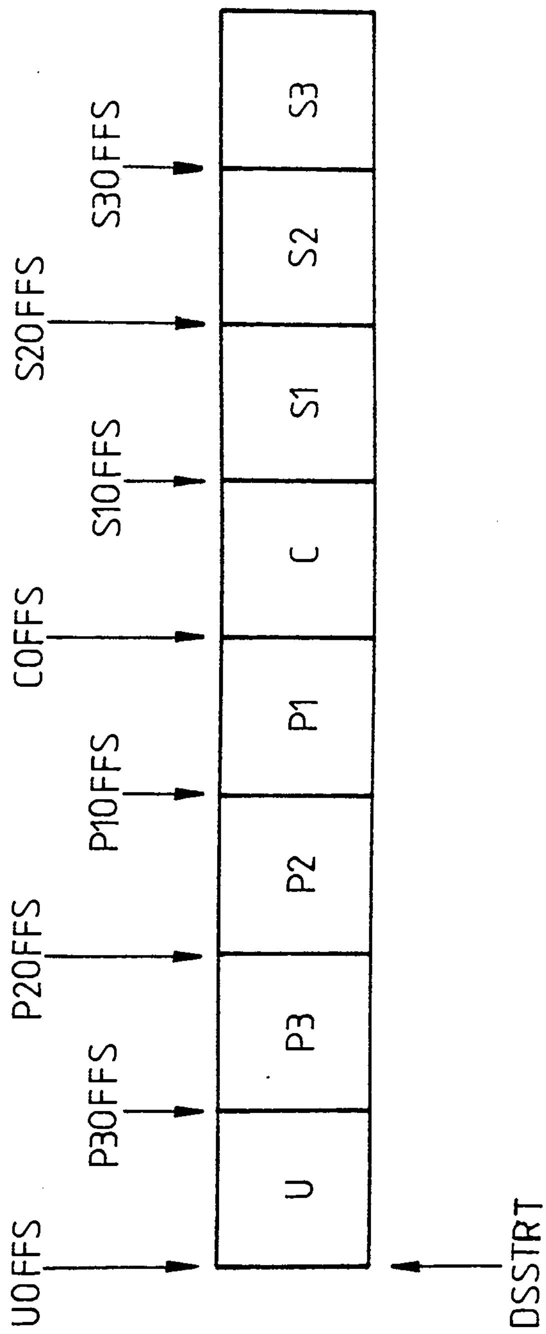


Figure 6-4: Data structure with segment offset pointers incremented

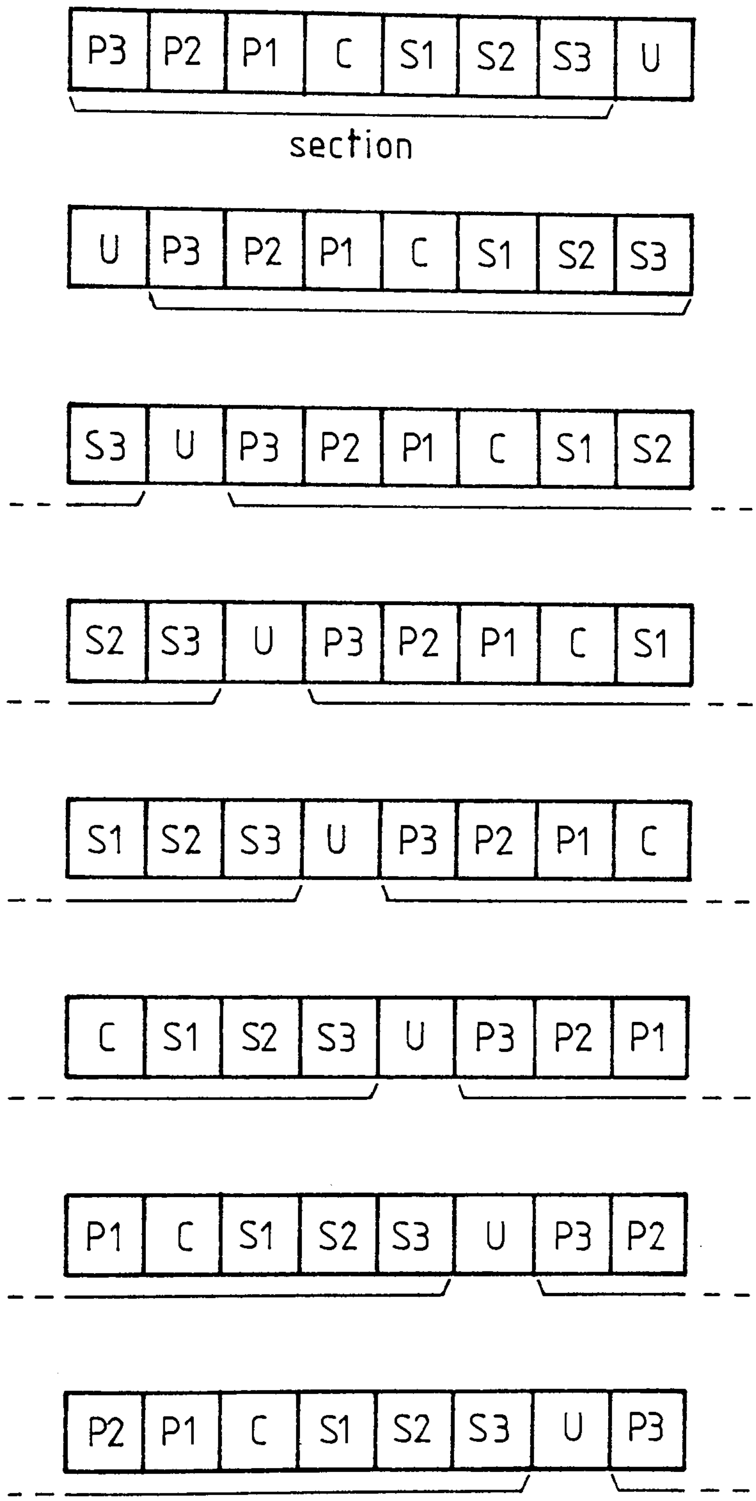


Figure 6-5: Progression of segments around the data structure

The structure of each segment within the data structure is illustrated in Figure 6-6. Each segment consists of 32 bytes of contiguous memory. The first byte in a segment, PTR, contains a pointer to the next available position in the segment; the second byte, NENTS, contains a record of the number of entries in the segment. The penultimate byte in a segment contains the extraneous entry prediction for that segment (this point will be expanded later). The final byte in a segment contains a record of the amount of time for which the processor was idle after completing the pitch-period estimate for that segment. The central portion of the segment is split into two halves. The first half contains the INTEG and COUNT entries received from the preprocessor during that segment; INTEG and COUNT values are stored as pairs. The second half contains 'inter-entry comparison' data, which is used by the extraneous entry prediction routines in ERSA level 3; again, this point will be expanded later.

Each segment can hold a maximum of seven pairs of values from the preprocessor. The 500Hz fundamental frequency upper limit imposed in ERSA means that adjacent entries in a segment are separated in time by at least 2mS; this leads to a maximum of seven pairs of entries in a 12.5mS segment.

Individual entries within the segment can be accessed by offsets from the current segment offset pointer. The extensive use of indexing in accessing values in the data structure was felt to be the most efficient means of addressing data.

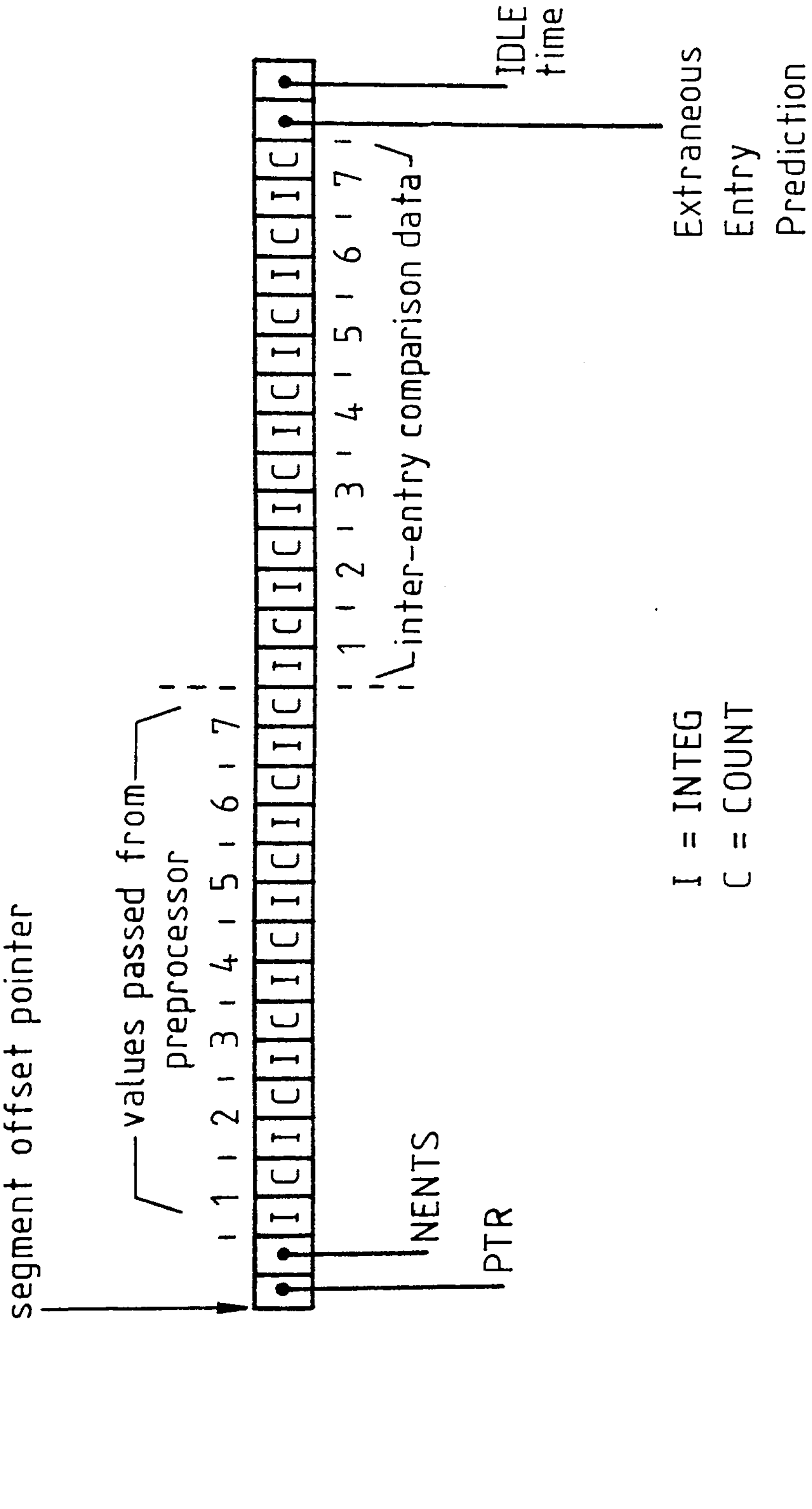


Figure 6-6: Data structure segment composition

6.6 BASIC PITCH-PERIOD ESTIMATION SCHEME

The basic pitch-period estimation scheme, which is embodied in ERSA level 1, involves selecting the COUNT entry which is nearest to the centre of the core segment; this entry is referred to as the 'core entry'.

6.6.1 Valid and invalid COUNT entries

Reference was made in Sections 4.6.2 and 4.7.3 to the maximum possible COUNT value of 25.5mS, which is due to a combination of counter word-length and clock frequency. In fact, any COUNT value greater than 20mS (equivalent to the lower fundamental frequency limit of 50Hz) is automatically set to 25.5mS, which is represented by a byte value of 255 ($FF hexadecimal). This means that every COUNT entry has a byte value which is either in the range 20 .. 200 (2mS .. 20mS) or else is set to 255. A COUNT value of 255 indicates that an unspecified interval greater than 20mS has elapsed since the last significant positive excursion cycle; there is no indication as to the actual length of the interval. For the purposes of core entry selection, a COUNT entry with a value of 255 is regarded as an 'invalid' entry. A 'valid' entry is one in the range 20 .. 200.

6.6.2 Voiced/unvoiced decision

The voiced/unvoiced decision is based upon the central five segments in the data structure, i.e. P2,P1,C,S1,S2. For the current core segment to be considered voiced, there must be at least two consecutive valid entries in the central five segments, and these entries must, at least partially, span the core segment. Figure 6-7 shows three examples of voiced core segments. Valid entries in the data structure are represented by the solid circles. An example of an unvoiced core segment is shown in Figure 6-8; in this case, there are two consecutive valid entries in the central five segments, but these do not span the core segment.

6.6.3 Selection of core entry

If the core segment contains only one entry, then that is considered to be the core entry (Figure 6-9a). If the core segment contains more than one entry, then the nearest entry to the centre of the core segment is considered to be the core entry, favouring the 'preceding' side of the segment in the case of an even number of entries in the core segment (Figure 6-9b). If the core segment is empty, then the nearest preceding entry is selected (Figure 6-9c). It will be seen from Figure 6-9c that the core segment does not have to contain an entry in order to be considered voiced: pitch-period estimation is based upon the central five segments in the data structure, but the value chosen is attributed to the core segment.

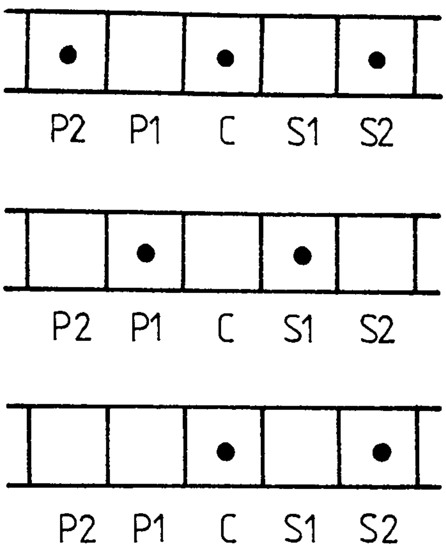


Figure 6-7: Voiced core segments

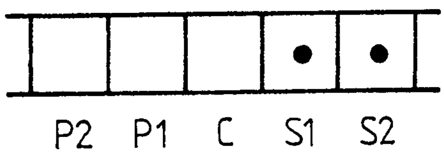


Figure 6-8: Unvoiced core segment

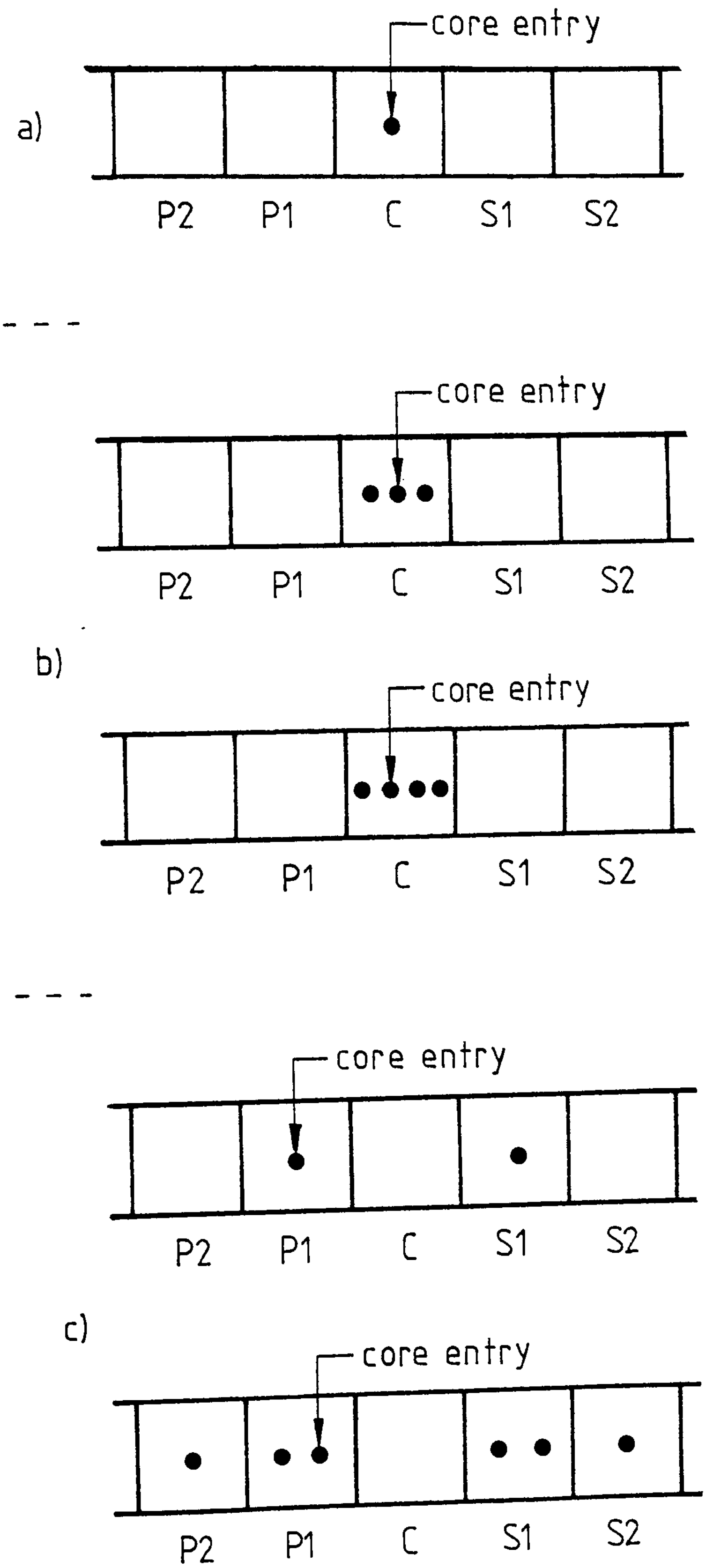


Figure 6-9: Selection of core entry

Once the core entry has been selected, the COUNT value of the core entry is output as the pitch-period estimate for the current core segment.

6.7 DETECTION AND CORRECTION OF ERRORS

The crude pitch-period estimation scheme embodied in ERSA level 1 assumes that all COUNT values in the data structure are valid pitch-period values. In practice this is not always the case: the data structure may contain entries relating to insignificant excursion cycles, or else some significant excursion cycles may have been missed. ERSA levels 2 and 3 include routines which attempt to detect and correct erroneous pitch-period estimates.

6.7.1 Prediction continuity check

The 'prediction continuity check' routines in ERSA level 2 attempt to detect any pitch-period estimate which is not consistent with the previous estimate, in other words to ensure that there is continuity between successive estimates. This approach is based upon the error detection/correction routines suggested by Reddy [R312]; a flow diagram for Reddy's algorithm is given in Figure 4-8, and Reddy's terminology is explained in Section 4.7.3.

The algorithm adopted in the present system is as follows:

```
calculate relative error (RE) between
previous and present pitch-period estimates;
IF    |RE| < 1/8
THEN accept the present estimate
ELIF |RE| < 1/4
THEN check for a 'hop'
ELIF RE < -1/4
THEN check for an 'extra marker'
ELIF RE > 3/4
THEN check for a 'hole'
ELSE {assume uncorrectable}
        accept the present estimate
ENDIF
```

The above algorithm assumes that the rate of change of pitch-period will never be so great as to give a relative error with an absolute value greater than 1/8 between successive segments. The algorithm has been tested on sections of speech containing rapid glides, and appears to perform well in these cases. The specific actions taken for a 'hop', 'hole' or 'extra marker' are detailed in Section 6.8.

6.7.2 Extraneous entry prediction

ERSA level 3 contains 'extraneous entry prediction' routines as outlined in Section 4.8.2. These routines are used to limit the application of corrective routines in ERSA level 2, in order to lessen the chances of attempting to correct entries which are, in fact, not in error. Two specific errors which are found to occur with ERSA level 2 routines are illustrated in Figures 6-10 and 6-11. Both Figures show the output from the integrator with the integral threshold superimposed. Excursion cycle integrals exceeding the threshold level are classed as 'significant'. In the example illustrated in Figure 6-10, the second excursion cycle integral (marked 'A') in an otherwise regular group of integrals falls below the threshold level, and is therefore excluded. Interval 'T' is consequently taken as the first pitch-period estimate. In such a case, in which there is an initial erroneous pitch-period estimate, it is possible that ERSA level 2 routines, comparing subsequent (correct) COUNT values with interval 'T', will assume the presence of extra markers in subsequent pitch-periods, and will remove these supposed extraneous entries, leading to an overall doubling of pitch-period value.

In the example illustrated by Figure 6-11, a section of voiced waveform contains one major insignificant excursion cycle per pitch-period. This is, in most cases, of sufficient energy to exceed the threshold. At the point marked 'B', the insignificant entry does fall below the threshold level. The natural action of

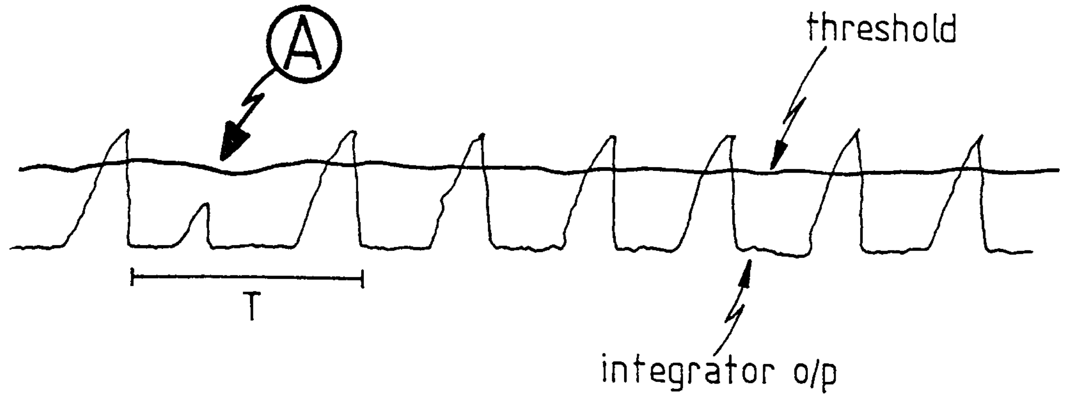


Figure 6-10: Example leading to errors in ERSA level 2

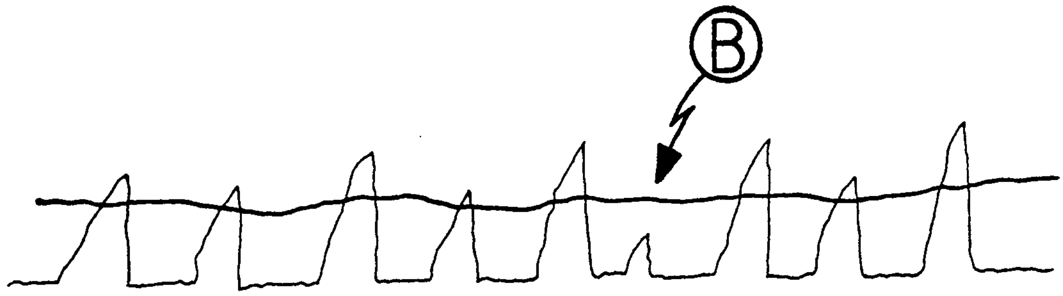


Figure 6-11: Further example leading to errors in ERSA level 2

ERSA level 2 routines will be to assume a 'hole' at that point, and to insert an additional entry. In this way, the prediction continuity routines may strengthen an estimation error.

The extraneous entry prediction routines of ERSA level 3 are used to lessen the likelihood of the above two errors. In the case of supposed extra markers in the waveform (Figure 6-10), the level 2 corrective mechanism is only applied if the extraneous entry prediction indicates an overall trend towards extraneous entries. In the case of an apparent hole (Figure 6-11), the level 2 corrective mechanism is only applied if the extraneous entry prediction suggests that there are no extraneous entries present. Thus, in Figure 6-10, for which the level 3 prediction will be one of no extraneous entries, COUNT values following 'T' are accepted as they are. In Figure 6-11, for which level 3 routines will predict the presence of extraneous entries, extra markers after point 'B' will be removed by level 2 routines.

6.8 DESCRIPTION OF THE LOW-LEVEL CODE

The low-level code will be described in this Section. A full listing of the code, with line-numbers, appears in Appendix 4.

6.8.1 Reserved bytes and equates

Declarations of reserved bytes and equates appear at the head of the software listing (lines 14..165). Wherever possible, flags and variables are positioned in zero page memory. In view of the increased speed of operation when handling zero page memory, values throughout the program are generally passed in the form of zero page variables rather than making use of the stack.

6.8.2 SETUP routine

This routine (line 380) is called when the low-level software is entered. It is responsible for altering the interrupt vector addresses, shifting the PET operating system's zero page data into a copy area, and initialising variables and system PIAs.

The IRQ interrupt source used by the PET housekeeping routines is disabled by altering an internal PIA register (line 387), and the interrupt mask is set (line 385). The $\overline{\text{NMI}}$ interrupt is effectively disabled by altering the vector address to point to a 'return from interrupt' (RTI) instruction (line 391). A loop in the code (lines 435..437) waits for the user to press the 'space' bar on the keyboard: this waiting period is introduced in order to allow the user to switch the preprocessor on with the interrupts disabled. When the system is fully powered up, the user presses the 'space' bar, and only then are the preprocessor PIAs initialised and the interrupts enabled. When this has been accomplished, the program jumps to the main pitch-period estimation routine, ESTIM.

6.8.3 FINISH routine

This routine (line 477) is called when it is required to return to normal PET operation. The preprocessor interrupt sources are disabled, the PET zero page values are restored, and the PET $\overline{IRQ}$ interrupt source is re-enabled.

6.8.4 ERROR routine

This routine (line 516) was used during the development of the software, and has been left in the code in case run-time debugging is ever required. When called, normally from a software 'patch', the routine restores the PET screen, and prints an error message together with the value of the byte labelled ERRNUM. The latter may be loaded with some meaningful value prior to the call to this routine.

6.8.5 NMISR routine

This, the $\overline{NMI}$ service routine (line 764), is called automatically whenever an interrupt is generated via the $\overline{NMI}$ input. This occurs whenever the preprocessor has COUNT and INTEG values ready to be passed over to the microcomputer.

