1

1.0

2.5

2.2

1.1

2.0

1.8

1.25

1.4

1.6

MicroD

3.5 Formant Extraction via LPC	42
3.6 Median Smoothing Pitch and Formant Contours	43
3.7 Limitations of LPC Modelling of Speech	44

4. FIRST GENERATION FEATURE EXTRACTION SPEECH PROCESSOR

4.1. Development and Target Hardware	47
4.1.1. The TI EVM	48
4.1.2. The CRC Board	48
4.1.3. The Stimulator Interface	51
4.1.4. The Sherbrooke Stimulator	52
4.1.5. The Carleton Electrode Array	55
4.2. Implementation of the SIFT algorithm	56
4.3. Implementation of Formant Extraction via LPC Spectra	64
4.4. System Software Description	65
4.4.1. Initial State of the System Software	66
4.4.2. Data Considerations	67
4.4.3. The Main Program	70
4.4.4. INPUT -- The Interrupt Service Routine	71
4.4.5. The Subroutines	73
4.4.6. Proposed Electrode Stimulation Algorithms	83
4.5. Feature Extraction on the CRC Board	85
4.6. System Operation and Timing	88
4.7. Summary	90

5. THE TMS320E17-BASED FEATURE EXTRACTION SYSTEM

5.1 Development and Target Hardware	92
--	-----------

**A Programmable Digital Signal Processor-Based
Speech Feature-Extraction System
for a Modular Cochlear Implant Prosthesis**

by

Peter A. Barszczewski, B. Eng.

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
Master of Engineering

Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
September, 1988

© copyright

1988, Peter A. Barszczewski, B. Eng.

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-54405 8

Carleton University
Ottawa, Canada K1S 5B6

The undersigned hereby recommend to the Faculty of Graduate Studies and
Research acceptance of the thesis,

**"A Programmable Digital Signal Processor-Based Speech Feature-Extraction
System for a Modular Cochlear Implant Prosthesis"**

submitted by Peter A. Barszczewski, B. Eng., in partial fulfillment of the requirements
for the degree of Master of Engineering.

Thesis Supervisor

Chairman,
Department of Systems and Computer
Engineering

Department of Systems and Computer Engineering

Faculty of Engineering

Carleton University

September, 1988

ABSTRACT

This thesis documents the development of three separate speech processors for a modular cochlear implant prosthesis. The purpose of the implant is to provide certain individuals who are profoundly deaf with an awareness of sound and to help them communicate. All processors extract the speech features: pitch, using the LPC-based simplified inverse filter tracking algorithm (SIFT); and formants, by LPC spectral envelope, in real-time. The first speech processor's system software, implemented on a TMS32010-based board requiring fourteen chips and approximately 1.6 watts, was transferred to a TMS320E17-based board requiring only three chips and approximately 0.5 watts. This second processor, limited by stringent program and data memory constraints, was abandoned. The system software was then implemented in a TMS320C25-based development environment where both performance and accuracy were increased. Test results using both formant-synthesized and real speech are provided for the TMS320C25-based system.

ACKNOWLEDGEMENTS

I would like to thank a few people who have made my life more bearable during this hell we call graduate studies. First and foremost, I would like to thank my supervisor, Dr L.R. Morris, for his support and encouragement given to me throughout this project. I would also like to thank Alan, George, Greg and Wilf for helping me miss deadlines, and generally not get any work done through extended lunch hours, dinners and parties, Dave Sword and Danny Lemay for their assistance in building some of the development hardware used in this project, Lois Mermelstien for helping edit the manuscript, and Jon Bosloy for suggesting this project in the first place.

Finally, I would like to thank and dedicate this work to my parents for their love, encouragement and support (and money) given to me while I complained and whined about how difficult life is.

1. INTRODUCTION TO COCHLEAR PROSTHESES

1.1 Introduction	1
1.2 Goals for the Speech Processor System	4
1.3 Achievements	7
1.4 Outline of Thesis	9

2. BACKGROUND ON HEARING AND SPEECH

2.1 The Sensation of Hearing	13
2.1.1 The Tonotopic Nature of the Cochlea	15
2.1.2 The Temporal Nature of the Cochlea	19
2.1.3 Loudness	21
2.2 The Speech Model	23
2.3 Speech Processing Strategies for Cochlear Implants	26
2.4 Conclusion	29

3. FEATURE EXTRACTION ALGORITHMS

3.1 Linear Predictive Coding	31
3.2 Determining LPC Parameters by the Autocorrelation Method	32
3.2.1 The Inverse Filter Formulation	33
3.2.2 Implementing the Autocorrelation Algorithm	35
3.3 The Simplified Inverse Filter Tracking (SIFT) Algorithm	38
3.4 Voiced/Unvoiced Decisions	41

3.5 Formant Extraction via LPC	42
3.6 Median Smoothing Pitch and Formant Contours	43
3.7 Limitations of LPC Modelling of Speech	44

4. FIRST GENERATION FEATURE EXTRACTION SPEECH PROCESSOR

4.1. Development and Target Hardware	47
4.1.1. The TI EVM	48
4.1.2. The CRC Board	48
4.1.3. The Stimulator Interface	51
4.1.4. The Sherbrooke Stimulator	52
4.1.5. The Carleton Electrode Array	55
4.2. Implementation of the SIFT algorithm	56
4.3. Implementation of Formant Extraction via LPC Spectra	64
4.4. System Software Description	65
4.4.1. Initial State of the System Software	66
4.4.2. Data Considerations	67
4.4.3. The Main Program	70
4.4.4. INPUT -- The Interrupt Service Routine	71
4.4.5. The Subroutines	73
4.4.6. Proposed Electrode Stimulation Algorithms	83
4.5. Feature Extraction on the CRC Board	85
4.6. System Operation and Timing	88
4.7. Summary	90

5. THE TMS320E17-BASED FEATURE EXTRACTION SYSTEM

5.1 Development and Target Hardware	92
--	-----------

5.1.1 The CRC Board-Based Target System	93
5.1.2 The Modified EVM Development System	96
5.2 System Software Modifications	98
5.2.1 Operations Between Frames	99
5.2.2 Additional Algorithm Modifications	102
5.2.3 Conditional Assembly	104
5.3 System Operation and Timing	105
5.4 Summary	107

6. THE TMS320C25-BASED FEATURE EXTRACTION SYSTEM

6.1. The Development System	109
6.1.1. The TMS320C25 Digital Signal Processor	110
6.1.2. The TMS320C25 Software Development System and Analog Interface Board	112
6.1.3. Developing a Target System	113
6.2. Incompatibilities Between the TMS320C25 and C10	116
6.3. System Software Enhancements	117
6.4. Software Speed/Space Optimization	120
6.4.1. Data Allocation and Initialization	120
6.4.2. Register-Based Modifications	123
6.4.3. The New System Control Loop	123
6.4.4. The Interrupt Service Routine	124
6.4.5. The Repeat Feature and the Autocorrelation Routine	125
6.5. System Operation and Timing	126
6.6. Error Analysis	128
6.7. Summary	140

7. AREAS FOR FUTURE WORK

7.1 Using the TI Assembler and Linker	142
7.2 Formant Tracking	143
7.3 Voiced-Unvoiced Decisions	144
7.4 Dealing with Noise	145
7.5 Hardware Considerations	146
7.6 Conclusions	147
Bibliography	149
Appendices	155

LIST OF FIGURES

Figure 1.1 - Cochlear Implant Hardware taken from [Har87b]	2
Figure 2.1 - The Human Ear, taken from [Moo82]	14
Figure 2.2 - Schematic sketch of the cochlea, stretched out from its actual spiral form, catching a mid-frequency traveling wave at a particular instant, taken from [Gei87]	17
Figure 2.3 - Schematic diagram of a stretched-out cochlear partition as seen from above	20
Figure 2.4 - Post-stimulus-time histograms for single auditory nerve fiber	22
Figure 4.1 - CRC board contains TMS320C10, PML parallel/serial interface, codec, 4K program RAM, downloading ROM, op-amp chip, RS232 interface, taken from [Sky86]	49
Figure 4.2 - The Sherbrooke Stimulator. (a) Block diagram of the stimulation module. (b) the command word format, taken from [Duv82]	54
Figure 4.3 - Block Diagram of the Feature Extraction Programs. (a) the pitch extraction program, (b) the formant extraction program	56
Figure 4.4 (a) through (f) - Real-time speech waveforms and diagnostic signals as seen on an oscilloscope	60
Figure 5.1 - Functional Block Diagram of the TMS320E17 Digital Signal Processor, taken from [TMS87]	94
Figure 5.2 - The CRC-board layout diagram illustrating the cutting which produces the E17-board (actual size)	95
Figure 5.3 - Data memory map versus time for the TMS320E17 system software	100
Figure 6.1 - Functional Block Diagram of TMS320C25 Digital Signal Processor, taken from [TMS86b]	111
Figure 6.2 - Functional Block Diagram of C25-Based Target System proposed in conjunction with Skywave	115
Figure 6.3 - Block diagram of the feature extraction program on the C25. (a) pitch extraction, (b) formant extraction	118
Figure 6.4 (a) through (e) - Five examples of LPC spectrum calculations with the TMS320C25 system software on synthetic speech	129
Figure 6.5 (a) through (f) - Five examples of formant extraction using the TMS320C25 system software on real speech	135

LIST OF TABLES

Table 4.1 - Map of program and data memory for the system software running on the CRC board	68
Table 4.2 - Memory module execution time and storage space for the system software running on the CRC board	89
Table 5.2 - Program module execution time and code space for the system software running on the E17 board	106
Table 6.1 - Data memory map for feature extraction system software running on the TMS320C25	122
Table 6.2 - TMS320C25 program module execution time and code space for the feature extraction system software running on the SWDS board at 20 MHz	127

LIST OF APPENDICES

Appendix A - TMS32010 source code listing of the feature extraction system software for the EVM and CRC board	156
Appendix B - TMS32010 source code listing Test demonstrating the CRC board to Sherbrooke stimulator interface	198
Appendix C - TMS320E17 source code listing of the feature extraction system software for the E17-board and test bed	201
Appendix D - TMS320C25 source code listing of the feature extraction system software for the SWDS	237

1. INTRODUCTION TO COCHLEAR PROSTHESES

1.1 Introduction

A cochlear implant is a neural prosthetic device that delivers electrical stimuli to the auditory nerve (the VIII cranial nerve). The purpose of the implant is to provide certain individuals who are profoundly deaf with an awareness of sound and to help them communicate. This is done by transforming sound into electrical signals that create auditory perception upon direct application to the auditory nerve. The device is intended for people who have a type of profound deafness caused by the destruction of the sensory hair cells either by disease or by certain life saving antibiotics [Har87a]. That is, the normal mechanical stimulation of neurons by the motion of hair cells, which are in turn displaced via the motion of the basilar membrane in response to sounds impinging on the outer ear, is not possible since the hair cells are inoperative. This means that normal sound transmission is impossible, and there is no benefit from conventional hearing aids. However, the auditory nerve can still be electrically stimulated by electrodes implanted within the cochlea as long as a population of electrically excitable auditory nerve fibres exists and is accessible for stimulation.

A cochlear prosthesis system is complex, but can be thought of as comprising four functional components: an external microphone with signal conditioning circuits, a signal processor, a stimulator, and an electrode or array of electrodes that excites the auditory nerve (Figure 1.1).

The external microphone, usually worn on the implantee's lapel, picks up sound and sends it to the signal processor, also worn externally. Before the signal is input to the

Figure 1.1 - Cochlear Implant Hardware taken from [Har87b].

processor some signal conditioning may occur. This may include automatic gain control to maintain the signal RMS value at an approximately constant level, syllabic compression to minimize distortion of speech signals imposed by the automatic gain control, pre-emphasis to enhance signal characteristics, and noise suppression. This signal conditioning does not necessarily have to be performed by analog circuitry, it can also be performed in software by a microprocessor.

The signal processor's main function is to code the sounds passed to it by the microphone and signal conditioning circuits into appropriate electrical signals to be applied to the auditory nerve by the stimulator.

The sounds the implantee perceives with present devices can be described as crude at best because implant technology and understanding of how neural information is processed by the brain are both still in their infancy. As a result, researchers have concentrated on providing processing especially appropriate for speech signals, since they are the basis for human communication.

The signal processor sends the appropriately coded speech information, in either analog or digital form, to a transmitting induction coil worn behind the ear (much like a normal hearing aid). This induction coil transmits the coded speech information to the stimulator via a transcutaneous electromagnetic induction link. A receiving induction coil is incorporated into the subcutaneous stimulator which is implanted within the mastoid bone (the bone directly behind the external ear). The stimulator electrically excites the electrodes with a mapped or transformed version of the information produced by the speech processor.

The electrodes, which are directly connected to the stimulator, are implanted within the cochlea itself. They excite the auditory nerve endings with the electrical

signals from the stimulator. Previous implant systems used either a single electrode or an electrode array, but it has become apparent that an electrode array, or multichannel system, is needed for speech recognition [Mic85].

The speech processor described in this thesis was developed as part of a multi-university project, funded by the Medical Research Council of Canada (MRC) and the National Science and Engineering Research Council (NSERC) [Har87b], [vdP87]. The project's major components include the programmable signal processor (developed under the direction of Dr. L. Robert Morris and the subject of this thesis), an implantable receiver/stimulator (developed under the direction of Dr. Francois Duval at Sherbrooke University) [Duv82], and a mass producible electrode array (developed under the direction of Dr. Patrick van der Puije at Carleton University) [vdp86].

1.2 Goals for the Speech Processor System

The two most-widely used processing strategies for multi-channel auditory prostheses are the compressed-analog-outputs strategy, used in the UCSF/Storz [Mer85] and Utah/MIT/Symbion [Edd86] devices, and the F_0/F_2 strategy of the Melbourne/Nucleus device [Tong87]. In describing the operation of these devices, some speech and hearing related terms are unavoidably introduced. These are described at length in Chapter 2.

The compressed-analog-outputs strategy maps the outputs of four bandpass filters to the implanted electrode array. The frequency bands covered by each filter are selected to correspond to the place frequency that its electrode is located at along the basilar membrane.

The F_0/F_2 strategy is designed to represent detected approximations to the speech parameters pitch (F_0) and second formant frequency (F_2). This is done by selecting a pair of electrodes from a multi-channel array according to the frequency of detected F_2 , where the position of the selected pair corresponds to the tonotopic ranking of pairs across the electrode array, and then delivering to the selected pair pulsatile stimuli at a rate proportional to the detected F_0 . The pulse amplitudes are computed using a loudness mapping function.

It became apparent early in this project (1984) that the Nucleus technique was achieving the most successful results [Har87a]. Further, its goals of using F_0 and F_2 were well understood within the context of well established speech models. It was also believed -- in the context of the formant vocoder model -- that extensions of the F_0/F_2 features to the full $F_0/F_1/F_2/F_3$ complex would yield better results.

Thus, at a relatively early date, this project was committed to attempt accurate $F_0/F_1/F_2/F_3$ complex extraction. The strategy's validity was confirmed relatively recently by the research group at the University of Melbourne (1987),

"As far as the selection of a speech processing strategy for cochlear implant patients is concerned, $F_0/F_1/F_2/F_3/B$ appears to be the most promising candidate." [Tong87].

Further, the group at Research Triangle Institute (1986) indicated that additional gains in speech understanding may be obtained with the speech spectral envelope (which would be produced indirectly through our formant extraction calculations),

"Our findings further indicate that (a) explicit extraction of formant frequencies is not essential to good performance and (b) a somewhat more-dense representation of speech spectra than that used in the $F_0/F_1/F_2$ strategy [e.g., the spectral envelope] might produce additional gains in performance, especially for consonants" [Wil86].

The appearance of the Texas Instruments TMS32010 digital signal processor chip in 1983 suggested that implementation of two well known speech feature extraction algorithms, formants via LPC envelope extraction and pitch via the LPC-based SIFT pitch extraction algorithm (both to be described in detail in Chapter 3), if successful, would accurately yield the desired feature extraction in a portable package. Further, it was felt that a software solution implemented on a microprocessor, which would probably have faster software-compatible successors, offered greater flexibility. This would not only allow for the evolution of more accurate versions of the feature extraction algorithms, but would allow progressively more of the pre-processing (automatic gain control, syllabic compression) and post processing (data preparation for the stimulator) to be performed within the single microprocessor. The ultimate goal would be a "three chip" system consisting of an analog front-end (e.g., codec or analog-to-digital converter), a microprocessor, and a microprocessor-to-stimulator interface (modulator).

It should be noted that while a preliminary analysis of the time-space requirements (done by Dr. L. Robert Morris in 1983-84) for implementing the formant and pitch extraction algorithms simultaneously on the TMS32010, suggested that it was possible, it was by no means certain. Similarly, the utility of the accurate, digitally extracted $F_0/F_1/F_2/F_3$ complex in achieving better results than the Nucleus F_0/F_2 analog extracted complex was not guaranteed in the 1983-84 time frame. However, because the LPC parameters, determined during formant extraction, together with pitch, embody an accurate, low bit rate model of the entire speech waveform, it was understood that should

the $F_0/F_1/F_2/F_3$ strategy fail, other mappings of the speech model (obtained from the available LPC parameters) could be made in response to feedback from actual testing.

1.3 Achievements

The results reported in this thesis covering work done from September 1986 to May 1988, include:

- 1) The successful implementation and initial testing of the combined formant/pitch extraction algorithms on the TMS32010, the original member of the TMS320 family.
- 2) The migration of the software, with some operational restrictions, to the TMS320E17, which essentially reduces the system from a 1.6 watts, 14 chip system occupying 177.5 cm² to a 0.5 watt, 3 chip system occupying only 100 cm².
- 3) The migration of a faster, enhanced accuracy version of the software to the TMS320C25, a newer member of the TMS320 family, thus allowing significant execution time and program space for the significant pre and post processing algorithms expected to be evolved.
- 4) The completion of detailed testing of the pitch and formant extraction system, on the TMS320C25, with synthesized and natural speech.

The work began with separate, untested, unoptimized, undocumented versions of the two feature extraction modules which had been mainly implemented by Dr. L. Robert Morris. Because of the severe data memory restrictions of the TMS32010, even the initial task of merging the two systems for concurrent real-time operation was non-trivial.

Indeed, at least one other cochlear implant research team suggested that the first achievement listed above would not be possible [Fin87].

Although delays in the developments of the other components of the intended all-Canadian system have not allowed testing of patients with our speech processor, recent results in experiments done by Stanford, RTI, and by Nucleus Corp. (with synthesized speech signals) show that our contention, that is the desirability and utility of real-time extraction of an accurate $F_0/F_1/F_2/F_3$ complex, have been borne out. Further, a significant comparatively recent development at Stanford University [Tow87] suggests that accurate tonotopic mapping of the multi-component $F_0/F_1/F_2/F_3$ complex (or even the entire spectral envelope) to the basilar membrane is indeed possible, since the problems of current spread due to high conductivity of the inner cochlear fluid have been solved.

At thesis presentation date, our results have been described at the following international forums:

- 1) IEEE International Conference on Acoustics, Speech, and Signal Processing, April 1987, Dallas [Mor87a].
- 2) Speech Technology Conference, April 1987, New York [Mor87b].
- 3) The Gordon Conference on Implantable Auditory Prosthesis, New London, New Hampshire, July 1987.
- 4) Ninth Annual IEEE Conference on Engineering in Medicine and Biology, November 1987, Boston [Mor87c].

- 5) IEEE International Conference on Acoustics, Speech, and Signal Processing, April 1988, New York [Mor88a].

A paper describing the overall results has been accepted for publication in the IEEE Transactions on Biomedical Engineering [Mor88b].

Finally, to attest to the utility of our advanced speech processor, interest in using and testing it with implant patients has been expressed by the following internationally recognized research groups:

- 1) The Stanford University Cochlear Prosthesis Project,
- 2) The "Speech Processors for Auditory Prosthesis" Project of the Neuroscience Program at the Research Triangle Institute and Duke University Medical Center, and
- 3) The Kresge Hearing Research Laboratory at Louisiana State University.

1.4 Outline of Thesis

The next chapter of this thesis, Chapter 2, contains background information on hearing and speech. The first section, on hearing, briefly explains how the ear works and the types of signals the auditory nerve generates. Included in this section is a comparison between the neural signals generated by acoustical and electrical stimulation. The next section describes a speech model, giving insight into the complex speech signal. Following is an overview on the types of speech processing performed for a cochlear implant -- either bandpass filter vocoder or feature extractor. The final section summarizes the points noted in the chapter, illustrating the design decisions made in developing our speech processor(s).

Chapter 3 describes the speech processing algorithms used by our processor. The first two sections give an overview on determining LPC parameters, i.e., prediction coefficients, from speech. These parameters are used by both the pitch and formant extraction algorithms. The next section describes the Simplified Inverse Filter Tracking (SIFT) algorithm used for pitch extraction and is followed by a section on using LPC parameters to extract formants. Finally, a description of some of the limitations involved in modelling speech with LPC, and hence using it for feature extraction, are given.

Chapter 4 describes our first generation speech processor based on the TMS32010 digital signal processor. This chapter extensively describes the software required to implement the feature extraction algorithms described in the previous chapter. Further, a description of the development and target systems used is given. A timing/space analysis concludes the chapter.

Chapter 5 describes a smaller, less power-hungry version of the first generation speech processor based on the TMS320E17 digital signal processor. The hardware and software modifications made to the original system are described in detail. This includes the development system built to assist in software debugging. A timing/space analysis of the system software concludes the chapter.

Chapter 6, describes our second generation speech processor, based on the TMS320C25 digital signal processor. This version is both slightly more accurate and faster executing than the original speech processor described in Chapter 4. A target board was never built but several were proposed, including a board made available to us by an Ottawa commercial firm. Included in this chapter is a summary of the feature extraction algorithm testing. More detailed testing was performed on this system than on the two previous systems due to the development environment's advanced debugging

capabilities. The testing was conducted with the aid of a formant synthesizer, which can synthesize speech with predetermined formant frequencies. This speech was subsequently analyzed with our system, thus allowing accuracy tests.

The final chapter, Chapter 7, details future research which I would like to see performed with respect to the speech processor which I would have liked to complete myself, if time had allowed.

2. BACKGROUND ON HEARING AND SPEECH

An action potential generated in a nerve fibre is a brief, stereotyped electrical event lasting for about 1 msec [Kuf84]. These signals are virtually identical in all nerve cells and are used by the brain to process all the information it receives and analyzes. The type of sensation perceived when a sensory nerve is stimulated is determined by the specific area in the central nervous system to which the nerve fibre leads. Thus, whether the auditory nerve is stimulated by normal "acoustical" means, or by a direct electrical signal, a perception of sound is created. This is the fundamental basis of the cochlear implant.

This chapter gives background information on hearing and speech processing. The first section discusses how the ear works and the neural responses which are elicited from acoustic and electrical stimulation. It is intended both to demonstrate some of the complexities of a cochlear implant system at a neural level and to show the limitations inherent in electrical stimulation which make it necessary to concentrate on speech processing instead of covering the full range of audible sounds. The next section, which describes a speech model, provides insight into speech signals and how they can be analyzed. The following section describes the types of speech processing strategies that have been used in cochlear implant systems and their limitations. A final section concludes the important points raised throughout the chapter explaining the design decisions that were made in developing our speech processor.

2.1 The Sensation of Hearing

The human auditory system is illustrated in Figure 2.1. In normal hearing, sound (air vibrations) travels through the auditory canal to the tympanic membrane, or ear drum, which vibrates in synchrony with the sound waves that strike it. These vibrations are transmitted through the middle ear by three small bones, called ossicles, to the cochlea, or inner ear. The smallest of the bones, the stapes, makes contact with a structure called the oval window, located at the base of the cochlea. The main purpose of the middle ear is to transform the sound waves from the air outside the tympanic membrane into pressure waves in the fluids that fill the cochlea. Impedance matching is provided between the disparate media to ensure efficient energy transfer.

The cochlea contains three parallel fluid-filled chambers, the scala vestibuli, scala media, and the scala tympani. The chambers are divided along the cochlea's length by two membranes, the basilar membrane and Reissner's membrane. The inner tip of the cochlea, opposite the oval window, is known as the apex. At the apex there is a small opening (the helicotrema) between the basilar membrane and the walls of the cochlea, so that fluid can flow between two chambers of the cochlea, the scala vestibuli and the scala tympani. Inward movement of the oval window causes a flow of fluid around the helicotrema. This results in a corresponding outward movement in a membrane covering a second opening in the cochlea, the round window. Thus, the oval window is an input port and the round window a pressure relief valve.

The motion of the oval window causes a pressure to be applied, by the fluids in the cochlea, along the whole length of the basilar membrane simultaneously. The fluid pressure on the basilar membrane displaces the sensory hair cells that lie along it. There are approximately 31,000 hair cells distributed along the basilar membrane, each of

Figure 2.1 - The Human Ear, taken from [Moo82].

which is connected to a nerve ending. Their displacement produces action potentials in the auditory nerve, which are sent to the central nervous system. It is the hair cells which transduce the mechanical movement of the basilar membrane into neural activity. The action potentials produced are what transmit the sensation of hearing to the brain.

The coding of acoustic signals into action potential firing patterns along the basilar membrane is very complex. Present knowledge of peripheral auditory coding cannot entirely specify what the firing patterns for complex signals should be, but there are several widely accepted theories with respect to the perception of pitch¹ and loudness. However, even the available specifications far exceed current abilities to precisely control the elicited nerve discharge patterns in either time or space with implanted electrodes, even though it is these complex firing patterns which a cochlear implant system must attempt to generate. As a result, knowledge of the complex neural response of a cochlea to normal acoustical stimulation, as well as its more primitive response to electrical stimulation by present-day electrodes is important in developing a cochlear implant system.

2.1.1 The Tonotopic Nature of the Cochlea

As sound vibrations are applied to the oval window by the stapes, they are transmitted to the fluids of the cochlea. The physical structure of the cochlea then causes a mechanical frequency decomposition of the complex waveforms.

An "unwound" cochlea is functionally equivalent to a fluid filled U-shaped tube, with the input port at one end and a pressure release membrane (the round window) at the other. There are two tubes, the scala tympani and scala vestibuli, coupled together by a

¹ The term pitch here refers to the conscious perception of sound frequency which could differ from the true vibratory frequency.

thin "elastic" membrane, the basilar membrane. Vibrations introduced by the stapes at the input port produce waves in the fluids of the cochlea that travel along the cochlear partition. This is illustrated in Figure 2.2.

Once the vibrations have entered the cochlea, it acts like a mechanical mass-stiffness transmission line. The cochlea is not uniform along its length; its stiffness decreases monotonically from the input port to the helicotrema (at the bottom of the U), thus causing the frequency decomposition of the waveforms inside. If a sine wave of frequency 1 kHz enters the cochlea, the stiffness of the cochlear partition, or basilar membrane, located at the input port dominates the cochlear-partition impedance, and the 1 kHz energy generated is carried away from the input. The further along the cochlea the wave progresses, the less stiff the partition becomes and subsequently the lower the resonance frequency of that section of the partition becomes. Eventually, the 1 kHz wave will reach a section of the partition that has a 1 kHz resonant frequency, and resonance will occur [Gei87]. The energy of the wave is absorbed in the resonance and will not propagate significantly further along the partition.

Thus, different frequencies will resonate at different locations along the cochlea. The cochlea can be crudely approximated by a linear system, so the response to a complex signal composed of many frequencies will cause many locations along the partition to vibrate simultaneously. The low frequency components will resonate near the apex (the helicotrema end), and the high frequency components will resonate at the base (the oval/round window end). It is in this way that the cochlea is "frequency-mapped". The auditory nerve fibres are positioned uniformly along the entire length of the basilar membrane (from stapes to helicotrema) and are thus situated to take advantage of the cochlea's spatial (tonotopic) frequency analysis. Each fibre is attached to a single sensory hair cell, which translates the mechanical vibrations along the basilar membrane

Figure 2.2 - Schematic sketch of the cochlea, stretched out from its actual spiral form, catching a mid-frequency traveling wave at a particular instant, taken from [Gei87].

into neural output. Thus, specific neurons will be activated by frequencies that cause vibrations in particular areas of the cochlea. This spatial organization of the cochlea is called the "place-pitch" principle.

In a normal cochlea, individual nerve fibres exhibit sharp frequency selectivity for various sound intensities. This frequency selectivity is due to the reasonably sharp mechanical resonances of the basilar membrane and the characteristics of the individual fibres. In contrast, direct electrical stimulation of the nerve fibres by implanted electrodes does not produce any spatially organized frequency selectivity. In order to restore spatial separation among nerve responses, it is necessary to prevent the electrical current from reaching those fibres that would not ordinarily respond to a given input sound. This requirement has prompted the use of arrays of electrodes, with each electrode located in the vicinity of the nerve fibre population it is intended to stimulate. Signal processing is then used to determine which electrode to activate at any moment, depending on the frequency composition of the incoming acoustic signal.

The spatial isolation of stimulus current is complicated by the fact that the cochlea is fluid filled, and a current flow between any two electrodes results in a potential field that spreads throughout the cochlea. This potential field has the effect of exciting neural sites that are in contact with it at different rates depending upon the intensity of the field at that location. One solution to this problem is to choose electrode geometries (source and sink locations) that produce fields which are strong enough locally to excite nearby neurons, but which fall off rapidly with distance so that the more distant fibres do not respond. The most promising geometry to date (which has not yet been implemented in human subjects) involves using a bipolar electrode pair (source and sink) at each stimulation site [Gei87]. The dipole is then rotated so that its axis is parallel to the target fibres along the basilar membrane. This is called a radial orientation. It is expected that

with this orientation, ten completely independent channels can be obtained along the basilar membrane.

A second method which can help reduce current spread for any given electrode configuration is a technique called deconvolution [Tow87]. The central idea behind this technique is to determine the function which maps current levels at the electrodes to excitation levels at each of the possible neural sites. This function can then be inverted to give electrode current patterns, each of which produces excitation at only one neural site. This technique is limited, however, by the fact that the function which maps current levels to excitation levels is determined by extremely time consuming psycho-acoustic tests which must be performed on each individual patient.

2.1.2 The Temporal Nature of the Cochlea

Although the location of nerve endings along the basilar membrane appears to play a central role in sound perception, the rate at which neurons fire is also important. That is, the encoding of pitch by the auditory system can be accomplished in a limited manner based on neural discharge rates that occur in synchrony with sound vibrations. Low frequency sounds can produce synchronized low frequency volleys of nerve discharges on a single nerve ending at a rate of between 50 and 300 Hz. These volleys are discriminable as pitch changes (Figure 2.3). Each of the six middle fibres shown in the diagram appear to be firing nearly randomly, but when they are summed together a definite temporal pattern exists [Gei87]. This pattern clearly carries information -- the distance between the peaks in the "sum" waveform yields an approximation to the stimulus frequency, regardless of individual nerve fibre discharge rates. Rates may be discriminable as high as 1000 to 2000 Hz, but differentiation of pitch deteriorates above 300 Hz [Fei87].

Figure 2.3 - Schematic diagram of a stretched-out cochlear partition as seen from above. The partition becomes wider and less stiff with distance from the input, taken from [Gei87].

The temporal nature of a normal cochlea is mimicked in a cochlear implant by the actual electrical waveforms applied to individual electrodes. These waveforms have the effect of determining the temporal impulse patterns that arise from the stimulated nerve fibres. Two different electrical waveforms are commonly used, pulsatile and sinusoidal.

In Figure 2.4, post-stimulus-time histograms, taken from [vdH87], illustrate auditory nerve fibre responses to both acoustic pulses (a loud "click" applied to the outer ear) and electrical current pulses. These histograms represent the probability of eliciting a nerve impulse from one fibre as a function of time after onset of stimulation. The electrical pulse yields a single spike whose latency is consistent from one stimulus to the next and is independent of location of the nerve fibre within the cochlea. It also does not show any delay from onset. The acoustic response is more variable in latency, and is multimodal, thus reflecting the filter ringing imposed by the inner ear (the system composed of the basilar membrane, nerve fibres, cochlear walls, etc). In contrast to acoustic sinusoids, electrical sinusoids result in excessive synchronization of impulses causing excitation to occur during both half cycles [vdH87]. Firing on both half cycles produces a spurious doubling of firing rate on the nerve fibre.

These examples of artificially stimulated nerve fibre behavior suggest that much of the richness of the acoustical response is lost when straightforward electrical stimuli are used. This is an important consideration in designing a speech processor if a reasonable perception of sound is to be attained.

2.1.3 Loudness

It is believed that a normal cochlea discriminates loudness in three separate ways [Fei87]. Firstly, an increase in the amplitude of the basilar membrane vibrations tends to increase the rate of neural firings, producing a perceived increase in loudness. Secondly,

Figure 2.4 - Post-stimulus-time histograms for single auditory nerve fiber. Left panel: 100 μ sec acoustic click, 500 presentations at 25 Hz. Right panel: 100 μ sec current pulse through bipolar intra-scalar electrodes, 100 presentations at 25 Hz. Bin width 50 μ sec. Ordinate in spikes/s calculated from $N/(\text{bin width})/(\# \text{ of stimulus repetitions})$, where N is the number of spikes in any bin. Propagation delay for acoustic clicks to reach the ear drum was 0.4 msec, taken from [vdH87].

the number of hair cells that are responding will increase as the area of vibration along the basilar membrane expands. This causes a "spatial summation" of impulses to occur, which is also perceived as an increase in loudness. Finally, there are certain specialized hair cells that do not cause a nerve ending to generate an action potential until vibrations reach high levels. It is suspected that these nerve endings transmit loudness information directly to the auditory nerve.