The underlying algorithm for this routine is as follows:

```
load COUNT and INTEG values from the preprocessor;
IF    the IRQ interrupt service routine is currently running
THEN set NMIOCC flag;
        EXIT from NMISR routine
ELIF COUNT < 2mS
THEN IF    this is the first entry in the update segment
        THEN set LT2MS flag   (see Section 6.8.6)
        ELIF the previous INTEG entry < the current one
        THEN remove the previous entry
        ELSE ignore the current entry
        ENDIF
ENDIF;
update the UPDATE segment;
IF    the COUNT entry to be entered into the UPDATE segment
        exceeds 20mS
THEN set the entry to $FF
ENDIF;
return from interrupt
```

6.8.6 IRQSR routine

This routine, the $\overline{IRQ}$ interrupt service routine (line 544), is called automatically whenever an $\overline{IRQ}$ interrupt is generated. These interrupts are generated by the preprocessor every 12.5mS, and indicate that a new pitch-period estimate is required. The operation of the routine is as follows:

```
IF    TRACE flag is set
THEN call routine COPYDS
ENDIF;
IF    INEST flag is set {ESTIM routine has not terminated}
THEN clear INEST flag;
      set current pitch-period estimate (THISPP)
      equal to the last estimate (PREVPP);
      alter return address on the stack
      to the start address of ESTIM routine
ENDIF;
IF    LT2MS flag is set {first COUNT in update seg < 2mS}
THEN IF    first INTEG in update seg < final INTEG in S3
      THEN remove first entry in update segment
      ELSE remove final entry in S3 segment
      ENDIF
ENDIF;
increment segment offset values;
set pointers for new S3 segment;
call inter-entry comparison routine IEC;
convert pitch-period estimate to frequency equivalent;
IF    there is room in the output table
THEN store pitch-period estimate in output table
ENDIF;
IF    NMIOCC flag is set {NMI interrupt has occurred}
THEN re-enter NMISR routine
ELSE return from interrupt
ENDIF
```

6.8.7 ESTIM routine

This routine (line 174) is the main pitch-period estimation routine. The major tasks of the routine are to determine whether the core segment is voiced, and, for a voiced core segment, to select the 'core entry'. The algorithms for these two processes have already been detailed in Sections 6.6.2 and 6.6.3. These algorithms have been implemented in the ESTIM routine: the voiced/unvoiced decision is made between lines 174 and 294, and the core entry selection process occupies lines 297..343. For unvoiced core segments, a pitch-period prediction of $FF is recorded. The prediction continuity check (PCC) routine is called (line 345), and the state of the ACCEPT flag is examined upon return from PCC (line 346). If ACCEPT is clear, i.e. the prediction is not acceptable, then the ESTIM routine is re-run to produce a revised pitch-period estimate. Otherwise, the pitch-period value is accepted as it stands, and the process enters a loop, which is terminated by one of two events: the generation of an $\overline{\text{IRQ}}$ interrupt request, or the user pressing the 'STOP' key on the PET keyboard. If the latter event is detected, the FINISH routine is called (line 367), the PET stack pointer is restored (line 368) and a return is made to normal PET operation (line 370). During the execution of the waiting loop, an idle time counter (IDLE) is incremented to indicate the amount of idle time spent during the current execution of ESTIM. An inner delay loop (lines 361..364) is included so that each increment of IDLE corresponds to 50μS of idle time. The IDLE value is stored in the last byte of the core segment (see Figure 6-6), so that a check may be made on real-time performance.

6.8.8 COPYDS routine

This routine (line 885) is called by the IRQSR routine if the TRACE flag is set. The routine creates a copy of the data structure elsewhere in RAM, the available area being from \$4000 to \$7FFF (the RAM limit for a 32k PET). When the available RAM has been filled, COPYDS calls the FINISH routine (line 896) and then returns to normal PET operation, so that the 'trace' of consecutive data structure copies can be examined using the TIM monitor.

The COPYDS routine employs self-modifying code: the statement on line 887 initially reads:

```
STA $4000,Y.
```

As successive data structure copies are made, the first byte of the address in the statement is incremented, thus:

```
STA $4000,Y
STA $4100,Y
STA $4200,Y
...
```

This continues until the address \$8000 is reached, whereupon control returns to the PET operating system. Between \$4000 and \$7FFF there is room for 64 consecutive copies of the data structure, so that each 'trace' lasts for 0.8 Sec. The trace facility was of great value during software development, as it gives a full dump of the data structure contents at 12.5mS intervals. The COPYDS routine is not used during normal ERSA operation.

6.8.9 PCC routine

This routine (line 912) is the prediction continuity check. For all voiced estimates, PCC calls the CALCRE routine (see Section 6.8.10), and then determines whether or not to call one of the corrective procedures. The algorithm is given in Section 6.7.1. The PCC routine returns with the ACCEPT flag set if the current estimate is acceptable, or cleared if corrective measures have been applied and the ESTIM routine is to be re-run.

6.8.10 CALCRE routine

This routine (line 1254) calculates the relative error between the value stored in PPRED and PREVPP, the latter being the previous pitch-period estimate. CALCRE returns with RELERR set to one of seven values, depending upon the calculated relative error value. The values of RELERR on return from CALCRE are given in the software listing (lines 1244..1251). The routine calculates PREVPP/2 and PREVPP/8 by logical shifts, and adds or subtracts various combinations of these fractional values to PREVPP while comparing the results with PPRED.

6.8.11 GETPRE routine

This routine (line 1093) and its partner GETSUC (see Section 6.8.12) are used by the ERSA level 2 corrective routines.

GETPRE searches for the 'preceding neighbour' of the core entry: this is the data structure entry which immediately precedes the core entry. Once the preceding neighbour has been found, GETPRE sets the following pointers and flags:

PSEG points to the segment containing the preceding neighbour;
PENT points to the entry within PSEG;
PREOFF is the offset from DSSTRT for the preceding neighbour;
PCOUNT is the COUNT value of the preceding neighbour;
PREFND is a flag set to indicate that a preceding neighbour has been found.

If no preceding neighbour is found, the flag PREFND is cleared.

6.8.12 GETSUC routine

This routine (line 1170) operates in much the same way as GETPRE, and locates the 'succeeding neighbour' of the core entry: that data structure entry which immediately follows the core entry. GETSUC sets bytes SSEG, SENT, SUCOFF, SCOUNT and SUCFND with respect to the succeeding entry; these bytes are used in the same way as PSEG, PENT, PREOFF, PCOUNT and PREFND in the GETPRE routine.

6.8.13 HOP routine

The HOP routine (line 953) is called from the PCC routine whenever the absolute value of the relative error is greater than 1/8 but less than 1/4. A 'hop' in Reddy's terminology (see Section 4.7.3) is a case in which an incorrect peak in the waveform is chosen as the major peak for a particular pitch-period measurement. In such a case, the pitch-period estimate is usually rather more or rather less than it should be, and one of the neighbouring COUNTS is correspondingly smaller or larger respectively. The corrective procedure adopted here is to make a new entry in the data structure with a COUNT value which is the mean of the core entry and the neighbouring COUNT value.

The algorithm is as follows:

```
IF    THISPP > PREVPP
THEN examine neighbouring entries;
        IF    a neighbouring entry < PREVPP,
              and within 1/4 of PREVPP
        THEN add the neighbouring COUNT to THISPP;
              divide by 2;
              enter the result as the COUNT value for
              both data structure entries
        ELSE accept THISPP as it is
        ENDIF
ELSE (PREVPP < THISPP)
        examine neighbouring entries;
        IF    a neighbouring entry > PREVPP,
              and within 1/4 PREVPP
        THEN add the neighbouring COUNT to THISPP;
              divide by 2;
              enter the result as the COUNT value for
              both data structure entries
        ELSE accept THISPP as it is
        ENDIF
ENDIF
```

XHOP counts the number of calls to the HOP routine, and XCHOP counts the number of correctable hops; this information was useful during software development.

6.8.14 ABSDIF routine

This routine (line 1563) calculates the absolute difference between PREVPP and the value of the accumulator on entry. On exit from ABSDIF, the accumulator is set to this absolute difference value.

6.8.15 EXM routine

The EXM routine (line 1362) processes cases in which the presence of an extra marker is suspected: this is an unwanted entry in the data structure, which results in a pitch-period estimate which is much shorter than would be expected. An attempt to remove the extra marker is only made if the extraneous entry prediction indicates that extraneous entries are likely to be present in the data structure. The extraneous entry prediction routine is called in line 1366. If the extraneous entry prediction indicates that no extraneous entries are likely, then the current pitch-period estimate is accepted as it is.

The extra marker corrective procedure involves locating the preceding and succeeding neighbours, adding the current pitch-period estimate to the COUNT entry of each, and selecting the combination with the lower absolute difference from PREVPP, the previous pitch-period estimate. If only one neighbour is found, the combination of this COUNT entry plus the current pitch-period estimate is used. The data structure is then amended so as to enter the chosen combination as a COUNT entry:

this effectively relocates the pitch-period marker. If neither neighbour is found, the present pitch-period estimate is accepted as it is.

XEXM counts the number of calls to the EXM routine.

6.8.16 HOLE routine

This routine (line 1588) is called to cope with a possible 'hole' - one or more missing entries in the data structure.

The extraneous entry prediction routine is called (line 1592), and an attempt to correct the hole is only made if the indication is that there are no extraneous entries in the data structure; otherwise, the current pitch-period estimate is accepted as it is. For simple holes, which may result in pitch-period doubling, an attempt is made to insert an entry by halving the curent pitch-period estimate. If the result is not close enough to the previous pitch-period estimate, then an attempt is made to insert an entry corresponding to the mean of the present pitch-period estimate and the COUNT of the succeeding neighbour. In cases where several significant waveform peaks have been missed, it is sometimes possible to recreate the missing entries in this way.

The underlying algorithm is as follows:

```
IF    THISPP < $FF
THEN divide THISPP by 2;
        IF    result is within ±1/8 of PREVPP
        THEN make a new entry in the data structure
                corresponding to this COUNT value;
                EXIT
        ENDIF
ENDIF;
search for succeeding neighbour;
IF    succeeding neighbour not found
THEN THISPP := $FF {unvoiced}
ELIF succeeding neighbour within ±1/4 of PREVPP
THEN form mean of PREVPP and COUNT of
        succeeding neighbour;
        make a new entry in the data structure
        corresponding to this COUNT value
ELSE THISPP := $FF {unvoiced}
ENDIF
```

XHOL counts the number of calls to the HOLE routine.

6.8.17 COMP routine

This routine (line 1800) performs a comparison operation for the extraneous entry prediction process. The nature of the inter-entry comparsion was described in Section 4.8.2. The COMP routine compares ICOMP1 with ICOMP2, placing the result in IECC. The possible results are given in the software listing (lines 1790..1797); the specific values used were chosen to facilitate bit-testing: $80 has the most significant bit set, and $01 has the least significant bit set.

6.8.18 ENTIEC routine

The ENTIEC routine (line 1864) enters inter-entry comparison data produced by the COMP routine (IECI and IECC) into the data structure.

On entry to the ENTIEC routine, the Y-register contains an entry location offset: the data are stored in two adjacent locations, the first of which is (S3OFFS + $10 + Y-register*2).

6.8.19 IEC routine

This routine (line 1721) is called from the IRQSR routine, and produces inter-entry comparison data for entry into the S3 segment of the data structure.

For each entry in the S3 segment, the IEC routine produces a comparison value for the INTEG and COUNT values, and stores this information in the data structure.

6.8.20 EXTRA routine

The EXTRA routine (line 1903) is called whenever a hole or extra marker is suspected. EXTRA constructs two lists, INTLIS and CNTLIS, comprising COUNT and INTEG comparison data from the data structure for the current 87.5mS section. It then calls the modified autocorrelation routine AUTO, and gives a prediction of the likelihood of extraneous entries being present in the current section. This prediction is saved in EEPRED, and is also placed in the penultimate byte in the current core segment.

The EEPRED prdediction is based upon a combination of modified autocorrelation results for the INTEG values and the COUNT values. Details of this combination were given in Section 4.8.2.

6.8.21 ACOMP routine

The ACOMP routine (line 2144) is called from the AUTO modified autocorrelation routine, and compares two values, AVAL1 and AVAL2, returning with the accumulator loaded with one of three values. These return values, and the conditions for which they occur, are given in the software listing, lines 2139..2141.

6.8.22 AUTO routine

This routine (line 2010) performs a modified autocorrelation operation on one of the two inter-entry comparison lists, INTLIS and CNTLIS.

On entry to the AUTO routine, LISST points to the start of the list and LISPTR contains the number of entries in the list. The details of the modified autocorrelation process have already been discussed in Section 4.8.2, and the implementation used in ERSA follows the algorithm outlined in that Section. AUTO returns with the accumulator loaded with the extraneous entry prediction for the list that has just undergone modified autocorrelation.

6.9 DESCRIPTION OF THE BASIC PROGRAM

By comparison with the low-level software, the BASIC program which interfaces between the user and ERSA is extremely simple. A listing of the BASIC program is given in Appendix 5. The program prints a heading and simple instructions to the user, and then executes a SYS 12644 instruction, which causes a jump to the ERSA SETUP routine. When the user terminates ERSA execution by pressing the STOP key, control returns to the PET operating system.

CHAPTER SEVEN

ERSA IN PRACTICE

7.1 USING ERSA

7.1.1 Setting up the microcomputer

The procedure for setting up the ERSA software has already been outlined in Chapter Six. The preprocessor unit should be left switched off until the software has been set running: this prevents interrupts being generated by the preprocessor and disrupting the microcomputer. The sequence of operations is as follows:

1) Load files ERSA-L and ERSA-H from disk or cassette.

2) Type 'RUN' to commence execution of the high-level code. This will cause instructions to appear on the screen.

3) Switch on the preprocessor.

4) Press the 'space' bar on the PET keyboard.

The ERSA software should now be running. When it is required to return to normal PET operation, press the 'STOP' key on the PET keyboard.

7.1.2 Setting up the controls

The digital storage oscilloscope which is used as a display device for ERSA is an integral part of the system, and certain functions of ERSA, such as display storage, are obtained by manipulating controls on the oscilloscope. The 'display mode' switch on the oscilloscope determines whether single-shot ('refreshed') or roll mode is to be used; examples of these two modes are given in Section 7.1.4. The timebase control governs the length of utterance which will fit on the screen (in single-shot mode), or the trace rolling rate (in roll mode); this control should be set to suit the application. With the timebase set to 500mS/cm, the width of the screen corresponds to 5 seconds. The shift and gain controls for the two Y-input channels will affect the position of the traces on the screen, and the amount of vertical deflection for a given change in fundamental frequency. Again, these controls should be set to suit the particular application. The 'Y mode' switch should be set the position marked 'CH1 & CH2', and the trigger source switch should be set to 'EXT+'.

The control unit associated with the ERSA preprocessor is designed to provide most of the control facilities required during use of the system. The gain control varies the level of signal fed to the compressor circuit. The inclusion of the latter allows the system to operate to a certain extent independently of gain setting. The optimum gain setting will be that providing the smoothest fundamental frequency contours; in practice, it has been found that it is sufficient to leave the gain control set to its 50% level. The control unit has inputs for a microphone and a tape-recorder: these inputs are switch-selectable. An 'invert signal' switch is provided to ensure that the major peaks in the voiced portions of the speech waveform are of positive polarity when fed to the integrator. The required setting of this control can be noted for use with any particular tape-recorder. If necessary, the signal at the input to the integrator can be examined with an oscilloscope to check the polarity of the major peaks. Generally, the correct setting of the 'invert signal' switch will result in a smoother fundamental frequency contour than that obtained with the incorrect setting. The three-position rotary 'display' switch on the control unit governs the position of the frequency and intensity traces on the oscilloscope. The 'manual trigger' pushbutton provides a trigger signal to the oscilloscope. The use of both of these controls is explained in Section 7.1.4.

7.1.3 The fundamental frequency display

The fundamental frequency display is produced by a 'table lookup' process based upon a pitch-period value. The frequency values are chosen so as to give a logarithmic display of fundamental frequency. This is the display mode adopted in the Laryngograph/Voiscope, and has the advantage that pattern proportions remain the same for speakers with high or low average fundamental frequency [F157,p40]. The main uses envisaged for ERSA do not require precise measurement of fundamental frequency, but rather the display of fundamental frequency contours. For this reason, it was felt unnecessary to build any frequency calibration facilities into ERSA. On-screen frequency calibration may be achieved, if required, by feeding standard frequency signals from a signal generator into ERSA, and noting the positions of the resultant traces.

7.1.4 Specific ERSA display configurations

A number of display configurations are possible with the ERSA system. The three that have proved most useful are outlined below.

7.1.4.1 Single sweep - frequency and intensity

This mode allows the frequency and intensity contours for a set length of input signal to be stored. The procedure is as follows:

1) Set the oscilloscope display mode switch to 'refreshed'.

2) Set the rotary 'display' switch on the control unit to position 'a', and depress the 'manual trigger' pushbutton: the oscilloscope will commence a single sweep refresh of the screen. The speech signal should be fed into the system while the sweep is in progress. At the end of the sweep, the display is automatically stored. With the 'display' switch in position 'a', the stored display has the frequency contour in the upper half of the screen, and the intensity contour in the lower half. These trace positions may be reversed by setting the 'display' switch

to position 'c'.

3) To clear the screen, set the 'display' switch to position 'b' ('OFF'), and execute a single sweep by depressing the manual trigger pushbutton.

7.1.4.2 Roll mode - frequency and intensity

In this mode the frequency and intensity contours roll across the screen continuously:

1) Set the oscilloscope display mode switch to 'roll'.

2) The two traces will begin to roll across the screen: speech may be fed into the system for analysis at any time.

3) To store a particular section of the display, depress the 'full store lock' button on the oscilloscope.

7.1.4.3 'Target and trial' display

This display mode is useful for teaching specific intonation patterns:

1) Set the 'display mode' switch on the oscilloscope to 'refreshed'.

2) Set the 'display' switch on the control unit to position 'c'. Depress the manual trigger pushbutton, and output a 'target' display on the screen. This is a model intonation pattern, and may be provided by a teacher, or from tape.

3) When the oscilloscope has completed its single sweep, depress the 'half lock' button on the oscilloscope. This retains the frequency contour in store on the lower half of the screen.

4) Set the 'display' switch on the control unit to position 'b', and provide a manual trigger signal: this clears the upper half of the screen.

5) Set the 'display' switch on the control unit to position 'a'. With the system as set up, the student may practice obtaining a close match to the 'target' display by depressing the manual trigger pushbutton and repeating the test utterance. Whenever this is done, the 'trial' frequency contour is displayed on the upper half of the screen, while the 'target' contour is retained on the lower half of the screen. The 'target' display will only be lost when the 'half lock' switch is released.

7.2 TESTING ERSA

Consideration was given to the various options available for testing the ERSA speech analysis system.

7.2.1 Approaches to the testing of pitch-period estimation systems

A search of the relevant literature revealed two main approaches to the testing of pitch-period estimation systems. The first involves comparison of the results producd by the system under test with those produced by a 'yardstick' system. Outputs from the new and established systems are generally superimposed in order to illustrate the degree to which the system under test matches up to the yardstick system. One of the most detailed tests of this nature known to the author is the comparative performance study conducted by Rabiner et al [R381], in which the yardstick is provided by the 'semiautomatic pitch detector' of McGonegal et al [M384]. One of the problems with this approach to the testing of algorithms would seem to be the selection of a yardstick: ideally this should be a system of absolute accuracy and reliability. No such system is known to the present author. Any meaning attached to a yardstick comparison is limited by the degree to which the accuracy of the yardstick is known. At best, such a comparative test can lead to a conclusion such as 'system A is nearly as good as system B'.

The second main method of testing pitch-period systems, applicable in those cases where the pitch-period estimation operation is to be used as part of a speech synthesis system, is to generate synthetic speech using the system under test, and to collect the subjective evaluations of human subjects hearing the output. The subjects may be asked to comment upon the 'naturalness' of the synthetic speech, or may be required to compare the quality of the output with some form of yardstick recording. A major investigation of this kind has been conducted by McGonegal [M386].

The present author feels that the second approach to testing is the more satisfactory, inasmuch as it is carried out with regard to the intended application of the system under test. Where the system is to be used to generate synthetic speech, it is meaningful to ask human subjects to comment upon the results obtained. Experimental procedures for handling the subjective evaluations of subjects are well established in the Social Sciences.

An attempt has been made to provide some test results for the ERSA system which take into account the intended application area: namely, the visual display of fundamental frequency contours of speech signals. Here, as stated earlier in this Chapter, the main requirement is the display of overall frequency patterns rather than absolute accuracy of fundamental frequency estimation at a given instant in time. It seems sensible to the present author to compare the results produced by ERSA with those produced by a conventional speech spectrograph, which is the device presently used to provide fundamental frequency contours in the intended application areas of ERSA. Furthermore, the test utterances used have been chosen from material currently of relevance to teaching and research work in the Department of Educational Enquiry at Aston.

The test utterances chosen are spoken by a range of adult male and female speakers. No attempt has been made to present results of the analysis of the speech of children, or of adults with speech defects: analysis of such speech falls outside the requirements of the present system, as stated in Chapter One. All the subjects used in the tests are native speakers of English.

7.2.2 Presentation of results

The results of the analysis of the test utterances are reproduced on the following pages. The test utterances are as follows:

1) The phrase "the teaching of the older students", spoken by five male and five female undergraduates. Recordings were made in normal room conditions directly on to the Series 700 spectrograph, using an AKG D190 microphone.

2) Two selected utterances from a BBC radio programme, spoken by a professional actor and actress. These utterances were taken from a tape-recording produced by the BBC, and are representative of material which is used to demonstrate the effective use of intonation. The recordings were transcribed on to the spectrograph.

For each utterance, a narrow-band spectrogram with expanded frequency scale, and a frequency contour produced by ERSA were generated. The results are reproduced in Figures 7-1 to 7-12; the ERSA outputs have been re-plotted, so that the timescales are the same as those of the spectrograms. The tape-recorded utterances represent a fairly wide range of fundamental frequencies, with most variation being found in the two BBC recordings. It will be seen that, on the whole, the ERSA output closely resembles the fundamental frequency contour visible on the spectrograms, and appears to follow fairly rapid changes in fundamental frequency: this is especially noticeable in the BBC recordings.

The application of ERSA level 2 and 3 error correcting procedures has resulted in the production of fundamental frequency contours which are much smoother than those produced by the level 1 routines. The action of the rules in cases where the precise location of significant excursion cycles is not clear is either to repeat the previous estimate, or else to output an 'unvoiced' estimate. The former action is responsible for the occasional appearance of 'stepped' sections of the fundamental frequency contour. ERSA level 1 routines operating on their own tend to produce spurious output estimates when erroneous entries are present in the data structure. The ability to distinguish between voiced and unvoiced sections of an utterance appears to be very good. ERSA tends to reject short bursts of voicing, but rarely produces a voiced estimate for an unvoiced section of speech. It will be noticed from the test results that the onset of a voiced section of speech is often slightly delayed in the ERSA output, and there is a similar tendency for ERSA to indicate the end of voicing slightly early: in other words the very beginning and end of a voiced section of speech are often missed. It is, however, rare for ERSA to lose the ability to track the fundamental frequency during a voiced section.

The present author's experience with ERSA suggests that the system produces results which are very similar to the fundamental frequency contours visible on narrow-band spectrograms. The system appears to work well with both male and female subjects.

7.3 FURTHER TESTING OF ERSA

It was decided at the outset of the present project that any testing of ERSA in the role of providing feedback information in a learning situation was beyond the scope of the project. It is, however, hoped that it will prove possible in the future to undertake an exhaustive study based upon this application of ERSA. It is also hoped that future studies may examine the performance of ERSA while analysing defective speech, and the speech of children.

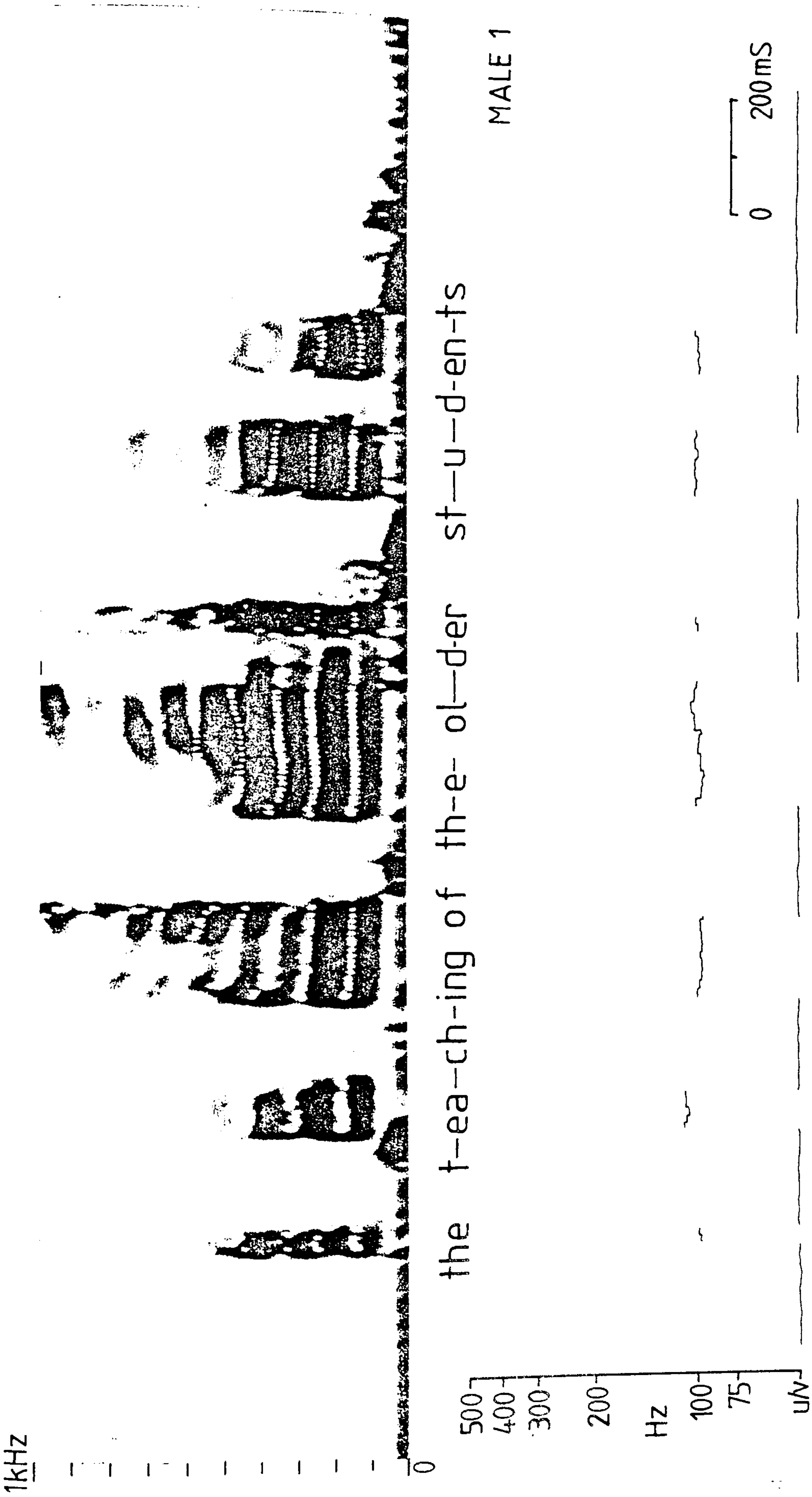


Figure 7-1: ERSA test results

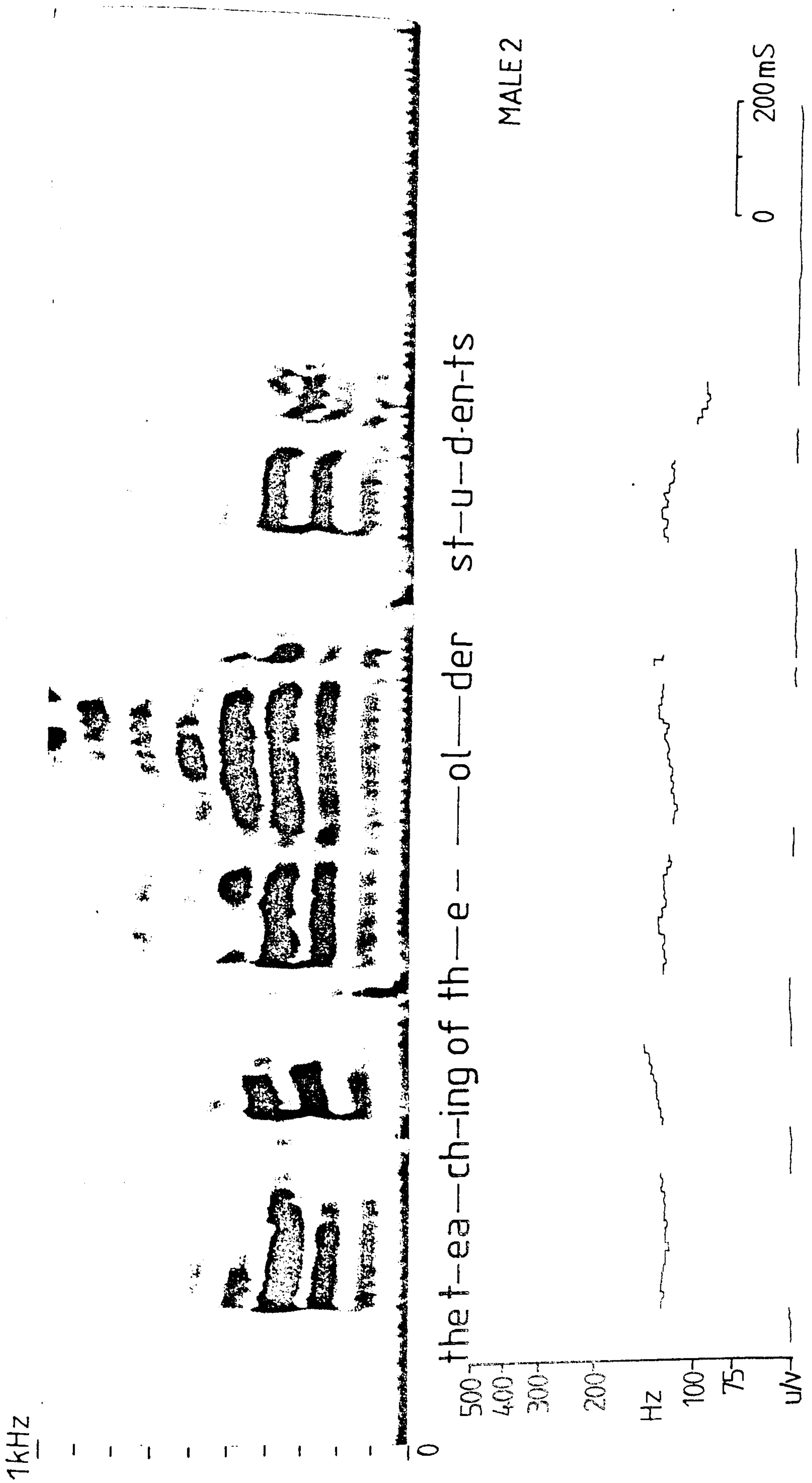


Figure 7-2: ERSA test results (contd.)

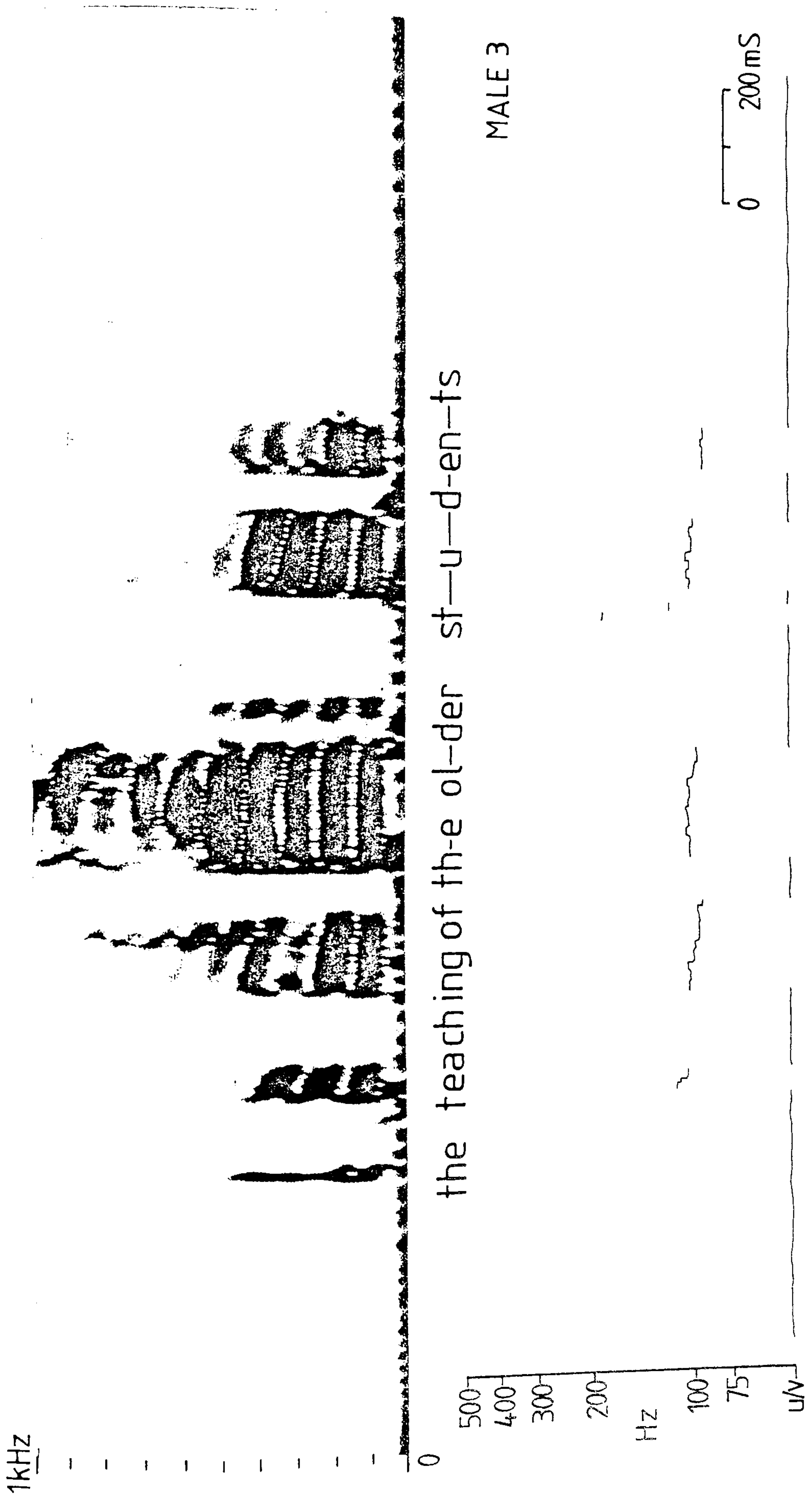


Figure 7-3: ERSA test results (contd.)

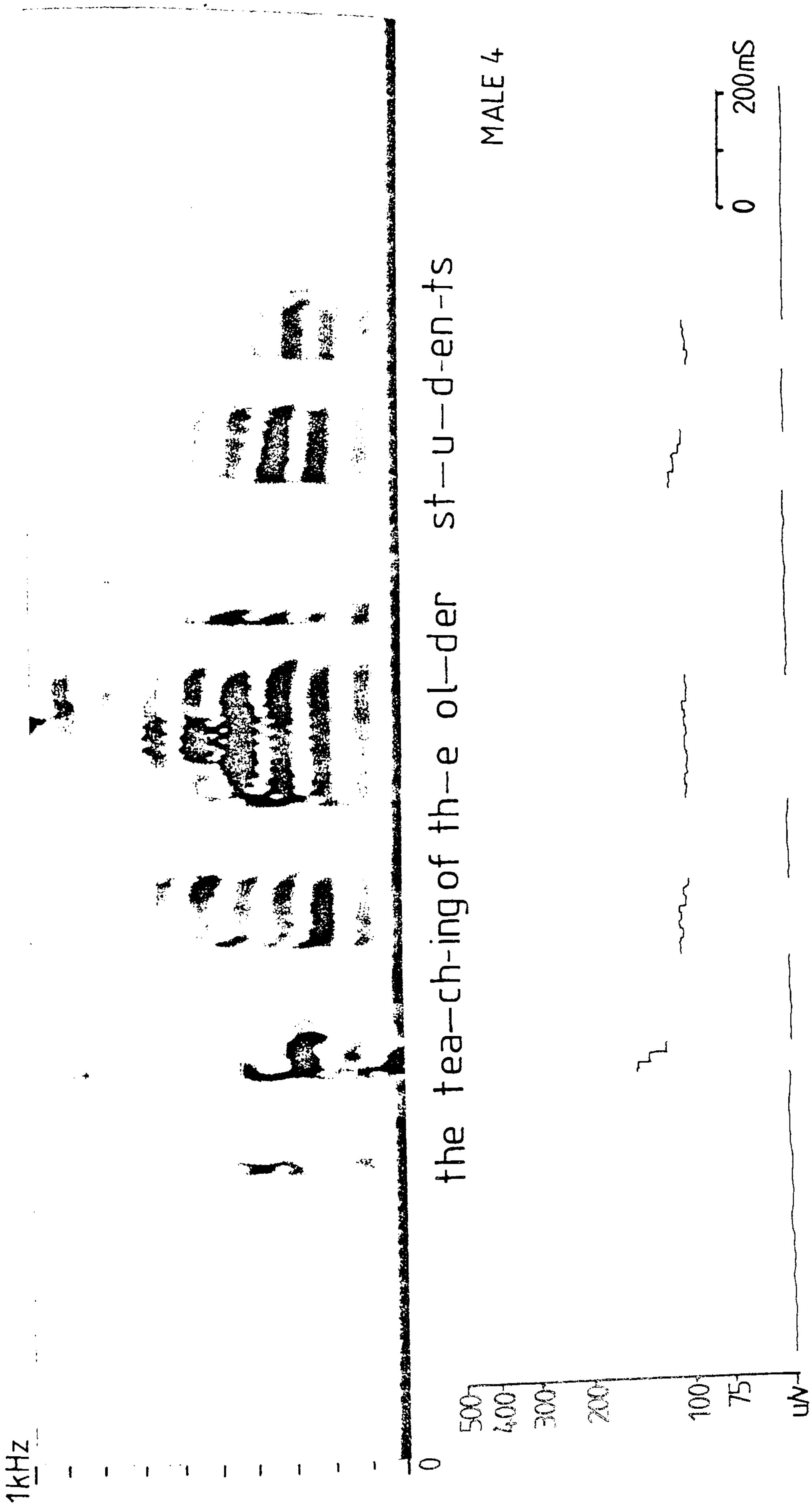


Figure 7-4: ERSA test results (contd.)

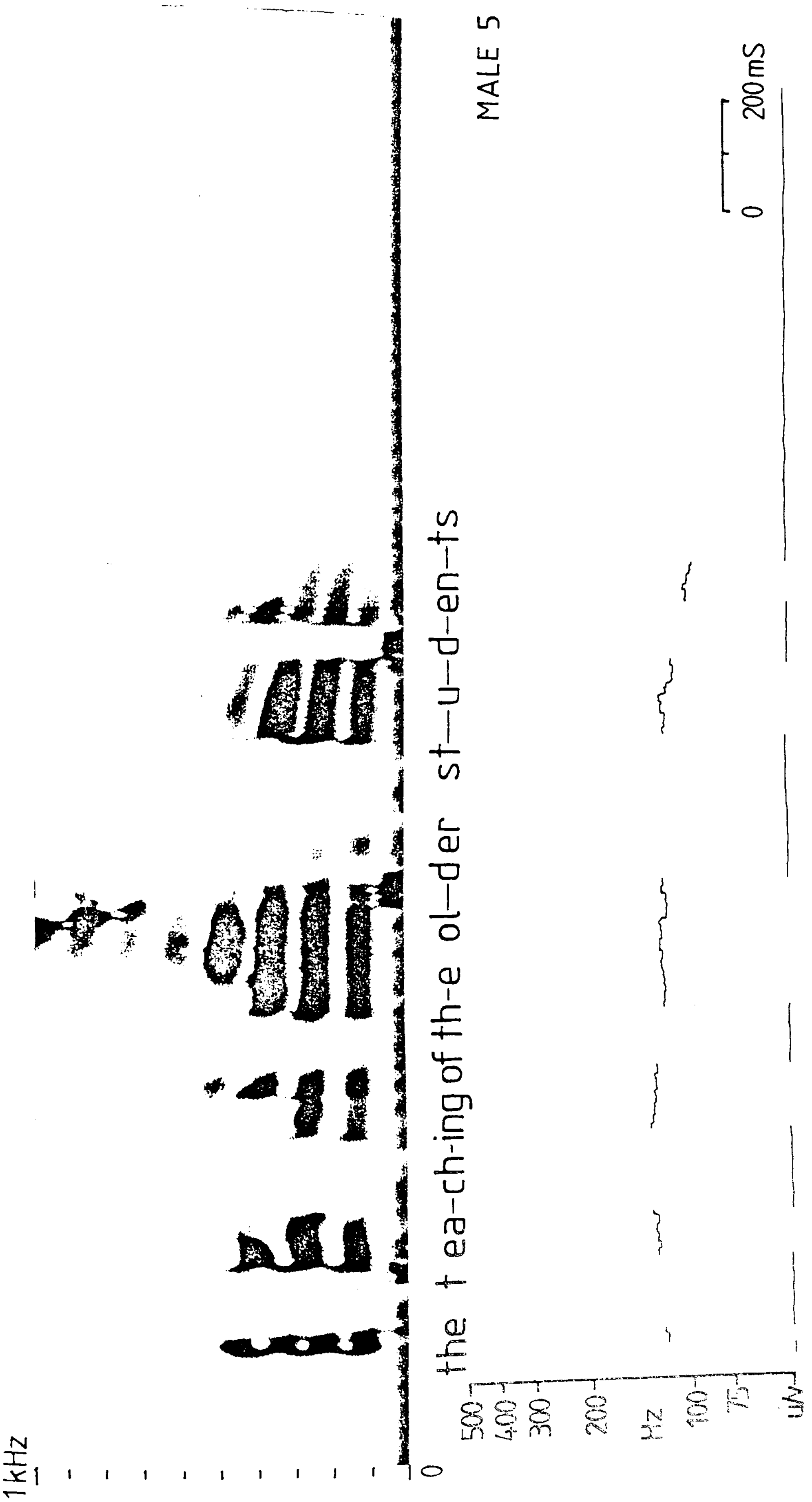


Figure 7-5: ERSA test results (contd.)

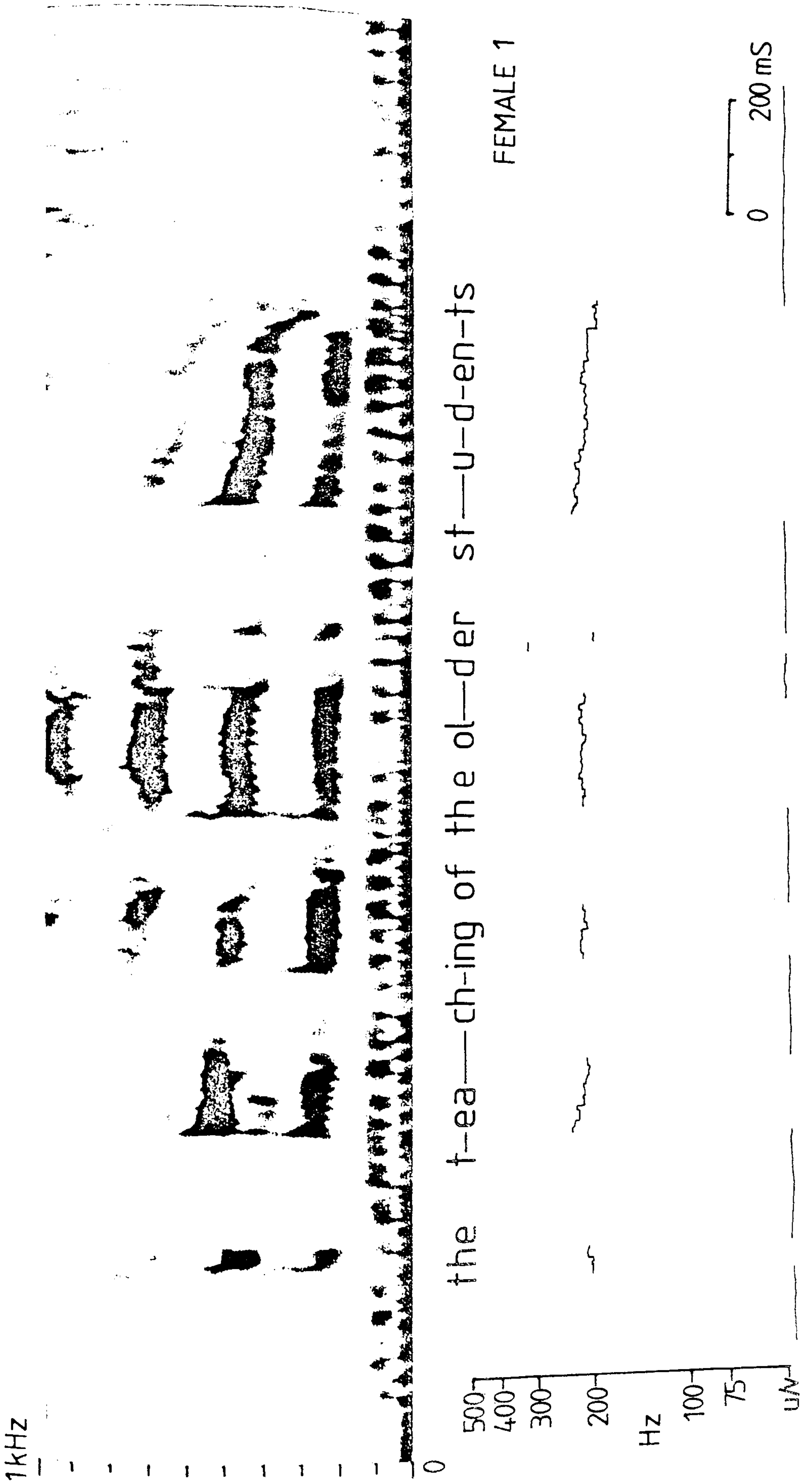


Figure 7-6: ERSA test results (contd.)

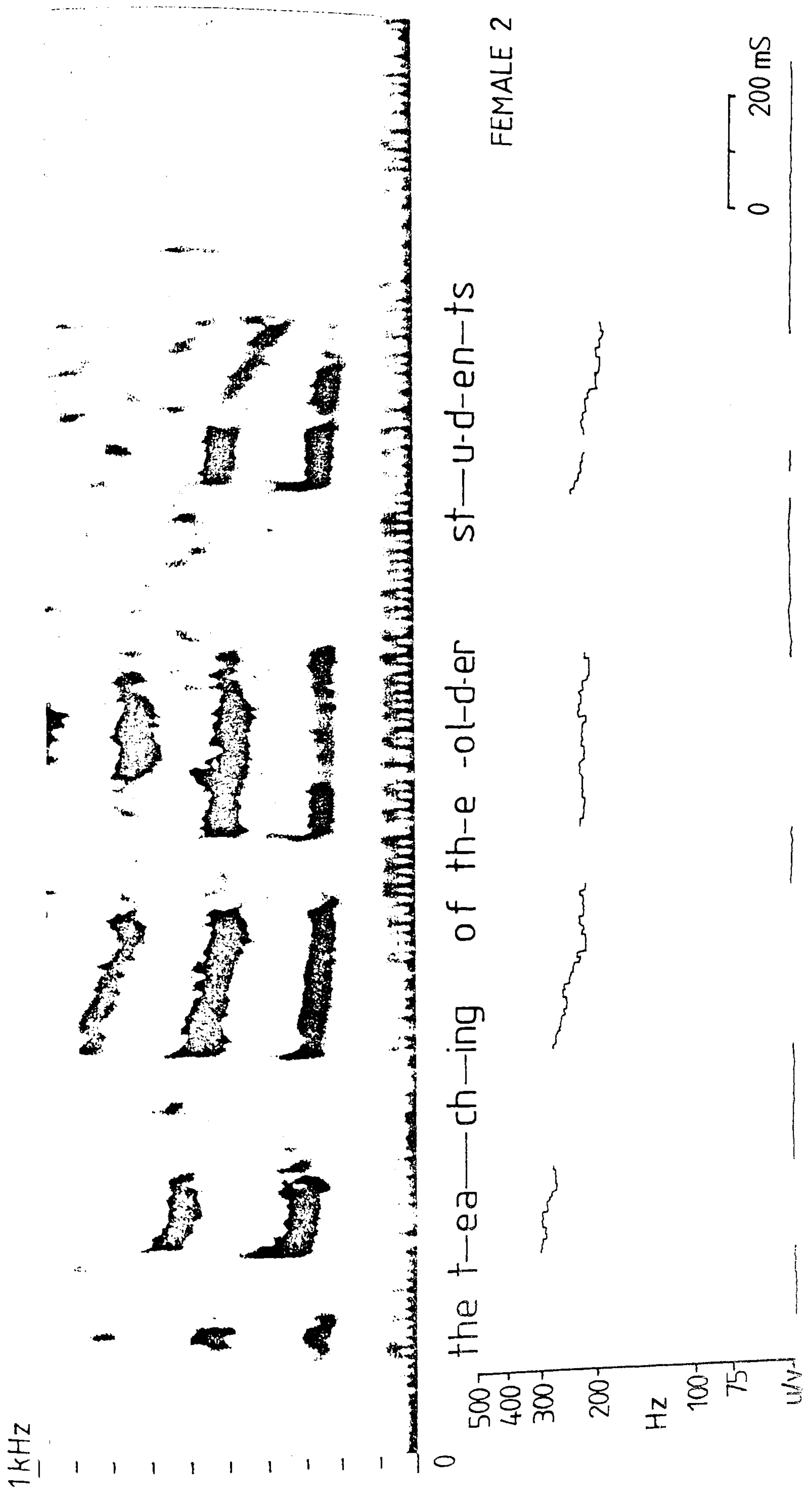


Figure 7-7: ERSA test results (contd.)

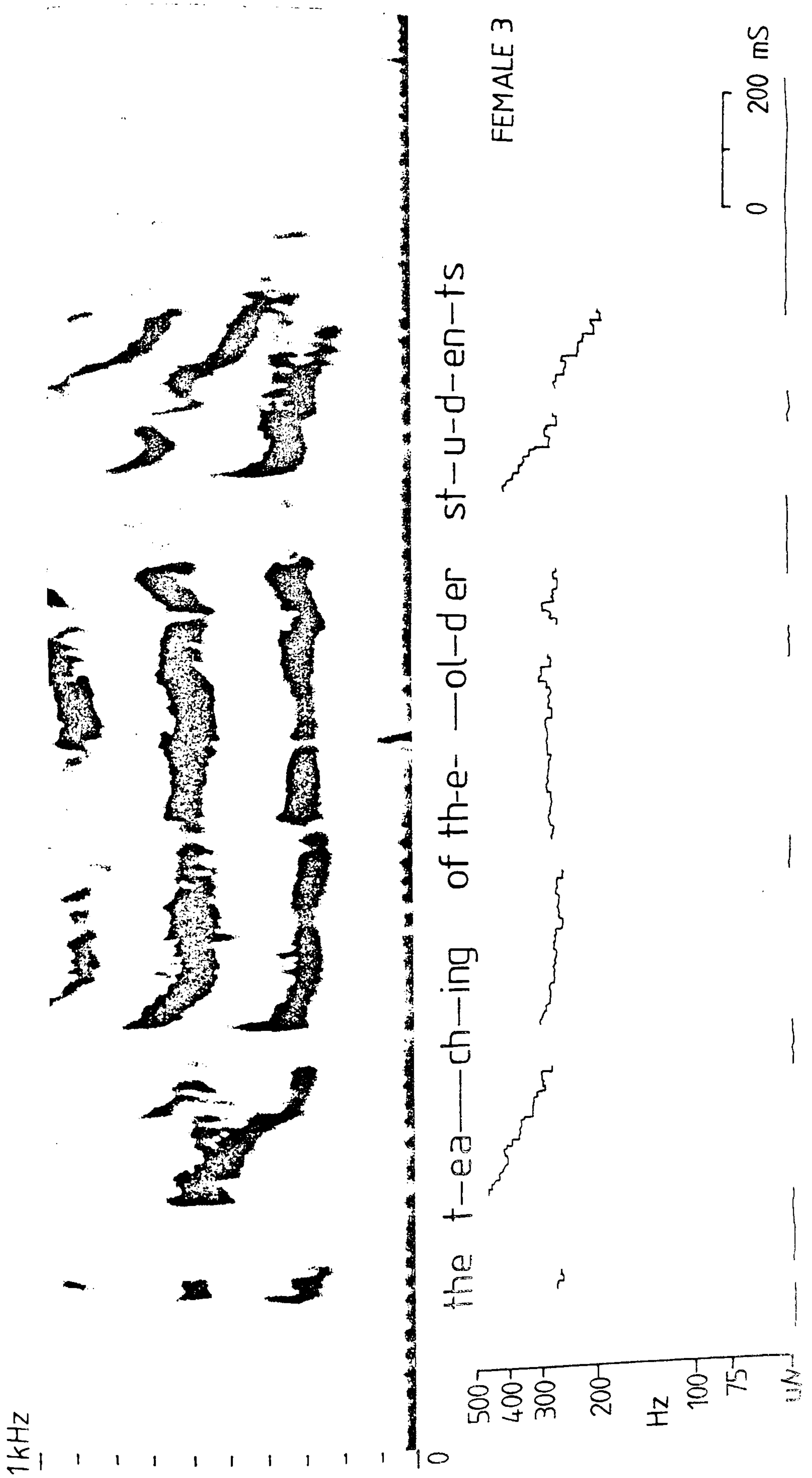


Figure 7-8: ERSA test results (contd.)