In a normal hearing patient, the auditory nerve compresses loudness perception by means of a cube-root law [Dar84]. This has the effect of allowing the discrimination of an extremely wide range of sound intensities. However, it has been found that the dynamic range for direct electrical stimulation of the auditory nerve is very limited compared with that of acoustic stimulation [Fei87]. Cochlear implant patients do not discriminate loudness levels with the same growth gradations as normal hearing subjects. Loudness growth with implants tends to increase linearly with electrical stimulus amplitude as compared to the cube-root with acoustical stimulation. Thus, a cochlear implant system should be designed to stimulate the auditory nerve in a manner that exploits the ability of the cochlea and central nervous system to discriminate intensity. This will assist the wearer to recognize the sound's source and information content.

2.2 The Speech Model

With the limitations involved in electrically exciting the auditory nerve, only a poor facsimile of normal hearing can be achieved by a cochlear implant. It simply cannot be expected that the full range of audible sounds could be perceived by an implantee. However, speech is the most important of sounds heard, as it is the basis for communication. As a result, speech processing is the main function of the signal

processor for cochlear implants. Before the various techniques used in speech processors can be discussed, an understanding of the nature of the speech waveform is important.

The vocal tract can be characterized as an acoustic tube that is terminated at one end by the vocal chords and at the other end by the lips. Another tube, the nasal tract, can also be connected to the main vocal tract by the movement of the velum at the back of the throat. Different sounds are produced by changing the shape of the vocal tract which is defined by the position of the articulators (lips, jaw, tongue, and velum).

There are two general categories of speech sounds which can be produced: voiced and unvoiced². Voiced sounds are produced by exciting the vocal tract with quasi-periodic pulses of air pressure, which are generated by air flowing through the vocal chords and causing them to vibrate. This category includes the majority of speech sounds such as vowels. Unvoiced sounds, such as fricatives, are produced by forcing air through a constriction in the vocal tract, causing turbulence which produces acoustic noise as the source of vocal tract excitation. In both cases, a wide-band excitation of the vocal tract is generated which in turn acts as a linear time-varying filter imposing its transmission properties on the source's frequency spectrum.

If the vocal tract shape is fixed, then the output of the system is the convolution of the excitation and the vocal tract impulse response. If the vocal tract shape changes slowly relative to the excitation, it is still reasonable to approximate the output, on a short-time basis, as a convolution of the excitation and vocal tract impulse response. Thus, the speech waveform is composed of segments during which the signal model properties remain approximately constant.

² This division does not include all possible speech sounds, such as mixed sounds which have both a voiced and unvoiced excitation of the vocal tract.

In modeling the sound production system, the excitation sources are represented separately: for voiced speech the filter is excited by a quasi-periodic pulse train corresponding to the fundamental frequency (F_0), and for unvoiced speech the filter is excited by white noise. The speech waveform is the output of the time-varying filter, approximating both the transmission properties of the vocal tract and the spectral properties of the glottal pulse shape. The speech waveform changes slowly enough that the filter has a "fixed" configuration for time intervals on the order of ten to fifty milliseconds. Thus, if the vocal tract were to be modelled by a digital filter, its coefficients would be updated about every ten to fifty milliseconds.

The frequency response of the vocal tract is a relatively smooth function of frequency, since it is an acoustic cavity. The frequency response is characterized primarily by resonances, called formant frequencies (F_1 , F_2 , F_3), corresponding to the cavity's resonant frequencies. Formants are manifested in the spectral domain by energy maxima at the resonant frequencies. For voiced speech, the quasi-periodic excitation's Fourier transform is composed of a line spectrum with harmonics spaced one fundamental frequency apart and an envelope reflecting the glottal pulse shape. Because the speech waveform is the convolution of the excitation and vocal tract impulse response, its spectrum is the product of their Fourier transforms. As the vocal tract shape changes to produce different sounds, the envelope of the composite speech spectrum also changes. Similarly, as the excitation period changes for voiced sounds, the spacing of the pitch harmonics will change and the envelope will reflect any change in the glottal pulse shape. Therefore it is necessary to view the spectrum on a short-time basis, where the time window is small enough that the features remain relatively constant within it.

Historically, the main features of interest in speech processing have been pitch, formant positions, and energy, as they together characterize voiced phonemes in

particular. The simplest view of speech perception maintains that speech is composed of a series of sounds or acoustic features, and that each feature, or set of features, corresponds to a particular phoneme. Thus, the speech sounds would have a one-to-one correspondence with the phonemes, and a sequence of speech sounds would be perceived as a sequence of phonemes, which would then be combined into words and phrases. Even though this view is limited (for example, it does not take into account the importance of transitions between phonemes [Moo82]), it provides a starting point for speech processing systems.

2.3 Speech Processing Strategies for Cochlear Implants

There are two speech coding strategies which have arisen for use with cochlear implants, feature extractors and vocoders³.

A feature extractor identifies salient aspects of the speech signal, analyzes those aspects, and delivers stimuli specifically related to those features. The delivered features may include pitch⁴ for voiced speech, the spectral envelope of the speech signal, formants frequencies and (possibly) amplitudes, and energy.

A vocoder attempts to recreate spatial firing patterns by approximating those originating from an intact cochlea with analogous analog stimuli. It samples the acoustic input through a series of bandpass filters, each filter approximating the frequency range of its electrode's site along the basilar membrane. The spectral energy at each of the channel outputs is mapped to the appropriate stimuli, then delivered to the electrodes.

³ The term vocoder in the cochlear implant field is misleading because it does not refer to the entire class of voice-coders, but a single style of speech processor which uses filter banks.

⁴ The term pitch here refers to the rate at which the vocal chords vibrate for voiced speech.

The two speech coding strategies reflect two schools of thought. One group, in recognition of the limited information transfer which is possible with implants, advocates extracting predetermined speech features and inputting this information unambiguously. The other group believes that too much should not be presupposed about what are the relevant cues in speech, especially if the rationale is based on normal psychoacoustics, and prefers to allow the central auditory system to do its own processing with whatever information one can squeeze in. In this case, speech and environmental sounds are passed through bandpass filters and the appropriate cochlear regions are stimulated with the output of each filter.

However, vocoders are based on a spectral analysis of incoming sound. One problem with speech is that the pitch of voice is convolved with the impulse response of the vocal tract. This causes the pitch and the lower formants (first and second) to strongly interact. With only a small number of electrodes or frequency-channels being stimulated within the cochlea and with current spreading between present day electrodes, this would have the effect of blurring both pitch and formant information necessary for the discrimination of speech. Thus, for a better perception of speech with present implant technology, the pitch and vocal tract spectral envelope should be separated and provided separately to the electrodes in some manner.

One manner of providing speech features to electrodes has been demonstrated in recent psychophysical studies on cochlear implant patients (as summarized in [Tong87]). This method suggests that electrical signal repetition rate can encode the fundamental frequency, F_0 , of the acoustic speech signals, electrode positions can encode formant frequencies, and electrical current intensity can encode the acoustic intensity of speech. Further, it has been demonstrated that, with synthesized speech, a speech processing

strategy which incorporates F_0 , F_1 , F_2 , and F_3 appears to be most promising when compared to more limited strategies which use a subset of these features [Tong87].

Unfortunately the speech strategies employed by Tong et. al. are limited in two ways. First, they used a bandpass technique to extract their speech features. This method has the same problem as the vocoder strategy, in that pitch and formant information strongly interact causing inaccurate results. Second, they mapped formants to a fixed range of electrodes. In their most promising strategy, F_1 was mapped to electrodes 4 to 9 (acoustic frequencies 1.8 kHz to 3.2 kHz), F_2 to electrodes 9 to 18 (acoustic frequencies 3.2 kHz to 8.3 kHz) and F_3 to electrodes 18 to 20 (acoustic frequencies 8.3 kHz to 10.3 kHz)⁵. The problems imposed with this range are that the formants are mapped to unnaturally high frequencies, and even with this range, formants should be allowed to cross these forced boundaries. This suggests that more accurate features need be extracted and a more flexible approach to stimulating the electrodes is needed.

A final note with respect to the actual implementation of any speech processing strategy should be made. The ideas of perception of pitch and loudness, for both acoustical and electrical stimuli, are not completely understood with regard to cochlear implants. It is for this reason that any implant system should be designed to be flexible, so as to take advantage of new developments in audiophysiology and psychoacoustics. To further support the argument for speech processor adaptability, patients who require implants often have abnormal cochleae, some with very poor neural survival. This could mean that a "place-pitch" strategy in general may not be useful, or that only certain electrodes will be discriminable. The system should be flexible enough to be

⁵ They were using a 22 channel electrode array with the numbering of electrodes from 1 to 22 in an apical to basal direction. The electrode array covered a range of acoustic frequencies from 1 kHz to 16 kHz.

3.2.2 Implementing the Autocorrelation Algorithm

Determining the prediction coefficients starts with a frame (or window) of speech samples, x_n , where $n = 0$ to $N-1$, which, according to the speech model, corresponds to the vocal tract impulse response convolved with the excitation function. The window size N should be large enough to allow sufficient spectral resolution but small enough to ensure that significant changes in the vocal tract structure over the period of measurement do not occur. Typically three or four pitch periods should be included in the window², which for the average male speaker means a window length of about 32 ms. Consequently, the number of samples, N , is a simple function of the window length and sampling rate.

The average spectrum of voiced speech falls with a frequency of 6 dB/octave due to the glottal wave shape and the lip radiation characteristics [Gray74]. In order to decrease the dynamic range required to represent the spectrum and reduce the probability of numerically caused instabilities, it is desirable to counteract this fall-off. A suitable pre-emphasis filter, which is easily implemented, is the first order differencing filter $1 - \mu z^{-1}$, where μ is called the pre-emphasis factor. It was shown in [Gray74] that the optimal value for μ , with respect to Gray's spectral flatness measure, is $r_x(1)/r_x(0)$, where $\{r_x(n)\}$ represents the autocorrelation sequence for the input speech data sequence $\{x_n\}$. However, a commonly used constant value for μ , which has been empirically determined, is 0.94 [Wong80]. Saito demonstrated that this simple differencing network has the effect on fixed-point arithmetic corresponding to a 3-bit improvement in analysis word length when compared to no pre-emphasis [Sai85].

² The analysis is called pitch asynchronous when the placement of the analysis window (frame) does not depend on the location of the pitch epochs.

speech complexes that a processing strategy which incorporated F_0 , F_1 , F_2 , and F_3 achieved good results.

Studies of artificially stimulated nerve fibres show that the response of electrically stimulated nerve fibres is almost deterministic when compared to the stochastic acoustical response. Also, a cochlear implant system should be designed to stimulate the auditory nerve in a manner that exploits the ability of the cochlea and central nervous system to discriminate loudness. If the response of a normal ear is to be mimicked, then this "intelligence" must be programmed into a speech processor. This implies a digital approach.

The above points suggest that a feature extracting speech processor must be accurate and intelligent to maximize information transfer and flexible to take advantage of future developments. The speech processor described in this thesis was designed taking these points into account.

As noted earlier, the design, algorithms and implementation of this processor was embarked upon in 1984 based on the early success of feature extraction-based analog processors. Recent experiments at other institutions which apparently confirm the utility of our strategy have proceeded in parallel with our development.

3. FEATURE EXTRACTION ALGORITHMS

In this chapter, the algorithms used for extracting features (i.e., pitch and formants) from speech will be examined. Chapter 2 described the necessity in speech analysis systems to effectively separate the excitation function from the vocal tract's impulse response. The technique used in the feature extraction program to perform this separation is based on linear prediction coding (LPC). After LPC theory is reviewed, its direct applications to feature extraction will be discussed. This includes using the autocorrelation method of LPC to extract prediction coefficients and hence formants via linear predictive spectra, and using a modified LPC technique to extract pitch via the Simplified Inverse Filtering Tracking (SIFT) Algorithm. Also included in this chapter are sections on making voiced/unvoiced decisions and smoothing the extracted feature contours. A final section is included which discusses some limitations in using LPC to model speech.

Throughout this chapter it is assumed that the linear predictive analysis is operating upon relatively noise free speech. LPC is known to have problems operating in noisy conditions. A discussion of the known solutions to these problems is left to Chapter 7, which covers future considerations.

3.1 Linear Predictive Coding

LPC is important in speech analysis systems because it approximates the speech model given in the previous chapter and is theoretically elegant. The fundamental concept behind LPC is to represent vocal tract information by estimating parameters of

the vocal tract model¹. The model used to describe the vocal tract is an all-pole rational transfer function which is excited by an impulse train for voiced speech and noise for unvoiced speech. The z-transform of the vocal tract transfer function is of the form:

$$H(z) = \frac{1}{1 + \sum_{k=1}^M a_k^* z^{-k}} \quad (3.1)$$

Thus, the parameters which describe the vocal tract model are the filter coefficients or predictors, the a_k 's. M is simply the order of a specific transfer function. A gain constant is not included in the model because only the spectral shape of the filter is important.

A speech segment is sufficiently complex that it does not always match the all-pole model above. For example, nasal sounds such as /n/ and /m/ are better represented by a pole-zero model. In general, however, the vocal tract transfer function is characterized by resonances which are well represented by complex pole pairs. Further, the effects of zeros in aural perception are not as important as those of poles [Sai85]. Thus, most of the important characteristics of the vocal tract transfer function are preserved in an all-pole model.

3.2 Determining LPC Parameters by the Autocorrelation Method

There are two basic methods used to determine the LPC parameters which describe the vocal tract model, autocovariance and autocorrelation. The autocorrelation method is used for both pitch and formant extraction in the feature extraction program. The inverse filter formulation of Markel and Gray [Mar73b] will be used to demonstrate the autocorrelation method for determining the LPC parameters. The inverse filter

¹ LPC only models the excitation function indirectly, as a residual to vocal tract calculations.

formulation is preferred over other formulations, such as maximum likelihood estimation [Sai85] or the spectral flatness formulation [Mar73b], because it is straightforward and gives greater insight into the LPC model. The benefits of the autocorrelation method over the autocovariance method are listed at the end of section 3.2.2.

3.2.1 The Inverse Filter Formulation

A digital all-zero filter of the form

$$A(z) = 1 + \sum_{i=1}^M a_i^* z^{-i} \quad (3.2)$$

is assumed, where $M+1$ defines the filter length. This is the inverse of the all-pole filter used to model the vocal tract transfer function given in Equation 3.1. Given an input sequence $\{x_n\}$ of length N (so that $x_n = 0$ for $n < 0$ and $n > N-1$), the coefficients $\{a_i\}$ must be determined such that the total energy at the filter output is minimized.

The solution to this digital inverse filter formulation is obtained using a least squares criterion as follows. The total energy $r_e(0)$ of the filter output is calculated from the error function, e_n , also called the residual, by

$$r_e(0) = \sum e_n^2, \quad \text{where} \quad e_n = x_n + \sum_{i=1}^M a_i^* x_{n-i} \quad (3.3)$$

The total energy is then minimized by taking the partial derivative of $r_e(0)$ with respect to a_k , $k = 1, 2, \dots, M$, setting each partial derivative to zero, and solving for $\{a_k\}$. The result is

$$\left(\sum_{i=1}^M a_i \right)^* \left(\sum_{n=0}^{N-1} x_{n-i}^* x_{n-k} \right) = - \sum_{n=0}^{N-1} x_{n-k}^* x_n \quad (3.4)$$

Since $x_n = 0$ for $n < 0$ and $n > N-1$, Equation 3.4 is equivalent to

$$\sum_{i=1}^M a_i * r_{i-j} = -r_j \quad j = 1, 2, \dots, M \quad (3.5)$$

were

$$r_i = r_{-i} = \sum_{N=0}^{N-1-i} x_n * x_{n+i} \quad (3.6)$$

If the signal were perfectly predictable, the residual, e_n , would be zero. In this case, according to filter theory, the input signal x_n would be the sum of a finite number of exponentially decaying sinusoids. As a result, e_n would be zero only within the boundaries of a single pitch period and have a distinct peak at a pitch period boundary. That would mean that the periodicity of the speech signal would be essentially the only information on the speech process which is definitely maintained in the residual.

In the frequency domain, the inverse transform is attempting to transform the input sequence into white noise (in a least squares sense), that is, a constant spectrum signal. If M goes to infinity, the inverse filter will theoretically predict the inverse of the input signal spectrum, resulting in a constant output spectrum. If M is finite, then it is no longer possible to span the input signal, and the filter can only be designed to approximate the inverse of the signal characteristics. Thus, the order M is the result of a trade-off: if M is too small, then a poor estimation of the inverse vocal tract's resonance structure is obtained; if M is too large, then the fine-grain structure due to the periodic excitation is estimated along with the resonance structure. What is desired is a close estimate of the envelope while ignoring the fine-grain structure.

3.2.2 Implementing the Autocorrelation Algorithm

Determining the prediction coefficients starts with a frame (or window) of speech samples, x_n , where $n = 0$ to $N-1$, which, according to the speech model, corresponds to the vocal tract impulse response convolved with the excitation function. The window size N should be large enough to allow sufficient spectral resolution but small enough to ensure that significant changes in the vocal tract structure over the period of measurement do not occur. Typically three or four pitch periods should be included in the window², which for the average male speaker means a window length of about 32 ms. Consequently, the number of samples, N , is a simple function of the window length and sampling rate.

The average spectrum of voiced speech falls with a frequency of 6 dB/octave due to the glottal wave shape and the lip radiation characteristics [Gray74]. In order to decrease the dynamic range required to represent the spectrum and reduce the probability of numerically caused instabilities, it is desirable to counteract this fall-off. A suitable pre-emphasis filter, which is easily implemented, is the first order differencing filter $1 - \mu z^{-1}$, where μ is called the pre-emphasis factor. It was shown in [Gray74] that the optimal value for μ , with respect to Gray's spectral flatness measure, is $r_x(1)/r_x(0)$, where $\{r_x(n)\}$ represents the autocorrelation sequence for the input speech data sequence $\{x_n\}$. However, a commonly used constant value for μ , which has been empirically determined, is 0.94 [Wong80]. Saito demonstrated that this simple differencing network has the effect on fixed-point arithmetic corresponding to a 3-bit improvement in analysis word length when compared to no pre-emphasis [Sai85].

² The analysis is called pitch asynchronous when the placement of the analysis window (frame) does not depend on the location of the pitch epochs.

A smooth Hamming window is then applied over a frame of samples. The Hamming window is utilized because it has been experimentally observed that for the pitch-asynchronous autocorrelation method, improved spectral representation is obtained since, as per Saito's interpretation, spectral matching is one interpretation of autocorrelation LPC [Mar73b]. Further, the application of a Hamming window reduces the ill-effects on prediction error caused by positioning of the analysis frame (that is, the actual location of pitch epochs within the analysis frame) [Rab77c]. The overall effect of applying a Hamming window is a more distinct resonance structure with narrower bandwidth estimations.

The prediction coefficients, a_k 's, are then calculated by solving Equation 3.5, a set of M equations in M unknowns. The coefficient matrix implied in Equation 3.5 is Toeplitz (i.e., elements along each diagonal are equal). A computationally efficient recursive method for solving sets of equations involving Toeplitz matrices was developed by Levinson with modifications provided by Robinson and Durbin [Mak75], [Mar76]. However, for a fixed-point implementation, the algorithm developed by Le Roux and Gueguen [LG77] is the most accurate [Sai85].

The Le Roux-Gueguen (L-G) algorithm does not compute the prediction coefficients directly, but an intermediate set of parameters, the reflection coefficients³ or k_i 's. These are so named since they determine the reflection coefficients of a stepwise constant acoustic tube model of the speech production system. The relationship between reflection and prediction coefficients is covered in detail in [Mak75] and [Mar76]. The L-G algorithm is important because all variables developed by L-G can be shown to be less than unity. For fixed-point implementation on the TMS32010, this means that

³ They are also called partial correlation (PARCOR) coefficients.

binary fractional representation can be used without any possibility of overflow, thus maximizing precision. In contrast, the Levinson algorithm simultaneously extracts both the reflection and prediction coefficients. The prediction coefficients, for tenth order LPC, are bounded approximately by ± 8 [Mor86]. This means that, in theory and in practice, a fixed-point implementation using the Levinson algorithm results in less accurate reflection coefficients than an equal precision fixed-point L-G implementation. The form shown here was taken from [Mor85a].

$$\begin{aligned} y_{m,1} &= r_m & \text{for } m = 1, 2, \dots, M \\ b_{m,1} &= r_m & \text{for } m = 0, 1, \dots, M \\ k_m &= -y_{m,m}/b_{m-1,m} \end{aligned} \quad (3.7a)$$

$$y_{m,i+1} = y_{m,i} + k_i * b_{m-1,i} \quad (3.7b)$$

$$b_{m,i+1} = b_{m-1,i} + k_i * y_{m,i} \quad \text{for } i = 1, 2, \dots, m-1 \quad (3.7c)$$

$$\text{for } m = 1, 2, \dots, M$$

Reflection coefficients are transformed into prediction coefficients by a procedure called Step-Up [Mar76].

$$a_{0,0} = 1$$

$$a_{m,i} = a_{m-1,i} \quad \text{for } i = 0 \quad (3.8a)$$

$$= a_{m-1,i} + k_m * a_{m-1,m-i} \quad \text{for } i = 1, 2, \dots, m-1 \quad (3.8b)$$

$$= k_m \quad \text{for } i = m \quad (3.8c)$$

$$\text{for } m = 1, 2, \dots, M.$$

It has been found that a good choice for the order of LPC analysis, M , is the sampling rate in kHz plus two to five [Mar76]. For example, with a sampling rate of 8 kHz the minimum order would be $(8 + 2) = 10$. An estimate of the resonance structure is then determined as the reciprocal of the inverse filter spectrum. The analysis procedure is applied to successive frames of the speech signal, so that the coefficients of the model are continually updated to reflect the changing vocal tract characteristics.

There are many benefits of the autocorrelation method, contributing to its wide acceptance. The Toeplitz coefficient matrix has the benefit that only one row of the matrix is needed to define it and it is guaranteed to be nonsingular [Mar73b]. With the autocorrelation method, in the absence of computational inaccuracies, the resulting all-pole filter is guaranteed to be stable [Mar73b]. Further, it has been experimentally determined that for pitch asynchronous analysis and for large analysis segment size (two or three times the average pitch period) the performance of both formulations in representing speech waveforms is the same [Cha74].

3.3 The Simplified Inverse Filter Tracking (SIFT) Algorithm

There are many algorithms available to extract pitch from speech [Hess83], but accuracy and speed must be considered. The pitch extracted has to be precise enough for it to be useful if open speech comprehension is to be achieved, and the algorithm must be sufficiently fast for real-time extraction on present day processors. These two limitations tend to work against each other, because accuracy implies a complex and hence slower algorithm, and speed implies simplicity and hence inaccuracy. The SIFT algorithm provides an adequate balance between these two constraints. This algorithm had not been widely used due to its heavy dependence on multiplication. However, multiplication operations are no longer a concern with the fast hardware multiply of present day digital signal processing chips.

Autocorrelation pitch detectors are, in general, very robust and reliable [Rab77a], [Rab76]. However, although the simple autocorrelation function of a segment of voiced speech usually displays a fairly notable peak at the pitch period, other peaks due to the formant structure of the signal makes this analysis unreliable. To eliminate the effects of the higher resonance structure, the input signal is usually bandlimited to below around

900 Hz where only the pitch information, first formant, and sometimes the second formant remain. Several techniques have been proposed to eliminate the remaining formant structure prior to autocorrelation analysis [Rab77a], one of these being to use the LPC-based inverse filter [Mar72b], [Mar74], [Mar76].

Evaluation of excitation parameters in an LPC-based system rests upon the examination of the residual signal obtained by filtering the original speech with the inverse predictor filter. Inverse filtering pre-whitens, or attempts to eliminate, the resonance structure imposed by the vocal tract (in a least squares sense). Autocorrelation analysis can then be performed on the residual signal to extract the pitch and help provide voiced/unvoiced decisions.

This procedure can be simplified because pitch information, as mentioned above, is contained below 900 Hz (other pitch information, such as harmonics of the fundamental, exist above 900 Hz). Therefore the speech signal could originally be sampled at about 2 kHz without much loss of pitch information. In collecting the input speech sequence, an 8 kHz sampling rate is required to enable formant extraction (as discussed in the next sub-section). This means that all frequency information from 0 to 4 kHz is maintained in the input signal. To effectively lower the sampling rate, the input signal is bandlimited to 900 Hz to prevent aliasing (a digital low-pass filter is used) and then downsampled at a ratio of 3:1, which produces an effective sampling rate of 2.67 kHz. By using a lower sampling rate, fewer sample points are required, so the total number of operations in the algorithm are reduced. Further, a resonance, or formant, is well represented by a complex-pole pair. Thus, because the maximum number of resonances below 900 Hz is two, a fourth-order inverse filter is adequate for whitening the spectrum.

The SIFT algorithm is summarized as follows: (a) the speech waveform is first prefiltered and downsampled, (b) this signal is then pre-emphasized and Hamming windowed, from which a fourth order inverse filter is calculated via the autocorrelation method of linear prediction, (c) the filtered signal (but not pre-emphasized or windowed) is passed through the inverse filter and the residual signal is obtained which is then Hamming windowed, (d) the autocorrelation sequence of the windowed residual signal is calculated, from which the pitch period, F_0 , is established from the largest peak in the autocorrelation sequence, (e) parabolic interpolation is applied in the region of the peak to get a better time estimate of the pitch and to compensate for downsampling, and finally (f) voiced/unvoiced decisions are made based upon the interpolated peak and other data, such as a zero crossing count and input energy threshold.

Note that the inverse filter was calculated from the pre-emphasized, windowed data, whereas the residual was obtained from the unwindowed and non-emphasized data. This has the effect of producing a one-pole low-pass filtered error signal (without low-frequency bias), which helps eliminate high-frequency "noise" in the residual signal. Thus, when a segment is unvoiced, the peak amplitudes in the residual are reduced, which in turn reduces spurious peaks in the autocorrelation sequence.

Although speech is a zero-mean process over long periods of time, a bias can exist due to windowing effects. Because the autocorrelation of a constant is a ramp, this can effect both the LPC coefficients extracted and the autocorrelation of the residual. As a result, better results are obtained if the speech frame is zero-meant prior to analysis [Mar74].

The entire range of the autocorrelation function does not have to be searched (nor calculated). That is, a lower limit for pitch values, typically 2.5 ms, and an upper limit, typically 20 ms, are enforced [Mar74].

3.4 Voiced/Unvoiced Decisions

For each speech segment, a voiced or unvoiced (V/UV) decision has to be made. The set of silent data frames does not have to be discriminated because such frames are treated the same as unvoiced sounds. That is, the electrodes will be excited randomly at an amplitude proportional to the input energy and if the input energy is sufficiently low, they will not be excited at all. The V/UV decision is usually performed by analyzing a set of parameters, such as short-time energy or zero crossing rate, determined from the current speech segment. There are close to a hundred parameters which can be extracted [Rab77b] to make a V/UV decision, but when many parameters are used, a statistical approach using pattern recognition is required to combine them [Atal76], [Sie79]. Further, many of the possible parameters do not provide a very strong discrimination among the sets, and training data for individual speakers and environmental conditions is required.

Currently, the V/UV algorithm suggested by Markel [Mar72b], [Mar74] is being used. The technique suggested by [Cam86] for the U.S. Government Standard LPC-10E algorithm, however, has been surveyed and is recommended for future implementation, as will be discussed in Chapter 7.

The parameters suggested by Markel are zero crossing rate (which gives an indication of high frequency content), short term energy (used for unvoiced-silence discrimination), and the height of the interpolated peak obtained from SIFT (which indicates the degree of voicing). Markel recommends that if the zero crossing rate,

extracted from the raw speech data, exceeds 2 zero-crossings/ms, then the frame should immediately be defined as unvoiced [Mar74]. Similarly, a threshold test on the short-term energy level should be performed. The exact threshold is a function of the particular system.

The height of the interpolated peak obtained in SIFT is a good indicator of voicing. Specifically, a peak below a variable threshold indicates that the speech segment is unvoiced. The actual threshold value is determined by the location of the interpolated peak, because when the pitch period is small, proportionally more pitch periods will be obtained per analysis interval, causing the peak to be higher. Similarly, as the peak location in time increases, the threshold is lowered. The threshold is determined linearly between the minimum (2.5 ms) and maximum (20 ms) peak locations.

3.5 Formant Extraction via LPC

As discussed earlier, formants are manifested in the spectral domain by energy maxima at the resonance frequencies of the vocal tract. Hence, if the spectrum of the vocal tract has been accurately evaluated for a short speech segment, formants can be extracted by simply examining the spectrum for peaks.

For spectral analysis of the speech signal's resonant structure, the inverse filter transfer function, $A(e^{j\omega})$, can be evaluated to estimate vocal tract characteristics at discrete frequencies. Specifically, equally spaced samples on the unit circle of the inverse filter can be obtained by applying a Discrete Fourier Transform (DFT), or more efficiently a Fast Fourier Transform (FFT), to the finite-length sequence $a^{-1}(n)$ which is defined as [Mar72a]:

$$\begin{aligned}
 a^{-1}(i) &= 1, & i &= 0 \\
 &= a_i, & i &= 1, 2, \dots, M \\
 &= 0, & i &= M+1, \dots, N'.
 \end{aligned}
 \tag{3.9}$$

The magnitude of the complex spectrum is calculated to obtain the smooth envelope of the inverse filter. This envelope can be inverted to get an approximation of the vocal tract spectrum, which is searched for peaks corresponding to formants. This spectral inversion is extraneous, as the inverse filter spectrum can be just as easily searched for nulls.

If N' equally spaced samples of $H(z)$ on the unit circle are desired, then the $M+1$ point sequence $h^{-1}(n)$ is augmented with $N'-M-1$ zeros and the transform is evaluated. N' can be chosen arbitrarily large to increase the frequency resolution at the expense of computation time. Our system has $N' = 64$ representing a spectrum from 0 to 4 kHz, which results in a spectral resolution of approximately 125 Hz. This means that formants are accurate to within 63 Hz, and any two peaks within 250 Hz of each other cannot be resolved. This is suitable for the few electrodes (8 or 16) being used. As will be noted later, in the third generation TMS320C25 system, N' can be increased if required.

Unvoiced sounds are not well represented by linear prediction. If the speech segment is unvoiced, then the formants extracted by the above procedure will be reasonably random. As a result, pitch extraction (and the voiced/unvoiced decision associated with it) is performed first; if the speech segment is unvoiced, formant extraction need not be performed at all.

Both the pitch and formant extraction algorithms described above are known to occasionally produce gross errors. This is due to the limitations of LPC modelling (as described in the next section) and the stochastic nature of natural speech. Smoothing

algorithms use past and future pitch or formant "samples" to smooth out obvious gross errors in the feature contours.

Rabiner et al. [Rab75] originally suggested applying nonlinear median smoothing to speech processing. The usual linear smoothers are inadequate for smoothing data which exhibits both local noise and sharp discontinuities. In contrast, median smoothing preserves sharp discontinuities (for example in the transition between voiced and unvoiced segments) and smooths out gross errors (also called outliers). The real benefit of applying median smoothing to pitch contours was demonstrated objectively in [Rab76] and subjectively in [McG77].

Median smoothing is a simple technique in which the present feature sample is replaced with the median of the samples surrounding (and including) it. The size of the running median (the number of points needing to be ranked) is always an odd number, with the smoother using an equal number of past and future frames. It depends on the minimum discontinuity duration to be preserved and the amount of delay the system can withstand before the features extracted have to be output. For example, if the median smoother wants to rid contours of two outliers in a row, then a five point median smoother is needed. But if a five point median smoother is used, then the feature being output would be delayed three frames: the current frame being processed, and the two future frames used by the smoother.

3.7 Limitations of LPC Modelling of Speech

LPC attempts to separate the vocal tract's excitation signal from the vocal tract impulse response. It does so by extracting $A(z)$, the transform of the inverse predictor filter, and $E(z)$, the z-transform of the residual. The ideal solution for both pitch extraction and formant analysis would be obtained if $1/A(z)$ were the vocal tract transfer

function and $E(z)$ were the excitation function⁴. The LPC algorithm, however, does not necessarily perform the separation in this desired way. This can cause problems with the feature extraction algorithms as will be demonstrated by the following examples.

With high pitched voices, common in women and children, the impulse response of the vocal tract does not decay enough before the onset of the next voiced impulse. This causes error in the analysis, for example, the deviation of extracted formant frequencies toward the nearest harmonics of the pitch frequency [Sai85]. In the case where F_0 coincides with F_1 , more than 95% of the total signal energy can be concentrated at this frequency [Hess83]. As a result, the inverse filter, $A(z)$, is sure to allocate one, or more likely several, of its complex zeros to that frequency. This, in turn, will cause most pitch information to be filtered out in the residual signal. This coincidence between F_0 and F_1 is the cause of SIFT's failure for high-pitched voices [Rab76], [McG77]. Further, it makes the resultant vocal tract spectral information unreliable.

In a nasal, the poles are resonances of the nasal tract, and the vocal tract is a closed side branch, causing zeros. This means that the speech process no longer satisfies an all-pole model, but a pole-zero (ARMA) model instead. The zero can cause F_2 , if it is close by, to be reduced in amplitude or even eliminated. Thus, the resultant vocal tract spectrum extracted using LPC analysis will not be accurate, since the effect of zeroes (which would be represented by poles in the inverse filter) are not taken into account.

LPC analysis poorly models unvoiced speech. This is because the acoustic characteristics of these sounds are not well represented through formants. Further, the spectral characteristics of the noise source tends to overwhelm the vocal tract resonances,

⁴ The excitation signal is not a perfect impulse train but includes the effects of opening and closing the glottis. The glottis effects can be absorbed by either the vocal tract transfer function or the z -transform of the excitation function without a loss of generality.

rendering the predictor coefficients, and hence the entire vocal tract model, inaccurate [McC74]. Parameters determined as a by-product of the LPC analysis are used for voiced-unvoiced decisions, but in no way are the characteristics of voiceless speech examined. This results in the limitation that all voiceless sounds are treated the same, and no distinction is made between them.

A final LPC limitation is that it is assumed that during the analysis interval the speech waveform is stationary and the system is time-invariant. With a fixed window size corresponding to 25-32 msec, this assumption is not always satisfied. This can happen with high pitched voices, where as many as ten pitch periods can be in the analysis frame, and in rapid transitional sounds such as plosive stops [Sai85].

4. FIRST GENERATION FEATURE EXTRACTION SPEECH PROCESSOR

This chapter covers the design of the first-generation speech processor which was developed at Carleton University between 1984 and 1987. The speech processor extracts features from speech, formats 24-bit instructions, and transmits them to the receiver/neural stimulator, which in turn sends pulses of varying amplitudes, durations, and polarity to the electrode array.

Digital techniques for real-time feature extraction became viable in 1983 when the first low-cost general purpose digital signal processor, the TMS32010, was released. The methods used to extract speech features are based on linear predictive coding (LPC). Despite LPC's limitations, as described at the end of the previous chapter, LPC analysis was thought to be the only accurate techniques also viable in real-time on a TMS32010. When the project was initiated in 1983, it was known that these techniques could be extended and made more accurate when more processing power in future DSPs became available.

4.1. Development and Target Hardware

The processor used to implement the feature extraction algorithms in this part of the project is the Texas Instruments TMS32010 digital signal processor [TMS87]. This processor was developed primarily for digital signal processing applications and has diverse features making it ideal for our application. In this part of the project, two distinct TMS32010-based systems are used. One, the Texas Instruments TMS32010 Evaluation Module [EVM85], herein referred to as the TI EVM, is used for software

development and debugging. The other, the Communications Research Centre's Carrot III, herein referred to as the CRC board, is the first-generation target system.

The CRC board had been chosen early in the project as the first generation target environment by Prof. L.R. Morris. Jon Bosloy modified the CRC board for downloading executable programs into program RAM and built the interface to the Sherbrooke stimulator [Bos86]. This environment was used unchanged throughout development.

4.1.1. The TI EVM

The TI EVM is a general-purpose development tool for the TMS32010. It contains a TMS32010 slave processor, and a TMS9995 master processor. The master processor allows the user to down-load TMS32010 programs to the board and execute these programs. Many software debugging facilities are provided.

The Texas Instruments Analog Interface Board, or AIB, plugs into the TI EVM and provides analog-to-digital and digital-to-analog converters, as well as input and output filters [AIB86]. This allows the system to be used for digital signal processing. Together, the EVM and AIB provide the necessary environment for developing and debugging feature extraction algorithms running in a TMS32010 environment.

4.1.2. The CRC Board

The CRC board is a compact TMS32010-based board (Figure 4.1). It contains a TMS320C10 processor, 4K words of program ROM, 4K words of program RAM, RS232C interface, codec, an input filter/pre-emphasis circuit, and an output filter/de-emphasis circuit. It was originally designed at the Communications Research Center (CRC) of the Canadian Department of Communications for use as a 2400 kbit/sec LPC-

Figure 4.1 - CRC board contains TMS320C10, PML parallel/serial interface, codec, 4K program RAM, downloading ROM, op-amp chip, RS232 interface, taken from [Sky86].

based vocoder. The rights to the board were then obtained by SkyWave Electronics Ltd. who are currently marketing it commercially [Sky86].

The codec uses standard μ -law compression (μ -law gives the equivalent of 13-bit sampling in compressed 8-bit samples) and is configured for a sampling frequency of 8065 kHz. The processor's input frequency is 16.512 MHz, instead of the usual 20 MHz, to allow it to be easily divided down to the 2.064 MHz required by the codec. A custom LSI chip on the board, the PD32HC01, handles the interface to the RAM and ROM, as well as interfacing the parallel TMS32010 to the serial codec. In addition, the PD32HC01 allows for multiple interrupt sources by multiplexing three interrupts onto the single TMS32010 interrupt input.

Prior to LPC operations, the input signal should be pre-emphasized as discussed in Chapter 3. This operation can be performed in hardware and is done by the pre-emphasis/input filter. This filter was also matched to the input filters on the codec and the frequency characteristics of a specific input transducer, the Primo electret microphone (part number EM95A).

The CRC board can be switch selected to use either program RAM or ROM. However, the board provided no downline loading or on-board monitor to communicate with the RAM. The processor was socketed, however, so that the program RAM could be used with an in-circuit emulator (ICE) during program development. Our EVM could not provide the type of ICE needed (it does have ICE capabilities, but must use its own memory), so the board had to be modified so programs could be developed in RAM. The use of EPROMS was not desirable due to the long erase times associated with such devices when frequent modifications were being made during development.

A downline-loading scheme was developed whereby the receive line of the RS-232C port was hard-wired to the BIO pin on the TMS32010, and a downline-loading program, which permanently resided in the top 128 word of program memory, was added in PROM [Bos86]. Thus, an additional chip select was needed to differentiate between the 128 word downline-loader in PROM and the remaining 3968 words of program RAM.

The downline-loader's operation is quite simple. It reads in the executable program in standard TI downline loadable format serially at 9600 bps via the RS-232C connection, reconstructs the serial bit stream into words, and writes those words into the program RAM.

The power consumption of the CRC board originally was 2.8 Watts, of which 1.5 Watts were consumed by the TMS32010 itself. A high speed CMOS successor to the TMS32010, the TMS320C10, was obtained whose power consumption is rated at 0.25 Watts. Where possible, high speed CMOS was used on the board to help minimize the power consumption. This gave the CRC board a total power consumption of 1.6 Watts.

The CRC board is very useful as a first-generation target hardware since it allows quick loading of programs into RAM, it is small, it has a reasonably low power consumption, and it demonstrates the feature extraction software.

4.1.3. The Stimulator Interface

An interface between the CRC board and the Sherbrooke stimulator was built for use in the Cochlear Implant Project [Bos86]. The design was based on an existing interface for the TI EVM developed at the Institute of Biomedical Engineering of the University of Toronto but optimized for the CRC board.

The Sherbrooke interface performs the following functions. It accepts a 24-bit command word from the TMS32010 and transmits it as a serial bit stream to the Sherbrooke stimulator. The command is transmitted most significant bit first. Communication with the stimulator is accomplished by providing the serial bit stream and a synchronized clock signal. The clock rate can be anywhere between 500 Kb/s and 1 Mb/s. The resultant bit stream is first fed through a Manchester encoder which converts the bit stream into the Manchester, or split-phase, data format. This formatted bit stream is then sent directly to the RF transmitter. In addition, the interface provides a 16-bit count-down timer which is used by the TMS32010 to time the transmission of commands. This primarily controls the on and off times for the stimulation. The entire interface board contains 16 chips occupying 235.75 cm² with a power consumption of about 90 mW [Bos86].

A single chip integrated "Manchester Encoder" was developed at the University of Sherbrooke to help miniaturize the interface [Duv86]. The encoder interfaces the radio frequency AM modulator to the speech processing microprocessor. It uses typical control signals and 8 bit latches and can therefore be controlled directly by the microprocessor data bus. The circuit performs a "3 parallel 8 bit words" to "serial 24 bit Manchester coded transmission word" conversion. With this chip, the only other circuit needed to complete the interface is a count-down timer to assist the digital signal processor in the timing of command words.

4.1.4. The Sherbrooke Stimulator

The function of the receiver/neural stimulator is to accept both power and processed speech, in the form of 24-bit command words, from the transmitter, and to generate the appropriate stimulation currents to be conducted by the electrodes to the

nerve fibres in the cochlea. The command words are transmitted serially on a transcutaneous electromagnetic induction link at a rate between 500 kbit/sec and 1 Mbit/sec. Its electronics are hermetically sealed in a bio-compatible titanium case, which contains a disconnect plug enabling direct access to implanted electrodes and easy stimulator replacement in case of failure, medical difficulties, or future developments [Duv84]. The custom-made microprocessor requires a maximum of 10 mW of power at full stimulation of all the electrodes. This power is transmitted by the speech processor over the transcutaneous RF link. A stimulator which can excite up to eight monopolar channels (1 electrode/channel) or four bipolar channels (2 electrodes/channel) has been built, and development of a sixteen electrode stimulator is underway.

A block diagram of the eight electrode stimulator (Figure 4.2a) and its 24-bit command word format (Figure 4.2b) are shown. The A field is set to "011" to synchronize the incoming data and indicate a stimulator command. The B field contains the various op-codes for stimulator commands. The C, D and E fields are used to select the various electrodes in the array. The F field is used to indicate the voltage to be applied to a selected electrode. A brief description of each of the stimulator's eight commands, which can be used to generate a variety of stimulation patterns, is given below [Duv82].

- 000 Preparation of the electrode specified by field C, with the voltage provided in field F, and a synchronized start of stimulation on all prepared electrodes with the electrode specified in field E as the ground return.
- 001 Same as op-code "000", except no ground return is used (field E is ignored).

Figure 4.2 - The Sherbrooke Stimulator. (a) Block diagram of the stimulation module. (b) the command word format, taken from [Duv82].

- 010 Preparation of the electrode specified by field C, with the voltage provided in field F, and start of stimulation on electrodes C and D with the return electrode specified by E.
- 011 Same as op-code "010", except no ground return is used.
- 100 End of stimulation on electrode C.
- 101 Preparation of electrode C without stimulation. The start of stimulation will then be synchronized with one (op-code "010" or "011") or many other electrodes (op-code "000" or "001").
- 110 End of stimulation on electrode C and D.
- 111 End of all stimulation and discharge of all coupling capacitors. The whole module is reset.

If the command words are transmitted at 500 kbit/sec (worst case), then the time required for the transmission of a command word is 48 μ sec. If every available electrode is to be stimulated asynchronously, then two command words must be sent to each of them, one at the start of stimulation and one at the end of stimulation. This implies that the maximum frequency of operation for all electrodes asynchronously is approximately 1.3 kHz. If less than eight electrodes are stimulated then this frequency is proportionately increased. This also implies that the minimum pulse width can be 48 μ sec, the time required to transmit the "end-of-stimulation" command.

4.1.5. The Carleton Electrode Array

The cylindrical electrode array delivers stimulation currents, derived from the Sherbrooke neural stimulator, to specific sites of the cochlea [vdP87]. It is both flexible

and small enough to be inserted into the cochlea with minimal damage to the surrounding soft tissue.

The electrode array was designed and fabricated using photolithographic techniques, common in printed circuit board design, allowing mass production at low cost. The shape, size, number, and layout of the stimulation pads can easily be varied by a computer aided design (CAD) process. However, the area of the stimulation pads determines the impedance and hence the voltage and current density that the electrodes will operate at. It is advantageous to make the pad area as large as possible which is the limiting factor in the size and number of electrodes.

With a cylindrically shaped electrode array, stimulation pads can be placed on opposite sides of the cylinder. Thus, radially oriented bipolar pulses can be generated, the importance of which was discussed towards the end of Sub-section 2.1.1.

The electrode array has been surgically inserted into a number of cadaver cochleas. The initial conclusions were that the array has suitable flexibility, with only minimal damage to the soft tissue, and the electrode pads were aligned in good positions along the basilar membrane [vdp87]. To test for toxicity, a planar version of the electrode array, not the cylindrical version to be used in humans, has been implanted in five chinchillas for various lengths of time with no adverse effects being noted. Accelerated life tests are underway as of this presentation date.

4.2. Implementation of the SIFT algorithm

A block diagram of the SIFT program's functional modules may be found in Figure 4.3a. A brief description of the program's operation for pitch extraction on the EVM board follows.

Figure 4.3 - Block Diagram of the Feature Extraction Programs. (a) the pitch extraction program, (b) the formant extraction program.

The program collects a frame of 246 speech samples at a 8065 Hz sampling rate. The sampling rate is chosen to match the one given on the CRC board for the codec. The samples are passed through a fifth-order digital low-pass filter with a corner frequency of 0.9 kHz. Then they are decimated by three (which means only every third filter output is kept) and Hamming windowed. The decimation is possible since the input signal bandwidth has been limited to less than a third of its original bandwidth. This leaves 82 points to work with.

The 82 data points are then converted to a zero mean form. This step is performed prior to autocorrelation so that a DC component will not appear in the autocorrelation results as a ramp. A zero through four lag by 82 point autocorrelation is computed, and the five autocorrelation results are passed through a fourth-order Le Roux-Gueguen (L-G) matrix inversion routine. The resultant four reflection coefficients, k_i 's, are converted to prediction coefficients, a_k 's, via step-up [Mar76]. The above computations derive an approximation to the inverse vocal-tract transfer function. Then the a_k 's are used to form a fourth-order FIR filter through which the 82 data points are passed, to spectrally flatten, or whiten, the signal. (Note that the a_k 's are updated every frame).

Since the above algorithm derives an approximation to the inverse vocal tract transfer function, passing the data points through this filter attempts to eliminate the resonance structure imposed by the vocal tract. The result is an approximation to the impulse train for voiced speech. The output function, represented by the 82 output points, will have peaks at times approximating those of the glottal pulses. The 30.5 ms time window covers between 3 and 5 pitch periods (for typical male speech). An 82 point autocorrelation is performed on this output function, and the position of the maximum peak of the autocorrelation gives the current pitch period of the sampled

speech. Incorporated in this algorithm are pitch extraction decision rules, which are used to determine whether the samples collected are for voiced or unvoiced speech. These heuristic rules use zero crossing counting, peak picking, and the computation of input energy as their basis for decision.

The 82 autocorrelation points are output to a digital-to-analog converter (codec on the CRC board, and DAC on the EVM) giving a diagnostic display on an oscilloscope (Figure 4.4 (a)-(e)). The input speech waveform can be displayed on the oscilloscope simultaneously, using an independent time base, verifying that the position of the peak in the autocorrelation function corresponds to the period of the speech waveform (Figure 4.4 (f)). The pitch period, as determined by the autocorrelation peak and decision rules, is also stored, so that when the peak is being output to the oscilloscope, an additional flag is output on a separate line. Thus, verification that the correct peak was indeed picked is possible. Subsequently, the diagnostic signal (from the DAC), the flag signal (from the drive latch on the CRC board and the C7 flip-flop on the AIB), and the original speech are displayed concurrently on the oscilloscope.

Several simplifications from the original SIFT algorithm were made. First, the speech waveform was not pre-emphasized prior to LPC calculations on the EVM board, because on the CRC board, the target system, the input filter does pre-emphasis in hardware. Second, the input speech data was windowed prior to being passed through the inverse filter. The original SIFT algorithm passed the unwindowed speech through the inverse filter and then windowed the residual signal. The original SIFT algorithm required that the prediction coefficients be calculated from a windowed version of the speech waveform, while unwindowed, or "clean" speech was to be passed through the inverse filter. The modification was made because there was not enough program memory to store both a windowed and clean speech frame.

Figure 4.4 (a) - Real-time diagnostic signal for the vowel /u/, as in up, as seen on an oscilloscope (the signal repeats periodically):

0.2-2.8 divisions on the oscilloscope represents 0-30.5 msec of the autocorrelated error signal, with the pitch peak at 1.1 div. (approx. 10.6 msec).

2.8-4.8 divisions on the oscilloscope represents 0-4.0 kHz of the LPC inverse spectral envelope. The formants, f_1 , f_2 , f_3 , and f_4 , are the nulls in the waveform at 3.2 div. (0.8 kHz), 3.4 div. (1.2 kHz), 4.0 div. (2.4 kHz), and 4.4 div. (3.2 kHz), respectively.

Figure 4.4 (b) - The vowel /e/, as in bed.

Figure 4.4 (c) - The vowel /r/, as in bird.

Figure 4.4 (d) - The vowel /au/, as in baud.

Figure 4.4 (e) - The vowel /u/, as in boot.

Figure 4.4 (f) - Real-time speech waveform (above) and diagnostic signal (below) as seen on an oscilloscope. Note that both traces are on independent time-bases, hence it is impossible to take a clear picture by a single sweep as in Figures 4.4 (a)-(e)

For the diagnostic signal: 0.6-5.5 divisions on the oscilloscope represents 0-30.5 msec of the autocorrelated error signal, with the pitch peak corresponding to 10.6 msec; and 5.6-9.4 divisions on the oscilloscope represents 0-4 kHz of the LPC inverse spectral envelope, with the first four formants correspond to 0.52 kHz, 1.7 kHz, 2.6 kHz, and 3.4 kHz. For the speech waveform, the distance between the peaks (the pitch) also corresponds to 10.6 msec.

As the speaker varies pitch, the autocorrelation peak on the diagnostic signal (at 2.2 div.) tracks the time domain pitch of the speech signal, with a 2 frame delay. By a simple time-base adjustment, real-time verification of pitch tracking is possible.

Finally, the entire autocorrelation sequence of the residual was calculated, not just the segment between 2.5 ms and 20 ms. Even though these limits for the time bounds of an acceptable peak were enforced in the decision rules, the entire sequence was calculated for the diagnostic display. It was felt that the display showing the entire autocorrelation would be useful during development (for example when being used with low pitched male voices and high pitched female or children's voices). If more computing time is needed in future developments, an abbreviated autocorrelation function could be computed.

4.3. Implementation of Formant Extraction via LPC Spectra

A block diagram of the program's flow for formant extraction may be found in Figure 4.3b. A brief description of the program's operation follows.

Two hundred of the 246 samples collected are pre-emphasized, Hamming windowed, and converted to a zero-mean signal. A 200-point by 10-lag autocorrelation is then performed on these windowed samples, the eleven normalized r_k 's are passed through a tenth-order Le Roux-Gueguen matrix inversion routine, and the resulting k_i 's are converted to a_k 's by Step-Up. These a_k 's are padded or extended with $(N-11)$ zeros, where $N = 64$, and used as input to a 64-point FFT algorithm which calculates the inverse filter's spectral envelope for the current speech frame. Using the magnitude of this spectrum, a null picking algorithm is all that is needed to extract estimates of the first three formants for the voiced speech. Note that the term "null" is used to indicate a local minima in the LPC spectrum.

The magnitude of the inverse filter spectrum is output through a D/A converter to be displayed on an oscilloscope. The final diagnostic display shows the autocorrelation signal, determined by the SIFT algorithm, followed by the magnitude of the inverse

spectral envelope. The position of the formants in the spectrum are also output on the flag signal, so that when the inverse spectrum is being displayed, the correct extraction of formants can be verified by visual comparison.

A few of the choices in analysis conditions for formant extraction will now be discussed. With a sample frequency of approximately 8 kHz, a minimum of 10th order LPC should be performed to adequately model the resonant structure of speech [Mar76]. The discrete resolution of the speech spectrum, N , in our system is 64 points. This provides sufficient accuracy for the number of electrodes typically used in a cochlear implant¹. A 64-point radix-4 FFT may be implemented relatively efficiently on a TMS32010. Finally, only 200 of the 246 input points are used in the 10-lag autocorrelation calculation. This is because of program memory limitations on the CRC board which will be discussed in greater detail at the end of Section 4.5. By using 200 points of the original 246 points only a small loss in accuracy results.

4.4. System Software Description

The following section gives a detailed description of the feature extraction program. The initial state of the system software, before work on this thesis began, will be described first, followed by data considerations, the main program's operation, the interrupt service routine, and each individual subroutine. The assembler implements conditional assembly [Mic85] which allowed for a single source code listing (Appendix A) for the EVM and CRC board versions. A flag in the source code decides whether the object code generated is for the EVM or the CRC board. The first section will describe the version which runs on the EVM. A following section will describe the code necessary for the CRC board.

¹ The Nucleus device uses the largest number of electrodes at 22.

4.4.1. Initial State of the System Software

When work on this thesis began, both pitch and formant extraction had been implemented in separate programs running on the EVM board. The pitch extraction algorithm also had a separate, slightly modified, version running on the CRC board. The majority of the SIFT algorithm had been converted from its original FORTRAN to TMS32010 assembly language by Prof. L.R. Morris, with the voiced/unvoiced decision routine (the subroutine DECIS) converted by Jon Bosloy [Bos86]. The majority of the formant extraction algorithm had also been implemented by Prof. L.R. Morris, with the null-picking algorithm (the subroutine NPIC) implemented by Leigh Bartlett at the University of Toronto. Little time and space optimization had been performed on the two programs, because they were "first drafts", and when running separately did not have any major limitations.

A major part of the work on this thesis was to combine the pitch and formant extraction algorithms into one single feature extraction program. This was a non-trivial task because both programs together would not fit in memory, neither had been written with future modifications in mind, and where documentation existed it was sparse. When the two programs were combined, most subroutines were rewritten, a common data structure was devised, and many corrections were made.

Program space, data space, and execution time were traded-off against each other to attain the real-time performance achieved. The initial projected schedule for first implants and tests in mid-1987 meant that a working TMS32010-based system was required in that time frame.

Although the later advent of the TMS320C25 would have alleviated much of the effort required and still have achieved real-time operation, this chip was not available

soon enough to have allowed a functioning DSP system to be integrated with the electrodes and stimulator at the required date. Ironically, the DSP system was indeed ready at that date, but the other system components were not, so that an implanted patient and system could not be tested.

4.4.2. Data Considerations

A data and program memory map is given in Table 4.1. To eliminate delay between successive frames of speech samples, the program uses double buffering. While the program is processing data, the interrupt service routine, herein referred to as the ISR, is filling the next input buffer (BUFF4) in the background. Once processing is complete and the input buffer is full, the 246 data samples in the input buffer are transferred to the processing buffer (BUFF1). This frees the input buffer from the processing activities, allowing both input and data processing to occur concurrently. Successive speech frames are contiguous with zero overlap, and no samples lost between frames. The input buffer and processing buffer cannot reside in data RAM because of their size, and are located in program RAM (PRAM) instead.

Data Type	Address	Description
Page 0	0-127	all intermediate variables used by subroutines
Page 1	0-4 5-11 12-15	intermediate values used by the FFT locations shared by the ISR and FFT locations used by the ISR only
PRAM	17-21	storage for ISR variables which would be overwritten by FFT while ISR is not executing
	22-28	storage for FFT variables which would be overwritten by ISR while FFT is executing
	29-33	storage of extracted pitch and formants
	34-2046	all system software except the FFT
	2047-2054	FTABLE - low-pass filter coefficients
	2055-2096	WIND1 - half a 82 point Hamming Window
	2097-2197	WIND2 - half a 200 point Hamming Window
	2198-2325	CNVRT - 128 point μ -law to linear conversion table
	2326-2407	SCOPE1 - diagnostic display buffer for storage of SIFT's 82 point autocorrelation sequence
	2408-2471	SCOPE2 - diagnostic display buffer for the 64 point speech spectral envelope
	2472-2594	BUFF4 - input buffer where the ISR dumps the data it collects
	2595-2630	BUFF3 - buffer where SIFT dumps the windowed/downsampled input buffer
	2631-2676	BUFF1 - processing buffer for SIFT
	2677-2890	BUFF2 - processing buffer for formant extraction
	2891-3388	not used
	3389-4095	FFT - the FFT subroutine

Table 4.1 - Map of program and data memory for the system software running on the CRC board.

The structure of the processing buffer was designed to minimize space. When the pitch algorithm downsamples from 246 to 82 points, 200 of the original data points must also remain for the formant extraction algorithm. However, the first 46 points can be

overwritten. This means that the 246 point processing buffer is split into two abutting buffers, the first being 46 points long (BUFF1) and the second 200 points long (BUFF2). The 82 point downsampled buffer (BUFF3) is also split into two abutting buffers, the first 36 points long (BUFF3) and the second 46 points long (BUFF1 again).

The final output of the pitch extraction algorithm is the 82 point autocorrelation signal and the address of the peak, which corresponds to the pitch. These are both stored in program RAM in buffer SCOPE1 and location APITCH respectively. The final output of the formant extraction is the 64 point magnitude of the LPC spectrum and the addresses of its nulls, which corresponds to the formants. These are also stored in program RAM, the envelope in buffer SCOPE2 and the 3 formants in locations AFMT1, AFMT2, and AFMT3, respectively. The two output buffers (SCOPE1 and SCOPE2) are abutted, so they can be easily output by the ISR as a single scope buffer. The oscilloscope's display is a split screen, with pitch information in the left-hand plane and formant information in the right-hand plane. The first value of the scope buffer is always a constant (03FFFFH) and is guaranteed to be the largest value output, so that the oscilloscope can be synchronized to it.

The FFT used in formant extraction requires use of the entire page 0 of data RAM and the first twelve locations of page 1. This means that any data normally stored in these locations must be stored in program RAM. The variables affected are: those used by the ISR; the pitch and formant locations to be flagged on the flag line; the decision parameter used by the subroutine DECIS (stored in DECT); and the input and output buffers.

4.4.3. The Main Program

The main program consists of; initialization; waiting for the ISR to fill the input buffer; transferring the input buffer to the processing buffer (which starts the ISR filling the next input buffer); and calling a series of subroutines to perform the necessary pitch and formant calculations on the processing buffer (while the ISR is filling the next input buffer).

The initialization consists of setting constants, initializing ISR variables, setting the clock counter to interrupt every 124 μsec , and initializing the AIB to transparent mode (which allows both input and output of data samples). Setting a counter flag then starts the ISR collecting data. This flag will be described in detail in the next section.

For **pitch extraction** the subroutines called in sequence are: **ZCD** -- the zero crossing detector/counter; **GETDAT** -- the filtering, downsampling and Hamming windowing routine; **ZMEAN** -- the routine which converts GETDAT's output to zero mean; **AUTO1** -- an autocorrelation routine, set to calculate autocorrelation coefficients (r_k 's) for 4 lags; **LG** -- a 4th order Le Roux-Gueguen algorithm for calculating LPC reflection coefficients (k_i 's) from r_k 's; **KTOA** -- converts the 5 k_i 's to prediction coefficients (a_k 's) via step-up; **INVFIL** -- the inverse filter routine which uses the a_k 's as filter coefficients and the output of getdat as input; **AUTO1** -- calculates 81 lags for the data filtered by INVFIL; and **DECIS** -- the pitch decision routine which uses the autocorrelation results and decision parameters as input to decide whether the signal is voiced or unvoiced and if it is voiced, finds the fundamental frequency. The address of the pitch, determined in DECIS, is stored so that the ISR can flag that value when it is being displayed on the oscilloscope.

For **formant extraction** the subroutines called are: **WINDAT** -- pre-emphasizes and Hamming windows 200 data points of the original input; **ZMEAN** -- which zero-means **WINDAT**'s output; **AUTO2** -- a 10 lag autocorrelation on the 200 data points; **NORMAL** -- normalizes the 11 r_k 's calculated by **AUTO2**; **LG** -- set to 10'th order for calculating reflection coefficients from r_k 's; **KTOA** -- converts the k_i 's to prediction coefficients via step-up; **PREFFT** -- normalizes the a_k 's and zero extends the a_k -array to 64 complex points; **FFT** -- calculates the inverse spectral envelope from the zero extended a_k -array; **MAGNTD** -- calculates the magnitude of the spectral envelope; and **NPIC** -- extracts the formants from the inverse envelope magnitude via null picking. Like **DECIS**, **NPIC** stores the addresses of the formants so that the **ISR** can flag them when they are being displayed on the scope.

Note that pitch and formant extraction share a number of software modules (e.g., **LG**, **KTOA**). As noted earlier, this sharing is crucial to the code space limitations of the TMS32010 and was a significant factor in choosing LPC-based formant and pitch extraction techniques. Details of the operation of each routine will be reviewed next.

4.4.4. INPUT -- The Interrupt Service Routine

The interrupt service routine, **INPUT**, performs three functions -- filling the input buffer, outputting the flag signal, and outputting the scope buffer. Special consideration is given to its data variables because of the TMS32010's limited amount of data memory.

The main program initiates **INPUT** to collect data by setting the variable, **ICOUNT**, to 246, the number of points per frame. **INPUT** uses this variable to count down each sample collected, until it reaches zero, at which point **INPUT** stops collecting data. **ICOUNT** will stop collecting data, i.e., miss samples, only if the system software

functions collectively take longer than 30.5 ms², the frame collection time, to compute. The interrupts can never be disabled because the scope buffer must be output continuously to synchronize the display on the oscilloscope. When the system functions take less than 30.5 ms, the main program tests ICOUNT in a loop until INPUT decrements it to zero, at which point it will transfer the newly collected input buffer to the processing buffer and reset ICOUNT to 246.

The flag signal is generated by toggling a flip-flop, accessed through output port PA0, on the AIB board. If the address of the output buffer sample corresponds to the address of the pitch or one of the three formants, then the flip-flop is toggled high, otherwise it is toggled low. The output buffer is simultaneously output through the digital-to-analog converter on the AIB board. The output and flag signal displayed on a dual-trace scope provides a useful diagnostic signal, demonstrating the accuracy and correctness of the two feature extraction algorithms. The output buffer is 146 words, the length of the 82 point autocorrelation plus the 64 point inverse spectrum, which means that the entire output buffer is displayed approximately 1.7 times per frame or 55 times per second.

There are only 144 data RAM locations in the TMS32010, 128 on page zero and 16 on page one. All subroutines confine their data use to page 0, except the FFT which uses all of page 0 and the first 12 locations of page 1. This leaves 4 locations on page 1 that the ISR can use without fear of being overwritten. These locations (ISTATE, IHIGH, ILOW, and IONE) are used to store the processor's pre-interrupt state and the constant "one". The data RAM locations that the FFT and INPUT share (ISMPL, ITEMP, ICOUNT, IPTR, PSCOPE, CSCOPE, and IFLAG) must be swapped in and out

² During various stages of development the processing time was greater than 30.5 ms. This feature is maintained in case additional features of unknown duration are added to the program.

of program RAM at the beginning and end of each interrupt to maintain mutual exclusion. This rather simple operation, data swapping, which is performed twice per interrupt, is the most time consuming of all system functions.

4.4.5. The Subroutines

Following are terse descriptions of the individual subroutines used in the feature extraction program. The time/space resources of the TMS32010 demanded that different time/space trade-offs be examined for many of the modules concerned, and it is reiterated that the significant achievements of this thesis -- real-time formant and pitch extraction on a TMS32010 -- were attained not only through selection of appropriate algorithms, but also through selection of algorithm implementation structure as described here.

CONVRT -- Convert Input Buffer to Processing Buffer: CONVRT takes the input buffer and transfers it to the processing buffer. This is performed immediately after the input buffer is full and all processing on the previous buffer is complete. To maintain data integrity, interrupts must be disabled during this operation. This is a safety feature, however, as the entire transfer takes less than 124 μ sec, the time between sequential interrupts.

ZCD -- The Zero Crossings Detector: ZCD scans the original 246 points in the processing buffer sequentially, XORing the current and previous values. A zero crossing occurs when one value is positive and the other is negative, thus causing the XOR operation to produce a negative result. The total number of zero crossings is stored in location NZ and is used later by DECIS to aid in the voiced/unvoiced decision.

PREFFT -- FFT Preparation Routine: This routine prepares the prediction coefficients, the NA array determined by KTOA, for the FFT. It normalizes the NA

values, with respect to the largest a_k in the array, to maximize precision in the fixed-point FFT. It then loads the a -array sequentially into locations 0, 2, ..., 20 on page 0, which correspond to the first 11 real points in the complex FFT input, and pads the remaining page 0 locations with zeroes. Finally, it moves all of page 0 to program RAM where the FFT expects its input.

MAGNTD -- Magnitude of Spectrum Calculation: The FFT returns the 32-point complex inverse spectral envelope in the first 64 locations of data RAM. MAGNTD calculates the complex component magnitude according to the following algorithm (where RXT is the magnitude of the real component and IYT is the magnitude of the imaginary component):

```

IF (RXT >= IYT) THEN
    IF (RXT <= 4*IYT) THEN RXT = (RXT-RXT/8) + IYT/2
    ELSE RXT = RXT + IYT/8
ELSE
    IF (IYT <= 4*RXT) THEN RXT = (IYT-IYT/8) + RXT/2
    ELSE RXT = IYT + RXT/8

```

The resultant RXT is the magnitude for each frequency sample, and it is stored in the formant scope buffer (twice for display reasons). Note that this magnitude approximation, well suited to the TMS32010's shifting capabilities, has an average error of 0.143% and a maximum error of 2.95% [Ada83].

PPIC -- Pitch Picker: PPIC takes NMIN and NMAX as the index limits in pitch's autocorrelation buffer to be searched. It searches between these limits and finds the position of the maximum value. It returns the peak value in MAXVAL, the program RAM address of the peak in the scope buffer in PTR2, and the index of the peak in the accumulator. This subroutine is called by the voiced/unvoiced decision routine, DECIS.

NPIC -- Formant Picker: NPIC searches the inverse spectral envelope buffer for three minimum, each minima corresponding to a formant. If it finds one, it stores the minimum's address in program RAM. To find a peak the routine looks at three successive values from the scope buffer and determines if the middle point is below both the end points. If the middle point is a minimum and less than -2.5 dB away from the maximum value between the last formant and the middle point (i.e., it is not a spurious minimum), then that sample is deemed to be a formant.

DECIS -- V/UV Decisions with Extrapolation: Two initial tests are performed on the current speech frame to see if it contains voiced or unvoiced speech. The first test is if the RMS threshold (IRO), determined by INVFIL, is below a certain level (IR1), and the second test is if the number of zero crossings (NZ), determined by ZCD, is above a certain level (IZM). If either of these are true then the frame is declared unvoiced. Otherwise, the peak between NMIN and NMAX (which are determined in the previous frame) is found using PPIC. The peak location (address in the scope buffer) is directly proportional to, and as a result determines, the pitch.