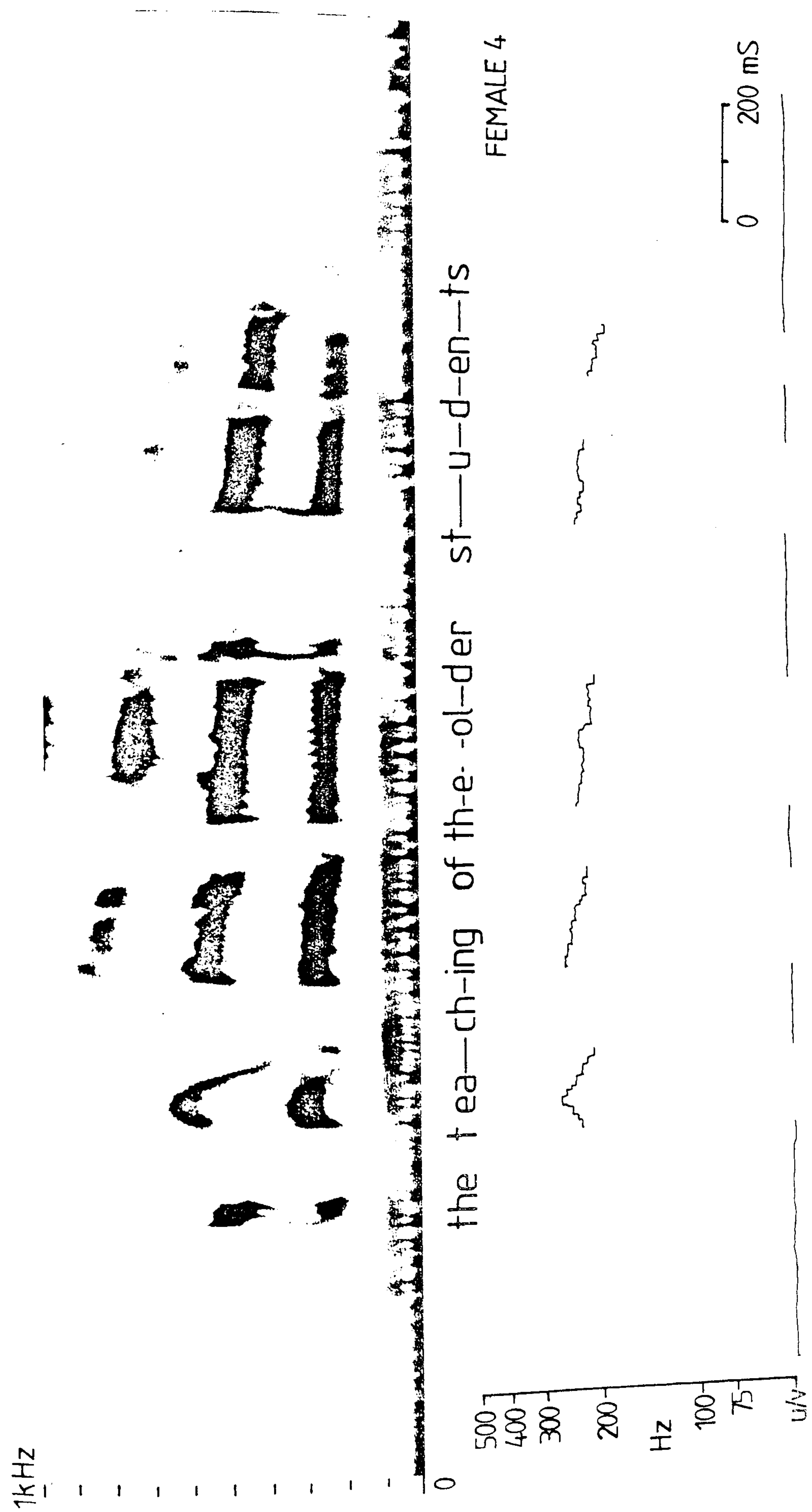


Figure 7-9: ERSA test results (contd.)

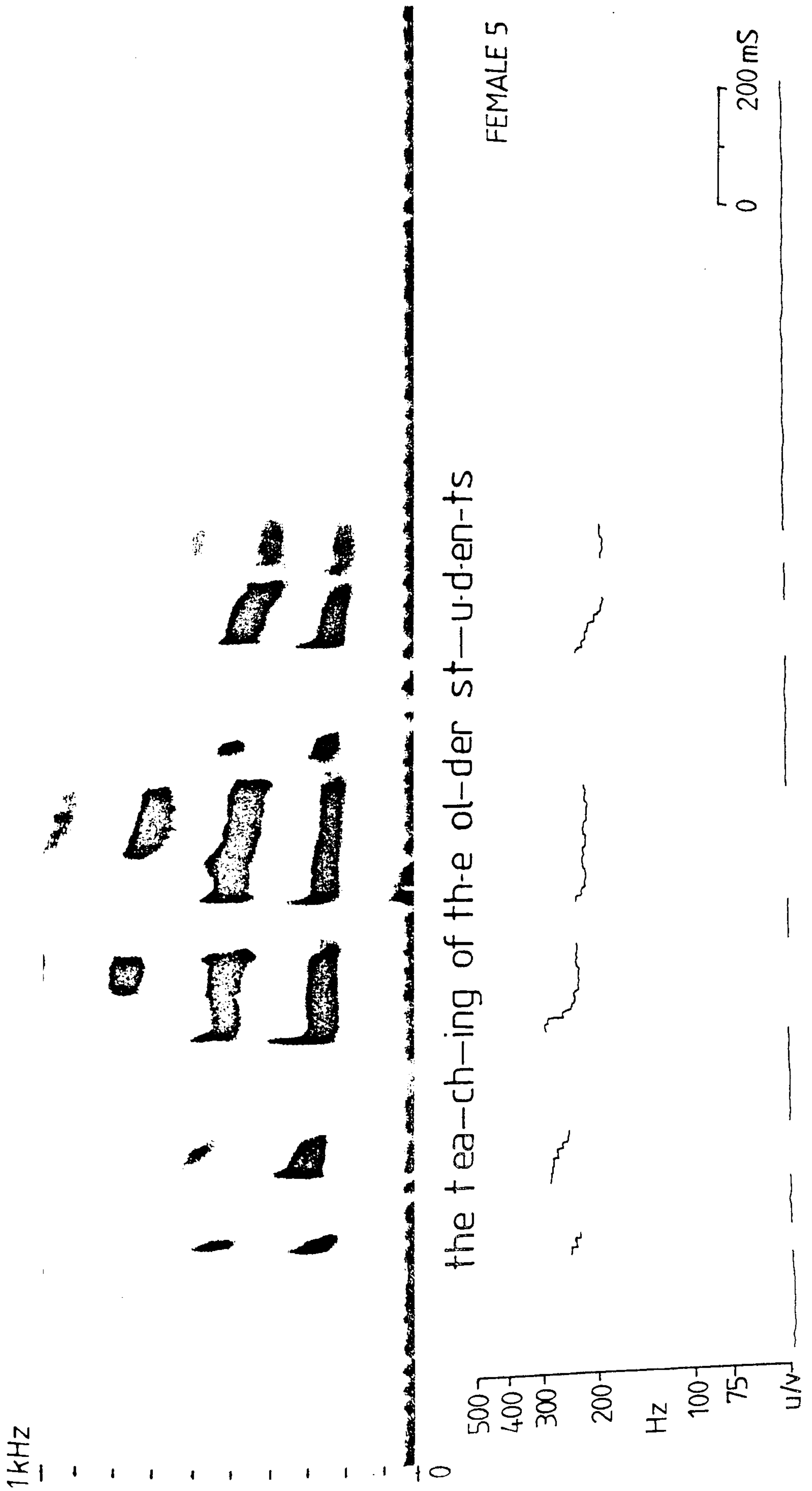


Figure 7-10: ERSA test results (contd.)

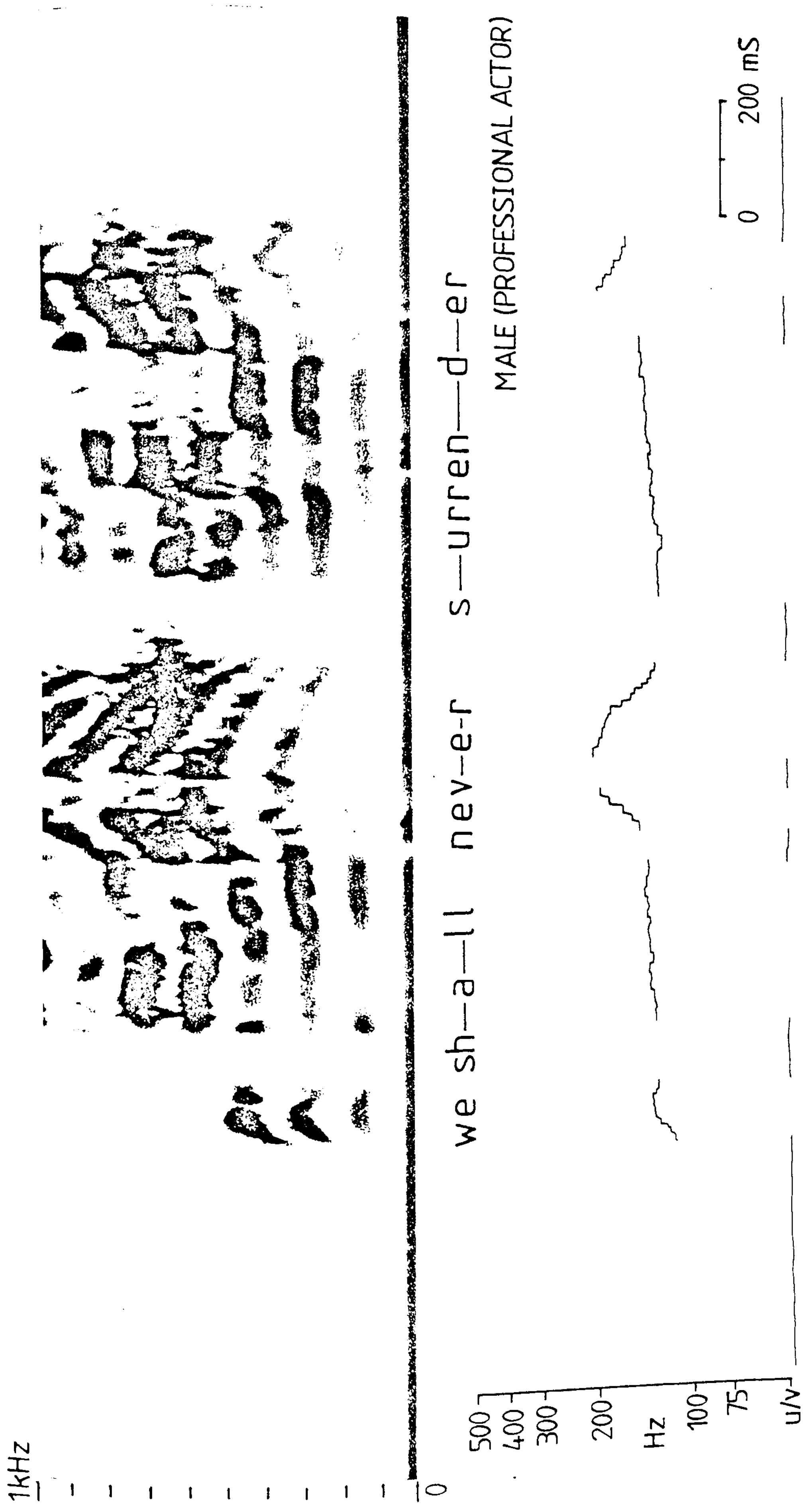


Figure 7-11: ERSA test results (contd.)

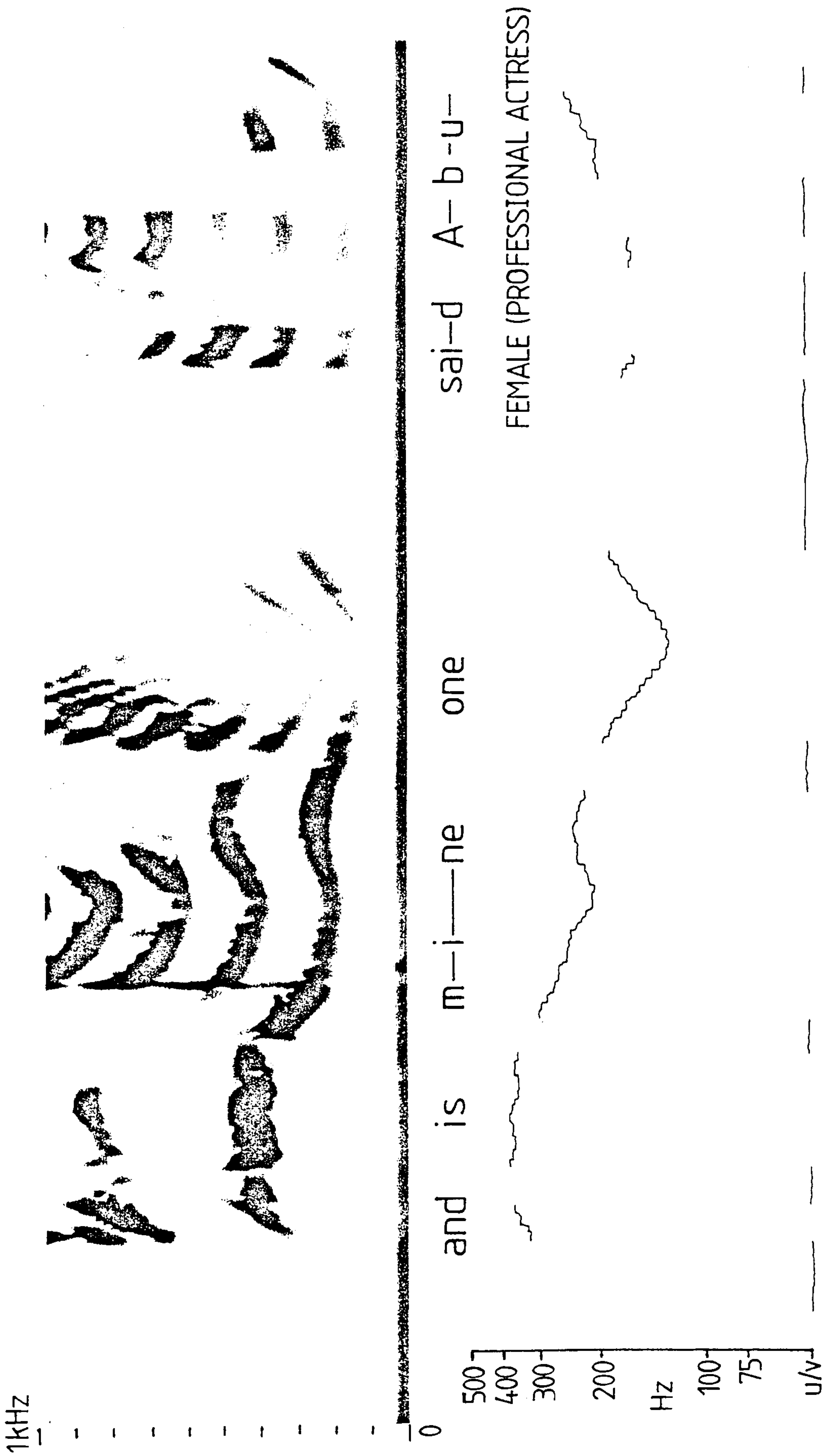


Figure 7-12: ERSA test results (contd.)

CHAPTER EIGHT

CONCLUDING REMARKS

8.1 FURTHER USES OF 'EXTRANEOUS ENTRY PREDICTION'

The ERSA level 3 extraneous entry prediction process was developed in order to control the use of level 2 error correction procedures, and to overcome two potential cases of erroneous application of error correction: these cases are discussed in Section 6.7.2. The author feels that the extraneous entry prediction process has potential for further use, in particular with respect to the case illustrated in Figure 8-1, in which a section of voiced speech waveform has persistent extraneous entries. With ERSA as it stands at present, such a case will almost certainly lead to pitch-period halving (frequency doubling), since the extraneous entry prediction process is only invoked in the case of a suspected hole or extra marker. With a waveform such as that shown in Figure 8-1, it might be possible to use the extraneous entry prediction to increase the integral threshold level, thereby increasing the chance of insignificant excursion cycles being excluded, or at least to warn the user of the possibility of frequency doubling. Such additional processing might, however, be impossible in real time on the 6502 microprocessor, especially in the case of a section of speech with high average fundamental frequency, which leads to a large number of entries in the data structure.

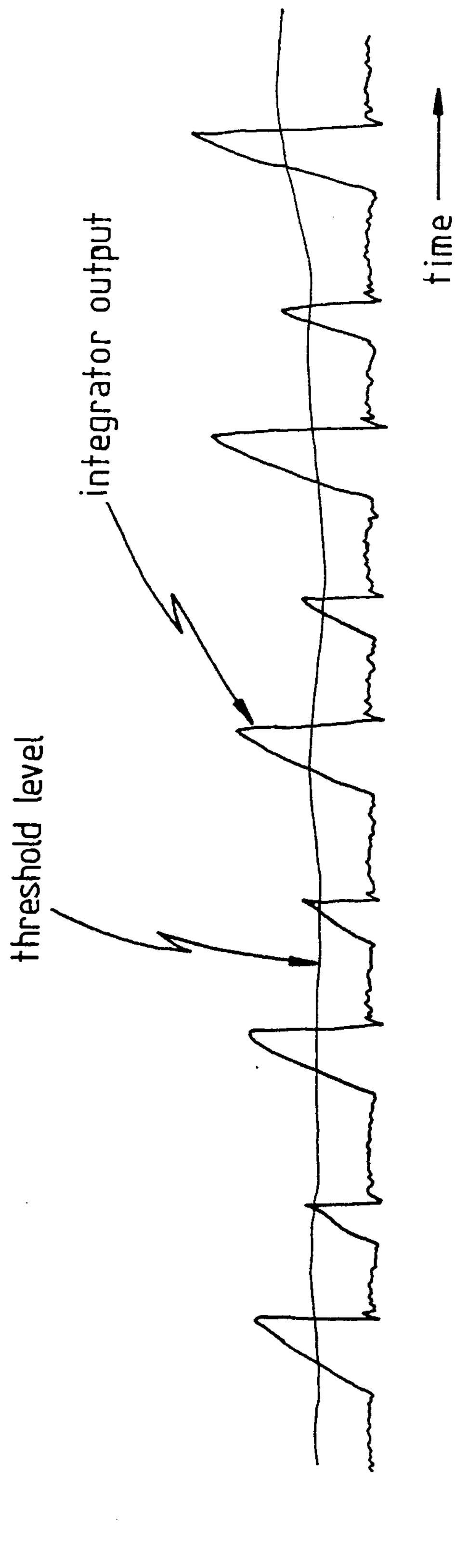


Figure 8-1: Section of voiced speech waveform with persistent extraneous entries

8.2 ALTERNATIVE MICROPROCESSORS

Since the time of development of the present project, a number of new microprocessors have become available.

There are now signal-processing microprocessors such as the Intel 2920 [H801], which contains an ADC and DAC on-chip, and has a combined digital/analogue instruction set. The use of such a microprocessor would, no doubt, simplify the design of a system such as ERSA.

Sixteen-bit microprocessors are now readily available, and offer increased processing power compared with eight-bit devices. In particular, a sixteen-bit microprocessor would allow use of a sixteen-bit clock for excursion-cycle interval timing, which would overcome the 25.5mS interval limit experienced with the present version of ERSA. Sixteen-bit devices have already found use in the implementation of digital filters [N802] [N803].

8.3 EXPERIENCE WITH ERSA

In its present state, the project hardware and software has proved robust and easy to use, and appears to fulfil the requirements set out in Chapter One. It is hoped that ERSA will be of assistance in future teaching of, and investigation into, human communication.

APPENDIX ONE

$\overline{NMI}$ SERVICE ROUTINES

$\overline{\text{NMI}}$ service routine number one

```
time (µS)
  7      (NMI automatic sequence overhead)
  3      NMISR  PHA                 save accumulator on stack
  2             TYA
  3             PHA                 save Y-register on stack
  4             LDA ADCIN           load sample from input port
  2             LDY £$0
  6             STA (POINTER),Y     store it at next location
  2             CLC
  3             LDA POINTER
  2             ADC £$1
  3             STA POINTER         increment lsb of pointer
  3             LDA POINTER+1
  2             ADC £$0
  3             STA POINTER+1       transfer any carry to msb
  4             PLA
  2             TAY                 restore Y-register
  4             PLA                 restore accumulator
  6             RTI                 return from interrupt
------
 59 µS
------
```

$\overline{NMI}$ service routine number two

```
time (µS)
  7      (NMI automatic sequence overhead)
  3      NMISR  PHA                 save accumulator on stack
  2             TYA
  3             PHA                 save Y-register on stack
  4             LDA ADCIN           load sample from input port
  2             LDY £$0
  6             STA (POINTER),Y     store it at next location
  5             INC POINTER         increment lsb of pointer
  3             BNE RETURN
  5             INC POINTER+1       increment msb of pointer
  4      RETURN PLA
  2             TAY                 restore Y-register
  4             PLA                 restore accumulator
  6             RTI                 return from interrupt
------
 51 µS    (best case)
 56 µS    (worst case)
------
```

$\overline{\text{NMI}}$ service routine number three

```
time (µS)
   7      (NMI automatic sequence overhead)
   3      NMISR  PHA                save accumulator on stack
   2             TYA
   3             PHA                save Y-register on stack
   4             LDA  ADCIN         load sample from input port
   3             LDY  OFFSET
   5             STA  POINTER,Y     store it at next location
   2             INY
   3             STY  OFFSET        increment offset
   3             BNE  RETURN
   6             INC  POINTER+1     increment pointer value
   4      RETURN PLA
   2             TAY                restore Y-register
   4             PLA                restore accumulator
   6             RTI
 ------
  51 µS      (best case)
  57 µS      (worst case)
 ------
```

APPENDIX TWO

CLIPPED AUTOCORRELATION ROUTINE

A2.1 ASSEMBLY LANGUAGE LISTING

```
!***********************************
!Clipped Autocorrelation Routine
!
!Version AUTOC5
!***********************************
!
!
! Zero page reserved bytes
!
              *=$B2
LAGS          *=*+1   autocorrelation lag counter
SUM           *=*+1   autocorrelation variable
TEMP          *=*+1   temporary storage byte
TEMPY         *=*+1   temporary storage byte
PKVAL         *=*+1   autocorrelation variable
PKNUM         *=*+1   autocorrelation variable
START         *=*+2   start address of current segment
CONTIN        *=*+1   continuation frame flag
CPK1          *=*+1   clipping peak - initial interval
CPK2          *=*+1   clipping peak - final interval
CLIPP         *=*+1   positive clipping threshold
CLIPN         *=*+1   negative clipping threshold
LIM1          *=*+1   upper limit of initial interval
LIM2          *=*+1   upper limit of final interval
TEMPX         *=*+1   temporary storage byte
ADDR          *=*+1   NMI service routine storage address
!
! Various equates
!
USRPRT        = $E841   PET user port address
CSTART        = $7400   start of clipped waveform area
PTR1          = $7380   pointer value for frame carry-over
PTR2          = $6F80   pointer value for frame carry-over
!
              *=$6000
SATAB         .WORD $6F80   start address table
              .WORD $7000
              .WORD $7080
              .WORD $7100
              .WORD $7180
              .WORD $7200
              .WORD $7280
              .WORD $7300
PKNUMS        *=*+8   peak numbers for 8 segments
PKVALS        *=*+8   peak values for 8 segments
ZLAGS         *=*+8   zero-lag values for 8 segments
!
! =========================================================
!
```

```
! NMI service routine
!
              *=$6100   (=24832 decimal)
NMISR         PHA
              TXA
              PHA
              TYA
              PHA
              LDA USPRT  load sample from user port
              EOR £$80     convert to 2's complement
              LDY £$00
              STA (ADDR),Y  store sample in next free location
              CLC
              LDA ADDR
              ADC £$01
              STA ADDR  increment lsb of ADDR
              LDA ADDR+1
              ADC £$00
              STA ADDR+1  add carry (if any) to msb of ADDR
RETURN        PLA
              TAY
              PLA
              TAX
              PLA
              RTI
!
! =======================================================
!
! Get samples from storage scope
!
              *=$6180   (=24960 decimal)
GET           LDA £$00
              STA ADDR  initialise lsb of ADDR
              LDA £$70
              STA ADDR+1  initialise msb of ADDR
LOOP          LDA ADDR+1  load msb of ADDR
              CMP £$74  end of input?
              BNE LOOP  no - continue waiting
              CLI  yes - return
              RTS
!
! =======================================================
!
! Autocorrelation routine
!
              *=$6200   (=25088 decimal)
AUTOC         SEI
              LDA £$00
              STA TEMPX  initialise segment counter
              LDA CONTIN  is this a continuation frame?
              BNE A1  yes - proceed
              INC TEMPX  no - ignore first segment
              INC TEMPX
A1            LDX TEMPX
              TXA
```

```
              CMP £$10   end of this frame?
              BNE A3   no - proceed
!
!  End of autocorrelation for this frame.
!  Move final 128 bytes for carry-over to next frame.
!
              LDX £$00
A2            LDA PTR1,X   load a sample
              STA PTR2,X   store in continuation area
              INX
              TXA
              CMP £$80   end of carry-over?
              BNE A2   no - repeat
              LDA £$FF
              STA CONTIN   set continuation frame flag
              CLI
              RTS   return
!
!
A3            LDA SATAB,X
              STA START
              INX
              LDA SATAB,X
              STA START+1   START = start address of seg
              INX
              STX TEMPX
!
! ==========================================================
!
! Clipping routine
!
              LDA £$00
              STA CPK1   initialise peak variable
              STA CPK2   initialise peak variable
              TAX
              TAY
C1            LDA (START),Y   load sample from segment
              ASL A   shift msb into carry bit
              BCS C2   negative value
              ROR A   restore sample
              JMP C3
!
! Process negative sample
!
C2            ROR A   restore sample
              EOR £$FF
              CLC
              ADC £$01   take 2's complement
!
! Find peak values for the initial and final intervals.
! Each interval is 85 samples long.
! The X-register determines whether the initial or final
! interval is being processed.
C3            CMP CPK1,X   is sample > current peak?
              BMI C4   no
```

```
               STA CPK1,X  yes - amend current peak value
      C4       INY
               TYA
               CMP LIM1,X  end of interval reached?
               BNE C1  no - go back and get next sample
               INX  yes
               LDY £$AB  start of final interval for this segment
               TXA
               CMP £$02  have both intervals been processed?
               BNE C1  no - go back and continue
    !
    ! CPK1 and CPK2 have now been evaluated
    !
    ! Determine the lower of the two peak values
    !
               LDA CPK1
               CMP CPK2
               BMI C5  CPK1 < CPK2
               LDA CPK2  CPK2 < CPK1
      C5       LSR A  divide by 2
               STA CLIPP  set positive clipping threshold
               EOR £$FF
               CLC
               ADC £$01  take 2's complement
               STA CLIPN  set negative clipping threshold
    !
    ! Clip waveform
    !
               LDY £$00
      C6       LDA (START),Y  load sample
               ASL A  examine msb of sample
               BCS C9  negative sample
               ROR A  positive sample - restore it
               CMP CLIPP  compare it with positive threshold
               BMI C8  sample < positive threshold
      C7       LDA £$41  sample >= positive threshold
               JMP C10
      C8       LDA £$00  sample is between thresholds
               JMP C10
      C9       ROR A  restore negative sample
               CMP CLIPN  compare with negative threshold
               BMI C8  sample > negative threshold
               LDA £$81  sample <= negative threshold
      C10      STA CSTART,Y  store clipped value
               INY
               TYA  end of segment?
               BNE C6  no - go back and get next sample
    !
    ! Clipping complete ...
    ! Autocorrelation follows.
    !
               LDA £$19  initial autocorrelation lag of 25 samples
               STA LAGS
               LDA £$00
               STA PKVAL  initialise PKVAL
```

```
        STA PKNUM   initialise PKNUM
M1      LDA £$00
        STA SUM   initialise SUM
        LDA £$FF
        SEC
        SBC LAGS
        TAY
M2      CLC
        LDA CSTART,Y   get clipped sample for autocorr,
        STA TEMP         and store it in TEMP
        TYA
        STA TEMPY
        ADC LAGS
        TAY
        LDA CSTART,Y   get the other clipped sample
        AND TEMP   AND the two clipped samples
        ROR A
        BCC M4   one sample was 0: leave SUM as it is
        BEQ M3   opposite polarity: decrement SUM
        INC SUM   peaks of same polarity: increment SUM
        JMP M4
M3      LDA SUM   is SUM > 0?
        BEQ M4   no - skip
        DEC SUM   yes - decrement it
M4      LDY TEMPY
        DEY   end of autocorrelation for this lag value?
        BNE M2   no - go back and continue
        LDA SUM   yes
        CMP PKVAL   is SUM > current peak value?
        BCC M5   no
        STA PKVAL yes - set current peak value to SUM
        LDA LAGS
        STA PKNUM   set current peak number to LAGS
M5      INC LAGS   increment autocorrelation lag value
        LDA LAGS
        CMP £$C9   final lag value reached?
        BNE M1   not yet - go back and continue
!
! Evaluate zero-lag autocorrelation value
!
        LDA £$00
        STA SUM   initialise SUM to zero
        STA LAGS   initial lag value for autocorrelation
M6      LDY LAGS
        LDA CSTART,Y   get clipped sample
        BEQ M7   not a peak
        INC SUM   peak: so increment SUM
M7      INC LAGS
        LDA LAGS
        CMP £$FF   end of segment?
        BNE M6   no - go back and continue
!
! Store results
!
        LDA TEMPX   calculate curr. seg. no. from TEMPX
```

```
LSR A
TAX
DEX
LDA PKNUM
STA PKNUMS,X   store peak number
LDA PKVAL
STA PKVALS,X   store peak value
LDA SUM
STA ZLAGS,X   store zero-lag value
JMP A1   continue processing
.END
```

A2.2 BASIC PROGRAM LISTING

```
10 POKE 52,0: POKE 53,80: PRINT "Memory limited to $5000"
20 REM  the above steps ensure that PET does not
30 REM  overwrite the machine code program.
40 REM
50 REM  **************************
60 REM  CLIPPED AUTOCORRELATION
70 REM
80 REM  Runs routine AUTOC5
90 REM  **************************
100 REM
110 POKE 186,0: REM clear CONTIN flag
120 POKE 191,85: REM set LIM1
130 POKE 192,0:  REM set LIM2
140 REM  configure PET
150 POKE 59459,0
160 POKE 59467,PEEK(59467) OR 1
170 REM  set NMI vector address
180 POKE 148,0: POKE 149,97
190 REM
200 PRINT "Ready for input from scope:"
210 PRINT "Press START on 4002 data output option"
220 REM  get data from scope
230 SYS 24960
240 PRINT "OK - commencing autocorrelation"
250 REM  clip and autocorrelate data
260 SYS 25088
270 PRINT "End of autocorrelation"
280 PRINT "Results....."
290 FOR I=1 TO 7
300 PRINT "SEGMENT ";I;"...PERIOD=";PEEK(24592+I)/10
310 PRINT "FREQUENCY=";
320 REM  check for division by zero
330 IF PEEK(24592+I)=0 THEN PRINT " ZERO": GOTO 350
340 PRINT 1000/PEEK(24592+I)
350 PRINT "PEAK VAL=";PEEK(24600+I);"IN SAMPLE ";PEEK(24592+I)
360 PRINT PEEK(24600+I)/PEEK(24608+I)*100;"% OF ZLAG VALUE ";
370 PRINT PEEK(24608+I)
380 PRINT
390 NEXT I
400 PRINT "----------------------"
410 STOP
```

A2.3 DESCRIPTION OF CLIPPED AUTOCORRELATION SYSTEM

A2.3.1 Description of the low-level software

This routine is similar in operation to the 'real-time digital hardware pitch detector' of Dubnowski et al [D334], and is designed to compute the autocorrelation function for seven 256-sample segments of centre- and peak-clipped speech waveform which are derived from a 1024-sample 'frame' of speech waveform samples fed into the PET microcomputer from the OS4000 storage oscilloscope using the OS4002 data output option. The oscilloscope timebase control is set so that the 1024 samples represent approximately 100mS of speech, giving an effective sampling rate of some 10kHz.

The structure of the 1024-sample frame and the seven 256-sample segments is illustrated in Figure A2-1. It will be seen that the segments overlap by 128 samples. Since the effective sampling rate is approximately 10kHz, each segment represents some 25.6mS of speech, and the segments are spaced at approximately 12.8mS intervals. (By comparison, Dubnowski's system uses overlapping 30mS segements spaced at 10mS intervals).

It will be seen from Figure A2-1 that an eighth segment, Segment 0, is shown, which carries over 128 samples from a previous frame. This facility was included to allow pseudo-continuous analysis of a speech signal by capturing successive 1024-sample

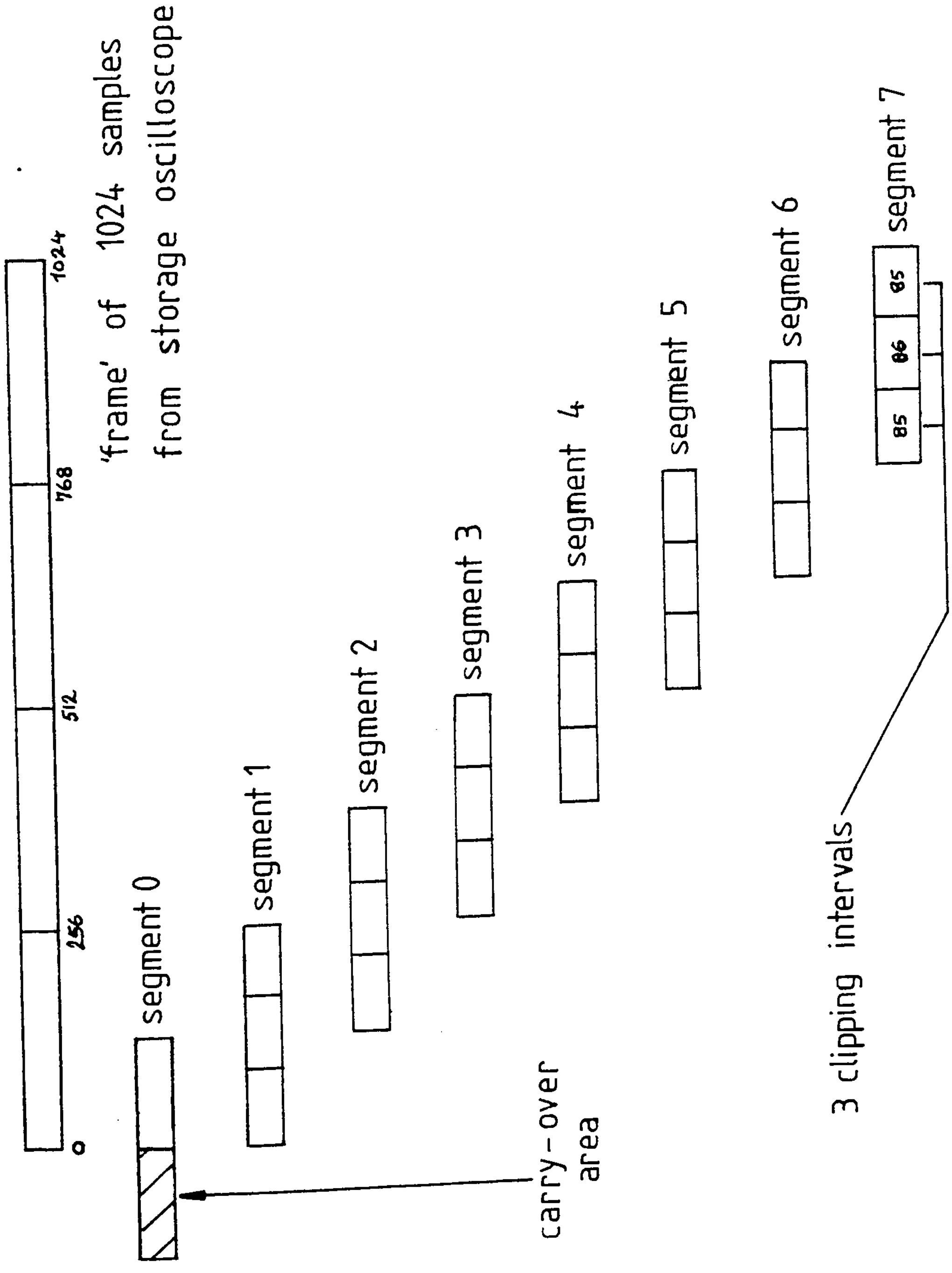


Figure A2-1: Clipped autocorrelation frame structure frames on the storage oscilloscope. At the end of each analysis phase, the final 128 bytes are 'carried over' to form the first 128 bytes of the next frame. This arrangement allows eight overlapping segments to be analysed in each 1024-sample frame. Some hardware was constructed which was designed to trigger the storage oscilloscope at predetermined points in a tape-recorded utterance, so that the signal could be fed into the microcomputer frame-by-frame. However, it was found that the oscilloscope timebase was slightly inaccurate and somewhat unstable, and it was consequently impossible to ensure that the frames carried on from one another without overlapping or missing small sections of the signal. Facilities for processing the 'continuation' frames have been left in the software, but in practice each frame is treated as an 'initial' frame, and only seven segments are formed and analysed. The CONTIN flag (line 18 of the low-level code) is set to zero to show that there is no 'carried over' half segment.

The first stage of the low-level software is the subroutine GET (line 81), which uses the $\overline{\text{NMI}}$ service routine NMISR (line 53) to load samples from the storage oscilloscope and store them in consecutive locations. Each sample is represented by one 8-bit byte.

The clipping routine (line 136) is based upon Dubnowski's approach. Each segment is split into three 'intervals', one initial and one final interval of 85 bytes each, and a centre interval of 86 bytes. The initial and final intervals are examined to locate the peak values in each. Once the peak values have been identified (line 175), the lower of the two is selected, and the clipping level is set to $\pm 50\%$ of this value. The clipping level chosen in Dubnowski's system is a variable parameter, but he reports that a level of $\pm 80\%$ of the lower peak gave good results. The 50% level chosen for this implementation was found to work well, and is easy to calculate.

The waveform is clipped (line 188), so that each sample takes on one of three values:

\$41 if the sample is greater than or equal to the positive clipping threshold,

\$81 if the sample is less than the negative clipping threshold,

\$00 if the sample is between the two thresholds.

(the '\$' indicates that these values are in hexadecimal notation).

The hexadecimal codes used - \$41, \$81, \$00 - are chosen to increase the efficiency of the autocorrelation routine, since the bit patterns are easy to test by simple arithmetic shifts.

Once the segment has been clipped, the autocorrelation function is computed as in Dubnowski's system:

$$R_x(m) = \sum_{n=0}^{255-m} x(n)x(n+m) \qquad 25 <= m <= 200 \qquad \text{...(A2.1)}$$

The range of lag values - 25 to 200 - is chosen to give a frequency range of 400Hz down to 50Hz. In addition, the zero lag autocorrelation is computed. Once autocorrelation has been carried out over the full range of lag values, the lag number containing the autocorrelation peak, the peak value itself, and the zero lag value are saved (line 269).

A2.3.2 Description of the high-level software

The BASIC program which accompanies the low-level software is designed to interact with the user, and displays the results of the autocorrelation process for the seven segments. For each segment, the pitch-period, fundamental frequency, lag value of autocorrelation peak, peak value, and percentage of zero lag value are displayed. The last result, the autocorrelation peak value expressed as a percentage of the zero lag value, can be used to make a voiced/unvoiced decision. Dubnowski suggests that a percentage value in excess of 30% indicates a voiced segment, lower percentage values indicating unvoiced segments. The present author has found this approach to be unreliable in some cases, as some waveform segments which are clearly voiced can give percentage values of less than 30%.

APPENDIX THREE

THE COUNTER-RAMP ADC

The basic counter-ramp ADC is illustrated in Figure A3-1, and contains a digital-to-analogue converter (DAC), a resettable binary counter, and a comparator. The binary counter is initialised to an all-zeroes state by the 'request conversion' line. A pulse train is fed into the clock line, each pulse incrementing the value of the binary counter. The output of the latter is fed to a DAC, whose output voltage, Vd, proportional to the binary value of the input word, is compared with the analogue input, Vi. When Vd exceeds Vi, the output of the comparator is low, disabling the AND gate and removing the clock source. The output of the binary counter, which represents the number of clock pulses received between system reset and the instant when Vd > Vi, is proportional to the value of Vi. The relationship between Vd, Vi and the clock pulse train is illustrated in Figure A3-2.

The time taken for Vd to acquire a potential just in excess of that of Vi, the ADC ACQUISITION TIME, depends upon the magnitude of Vi, and upon the clock period. If Vi is represented by 'n' clock pulses, and if the clock period is 'T' seconds, the acquisition time is nT seconds. The CONVERSION TIME of the ADC, which depends upon the acquisition time, is limited by the need to reset the counter to its all-zeroes condition prior to each conversion.