The SIFT algorithm, however, is known to produce errors due to a doubling phenomenon, where the peak extracted is approximately twice the real pitch. A test is performed by looking in the area of one half the previously extracted peak and finding a second peak. If this peak is greater than 0.85 times the previous peak, then doubling is assumed and the new peak is selected as the pitch. To correct for the loss of time resolution due to the previous downsampling, parabolic interpolation is performed to increase the accuracy of the pitch estimate. As the final test, if the peak height, determined after parabolic interpolation, is below a variable threshold, the frame is declared unvoiced. If the previous frame was unvoiced, then the variable threshold is determined by the equation

$$0.4 * (16383 - 300 * L)$$

where L is the peak position. If the previous frame was voiced, then the variable threshold is determined by the equation

$$0.3 * (16383 - 300 * L)$$

Anomalies can occur, such as a peak in a voiced segment being slightly below the threshold, whereas the preceding and following segments cross the threshold. Thus, the output of the calculated pitch value is delayed two frames so that linear extrapolation can be used to calculate the current pitch based on the previous frame and future frames. The first rule used to correct such anomalies is

$$\text{IF } (\text{ABS}(N1 - N3) - 0.4 * N3 \leq 0) \text{ THEN } N2 = (N1 + N3) / 2$$

where $N1$, $N2$ and $N3$ are contiguous past pitch estimates. Note that each sample being output, $N2$, is delayed two frames. This rule utilizes linear extrapolation, between the past and future frame, to help smooth pitch contours, but will not be applied if a large pitch variation occurs. A second rule is used if the past value, $N3$, is unvoiced

$$\begin{aligned} \text{IF } (N3 = 0) \text{ AND } (N2 \neq 0) \text{ AND } (\text{ABS}(N0 - N1) \leq 0.2 * N1) \\ \text{THEN } N2 = 2 * N1 - N0. \end{aligned}$$

where $N0$ is the recently extracted pitch estimate. In this case the linear extrapolation uses future pitch estimates only.

This final part of the algorithm determines the decision parameters, N_{MIN} , N_{MAX} , $N3$, $N2$ and $N1$, for the next frame. They are written to program RAM. N_{MIN} and N_{MAX} , the search range for PPIC, are $(L-7)$ and $(L+7)$, respectively, where L is the peak position of the current frame. N_{MIN} cannot be less than 4, which corresponds to a pitch of 1.49 msec (672 Hz), and N_{MAX} cannot be greater than 41, which corresponds to

a pitch of 15.25 msec (66 Hz). N1, N2 and N3 are the current peak position for this frame, the past peak position, and the peak position from two frames ago (the linear extrapolated value), respectively. It is then N3 which is output as the current peak.

GETDAT -- Filter/Downsample/Window for Pitch: All 246 data points are passed through a 0.9 kHz lowpass anti-aliasing filter, via the subroutine FILTER, and only one out of three filtered data points (0,3,6, ... ,243) gets extracted and multiplied by a Hamming window, running summed, and then stored sequentially in the temporary program RAM buffer, BUFF3. This downsamples the input frame from 246 to $(246/3=)$ 82 points. The running sum is stored in page zero locations XLOW and XHIGH and is used by the subroutine ZMEAN, directly following GETDAT, to zero-mean the resultant signal. Note that BUFF3 overwrites 46 of the original 246 samples, which leaves only 200 data points for formant extraction. Due to symmetry in the window function, GETDAT gets called twice, once to go forward through the half-length window function, and once to go backwards. The second call enters GETDAT in the middle of the subroutine because initialization has already been performed. This saves storing the other half of the window function, which would require 41 program memory locations.

WINDAT -- Pre-emphasize and Hamming Window: This routine pre-emphasizes and Hamming windows the 200 samples in BUFF2 and stores them back in BUFF2 according to the following equation:

$$SIG = (SIG - 7/8 * ILAST) * WIND$$

where SIG is the current sample, ILAST is the previous unwindowed sample, 7/8 is the pre-emphasis factor (μ), and WIND is a 200 point Hamming window sample taken from a program RAM array. The first value of ILAST is initialized to zero. Like GETDAT, WINDAT is called twice due to symmetry in the window function, thus saving 100

program memory locations. A running sum of the SIG results are kept to zero mean the results directly after WINDAT.

ZMEAN -- Convert Signal to Zero Mean: ZMEAN takes the 32-bit running sum (XLOW, XHIGH) of the windowed, downsampled input frame, calculated by GETDAT or WINDAT, and divides it by the number of points (82 or 200, respectively) which gives the mean value for the frame. It then subtracts the mean, or DC component, from each point in the frame. This removes any DC component from the input signal, caused by framing effects. A DC component would show up in subsequent autocorrelations as an erroneous ramp.

AUTO1 -- Autocorrelation for Pitch Extraction: The number of lags to be calculated, 4 in the first phase and 81 in the second, is passed to this routine in the accumulator. The routine reads the 82 point frame from program RAM and stores it in locations 0-81 on page 0 of data RAM. It then computes the sum of the squares of the data points to obtain r_0 (the energy of that frame). r_0 is the largest possible value of an autocorrelation, by definition, so it can be normalized to a constant and all other r_k 's normalized with respect to it. r_0 is normalized by: (a) loading the 32-bit r_0 into the accumulator; (b) shifting the accumulator left until there is a 1 in bit 30; (c) storing the resultant number of shifts in SCOUNT; (d) the shifted 32-bit r_0 is truncated to 16-bits; (e) the 16-bit r_0 is divided into the constant 03FFFH, which produces a fractional result; (f) and storing the 16-bit result in ZMASK. Subsequent lags, when multiplied by ZMASK and shifted by SCOUNT, produce the desired normalized results. The resultant 16-bit lags, r_k 's, are stored in program RAM.

Maximal time efficiency in computing an 82-point, 81-lag autocorrelation on the TMS32010 would require 81 in-line LTA/MPY instruction pairs; these two instructions

implement a 16x16 multiply and 32-bit accumulate in 400 nsec. The alternative, a looped construct (LTA/MPY/BANZ), doubles execution time since the BANZ loop instruction requires 2 cycles. The 162 memory locations required for the in-line solution were deemed unacceptable. A compromise, 8 in-line single cycle LTA/MPY instruction pairs terminated with BANZ (18 program memory locations), gives an effective 16x16 multiply, 32-bit accumulate time of $(8*2+2)/8 = 2.25$ cycles, compared to 2 cycles for in-line code and 4 cycles for the simple loop. However, this solution entails a non-standard program structure which involves computing where to enter the loop structure, since the total number of multiply-accumulates required varies with the lag count. Fortunately, this computation is minor compared to the approximate halving of the fully looped autocorrelation execution time achieved by this construct.

The obvious implementation for normalization on a TMS32010 involves repeatedly doubling the 32-bit accumulator contents for the required shift count. This requires about $81*(7+11*16) = 14,823$ cycles = 2.96 msec for an average shift of 16 bits for each result of the 81-lag autocorrelation. This time is unacceptable. An alternative approach is to use the multiplier unit to effect a left shift of each 32-bit result. To implement this strategy efficiently, 4 specialized procedures are required, with one selected depending on the shift count n ; $32 > n > 16$; $n = 16$; $16 > n > 0$; and $n = 0$. Worst case requires 19 cycles for procedure selection and normalization, for a total of $81*19 = 1539$ cycles = 0.31 msec, about an order of magnitude faster than the "brute force" technique.

AUTO2 -- Autocorrelation for Formant Extraction: The autocorrelation routine for formant extraction, is different from AUTO1 because 200 points (and eleven 32-bit results) cannot fit into the 144 word data RAM at one time. Thus, autocorrelation is performed via 8 steps of 25 points each, with partial 32-bit results for all 11 r_k 's

loaded, updated, and stored at each step. Specifically, 35 points are read in at a time in an outer loop, and new results for all 11 lags are calculated (and accumulated with results from earlier buffers) in an inner loop. The optimization techniques used in AUTO1 for the loop construct were not employed and simple looping was used, because only ten lags are being computed and the time savings would be small for the considerable effort.

NORMAL -- Normalization for AUTO2: NORMAL is a separate subroutine which normalizes the autocorrelation coefficients produced by AUTO2. Its operation is similar to the normalization section of AUTO1, normalizing all results to the constant 03FFFH for lag zero. However, it does not employ fast normalization using the multiplier.

FILTER -- 5th-order FIR Low Pass Filter: This is a fifth order biquad FIR low pass filter routine with a corner frequency of 0.9 kHz. It is called by GETDAT and used as an anti-aliasing filter prior to downsampling.

LG and KTOA -- Matrix Inversion and Step-Up: The LG program calculates reflection coefficients (k_i 's) from the normalized autocorrelation coefficients (r_k 's) via the Le Roux-Gueguen (L-G) technique [LG77], which is well suited to fixed-point arithmetic. KTOA uses the Step-Up algorithm [Mar76] to transform the k_i 's into a_k 's (prediction coefficients).

As discussed in Chapter 3, the L-G algorithm is important because all variables developed by L-G can be shown to be less than unity, which means binary fractional representation can be used without any possibility of overflow, maximizing precision. Both L-G and Step-Up have similar structures; an outer loop, executed M times, contains an inner loop whose length is the current outer loop counter value. Also, the variables accessed by the inner loop are sequentially indexed by the inner loop counter. This

correctly suggests that the TMS32010's autoincrement and decrement registers can access the variables with no explicit addressing overhead. The remaining operations thus consist of data-independent loop control and the data-dependent operations which in fact compute the results.

The novel Multiple-Entry, Single-Return (MESR) Subroutine [Mor86] is illustrated below, where the inner loop is expanded $M-1$ times in-line. Then, for each of $M-1$ run-time executions of the outer loop ($I = 2 \dots M$), YI and $BI(1)$ are loaded, the pointer (J) is initialized to unity, and entry to the MESR is made I expansions from the bottom. In practice, the preamble and register load is expanded $M-1$ times and $M-1$ pre-computed in-line calls are made to the MESR. Also, most TMS32010 pointer increment/decrements are free. For both L-G and Step-Up, MESR retains most of the advantages of in-line autogen code (e.g., zero explicit run-time control calculations) but with a code length an order of magnitude less (for $M = 10$). The total effect is to minimize both computation time and code length in LPC related computation.


```

SUBROUTINE LG(R,N,K)
DIMENSION R(14),BI(14),BIM1(14),K(14)
C
K(1) = -R(2)/R(1)
BIM1(1) = R(2)
BIM1(2) = R(1)+K(1)*R(2)
DO 6 I = 2,N
  YI = R(I+1)
  BI(1) = YI
  IM1 = I-1
  J=1
C
  GO TO (1,2,3,4,5), IM1
C
5  BI(J+1) = BIM1(J) + K(J)*YI
   YI = YI + K(J)*BIM1(J)
   BIM1(J) = BI(J)
   J = J + 1
4  BI(J+1) = BIM1(J) + K(J)*YI
   YI = YI + K(J)*BIM1(J)
   BIM1(J) = BI(J)
   J = J + 1
3  BI(J+1) = BIM1(J) + K(J)*YI
   YI = YI + K(J)*BIM1(J)
   BIM1(J) = BI(J)
   J = J + 1
2  BI(J+1) = BIM1(J) + K(J)*YI
   YI = YI + K(J)*BIM1(J)
   BIM1(J) = BI(J)
   J = J + 1
1  BI(J+1) = BIM1(J) + K(J)*YI
   YI = YI + K(J)*BIM1(J)
   BIM1(J) = BI(J)
C
   K(I) = -YI/BIM1(I)
   BIM1(I+1) = BIM1(I)+K(I)*YI
   BIM1(I) = BI(I)
6  CONTINUE
RETURN
END

```

DIVNEW -- Fixed Point Division: This is a fixed point division algorithm in which the divisor is passed in data RAM location DR, the 32-bit dividend in the accumulator, and the quotient is returned in the accumulator. The result is the negative of the true quotient.

INVFIL -- Inverse Filter: This routine uses the prediction coefficients, a_k 's, generated by LG and KTOA as the filter coefficients for a fourth-order FIR filter. This filter is used to spectrally flatten the windowed-filtered-downsampled speech waveform. The data points are taken from, and the results stored in, the program RAM buffer BUFF3.

FFT -- The Fast Fourier Transform: a 64-point radix-4 complex fast Fourier transform, was evolved from programs described in detail in [Mor85b]. While a straight-line code program executes in 0.535 msec, it requires 2700 words of program code, prohibitive for the current system. Thus, the first and last of the three passes were looped, resulting in a program occupying 707 words and requiring about 1.2 msec for execution. This program, written by Prof. L.R. Morris, was acquired in executable form.

4.4.6. Proposed Electrode Stimulation Algorithms

A test program, modified from the one given in [Bos86], which demonstrates the Sherbrooke stimulator interface is given in Appendix B. It outputs the 24-bit binary pattern 1100 0011 1111 0000 1111 1101 (C3F0FD). This particular pattern was chosen because it is easily recognizable and non-symmetrical. It is output approximately once every 50 μ sec by making the counter value 1Ah. This corresponds to the delay:

$$\begin{aligned} \text{delay} &= (\text{counter value}) / (\text{stimulator I/F clock}) \\ &= 26 / 516000 \\ &= 50.4 \text{ usec} \end{aligned}$$

A scheme which could be used to stimulate the electrodes in real-time is now described.

A subroutine would determine 24-bit stimulation command words and their respective counter values and an ISR would service the interrupts from the count-down

2

1.0

28

25

32

22

16

20

1.1

2.0

1.8

1.25

1.4

1.6

Micro-D

timer. The formant and energy information would be used to select the electrodes and their respective current levels, which in turn determines the actual command words used for stimulation. Each command word requires a count-down delay time determined by either the serial transfer interval, due to the delay between transmitting successive commands when multiple electrodes are to be stimulated simultaneously, or the pitch value for psycho-acoustic information transfer. The stimulator ISR would transmit to the interface the current command word, taken from a circular buffer, and the counter value corresponding to the delay before the next command is to be transferred.

There are many possible stimulation algorithms which could be used with this versatile device. Three formant-selected electrodes could be stimulated synchronously at a rate proportional to the pitch period and with amplitudes proportional to input energy. Using the op-codes given in Section 4.1.4, this would require the transmission of a maximum of four command words per pitch period ("010" for formant 1 (F_1), "010" for F_2 , "000" for F_3 , start stimulation with a ground return, and "111" end of stimulation). A second scheme would be the same as the one above except that the formant selected electrode pulses would be interleaved. This would require a maximum of six command words per pitch period ("000" for F_1 , "111" end stimulation, "000" for F_2 , "111" end stimulation, "000" for F_3 , and "111" end stimulation). A third and final scheme, which is more appropriate for a sixteen channel implant, would be the same as scheme one except that bipolar stimulation would be used. This would require a maximum of seven command words per pitch period (two "100" for the bipolar electrodes selected by F_1 , two "100" for the bipolar electrodes selected by F_2 , one "100" for one of the bipolar electrodes selected by F_3 , "010" for the other format 2 selected electrode and a synchronized start of stimulation, and "111" end stimulation).

4.5. Feature Extraction on the CRC Board

Several modifications are necessary for the EVM version of the feature extraction program to run on the CRC board. This is facilitated by Microsoft's macro assembler, MASM, which allows for conditional assembly [Mic85]. This means that both versions of the feature extraction program, the CRC version and the EVM version, may coexist in the same source code. By setting a flag at the beginning of the source code, the macro assembler will either assemble the code for the CRC board or for the EVM. The different code segments are surrounded by "if-else-endif" assembler directives. This maintains integrity between the two versions when modifications are made. There are eight differences between the two versions and all are related to the hardware discrepancies between the two systems. The following describes these differences.

- 1) The first 3 lines of program code are changed from:

START: B	GO	to	START:	DW	80H DUP (?)
B	INPUT			B	GO
				B	INPUT

to make room for the CRC board's downline loader. The downline loader resides in ROM in the first 128 locations of program memory.

- 2) The code necessary to initialize the interrupt counter and the operation mode on the EVM is changed to initialize the codec as follows:

LACK	3D		;ENABLE CODEC A/D INT. AND
SACL	TEMP		; TURN DRIVE BIT OFF (DRIVE=0)
OUT	TEMP, PA6		

The drive bit is connected to a light emitting diode, LED, via a latch and is used as the flag signal described earlier.

- 3) The codec's 8-bit μ -law samples are converted into 14-bit samples (13 bits plus sign) whereas the EVM collects linear 12 bit samples. As a result, the input energy threshold, experimentally determined, is changed from 2^{11} to 2^{15} , to compensate for the larger 14 bit samples. Thus the pertinent code is changed from:

SUB ONE, S11 to SUB ONE, S15

This is used after AUTO1 is called for the first time in the pitch calculations.

- 4) The subroutine CONVRT is replaced to reflect the fact that the CRC's input buffer is 123 words of 8-bit samples (two 8-bit samples per 16-bit word). This subroutine converts the 123 word, 8-bit, μ -law input buffer into a 246 word, 14-bit, two's compliment processing buffer. The 8-bit μ -law to 14-bit two's compliment conversion is performed via table look-up [TMS86].
- 5) In the subroutine WINDAT, the code which pre-emphasizes the input signal is removed since the hardware pre-emphasis filter on the CRC board performs this function. Further, the 1-bit left shift, which occurs when the windowed signal is being stored, is also removed. With 14-bit samples, overflow problems would occur later on in the program.
- 6) There are two separate versions of the interrupt service routine. The CRC version of INPUT contains a two's compliment to μ -law conversion [TMS86] which is necessary to output data to the codec. INPUT also stores the 8-bit input samples in 123 locations (two 8-bit samples per 16-bit word) instead of 246 locations (one 12-bit sample per 16-bit word) as on the EVM. Further, the output port needed for the diagnostic signal is mapped to a different port address, and the register

accessed for the flag signal is mapped to a different port and requires a different initialization constant.

- 7) As a result of point 6), the input buffer, BUFF4, requires only 123 words on the CRC board due to the 8-bit μ -law input data. The EVM requires 246 words for its 12-bit linear samples.
- 8) The CRC board contains a compact downline loader. The EVM's downline loader can accept instructions embedded in the object code that inform the downline loader at what address to start downline loading executable code. This feature is used when the FFT is "linked"³ to the main program: the starting address of the FFT is embedded in its executable code. The CRC board's downline loader does not possess this capability, and the main program must be padded, from the end of the program to the beginning of the FFT, with zeros, so that the FFT will be loaded in the proper location. Note that the FFT is always placed at the very end of program memory, the last instruction being in location 4095.

With the changes described above, it is apparent why the 10-lag autocorrelation in the formant extraction algorithm only uses 200 input points instead of the full 246. With the addition of the 128 word downline loader and the conversion software to go between μ -law and linear samples, there was not enough space in program RAM to store the full 246 word buffer between pitch and formant calculations. As a result, 46 of the 246 points were overwritten. This also explains the reasons for the data buffer scheme described in Sub-section 4.4.2.

³ The "linking" is performed manually by merging the main program's and FFT's executable code with a word processor.

4.6. System Operation and Timing

The codec on the CRC board and the clock timer on the EVM operate at 8065 Hz, which means that collecting one frame of 246 samples takes 30.5 msec. Through the use of double buffering, samples are collected and processed concurrently. This means that, if there is to be no time lag between collecting and processing successive buffers, all calculations must be completed within the 30.5 msec collection time. The actual module execution times per frame (msec.), for a 16.512 MHz clock, and program code storage space (16-bit words), are given in Table 4.2.

Program Module		Execution Time	Code Space
PROGRAM CONTROL		----	407
ISR	(essential)	3.64	306
	(display)	7.09	---
ZCD	(246 pts)	0.84	25
GETDAT&FILTER	(246 pts)	2.20	96
ZMEAN	(82 pts)	0.20	21 ^s
AUTO1&NORM	(4x82)	0.48	197
AUTO1&NORM	(81x82)	2.58	---
LG	(M=4)	0.14	---#
KTOA	(M=4)	0.03	---*
INVFIL	(82 pts)	0.51	39
DECIS&PPIC	(82 pts)	0.20	238
WINDAT	(200 pts)	0.84	34
ZMEAN	(200 pts)	0.40	---s
AUTO2&NORM	(11x200)	3.01	159
LG	(M=11)	0.34	213#
KTOA	(M=11)	0.10	124*
PREFFT	(M=11)	0.51	56
FFT	(64 cmplx pts)	1.20	707
MAGNTD	(64 to 32)	0.28	53
NPIC		0.20	78
data storage		----	845
free space		----	498
free time		5.71	---
Total		30.5 msec	4096 words

$$\text{ISR}(10.73) + \text{pitch}(7.18) + \text{formants}(6.48) + \text{free time}(5.71) \\ = \text{data collection time } (246 \times 124 \mu\text{sec}) = 30.5 \text{ msec}$$

Table 4.2 - Memory module execution time and storage space for the system software running on the CRC board.

Thus, the significant goal of implementation of the real-time pitch and formant extraction, subject to the severe time/space constraints of the TMS32010 has been achieved. Further, our timing analysis shows that feature extraction is completed in 17.30 msec per frame when graphics diagnostic output is not required. This leaves 12.8 msec per frame of "free" processing time. There is also 498 words of unused data

storage. This time and space may be used for mapping pitch and formants to the Sherbrooke stimulator and future enhancements.

4.7. Summary

A powerful, low-cost, programmable, extendible system has been created which, based on the earlier successes of less-accurate, "hard-wired" devices, should yield positive results as a signal processor for cochlear implants. It is re-emphasized that the accurate inverse spectral envelope and derived formants obtained via LPC, although important in the context of previous cochlear implant experience, are only two of many possible mappings of the LPC coefficients, which, together with pitch and gain information, suffice to recreate a relatively low bit-rate model of the vocal system, and hence the speech waveform itself. In fact, the fundamental basis for design of the signal processor is to embody a flexibility, generality, and adaptability which should obviate the need for any new hardware design in response to feedback from subject testing.

The CRC board based system has demonstrated that feature extraction via LPC analysis is feasible on a TMS32010-based system. However, the system is still far from complete, as the extracted features still have to be mapped to electrodes, and then used in a patient feedback situation. It is expected that the signal processor will undergo independent tests using existing implanted electrodes, such as those used by the Nucleus device. Since the Canadian group had no implants at a time when the software/hardware system was ready for testing, contacts were made with groups in the United States who have the capabilities to perform the necessary testing and also require an accurate speech processor. For laboratory testing the CRC board's output would be sampled (either the flag signal or the diagnostic signal or both) by a separate signal processor which would perform the mapping. Once a suitable mapping function had been determined

experimentally, efforts would begin to modify the CRC board so that it could be used as a take home device. This would mean designing a new hardware system largely based on the CRC board but which can be directly interfaced to other groups systems and can be run off battery power.

Further, additional signal conditioning must be added to the input signal if the device is to be used outside of the laboratory. The signal conditioning required is noise suppression, automatic gain control, and syllabic compression. These can be added as software modules (not requiring any additional hardware) which would be implemented in the additional time and space available on the CRC board. However these modules are beyond the scope of this thesis.

The next chapter describes a TMS320E17-based system. This system is a modified CRC board that succeeds in lowering the total power consumption to below 1 Watt. Low power consumption is essential for extended use when running off battery power, as would be the case in a portable cochlear implant system.

5. THE TMS320E17-BASED FEATURE EXTRACTION SYSTEM

A new system, the E17-board, was developed around the TMX320E17¹, herein referred to as the E17, the recently released digital signal processor from Texas Instruments. The E17 has an identical instruction set to the TMS32010, but contains additional hardware. The rationale for developing a speech processor around the E17 was an attempt to lower the power consumption and board size, while maintaining the same functionality as the existing CRC board.

5.1 Development and Target Hardware

Four members of the TMS320 first generation digital signal processing family [TMS87] were used in this project: the TMS32010 was used on the CRC board and EVM; the TMS320E17 (E17) was used on the E17-board; and both the TMS320E15 (E15) and TMS32011 were used in the hardware development environment built to write programs for the E17-board.

Texas Instruments released the TMS32010 digital signal processor in 1983. An object-code compatible sister processor followed, the TMS32011, which contains additional interface hardware for large volume applications. This interface hardware includes a dual-channel serial port with two receive and two transmit channels capable of full-duplex direct interface to combo-codec circuits² and μ -law-to-linear/linear-to- μ -law decoding circuitry. Early in 1987, the TMS32010 was updated to the E15. This

¹ The X in TMX320E17 stands for a beta release of the processor. When this system was being developed the production version, TMS320E17, was not available.

² In comparison, the CRC board uses a PD32HC01 "glue" chip from Pacific Microcircuits to interface the TMS32010 to the codec.

processor is both object-code and pin-for-pin compatible with the TMS32010 with two new features added. It contains 256 words of data memory, compared to 144 words on the TMS32010, and 4K of on-chip EPROM with copyright security for program memory. The E17 (Figure 5.1) was released at the same time as the E15. It is pin-for-pin compatible with the TMS32011 and contains the extra memory of the E15 with the interface hardware of the TMS32011. It is important to note that an E17 is equivalent to a CMOS TMS32010 plus the interface hardware of the TMS32011 plus the extra memory of the E15.

5.1.1 The CRC Board-Based Target System

The E17-based target system developed is a stand-alone system designed from the original CRC board. Because this system uses the TMS320E17, much of the original hardware on the CRC board was abandoned. The E17 contains all the necessary circuitry to interface to the codec so that the glue chip used on the CRC board (PD32HC01) is not needed, and the clock can be changed from 16 MHz to 20 MHz. The system software was modified to run out of the E17's on-chip EPROM so that no external program memory was required. Further, because the program resides permanently in EPROM the RS-232C circuitry is not needed. The only hardware that is used from the CRC board is the 40-pin socket for the E17, the pre-emphasis circuitry, the codec, and the power supply terminal block.

Instead of designing a new board, which would have been unacceptably time consuming for a feasibility study, a second CRC board was obtained and physically modified. All connections to the glue chip, external memory and RS-232C circuitry were cut. This circuitry was contained on one side of the board (Figure 5.2), so that it could be cut off. New connections, from the E17 to the codec, were made via wire-wrap. The

Figure 5.1 - Functional Block Diagram of the TMS320E17 Digital Signal Processor, taken from [TMS87].

Figure 5.2 - The CRC-board layout diagram illustrating the cutting which produces the E17-board (actual size).

pre-emphasis circuitry, from the input port to the codec was left unmodified. The resulting board was not pretty, but was fully functional and adequate for software demonstration. If this speech processor system were later to be pursued, a new printed circuit board would be made which would be mass producible, smaller, and have low enough power demand for battery-powered portability.

The glue chip on the CRC board contains the drive latch used to display the flag signal. Since the glue chip was abandoned, a new latch was needed. The XF pin on the E17/TMS32011 is latched from the XF bit in the control register. By toggling this bit, the pin toggles and is suitable for displaying the flag signal.

Finally, the E17 requires an input crystal of 20.48 MHz for the framing pulse, which drives the codec, to attain the required 8 kHz repetition rate. The CRC board requires a 16.512 MHz crystal which is divided down by the glue chip to a frequency of 8065 Hz. There is no hardware except the codec which depends on the input crystal so that increasing the crystal speed has no undesirable effects. A benefit is that the feature extraction program runs about twenty percent faster on the E17-board compared to the CRC board. The decrease in sampling rate, from 8065 Hz to 8 kHz, is too small to make a noticeable difference in the program's performance.

5.1.2 The Modified EVM Development System

The TMS32010-based development environment, the EVM board, could not be used in developing E17-based system software because it could not take advantage of the additional E17 hardware. A suitable environment, which combined the development tools of the EVM and the I/O capabilities of the E17, was built.

The TMS32011/E17 features a peripheral mode for the development and testing of RCM code [TMS87]. In peripheral mode, the device functions as a serial port peripheral to a standard TMS32010/E15 processor with only the serial port, interrupt synchronizer, system control register, and μ -law hardware functional. All other logic on the device is unused. This mode allows real-time emulation capability for the development of TMS32011/E17 applications. Thus, if a TMS32010/E15 is connected in parallel with an TMS32011/E17, a program will execute on the TMS32010/E15 but use the additional hardware of the TMS32011/E17. This scenario was implemented to connect an E15, on the EVM, in parallel with an TMS32011, on a separate, modified, CRC board, through the EVM's target connector. The target connector's output pins were buffered versions of the E15's output pins.

The CRC board was modified in the following way. The PD32HC01 glue chip was pulled and its socket used for the target connector. All original connections to the glue chip and TMS32010 were cut. The TMS32010 was pulled and replaced with a TMS32011. All new connections were wire-wrapped so that: the TMS32011 was in peripheral mode; all the necessary connections between the target connector (the EVM's E15) and the TMS32011 were made for successful peripheral mode operation; and the TMS32011 was connected properly to the codec. All the analog circuitry at the front-end of the codec was left alone. With this scheme, the program would run on the EVM board and use the I/O hardware on the modified CRC board. Thus, the development tools of the EVM could be used in a hardware environment functionally equivalent to the final target board.

As discussed earlier, the TMS32011/E17 requires an input clock signal of 20.48 MHz in order for the framing pulse, which drives the codec, to be exactly 8 kHz. For this system to work correctly, the EVM's 20.0 MHz crystal had to be pulled and replaced

with a 20.48 MHz crystal. The effects of this change could not be predicted in advance. The support chips on the EVM are TTL (normal, S, LS, and ALS) and can handle the increase in speed. It is not sure, however, if any of the EVM's monitor routines depend on clock timing. The EVM has been running at the higher speed since May of 1987 without any problems as of thesis publication date, but the monitor has not been extensively tested and timing problems could arise in the future.

5.2 System Software Modifications

This section describes the modifications which were made to the CRC board's system software so it would run on the E17-based system(s) described above. To use the E17's on chip EPROM, the software could not store data in program memory, so that all data manipulations have to be performed in data RAM. The software on the CRC board uses program RAM to store the diagnostic output buffer (146 words), the μ -law input buffer (123 words), the processing buffer(s) (293 words), and some temporary storage variables (23 words). This totals 584 words. Clearly, restructuring of the system software is necessary in order to get all calculations within the 256 word data RAM. A source code listing of the system software for the E17 board can be found in Appendix C.

One memory saving is to eliminate the diagnostic buffer (146 words) which outputs SIFT's autocorrelation sequence and the inverse filter's spectrum through the codec to an oscilloscope. It is used for testing purposes only. The processing and input buffers together occupy 416 words. Thus, double buffering would have to be eliminated. But if double buffering were abolished, no data would be collected while feature extraction calculations occurred. Feature extraction takes on the order of 20 msec, which is long enough to miss important vocal cues. Therefore, simply eliminating double buffering was not an adequate solution. Instead, all calculations requiring the full use of

data memory are performed first, and the operations that did not use full memory are performed while the ISR collects data in the background. Thus, there is a minimal delay between the processing of input buffers. To achieve this, a highly optimized, fairly complicated and relatively sophisticated approach was developed.

A data map (Figure 5.3) shows the order of operations and the state of page zero and page one of data memory after each subroutine has executed. This gives a graphical illustration of the program's operation and the location of variables during execution. Figure 5.4 should be referred to while reading sections 5.2.1 and 5.2.2.

5.2.1 Operations Between Frames

The subroutines requiring the full use of data memory are those that use the unmodified data collected by the ISR. These routines cannot execute while the ISR is collecting data, as there is not enough data RAM for more than one data buffer. Once these routines have finished execution, the input buffer is no longer needed and can be overwritten. Thus, these routines are performed between buffers, and the double buffering scheme used on the CRC board was modified. These routines are: **ZCD** -- the zero crossings counter; **GETDAT** -- the filtering, downsampling and windowing routine for pitch extraction; **WINDAT** -- the 200 point windowing routine for formant extraction; and **AUTO2** and **NORMAL** -- the 200 point by 10 lag autocorrelation and normalization necessary for formant extraction. Further, these routines were modified to operate on μ -law format input data, as there is not enough room to expand the input buffer into a 246 point linear array with space remaining for temporary variables. This means that the subroutine **CONVRT** which performed the μ -law to linear conversion on the CRC board can be eliminated.

PAGE 0

PAGE 1

Figure 5.3 - Data memory map versus time for the TMS320E17 system software. Each entry shows the contents of data memory for that system operation.

ZCD was easily modified. Its input now comes from data RAM and is in μ -law format. The μ -law format need not be expanded to linear because the sign bit in μ -law is maintained. Two tests were added because each of the 123 input words contain two μ -law samples. If only one sample per iteration is used then a test would be necessary to determine if it is a high or low byte. Thus, operating on two samples per loop minimizes loop control.

WINDAT on the CRC board takes two hundred linear samples from the processing buffer and windows them. If this were done, then two hundred additional data location would be used, which implies overwriting the input buffer before WINDAT has a chance to use it. To get around this, the input data is windowed by AUTO2, thus eliminating WINDAT. That is, when AUTO2 takes sixty samples for each buffer³, it converts them to linear and windows them prior to autocorrelation.

NORMAL, on the other hand, has to be performed before the ISR is restarted because contents of the 22 locations output by AUTO2 (11 32-bit autocorrelation coefficients) have to be reduced to 11 16-bit normalized coefficients. If this were not done, then there would not be enough temporary locations on page one for GETDAT to operate. GETDAT operates after AUTO2, and requires substantial memory on page zero for the downsampled array and the filter's delay line and coefficients.

GETDAT accepts 246 samples (123 words in μ -law format) and outputs 82 downsampled linear points. Once GETDAT has started, the ISR can start collecting data because GETDAT performs its operations much faster than the 124 μ sec sampling time. Therefore there was no worry about the new data overwriting the old data. The only

³ On the E17-board the autocorrelation is performed in 4 buffers of 60 samples instead of 8 buffers of 35 samples with the CRC board. AUTO2 is described in Sub-section 4.4.5.

conditions are that the first two samples must be processed before the ISR is started (or else there is a possibility that the ISR will overwrite the first word before it is processed), and the other data RAM intensive system operations, described above, must be performed before GETDAT. The input data is stored two samples per word, so optimal performance is obtained if the number of samples processed per loop iteration is a factor of two. Similarly, the data is downsampled by a 3:1 ratio so optimal performance is obtained if the number of samples processed is a factor of three. Thus, six loop samples ($6=2*3$) are processed per loop iteration.

5.2.2 Additional Algorithm Modifications

The data output from the codec is inverted. On the CRC board, the glue chip (PD32HC01) performs the necessary inversion (one's compliment) automatically in hardware. On the E17-board this has to be done by the ISR.