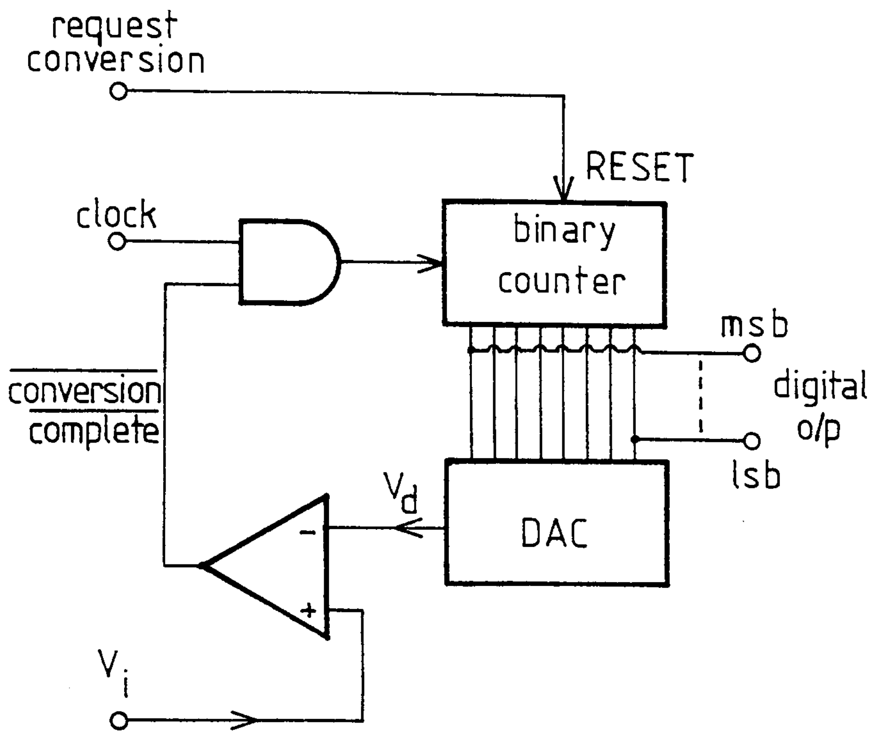


Figure A3-1: Counter-ramp ADC circuit

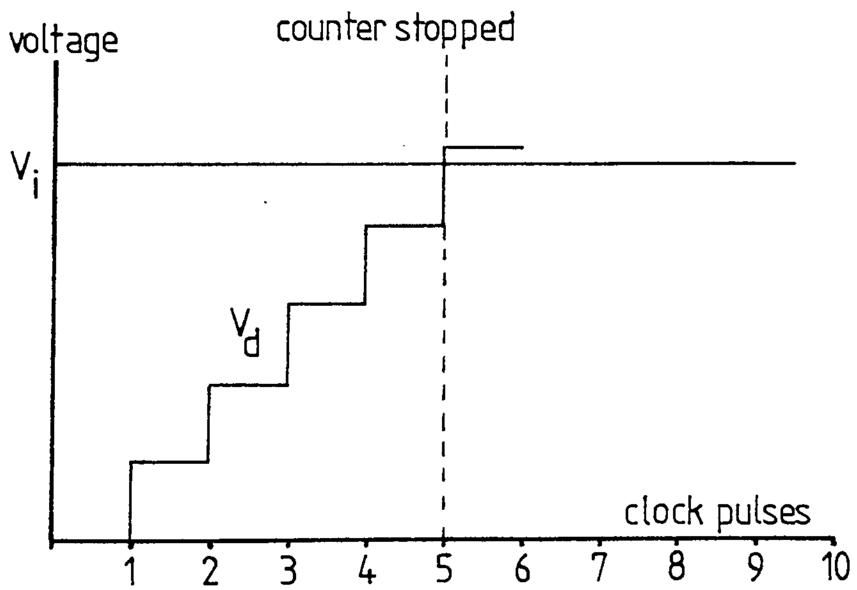


Figure A3-2: Timing diagram for Figure A3-1

APPENDIX FOUR

LISTING OF THE LOW-LEVEL SOFTWARE FOR THE PROJECT

```
! ************************************************
! Educational Research Speech Analysis System
!
!       ERSA LEVEL 3                VERSION 3.2
!
! J.R.Runchman, University of Aston in Birmingham
! ************************************************
!
!
          *=$00   set program counter to $0000
!
! Equates
!
DSSTRT  =$3C00   start address of data structure
OPSTRT  =$3D00   start address of output area
CONTAB  =$3E00   start of period -> freq table
ZPSTRT  =$3F00   start of zero page copy area
PIA1A   =$A810   PIA1 data register A
PIA1B   =$A812   PIA1 data register B
PIA1CA  =$A811   PIA1 control register A
PIA1CB  =$A813   PIA1 control register B
PIA2B   =$A822   PIA2 data register B
PIA2CB  =$A823   PIA2 control register B
KBDROW  =$E812   PET keyboard register
WRTWO   =$E784   PET routine - write 2 bytes
SPAC2   =$FDCA   PET routine - write 2 spaces
WROB    =$E775   PET routine - write 1 byte
SYSINT  =$E813   PET interrupt control register
ESTIML  =$01     lsb of ESTIM routine
ESTIMH  =$30     msb of ESTIM routine
!
! Reserved bytes
!
P3OFFS  *=*+1   P3 segment offset from DSSTRT
P2OFFS  *=*+1   P2 segment offset from DSSTRT
P1OFFS  *=*+1   P1 segment offset from DSSTRT
COFFS   *=*+1   core segment offset from DSSTRT
S1OFFS  *=*+1   S1 segment offset from DSSTRT
S2OFFS  *=*+1   S2 segment offset from DSSTRT
S3OFFS  *=*+1   S3 segment offset from DSSTRT
UOFFS   *=*+1   update segment offset from DSSTRT
IDLE    *=*+1   idle time during processing
THISPP  *=*+1   current pitch-period estimate
PREVPP  *=*+1   previous pitch-period estimate
COUNT   *=*+1   count value from preprocessor
INTEG   *=*+1   integral value from preprocessor
UPDPTR  *=*+1   pointer to next space in update seg
OPPTR   *=*+1   pointer to next space in output area
ERRNUM  *=*+1   error number
RELERR  *=*+1   relative error value
DIV2    *=*+1   PREVPP/2
DIV8    *=*+1   PREVPP/8
INTLIS  *=*+$31   inter-entry comparison list 1
CNTLIS  *=*+$31   inter-entry comparison list 2
EEPRED  *=*+1   extraneous entry prediction
```

```
INEST   *=*+1   flag: ESTIM routine in progress
INIRQ   *=*+1   flag: IRQSR routine in progress
NMIOCC  *=*+1   flag: NMI occurred during IRQSR
LT2MS   *=*+1   flag: first update count < 2mS
WAIT    *=*+1   flag: wait for next segment
ACCEPT  *=*+1   flag: current estimate is OK
!
! Reserved bytes for ESTIM routine
!
SEG     *=*+1
ENT     *=*+1
!
! Reserved bytes for IRQSR routine
!
ADDCNT  *=*+1
PREVC   *=*+1
S3PTR   *=*+1
UPLIM   *=*+1
!
! Reserved bytes for GETPRE routine
!
PSEG    *=*+1
PENT    *=*+1
PCOUNT  *=*+1
COROFF  *=*+1
PREOFF  *=*+1
PREFND  *=*+1
GETPA   *=*+1
!
! Reserved bytes for GETSUC routine
!
SSEG    *=*+1
SENT    *=*+1
SCOUNT  *=*+1
SUCOFF  *=*+1
SUCFND  *=*+1
GETSA   *=*+1
!
! Reserved byte for CALCRE routine
!
PPRED   *=*+1
!
! Reserved bytes for EXM routine
!
SUM1    *=*+1
SUM2    *=*+1
DIFF1   *=*+1
DIFF2   *=*+1
PPDIFF  *=*+1
EXMA    *=*+1
!
! Reserved bytes for HOLE routine
!
HOLEA   *=*+1
HOLEB   *=*+1
```

```
!
! Reserved byte for IEC routine
!
S3NENT   *=*+1
!
!Reserved bytes for COMP routine
!
CCOMP1   *=*+1
ICOMP1   *=*+1
CCOMP2   *=*+1
ICOMP2   *=*+1
EIGHTH   *=*+1
COMPHI   *=*+1
COMPLO   *=*+1
!
! Reserved bytes for ENTIEC routine
!
IECC     *=*+1
IECI     *=*+1
!
! Reserved bytes for EXTRA routine
!
LISPTR   *=*+1
SEGN     *=*+1
SAVEX    *=*+1
LISST    *=*+1
IPRED    *=*+1
CPRED    *=*+1
!
! Reserved bytes for AUTO routine
!
LAG0     *=*+1
LAG1     *=*+1
LAG2     *=*+1
LAG3     *=*+1
!
! Reserved bytes for ACOMP routine
!
AVAL1    *=*+1
AVAL2    *=*+1
!
! Non-zero page reserved bytes
!
         *=2FF9   set program counter to $2FF9
TRACE    *=*+1   flag: trace data structure changes
XHOP     *=*+1   counter: number of hops
XEXM     *=*+1   counter: number of extra markers
XHOL     *=*+1   counter: number of holes
XUNC     *=*+1   counter: number of uncorrectable errors
XCHOP    *=*+1   counter: number of correctable hops
STKPTR   *=*+1   copy of PET stack pointer
!
! Target for interrupts when no action required
!
RTIBYT   .BYT $40   RTI instruction
```

```
! ****************
! ESTIM ROUTINE
! ****************
!
! Estimate pitch-period for current core segment
!
ESTIM   LDA £$01
        STA INEST  set flag
        LDX COFFS
        INX
        LDA DSSTRT,X  number of entries in core seg
        BNE EST01  one or more entries
        JMP NOENTS  no entries in core seg
EST01   CMP £$01
        BEQ ONEENT  1 entry in core seg
        CMP £$03
        BCC EST02
        JMP VOICED
EST02   INX
        INX
        LDA DSSTRT,X  first count in core seg
        CMP £$FF
        BEQ EST03
        JMP VOICED
!
! At this stage, we know that the core segment
! contains two entries, the first =$FF, the
! second <$FF.  We must search S1 and S2:
! if the first entry encountered is <$FF then
! we have a voiced segment;  otherwise the
! segment is unvoiced.
!
EST03   LDX S1OFFS
        INX
        LDA DSSTRT,X  number of entries in S1 seg
        BEQ SRCHS2  since S1 is empty
        INX
        INX
        LDA DSSTRT,X  first count in S1 seg
        CMP £$FF
        BNE EST04
        JMP UNVOIC  segment is unvoiced
EST04   JMP VOICED  segment is voiced
SRCHS2  LDX S2OFFS
        INX
        LDA DSSTRT,X  number of entries in S2 seg
        BNE EST05
        JMP UNVOIC  segment is unvoiced
EST05   INX
        INX
        LDA DSSTRT,X  first count in S2 seg
        CMP £$FF
        BNE EST06
        JMP UNVOIC  segment is unvoiced
EST06   JMP VOICED  segment is voiced
```

```
ONEENT  INX
        INX
        LDA DSSTRT,X  first count in core seg
        CMP £$FF
        BEQ UNVOIC  segment is unvoiced
!
! Segment contains one valid entry.
! We must search P1,P2,S1,S2 looking for
! a non-$FF entry adjacent to the
! core segment.  If found, then the
! segment is voiced;  otherwise, the
! segment is unvoiced.
!
        LDX P1OFFS
        INX
        LDA DSSTRT,X  number of entries in P1 seg
        BEQ SRCHP2  since P1 is empty
        DEX
        LDA DSSTRT,X  P1 seg pointer
        CLC
        ADC P1OFFS  add to P1 seg offset
        TAX
        DEX
        LDA DSSTRT,X  final count in P1 seg
        CMP £$FF
        BEQ SRCHS1  final P1 entry =$FF
        JMP VOICED  segment is voiced
SRCHP2  LDX P2OFFS
        INX
        LDA DSSTRT,X  number of entries in P2 seg
        BEQ SRCHS1  P1 and P2 are both empty
        DEX
        LDA DSSTRT,X  P2 seg pointer
        CLC
        ADC P2OFFS  add to offset for P2 seg
        TAX
        DEX
        LDA DSSTRT,X  final count in P2 seg
        CMP £$FF
        BCC VOICED  non-$FF entry, therefore voiced
SRCHS1  LDX S1OFFS
        INX
        LDA DSSTRT,X  number of entries in S1 seg
        BEQ SRCHS2  since S1 empty
        INX
        INX
        LDA DSSTRT,X  first count in S1 seg
        CMP £$FF
        BEQ UNVOIC  segment is unvoiced
        JMP VOICED  segment is voiced
NOENTS  LDX P1OFFS
        INX
        LDA DSSTRT,X  number of entries in P1 seg
        BEQ UNVOIC  since P1 empty
        DEX
```

```
        LDA DSSTRT,X   pointer for P1 seg
        CLC
        ADC P1OFFS   add to offset
        TAX
        DEX
        LDA DSSTRT,X   final count in P1 seg
        CMP £$FF
        BEQ UNVOIC   final P1 entry =$FF
        LDX S1OFFS   check S1
        INX
        LDA DSSTRT,X   number of entries in S1 seg
        BEQ UNVOIC   since S1 empty
        INX
        INX
        LDA DSSTRT,X   first count in 1 seg
        CMP £$FF
        BNE VOICED   segment is voiced
UNVOIC  LDS £$FF   this segment is unvoiced
        JMP OUTPUT
VOICED  LDX COFFS
        INX
        LDA DSSTRT,X   number of entries in core seg
        BEQ NONE   core seg empty;  try P1
        CMP £$02
        BCS MANY   two or more entries in core seg
!
! There is exactly one entry in the core segment
!
        INX
        INX
        LDA £$03
        STA ENT
        STA SEG
        STX COROFF
        LDA DSSTRT,X   core entry
        JMP OUTPUT
MANY    LSR A   halve number of core seg entries
        BCC CLEAR
        ADC £$00   increment A (since carry set)
        CLC
CLEAR   ROL A   double number of entries in core seg
        TAX
        INX
        STX ENT
        LDX £$03
        STX SEG
        CLC
        ADC COFFS
        TAX
        INX   add one
        LDA DSSTRT,X   core segment entry nearest centre
        STX COROFF
        JMP OUTPUT
NONE    LDX P1OFFS
        LDA DSSTRT,X   pointer for P1 seg
```

```
        TAX
        DEX
        STX ENT
        LDX £$02
        STX SEG
        CLC
        ADC P1OFFS
        TAX
        DEX
        STX COROFF
        LDA DSSTRT,X  last count in P1 seg
OUTPUT  STA THISPP  store estimate for this segment
        JSR PCC  check continuity of predictions
        LDA ACCEPT  is current estimate acceptable?
        BNE THISOK yes
        JMP ESTIM  no - re-estimate
THISOK  LDA £$01
        STA WAIT  set flag
        LDA £$00
        STA INEST  clear flag
        STA IDLE  clear idle time counter
ENDLP   LDA KBDROW  scan keyboard
        CMP £$EF
        BEQ STOPK  STOP key pressed
        LDA WAIT
        BNE INCIDL  still waiting
        JMP ESTIM  start on new segment
INCIDL  LDX £$00  delay loop
IDLELP  INX
        CPX £$03
        NOP
        BNE IDLELP
        INC IDLE  increment idle time counter
        JMP ENDLP
STOPK   JSR FINISH  since user has pressed STOP
        LDX STKPTR  restore stack pointer
        TXS
        RTS
!
!
```

```
! ****************
! SETUP ROUTINE
! ****************
!
! Initialise variables and PIAs, and set up
! interrupt vector addresses.
!
          TSX
          STX STKPTR  save stack pointer
!
! Disable system interrupts
!
          SEI
          LDA £$3C
          STA SYSINT  alter PET interrupt control
!
! Alter NMI vector address to point to 'RTI'
!
          LDA £$00
          STA $94  lsb of NMI
          LDA £$30
          STA $95  msb of NMI
!
! Shift zero page to copy area
!
          LDX £$00
SHIFTZ    LDA $0,X
          STA ZPSTRT,X
          INX
          BNE SHIFTZ
!
! Initialise variables
!
          LDX £$00
          TXA
INITVA    STA P3OFFS,X
          CLC
          ADC £$20
          INX
          CPX £$08
          BCC INITVA
          LDA £$FF
          STA THISPP
          STA PREVPP
          LDA £$00
          STA INEST
          STA INIRQ
          STA NMIOCC
          STA ADDCNT
          STA LT2MS
          STA OPPTR
          STA XHOP
          STA XEXM
          STA XHOL
          STA XUNC
```

```
        STA XCHOP
        LDX UOFFS   get pointer for update seg
        LDA £$02
        STA DSSTRT,X
!
! Wait for SPACE bar to be pressed
!
SPLOOP  LDA KBDROW
        CMP £$FB   space?
        BNE SPLOOP   no - continue waiting
!
! Set up preprocessor PIAs
!
        LDA £$00
        STA PIA1CA   address ddr's
        STA PIA1CB
        STA PIA2CB
        STA PIA1A    both PIA1 ports i/p
        STA PIA1B
        LDA £$FF
        STA PIA2B    PIA2 B port is o/p
        LDA £$24     00100100
        STA PIA1CA
        LDA £$34     00110100
        STA PIA1CB
        LDA £$37     00110111
        STA PIA2CB
!
! Alter interrupt vectors
!
        LDA £$7E
        STA $94   lsb of NMI
        LDA £$33
        STA $95   msb of NMI
        LDA £$3C
        STA $90   lsb of IRQ
        LDA £$32
        STA $91   msb of IRQ
        CLI
        JMP ESTIM   commence pitch-period estimation
!
!
!
```

```
! *****************
! FINISH ROUTINE
! *****************
!
! Prepare to restore PET operating system
!
FINISH   SEI
!
! Alter NMI vector to point to 'RTI'
!
         LDA £$00
         STA $94   lsb of NMI
         LDA £$30
         STA $95   msb of NMI
!
! Disable preprocessor IRQ
!
         LDA £$00
         STA PIA2CB
!
! Restore zero page for PET
!
         LDX £$00
RESTZ    LDA ZPSTRT,X
         STA $0,X
         INX
         BNE RESTZ
!
! Restore system interrupts
!
         LDA £$3D
         STA SYSINT
         CLI
         RTS
!
!
!
```

```
! *****************
! ERROR ROUTINE
! ****************
!
! Utility routine used during software development.
! Restores PET operating system, and prints
! error message number contained in ERRNUM.
!
ERROR   JSR FINISH   restore PET
!
! Print 'ERROR!'
!
        LDX £'E
        LDA £'R
        JSR WRTWO
        LDX £'R
        LDA £'O
        JSR WRTWO
        LDX £'R
        LDA £'!
        JSR WRTWO
        JSR SPAC2
        LDA ERRNUM
        JSR WROB   print error number
        LDX STKPTR   restore stack pointer
        TXS
        RTS   return to PET operating system
!
!
!
```

```
! **********************
! IRQ SERVICE ROUTINE
! **********************
!
! IRQ interrupts occur at the start of each segment
!
IRQSR    LDA £$01
         STA INIRQ   set flag
         LDA £$00
         STA WAIT   clear flag
         LDA PIA2B   clear PIA flag
         LDA TRACE   is trace required?
         BEQ IRQ1   no
         LDX S1OFFS   yes
         DEX
         LDA IDLE
         STA DSSTRT,X
         JSR COPYDS   copy data structure
IRQ1     LDA INEST   check flag
         BEQ CHECK1   continue
!
! Estimation routine was in progress when this
! interrupt occurred.  We must tamper with the
! stack so that RTI from this interrupt returns
! to the start of ESTIM instead of returning to
! the point of interruption.
!
         LDA £$00
         STA INEST   clear flag
         LDA PREVPP
         STA THISPP   set THISPP equal to PREVPP
         TSX
         LDA £ESTIML   force ESTIM address onto stack
         STA $0105,X
         LDA £ESTIMH
         STA $0106,X
CHECK1   LDA LT2MS   check flag
         BNE IRQ01
         JMP INCOFF   continue
!
! First entry in the update segment has
! a count <2mS;  we must compare it with the final
! entry in S3 segment, and discard the lower of
! the two integral values.
!
IRQ01    LDA £$00
         STA LT2MS   clear flag
         LDX S3OFFS
         LDA DSSTRT,X   pointer for S3 seg
         STA S3PTR
         TXA
         CLC
         ADC S3PTR
         TAX
         DEX
```

```
         DEX
         LDA DSSTRT,X   final integral in S3 seg
         LDX UOFFS
         INX
         INX
         CMP DSSTRT,X   first integral in update segment
         BCC S3LTUP   update seg integral is the larger
!
! S3 segment integral is the larger of the two:
! remove first update segment entry, adding
! first count to the second count.
! This involves a shift to the left of values
! in the update segment.
!
         DEX
         DEX
         LDA DSSTRT,X   pointer for update seg
         CMP £$04
         BEQ UPHAS1   only one entry in update seg
         INX
         INX
         INX
         LDA DSSTRT,X   first count in update seg
         INX
         INX
         CLC
         ADC DSSTRT,X   second count in update seg
         BCC CHECK2
         JMP CHECK3   addition overflow
CHECK2   CMP £$C9   is result > 20mS?
         BCC CHECK4   no
CHECK3   LDA £$FF   yes - set to £$FF
CHECK4   STA DSSTRT,X
!
! Carry out shift
!
SHIFTL   LDX UOFFS
         LDA DSSTRT,X
         CLC
         ADC UOFFS
         STA UPLIM   upper limit for shift
         INX
         INX
         INX
         INX
GET      LDA DSSTRT,X
         DEX
         DEX
         STA DSSTRT,X
         INX
         INX
         INX
         LDA DSSTRT,X
         DEX
         DEX
```

```
        STA  DSSTRT,X
        INX
        INX
        INX
        TXA
        CMP  UPLIM   shift limit reached?
        BCC  GET   not yet
DECPTR  LDX  UOFFS   decrement pointer for update seg
        DEC  DSSTRT,X
        DEC  DSSTRT,X
        JMP  INCOFF
UPHAS1  LDA  ADDCNT
        INX
        INX
        INX
        CLC
        ADC  DSSTRT,X   first count in update segment
        BCC  UP1
        LDA   £$FF   count overflow
UP1     STA  ADDCNT
        JMP  DECPTR
!
! Remove final entry in S3 seg
!
S3LTUP  LDX  S3OFFS
        LDA  DSSTRT,X   pointer for S3 seg
        STA  S3PTR
        SEC
        SBC  £$02
        STA  DSSTRT,X   decrement S3 seg pointer
        INX
        DEC  DSSTRT,X   decrement no. of entries
        DEX
        TXA
        CLC
        ADC  S3PTR
        TAX
        DEX
        LDA  DSSTRT,X   final count in S3 Sseg
        LDX  UOFFS
        INX
        INX
        CLC
        ADC  DSSTRT,X   add to first count in update seg
        BCC  S31
        JMP  S32
S31     CMP  £$C9   is result > 20mS?
        BCC  S33   no
S32     LDA  £$FF   yes - set to $FF
S33     STA  DSSTRT,X   first count in update seg
!
! Increment offsets from DSSTRT
!
INCOFF  LDX  £$07
INCR    CLC
```

```
        LDA P3OFFS,X
        ADC £$20
        STA P3OFFS,X
        DEX
        BPL INCR
        LDX UOFFS
        LDA £$02
        STA DSSTRT,X  set new pointer for update seg
!
! Calculate number of entries in new S3 seg
!
        LDX S3OFFS
        LDA DSSTRT,X
        LSR A
        SEC
        SBC £$01
        INX
        STA DSSTRT,X  number of entries in S3 seg
!
! Run inter-entry comparison routine
!
        JSR IEC
        LDA THISPP
        STA PREVPP  save current pitch-period estimate
        TAX
        LDA CONTAB,X  pitch -> frequency conversion
        STA PIA2B  output to DAC
        TXA  get back pitch-period estimate
        LDX OPPTR
        CPX £$FF
        BEQ CHKNMI  output table full
        STA OPSTRT,X  store estimate in o/p table
        INC OPPTR
CHKNMI  LDA NMIOCC  has an NMI occurred?
        BEQ IRQEND  no
!
! An NMI occurred during this IRQSR
!
        LDA £$00
        STA NMIOCC  clear flag
        JMP ENTER  jump into NMISR
IRQEND  LDA £$00
        STA NMIOCC  clear flag
        STA INIRQ  clear flag
        PLA
        TAY
        PLA
        TAX
        PLA
        RTI
!
!
!
```

```
! **********************
! NMI SERVICE ROUTINE
! **********************
!
! An NMI interrupt indicates that data
! are available from the preprocessor.
!
!
NMISR    PHA
         TXA
         PHA
         TYA
         PHA
         LDA PIA1B   count i/p from preprocessor
         STA COUNT
         LDA PIA1A   integral i/p from preprocessor
         STA INTEG
         LDA INIRQ   check flag
         BEQ ENTER   go ahead
         STA NMIOCC  IRQSR in progress
         JMP NMIEND
ENTER    LDA COUNT
         CMP £$14
         BCS ENT3    count >= 2mS
         LDX UOFFS
         LDA DSSTRT,X   pointer for update seg
         CMP £$02
         BEQ ENT2    first entry < 2mS
!
! Count is less than 2mS, so compare
! integral value with previous integral,
! and retain only the larger of the two.
!
         STA UPDPTR
         SEC
         SBC £$02
         CLC
         ADC UOFFS
         TAX
         LDA DSSTRT,X   final integral in update segment
         CMP INTEG
         BCC PRLTCU  previous integral < current one
!
! Previous integral entry is greater than or
! equal to the present one.  The present one
! is to be ignored.
!
         LDA COUNT
         STA PREVC
         LDA ADDCNT
         CLC
         ADC PREVC
         BCC ENT1    ADDCNT <= $FF
         LDA £$FF
ENT1     STA ADDCNT
```

```
           JMP  NMIEND
PRLTCU     INX
           LDA  DSSTRT,X   previous count
           STA  PREVC
           DEC  UPDPTR
           DEC  UPDPTR
           LDA  UPDPTR
           LDX  UOFFS
           STA  DSSTRT,X   decrement update pointer by 2
           LDA  ADDCNT
           CLC
           ADC  PREVC   overflow?
           BCC  ENT11   no
           LDA  £$FF   yes
ENT11      STA  ADDCNT
           CMP  £$14
           BCC  ENT3   < 2mS
           LDA  UPDPTR
           CMP  £$02
           BNE  ENT3
!
! First count in update segment
! is now greater than or equal to 2mS.
!
           LDA  £$00
           STA  LT2MS   clear flag
           JMP  ENT3
ENT2       LDA  £$01
           STA  LT2MS   set flag
ENT3       LDX  UOFFS
           LDA  DSSTRT,X
           STA  UPDPTR
!
! Increment pointer for update segment
!
           CLC
           ADC  £$02
           STA  DSSTRT,X
           TXA
           CLC
           ADC  UPDPTR   find next space in update seg
           TAX
           LDA  INTEG
           STA  DSSTRT,X   store integral value
           LDA  COUNT
           CLC
           ADC  ADDCNT   add in any additional count
           BCC  ENT4   overflow?
           JMP  ENT5   yes
ENT4       CMP  £$C9   is result > 20mS
           BCC  ENT6   no
ENT5       LDA  £$FF
ENT6       STA  COUNT   set count to $FF
           INX
           STA  DSSTRT,X   store count value
```

```
              LDA £$0
              STA ADDCNT   set additional count to 0
              STA INIRQ   clear flag
      NMIEND  PLA
              TAY
              PLA
              TAX
              PLA
              RTI
      !
      !
      !
```

```
! ******************
! COPYDS ROUTINE
! ******************
!
! Create copy of data structure for trace purposes
!
!
COPYDS  LDY £$00
COPY01  LDA DSSTRT,Y
        STA $4000,Y
        INY
        BNE COPY01
        INC $3438  modify machine code
        LDA $3438
        CMP £$80  room left in RAM?
        BCC COPY02  yes
        LDA £$40  no
        STA $3438
        JSR FINISH  prepare to restore PET o.s.
        LDX STKPTR
        TXS
COPY02  RTS
!
!
!
```

```
! ***************
! PCC ROUTINE
! ***************
!
! Prediction continuity check.
! Attempts to trap erroneous pitch-period
! predictions.  Called from ESTIM routine.
!
!
PCC      LDA £$00
         STA ACCEPT   reset flag
         STA PREFND   reset flag
         STA SUCFND   reset flag
         LDA PREVPP
         CMP £$FF
         BNE PCC02   last prediction was not $FF
!
! Last prediction was unvoiced.
!
PCC01    INC ACCEPT   set flag
         RTS
PCC02    LDA THISPP
         STA PPRED
         JSR CALCRE   calculate relative error
         LDA RELERR
         BEQ PCC01   since relative error = 0
         CMP £$01
         BEQ PCC01   since relative error = 1
         CMP £$04
         BCS PCC03   since relative error >= 4
         JSR HOP   check for hop
         RTS
PCC03    BNE PCC04   since rel. error > 4
         JSR EXM   check for extra marker
         RTS
PCC04    CMP £$05
         BNE PCC01   uncorrectable
         JSR HOLE   check for hole
         RTS
!
!
!
```

```
! ***************
! HOP ROUTINE
! ***************
!
! THISPP is within one quarter of PREVPP.
! We must check to see whether a hop has occurred
!
!
HOP      LDA XHOP
         CMP £$FF
         BEQ HOP01
         INC XHOP  increment counter
HOP01    LDA PREVPP
         CMP THISPP
         BCS HOP09  THISPP < PREVPP
!
! THISPP is greater than PREVPP.
! Get preceding neighbour.
!
         JSR GETPRE
         BIT PREFND
         BPL HOP051  preceding neighbour not found
         LDA PCOUNT
         STA PPRED
         JSR CALCRE  calculate relative error
         LDA RELERR
!
! Is preceding neighbour less than PREVPP,
! and within one quarter of PREVPP?
!
         CMP £$03
         BNE HOP05  no - look at succeeding neighbour
!
! Preceding neighbour is less than PREVPP,
! and within one quarter of PREVPP.
! We must alter the count entries of the
! preceding and current entries.
!
HOP02    LDA PCOUNT
         CLC
         ADC THISPP  form THISPP + PCOUNT
         ROR A  (THISPP + PCOUNT)/2
         LDX PREOFF
         STA DSSTRT,X  alter preceding entry
         BCC HOP03
         ADC £$00  increment accumulator
HOP03    LDX COROFF
         STA DSSTRT,X  alter core entry
         LDA XHOP
         CMP £$FF
         BEQ HOP04
         INC XCHOP  increment counter
HOP04    RTS
!
! Get succeeding neighbour
```

```
!
HOP05   JSR GETSUC
        BIT SUCFND
        BPL HOP051  succeeding neighbour not found
        LDA SCOUNT
        STA PPRED
        JSR CALCRE  calculate relative error
        LDA RELERR
!
! Is succeeding neighbour less than PREVPP,
! and within one quarter of PREVPP?
!
        CMP £$03
        BEQ HOP06  yes
!
! Hop is not correctable - accept as it is
!
HOP051  INC ACCEPT
        RTS
!
! Succeeding neighbour is less than PREVPP,
! and is within one quarter of PREVPP.
!
HOP06   LDA SCOUNT
        CLC
        ADC THISPP
        ROR A  (THISPP + SCOUNT)/2
        LDX SUCOFF
        STA DSSTRT,X  alter succeeding entry
        BCC HOP07
        ADC £$00  increment accumulator
HOP07   LDX COROFF
        STA DSSTRT,X  alter core entry
        LDA XCHOP
        CMP £$FF
        BEQ HOP08
        INC XCHOP  increment counter
HOP08   RTS
!
! THISPP < PREVPP
! We must get the preceding neighbour.
!
HOP09   JSR GETPRE
        BIT PREFND
        BPL HOP11  preceding neighbour not found
        LDA PCOUNT
        STA PPRED
        JSR CALCRE  calculate relative error
        LDA RELERR
!
! Is preceding neighbour greater than PREVPP,
! and within one quarter of PREVPP?
!
        CMP £$02
        BNE HOP10  no - get succeeding neighbour
```

```
!
! Yes, it is.  We must alter counts accordingly.
!
          JMP HOP02
!
! Get succeeding neighbour.
!
HOP10     JSR GETSUC
          BIT SUCFND
          BPL HOP11  succeeding neighbour not found
          LDA SCOUNT
          STA PPRED
          JSR CALCRE  calculate relative error
          LDA RELERR
!
! Is succeeding neighbour greater than PREVPP,
! and within one quarter of PREVPP?
!
          CMP £$02
          BEQ HOP06  yes - alter counts
!
! Hop is not correctable.
!
HOP11     INC ACCEPT
          RTS
!
!
!
```

```
! *****************
! GETPRE ROUTINE
! *****************
!
! Sets PSEG and PENT to point to the preceding
! neighbour, PCOUNT to the count value of the
! preceding neighbour, and PREOFF to the offset
! from DSSTRT for the preceding neighbour.
!
!
GETPRE  LDA £$00
        STA PREFND   clear flag
        LDA ENT
        CMP £$03
        BEQ GETP01   first entry in this segment
!
! Core entry is not the first entry in this seg.
!
        SEC
        SBC £$02
        STA PENT   ENT - 2
        LDA SEG
        STA PSEG
        LDA COROFF
        SEC
        SBC £$02
        STA PREOFF   COROFF - 2
        TAX
        LDA DSSTRT,X
        STA PCOUNT   count for preceding entry
        LDA £$80
        STA PREFND   set 'found' flag
        RTS
!
! We must search for the final entry in the
! first preceding non-empty segment.
!
GETP01  LDX SEG
        DEX
GETP02  TXA
        PHA   save X-register on stack
        LDA P3OFFS,X   offset for previous segment
        TAX
        LDA DSSTRT,X   pointer for previous segment
        CMP £$02
        BNE GETP03   segment is non-empty
        PLA
        TAX   restore current offset index
        DEX
        BPL GETP02
!
! All previous segments are empty
!
        RTS
!
```

```
! Non-empty segment
!
GETP03  STA GETPA   save accumulator
        DEC GETPA
        TXA
        CLC
        ADC GETPA   (segment offset + pointer - 1)
        TAX
        LDA DSSTRT,X   preceding count
        STA PCOUNT
        LDA GETPA
        STA PENT
        STX PREOFF
        PLA   restore current offset index
        STA PSEG
        LDA £$80
        STA PREFND   set flag
        RTS
!
!
!
```

```
! *****************
! GETSUC ROUTINE
! *****************
!
! Sets SSEG and SSENT to point to the
! succeeding neighbour, SCOUNT to the count
! value of the succeeding neighbour, and
! SUCOFF to the offset from DSSTRT for the
! succeeding neighbour.
!
!
GETSUC  LDA £$00
        STA SUCFND   clear flag
        LDX SEG
        LDA P3OFFS,X   offset for seg with core entry
        TAX
        LDA DSSTRT,X   pointer for this segment
        STA GETSA   save accumulator value
        LDA ENT
        CLC
        ADC £$02   ENT + 2
        CMP GETSA   compare with pointer
        BCS GETS01   final entry in this segment
!
! The core entry is not the final entry
! in this segment.
!
        STA SENT
        LDA SEG
        STA SSEG
        TXA
        CLC
        ADC SENT
        STA SUCOFF
        TAX
        LDA DSSTRT,X
        STA SCOUNT
        LDA £$80
        STA SUCFND   set 'found' flag
        RTS
!
! Get the first entry in the first
! succeeding non-empty segment.
!
GETS01  LDX SEG
        INX
GETS02  TXA
        PHA   save X-register on stack
        LDA P3OFFS,X   offset for subsequent segment
        TAX
        LDA DSSTRT,X   pointer for this segment
        CMP £$02   is segment empty?
        BNE GETS03   no - get first entry
        PLA   yes - try next segment
        TAX
```

```
             INX
             CPX £$07
             BCC GETS02
!
! All subsequent segments are empty.
!
             RTS
GETS03       LDA £$03
             STA SENT
             INX
             INX
             INX
             STX SUCOFF
             LDA DSSTRT,X
             STA SCOUNT
             PLA  restore X-register
             STA SSEG
             LDA £$80
             STA SUCFND  set 'found' flag
             RTS
!
!
!
```

```
! *****************
! CALCRE ROUTINE
! *****************
!
! Calculate the Relative Error between PPRED
! and PREVPP, and set RELERR accordingly ...
!
!    PPRED equal to PREVPP ... $00
!    PPRED > PREVPP and within one eighth ... $00
!    PPRED < PREVPP and within one eighth ... $01
!    PPRED > PREVPP and within one quarter ... $02
!    PPRED < PREVPP and within one quarter ... $03
!    PPRED < PREVPP and not within one quarter ... $04
!    PPRED > PREVPP and not within three quarters ... $05
!    All other cases ... $FF
!
!
CALCRE  LDA PREVPP
        LSR A
        STA DIV2   (PREVPP/2)
        LSR A
        LSR A
        STA DIV8   (PREVPP/8)
        LDA PREVPP
        CMP PPRED
        BNE CALC02
CALC01  LDA £$00
        STA RELERR   return RELERR = 0
        RTS
CALC02  BCC CALC06   PPRED > PREVPP
!
! PPRED < PREVPP
        SEC
        SBC DIV8   (PREVPP - (PREVPP/8))
        CMP PPRED
        BEQ CALC03
!
! If carry bit is set, then
! PPRED is less than PREVPP, and is not
! within one eighth.
!
        BCS CALC04
!
! PPRED is less than PREVPP and is within
! one eighth of PREVPP.
!
CALC03  LDA £$01
        STA RELERR   return RELERR = 1
        RTS
!
! PPRED is less than PREVPP, and is not
! within one eighth of PREVPP.
!
CALC04  SEC
        SBC DIV8   (PREVPP - (PREVPP/8))
```

```
        CMP PPRED
!
! Is PPRED less than PREVPP, and
! within one quarter of PREVPP?
!
        BCC CALC05  yes
        BEQ CALC05  yes
!
! PPRED is less than PREVPP, and is not
! within one quarter of PREVPP.
!
        LDA £$04
        STA RELERR  return RELERR = 4
        RTS
CALC05  LDA £$03
        STA RELERR  return RELERR = 3
        RTS
!
! PPRED is greater than PREVPP.
!
CALC06  CLC
        ADC DIV8  (PREVPP + (PREVPP/8))
        CMP PPRED
!
! Is PPRED greater than PREVPP and
! within one eighth of PREVPP?
!
        BCS CALC01  yes
        CLC  no
        ADC DIV8  (PREVPP + (PREVPP/4))
        CMP PPRED
!
! Is PPRED greater than PREVPP and not
! within one quarter of PREVPP?
!
        BCC CALC07  yes
!
! PPRED is greater than PREVPP and is
! within one quarter of PREVPP.
!
        LDA £$02
        STA RELERR  return RELERR = 2
        RTS
CALC07  CLC
        ADC DIV2  (PREVPP + 0.75*PREVPP)
        BCC CALC08
        LDA £$FF  overflow on addition
CALC08  CMP PPRED
!
! Is PPRED greater than PREVPP and
! within three quarters of PREVPP?
!
        BCS CALC09  yes
        LDA £$05  no
        STA RELERR  return RELERR = 5
```

```
        RTS
CALC09  LDA £$FF
        STA RELERR   return RELERR = $FF
        RTS
!
!
!
```

```
! ***************
! EXM ROUTINE
! ***************
!
! Routine to process extra markers.
!
!
!
EXM      LDA XEXM
         CMP £$FF
         BEQ EXM01
         INC XEXM   increment counter
EXM01    JSR EXTRA   extraneous entry prediction
         LDA EEPRED
         BNE EXM01A   proceed with extra marker correction
!
! EEPRED is low, therefore ignore possible
! extra marker.
!
         INC ACCEPT
         RTS
!
! Calculate absolute difference between
! THISPP and PREVPP.
!
EXM01A   LDA THISPP
         JSR ABSDIF
         STA PPDIFF
         LDA £$00
         STA SUM1
         STA SUM2
         JSR GETPRE   get preceding neighbour
         LDA PREFND
         BPL EXM03   preceding neighbour not found
         LDA PCOUNT
         CLC
         ADC THISPP
         BCC EXM02
         LDA £$FF   overflow on addition
EXM02    STA SUM1
         JSR ABSDIF
         STA DIFF1
EXM03    JSR GETSUC
         LDA SUCFND
         BPL EXM05   succeeding neighbour not found
         LDA SCOUNT
         CLC
         ADC THISPP
         BCC EXM04
         LDA £$FF   overflow on addition
EXM04    STA SUM2
         JSR ABSDIF
         STA DIFF2
EXM05    LDA SUM1
         BNE EXM06
```

```
                JMP EXM07   one or both not found
EXM06           LDA SUM2
                BNE EXM09   both neighbours found
EXM07           LDA SUM1
                CLC
                ADC SUM2
                BNE EXM08   one neighbour was found
!
! Neither neighbour was found ...
! accept as is.
!
                INC ACCEPT
                RTS
!
! Determine which neighbour was not found
!
EXM08           LDA SUM1
                BEQ EXM10   preceding wasn't found
                JMP EXM13   succeeding wasn't found
!
! Both neighbouring entries were found ...
! select the combination with the lower
! absolute difference from PREVPP.
!
EXM09           LDA DIFF2
                CMP DIFF1
                BCS EXM13   select SUM1
!
! DIFF2 < DIFF1, so select SUM2
!
EXM10           LDA DIFF2
                CMP PPDIFF
                BCC EXM10A
!
! Accept as is.
!
                INC ACCEPT
                RTS
!
! Set count of succeeding entry to SUM2.
!
EXM10A          LDX SUCOFF
                LDA SUM2
                STA DSSTRT,X
!
! We must now remove the core entry.
!
                LDX SEG
                LDA P3OFFS,X
                TAX
                CLC
                ADC DSSTRT,X
                STA EXMA    save this value
                LDX COROFF
EXM11           INX
```

```
          CPX EXMA
          BEQ EXM12   end of shift
!
! Shift entries to the left.
!
          LDA DSSTRT,X
          DEX
          DEX
          STA DSSTRT,X
          INX
          INX
          INX
          LDA DSSTRT,X
          DEX
          DEX
          STA DSSTRT,X
          INX
          INX
          JMP EXM11
!
! Decrement pointer and number of entries
! for this segment.
!
EXM12     LDX SEG
          LDA P3OFFS,X
          TAX
          DEC DSSTRT,X
          DEC DSSTRT,X  decrement pointer by 2
          INX
          DEC DSSTRT,X  decrement number of entries
          RTS
!
! DIFF2 > DIFF1, so select SUM1
!
EXM13     LDA DIFF1
          CMP PPDIFF
          BCC EXM13A
!
! Accept as is.
!
          INC ACCEPT
          RTS
EXM13A    LDX COROFF
          LDA SUM1
          STA DSSTRT,X  set core count to SUM1
!
! We must now remove the preceding entry.
!
          LDX PSEG
          LDA P3OFFS,X
          TAX
          CLC
          ADC DSSTRT,X
          STA EXMA
          LDX PREOFF
```

```
EXM14   INX
        CPX EXMA
        BEQ EXM15   end of shift
!
! Shift entries to the left.
!
        LDA DSSTRT,X
        DEX
        DEX
        STA DSSTRT,X
        INX
        INX
        INX
        LDA DSSTRT,X
        DEX
        DEX
        STA DSSTRT,X
        INX
        INX
        JMP EXM14
!
! Decrement pointer and number of entries
! for this segment.
!
EXM15   LDX PSEG
        LDA P3OFFS,X
        TAX
        DEC DSSTRT,X
        DEC DSSTRT,X   decrement pointer by 2
        INX
        DEC DSSTRT,X   decrement number of entries
        RTS
!
!
!
```

```
! *****************
! ABSDIF ROUTINE
! *****************
!
! Compute absolute difference between PREVPP
! and accumulator.  This routine returns with
! the accumulator set to this difference value.
!
!
ABSDIF   SEC
         SBC PREVPP
         BCS ABS01   result is positive
!
! Result is negative ... take the 2's complement of
! the accumulator to yield the absolute difference.
!
         EOR £$FF
         CLC
         ADC £$01
!
! The accumulator mow contains the absolute
! difference value.
!
ABS01    RTS
!
!
!
```

```
! ****************
! HOLE ROUTINE
! ****************
!
! Routine to process a hole.
!
!
HOLE      LDA XHOL
          CMP £$FF
          BEQ HOL01
          INC XHOL  increment counter
HOL01     JSR EXTRA  extraneous entry prediction
          LDA EEPRED
          CMP £$02  is EEPRED < 2?
          BCC HOL01A  yes - proceed with hole correction
!
! EEPRED is high, therefore ignore possible hole.
!
          INC ACCEPT
          RTS
HOL01A    LDA THISPP
          CMP £$FF
          BEQ HOL06  THISPP is unvoiced
!
! Current prediction is voiced ... check
! to see whether one marker has been missed
!
          LSR A  (THISPP/2)
          STA PPRED
          JSR CALCRE
          LDA RELERR
          BEQ HOL02
          CMP £$01
          BEQ HOL02
!
! Not a simple case of one missed marker.
!
          LDA £$FF
          STA THISPP
          JMP HOL06
!
! It would seem that one marker has been missed ...
! we must insert an extra marker before the core
! entry, and alter the core enntry itself.
!
HOL02     LDA THISPP
          LSR A
          STA HOLEA  (THISPP/2)
          STA HOLEB
          BCC HOL03
          INC HOLEB  adjust for carry
HOL03     LDX SEG
          LDA P3OFFS,X
          TAX
          CLC
```

```
          ADC  DSSTRT,X
          TAX
HOL04     DEX
          DEX
          LDA  DSSTRT,X
          INX
          INX
          STA  DSSTRT,X
          DEX
          CPX  COROFF   core entry reached?
          BEQ  HOL05   yes
          INX   no - continue
          INX
          STA  DSSTRT,X
          DEX
          DEX
          DEX
          JMP  HOL04
!
! The core entry has been reached.
!
HOL05     LDA  HOLEA
          STA  DSSTRT,X   insert extra entry
          INX
          INX
          LDA  HOLEB
          STA  DSSTRT,X   alter core entry
!
! Increment pointer and number of entries.
!
          LDX  SEG
          LDA  P3OFFS,X
          TAX
          INC  DSSTRT,X
          INC  DSSTRT,X   increment pointer by two
          INX
          INC  DSSTRT,X   increment number of entries
          RTS
!
! This is not a simple case of one missing marker.
! We must search for and examine the succeeding
! neighbour ... if it is within one quarter of
! PREVPP then we can place THISPP half way between
! the two values.  Otherwise we must assume that
! this segment is unvoiced.
!
HOL06     JSR  GETSUC
          BIT  SUCFND   found it?
          BMI  HOL07   yes
!
! Succeeding entry not found.
!
          INC  ACCEPT
          RTS
HOL07     LDA  SCOUNT
```

```
                STA  PPRED
                JSR  CALCRE
                LDA  RELERR
                CMP  £$04
                BCC  HOL08   SCOUNT is within limits
!
! SCOUNT is outside the permissible range.
! We must predict that this segment is unvoiced.
!
                INC  ACCEPT
                RTS
HOL08           LDA  PREVPP
                CLC
                ADC  COUNT
                ROR  A    (PREVPP + SCOUNT)/2
                STA  THISPP
                INC  ACCEPT
                RTS
!
!
!
```

```
!  ***************
!  IEC  ROUTINE
!  ***************
!
!  This  routine  produces  inter-entry
!  comparison  data  for  entry  into  the
!  S3  segment.
!
!
IEC         LDY  £$00   Y-register  counts  IEC  entries
            LDX  S3OFFS
            LDA  DSSTRT,X
            CMP  £$02
            BNE  IEC1
            RTS   since  update  segment  is  empty
IEC1        LDX  S2OFFS
            LDA  DSSTRT,X
            CMP  £$02
            BEQ  IEC2   since  S2  segment  is  empty
!
!  Get  final  entry  in  S2  segment.
!
            CLC
            ADC  S1OFFS
            JMP  IEC11
IEC3        LDA  £$00
            STA  IECC
            STA  IECI
            JSR  ENTIEC   enter  values
            JMP  IEC5
IEC4        LDX  S3OFFS
            INX
            INX
            LDA  DSSTRT,X   first  integral  in  S3
            STA  ICOMP2
            INX
            LDA  DSSTRT,X   first  count  in  S3
            STA  CCOMP2
            JSR  COMP
            JSR  ENTIEC
IEC5        LDX  S3OFFS
            INX
            LDA  DSSTRT,X   number  of  entries  in  S3  seg
            STA  S3NENT
IEC6        INY   increment  IEC  entry  counter
            CPY  S3NENT
            BEQ  IEC7   all  entries  accounted  for
            TYA   perform  comparison
            CLC
            ROL  A
            ADC  S3OFFS
            TAX
            LDA  DSSTRT,X
            STA  ICOMP1
            INX
```

```
               STA  CCOMP1
               INX
               LDA  DSSTRT,X
               STA  ICOMP2
               INX
               LDA  DSSTRT,X
               STA  CCOMP2
               JSR  COMP
               JSR  ENTIEC
               JMP  IEC6
        IEC7   RTS   end of IEC routine
        !
        !
        !
```

```
! ****************
! COMP ROUTINE
! ****************
!
! This subroutine compares ICOMP1 with ICOMP2,
! placing the result in IECI, and compares
! CCOMP1 with CCOMP2, placing the result in
! IECC.  The results are as follows:
!
!    ICOMP1 < ICOMP2    result  $80
!    CCOMP1 < CCOMP2    result  $80
!
!    ICOMP1 = ICOMP2    result  $00
!    CCOMP1 = CCOMP2    result  $00
!
!    ICOMP1 > ICOMP2    result  $01
!    CCOMP1 > CCOMP2    result  $01
!
!
COMP     LDA ICOMP1  compare integrals
         LSR A
         LSR A
         LSR A  divide by 8
         STA EIGHTH
         CLC
         ADC ICOMP1
         STA COMPHI
         LDA ICOMP1
         SEC
         SBC EIGHTH
         STA COMPLO
         LDA ICOMP2
         CMP COMPHI
         BCC COMP1
         BEQ COMP1
         LDA £$01  (>)
         JMP COMPC
COMP1    CMP COMPLO
         BCS COMP2
         LDA £$80  (<)
         JMP COMPC
COMP2    LDA £$00  (=)
COMPC    STA IECI
         LDA CCOMP1  compare counts
         LSR A
         LSR A
         LSR A  divide by eight
         STA EIGHTH
         CLC
         ADC CCOMP1
         STA COMPHI
         LDA CCOMP1
         SEC
         SBC EIGHTH
         STA COMPLO
```

```
          LDA  CCOMP2
          CMP  COMPHI
          BCC  COMP3
          BEQ  COMP3
          LDA  £$01    (>)
          JMP  COMP5
COMP3     CMP  COMPLO
          BCS  COMP4
          LDA  £$80    (<)
          JMP  COMP5
COMP4     LDA  £$00    (=)
COMP5     STA  IECC
          RTS
!
!
!
```

```
! ******************
! ENTIEC ROUTINE
! ******************
!
! This routine enters inter-entry comparison
! data into the data structure.
! The Y-register contains the entry location
! offset.  Data are in IECI and IECC.
! Data are stored in two adjacent locations,
! the first being (S3OFFS + $10 + Y-register*2).
!
!
ENTIEC    TYA
          CLC
          ROL A
          CLC
          ADC £$10
          ADC S3OFFS
          TAX
          LDA IECI
          STA DSSTRT,X
          INX
          LDA IECC
          STA DSSTRT,X
          RTS
!
!
!
```

```
! *****************
! EXTRA ROUTINE
! *****************
!
! Extraneous entry prediction routine.
! This routine constructs IEC lists INTLIS
! and CNTLIS for the current 87.5mS section,
! performs a modified autocorrelation routine
! on these lists, and on the basis of the results
! obtained gives a prediction of the likelihood
! of extraneous entries being present in the
! data structure.  This prediction is saved in
! EEPRED and stored in the current core segment,
! and takes the following values:
!
!       $00 ... no extraneous entries
!       $01 ... possibility of extraneous entries
!       $02 ... strong possibility of one extraneous
!                     entry per pitch-period
!       $03 ... strong possibility of two extraneous
!                     entries per pitch-period.
!
!
EXTRA    LDY £$00
         STY LISPTR
EXTR1    LDX P3OFFS,Y
         INX
         LDA DSSTRT,X  number of entries for current seg
         BEQ EXTR3  this segment is empty: go to next one
         STA SEGN
         TXA
         CLC
         ADC £$0E
         TAX
!
! Get all entries from this segment.
!
EXTR2    INX
         LDA DSSTRT,X  integral entry
         STX SAVEX  save X-register
         LDX LISPTR  pointer for lists
         STA INTLIS,X  put value into INTLIS
         LDX SAVEX  restore X-register
         INX
         LDA DSSTRT,X  count entry
         STX SAVEX  save X-register
         LDX LISPTR
         STA CNTLIS,X  store value in CNTLIS
         LDX SAVEX
         INC LISPTR
         DEC SEGN
         BEQ EXTR3  all entries in this seg collected
         JMP EXTR2  get next entry from this segment
EXTR3    INY
         CPY £$07
```

```
                BNE EXTR1   go on to next segment
!
! Lists have now been constructed.
!
                LDA LISPTR
!
! A minimum of four entries must be present for
! modified autocorrelation routine to be carried
! out.  Are there enough?
!
                CMP  £$04
                BCS EXTR5   yes - proceed with autocorrelation
                LDA £$00
EXTR4           STA EEPRED  extraneous entry prediction
                LDX S1OFFS
                DEX
                DEX
                STA DSSTRT,X  store EEPRED in core segment
                RTS
EXTR5           LDA INTLIS
                STA LISST
                JSR AUTO   perform modified autocorrelation
                STA IPRED
                LDA CNTLIS
                STA LISST
                JSR AUTO   perform modified autocorrelation
                STA CPRED
!
! Compare IPRED with CPRED and combine them
! to produce EEPRED.  If the values are equal,
! then select this value ...
!
                LDA IPRED
                CMP CPRED
                BEQ EXTR6   values are equal
!
! If one value is 0 or one value is 1
! then select the other value ...
!
                LDA IPRED
                BEQ EXTR6   IPRED = 0
                CMP £$01
                BEQ EXTR6   IPRED = 1
                LDA CPRED
                BEQ EXTR7   CPRED = 0
                CMP £$01
                BEQ EXTR7   CPRED = 1
!
! If neither of the above two conditions has been
! satisfied, then select an EEPRED value of 1,
! since the predictions clash.
!
                LDA £$01
                JMP EXTR4
EXTR6           LDA CPRED
```

```
                JMP  EXTR4
      EXTR7     LDA  IPRED
                JMP  EXTR4
      !
      ! End of EXTRA routine
      !
      !
      !
```

```
! ****************
! AUTO ROUTINE
! ****************
!
! This routine performs a modified autocorrelation
! operation on an inter-entry comparison list,
! starting at 'LISST', with 'LISPTR' entries.
! Before exit from this routine, the accumulator is
! loaded with the prediction.
!
! Zero lag computation
!
AUTO      LDX £$00
          STX LAG0
          STX LAG1
          STX LAG2
          STX LAG3
AUTO1     CPX LISPTR
          BEQ AUTO3   end of list
          LDA LISST,X
          BEQ AUTO2
          INC LAG0
AUTO2     INX
          JMP AUTO1
AUTO3     LDA LAG0
          CMP £$02  is zero lag value < 2?
          BCS AUTO4   no - so continue
          LDA £$00  yes - return prediction 0
          RTS
!
! One lag
!
AUTO4     LDX £$00
AUTO5     LDA LISST,X
          STA AVAL1
          INX
          CPX LISPTR
          BCS AUTO8   end of list
          LDA LISST,X
          STA AVAL2
          JSR ACOMP
          BEQ AUTO5   no action
          BMI AUTO7   clash
          INC LAG1
          JMP AUTO5
AUTO7     DEC LAG1
          JMP AUTO5
!
! Two lag
!
AUTO8     LDX £$00
AUTO9     LDA LISST,X
          STA AVAL1
          INX
          INX
```

```
          CPX  LISPTR
          BCS  AUTO12   end of list
          LDA  LISST,X
          STA  AVAL2
          JSR  ACOMP
          BEQ  AUTO10   no action
          BMI  AUTO11   clash
          INC  LAG2
AUTO10    DEX
          JMP  AUTO9
AUTO11    DEC  LAG2
          JMP  AUTO10
!
! Three lag
!
AUTO12    LDX  £$00
AUTO13    LDA  LISST,X
          STA  AVAL1
          INX
          INX
          INX
          CPX  LISPTR
          BCS  AUTO16   end of list
          LDA  LISST,X
          STA  AVAL2
          JSR  ACOMP
          BEQ  AUTO14   no action
          BMI  AUTO15   clash
          INC  LAG3
AUTO14    DEX
          DEX
          JMP  AUTO13
AUTO15    DEC  LAG3
          JMP  AUTO14
!
! Modified autocorrelation is complete.
!
AUTO16    LDA  LAG1
          BMI  AUTO19   LAG1 < 0
          CMP  LAG2
          BCC  AUTO18   possible extraneous entries
          BEQ  AUTO18   possible extraneous entries
          CMP  LAG3
          BCC  AUTO18   possible extraneous entries
          BEQ  AUTO18   possible extraneous entries
AUTO17    LDA  £$00     return prediction of 0
          RTS
AUTO18    LDA  £$01     return prediction of 1
          RTS
AUTO19    LDA  LAG2
          BMI  AUTO22   LAG2 < 0
          BEQ  AUTO22   LAG2 = 0
          LDA  LAG3
!
! LAG2 is greater than zero.
```

```
!
           BMI AUTO21   LAG3 < 0   ... predict 2
           BEQ AUTO21   LAG3 = 0   ... predict 2
!
! LAG2 and LAG3 are both positive.
!
           CMP LAG2
           BEQ AUTO18   return prediction of 1
           BCC AUTO21   LAG3 < LAG2
AUTO20     LDA £$03     return prediction of 3
           RTS
AUTO21     LDA £$02     return prediction of 2
           RTS
AUTO22     LDA LAG3
           BMI AUTO17   neither is > 0
           BEQ AUTO17   neither is > 0
           JMP AUTO20   return prediction of 3
!
! End of AUTO routine
!
!
!
```

```
! *****************
! ACOMP ROUTINE
! *****************
!
! Compares two values, AVAL1 and AVAL2 for the
! purposes of modified autocorrelation.
! The accumulator is loaded before return from
! this routine as follows:
!
!    AVAL1 and/or AVAL2 = 0 ... $00
!    AVAL1 = AVAL2 ........... $01
!    AVAL1 ≠ AVAL2 .......... $80
!
!
ACOMP    LDA AVAL1
         BNE ACOMP2
ACOMP1   LDA £$00   return $00
         RTS
ACOMP2   LDA AVAL2
         BEQ ACOMP1
         CMP AVAL1
         BEQ ACOMP3   values are equal
         LDA £$80   return $80
         RTS
ACOMP3   LDA £$01   return $01
         RTS
!
! ==========================================
!
         .END
```