The ISR requires a minimum of thirteen data locations for its operation. Upon execution of any ISR, the TMS32010/E17 automatically switches the data pointer to page one. Thus, it is preferable to have all data that the ISR references, all temporary variables, and the input buffer on page one. This totals 136 words ($13 + 123$) whereas page one has only 128 locations. Either some data would have to go on page zero or the size of the input buffer would have to be smaller. It would be impossible to have some locations on page zero, because other operations such as the FFT require the entire use of page zero. Thus, the input buffer was shortened. The input buffer's optimum size requires that there are at least thirteen free locations on page one, and its length is divisible by six (for optimal performance of GETDAT, as discussed above). As a result, the input buffer was shrunk to 228 samples (114 words, with 2 μ -law samples/word) which corresponds to a window length of 28.5 msec. This leaves one free location on

page one. Because the input buffer is pared down, the downsampled signal's array is only $(228/3=)$ 76 words long.

AUTO1, the autocorrelation routine for pitch, operates on a 76 word input array. It outputs either 4 lags or 75 lags of normalized data. In order for both the input and output data of AUTO1 to fit into memory, the output data is stored on page one at the bottom of the input buffer (the end which is filled last by the ISR). This could result in a data conflict between the data being collected by the ISR and the data output by AUTO1. However, it was verified that the input buffer did not fill its final 76 words until well after AUTO1 had finished execution. Thus no conflict existed.

All other routines were modified so that they ran without any temporary variables in program memory. Most routines on the CRC/EVM board had input/output arrays in program memory, where now all arrays are stored in data memory. This has the benefit of using indirect addressing instead of TBLR/W (table read or table write) instructions giving an additional increase in speed. These modifications are straightforward and will not be described further.

The final change, abandoning the diagnostic signal, was made because there was no storage space in data memory for it. This leaves only the flag signal, a simple toggle requiring a counter and pitch and formant locations, displayed on the XF pin. If only the flag signal is output, then because of the "random" nature of the extracted pitch and formants, it would be impossible to synchronize its display on an oscilloscope. To synchronize the display, a constant buffer, the length of a normal output buffer (76 points for the autocorrelation signal and 64 points for the inverse spectrum), is stored in program RAM and output by the ISR to the codec. It contains all zeroes except for a synchronizing pulse, for the oscilloscope, and a distance pulse, to separate the pitch and

formant displays. Thus, the oscilloscope is synchronized by the synch signal from the codec, and the pitch and formant locations are displayed on the XF pin with respect to the synch signal.

5.2.3 Conditional Assembly

When the program runs in the development environment, it is running out of the program RAM on the EVM which can use program RAM to store data. This is not necessary (or possible) for normal operation on the target E17, but it is useful for storing the diagnostic signal (the autocorrelation signal produced by SIFT, and the inverse spectrum) which is displayed by the ISR to verify that the program is operating correctly in the development environment.

For conditional assembly, a flag, which determines whether the program is assembled for the development or target environment, is set at the beginning of the source code. If the program is assembled for the development environment, then two sections of code are added which copy the autocorrelation signal and inverse filter spectrum to the output buffer in program RAM. As a result, the ISR will send the diagnostic signal to the codec instead of the simple synchronizing signal. By using conditional assembly, any modifications made with respect to program operation are reflected in both versions of the object code. Thus (and most importantly), the code for the target E17 can be tested via the visual diagnostics available on the development system.

Note that programs destined to be blown into EPROM on the target environment will also run out of program RAM on the development environment.

5.3 System Operation and Timing

The E17-board is a stand-alone system with the software running out of on-board EPROM. To operate the system, a power line is connected and the power is turned on. Input is provided by a microphone connected to the MIC input port. The pre-emphasis circuit is the same as the one on the CRC board, with optimum performance obtained if the PRIMO Electret microphone is used. To observe the output, a probe is connected to the synchronizing signal which is the output of the codec (the ear port), and a second probe is connected to the XF pin on the E17 itself (pin 28) which gives the flag signal. This is the same display as described for the CRC board (Figure 4.4) except that the diagnostic signal, displaying the autocorrelation signal and LPC spectral envelope, is not given.

With the codec on the E17-board operating at 8 kHz, collecting 228 samples takes 28.5 ms. As discussed above, this version of the feature extraction program does not implement pure double buffering, causing samples to be lost between frames. A summary of the code storage space and module execution times is in Table 5.2.

Program Module		Execution Time	Code Space
PROGRAM CONTROL		----	239
ISR		3.64	90
ZCD	(228 pts)	0.84	31
AUTO2&NORM	(11x200)	3.01	257
GETDAT&FILTER	(228 pts)	2.20	229
ZMEAN	(76 pts)	0.20	18
AUTO1&NORM	(4x76)	0.48	190\$
LG	(M=4)	0.14	---#
KTOA	(M=4)	0.03	---*
INVFIL	(76 pts)	0.51	38
AUTO1&NORM	(75x76)	2.58	---\$
DECIS&PPIC	(76 pts)	0.20	217
LG	(M=11)	0.34	213 [‡]
KTOA	(M=11)	0.10	124*
PREFFT	(M=11)	0.51	45
FFT	(64 cmplx pts)	1.20	1327
MAGNTD	(64 to 32)	0.28	46
NPIC		0.20	59
data storage		----	566
free space		----	407
free time		6.11	---
Total		28.5 msec	4096 words

$ZCD(0.84) + AUTO2\&NORM(3.01) = \text{delay between frames}(3.85)$

$ISR(10.73) + \text{additional processing}(6.48) + \text{free time}(6.11)$
 $= \text{data collection time}(228 \times 125 \mu\text{sec}) = 28.5 \text{ msec}$

Table 5.2 - Program module execution time and code space for the system software running on the E17 board.

As can be seen by the table, the delay between frames is about 3.9 msec, the free processing time (while data is being collected) is 6.11 msec and the amount of free space left for further program development is 537 words.

5.4 Summary

The E17 system is more desirable than the CRC board because it is more compact, has a lower chip count, and consumes less power. There are, however, several problems with the system that caused it to be abandoned as the speech processor for a cochlear implant.

The most serious of its inadequacies is the almost 4 msec delay between frames, during which there is no data being collected. One of the primary functions of a cochlear implant is to assist in lip reading. This delay is large enough to miss unvoiced cue's, which are important in understanding speech when lip reading. If the system misses any vocal cues, it will be deficient.

The work at Research Triangle, only made available after the E17 system was finished, indicates that a high-resolution representation of the speech spectral envelope may be essential for good speech recognition [Wil87]. This means it would be best to store the entire inverse filter spectrum, instead of just the formants. The nature of sending speech information to the electrodes is that they must be updated at regular intervals. This means that any information being sent to the electrodes must be stored in memory throughout the processing time. The E17 system can not do this because there is simply no data memory left where the 32 word spectrum could be stored. As a result, the spectrum, though extracted, cannot be used.

A third problem with the system is that only 537 words of program memory is free for further development. This affects the system in two ways: there are still additional software modules which have to be added; and there should be enough space to add future software modules. Modules which have to be added include a procedure to map speech features to electrodes (which would include a second ISR and a mapping

matrix), noise suppression, automatic gain control, and syllabic compression. Additional space should be left over for adding features which result from field testing and may suit individual patient needs.

A final problem with the E17 system, which is also a problem with the CRC board, is that the sampling rate is fixed at 8 kHz. There is additional speech spectral information above 4 kHz, which may be useful for implantees, that both systems cannot represent. Again, if the system is to be flexible this limitation should not exist.

For these reasons, and not withstanding the expected appearance of an expanded memory version of the E17, development of a third system (not having any of these limitations) was begun. This is the topic of the next chapter.

Note that although most of the limitations of the E17 system were realized before the system was constructed, the possibility of having a portable, very low powered system -- available albeit with reduced performance -- together with the relatively small amount of new hardware/software effort required to implement the system, suggested that the effort would be worthwhile. That the system was successfully realized in a relatively short time reflected the significant effort taken in modularizing the software structure of the original TMS32010 system. Further, preparation of Figure 5.3 before hand, almost guaranteed the ease of software modification. A by-product of the effort was the creation of a relatively innovative and elegant E17 development environment by the techniques described earlier.

6. THE TMS320C25-BASED FEATURE EXTRACTION SYSTEM

Normally, the cycle of patient-test/software-modify, using the TMS32010 system, would have been embarked upon starting in the mid-1987 time frame. However, as noted earlier, an implanted patient was not available due to delays in development of the Sherbrooke stimulator and Carleton electrode array system components.

Since the TMS320C25 processor became available in the relevant time frame, it was decided to transfer the software to that processor so as to utilize the research/development time now made available and also in anticipation of the extra processing power that would be needed to implement the Stanford "deconvolution" algorithm, only made public in June 1987 [Tow87] (see Sub-section 2.1.1).

Thus, the final system under development is based upon the TMS320C25, herein referred to as the C25, digital signal processor. The C25 is the most advanced in the fixed-point digital signal processing family of microprocessors from Texas Instruments as of thesis publication date. Because the feature extraction program had been developed on previous generations of the DSP, transferring the software to the C25 was relatively straight-forward. This chapter therefore describes the efforts undertaken to develop a speech processor system for a cochlear implant based on the C25.

6.1. The Development System

A target system based on the C25 has not been built. All development has proceeded on Texas Instruments' software development system (SWDS) equipped with an analog interface board (AIB). A target system which only runs the feature extraction software would be very compact, consisting of the C25, an ADC/DAC, memory, and a

pre-emphasis circuit. A final speech processor for a cochlear implant must also include the circuitry to interface to a stimulator which drives the multi-electrode array. This includes a count-down timer, the Sherbrooke Manchester encoder chip and an RF modulator.

The first two parts of this section describe the enhancements of the C25 over the TMS320C10 (C10) digital signal processor and the software development system used to transfer the feature extraction software to the C25. The final part of this section discusses development of a target hardware system.

6.1.1. The TMS320C25 Digital Signal Processor

The main problems with the speech processors developed around the C10 and E17 concerned: the small amount of data and program memory available; the amount of processor time left over to perform additional tasks; and the "convoluted" (i.e., not easily maintainable) code required for the program to operate given the severe memory and time constraints imposed. All of these problems were eliminated by going to the more powerful second generation digital signal processor from Texas Instruments, the TMS320C25. It is again reiterated that the initial schedule for processor development and system implementation (1984-1987) meant that the TMS320C10 had to be used as a starting point: the C25 became available only at a much later date. Its compatibility with the TMS320C10 confirmed our faith in the "family concept" of processors, and the advantages therein, as noted below.

The architecture of the C25 is based upon that of the C10 (Figure 6.1). It is highly source code compatible with the C10 which means transferring code from the C10 to the C25 is relatively straightforward. In many applications it is capable of twice the throughput of the first-generation devices through architectural enhancements alone. The

Figure 6.1 - Functional Block Diagram of TMS320C25 Digital Signal Processor, taken from [TMS86b].

device can operate at 40 MHz for an instruction cycle time of 100 ns (10 MIPS) which means an overall increase in speed of up to four times. Additionally, its enhanced instruction set (133 instructions), large on-chip data memory (544 words), large memory spaces (64K of data and program memory), on-chip serial port with codec interface circuitry, and hardware timer made it desirable as the final target digital signal processor for the cochlear implant system.

Other enhancements in the C25, which are important for this application, include eight auxiliary registers (an increase of six over the TMS32010), an eight-level hardware stack, an IDLE instruction which powers the processor down between interrupts, improved external I/O interfacing and low-power consumption due to it being a CMOS device. Further, a repeat construct is used with single instructions such as multiply/accumulates, normalization, block moves and table read/writes, allowing a such instructions to be performed up to 256 times, thus eliminating looping overhead.

6.1.2. The TMS320C25 Software Development System and Analog Interface Board

The TMS320C25 software development system (SWDS) is an IBM PC (or compatible) resident development tool that provides software development support for the TMS32020 and TMS320C25 digital signal processors. The SWDS contains a TMS320C25 and 24K of full speed RAM which can be switch selected, as program memory or data memory, in various increments. In-circuit emulation (ICE) adapters, which reside outside the PC, are available and are connected to the SWDS via two 40 conductor ribbon cables. They allow the TMS320C25 on the SWDS to perform I/O operations with a target system. An ICE adapter board is provided, which is used to connect the SWDS directly to the analog interface board (AIB). The AIB board is the

same board which was used for receiving speech signals and outputting diagnostic displays in developing the C10 program code.

The development software which runs on the SWDS board is a high-level, easy-to-use debugging environment. Programs can be developed in either a windowing or command driven environment giving full control over all aspects of the C25's state including all flags, registers, program and data memory, and input signals (interrupt and I/O lines). Further, sophisticated features such as data logging (assigning I/O disk files to I/O ports for the reception or transmission of data) and "animated" execution using symbolic reverse assembly are provided.

It should be noted that Texas Instruments' assembler is not being used. A C25 cross-assembler [Leu87] running under Microsoft's (MASM) assembler on the PC was used because of its compatibility with previously developed C10 source code.

The approach taken in developing the C25 system software was to first get the C10 version, with a minimum number of changes, running in the C25 development environment. Then optimization was performed to take advantage of the extra architectural power on the C25. Once optimization was complete, it could be transferred to a target system. In this way software development could proceed without an existing final target system hardware. The same approach had been used in developing the original C10 software.

6.1.3. Developing a Target System

There are four paths being considered in developing a C25-based target system. The first and most obvious is to develop a system "in-house". This system would contain a C25 connected to an ADC/DAC with the necessary pre-emphasis and input filter

circuitry matched to a specific microphone. It would also have to contain expansion capabilities so the interface to the electrode stimulator could be added, or modified, at a later date.

The second path, undertaken in September of 1987, has been to work in conjunction with Skywave Electronics Ltd. Skywave bought the rights to the Carrot vocoder board, originally developed at CRC, used as the target board for our C10-based system. They have (mid 1988) developed a C25-based vocoder system. Meetings have been held and correspondence maintained with the staff at Skywave to ensure that the system contains the functionality needed by a speech processor for a cochlear implant. A functional block diagram of the proposed system is illustrated in Figure 6.2. The advantage of this path is that a fully functional target system can be acquired with a minimum amount of development time and cost. However, this system is not optimal for our application. It is not as compact as possible, difficult to expand, and not developed with portability (i.e., battery operation) in mind.

The third path, also undertaken in September of 1987, has been to work in conjunction with the cochlear implant research group at Stanford University in California [Whi86]. Contact with the Stanford group had been established through meetings at conferences (the Gordon Conference, EMBS'87 and ICASSP'88) and correspondence. They have been concentrating on the mapping of speech features to electrodes. To our advantage they have also been using the digital signal processors from Texas Instruments (the C10, TMS32020 and C25). They have developed a "home-grown" in-lab development system and are currently developing a portable C25-based speech processor [Kna88]. They are quite advanced in both psycho-acoustical research and hardware capabilities, but their feature extraction software has been limited to a digital implementation of bandpass filters (the limitations of this method for feature extraction

Figure 6.2 - Functional Block Diagram of C25-Based Target System: proposed in conjunction with Skywave.

has already been discussed). They find our software both advanced and desirable and would like to work together to get it running on their system(s).

The fourth and final path being followed is to work with the speech processor for auditory prostheses group at Research Triangle Institute (RTI) in North Carolina. Their purpose is to design and evaluate speech processors for auditory prostheses [Wil86]. The design end of their purpose has been centered around finding an optimal mapping of speech information with either feature-extraction or vocoder-type processors. They have primarily been using other group's feature extraction devices and their own mapping hardware and algorithms. The evaluation end of their research has been to test various strategies on patients by devising batteries of psycho-acoustical tests. They would like to try our feature extraction processor with the various mapping strategies they have developed. Unfortunately they have not been using the Texas Instrument DSP's. This would mean either transferring our software to one of their systems or developing interface circuitry between one of our systems and their hardware.

All four paths have their benefits, but what is finally needed is a speech processor which accurately extracts speech features, and has the necessary hardware and intelligence to drive a stimulator. Thus, it is hopeful that work continues with the groups at Stanford and RTI. Not only are these groups already established, with the necessary infrastructure, funding and equipment, but they are they are doing some of the most advanced research in the field.

6.2. Incompatibilities Between the TMS320C25 and C10

Even though the C25 and C10 are source code compatible, several architectural differences made running a reassembled, but unchanged, version of the system software impossible, thus resulting in the following changes. The memory map on the C25 is

different from the memory map on the C10. Page 0 of the C25's data memory map contains only block B2 (32 words at address 96-127) and some memory-mapped registers. The primary on-chip RAM blocks B0 and B1 reside on pages 4-7. The feature extraction program on the C10 used the BANZ instruction to implement both a loop counter and a memory address pointer for arrays or buffers based at location 0 in data memory. This double function use of an auxiliary register was necessitated by the limited number of auxiliary registers on the C10. Since blocks B0, B1 and B2 in the C25 are located at addresses other than 0, two separate auxiliary registers, one for loop count and one for memory address, were used. The C25 contains eight registers so that using separate registers was not a problem. Also, since the BANZ instruction on the C25 can switch addressing registers, efficiency is not lost.

In the direct addressing mode, the SST (store status register ST0) and SST1 (store status register ST1) of the C25 automatically sets the data memory page pointer to 0 rather than 1 as on the C10. These instructions are used to save the current state of the system upon execution of an interrupt service routine. Thus, the data memory addressing values should be between 96 and 127 to store the status registers in data block B2 (page 0) during an ISR.

6.3. System Software Enhancements

Because of the greater processing power and memory capabilities of the C25, many enhancements to the original system software (the TMS32010 version) could be made. This section describes those enhancements. A modified system software functional block diagram can be found in Figure 6.3 and a listing of the system software can be found in Appendix D.

Figure 6.3 - Block diagram of the feature extraction program on the C25. (a) pitch extraction, (b) formant extraction.

For speech framing, a double buffer scheme with a fifty percent frame overlap is employed. A frame of 123 samples is collected at an 8 kHz sampling rate and appended to the previous 123 samples. The 123 sample frame collected is stored unmodified for the subsequent frame. This results in a frame processing rate of approximately 65 Hz, compared to 33 Hz for the C10. To speed pitch extraction calculations, 41 words (half of the 82 points used by the rest of the pitch calculations) of SIFT's filtered and downsampled speech is stored between frames. This can be done because the filter delay line only depends upon past samples, not future ones.

In calculating the residual signal during the SIFT algorithm, the system software on the TMS32010 used the windowed version of the speech signal as input to the inverse filter. This deviation from the original SIFT algorithm was necessitated by the limited memory capabilities of the TMS32010. With the C25, the filtered, pre-emphasized (but not windowed) speech signal can be saved during the LPC calculations and subsequently passed through the inverse filter. The resultant residual signal is then windowed with a smooth Hamming window.

On the CRC board, pre-emphasis was performed in hardware by an analog differentiating circuit at the codec's input. This implies a fixed pre-emphasis factor. With the C25 system there is no hardware pre-emphasis at the ADC input and pre-emphasis is performed in software with the optimal pre-emphasis factor, μ , of $r_x(1)/r_x(0)$ as discussed in Sub-section 3.2.2. The optimal pre-emphasis factor is easily calculated with the general purpose autocorrelation routine discussed later on in Sub-section 6.4.5.

The SWDS contains data logging capabilities which were used to pass two seconds worth of stored speech through the feature extraction software, in non-real-time, and collect data to disk on each frame analyzed. This was used to optimize the voiced-

unvoiced decision parameters used by DECIS. The maximum number of zero crossings per frame to declare a frame unvoiced was set at 84 (which is equivalent 2.73 crossing/msec). The input energy threshold that a speech segment must cross to be declared voiced was set at 5×10^5 (11.9 volts²/frame). Finally, the range which DECIS tracks the current pitch value based on the last pitch value was set at ± 8 (± 1 msec/frame). This last variable enforces the constraint that the pitch cannot move more than 1 msec per frame, which in turn is used to help minimize doubling and halving effects.

Finally, the features collected, pitch and formants, are smoothed with a 3-point median filter. This results in a delay of only one frame (15.375 msec) before the extracted features are output. The substantial benefits of this form of nonlinear smoothing on speech features were demonstrated in [Rab75] and [Rab76].

6.4. Software Speed/Space Optimization

The advanced architecture and enhanced instruction set of the C25 were exploited to increase system software performance over that on the C10. These changes had the additional benefit of making the source code more readable and structured.

6.4.1. Data Allocation and Initialization

The C25 can directly address 64K of data memory with 544 words on-chip. The same paging scheme as the C10 is used, except that instead of 2 data pages there are up to 512. This means that for direct addressing, the data memory page pointer (DP) is 9 bits wide instead of 1. With all system variables and arrays in data memory, indirect or direct addressing is always used to access data. Thus, the time consuming program-data memory transfer overhead associated with the C10 is avoided. The penalty of having

data stored in off-chip data memory is that a target system must contain additional memory chips. Currently the system uses 640 words (page 8 through C) of off-chip data memory.

Block B2, page 0 locations 96-127, is used as a "safe" block where important data could be stored between frames without being overwritten. That is, these locations would not be shared with other system variables. Important data stored in this block included all ISR variables, the current pitch and formant values, pitch and formant tracking variables, and voiced-unvoiced decision variables.

The system software uses the data memory map given in Table 6.1. Page 0 (locations 96-127 only) contains important data which cannot be lost between successive iterations of the program. Pages 1, 2 and 3 on the C25 are reserved. Page 4 and 5 are configured as program memory so that the autocorrelation routine can use a pipelined single-cycle multiply-accumulate instruction (MAC). Page 6 contains scratch variables used by most subroutines. On page 7 and 8, locations 0-245 are used as a processing buffer and locations 246-253 contain the low-pass filter delay line. Page 9 and 10 are used as a 246 word unprocessed linear input buffer. The first 123 words on page 11 are used by the ISR to store the current half-frame being collected in parallel with system operations. On page 12, locations 0-81 are used as the diagnostic output buffer which is passed through the digital-to-analog converter to the oscilloscope. Locations 82-122 on the same page are the 41 point low-pass filtered signal from the last half-frame.

With all scratch locations on page 6, the data page (DP) pointer is always pointing to page 6, except during execution of the ISR and storage of important data. This is because all other pages contain arrays which are most efficiently accessed using the indirect addressing mode, which does not use the DP pointer.

Data Type	Address	Description
Page 0	0-5	reserved for memory mapped registers
	6-95	reserved for internal TMS320C25 use
	96-101	interrupt service routine variables
	102-105	feature storage (pitch and formants)
	106-114	formant tracking variables
	115-122	pitch decision variables
	123-125	not used
	126-127	reserved for use by the SWDS
Pages 4-5	0-245	used by autocorrelation routine for MAC instruction
	246-255	not used
Page 6	0-127	intermediate variable storage for all subroutines
Pages 7-8	0-245	processing buffer
	246-253	delay-line for low-pass filter
	254-255	not used
Pages 9-10	0-245	unmodified input speech frame
	246-255	not used
Page 11	0-123	half speech frame filled by the ISR
	124-127	not used
Page 12	0-81	output buffer for diagnostics on the oscilloscope
	82-122	low-pass filtered signal from last frame
	123-127	not used
Pages 13-511		not used
PRAM	0-7	external interrupt vector tables
	8-23	reserved for internal TMS320C25 use
	24-31	internal trap and interrupt vector tables
	32-1796	all system software except for the FFT subroutine
	1797-1851	Hamming window and low-pass filter coefficients
	1852-3381	not used
	3382-4095	used by FFT subroutine
	4096-65519	not used
	65520-65535	on chip RAM used by autocorrelation for MAC instruction

Table 6.1 - Data memory map for feature extraction system software running on the TMS320C25.

6.4.2. Register-Based Modifications

The eight auxiliary registers on the C25 have resulted in many incremental changes throughout the program, which when added up, have had the greatest effect in reducing overall system execution time.

All counters which were previously located in data memory are now stored in auxiliary registers. When a counter is in an auxiliary register, the BANC instruction can both test if it has reached zero and if it hasn't, automatically decrement it. This takes three cycles. When a counter in data memory is used, after it has been tested it must be loaded into the accumulator, decremented, then saved. This takes six cycles. Thus, a counter in an auxiliary register saves three cycles per loop iteration.

With all data arrays stored in data memory, we can use indirect addressing with an auxiliary register as a pointer. Previously, input and output buffers were stored in program memory and all pointers had to be directly manipulated in the accumulator. This meant loading the pointer into the accumulator, performing a table read/write, incrementing the pointer, and saving the pointer before the data could be manipulated. This took six cycles per iteration. The data pointer can now be manipulated via the auxiliary register free of time overhead and these six cycles are eliminated.

These modifications, possible because of the extra auxiliary registers, were made to most subroutines and are straightforward. They will not be described individually.

6.4.3. The New System Control Loop

Each subroutine, with the exception of the FFT, were modified as discussed above. With all data pointers and counters passed in auxiliary registers, their initialization was moved outside the individual subroutines into the system control loop.

As a result, each procedure was changed into a general purpose procedure which is passed the location and amount of data to operate on via the auxiliary registers. In a high level language this would be equivalent to removing global variable declarations. Each general purpose routine was easily tested for correctness in a test program (driver) and then added to the system software without modification. Again, the restrictions of the earlier C10 system meant that if a working system of the significant complexity involved was to be realized on such a system -- and it was -- then many non-standard programming practices had to be utilized.

For example, ZCD the zero crossing detector, is now a general purpose routine which can be used for any size array, located anywhere in memory, simply by passing it different values in its registers. Its correctness was checked in a separate test program which loaded dummy data with a known number of zero crossings which the ZCD result could be compared to.

By gathering all subroutine initialization into the system control procedure, and passing the values to each individual subroutine via the auxiliary registers, the location of all buffers can be easily monitored. That is, the main program loop is both easy to read and modify. For example, if the size of the input buffer is changed from 246 to 256, then only the system control loop has to be changed. The individual subroutines do not have to be considered.

6.4.4. The Interrupt Service Routine

Three of the eight auxiliary registers, AR6, AR7 and AR8, are used strictly by the ISR. They cannot be used by any other procedure in the system software. They are used for indirect addressing of the input and output buffers and as the output data counter. Swapping the registers in and out of data memory was thought unnecessary because it

would have added six more cycles per ISR call and required three more locations on page 0 when the system software did not require use of these registers. It would have been desirable to use one more auxiliary register as the input data counter. This would have meant a minimum overhead for both addressing and counting. Unfortunately, the fourth register could not be sacrificed from the main program and a data memory counter had to be used. This has the negative effects of using one more page 0 location and adding three additional cycles per ISR execution.

With the C25, the data memory page on which interrupts are generally serviced is page 0. This page contains 32 locations used for ISR variables and important data discussed earlier. It is not necessary to swap data in and out of program memory for data integrity during the ISR as was done on the C10 system. This considerably speeds up overall system execution time.

Finally, the SWDS/AIB system does not have a latch which is easily accessible. Therefore, the flag signal used to display the "picked" pitch and formants cannot be displayed. This is peculiar to the development system. Once a target system is obtained, the XF pin on the C25 itself will be used. Unfortunately, it is difficult to access the XF pin on the SWDS board due to the board design and its location within the PC cabinet.

6.4.5. The Repeat Feature and the Autocorrelation Routine

The C25 includes a repeat feature that allows a single instruction to be performed up to 256 times without additional looping overhead. The repeat counter (RPTC) is loaded with either a data memory value (RPT instruction) or an immediate value (RPTK instruction). Those instructions that are normally multicycle are pipelined when using the repeat feature, and effectively become single-cycle instructions. For example, the MAC (multiply-accumulate) instruction, which is heavily used in the autocorrelation

routine, is a four cycle instruction, but when repeated, becomes single cycle. The repeat feature was also used for data transfers and to create single instruction "in-line code".

The pipelined single cycle multiply-accumulate instruction (REPT/MAC), in conjunction with the 544 words of on-chip data memory, enabled the 3 autocorrelations (10 lag x 246 point, 4 lag x 82 point, and 50 lag x 82 point) to be performed using a single, general purpose routine. The total execution time of the three autocorrelations is 3.08 msec compared to 6.07 msec on the C10. This yields a 4:1 speed increase and a significant reduction in code space when compared to the separate, differently structured autocorrelation routines (AUTO1 and AUTO2/NORMAL) required on the C10 system.

The autocorrelation for formant extraction uses the full 246 point speech frame, compared to only 200 points used on the C10. This modification, made possible by the extra data memory available, makes formant calculations slightly more accurate.

6.5. System Operation and Timing

The clock timer on the SWDS operates at 8.0 KHz. Therefore, to collect one buffer of 246 samples takes 30.75 msec. As discussed earlier double buffering is employed with each speech frame being 30.75 msec (246 points) collected at a 65 Hz framing rate. Thus, all calculations must be completed within the 15.375 msec half-frame collection time. The module execution times (msec) and program code storage space (16-bit words) for the feature extraction system running on the SWDS are given in Table 6.2.

Program Module		Execution Time	Code Space
PROGRAM CONTROL		----	316
DIVNEW		----	84
SORT		----	28
ISR		0.56	27
ZCD	(246 pts)	0.50	17
GETDAT&FILTER	(123 pts)	0.80	59
PREEMF	(82 pts)	0.34	27&
WINDOW	(82 pts)	0.30	38^
AUTO	(4x82)	0.28	82\$
LG	(M=4)	0.14	230#
KTOA	(M=4)	0.04	130*
INVFIL	(82 pts)	0.30	25
WINDOW	(82 pts)	0.30	----^
AUTO	(81x82)	2.04	----\$
DECIS&PPIC	(82 pts)	0.20	214
PREEMF	(246 pts)	0.60	----&
WINDOW	(246 pts)	0.86	----^
AUTO	(10x246)	0.76	----\$
LG	(M=10)	0.34	----#
KTOA	(M=10)	0.10	----*
PREFFT	(M=10)	0.60	48
FFT	(64 cmplx pts)	1.20	714
MAGNTD	(64 to 32)	0.20	38
NPIC	(3 formants)	0.10	35
FTRACK	(3 formants)	0.20	68
data storage		----	385
free space		----	1531
free time		4.62	----
Total		15.38 msec	4096 words

ISR(0.56) + pitch(5.24) + formants(4.96) + free time(4.62)
= data collection time (123x125 μ sec) = 15.38 msec

Table 6.2 - TMS320C25 program module execution time and code space for the feature extraction system software running on the SWDS board at 20 MHz.

The enhancements made in going from the C10 to the C25 resulted in a system that operates in less than 70 percent of real-time at twice the frame rate.

The C25 contains an IDLE instruction which can power down the processor, except for the interrupt circuitry, to consume less power. The full-speed power consumption of the C25 can be reduced if the system IDLEs after system operations are complete and only the ISR is running, filling the next speech buffer. This implies that to lower overall power consumption, system execution time should be minimized, thus maximizing IDLE time.

6.6. Error Analysis

The data logging capabilities of the SWDS were used to run tests on the feature extraction program with both real and synthetic speech in non-real-time.

A formant synthesizer was used to generate synthetic speech to test the system software's LPC calculation accuracy. Various formant combinations corresponding to voiced sounds were converted into prediction coefficients via polynomial expansion. The polynomial was then used as the denominator of a filter corresponding to the vocal-tract transfer function, as per the speech model. An impulse train was passed through the filter to generate synthetic vowels. These were used as input to the system software and the LPC spectrum, calculated by the FFT during formant extraction, was output to disk.

The FFT in the system software calculates the LPC spectrum from prediction coefficients derived from the synthetic speech input. Thus, this LPC spectrum can be compared to a floating-point LPC spectrum derived from an FFT of the original prediction coefficients resulting from the polynomial expansion in the formant synthesizer. In this way, the accuracy of the fixed-point system software's LPC calculations can be demonstrated. Five examples are shown in Figure 6.4. The FFT of the synthesized speech itself is included in both plots as a reference point. As is

Figure 6.4 (a) through (e) - Five examples of LPC spectrum calculations with the TMS320C25 system software on synthetic speech.

demonstrated by the figures, the system software's calculation of the LPC spectra is virtually identical to the original synthesized speech spectrum being input.

Recorded speech segments of one second duration were passed through the system software and the speech features extracted were logged to disk. The most dramatic display of software accuracy is given in Figure 6.5 where the extracted formants for five speech segments are over-laid on top of the corresponding speech spectrographs, generated via the Real-time Speech Lab (RSL) developed by Prof. Morris [Mor88c]. It is clear from the results of Figure 6.5 that the system software is accurately extracting speech features.

From these plots of real speech, especially Figure 6.5(b), it is illustrated that the main problem is with weak formants. When a formant is too weak to be extracted, the next higher formant is substituted in its place. For example, in the figure at 0.7 seconds, F_3 is too weak to be extracted and F_4 is erroneously selected as F_3 . If only one frame is in error then the 3-point median smoother will smooth the erroneous second and third formant tracks. But, if the weak formant exists for more than one frame, as is the case in this example, then more powerful tracking rules are needed.

The results given in Figure 6.5, even though demonstrating accuracy, should be approached with caution. The nature of speech is stochastic and as a result, it is more important to know the system's limitations than its strengths. LPC analysis is known to be less accurate on the high-pitched voices of women and children, even though we show good results using both a women's voice, Figure 6.5(c), and a child's, Figure 6.5(e). Further, the voiced-unvoiced decisions were based on thresholds empirically determined in our laboratory. In other environmental conditions, with different background noise

Figure 6.5 (a) - Formant extraction using the TMS320C25 system software on the real speech test sentence "we were away a year ago ..." spoken by an adult male.

Figure 6.5 (b) - Test sentence "may we all learn .." spoken by an adult male.

Figure 6.5 (c) - Test sentence "we were away ..." spoken by an adult female.

Figure 6.5 (d) - Test sentence "one two three" spoken by an adult male.

Figure 6.5 (e) - Test sentence "we were away ..." spoken by a male child.

level characteristics, these thresholds may no longer be optimal. Thus, the accuracy of the voiced-unvoiced boundaries in the figures for real speech may be misleading.