APPENDIX FIVE

LISTING OF THE HIGH-LEVEL SOFTWARE FOR THE PROJECT

```
100 PRINT "<clear screen>"
110 PRINT "Educational Research Speech Analysis System"
120 PRINT
130 PRINT "ERSA Level 3             Version 3.2"
140 PRINT: PRINT
150 PRINT "Switch on the preprocessor, then press SPACE."
160 PRINT
170 PRINT "To return to normal PET operation, press STOP."
180 SYS 12644: REM jump to SETUP routine
190 END
```

REFERENCES

References appear in the text as a four-character code enclosed in square brackets, e.g.

[D334]

The letter is the initial of the first author's surname. The first digit indicates the Chapter in which the reference is first quoted. The remaining two digits form a sequence number for references within that Chapter. Thus, '[D334]' is the 34th reference in Chapter 3.

The references are listed on the following pages in alphabetical order of initials, and numerically within each letter group.

A104 Annett J
Feedback and human behaviour
Penguin. Harmondsworth (1969)

A135 Abberton E, Fourcin AJ
A visual display for teaching intonation and rhythm
British Council, ELT Documents(73/5) (1973), 2-6

A144 Angelocci A, Kopp G, Holbrook A
The vowel formants of deaf and normal hearing - eleven- to fourteen-year-old boys
J.Speech Hear.Disorders, 29 (1964), 156-170

A152 Anderson F
An experimental pitch indicator for training deaf scholars
J.Acoust.Soc.Amer., 32 (1960), 1065-1074

A369 Atal BS, Hanauer SL
Speech analysis and synthesis by linear prediction of the speech wave
J.Acoust.Soc.Amer., 50 (1971), 637-655

A376 Atal BS, Rabiner LR
A pattern recognition approach to unvoiced-voiced-silence classification, with applications to speech recognition
IEEE Trans.Acoust.Speech and Signal Proc., 24 (1976), 201-212

A383 Atal BS
Unpublished work
Communicated to [R381] (1976)

A401 Allen J
Computer architecture for signal processing
Proc.IEEE, 63 (1975), 624-633

A431 Analog Devices Inc.
Real-time interfaces are Motorola compatible
Analog Dialog, 12, No.2 (1978), 10

A432 Analog Devices Inc.
Motorola micromodule compatible analog I/O subsystem model RTI-1230, data sheet
Analog Devices Inc, Norwood, Massachusetts (1978)

A502 Analog Devices Inc.
Integrated circuit true rms-to-dc converter, type AD536A
Analog Devices Inc., Norwood, Massachusetts (1979)

B101 Bradshaw J
Acoustics of utterances
Univ.of Aston, Dept.of Educational Enquiry, Internal Report (Oct 1976)

B116 Bolinger DL
Intonation
Penguin, Harmondsworth (1972)

B121 Bolinger DL
A theory of pitch accent in English
Word, 14 (1958), 109-149

B142 Borrild K
Experience with the design and use of technical aids for the training of deaf and hard of hearing children
Amer.Ann.Deaf, 113 (1968), 168-177

B155 Boothroyd A
Some experiments on the control of voice in the profoundly deaf using a pitch extractor and storage oscilloscope display
IEEE Trans.Audio Electroacoust., 21 (1973), 274-278

B156 Boothroyd A, Decker M
Control of pitch by the deaf
Audiology, 11 (1972), 343-353

B321 Bezdel W, Chandler HJ
Results of an analysis and recognition of vowels by computer using zero-crossing data
Proc. Inst. Elec. Eng., 112 (1965), 2060-2066

B325 Baker JM
Time-domain analysis and segmentation of connected speech in Fant G. (ed), "Speech Communication", Proc.Sp.Comm.Sem., Stockholm, 3 (1974), 369-383

B329 Blankinship WA
Note on computing autocorrelations
IEEE Trans.Acoust.Speech and Signal Proc., 22 (1974), 76-77

B339 Bergland GD
Fast Fourier transform hardware implemetations - a survey
IEEE Trans.Audio and Electroacoust., 17 (1969), 109-119

B342 Blackman RB, Tukey JW
The measurement of power spectra
Dover, New York (1959)

B357 Bogert BP, Healy MJR, Tukey JW
The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking in Proc.Symposium on Time Series Analysis, M.Rosenblatt(ed), John Wiley & Sons, New York (1963), 209-243

B408 Bruce JD
Digital signal processing concepts
IEEE Trans.Audio Electroacoust., 18 (1970), 344-353

B417 Brodersen RW, White RM
New technologies for signal processing
Science, 195 (Mar 1977), 1216-1222

C112 Christina RW
Minimum visual feedback processing time for amendment of an incorrect movement
Perceptual and Motor Skills, 31 (1970), 991-994

C115 Crystal D
Prosodic systems and intonation in English
Cambridge University Press, Cambridge (1969)

C126 Coulthard M
An introduction to discourse analysis
Longman, London (1977)

C131 Catford JC, Pisoni DB
Auditory vs. articulatory training in exotic sounds
Modern Lang.J., 54 (1970), 477-481

C221 Cniba T, Kajiyama M
The vowel, its nature and structure
Tokyo-Kaiselkau, Tokyo (1941)

C324 Chang SH, Pihl GE, Wiren J
The intervalgram as a visual representation of speech sounds
J.Acoust.Soc.Amer., 23 (1951), 675-679

C378 Comer DJ
The use of waveform asymmetry to identify voiced sounds
IEEE Trans.Audio Electroacoust., 16 (1968), 500-506

C411 Connelly JA, Prescott G
Active filter improves tracking and capture ranges of PLL
Electronic Design, 7 (Apr 1975), 128-129

C424 Commodore Business Machines Inc.
PET 2001-32N personal computer user manual
CBM Inc., Santa Clara, CA (1979)

C425 Crabb P
Educational Computing register. Which is the most popular computer?
Educational Computing, 2 (1981), 56-59

C603 Commodore Business Machines Inc.
PET communication with the outside world
CBM Inc., Santa Clara, CA (1978)

D129 Denes PB
The use of speech analysis and synthesis in speech training
in [L158] (1970)

D130 Davis D
Acoustical tests and measurements
Foulsham-Sams, Slough (1965)

D153 Dolansky L, Phillips ND, Bass SD, Pronovost WL, Anderson DC
An intonation display system for the deaf
Acustica, 25 (1971), 190-202

D204 Davies DV (editor)
Gray's anatomy
Longman's, London, 34th.Edition (1967)

D220 Dudley H
The carrier nature of speech
Bell Syst.Tech.J., 19 (1940), 495-515

D225 Dunn HK
The calculation of vowel resonances, and an electrical vocal tract
J.Acoust.Soc.Amer, 22 (1950), 740-753

D306 Dolansky LO
An instantaneous pitch-period indicator
J.Acoust.Soc.Amer., 27 (1955), 67-72

D334 Dubnowski JJ, Schafer RW, Rabiner LR
Real-time digital hardware pitch detector
IEEE Trans.Acoust.Speech and Signal Proc., 24 (1976), 2-8

D363 DeLellis J
Real-time digital-cepstrum pitch extraction on a 12-bit machine
J.Acoust.Soc.Amer., 46 (1969), 81

F107 Fourcin AJ, Abberton E
First applications of a new Laryngograph
Med.and Biol.Illustr., 21 (1971), 172-182

F118 Fonagy I
Electro-physiological and acoustic correlates of stress and stress perception
J.Speech Hear.Res., 9 (1966), 231-244

F124 Fry DB
Experiments in the perception of stress
Language and Speech, 1 (1958), 126-152

F134 Fourcin AJ
Perceptual mechanisms at the first level of speech processing
Proc.VII Int.Cong.Phon.Sci., Montreal (1972), 48-62

F139 Fourcin AJ, Abberton E
The Laryngograph and the Voiscope in speech therapy
in Proc.XVI Int.Congr.of Logopedics and Phoniatrics, Karger, Basel (1974), 116-122

F157 Fourcin AJ, Simon C
Speech and hearing - work in progress
University College London, Dept.of Phonetics and Linguistics (1976)

F201 Fant G
Acoustic theory of speech production
Mouton, 's-Gravenhage, (1960)

F202 Farnsworth DW
High-speed motion pictures of the human vocal cords
Bell Labs Record, 18 (1940), 203-208

F203 Flanagan JL
Speech analysis, synthesis and perception
Springer-Verlag, Berlin (1965)

F205 Fletcher H
Speech and hearing in communication
van Nostrand, New York (1953)

F208 Fletcher WW
A study of internal laryngeal activity in relation to vocal intensity
Northwestern Univ.Evanston, Illinois, PhD thesis (1950)

F209 Flanagan JL
Some properties of the glottal sound source
J.Speech Hear.Res, 1 (1958), 90-116

F210 Flanagan JL, Landgraf L
Self oscillating source for vocal tract synthesizers
IEEE Trans.Audio Electroacoust., 16 (1968), 57-64

F211 Flanagan JL, Cherry L
Excitation of vocal tract synthesizers
J.Acoust.Soc.Amer, 45 (1969), 764-769

F218 Fujimura O
Analysis of nasal consonants
J.Acoust.Soc.Amer, 34 (1962), 1865-1875

F222 Fant G
The acoustics of speech
Proc.III Int.Congr.Acoust., Stuttgart, (1959), 88-201

F224 Flanagan JL
Voices of men and machines
J.Acoust.Soc.Amer, 51 (1972), 1375-1387

F307 Filip M
Envelope periodicity detection
J.Acoust.Soc.Amer., 45 (1969), 719-732

F308 Fourcin AJ, West JE
Larynx movement detector
Progress Report, Phonetics Lab., University College London (1968)

F310 Fabre P
Un procede electrique percutane d'inscription de l'accolement glottique au cours de la phonation: glottographie de haute frequence. Premiers resultats.
Bull.Acad.Nat.Med, 141 (1957), 69-99

F353 Flanagan JL, Golden RM
Phase vocoder
Bell Syst.Tech.J., 45 (1966), 1493-1509

F430 Fullagar D, Bradshaw P, Evans L, O'Neill B
Interfacing data converters and microprocessors
Electronics, 49 (Dec 1976), 81-89

G305 Gruenz OO, Schott LO
Extraction and portrayal of pitch of speech sounds
J.Acoust.Soc.Amer., 21 (1949), 487-495

G315 Gold B
Description of a computer program for pitch detection
Proc IV Int. Congr. Acoust., paper G34, Copenhagen (1962)

G316 Gold B
Computer program for pitch extraction
J.Acoust.Soc.Amer., 34 (1962), 916-921

G317 Gold B, Rabiner LR
Parallel processing techniques for estimating pitch periods of speech in the time domain
J.Acoust.Soc.Amer., 46 (1969), 442-449

G323 Geckinli NC, Yavuz D
Algorithm for pitch extraction using zero-crossing interval sequence
IEEE Trans.Acoust.Speech and Signal Proc., 25 (1977), 559-564

G332 Gill JS
Automatic extraction of the excitation function of speech with particular reference to the use of correlation methods
Proc III Int. Congr. Acoust., Stuttgart (1959), 217-220

G347 Gill JS
A versatile method for short-term spectrum analysis in 'real-time'
Nature, 189 (1961), 117-119

G403 Gold B, Lebow IL, McHugh PG, Rader CM
The FDP, a fast programmable signal processor
IEEE Trans.Computers, 20 (1971), 33-38

G405 Gold B, Rader CM
Digital processing of signals
McGraw-Hill, New York (1969)

G416 Graeme JG
Applications of operational amplifiers - third generation techniques
McGraw-Hill Kogakusha, Tokyo, (1973)

G427 Gould Advance
OS4002 digital storage oscilloscope manual
Gould Advance (1976)

H105 Hilgard ER, Atkinson RC, Atkinson RL
Introduction to psychology
Harcourt Brace Jovanovitch, New York, 5th Edition (1971)

H122 Hulbert HH
Voice training in speech and song
University Tutorial Press, London (1928)

H136 Halliday MAK
A course in spoken English: intonation
Oxford University Press, Oxford (1970)

H147 Hudgins CV
Visual aids in the correction of speech
Volta Review, 37 (1935), 637-704

H207 Holmes JN
An investigation of the volume velocity waveform at the larynx during speech by means of an inverse filter
Proc IV Int.Congr.Acoust., Copenhagen (1962)

H215 Hughes GW, Halle M
Spectral properties of fricative consonants
J.Acoust.Soc.Amer, 28 (1956), 303-310

H216 Heinz JM, Stevens KN
On the properties of voiceless fricative consonants
J.Acoust.Soc.Amer, 33 (1961), 589-596

H217 Halle M, Hughes GW, Radley JPA
Acoustic properties of stop consonants
J.Acoust.Soc.Amer, 29 (1957), 107-116

H302 Hsu HP
Fourier analysis
Simon & Schuster, New York (1970)

H428 Hampshire N
Fourier transforms on the PET
Practical Computing, 2, No.9 (Dec 1979), 111-121

H604 Hampshire N
The PET revealed
Computabits, Yeovil (1979)

H801 Holm RE
First single-chip signal processor simplifies analog design problems
in Bursky D.(ed), 'Components for microcomputer system design', Hayden, New Jersey (1980)

I341 IEEE Digital Signal Processing Committee
Programs for digital signal processing
IEEE Press, New York, Chapter 1 'Discrete Fourier transform programs' (1979)

I366 IEEE Digital Signal Processing Committee
Programs for digital signal processing
IEEE Press, New York, Chapter 7 'Cepstral Analysis' (1979)

I370 Itakura F, Saito S
An analysis-synthesis telephony system based on maximum likelihood method
Electronics Commun.Japan, 53 (1970), 36-43

I374 IEEE Digital Signal Processing Committee
Programs for digital signal processing
IEEE Press, New York, Chapter 4 'Linear prediction analysis of speech signals' (1979)

I418 IEEE Spectrum
Editorial comment
IEEE Spectrum, 17 (1980), 10

I419 Intel Corporation
Component data catalog
Intel Corp., Santa Clara, CA (1979)

J137 James EF
The speech analyzer of the University of Toronto
in [L158] (1970)

J219 Jassem W
The acoustics of consonants
in Sovijarvi A, Aalto P(eds), Proc IV Int.Congr.Phon.Sci., Mouton, The Hague (1962), 50-72

J409 Jackson LB, Kaiser JF, McDonald HS
An approach to the implementation of digital filters
IEEE Trans.Audio Electroacoust., 16 (1968), 413-421

J601 Jackson MA
Principles of program design
Academic Press, London (1975)

K133 Kalikow DN, Swets JA
Experiments with computer-controlled displays in second-language learning
IEEE Trans.Audio Electroacoust., 20 (1972), 23-28

K344 Koenig W, Dunn HK, Lacey LY
The sound spectrograph
J.Acoust.Soc.Amer., 18 (1946), 19-49

K364 Kelly JM, Kennedy RN
An experimental cepstrum pitch detector for use in a 2400-bit/sec channel vocoder
ASA 72nd Meeting, paper IH3 (1966)

K380 Knorr S
Reliable voiced/unvoiced decision
IEEE Trans.Acoust.Speech and Signal Proc., 27 (1979), 263-267

K426 Keay CSL
Evaluating microprocessors for real-time service
Practical Computing, 3, No.12 (Dec 1980), 72-73

K608 Kane J
An introduction to microcomputers - Vol III: some real support devices
Osborne & Associates, Berkeley, California (1978)

L103 Lehiste I, Peterson GE
Some basic considerations in the analysis of intonation
J.Acoust.Soc.Amer., 33 (1961), 419-425

L108 Longuet-Higgins HC
Personal communication
(1978)

L125 Lehiste I, Peterson GE
Vowel amplitude and phonemic stress in American English
J.Acoust.Soc.Amer., 31 (1959), 428-435

L128 Lieberman P
On the acoustic basis of the perception of intonation by linguists
Word, 21 (1965), 40-54

L132 Lierberman AM, Cooper FS, Harris KS, MacNeilage PF
A motor theory of speech perception
Proc.Speech Comm.Seminar, Stockholm, paper D3 (1963)

L158 Leon PR, Faure G, Rigault A
Prosodic features analysis
Didier, Paris (1970)

L318 Licklider JCR, Pollack I
Effects of differentiation, integration, and infinite peak clipping upon the intelligibility of speech
J.Acoust.Soc.Amer., 20 (1948), 42-51

L330 Lopresti PV, Suri HL
A fast algorithm for the estimation of autocorrelation functions
IEEE Trans.Acoust.Speech and Signal Proc., 22 (1974), 449-453

L609 Leventhal LA
6800 assembly language programming
Osborne & Associates, Berkeley, California (1978)

M117 Martinet A
Elements of general linguistics
Faber and Faber, London, (translated by E.Palmer) (1964)

M141 Makepeace A, Shutt L, Pappin J, Bowes J, Fourcin A
Laryngographic study of post operative sore throat
in [F157] (1976)

M146 Martony J
On the correction of the voice pitch level for severely hard of hearing subjects
Amer.Ann.Deaf, 113 (1968), 195-202

M206 Miller RL
Nature of the vocal cord wave
J.Acoust.Soc.Amer, 31 (1959), 667-677

M313 Miller NJ
Pitch detection by data reduction
IEEE Trans.Acoust.Speech and Signal Proc., 23 (1975), 72-79

M320 Munson WA, Montgomery HC
A speech analyzer and synthesizer
J.Acoust.Soc.Amer., 22 (1950), 678

M327 Markel JD, Gray AH
On autocorrelation equations as applied to speech analysis
IEEE Trans.Audio and Electroacoust., 21 (1973), 69-79

M336 Moorer JA
The optimum comb method of pitch period analysis of continuous digitalized speech
IEEE Trans.Acoust.Speech and Signal Proc., 22 (1974), 330-338

M343 Mathews MV, Miller JE, David EE
Pitch synchronous analysis of voiced sounds
J.Acoust.Soc.Amer., 33 (1961), 179-186

M349 Morris LR
Fast speech spectrogram reduction and display on minicomputer/graphics processors
IEEE Trans.Acoust.Speech and Signal Proc., 23 (1975), 297-300

M350 Mermelstein P
Computer generated spectrogram displays for on-line speech research
IEEE Trans.Audio and Electroacoust., 19 (1971), 44-47

M355 Miller RL
Performance characteristics of an experimental harmonic identification pitch extraction (HIPEX) system
J.Acoust.Soc.Amer., 47 (1970), 1593-1601

M367 Makhoul J
Linear prediction: a tutorial review
Proc.IEEE, 63 (1975), 561-580

M368 Markel JD, Gray AH
Linear prediction of speech
Springer-Verlag, New York (1976)

M371 Markel JD
The SIFT algorithm for fundamental frequency estimation
IEEE Trans.Audio Electroacoust., 20 (1972), 367-377

M373 Maksym JN
Real-time pitch extraction by adaptive prediction of the speech waveform
IEEE Trans.Audio Electroacoust., 21 (1973), 149-153

M384 McGonegal CA, Rabiner LR, Rosenberg AE
A semiautomatic pitch detector (SAPD)
IEEE Trans.Acoust.Speech Signal Proc., 23 (1975), 570-574

M386 McGonegal CA, Rabiner LR, Rosenberg AE
A subjective evaluation of pitch detection methods using LPC synthesised speech
IEEE Trans.Acoust.Speech Signal Proc., 25 (1977), 221-229

M404 Maitra S, Davis CR
A speech digitizer at 2400 bits/s
IEEE Trans.Acoust.Speech Signal Proc., 27 (1979), 729-733

M410 Moschytz GS
Miniaturized RC filters
Bell Syst.Tech.J., 44 (1965), 823-870

M414 Millman J
Microelectronics: digital and analog circuits and systems
McGraw-Hill, New York (1979)

M421 MOS Technology Inc.
MCS6500 microcomputer family hardware manual
MOS Technology Inc., Norristown PA (1976)

M422 Motorola Inc.
M6800 microprocessor programming manual
Motorola Inc., Geneva (1977)

M423 Motorola Inc.
M6800 microprocessor application manual
Motorola Inc., Geneva (1977)

M434 Malmstadt HV, Enke CG, Crouch SR
Electronic measurements for scientists
W.A.Benjamin, Menlo Park, CA (1974)

M606 MOS Technology Inc.
MCS6502 microcomputer family programming manual
MOS Technology, Norristown, PA (1976)

N143 Nickerson RS, Stevens KN
Teaching speech to the deaf: can a computer help?
IEEE Trans.Audio Electroacoust., 21 (1973), 445-455

N322 Niederjohn RJ
A mathematical formulation and comparison of zero-crossing analysis techniques which have been applied to automatic speech recognition
IEEE Trans.Acoust.Speech and Signal Proc., 23 (1975), 373-380

N356 Noll AM
Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate
in Fox j (ed), 'Computer processing in communications proceedings', Polytechnic Press, New York (1969)

N359 Noll AM
Short-time spectrum and 'cepstrum' techniques for vocal-pitch detection
J.Acoust.Soc.Amer., 36 (1964), 296-302

N360 Noll AM
Cepstrum pitch determination
J.Acoust.Soc.Amer., 41 (1967), 293-309

N433 Niederjohn RJ, Stick PP
A computer interface for efficient z/c interval measurement
IEEE Trans.Computers, 24 (1975), 329-331

N802 Nagle HT(Jr), Nelson VP
Digital filter implementation on 16-bit microcomputers
IEEE Micro, 1 (1981), 23-32

N803 Nelson VP, Nagle HT(Jr)
Digital filtering performance comparison study of 16-bit microcomputers
IEEE Micro, 1 (1981), 32-41

O127 O'Connor JD, Arnold GF
Intonation of colloquial English
Longman, London (1973)

O351 Oppenheim AV
Speech spectrograms using the fast Fourier transform
IEEE. Spectrum, 7 (1970), 57-62

O358 Oppenheim AV, Schafer RW, Stockham TG
Nonlinear filtering of multiplied and convolved signals
Proc.IEEE, 56 (1968), 1264-1291

O361 Oppenheim AV, Schafer RW
Homomorphic analysis of speech
IEEE Trans.Audio Electroacoust., 16 (1968), 221-226

O407 Oppenheim AV, Schafer RW
Digital signal processing
Prentice-Hall, Englewood Cliffs NJ (1975)

O607 Osborne A
An introduction to microcomputers - Vol II: some real products
Osborne & Associates, Berkeley, California (1978)

O610 Osborne A
6800 programming for logic design
Osborne & Associates, Berkeley, California (1977)

P106 Pask G
The cybernetics of human learning and performance
Hutchinson, London (1975)

P114 Pickett JM
The sounds of speech communication
University Park Press, Baltimore (1980)

P119 Potter S
Modern linguistics
Deutsch, London (1957)

P148 Pronovost W
Developments in visual displays of speech information
Volta Review, 69 (1967), 365-373

P149 Pickett JM
Recent research on speech-analyzing aids for the deaf
IEEE Trans.Audio Electroacoust., 16 (1968), 227-234

P150 Pickett JM
Some applications of speech analysis to communication aids for the deaf
IEEE Trans.Audio Electroacoust., 17 (1969), 283-289

P151 Pickett JM
Status of speech-analyzing communication aids for the deaf
IEEE Trans.Audio Electroacoust., 20 (1972), 3-8

P319 Peterson E
Frequency detection and speech formants
J.Acoust.Soc.Amer., 23 (1951), 668-674

P328 Pfeifer LL
Multiplication reduction in short-term autocorrelation
IEEE Trans.Audio and Electroacoust., 21 (1973), 556-558

P345 Potter RK, Kopp GA, Green HC
Visible Speech
van Nostrand, New York (1947)

P346 Prestigiacomo AJ
Plastic tape sound spectrograph
J.Speech Hear.Disorders, 22 (1957), 321-327

P602 Palmer PF
Structured programming techniques in interrupt-driven routines
ICL Tech.J., 1 (Nov 1979), 247-264

P605 Preston C
A hitch-hiker's guide to the PET
ACT Petsoft, Birmingham (1979)

R113 Rostron AB, Welbourn CP
A computer assisted system for the extraction and visual display of pitch
Behavior and Research Methods and Instrumentation, 8 (1976), 456-459

R123 Ridley F
A manual of elocution for teacher and student
Samuel French, London (1928)

R212 Rothenberg M
A new inverse-filtering technique for deriving the glottal air waveform during voicing
J.Acoust.Soc.Amer, 53 (1973), 1632-1645

R301 Rabiner LR
On the use of autocorrelation analysis for pitch detection
IEEE Trans.Acoust.Speech and Signal Proc., 25 (1977), 24-33

R311 Reddy DR
Segmentation of speech sounds
J.Acoust.Soc.Amer., 40 (1966), 307-312

R312 Reddy DR
Pitch period determination of speech sounds
Comm. ACM, 10 (1967), 343-348

R326 Rader CM
An improved algorithm for high speed autocorrelation with applications to spectral estimation
IEEE Trans.Audio and Electroacoust., 18 (1970), 439-441

R331 Reddy NS, Reddy VU
High-speed computation of autocorrelation using rectangular transforms
IEEE Trans.Acoust.Speech and Signal Proc., 28 (1980), 481-483

R335 Ross MJ, Shaffer HL, Cohen A, Freudberg R, Manley HJ
AMDF pitch extractor
IEEE Trans.Acoust.Speech and Signal Proc., 22 (1974), 353-362

R348 Rothauser E
Digitalised sound spectrograph using FFT and multipoint techniques
J.Acoust.Soc.Amer., 45 (1969), 308

R381 Rabiner LR, Cheng MJ, Rosenberg AE, McGonegal CA
A comparative performance study of several pitch detection algorithms
IEEE Trans.Acoust.Speech Signal Proc., 24 (1976), 399-418

R385 Rabiner LR, Sambur MR, Schmidt CE
Applications of a nonlinear smoothing algorithm to speech processing
IEEE Trans.Acoust.Speech Signal Proc., 23 (1975), 552-557

R406 Rabiner LR, Gold B
Theory and application of digital signal processing
Prentice-Hall, Englewood Cliffs NJ (1975)

R412 RCA Solid State Division
COS/MOS micropower phase-locked loop
RCA, Somerville NJ, Data Sheet No.1099 (1978)

R413 Reintjes P
Phase-locked generator converts, filters most inputs
Electronics, 51 (Feb 1978), 113

R429 Rogers B
Fast Fourier tranforms
Practical Computing, 3, No.12 (Dec 1980), 91-93

R501 RS Components Ltd.
Data Sheet R/2983 - electronic attenuator 306-803
RS Components Ltd., London (Dec 1977)

R503 RS Components Ltd.
Data Sheet R/2911 - 8 bit D to A/A to D converter IC
RS Components Ltd., London (Mar 1977)

S110 Smith WM, McCrary JW, Smith KU
Delayed visual feedback and behaviour
Science, 132 (1960), 1013-1014

S111 Smith WM
Feedback - real-time delayed vision of ones own tracking behaviour
Science, 176 (1972), 339-340

S154 Stratton WD
Intonation feedback for the deaf through a tactile display
Volta Review, 76 (1974), 26-35

S214 Strevens P
Spectra of fricative noise in human speech
Language and Speech, 3 (1960), 32-49

S223 Stevens KN, House AS
An acoustical theory of vowel production and some of its implications
J.Speech Hear.Res., 4 (1961), 303-320

S226 Stevens KN, House AS
Studies of formant transitions using a vocal tract analog
J.Acoust.Soc.Amer, 28 (1956), 578-585

S227 Schafer RW, Rabiner LR
Digital representations of speech signals
Proc.IEEE, 63 (1975), 662-677

S303 Schwartz M, Shaw L
Signal processing: discrete spectral analysis, detection and estimation
McGraw-Hill Kogakusha, Tokyo (1975)

S333 Sondhi MM
New methods of pitch period detection
IEEE Trans.Audio and Electroacoust., 16 (1968), 262-266

S337 Schafer RW
Design and simulation of digital filter banks for speech analysis
J.Acoust.Soc.Amer., 51 (1972), 110

S338 Schafer RW, Rabiner LR
Design of digital filter banks for speech analysis
Bell Syst.Tech.J., 50 (1971), 3097-3115

S340 Singleton RC
A short bibliography on the fast Fourier transform
IEEE Trans.Audio and Electroacoust., 17 (1969), 166-169

S352 Silverman HR, Dixon NR
A parametrically controlled spectral analysis system for speech
IEEE Trans.Acoust.Speech and Signal Proc., 22 (1974), 362-381

S354 Schroeder MR
Period histogram and product spectrum: new methods for fundamental frequency measurement
J.Acoust.Soc.Amer., 43 (1968), 829-834

S377 Siegel LJ
A procedure for using pattern classification techniques to obtain a voiced/unvoiced classifier
IEEE Trans.Acoust.Speech and Signal Proc., 27 (1979), 83-88

S382 Schafer RW, Rabiner LR
System for automatic formant analysis of voiced speech
J.Acoust.Soc.Amer, 47 (1970), 634-648

T120 Trager GL, Smith HL
An outline of English structure
American Council of Learned Studies, Washington, Studies in Linguistics, Occasional paper no.3 (1951)

T213 Titze I, Talkin D
A theoretical study of the effects of the various laryngeal configurations on the acoustics of phonation
J.Acoust.Soc.Amer, 66 (1979), 60-74

T304 Taub H, Schilling DL
Principles of communication systems
McGraw-Hill Kogakusha, Tokyo (1971)

T314 Tucker WH, Bates RHT
A pitch estimation algorithm for speech and music
IEEE Trans.Acoust.Speech and Signal Proc., 26 (1978), 597-604

T372 Texas Instruments Inc.
Solid State Speech products and services from Texas Instruments
Texas Instruments Inc., Austin, Texas (1980)

T415 Tobey GE, Graeme JG, Huelsman LP
Operational amplifiers - design and applications
McGraw-Hill Kogakusha, Tokyo (1971)

U379 Un CK, Lee HH
Voiced/unvoiced/silence discrimination of speech by delta modulation
IEEE Trans.Acoust.Speech and Signal Proc., 28 (1980), 398-407

V102 Voice Identification Inc.
Series 700 Sound Spectrograph Operating and Maintenance Manual
Voice Identification Inc. (1974)

W109 Wargo MJ
Delayed sensory feedback in visual and auditory tracking
Perceptual and Motor Skills, 24 (1967), 55-62

W138 Ward IC
Defects of speech - their nature and cure
Dent, London (1923)

W140 Wechsler E
Laryngographic study of voice disorders
in [F157] (1976)

W145 Willemain TR, Lee FF
Tactile pitch displays for the deaf
IEEE Trans.Audio Electroacoust., 20 (1972), 9-17

W309 West J, Goonewardane M, Hayers G, Fourcin AJ
The Laryngograph and Voiscope
in [F157] (1976)

W362 Weiss MR, Vogel RP, Harris CM
Implementation of a pitch extractor of the double-spectrum-analysis type
J.Acoust.Soc.Amer., 40 (1966), 657-662

W365 Wood CA
Computer-generated spectrograms and cepstrograms
J.Acoust.Soc.Amer., 51 (1972), 132

W375 Witten IH
Algorithms for adaptive linear prediction
Comp.J., 23 (1980), 78-84

Z402 Zeman J, Nagle HT
A high-speed microprogrammable digital signal processor employing distributed arithmetic
IEEE Trans.Computers, 29 (1980), 134-144

Z420 Zilog Inc.
Zilog Z-80 family technical overview
Zilog Inc., Cupertino, CA (Aug 1980)