6.7. Summary

The feature extraction program has been successfully transferred to the TMS320C25 digital signal processor. The advanced hardware on the C25 has lead to the system software being more readable, easier to maintain, and more accurate than the previous TMS32010-based systems. Further, the speech features are output at twice the rate. By utilizing the IDLE instruction, power consumption is minimized.

A target system has not been developed but several paths have been examined. The most promising path involves working with the research groups at Stanford and Research Triangle Institute. Work with these groups will help develop the necessary mapping algorithm of features to electrodes by using patients implanted with existing commercial cochlear implant systems. Collaboration with other groups and working with existing systems will help gain experience working with patients and give insight into problems encountered with implementation.

Testing of the system software on synthesized speech has shown that the LPC calculations, which are the fundamental basis for the system software, are performed with a high degree of accuracy. Testing of the system software on real speech has shown that the system software extracts pitch and formants accurately when using male speech with optimized voiced-unvoiced decision parameters.

Thus, the LPC model -- which is the basis of the feature extraction system -- is of reasonable accuracy, especially within the context of an 8 to 22 electrode system.

The following chapter describes future enhancements which could increase the accuracy of the features extracted, both in and out of the laboratory.

7. AREAS FOR FUTURE WORK

The development of a speech processor is an ongoing project. Any speech processor system should be easily adaptable to future developments in both digital signal processing and psycho-acoustics. This chapter recommends modifications to the existing speech processor system, specifically the C25 version, which would make it more robust and accurate.

7.1 Using the TI Assembler and Linker

A simple modification is to convert the system source code from the DSPS-Microsoft format to the standard Texas Instruments format. By using the TI assembler, the system software could be divided into manageable chunks which are assembled separately. Their object code would then be linked together with the TI linker [TMS86b]. The DSPS-Microsoft assembler does not have automatic linking capabilities. Dividing the software into manageable sections is facilitated by having each subroutine passed data in auxiliary registers.

Using a linker would make it easier for more than one person to work on the system software at a given time. Each person would modify one module at a time with all other modules fixed in object code. A degree of security would be added because a person working on the project would only see pertinent sections and not the entire source code. For example, if the group at Stanford University wanted our software running on their hardware they would only need to modify the global data declarations, hardware initialization, and interrupt service routine. They would then link their custom procedures to ours without seeing the bulk of our source code.

7.2 Formant Tracking

Given the LPC inverse spectrum, formant "peaks" show up as nulls; a simple null-picker is used to extract the first three formants; and the resultant formant trajectories are passed through a 3-point nonlinear median smoother. This method was shown to produce accurate formant trajectories. However, problems still arise with missing peaks, due to not enough energy in the speech segment, and two formants merging into one peak. Note that peak mergers do not have to be separated in our system, because two merged peaks would be mapped to the same electrode anyway. This situation should be recognized, however, so that when it occurs the fourth formant will not be mistaken for the third.

With null-picking, formant tracking rules are often applied to select the appropriate nulls as the first three formants and smooth any erroneous data. Formant tracking algorithms have not been implemented in the system software, but several have been examined for their suitability in our system.

The popular formant tracking algorithm put forth by McCandless suggests that the vocal tract transfer function, Equation 3.1, be recomputed at radii inside the unit circle to enhance the formants and hopefully separate two merged peaks [McC74], [Sen76], [Wag82]. Incorporated in this algorithm are rules to eliminate spurious and missing formants. However, her algorithm is unsuitable for implementation in our system because it requires backward and forward movement in time. This means all data is collected before it is processed and hence the algorithm does not run in real-time.

The most easily implemented algorithm for formant tracking discovered so far is Markel's [Mar73a]. With Markel's algorithm, the formant trajectory estimates for each frame are based solely upon the locations of the nulls in the corresponding LPC

spectrum. First the trajectories are estimated using only local information to satisfy continuity, and then smoothing is applied to insure that a maximum rate of change of the trajectories is not exceeded. Smoothing can be achieved with either a linear filter, as suggested by Markel, or with the nonlinear median filter suggested by Rabiner. One additional constraint is a maximum range of 3 kHz for the first three formant trajectories.

Other promising algorithms involve using linear predictive phase spectra [Red84] or hidden Markov models, and vector quantization [Kop86] have been proposed. But these algorithms are thought to be too complex for implementation on present-day digital signal processors. As a result, they will not be discussed.

Finally, Christensen et. al. found that picking peaks in the spectrum of the negative of the second derivative of the LPC spectrum is superior to using the LPC spectrum's nulls [Chr76]. This method was shown to be successful in separating merged peaks and could be used to obtain more accurate data (nulls) prior to applying a formant tracker.

7.3 Voiced-Unvoiced Decisions

Correct voicing state identification, that is voiced-unvoiced decisions, is critical in any speech analysis system. An adaptive linear discriminator has been proposed to discriminate the speech signal as voiced or unvoiced speech [Cam86]. This algorithm is recommended because it achieves accurate results, adapts its decision coefficients to slowly changing background environments, and can be implemented on present-day digital signal processors.

The voicing algorithm consists of: input speech low-pass filtering; voicing parameter measurement; voicing classification; and smoothing of voicing classifications.

The voicing parameters -- low-band speech energy, zero crossing rate, first reflection coefficient, pre-emphasized energy ratio, second reflection coefficient, causal pitch prediction gains, and noncausal pitch prediction gains -- are calculated on a half-frame basis. The measured signal parameters are combined into a value which determines the voicing for that half frame. The decision coefficients which combine the measured signal parameters are calculated manually with a training set (this training set is only performed once; not for every speaker). Different sets of coefficients are used depending upon the input signal to noise ratio. Finally, a modified median smoother is used to smooth the final voicing discrimination function.

Once an unvoiced decision is made, further feature extraction calculations do not have to be performed. This is desirable because by minimizing further calculations the system can utilize the IDLE instruction to minimize power consumption. By separating the V-UV decision from pitch extraction this decision can be reached quicker.

7.4 Dealing with Noise

A final limitation with LPC is that it is sensitive to additive noise. Since noise is not predictable, it will show up in the residual signal and may completely mask the excitation function. This would cause the SIFT algorithm to choose incorrect peaks in the autocorrelation function as the pitch. Further, it can be shown that in the presence of noise, the autocorrelation method of LPC estimation produces biased estimates of the LPC parameters [Ahm86]. Fortunately, there are numerous algorithms available to help overcome the bias in estimates. The most successful is reported to be preprocessing by least mean square error adaptive filtering [Ahm86].

Cochlear implant systems have primarily been regarded as lip-reading aids, though with "star" patients "blind" (e.g., telephone) conversations are possible. That is,

the combination of lip-reading plus electrical input to the cochlea has been shown to provide significantly better results than lip-reading alone. Thus, if the proposed system provides superior results in short distance, visual contact lip-reading situations, in which a relatively high speech to background noise ratio would be expected, then the system would be regarded as successful and the methods described in [Ahm86] could then be applied in situations where increased noise is present. The fact that LPC is utilized in many military speech systems, where mechanical background noises are prevalent, has led to many studies for reduction of noise interference in LPC.

7.5 Hardware Considerations

There has not been final speech processor hardware built for our system. This sub-section discusses a few considerations regarding a final digital signal processor, portability, and the difficulties to be encountered in the hardware design. Some of these points were touched upon in the last chapter.

The TMS320C25, at full speed, is powerful enough to be the digital signal processor for a cochlear implant speech analysis system. Other processors, such as the Motorola 56000/1, are also powerful enough, but would require a considerable amount of time to transfer the system software. New floating-point DSPs, such as the third generation TMS320C30, are attractive from a computational point of view, but are too power-hungry to be considered in pocketable, battery operated system. Thus, the C25, or any new processor which simply extends the C25's architecture should be targeted as the final DSP for our system.

The final hardware system has to be both portable and pocketable to gain patient acceptance. For battery operation, power consumption of the unit must be kept to a minimum. This will extend the operation time between battery changes and minimize the

number and size of the batteries needed to drive the device. To keep the unit's size down, chip count has to be minimized. This can be achieved by using VLSI chips which combine many functions. For example, the recently released TLC32040 codec from Texas Instruments alleviates the need for any "glue" logic to interface to the C25. A new version of the C25 contains 4K of EPROM, alleviating the need for any external program RAM chips. The Manchester encoder chip developed at Sherbrooke University integrates the circuitry needed to interface to the Sherbrooke stimulator (except for a single count-down timer and the external transmitter).

The functionality and most of the components for a final target system have been defined. One foreseeable difficulty in developing the final hardware will be separating the analog and digital lines on the printed circuit board. Good analog-digital separation is required to minimize the noise picked up on the analog lines which would subsequently be superimposed on the input speech signal. This is a non-trivial task which will require many trial and error circuit board designs.

7.6 Conclusions

This section has discussed some areas for future development with respect to the speech processor discussed in this thesis. Several algorithmic enhancements were recommended including Markel's formant tracker, the US government's LPC-10E voiced-unvoiced decision rules, and Ahmed's noise suppression algorithms. A few final points regarding hardware implementation were also discussed including selecting a final programmable DSP, component selection, portability, and circuit board layout.

A final recommendation regarding work on the speech processor is that work be continued quickly, otherwise other groups will catch up on our lead time. Nucleus Corp. is currently developing a speech processor using an $F_0/F_1/F_2/F_3$ strategy -- our original

premise circa 1984. However, they are using a digital filter bank technique to obtain the pitch and formants and have not yet realized a portable $F_0/F_1/F_2/F_3$ system. It will not be long before they realize more accurate feature extraction techniques.

BIBLIOGRAPHY

- [Ada83] WT Adams, J Brady, "Magnitude approximations for microprocessor implementation", *IEEE Micro*, vol 3, pp 27-31, October 1983.
- [Ahm86] MS Ahmed, AK Abbasi, "Comparison of LPC Estimation Algorithms from Noisy Speech", *Proceedings*, 111th Meeting of the Acoustical Society of America, May 12-16, 1986.
- [AIB86] *TMS32010 Analog Interface Board User's Guide*, Preliminary, Digital Signal Processing Products, Texas Instruments, Huston Texas, AIB Part Number RTC/EVM320C-06 Rev 1.0, December 1986.
- [Atal76] BS Atal, LR Rabiner, "A Pattern Recognition Approach to Voiced-Unvoiced-Silence Classification with Applications to Speech Recognition", *IEEE Trans. ASSP*, vol. ASSP-24, pp 201-212, June 1976.
- [Bos86] JL Bosloy, "TMS 320 Hardware/Software for a Cochlear Implant", 94.498 *Project Report*, Supervised by Prof. L.R. Morris, Dept. of Systems and Computer Eng., Carleton University, Ottawa, April 1986.
- [Cam86] JP Cambell, Jr., TE Tremain, "Voiced/Unvoiced Classification of Speech with Applications to the U.S. Government LPC-10E Algorithm", *Proceedings*, ICASSP, Tokyo, Japan, paper 9.11, 1986.
- [Cha74] S Chandra, WC Lin, "Experimental Comparison Between Stationary and Nonstationary Formulations of Linear Prediction Applied to Voiced Speech Analysis", *IEEE Trans. ASSP*, vol. ASSP-22, pp 403-415, December 1974.
- [Chr76] RL Christensen, et al., "A Comparison of Three Methods of Extracting Resonance Information from Predictor-Coefficient Coded Speech", *IEEE Trans. ASSP*, vol. ASSP-24, pp 8-14, February 1976.
- [Dar84] I Darian-Smith (Ed.), *Handbook of Physiology, Section 1: The Nervous System*, Volume III, Part 2, American Physiological Society, Bethesda, Maryland, 1984.
- [Duv82] F Duval, et. al, "A Stimulation Module for an Auditory Prosthesis", *Internal Status Report*, 1982.
- [Duv84] F Duval, et. al, "Design and Test of the Single Chip Multichannel Stimulator", *Internal Status Report*, July 1984.
- [Duv86] F Duval, et. al, "Integrated Manchester Encoder", *Internal Status Report*, 1986.
- [Edd86] DK Eddington, "Results with the Symbion prosthesis", *Proceedings in press*, International Cochlear Implant Symposium and Workshop, supplement to Annals Otology, Rhinology and Laryngology, 1986.

- [EVM85] *TMS32010 Evaluation Module User's Guide*, Digital Signal Processing Products, Texas Instruments, Huston Texas, Part Number SPRU005A, 1985.
- [Fei87] E Feigenbaum, "Cochlear Implant Devices for the Profoundly Hearing Impaired", *IEEE Engineering in Medicine and Biology Magazine*, Vol. 6, No. 2, pp 10-21, June 1987.
- [Fin87] CC Finley, *personal communication*, Research Triangle Institute, 1987.
- [Gei87] CD Geisler, "Coding of Acoustic Signals on the Auditory Nerve", *IEEE Engineering in Medicine and Biology Magazine*, Vol. 6, No. 2, pp 22-28, June 1987.
- [Gray74] AH Gray, Jr., JD Markel, "A Spectral-Flatness Measure for Studying the Autocorrelation Method of Linear Prediction of Speech Analysis", *IEEE Trans. ASSP*, vol. ASSP-22, pp 207-217, June 1974.
- [Har87a] RV Harrison, "Cochlear implants: a review of the principles and important physiological factors", *The Journal of Otolaryngology*, pp 268-275, May 1987.
- [Har87b] RV Harrison, LR Morris, et. al., "Technical development of an implantable cochlear prosthesis in Canada", *The Journal of Otolaryngology*, pp 311-315, May 1987.
- [Hess83] W Hess, *Pitch Determination of Speech Signals: Algorithms and Devices*, Springer-Verlag, Berlin, Germany, 1983.
- [Kna88] RB Knapp, B Townshend, "A Real-time Digital Speech Processing System for an Auditory Prosthesis", *Proceedings, ICASSP*, New York, A1.1, pp 2493-2495, April 1988.
- [Kop86] GE Kopec, "Formant Tracking Using Hidden Markov Models and Vector Quantization", *IEEE Trans. on ASSP*, vol. ASSP-34(6), pp 709-729, August 1986.
- [Kuf84] SW Kuffler, JG Nicholls, AR Martin, *From Neuron to Brain: A Cellular Approach to the Function of the Nervous System*, Second Edition, Sinauer Associates Inc., Sunderland Massachusetts, 1984.
- [Leu87] J Leung, "Cross Assembler for TMS32020 and Software Transport from TMS32010 to TMS32020", *94.498 Project Report*, Supervised by Prof. L.R. Morris, Dept. of Systems and Computer Eng., Carleton University, Ottawa, April 1987.
- [LG77] J Le Roux, C Gueguen, "A Fixed Point Computation of Partial Correlation Coefficients", *IEEE Trans. ASSP*, vol. ASSP-25, pp 257-259, June 1977.
- [Mak75] J Makhoul, "Linear Prediction: A Tutorial Review", *Proc. IEEE*, vol. 63, pp 561-580, April 1975.

- [Mar72a] JD Markel, "Digital Inverse Filtering--A New Tool for Formant Trajectory Estimation", *IEEE Trans. on Audio and Elect.*, vol AU-20, pp 129-137, June 1972.
- [Mar72b] JD Markel, "The SIFT Algorithm for Fundamental Frequency Estimation", *IEEE Trans. on Audio and Elect.*, vol. AU-20, pp 367-377, December 1972.
- [Mar73a] JD Markel, "Application of a Digital Inverse Filter for Automatic Formant and F_0 Analysis", *IEEE Trans on Audio and Elect.*, vol AU-21, pp 154-160, June 1973.
- [Mar73b] JD Markel, AH Gray, Jr., H Wakita, "Linear Prediction of Speech-Theory", *Speech Communications Research Laboratory, Inc.*, Santa Barbara, California, SCRL Monograph No. 10, September 1973.
- [Mar74] JD Markel, AH Gray, Jr, "A Linear Prediction Vocoder Simulation Based upon the Autocorrelation Method", *IEEE Trans. ASSP*, vol. ASSP-22, pp 124-134, April 1974.
- [Mar76] JD Markel, AH Gray, Jr, *Linear Prediction of Speech*, Springer-Verlag, Berlin, Germany, 1976.
- [McC74] SS McCandless, "An Algorithm for Automatic Formant Extraction Using Linear Prediction Spectra", *IEEE Trans. ASSP*, vol. ASSP-22, pp 135-193, April 1974.
- [McG77] CA McGonegal, LR Rabiner, et al., "A Subjective Evaluation of Pitch Detection Methods Using LPC Synthesized Speech", *IEEE Trans. ASSP*, vol. ASSP-25, pp 221-229, June 1977.
- [Mer85] MM Merzenich, "UCSF Cochlear Implant Device", in: *Cochlear Implants*, RA Schindler, MM Merzenich, Eds, Raven Press, New York, pp 121-129, 1985.
- [Mic85] *Microsoft Macro Assembler for the MS-DOS Operating System: User's Guide*, Version 4.0, Microsoft Corporation, Part Number 016-014-023, October 1985.
- [Mic85] RP Michelson, "Cochlear Implants: Personal Perspectives", in: *Cochlear Implants*, RA Schindler, MM Merzenich, Eds, Raven Press, New York, pp 9-11, 1985.
- [Moo82] BCJ Moore, *An Introduction to the Psychology of Hearing*, Second Edition, Academic Press, London, 1982.
- [Mor85a] LR Morris, *Digital Signal Processing Software (Texas Instruments' TMS 320 & DEC J-11)*, Fifth Edition, DSPS Inc, Ottawa, Ontario, January 1985.
- [Mor85b] LR Morris, "Structural considerations for large FFT programs on the TMS32010", *Proceedings, ICASSP*, pp 1648-1651, April 1985.

- [Mor86] LR Morris, "Time-space optimization of TMS32010 software for Le Roux-Gueguen LPC matrix inversion and k-to-a parameter transformation", *Proceedings, IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, New York, September 1986.
- [Mor87a] LR Morris, PA Barszczewski, JL Bosloy, "Algorithm Selection and Software Optimization for a DSP micro-based Speech Processor for a Multi-Electrode Cochlear Implant", *Proceedings, IEEE ICASSP*, pp 1648-1651, April 1987, Dallas.
- [Mor87b] LR Morris, PA Barszczewski, "A Programmable Signal Processing/Stimulation System for a Multi-Electrode Cochlear Implant", *Proceedings, Speech Technology Conference*, pp 305-307, April 1987, New York.
- [Mor87c] LR Morris, PA Barszczewski, "Algorithms, Hardware, and Software for a DSP Microcomputer-Based Multi-Electrode Cochlear Implant System", *Proceedings, Ninth Annual IEEE EMBS Conference*, November 1987, Boston.
- [Mor88a] LR Morris, PA Barszczewski, "Design and Evolution of a Powerful Pocket-sized DSP Speech Processing System for a Cochlear Implant and Other Hearing Prosthesis Applications", *Proceedings, IEEE ICASSP*, New York, A1.7, pp 2516-2519, April 1988.
- [Mor88b] LR Morris, PA Barszczewski, "Algorithms, Hardware and Software for a Digital Signal Processor Microcomputer-Based, Speech Processor in a Multi-Electrode Cochlear Implant System", in press, *IEEE Trans. BME*, July 1988.
- [Mor88c] LR Morris, *personal communication*, Digital Signal Processing Software (DSPS) Ltd., Nepean, Ont., 1988.
- [Rab75] LR Rabiner, et al., "Applications of a Nonlinear Smoothing Algorithm to Speech Processing", *IEEE Trans. on ASSP*, vol. ASSP-23, pp 552-557, December 1975.
- [Rab76] LR Rabiner, et al., "A Comparative Performance Study of Several Pitch Detection Algorithms", *IEEE Trans. on ASSP*, vol. ASSP-24, pp 399-418, October 1976.
- [Rab77a] LR Rabiner, "On the Use of Autocorrelation Analysis for Pitch Detection", *IEEE Trans. on ASSP*, vol. ASSP-25, pp 24-33, February 1977.
- [Rab77b] LR Rabiner, et al., "Evaluation of a Statistical Approach to Voiced Unvoiced-Silence Analysis for Telephone-Quality Speech", *Bell Syst. Tech. J.*, vol. 56, pp 455-482, March 1977.
- [Rab77c] LR Rabiner, et al., "LPC Prediction Error - Analysis of Its Variation with the Position of the Analysis Frame", *IEEE Trans. on ASSP*, vol. ASSP-25, pp 434-442, October 1977.

- [Red84] NS Reddy, MNS Swamy, "High-Resolution Formant Extraction from Linear-Prediction Phase Spectra", *IEEE Trans on ASSP*, vol. ASSP-32, pp 1136-1144, December 1984.
- [Sai85] S Saito and K Nakata, *Fundamentals of Speech Signal Processing*, Academic Press, Orlando, Florida, 1985.
- [Sen76] SS Seneff (M. Landless), "Modifications to Formant Tracking Algorithm of April 1974", *IEEE Trans. on ASSP*, vol. ASSP-24, pp 192-193, April 1976.
- [Sie79] LJ Siegel, "A Procedure for Using Pattern Classification Techniques to Obtain a Voiced/Unvoiced Classifier", *IEEE Trans. on ASSP*, vol. ASSP-27, pp 83-89, February 1979.
- [Sky86] *2400 BPS Full Duplex Linear Predictive Vocoder Model LPC-320, Operating & Maintenance Manual*, Rev 1, SkyWave Electronics Ltd., Kanata Ontario, September 1986.
- [TMS86a] *Digital signal Processing Applications with the TMS320 Family: Theory, Algorithms, and Implementations*, Texas Instruments, Huston Texas, Part Number SPRA012, 1986.
- [TMS86b] *TMS320C25 User's Guide*, Preliminary, Digital Signal Processor Products, Texas Instruments, Huston Texas, Part Number SPRU012, 1986.
- [TMS87] *First Generation TMS320 User's Guide*, Digital Signal Processor Products, Part Number SPRU013, Texas Instruments, Houston, Texas, January 1987.
- [Tong87] YC Tong, et. al., "Speech Processor for Auditory Prostheses", 6th Quarterly Progress Report, *NIH Contract N01-NS-5-2388*, University of Melbourne, January 1 through March 31, 1987.
- [Tow87] B Townshend, RL White, "Reduction of Electrical Interaction in Auditory Prostheses", *IEEE Trans. BME*, Vol BME-34, No. 11, pp 891-897, November 1987.
- [vdH87] C van den Honert, P Stypulkowski, "Coding of Electrical Stimuli on the VIIIth Nerve", *IEEE Engineering in Medicine and Biology Magazine*, Vol. 6, No. 2, pp 33-35, June 1987.
- [vdP86] P van der Puije, et. al., "Cylindrical Cochlear Electrode Array for use in Humans", submitted for Publication, *Annals of Otolaryngology, Rhinology, and Laryngology*, November 1986.
- [vdP87] P van der Puije, F Duval, J. Morris, "The Canadian Cochlear Implant Project: Design Objectives and Development", *Proceedings, International Cochlear Implant Symposium*, Stadthalle Durin, in press, September 1987.
- [Wag82] M Wagner, "Formant Extraction Algorithm in Error", *IEEE Trans. on ASSP*, vol. ASSP-30, p 520, June 1982.

- [Whi86] RL White, LJ Dent, RB Knapp, et. al., "Development of a Portable Speech Processor for a Multiple-Channel Auditory Prosthesis", 1st-4th Quarterly Progress Report, *NIH Contract N01-NS-5-2395*, Stanford University, October 1, 1985 through September 30, 1986.
- [Whi87] RL White, "System Design of a Cochlear Implant", *IEEE Engineering in Medicine and Biology Magazine*, Vol. 6, No. 2, pp 42-49, June 1987.
- [Wil86] BS Wilson, CC Finley, DT Lawson, "Speech Processors for Auditory Prostheses", 1st-5th Quarterly Progress Report, *NIH Contract N01-NS-5-2396*, Research Triangle Institute, September 27, 1985 through December 26, 1986.
- [Wong80] DY Wong, et al., "Spectral Mismatch Due to Pre-emphasis in LPC Analysis/Synthesis", *IEEE Trans. on ASSP*, vol. ASSP-28, pp 263-264, April 1980.

APPENDICES

Appendix A

TMS32010 source code listing of the feature extraction system software for the EVM and CRC board.

```

PAGE 63,132
;
;----- FTR12.ASM -----
;
; COPYRIGHT (C) MAY 1988 PETER A. BARSZCZEWSKI, CARLETON UNIVERSITY,
; ALL RIGHTS RESERVED.
;
;
;      IF1
;      INCLUDE C:\ASM\TIMACG.ASM
;      INCLUDE C:\ASM\W200.MAC
;      INCLUDE C:\ASM\W82.MAC
;      END.F
;
;
;----- CONSTANTS -----
;
;      .RADIX 10
;
;      CRC=1                      ;CONDITIONAL ASSEMBLY TEST VARIABLE
;                                ;0 (FALSE)-EVM 1 (TRUE)-CRC
;
;      DISPLAY MESSAGE DURING ASSEMBLY SO USER KNOWS WHATS GOING ON
;
;      IF CRC
;      IF1
%OUT *          ASSEMBLING FOR CRC BOARD
;      ENDIF
;      ELSE
;      IF1
%OUT *          ASSEMBLING FOR EVM BOARD
;      ENDIF
;      ENDIF
;
;      FFTLOC=00D3DH ;STARTING ADDRESS OF FFT
;
;      NPTS1=246D    ;NUMBER OF POINTS COLLECTED AND USED FOR PITCH EXTRACTION
;      M1=4D         ;ORDER
;      OUTB1=NPTS1/3 ;NUMBER OF POINTS IN PITCH'S OUTPUT BUFFER
;      W1=OUTB1/2    ;NUMBER OF POINTS IN PITCH'S WINDOW FUNCTION
;
;      JMI=4D        ;LOWER LIMIT FOR PEAK PICKER
;      JMA=41D       ;UPPER LIMIT FOR PEAK PICKER
;      IZM=100D      ;MAX NUMBER OF ZERO CROSSINGS FOR VOICED = 47
;      IRI=4D        ;THRESHOLD FOR SIGNAL RMS
;
;      NPTS2=200D    ;NUMBER OF POINTS USED FOR FORMANT EXTRACTION
;      M2=10D        ;ORDER
;      OUTB2=64D     ;NUMBER OF POINTS IN ENVELOPE'S OUTPUT BUFFER
;      W2=NPTS2/2    ;NUMBER OF POINTS IN ENVELOPE'S WINDOW FUNCTION
;
;      IT=8D         ;FOR AUTO2'S BUFFERING SCHEME
;      NR=NPTS2/IT
;      BEND=NR+M2-1
;
;      NSPECT=32     ;# OF BYTES IN MAGNITUDE OF SPECTRUM
;
;
;----- MACROS -----
;
;      DEFBF MACRO    I,J1,J2

```

```

DATA      B&I
SAME      NA&J1
DATA      XD&I
SAME      NA&J2
ENDM

;

DEF MACRO      I,J
DATA      LOW&I
SAME      FK&J
DATA      HIGH&I
SAME      RX&J
ENDM

;

CALFFT MACRO
DW      0F800H
DW      FFTLOC
ENDM

;

TRGEN1 MACRO      I
DW      TRI&I
ENDM

;

TRGEN2 MACRO      I
DW      TSI&I
ENDM

;

;----- PAGE 0 -----
;

DORG0

;

I=1
REPT      M2+1
J1=2*I-1
J2=2*I
DEFBFB      %I,%J1,%J2
I=I+1
ENDM

;

;WARNING:  NO, N1, N2, N3, NMIN, NMAX MUST BE CONTIGUOUS
;

RES      15D                ;AUTO2 USES DRAM 0 TO 37D
DATA      Y1                ;FILTER
SAME      NO                ;DECIS
DATA      Y11               ;FILTER
SAME      N1                ;DECIS
DATA      Y2                ;FILTER
SAME      N2                ;DECIS
DATA      Y21               ;FILTER
SAME      N3                ;DECIS
DATA      Y22               ;FILTER
SAME      NMIN              ;DECIS
DATA      Y3                ;FILTER
SAME      NMAX              ;DECIS
DATA      Y31               ;FILTER
SAME      L                 ;DECIS
DATA      Y32               ;FILTER
SAME      ALI               ;DECIS
DATA      LL                ;DECIS
SAME      NRCNT             ;AUTO2
DATA      AP                ;DECIS

```

```

SAME      NRD                ;AUTO2
DATA      LAST               ;ZCD,WINDAT
SAME      LAGCNT             ;AUTO2
DATA      IX                 ;PPIC
SAME      Y                  ;LG
DATA      LGFLAG             ;LG,KTOA
DATA      WPTR               ;GETDAT,WINDAT
DATA      WIND               ;GETDAT,WINDAT
DATA      WONE               ;GETDAT,WINDAT
DATA      BB                 ;DECIS
DATA      SLM1               ;DECIS
DATA      SL                 ;DECIS
DATA      SLP1               ;DECIS
DATA      BSTRT1             ;NPIC
DATA      BSTRT2             ;NPIC
DATA      BSTRT3             ;NPIC
;
I=0
J=1                ;DEFINE FK's, RX's
REPT      M2+1
DEF       %I,%J
I=I+1
J=J+1
ENDM
;
;NOTE:  AUTO1 USES 0-81D DRAM - THE VARIABLES BELOW MUST SURVIVE!
;
DATA      ONE                 ;GENERAL PURPOSE CONSTANT
DATA      DPTR               ;GENERAL PURPOSE PRIMARY DATA POINTER
DATA      PTR2               ;GENERAL PURPOSE SECONDARY DATA POINTER
DATA      COUNT              ;GENERAL PURPOSE COUNTER
DATA      TEMP               ;GENERAL PURPOSE TEMPORARY VARIABLE
DATA      SIG                ;GENERAL PURPOSE DATA SAMPLE
DATA      XLOW               ;GETDAT,ZMEAN,AUTO1
DATA      XHIGH              ;GETDAT,ZMEAN,AUTO1
DATA      DR                 ;DIVIDE
DATA      DDH                ;DIVIDE
DATA      DDL                ;DIVIDE
DATA      PLUS1              ;DIVIDE
DATA      TEMP2              ;DIVIDE
DATA      DUMMY              ;DIVIDE, LG
DATA      ZMASK              ;AUTO
DATA      DELTA              ;AUTO
DATA      SPTR2              ;AUTO
DATA      SCOUNT             ;AUTO
DATA      ENRLO              ;AUTO      (LOW 16 BITS OF ENERGY)
DATA      ENRHI              ;AUTO      (HIGH 16 BITS OF ENERGY)
DATA      BASE               ;AUTO1
DATA      TEMPN              ;AUTO1
DATA      QTEMP              ;AUTO1
DATA      POWER              ;AUTO1
DATA      DP1                ;AUTO1
DATA      NZ                 ;ZCD
DATA      IRO                ;MAIN,DECIS
DATA      MAXVAL             ;NPIC, PPIC
DATA      RXT                ;MAGNTD
DATA      IYT                ;MAGNTD
;
;
;----- PAGE 1 -----
;

```

```

DORG1
;
RES      4D                ; 4 SPACES FOR THE FFT
DATA     INUMBR            ; USED BY BOTH THE FFT AND PREFFT
;
DATA     ISMPL             ; USED BY BOTH FFT AND INPUT ISR
DATA     ITEMP
DATA     ICOUNT
DATA     IPTR
DATA     PSCOPE
DATA     CSCOPE
DATA     IFLAG
;
DATA     ISTATE
DATA     IONE              ; USED BY INPUT ISR ONLY
DATA     IHIGH
DATA     ILOW
;
;
;----- PRAM -----
;
AORG                ; START PROGRAM SECTION
;
START:
;
IF CRC
    DW      080H      DUP(?) ; OFFSET FOR CRC BOARD'S DOWNLINE LOADER
ENDIF
;
B          GO
B          INPUT
;
ATABLE:    ADDR  FTABLE        ; FILTER CONSTANT TABLE
AWIND1:    ADDR  WIND1         ; WINDOW FUNCTION TABLE FOR PITCH
AWIND2:    ADDR  WIND2         ; WINDOW FUNCTION TABLE FOR FORMANTS
ACNVRT:    ADDR  CNVRT         ; MU-LAW TO 2'S COMPL. CONVERSION TABLE
ATARG:     ADDR  TARGET        ; BASE ADDRESS CALA IN AUTO1
;
ACOEFS:    DW      8          DUP (0) ; DELAY LINE STORAGE FOR DOWNSAMPLING FILTER
;
APREBF:    DW      5          DUP (0) ; VARIABLES FOR INITIAL 5 RX'S FOR PITCH
;
; DECISION VARIABLES FOR DECIS
ADECT:     DW      3          DUP (0) ; N1 = 0, N2 = 0, N3 = 0
          DW      JMI         ; NMIN = JMI
          DW      JMA         ; NMAX = JMA
;
ASCP1:     ADDR  SCOPE1        ; SCOPE BUFFERS
ASCP2:     ADDR  SCOPE2
;
AB3:       ADDR  BUFF3
AB1:       ADDR  BUFF1         ; INPUT BUFFERS
AB2:       ADDR  BUFF2
AB4:       ADDR  BUFF4
;
ACOUNT:    DW      0          ; RESERVE SWAP STORAGE FOR FFT AND
AIPTTR:    DW      0          ; INTERRUPT ROUTINE
APSCOPE:   DW      0
ACSCOPE:   DW      0
AFLAG:     DW      0
AFFTSTG:   DW      0,0,0,0,0,0,0,0

```



```

;
APITCH: DW 0 ;STORAGE FOR ADDRESS OF PEAKS AND FORMANTS
AFMT1 DW 0 ; IN THE SCOPE BUFFER
AFMT2 DW 0
AFMT3 DW 0
AFMT4 DW 0
;
;
;----- INITIALIZATION -----
;
GO: LDPK 1
;
LACK 1 ;CONSTANT ON PAGE 1
SACL IONE
;
LDPK 0
SACL ONE ;CONSTANT ON PAGE 0
;
;LOAD SAMPLING ISR DATA
;
ZAC ;TOGGLE FLAG FOR DATA INPUT ON PAGE 1
SACL TEMP
LACKA AFLAG
TBLW TEMP
;
LACKA ASCP1
TBLR TEMP ;POINTER FOR DISPLAY OF SCOPE BUFFER
LACKA APSCOPE
TBLW TEMP
;
LACK (OUTB1+OUTB2)
SACL TEMP ;COUNTER FOR DISPLAY OF SCOPE BUFFER
LACKA ACSCOPE
TBLW TEMP
;
LACKA AB4
TBLR TEMP ;INITIALIZE INPUT BUFFER POINTER
LACKA AIPTR
TBLW TEMP
;
LACK NPTS1
SACL TEMP ;INITIALIZE INPUT BUFFER COUNTER (WHICH
LACKA ACCOUNT ; IS ALSO A FLAG TO START COLLECTING)
TBLW TEMP
;
; INITIALIZE I/O PORTS FOR CRC OR EVM
;
IF CRC
LACK 3D ;FOR CRC BOARD ENABLE CODEC A/D INTERRUPT
SACL TEMP ; AND TURN DRIVE OFF (DRIVE = 0)
OUT TEMP,PA6
ELSE
LACK 89D
SACL TEMP
LAC ONE,S9 ;N = 512
SUB TEMP ;N = 512-89 = 423 --> F = (19.07*N) HZ
SACL TEMP ; FREQ. = 8067 HZ; PERIOD = 124 USEC
OUT TEMP,PA1 ;LOAD CLOCK COUNTER ON EVM
;
LACK 10D ;CDA1=1, CAD=1 (P. 3-6 OF PRELIMINARY MANUAL)
SACL TEMP

```

```

        OUT      TEMP,PAO      ;D/A IN TRANSPARENT MODE, START A/D
    ENDIF
;
        EINT                ;ENABLE INTERRUPTS
;
;----- THE MAIN PROGRAM -----
;
HERE:   LACKA    ACOUNT      ;FINISH COLLECTING DATA
        TBLR     TEMP
        LAC      TEMP
        BNZ      HERE
;
        CALL     CONVRT      ;SWAP BUFFER, CONVERT MU-LAW TO 2'S COMPL.
                                ; WHILE DATA IS NOT BEING COLLECTED
;
        LACKA    AB4
        TBLR     TEMP        ;INITIALIZE INPUT BUFFER POINTER
        LACKA    AIPTR
        TBLW     TEMP
;
        LACK     NPTS1
        SACL     TEMP        ;INITIALIZE INPUT BUFFER COUNTER (WHICH
        LACKA    ACOUNT      ; IS ALSO A FLAG TO START COLLECTING)
        TBLW     TEMP
;
;----- PITCH EXTRACTION CALCULATIONS -----
;
PSTRT:  CALL     ZCD          ;COUNT NUMBER OF ZERO CROSSINGS
;
        LARK     ARO,Y1      ;READ IN GETDAT'S DELAY LINE FROM LAST FRAME
        LARK     ARI,7
        LACKA    ACOEFS
LEWP1:  LARP     0
        TBLR     *+,ARI
        ADD      ONE
        BANZ     LEWP1
;
        CALL     GETDAT      ;FILTER, WINDOW AND DOWNSAMPLE PITCH'S INPUT
        LARK     ARO,(W1-1)  ;BACKWARD COUNT
        ZAC
        SUB      WONE        ;BACKWARD THRU WINDOW FUNCTION
        SACL     WONE
        CALL     FLOOP       ;CALL MIDDLE OF GETDAT
;
        LARK     ARO,Y1      ;STORE GETDAT'S DELAY LINE FOR NEXT FRAME
        LARK     ARI,7
        LACKA    ACOEFS
LEWP2:  LARP     0
        TBLW     *+,ARI
        ADD      ONE
        BANZ     LEWP2
;
        LACK     OUTB1       ;DIVISOR IN (RUNNING SUM)/(NUMBER OF POINTS)
        SACL     DR
        LARK     ARO,(OUTB1-1) ;NUMBER OF POINTS IN I/O BUFFER
        LACKA    AB3         ;POINTER TO DATA IN AND OUT
        TBLR     DPTR
        CALL     ZMEAN       ;CONVERT TO ZERO MEAN
;

```

```

        LACKA  APREBF
        SACL  SPTR2      ;1'ST TIME USE APREBF BUFFER
        LACK  4
        CALL  AUTO1      ;4 LAG AUTO
;
        ZALS  ENRLO      ;INPUT ENERGY (32 BITS)
        ADDH  ENRHI
        IF CRC
            SUB  ONE,S14   ;THRESHOLD FOR CRC'S 14 BIT SAMPLES - 2**15
        ELSE
            SUB  ONE,S11   ;THRESHOLD FOR EVM'S 12 BIT SAMPLES - 2**11
        ENDIF
        BGEZ  AD63        ;ENERGY HIGH ENOUGH
        ZAC
        SACL  IR0         ;TELL DECIS UNVOICED
        B     AD50
;
AD63:   LACKA  APREBF      ;5 COEFS IN PRE-BUFFER
        TBLR  RX1
        ADD   ONE
        TBLR  RX2
        ADD   ONE
        TBLR  RX3
        ADD   ONE
        TBLR  RX4
        ADD   ONE
        TBLR  RX5
;
        LACK  OD         ;FLAG LG FOR 5TH ORDER AND RETURN HERE
        SACL  LGFLAG
        B     LG
;
LGRET1: CALL  KTOA        ;FK'S TO NA'S
;
        CALL  INVFIL      ;FILTER USING NA'S
;
        LACK  255D
        SACL  IR0         ;FAKE THIS FOR NOW
;
        LACKA  ASCP1
        TBLR  SPTR2      ;SECOND TIME USE SCOPE BUFFER
        LACK  (OUTB1-1)
        CALL  AUTO1      ;AUTOCORRELATE FILTERED SIGNAL
;
AD50:   CALL  DECIS       ;PITCH DECISION ROUTINE
;
;----- FORMANT EXTRACTION CALCULATIONS -----
;
FSTRT:  CALL  WINDAT      ;WINDOW ENV'S INPUT
;
        LARK  AR0,(W2-1)  ;BACKWARD COUNT
        ZAC
        SUB   WONE       ;BACKWARD THRU WINDOW FUNCTION
        SACL  WONE
        CALL  MIDDLE     ;CALL MIDDLE OF WINDAT
;
        LACK  OUTB2      ;DIVISOR IN (RUNNING SUM)/(NUMBER OF POINTS)
        SACL  DR
        LARK  AR0,(OUTB2-1) ;NUMBER OF POINTS IN I/O BUFFER
        LACKA AB2        ;POINTER TO DATA IN AND OUT

```

```

        TBLR    DPTR
CALL     ZMEAN          ;CONVERT TO ZERO MEAN
;
        CALL    AUTO2          ;11 LAG AUTO
;
        LARK    ARO,LOW0        ;NORMALIZE THE RX'S
        LARK    AR1,M2-1
        CALL    NORMAL
;
        LACK    1D              ;FLAG LG FOR 11TH ORDER AND RETURN HERE
        SACL    LGFLAG
        B       LG              ;RX'S TO FK'S
;
LGRET2:  LAC     ONE,S13         ;MODIFY FK FOR BETTER FORMANT PEAKING
        SACL    FK11
;
        CALL    KTOA            ;FK'S TO NA'S
;
        CALL    PREFFT          ;PREPARE NA'S AND DRAM FOR FFT
;
        CALFFT          ;DETERMINE FREQ. SPECTRUM USING 11 NA'S
        ADDR    BUFF2
;
        CALL    MAGNTD          ;CALCULATE MAGNITUDE OF SPECTRUM
;
        CALL    NPIC            ;FIND FORMANTS IN SPECTRUM
;
        B       HERE            ;AND GO AGAIN
;
;
;----- CONVERT -----
;          EVM - SWAPS INPUT BUFFERS
;          CRC - SWAPS INPUT BUFFERS AND CONVERTS MU-LAW TO 2'S COMP.
;
        IF CRC
;
; ***** CONVERT FOR CRC BOARD *****
;
CONVRT:  LACKA    AB4              ;ADDRESS OF INPUT BUFFER
        TBLR     DPTR
        LACKA    AB1              ;ADDRESS OF PROCESSING BUFFER
        TBLR     PTR2
        LARK     ARO,(NPTS1-1)    ;NUMBER OF SAMPLES TO BE TRANSFERED
        ZAC
        SACL     COUNT            ;FLAG FOR HIGH/LOW BYTE OF INPUT DATA
        LARP     0D
;
CLOOP:   LAC      DPTR              ;READ SAMPLE FROM INPUT BUFFER
        TBLR     SIG
        ADD      COUNT            ;INCRIMENT POINTER ONLY IF LOW BYTE
        SACL     DPTR
;
        LAC      ONE
        XOR      COUNT            ;TOGGLE FLAG
        SACL     COUNT
        BZ       CONT3            ;FIRST HIGH BYTE, THEN LOW BYTE
;
HGHBYT:  LAC      SIG,S8           ;GET HIGH BYTE FROM SIG
        SACH     SIG
;
CONT3:   LACKA    ACNVRT           ;CONVERT MU-LAW SAMPLE TO 2'S COMPL.

```

```

        TBLR    TEMP
        LACK    7FH          ;GET INDEX IN CONVERSION TABLE
        AND     SIG
        ADD     TEMP          ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    TEMP          ;READ IN CONVERTED POSITIVE SAMPLE
;
        LACK    80H
        AND     SIG
        BZ      CONT4         ;WAS IT A NEGATIVE SAMPLE ?
;
        LACK    0             ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     TEMP
        SACL    TEMP
;
CONT4:   LAC     PTR2          ;WRITE SAMPLE TO PROCESSING BUFFER
        TBLW    TEMP
        ADD     ONE           ;INCRIMENT POINTER
        SACL    PTR2
;
        BANZ    CLOOP         ;LOOP FOR ALL SAMPLES COLLECTED
;
        RET
;
        ELSE
;
; ***** CONVRT FOR EVM BOARD *****
;
CONVRT:  LACKA   AB4           ;ADDRESS OF INPUT BUFFER
        TBLR    DPTR
        LACKA   AB1           ;ADDRESS OF PROCESSING BUFFER
        TBLR    PTR2
        LARK    ARO, (NPTS1-1) ;NUMBER OF SAMPLES TO BE TRANSFERED
        LARP    OD
;
CLOOP:   LAC     DPTR          ;READ SAMPLE FROM INPUT BUFFER
        TBLR    SIG
        ADD     ONE           ;INCRIMENT POINTER
        SACL    DPTR
;
        LAC     PTR2          ;WRITE SAMPLE TO PROCESSING BUFFER
        TBLW    SIG
        ADD     ONE           ;INCRIMENT POINTER
        SACL    PTR2
;
        BANZ    CLOOP         ;LOOP FOR ALL SAMPLES COLLECTED
;
        RET
;
        ENDIF
;
; ----- ZERO CROSSINGS DETECTOR -----
;
ZCD:     LACKA   AB1           ;ADDRESS OF START OF INPUT BUFFER
        TBLR    DPTR
        LARK    ARO, NPTS1-2   ;LOOP COUNTER
        LARK    AR1, 0         ;INITIALIZE ZERO CROSSINGS COUNT
        LARP    0
        LAC     DPTR
        TBLR    LAST          ;PREVIOUS DATA POINT
;

```

```

ZLOOP:  ADD     ONE
        SACL    DPTR
        TBLR    TEMP           ;CURRENT DATA POINT
        LAC     TEMP
        XOR     LAST           ;CHECK FOR CHANGE OF SIGN BETWEEN PREVIOUS
        SACL    LAST           ;AND CURRENT DATA POINTS
        LAC     LAST
        BGEZ    NOZC          ;IF NOT NEGATIVE, THEN NO ZERO CROSSING
        LARP     1
        MAR     *+,ARO         ;INCREMENT ZERO CROSSINGS COUNT
NOZC:   LAC     TEMP
        SACL    LAST           ;PREVIOUS DATA POINT = CURRENT DATA POINT
        LAC     DPTR
        BANZ    ZLOOP          ;LOOP UNTIL DONE
;
        SAR     AR1,N2         ;STORE ZERO CROSSINGS COUNT
        RET
;
;----- GETDAT -----
;          FILTER, DOWNSAMPLE, DECIMATE BY THREE, WINDOW
;
GETDAT: LARP     0
        LACKA    AB1
        TBLR     DPTR          ;POINTER TO DATA IN
;
        LACKA    AB3
        TBLR     PTR2          ;POINTER TO DATA OUT (EVERY THIRD POINT)
;
        LACKA    AWIND1
        TBLR     WPTR          ;POINTER TO WINDOW
;
        LACKA    ATABLE
        TBLR     TEMP          ;ADDRESS OF COEFFICIENT TABLE
        LAC      TEMP          ;INTO ACC
        LARK     ARO,NA1       ;ADDRESS OF FILTER COEFS IN DRAM
;
        REPT     8D
        TBLR     *+,ARO
        ADD      ONE
        ENDM
;
        ZAC
        SACL     XLOW          ;ZERO RUNNING SUM
        SACL     XHIGH
;
        LARK     ARO,(W1-1)    ;FORWARD COUNT
        LACK     1
        SACL     WONE          ;FORWARD THRU WINDOW FUNCTION
;
;FILTER, DOWNSAMPLE, AND WINDOW
;
FLOOP:  CALL     FILTER        ;FILTER 1ST POINT
;
        LAC      WPTR          ;GET WINDOW SAMPLE
        TBLR     WIND
        ADD      WONE          ;WONE = 1 - FORWARD THRU WINDOW FUNCTION
        SACL     WPTR          ;WONE = -1 - BACKWARD THRU WINDOW FUNCTION
;
        LT       WIND
        MPY      SIG           ;SIG NOW WINDOWED

```

```

      PAC
      SACH      SIG,S1
;
      ZALS      XLOW           ;RUNNING SUM TO BE USED LATER
      ADDH      XHIGH
      ADD       SIG
      SACL      XLOW
      SACH      XHIGH
;
      LAC       PTR2           ;STORE IT - 1'ST, 4'TH ETC. FILTERED POINT
      TBLM      SIG
      ADD       ONE
      SACL      PTR2
;
      CALL      FILTER         ;FILTER 2ND POINT
;
      CALL      FILTER         ;FILTER 3RD POINT
;
      BANZ      FLOOP
;
      RET
;
;
;----- LOW PASS FILTER -----
;              5TH ORDER BI-QUAD FIR LOW PASS FILTER
;              WITH CORNER FREQUENCY AT 1.1 KHZ
;
FILTER: LAC      DPTR           ;GET NEXT SAMPLE
      TBLR      SIG
      ADD       ONE
      SACL      DPTR
;
      ZAC              ;1ST STAGE - ORDER 1
      LT          Y11
      MPY         NA1
      LTA         SIG
      MPY         NA2
      LTA         Y11
      SACH        Y1,S4
      DMOV        Y1           ;Y11 := Y1
;
      ZAC              ;2ND STAGE - ORDER 2
      MPY         NA5
      LTA         Y22
      MPY         NA4
      "D          Y21           ;Y22 := Y21
      MPY         NA3
      LTA         Y21
      SACH        Y2,S4
      DMOV        Y2           ;Y21 := Y2
;
      ZAC              ;3RD STAGE - ORDER 2
      MPY         NA8
      LTA         Y32
      MPY         NA7
      LTD         Y31           ;Y32 := Y31
      MPY         NA6
      APAC
      SACH        Y3,S4
      DMOV        Y3           ;Y31 := Y3
;

```

```

        SACH    SIG,84          ;FILTER OUTPUT
;
        RET
;
;----- CONVERT SIGNAL TO ZERO MEAN -----
;
2MEAN:  ZALS    XLOW
        ADDH    XHIGH
        CALL    DIVMEM          ;MEAN = (RUNNING SUM)/(NUMBER OF POINTS)
        SACL    TEMP
        ZAC
        SUB     TEMP
        SACL    TEMP          ;SAVE MEAN
;
        LARP    0D             ;USE PAGE 0
        LAC     DPTR
;
MLOOP:  TBLR    SIG
        LAC     SIG
        SUB     TEMP          ;SUBTRACT MEAN
        SACL    SIG
        LAC     DPTR          ;STORE SAMPLE IN SAME LOCATION
        TBLW    SIG
        ADD     ONE           ;UPDATE COUNTER
        SACL    DPTR
        BANZ    MLOOP
;
        RET
;
;----- PITCH'S AUTOCORRELATION -----
;
AUTO1:  SACL    TEMP          ;NUMBER OF LAGS
;
        LARK    AR1,(OUTB1-1) ;WILL READ 82 POINTS
;
        LACKA   AB3           ;POINTER TO DATA OUT
        TBLR    DPTR
        LACK    (OUTB1-1)
        ADD     DPTR
;
        LARP    1
RLOOP:  TBLR    *,AR1          ;READ 82 POINTS INTO MEMORY 0, 1, ... 81
        SUB     ONE
        BANZ    RLOOP
;
        LAC     ONE,514       ;INITIALIZE MASK
        SACL    ZMASK
        ZAC
        SACL    SCOUNT          ;INITIALIZE NORMALIZATION LOOP COUNT
;
; COMPUTE SUM OF SQUARES OF POINTS - WHICH GIVES R[0] AND ENERGY
;
        LARK    AR1,(OUTB1-1) ;82 POINTS
        ZAC          ;ACC := 0
        MPYK     0          ;P := 0
;
        LARP    1
SQU:    LTA     *,AR1          ;S(I)
        MPY     *,AR1          ;S(I)

```



```

      BANZ    SQU
      APAC

;
      SACL    XLOW          ;SAVE SUM OF SQUARES
      SACH    XHIGH
      SACL    ENRLO        ;ENERGY USED LATER BY DECIS
      SACH    ENRHI

;
; NORMALIZE R(0) TO CALCULATE SCOUNT FOR LATER USE IN NORMALIZATION
;
      BNZ     GT0           ;R(0) = 0?
      LACK    1
      SACL    XLOW
GT0:   LAC     XHIGH        ;ALREADY NORMALIZED?
      AND     ZMASK
      BNZ     NFIN        ;NORM FIN IF R(0) = 01XXXXX
;
NORM00: LAC     ONE         ;INCREMENT NORMALIZATION LOOP COUNT
      ADD     SCOUNT
      SACL    SCOUNT
;
      ZALS    XLOW
      ADDH    XHIGH
      ADDH    XHIGH        ;DOUBLE IT
      ADDS    XLOW
      SACL    XLOW        ;SAVE IT
      SACH    XHIGH
;
      LAC     XHIGH
      AND     ZMASK
      BZ      NORM00       ;KEEP ON GOING
;
; DETERMINE ZMASK TO ADJUST NORMALIZED RESULTS FOR DISPLAY
;
NFIN:  LAC     XHIGH
      SACL    DR
      LAC     ONE,S15
      SUB     ONE          ; 32767
      SACL    ZMASK
      LAC     ZMASK,S14
      CALL    DIVNEW
      SACL    ZMASK        ;ZMASK = [32767*2**14/R(0)]
      ZAC
      SUB     ZMASK
      SACL    ZMASK
;
      LT      XHIGH
      MPY     ZMASK
      PAC
      SACH    XHIGH,S1     ;R(0*)[ ]/2**15 = 32767/2 = 16383 = CONSTANT
;
      LAC     SPTR2
      TBLW    XHIGH        ;1'ST POINT OUT
      ADD     ONE
      SACL    SPTR2
;
; CALCULATE POWERS TO BE USED LATER FOR NORMALIZATION
;
      LAC     SCOUNT
      BZ      CONT5        ;SCOUNT = 0 -> NO NORMALIZATION REQUIRED
      SUB     ONE,S4       ;ACC = (SCOUNT-16)

```

```

        BGZ      PWR2      ;SCOUNT > 16 -> POWER=2**(SCOUNT-16)
        BZ       CONT5     ;SCOUNT = 16 -> NO POWER REQUIRED
                          ;SCOUNT < 16 -> POWER=2**(SCOUNT)
;
PWR1:   LAC      SCOUNT     ;I = SCOUNT
        B        PWR
;
PWR2:   LAC      SCOUNT
        SUB      ONE,S4     ;I = SCOUNT-16
;
; POWER = 2**I AND QTEMP = 2**I - 1
;
PWR:    SUB      ONE        ;I=I-1 FOR BANZ
        SACL     TEMPN
        LAR      ARI,TEMPN
        LARP     1
        LAC      ONE
        SACL     POWER      ;ACC := 1
PLOOP:  ADD      POWER      ;DOUBLE 2 I TIMES
        SACL     POWER
        BANZ     PLOOP
        SUB      ONE        ;ACC := 2**I - 1
        SACL     QTEMP      ;CREATE MASK (ALL 1'S)
;
; OUTER LOOP FOR NUMBER OF LAGS IN TEMP
;
CONT5:  LACK     (OUTB1-2)
        SACL     DELTA      ;1'ST IS DONE
;
        LACKA    ATANG      ;BASE ADDRESS FOR CALA
        TBLR     BASE
;
OUTER:  LARK     ARI,(OUTB1-1) ;INDEX THROUGH DATA POINTS
        LAR      ARO,DELTA   ;DELTA = 80 FIRST TIME AROUND
;
        MPYK     0           ;P := 0
        LARP     0           ;USE ARO FIRST
;
; SPECIAL FAST ALGORITHM FOR INNER LOOP SUMMATION.
; MUST DETERMINE WHERE TO ENTER IN-LINE CODE
;
        LAC      ONE        ;CALCULATE WHERE TO START LOOPING
        ADD      DELTA
        SACL     DP1         ;DP1 := DELTA + 1 = NUMBER OF LOOPS
        LAC      DP1,S13
        SACH     TEMPN      ;(DP1 MOD 8)
        LAC      TEMPN,S4   ;16 * (DP1 MOD 8)
        SUB      DP1,S1     ;OFFSET := -2[DP1-8*(DP1 MOD 8)]
                          ;:= -2[REM OF DP1/8]
;
        BNZ      REM        ;IF [REM = 0] THEN OFFSET = -16
        ZAC
        SUB      ONE,S4     ;OFFSET = -16
REM:    ADD      BASE        ;ADD OFFSET TO BASE ADDRESS (TARGET)
        SACL     TEMPN      ;YOU CAN'T ZERO ACCUM. SO SAVE
        CALA         ;JUMP TO INDEXED ADDRESS
;
; INNER MULT./ACC. LOOP - IN-LINE CODE USED FOR SPEED.
;
AUTLP:  REPT     7
        LTA      *-,ARI

```

```

        MPY      *-,ARO
        ENDM
        LTA      *,AR1          ;DON'T DECREMENT ARO
        MPY      *-,ARO
TARGET: BANEZ    AUTLP          ;BANEZ DOES FINAL DECREMENT ON ARO
        APAC          ;FINAL SUM
;
        SUB      TEMPN          ;ADJUST THE ACCUMULATOR BY PREVIOUS CONTENTS
        SACL     XLOW          ;STORE RESULTS
        SACH     XHIGH
        POP          ;CLEAN UP THE STACK FROM CALA
;
; NORMALIZE THE RESULTS (A SPECIAL FAST ALGORITHM)
;
        LAC      SCOUNT        ;GET NORMALIZATION COUNT
        BZ       NORM0          ;ALREADY NORMALIZED.
        SUB      ONE,S4         ;SHIFT = SCOUNT - 16
        BGZ      NORM2          ;SHIFT > 16
        BZ       NORM3          ;SHIFT = 16
        ADD      ONE
        BZ       NORM4          ;SHIFT = 15 (SPECIAL CASE)
        SUB      ONE
;
; CASE 1: 1 <= SHIFT COUNT <= 14
;
NORM1:  LT       POWER          ;GET 2**I IN T REG
        MPY      XLOW          ;LOW (SIGN EXTENSION!) * 2**I
        PAC
        SACH     TEMPN          ;/2**16
        ZALS     TEMPN          ;LOW * 2**(I-16)
        AND      QTEMP          ;ZAP SIGN BIT EXTENSION IN LOW WORD
        MPY      XHIGH          ; HIGH * 2**I
        *AC      ; + LOW * 2**(I-16)
        SACL     XHIGH
        B        NORM0
;
; CASE 2: SHIFT COUNT > 16 (COMPUTE XLOW * 2**(SCOUNT-16))
;
NORM2:  LT       POWER          ;GET 2**I IN T REG
        MPY      XLOW          ;XLOW * 2**(SCOUNT-16)
        PAC
        SACL     XHIGH
        B        NORM0
;
; CASE 3: SHIFT COUNT = 16 (TRANSFER LOW BYTE TO HIGH BYTE)
;
NORM3:  LAC      XLOW          ;GET LOW
        SACL     XHIGH          ;INTO HIGH
        B        NORM0
;
; CASE 4: SHIFT COUNT = 15 (SPECIAL CASE)
;
NORM4:  LAR      ARO,SCOUNT
        ZAC
NLP2:   ADDH     XHIGH
        ADDS     XLOW
        SACL     XLOW
        SACH     XHIGH
        BANEZ    NLP2
;
; CASE 0: ALREADY NORMALIZED (ADJUST RESULTS FOR DISPLAY)

```

```

;
NORM0: LT      XHIGH      ; R(I)/2 < R(I)*[ ]/2**15 < R(I)
      MPY      ZMASK      ; AS 2**14 < [ ] < 2**15-1
      PAC
      SACH     XHIGH,S1
;
      LAC      SPTR2
      TBLN     XHIGH      ;DUMP HIGH WORD OF NORM R(I) INTO PRAM
      ADD      ONE
      SACL     SPTR2
;
      LAC      DELTA      ;DECRIMENT LAG POINTER
      SUB      ONE
      SACL     DELTA
;
      LAC      TEMP      ;DECRIMENT LAG COUNT
      SUB      ONE
      SACL     TEMP
;
      BNZ      OUTER
      RET
;
;
;----- THE LG ALGORITHM -----
;
LG:    LARP     0          ;ARP = 0; DP = 0
;
      LAC      RX1        ;R(1) *2**15
      SACL     DR         ;DIVISOR = R(1) *2**15
;
      LAC      RX2,S15    ;DIVIDEND = R(2) *2**30
      CALL     DIVNEW     ; (R(2)/R(1)) *2**15
      SACL     FK1       ;FK(1) = (-R2/R1) *2**15
;
      LAC      RX2        ;R(2) *2**15
      SACL     XD1       ;D(1) = R(2) *2**15
;
      LAC      RX1        ;DIVISOR=R(1) *2**15
      SACL     DR
;
      LT       RX2        ;R(2) *2**15
      MPY      RX2        ; (R(2))**2 *2**30
      PAC      DIVIDEND = (R(2)**2 *2**30
      CALL     DIVNEW     ; ((R(2)**2)/R(1)) *2**15
      ADD      RX1        ; (R(1) - (R(2)**2/R(1))) *2**15
      SACL     XD2       ;D(2) = (R(1) - (R(2)**2/R(1))) *2**15
;
      LG1 MACRO      I,J
      LAC      RX&I,S15
      SACH     B1,X1
      MPYK     0
      LARK     AR0,FK1
      LARK     AR1,XD1
      CALL     LG&J&S
      ENDM
;
      J=1
      I=3
      REPT     M1-1
      LG1      %I,%J
      I=I+1

```

```

        J=J+1
        ENDM
;
        LAC      LGFLAG      ;WHERE TO RETURN? - STACK TOO DEEP FOR RET
        B2      LGRET1      ;      ORDER OF 4 FOR PITCH
;
        REPT     (M2-1)-(M1-1)
        LG1      %I,%J
        I=I+1
        J=J+1
        ENDM
;
        B      LGRET2      ;      ORDER OF 10 FOR ENVELOPE
;
        LG2 MACRO J
        MAR      *+,AR1
        MAR      *+,ARO
LG6J&S:
        LTA      *+,ARO
        MAR      *+,AR1
        SACH     Y,X1
        LAC      *+,AR1,S15
        MPY      Y
        APAC
        SACH     *-,AR1,X1
        LAC      Y,S15
        MPY      *-,AR1
        DMOV     *+,AR1
        ENDM
;
        J=M2-1
        REPT     M2-1
        LG2      %J
        J=J-1
        ENDM
;
LG0S:   APAC
        SACH     ',X1
        MAR      *+,AR1
        MAR      *+,AR1
        LAC      *,ARO
        SACL     DR          ;DIVISOR = D(I) *2**15
        LAC      Y,S15       ;DIVIDEND= Y *2**30
        CALL     DIVNEW      ;(Y/D(I)) *2**15
        SACL     *,AR1       ;FK(I) = -(Y/D(I)) *2**15
        LAC      *-,ARO,S15  ;D(I) *2**30
        LT       *,AR1       ;Y *2**15
        MPY      Y          ;(Y * FK(I)) *2**30
        APAC          ;(D(I) + Y * FK(I)) *2**30
        DMOV     *+,AR1
        MAR      *+,AR1
        MAR      *+,AR1
        SACH     *,AR(X1    ;D(I+1) = (D(I) + Y * FK(I)) *2**15
        RET
;
;
;----- TRANSFORM K'S TO A'S -----
;
;W A R N I N G : NA'S MUST BE IN CONTIGUOUS ARRAY
;
        STIB MACRO I,IP,IQ,MN2

```

```

LARK   ARO,NA6IQ      ;ADDR OF NA(IQ)
LARK   AR1,NA6IP      ;ADDR OF NA(IP)
LAC    FK&I          ;PASS FK'I IN TEMP
SACL   TEMP
LACK   MM2-1         ;NUMBER OF LOOPS INTO ACC FOR CALL
CALL   LPI           ;CALL CALC SUB
ENDM

;
ST2B MACRO   XIFLAG,I,P1,P2
;
IFE       XIFLAG      ;THIS DONE ONLY FOR EVEN VALUES OF I
LAC       NA6P1,S15   ;NA6P1*2**15
LT        NA6P1
MPY       FK&I        ;NA6P1*(FK&I*2**15)
APAC
SACH      NA6P1,X1    ;NA6P1*(1+FK&I)*2**15
ENDIF
;
LAC       FK&I,S13    ;[FK&I*2**15]*2**13
SACH      NA6P2       ;[FK&I*2**28]/2**16-FK&I*2**12
ENDM
;
LPI:     SACL   COUNT      ;SAVE COUNT
LARP     1              ;USE AR1 FIRST
;
LPT:     LAC    *,ARO,S15   ;NA6IP*2**15
LT        TEMP           ;FK&I*2**15
MPY       *,AR1          ;P := FK&I*NA6IQ*2**15
LTA       *,AR1          ;T := NA6IP,
;ACC := [NA6IP+FK&I*NA6IQ]*2**15
SACH      *+,ARO,X1       ;NA6IP := [NA6IP+FK&I*NA6IQ]*2**15
;
LAC       *,ARO,S15      ;NA6IQ*2**15
;NA6IP STILL IN T REG
MPY       TEMP           ;P := NA6IP*FK&I*2**15
APAC
SACH      *-,AR1,X1      ;ACC := [NA6IQ+FK&I*NA6IP]*2**15
;NA6IQ := [NA6IQ+FK&I*NA6IQ]*2**15 (OLD)
;
LAC       COUNT
SUB       ONE            ;DECREMENT LOOP COUNT
SACL     COUNT
;
BG2      LPT
RET
;
;-----
;
KTOA:    LAC      ONE,S12
SACL     NA1          ;NA1 := 2**12 = 1 IN OUR CONVENTION
;
LT       FK1
MPY      FK2          ;FK1*FK2*2**30
PAC
ADD      FK1,S15      ;[FK1+FK1*FK2*2**15]*2**15
SACH     TEMP
LAC      TEMP,S14
SACH     NA2          ;NA2 :=[FK1+FK1*FK2]*2**12
;
LAC      FK2,S13
SACH     NA3          ;NA3 := FK2*2**12
;

```

```

      I=3
      REPT (M1-2)          ;TRANSFORM FIRST PK'S FOR PITCH AND ENV.
      MN2=(I-1)/2+1
      IQ=I
      IP=2
      ST1B  %I,%II,%IQ,%MN2
      XIFLAG=2*(I/2)-I
      P1=MN2+1
      P2=I+1
;
      ST2B %XIFLAG,%I,%P1,%P2
;
      I=I+1
      ENDM
;
      LAC    LGFLAG          ;IF FLAG=1 THEN ENV
      BZ     RETURN          ;IF FLAG=0 THEN PITCH
;
CONTINU:
      REPT (M2-2)-(M1-2)    ;TRANSFORM THE REMAINING K'S FOR ENV ONLY
      MN2=(I-1)/2+1
      IQ=I
      IP=
      ST  %I,%IP,%IQ,%MN2
      XIFLAG=2*(I/2)-I
      P1=MN2+1
      P2=I+1
;
      ST2B %XIFLAG,%I,%P1,%P2
;
      I=I+1
      ENDM
;
RETURN: RET
;
;
;----- INVERSE FILTER -----
;          SIG NA6-NA9 + DUMMY IN 22: COEFFS NA2-NA5
;
INVFIL: LACKA  AB3
        TBLR   DPTR          ;SA OF ARRAY
;
        LACK    OUTB1
        SACL    COUNT        ;NUMBER OF POINTS
;
        LARK    ARO,NA9      ;END OF DELAY LINE
        LARP    0
;
        ZAC
        SACL    *-,ARO      ;ZERO THE DELAY LINE
        SACL    *-,ARO
        SACL    *-,ARO
        SACL    *-,ARO
;
FLP:    LARK    ARO,NA9      ;ARO POINT TO END OF S ARRAY
        LARK    AR1,NA5     ;AR1 POINT TO END OF COEFF ARRAY
;
        LAC     DPTR          ;GET INPUT SAMPLE
        TBLR    SIG
        LAC     SIG,S12      ;IX(I)
        MPYK    0            ;P := 0

```

```

LTD    *-,AR1      ;LTD (AR0)- /LOAD S, ACC, SHIFT, AR1 NEXT
MPY    *-,AR0      ;MPY (AR1)- /MULT X COEFF, AR0 NEXT
LTD    *-,AR1      ;LTD (AR0)- /LOAD S, ACC, SHIFT, AR1 NEXT
MPY    *-,AR0      ;MPY (AR1)- /MULT X COEFF, AR0 NEXT
LTD    *-,AR1      ;LTD (AR0)- /LOAD S, ACC, SHIFT, AR1 NEXT
MPY    *-,AR0      ;MPY (AR1)- /MULT X COEFF, AR0 NEXT
;
LTD    *,AR1       ;LTD (AR0) /LOAD S, ACC, SHIFT, AR1 NEXT
MPY    *,AR0       ;MPY (AR1) /MULT X COEFF, AR0 NEXT
AFAC
SACH   TEMP,S4     ;FINAL ADD P TO ACC
;                                     ;OUTPUT OF CONVOLUTION
LAC    SIG         ;CURRENT IX
SACL   *,AR0       ;INTO DELAY LINE
;
LAC    DPTR        ;OUTPUT TO PRAM
TBLW   TEMP
ADD    ONE
SACL   DPTR
;
LAC    COUNT       ;DECRIMENT COUNTER
SUB    ONE
SACL   COUNT
BNZ    FLP
;
RET
;
;
;----- DECIS -----
;
DECIS: LACKA      ADECT
      TBLR      N1      ;GET DECISION PARAMETERS FROM LAST FRAME
      ADD       ONE
      TBLR      N2
      ADD       ONE
      TBLR      N3
      ADD       ONE
      TBLR      NMIN
      ADD       ONE
      TBLR      NMAX
;
      LACK      IR1     ;SIGNAL RMS THRESHOLD
      SUB       IR0     ;SIGNAL RMS
      BGEZ      DE253   ;UNVOICED IF RMS THRESHOLD >= SIGNAL RMS
;
      LACK      IZM     ;ZERO CROSSINGS THRESHOLD
      SUB       NZ      ;ZERO CROSSINGS COUNT
      BLZ       DE253   ;UNVOICED IF ZC THRESHOLD < ZC COUNT
;
      CALL      PPIC    ;FIND PEAK IN RANGE OF NMIN TO NMAX
      SACL      L       ;SAVE PEAK LOCATION
;
; TEST FOR POSSIBLE DOUBLING IN SEARCH RANGE (NMIN, NMAX-L-L/4)
;
      LAC       L,S14
      SACH      TEMP    ;TEMP = L/4
      LAC       L
      SUB       TEMP
      SACL      NMAX    ;NMAX = L - L/4
      SUB       NMIN
      BLZ      DE226    ;SKIP TEST IF NMAX < NMIN

```



```

CALL    PFIC      ;FIND PEAK IN RANGE OF NMIN TO NMAX
SACL    LL        ;SAVE PEAK LOCATION

;
LACKA   ASCP1
TBLR    DPTR      ;SCOPE BUFFER = IF() ARRAY
LAC      DPTR
ADD      L
TBLR    TEMP      ;IF(L)
LT       TEMP
LAC      DPTR
ADD      LL
TBLR    TEMP      ;IF(LL)
LAC      TEMP,S12
MPYK    -3481D     ;-0.85
APAC
BLEZ     DE226     ;IF (IF(LL) <= 0.85*IF(L)) THEN NO DOUBLING
LAC      LL
SACL    L         ;ELSE TAKE PEAK AT LL

;
; PARABOLIC INTERPOLATION TO FIND HEIGHT AND POSITION OF 'TRUE' PEAK
;
DE226:  LACKA   ASCP1
TBLR    DPTR      ;ADDRESS OF START OF SCOPE BUFFER
LAC      DPTR
ADD      L
TBLR    SL        ;IF(L)
SUB      ONE
TBLR    SLM1      ;IF(L-1)
ADD      ONE,S1
TBLR    SLP1      ;IF(L+1)

;
LAC      SL
SUB      SLM1
BLEZ     DE253     ;UNVOICED IF IF(L) <= IF(L-1)
LAC      SL
SUB      SLP1
BLEZ     DE253     ;UNVOICED IF IF(L) <= IF(L+1)

;
LAC      SL,S15    ;-AA = (2*IF(L) - IF(L-1) -
SUB      SLM1,S14   ;      IF(L+1))/2
SUB      SLP1,S14
SACH     DR,S1      ;-AA > 0 ALWAYS

;
LAC      SLP1,S14
SUB      SLM1,S14
SACH     BB         ;BB = (IF(L+1) - IF(L-1))/4
LT       BB
MPY      BB
PAC      ;BB*BB
CALL     DIVNEW
SACL     TEMP      ;BB*BL 'AA = -(BB*BB/(-AA))
LAC      SL
SUB      TEMP
SACL     AP        ;AP = IF(L) - BB*BB/AA

;
LAC      BB,S7     ;BB*(2**7)
CALL     DIVNEW
SACL     TEMP      ;BB*(2**7)/AA = -(BB*(2**7)/(-AA))
LAC      L,S7      ;L*(2**7)
SUB      TEMP
SACL     ALI       ;ALI = (L - BB/AA)*(2**7)

```

```

;
; DECISIONS
;
      LAC      DPTR
      TBLR     TEMP          ;IF (0)
      LAC      TEMP
      LT       L
      MPYK     -300
      APAC
      SACL     TEMP          ;IF (0) - 300*L
      LT       TEMP
      LAC      AP,S12
      MPYK     -1638         ; -0.4
      APAC
      BGEZ     DE820         ;PEAK OK IF AP >= 0.4*(IF(0) - 300*L)
DE251: LAC      N1
      BZ       DE253         ;IF N1 = 0, THEN UNVOICED
DE252: LAC      AP,S12
      MPYK     -1229         ; -0.3
      APAC
      BGEZ     DE820         ;PEAK OK IF AP >= 0.3*(IF(0) - 300*L)
;
DE253: ZAC
      SACL     N0            ;MARK AS UNVOICED
      B        DE250
;
DE820: LAC      ONE,S6       ;0.5*(2**7) = 2**6
      LT       ALI          ;(L - BB/AA)*(2**7)
      MPYK     3
      APAC     ;((L - BB/AA)*3 + 0.5)*(2**7)
      SACL     TEMP
      LAC      TEMP,S9      ;((L - BB/AA)*3 + 0.5)*(2**16)
      SACH     N0           ;N0 = TRUNC((L - BB/AA)*3 + 0.5)
;
DE250: LAC      N1,S12
      SUB      N3,S12
      ABS
      LT       N3
      MPYK     -1638         ; -0.4
      APAC
      BGEZ     DE999         ;ABS(N1 - N3) - 0.4*N3 > 0
DE994: LAC      N1,S15
      ADD      N3,S15
      SACH     N2            ;N2 = (N1 + N3)/2
;
; IF N3 = 0, N2 <> 0, & N0 AND N1 ARE CLOSE, EXTRAPOLATE FOR FIRST VOICED
; TERM USING LINEAR EXTRAPOLATION
;
DE999: LAC      N3
      BNZ     DE891          ;IF (N3 <> 0) THEN NO EXTRAPOLATION
      LAC      N2
      BZ       DE891         ;IF N2 = 0 THEN NO EXTRAPOLATION
      LAC      N0,S12
      SUB      N1,S12
      ABS
      LT       N1
      MPYK     -819          ; -0.2
      APAC
      BGEZ     DE891         ;IF ABS(N0-N1) > 0.2*N1 THEN NO EXTRAPOLATION
      LAC      N1,S1
      SUB      N0
      ; ELSE N2 = 2*N1-N0 (LINEAR EXTRAPOLATION)

```

```

                SACL    N2
;
; UPDATE FRAMES
;
DE891:  DMOV    N2            ;N3 = N2
        DMOV    N1            ;N2 = N1
        DMOV    N0            ;N1 = N0
;
; DETERMINE SEARCH RANGE FOR NEXT FRAME:  NMIN = L - 7 < X < NMAX = L + 7
; IF (L - 7) < JMI THEN NMIN = JMI      IF (L + 7) > JMA THEN NMAX = JMA
;
        LACK    JMI
        SACL    NMIN            ;NMIN = JMI
        LACK    JMA
        SACL    NMAX            ;NMAX = JMA
        LAC     N0
        BZ      DE888            ;IF PRESENT FRAME UNVOICED, THEN LIMITS SET
;
        LT      ONE
        LAC     L
        MPYK    -7
        APAC
        SACL    NMIN            ;NMIN = L - 7
        MPYK    -JMI
        APAC
        BGEZ    DE886            ;IF NMIN >= JMI THEN OK
        LACK    JMI
        SACL    NMIN            ;OTHERWISE NMIN = JMI
;
DE886:  LACK    7
        ADD     L
        SACL    NMAX            ;NMAX = L + 7
        MPYK    -JMA
        APAC
        BLEZ    DE888            ;IF NMAX <= JMA THEN OK
        LACK    JMA
        SACL    NMAX            ;OTHERWISE NMAX = JMA
;
DE888:  LAC     N3                ;CURRENT PITCH, DETERMINED LAST FRAME
        BNZ     VOICED            ;IF (N3 <> 0) THEN VOICED
        LACK    0                ; ELSE PITCH = 0 FOR ISR
        B       STRPTC            ;STORE PITCH IN PRAM
;
VOICED: LACK    3
        SACL    DR                ;FOR DISPLAY, DIVIDE PITCH BY 3
        LAC     N3
        CALL    DIVNEW
        SACL    TEMP            ;TEMP = -(N3/3)
        LACKA   ASCP1            ;GET ADDRESS OF TOP OF SCOPE BUFFER
        TBLR    DPTR
        LAC     DPTR
        SUB     TEMP            ;PITCH = (TOP OF SCOPE BUFFER) + N3/3
;
STRPTC: SACL    TEMP            ;STORE PITCH IN PRAM - FFT WIPES OUT DRAM
        LACKA   APITCH
        TBLW    TEMP
;
        LACKA   ADECT            ;STORE DECISION PARAMS IN PRAM FOR NEXT FRAME
        TBLW    N1
        ADD     ONE
        TBLW    N2

```

```

        ADD     ONE
        TBLW    N3
        ADD     ONE
        TBLW    NMIN
        ADD     ONE
        TBLW    NMAX
;
        RET
;
;
;----- PEAK PICKER FOR PITCH -----
;
PPIC:   LARP     0
        LACKA    ASCP1
        TBLR     IX           ;ADDRESS OF START OF SCOPE BUFFER
        LAC      IX
        ADD      NMIN
        SACL     DPTR         ;ADDRESS OF FIRST POINT TO SEARCH
        LAC      NMAX
        SUB      NMIN
        SACL     TEMP
        LAR      AR0,TEMP     ;LOOP COUNTER
        LAC      ONE,S15
        SACL     MAXVAL       ;INITIALIZE MAXVAL TO -32768
;
FPEAK:  LAC      DPTR
        TBLR     TEMP         ;CURRENT DATA POINT
        ADD      ONE
        SACL     DPTR
        LAC      MAXVAL
        SUB      TEMP
        BGEZ     LOWER       ;IF (MAXVAL - TEMP) >= 0, THEN LOWER
HIGHER: LAC      TEMP
        SACL     MAXVAL       ;MAXVAL = CURRENT DATA POINT
        LAC      DPTR
        SACL     PTR2         ;ADDRESS OF PEAK
LOWER:  BANZ     FPEAK        ;LOOP UNTIL DONE
;
        LAC      PTR2         ;ADDRESS OF PEAK
        SUB      IX           ;PUT INTO RANGE 1 ... (OUTB1)
        SUB      ONE          ;PUT INTO RANGE 0 ... (OUTB1) - 1
        RET
;
;
;----- ENV'S DIFF. WINDOW ALG. -----
;
WINDAT: LACKA    AB2
        TBLR     DPTR         ;POINTER TO DATA IN
;
        LACKA    AWIND2
        TBLR     WPTR         ;POINTER TO WINDOW
;
        ZAC
        SACL     LAST         ;PREVIOUS POINT INITIALIZED
        SACL     XLOW         ;ZERO RUNNING SUM
        SACL     XHIGH
;
        LARK     AR0,(W2-1)   ;FORWARD COUNT
        LACK     1
        SACL     WONE        ;FORWARD THRU WINDOW FUNCTION

```

3 OF/DE 3

MICRO

```

        LARP      0
;
MIDDLE: LAC      DPTR
;
;
WLOOP: TBLR      SIG          ;GET INPUT - CURRENT POINT
;
        IF CRC          ;CRC BOARD ALREADY HAS DIFFERENTIAL INPUT
        ELSE
            ZALH      SIG          ;DIFFERENTIALLY WINDOW DATA FOR EVM
            SACH      TEMP
            SUBH      LAST          ;SIG-LAST= DIFF SIG
            ADD      LAST,S13       ;SIG-LAST+LAST/8 = SIG-0.875*LAST
            SACH      SIG
            LAC      TEMP
            SACL      LAST
        ENDIF
;
        LAC      WPTR          ;GET WINDOW SAMPLE
        TBLR      WIND
        ADD      WONE          ;WONE = 1 - FORWARD THRU WINDOW FUNCTION
        SACL      WPTR         ;WONE = -1 - BACKWARD THRU WINDOW FUNCTION
;
        LT      WIND
        MPY      SIG          ;SIG NOW WINDOWED
        PAC
;
        IF CRC
            SACH      SIG          ;FOR CRC DON'T SHIFT RESULT FOR DISPLAY
        ELSE
            SACH      SIG,S1       ;FOR EVM SHIFT RESULT FOR DISPLAY
        ENDIF
;
        ZALS      XLOW          ;CALCULATE RUNNING SUM TO BE USED IN ZMEAN
        ADDH      XHIGH
        ADD      SIG
        SACL      XLOW
        SACH      XHIGH
;
        LAC      DPTR          ;STORE IT
        TBLW      SIG
        ADD      ONE
        SACL      DPTR
;
        BANA      WLOOP
;
        RET
;
;
;===== ENV'S AUTOCORRELATION =====
;
AUTO2:  LACK      NR
        SACL      NRD          ;NUMBER OF POINTS IN EACH BUFFER (25)
;
        LACK      IT
        SACL      NRCNT        ;NUMBER OF BUFFERS TO DO = (N / NR) 8
;
;R(DELTA) := 0, DELTA = 0, ... ,M2
;
M2P1=(2*M2)+1
        ZAC          ;ZERO THE ACCUMULATOR
        LARK      ARO,LOW0

```

```

        LARK      AR1,M2P1
;
AUTOL:  LARP      0                ;ZERO R(DELTA)'S
        SACL      '+,AR1
        BANZ      AUTOL
;
;POINT TO BUFFERSTART
;
        LACKA     AB2
        TBLR      PTR2            ;GET ADDRESS OF BUFF2
;
        LACK      M2-1            ; M2-1
        ADD       PTR2            ; + ADDRESS OF START OF BUFFER IN TABLE MEMORY
        ADD       NRD             ; + NUMBER OF POINTS IN BUFFER
        SACL      DPTR            ; = POINTER TO END OF FIRST BUFFER IN PRAM
;
;AUTOCORRELATION ALGORITHM
;
;FOR I = 1 TO IT (WHERE IT = N/NR, AND NR = NUMBER OF POINTS PER ITERATION)
;  FOR DELTA = 0 TO M2 (WHERE M2 = NUMBER OF LAGS)
;    R(DELTA) = R(DELTA) + SUM(I), WHERE SUM(I) = SUM[X(N)*X(N+DELTA)]
;    NEXT DELTA
;  NEXT I
;
;FOR I = 1 TO IT, ** READ IN A BUFFERFUL OF POINTS **
;
AGAIN:  LARK      ARO,3END          ;ARO POINT TO NRD'TH (60'TH) POINT IN DATA RAM
        LARP      0                ;USE ARO
        LAC       DPTR            ;END OF DATA IN TABLE MEMORY
;
;NOTE: REDUCE AVERAGE READ TIME TO 4.4 FROM 6 CYCLES VIA INLINE CODE
;
READL:
        REPT      4D
        TBLR      *-,ARO          ;READ NR POINTS IN BACKWARDS
        SUB       ONE             ;STEP ACC BACK
        ENDM
;
        TBLR      *,ARO          ;READ NR POINTS IN BACKWARDS
        SUB       ONE             ;STEP ACC BACK
        BANZ      READL
;
;SETUP FOR INNER LOOP
;
        ZAC
        SACL      DELTA            ;DELTA := 0
        LACK      M2+1
        SACL      LAGCNT          ;LAGCOUNT := M2+1
;
        LARK      ARO,LOW0        ;POINT TO R(0)
;
;DO ALL LAGS (0 TO M2) FOR THIS BUFFER, THEN DO NEXT BUFFER.
;
LAG:    LARP      0                ;USE ARO FIRST
;
        ZALS      *+,ARO          ;LOAD LOW R(DELTA)
        ADDH      *-,ARO          ;LOAD HIGH R(DELTA)
        SAR       ARO,TEMP        ;SAVE ARO
;
;R(DELTA) := R(DELTA) + SUM[X(N)*X(N+DELTA)]
;

```

```

SUM:  LARK  ARO,0      ;STARTS AT ZERO
      LAR   ARI,DELTA  ;STARTS AT DELTA, DELTA = 0, ... ,M2
;
      LT    *+,ARI
      MPY   *+,ARO
      REPT  NR-1
      LTA   *+,ARI
      MPY   *+,ARO
      ENDM
      APAC                      ;FINAL ADD
;
      LAR   ARO,TEMP      ;RESTORE ARO
      SACL  *+,ARO        ;SAVE LOW R(DELTA)
      SACH  *+,ARO        ;SAVE HIGH R(DELTA) AND ON TO NEXT
;
      LAC   DELTA
      ADD   ONE           ;DELTA := DELTA + 1
      SACL  DELTA
;
      LAC   LAGCNT
      SUB   ONE           ;DECREMENT LAG COUNT
      SACL  LAGCNT
      BGZ   LAG           ;ONTO NEXT LAG FOR THIS BUFFERFUL
;
      LAC   DPTR
      ADD   NRD
      SACL  DPTR
;
      LAC   NRCNT
      SUB   ONE           ;DECREMENT BUFFER COUNT
      SACL  NRCNT
      BGZ   AGAIN        ;OUTER LOOP, PER BUFFER
      RET
;
;
;===== R(I)'S COMPUTED. NOW HAVE TO NORMALIZE THE THEM =====
;
NORMAL: LAC   ONE,S14      ;ACC := 1
      SACL  7MASK          ;MASK := 010000..00
      ZAC                      ;ACC := 0
      SACL  SCOUNT        ;SCOUNT := 0
      LARP  0             ;USE ARO FIRST
;
      MAR   *+,ARO        ;SKIP LOW
      LAC   *-,ARO        ;GET HIGH, POINTERS BACK
      AND   7MASK
      BZ    NORMM
      RET                  ;NO NORM NEEDED
;
;DOUBLE R(0) UNTIL ACC = 01XXXXXXX.XXXXX (32 BITS)
;
NORMM:  LACK  1           ;| INCREMENT SHIFT COUNTER
      ADD   SCOUNT        ;| SCOUNT := SCOUNT + 1
      SACL  SCOUNT        ;| AND STORE IT.
      ZALS  *+,ARO        ;ACC := R(0)
      ADDH  *-,ARO
      ADDS  *+,ARO        ;ACC := ACC + R(0) = 2*R(0)
      ADDH  *-,ARO
      SACL  *+,ARO        ;R(0) := 2*R(0)
      SACH  *-,ARO
;

```



```

        LAC      *-,AR0      ;LOAD HIGH R(0) TO TEST, POINT TO LOW0
        AND      ZMASK      ;IS BIT 15 1?
        BZ       NORMM      ;NO. DOUBLE R(0) AGAIN
;
;NOW DO OTHERS R(I), I = 1,...,M2
;
        MAR      *+,AR0      ;SKIP OVER R(0) TO POINT TO R(1)
        MAR      *+,AR0
;
;NORMALIZATION
;
NL:      LARP     0           ;USE AR0 FIRST
NORMS:   LAC      SCOUNT    ;GET SHIFT COUNT
        SACL     COUNT      ;COUNT := SCOUNT
NORM:    ZALS     *+,AR0      ;LOAD R(I) LOW
        ADDH     *-,AR0      ; AND R(I) HIGH
        ADDS     *+,AR0      ;ADD R(I) LOW
        ADDH     *-,AR0      ; ADD R(I) HIGH [DOUBLE R(I)!]
        SACL     *+,AR0      ;STORE R(I) LOW
        SACH     *-,AR0      ; AND R(I) HIGH
;
        LAC      COUNT      ;|
        SUB      ONE        ;COUNT := COUNT - 1
        SACL     COUNT      ;|
        BNZ      NORM       ;UNTIL COUNT = 0
;
        MAR      *+,AR0      ;SKIP OVER R(I) PAIR JUST DONE
        MAR      *+,AR1
;
        BANZ     NL         ;AND DO NEXT R(I) UNTIL M2 DONE
        RET
;
;
;----- PREFFT -----
;
;NOW: NA1 IN 0, NA2 IN 1, ... ,NA11 IN 10
;DOUBLE NA'S UNTIL 16K <LARGEST < 32767
;MOVE NA1-NA11 TO NA12-NA22 AND FILL REST OF DRAM WITH 0'S
;MOVE DATA FROM PAGE 0 TO BUFF2 (PRAM) FOR THE FFT
;
PREFFT:  LARK     AR0,NA11
        LARK     AR1,NA22
        LARP     0
SW1:     LAC      *,AR1      ;NA11,...,NA1
        SACL     *-,AR0      ;NA22,...,NA12
        BANZ     SW1
;
DSTART:  LARK     AR0,NA22    ;MOVE BACKWARDS THRU ARRAY
        LARK     AR1,M2      ;DO M2+1 TIMES
DOUBLE:  LARP     0
        LT       *,AR0      ;OLD NA22,...,NA12
        LAC      *,AR0,S1    ;2*OLD
        SACL     *,AR0      ;SAVE IT
        MPY      *-,AR1      ;OLD*NEW=OLD*(2*OLD)
        PAC
        BLZ      OVFL        ;--* OR +*- = -, ++* OR -*- = +
        BANZ     DOUBLE
;
;MOVE NA12-NA22 TO NA1-NA11 IF NO OVERFLOW DURING PASS
;
        LARK     AR0,NA11

```

```

      LARK    AR1,NA22
SW2:   LARP    1
      LAC     *~,ARO           ;NA22,...,NA12
      SACL    *,ARO           ;NA11,...,NA1
      BANZ    SW2             ;BANZ DEC ARO
      B       DSTART

;
;GOOD RESULTS IN NA1-NA11 AS NO MOV IF OVERFLOW DURING LAST PASS
;
OVFL:  LARK    ARO,(2*M2+1)    ;21 FOR M2=10
      LARK    AR1,M2          ;10 FOR M2=10
;
;LOAD NA'S INTO FIRST REAL PART OF FFT
;
NALP:  LARP    0              ;USE ARO FIRST
      ZAC
      SACL    *~,AR1          ; 0 INTO 21,19, 17, ..., 1
      LAC     *,ARO           ; (10) INTO 20, ..., (1) INTO 2, (0) INTO 0
      SACL    *~,AR1
      BANZ    NALP            ;AR1 DECREMENTED BY BANZ
;
;FILL REST OF MEMORY WITH ZEROES
;
      LARK    ARO,22D          ; 22, 21, 127
      LARK    AR1,105D         ;106 ZEROS
      ZAC
;
ZACLP: LARP    0
      SACL    *+,AR1
      BANZ    ZACLP
;
;MUST MOVE DATA FROM PAGE ZERO TO BUFF2 FOR FFT
;
MPG00T: LDPK    1
      LARK    AR1,127D
      LACKA    AB2
      TBLR     INUMBR
      LACK     127D
      ADD      INUMBR
;
      LARP    1
MOVITQ: TBLW    *,AR1
      SUB      IONE
      BANZ     MOVITQ
;
      RET
;
;
;----- MAGNTD -----
;
MAGNTD: LDPK    0D              ;PAGE 0
      LACK     1D              ;RETURN CONSTANT CANCELLED BY FFT
      SACL     ONE
;
      LARK     ARO,0
      LARK     AR1,(NSPECT-1)
      LACKA    ASCP2
      TBLR     DPTR            ;SECOND TIME USE SCOPE BUFFER
;
;MAGNITUDE OF SPECTRUM CALCULATION, RESULT STORED IN SCOPE BUFFER
;

```

```

STORE:  LARP      0
;
      LAC      *+,ARO      ;PICK UP REAL
      ABS
      SACL     RXT      ;REAL
;
      LAC      *+,AR1      ;PICK UP IMAGINARY
      ABS
      SACL     IYT      ;IMAG
;
      SUB      RXT
      BGZ      YGX
;
XGY:   LAC      RXT      ;REAL >= IMAGINARY
      SUB      IYT,S2
      BGZ      POS1
;
      ZALH     RXT
      SUB      RXT,S13
      ADD      IYT,S15
      B        FINMAG
;
POS1:  ZALH     RXT
      ADD      IYT,S13
      B        FINMAG
;
YGX:   LAC      IYT      ;IMAGINARY > REAL
      SUB      RXT,S2
      BGZ      POS2
;
      ZALH     IYT
      SUB      IYT,S13
      ADD      RXT,S15
      B        FINMAG
;
POS2:  ZALH     IYT
      ADD      RXT,S13      ;ACC. HAS THE RESULT
;
FINMAG: SACH     RXT,S4      ;JUSTIFY IT FOR DISPLAY - SHIFT LEFT 4
      LAC      RXT,S14      ;SHIFT RIGHT 2 TOTAL
      SACH     RXT      ;TOTAL = SHIFT LEFT 2
      LAC      DPTR
      TBLW     RXT
      ADD      ONE
      TBLW     RXT      ;XT (MAG) OUT TWICE
      ADD      ONE
      SACL     DPTR
;
      BANZ     STORE      ;AND AGAIN
;
      RET
;
;----- NULL PICKER FOR FORMANTS -----
;
NPIC:  LACK      3D      ;NUMBER OF FORMANTS TO PICK
      SACL     COUNT
;
      LACKA     AFMT1      ;ZERO PREVIOUS FORMANT NULLS
      TBLW     ONE
      ADD      ONE
      TBLW     ONE

```

```

      ADD     ONE
      TBLW    ONE
;
      LARK    ARI, (NSPECT-5) ;26 BUFFER SHIFTS MAX
      LARK    ARO, BSTRT1      ;SET CHECK BUFFER POINTER=3
      LACKA   ASCP2            ;GET POINTER TO SCOPE BUFFER
      LARP     0
      TBLR    DPTR
      LAC     DPTR              ;READ POINTER FROM PRAM
      ADD     ONE, S1          ;MOVE POINTER PAST SYNC PULSE
;                               ;NOW READ IN FIRST THREE POINTS
      TBLR    *
      SACL    DPTR
      LAC     *+
      SACL    MAXVAL
      LAC     DPTR
      ADD     ONE, S1          ;ADVANCE ADDRESS BY TWO
      TBLR    *+              ;READ ANOTHER POINT IN
      ADD     ONE, S1          ;
      TBLR    *                ;LEAVE ARO -> END OF BUFFER
      ADD     ONE, S1          ;
      SACL    DPTR            ;SAVE SCOPE BUFFER POINTER
      B       SPICK           ;SKIP BUFFER SHIFT ON 1ST PASS
;
;SHIFT BUFFER HERE
;
CHKMIN: LARP     0              ;DO BUFFER SHIFT HERE
      LARK    ARO, BSTRT2      ;ARO-> MIDDLE POINT
      LAC     *-              ;SHIFT MIDDLE BACK ONE
      SACL    *+              ;POSITION
      MAR     *+              ;ADVANCE ARO TO END OF
      LAC     *-              ;BUFFER AND MOVE LAST
      SACL    *+              ;POINT TO MIDDLE OF BUFFER
      LAC     DPTR            ;NOW READ IN A NEW POINT
      TBLR    *                ;TO LAST POSITION IN BUFFER
      ADD     ONE, S1          ;ADVANCE SCOPE BUFFER POINTER
      SACL    DPTR            ;AND SAVE IT
;
      LAC     MAXVAL           ;CHECK FOR MAX POINT
      SUB     *
      BGEZ    SPICK           ;IF * > MAXVAL THEN
      LAC     *                ;MAXVAL = *
      SACL    MAXVAL
;
;CHECK FOR A NULL POINT
;
SPICK:  LAC     *-              ;GET LAST POINT
      SUB     *-              ;SUB LAST FROM MIDDLE
      BLEZ    ENDPK           ;IF C < B NO MIN FOUND
      LAC     *+              ;GET FIRST POINT
      SUB     *                ;SUB MIDDLE FROM FIRST
      BLEZ    ENDPK           ;IF A < B NO MIN FOUND
;
;CHECK FOR NULL CONDITION OF -2.5 dB FROM PREV MAXVAL
;
      LT      MAXVAL           ;
      MPYK    1535D           ;CALC 0.75 OF MAXVAL
      PAC     ;
      SACH    TEMP, X4         ;12 BIT MULT, SHIFT ACC <- X4
      LAC     TEMP, S1         ;SUB NULLVAL FROM 0.75*MAXVAL
      SUB     *                ;I.E. -2.5 dB FROM MAX VAL

```

```

        BLZ      BADPK      ;IF RESULT POS MIN IS OK:TAG IT
        LAC      DPTR      ;GET SCOPE BUFFER POINTER
        SUB      ONE,S2    ;MOVE IT BACK 4 PLACES
        SACL     TEMP      ;SAVE POINTER TO PEAK
;
        LACKA    AFMT4     ;LOAD END ADDRESS OF FORMANT STORAGE
        SUB      COUNT     ; AND INDEX TO PROPER FORMANT
        TBLW     TEMP      ;STORE ADDRESS OF FORMANT IN SCOPE BUFFER
;
        LAC      COUNT     ;INC PEAK COUNTER
        SUB      ONE
        BLEZ     DONEPK    ;IF FOUND THREE GOOD PEAKS
        SACL     COUNT
BADPK:   ZAC      ;BUG OUT, ELSE
        SACL     MAXVAL    ;RESET MAXVAL, BUT CHECK IF AT END OF BUFFER
;
;DONE ALL THE BUFFER SHIFTS ?
;
ENDPK:   LARP     1
        BANZ     CHKMIN    ;DEC ARI AND IF NOT 0 LOOP BACK
;
DONEPK:  RET
;
;----- THE ISR ROUTINE -----
;
        IF CRC
;
; ***** ISR FOR CRC BOARD *****
;
INPUT:   SST      ISTATE    ;STORE PRE-ISR STATE
        LDPK     1
        SACL     ILOW
        SACH     IHIGH
;
        LACKA    AFFTSTG    ;STORE FFT VARIABLES IN PRAM
        TBLW     ISMPL
        ADD      IONE
        TBLW     ITEMP
        ADD      IONE
        TBLW     ICOUNT
        ADD      IONE
        TBLW     IPTR
        ADD      IONE
        TBLW     PSCOPE
        ADD      IONE
        TBLW     CSCOPE
        ADD      IONE
        TBLW     IFLAG
;
        LACKA    ACOUNT    ;RETURN ISR VARIABLES
        TBLR     ICOUNT
        ADD      IONE
        TBLR     IPTR
        ADD      IONE
        TBLR     PSCOPE
        ADD      IONE
        TBLR     CSCOPE
        ADD      IONE
        TBLR     IFLAG
;

```

```

        LAC      ICOUNT      ;IF 0 THEN INPUT BUFFER FULL OR BUFFER SWAP
        BZ       SOUT        ; SO -- DON'T -- COLLECT DATA
;
        IN       ISMPL,PA3    ;GET MU-LAW SAMPLE
;
; STORE 2 SAMPLES PER WORD, FIRST IN HIGH BYTE, SECOND IN LOW BYTE
;
        LAC      IONE
        XOR      IFLAG      ;TOGGLE FLAG
        SACL     IFLAG
        BZ       ILWBYT     ;IF ZERO THEN INSERT LOW BYTE
;
INGHBT: LAC      IPTR        ;WRITE HIGH BYTE ONLY TO INPUT BUFFER AND
        TBLW     ISMPL      ; DON'T INCRIMENT INPUT POINTER
        B        CONT
;
ILWBYT: LAC      IPTR        ;RETRIEVE HIGH BYTE
        TBLR     ITEMP
        LACK     OFFH      ;MASK OFF GARBAGE FROM LOW BYTE
        AND      ISMPL
        ADD      ITEMP,S8   ;SHIFT HIGH BYTE OVER AND ADD TO LOW BYTE
        SACL     ISMPL
        LAC      IPTR      ;WRITE BOTH BYTES TO INPUT BUFFER
        TBLW     ISMPL
        ADD      IONE      ;INCRIMENT INPUT POINTER
        SACL     IPTR
;
CONT:   LAC      ICOUNT      ;DECRIMENT INPUT COUNTER FOR EACH SAMPLE
        SUB      IONE
        SACL     ICOUNT
;
;OUTPUT RESULT
;
SOUT:   LAC      PSCOPE      ;GET OUTPUT SAMPLE
        TBLR     ISMPL
;
;CONVERT 2'S COMPLEMENT NUMBER TO MU-LAW
;
        LAC      ISMPL,S14   ;ISMPL IS A 16-BIT 2'S COMPL. NUMBER
        SACH     ISMPL      ;CONVERT TO A 14-BIT 2'S COMPL. NUMBER
        LAC      IONE,S15
        AND      ISMPL
        SNZ      MINUS
        LACK     33D
        ADD      ISMPL
        SACL     ISMPL
        SUB      IONE,S13
        BGE     OVER
        LAC      IONE,S9
        SUB      ISMPL
        BGZ     LITTLE
        LAC      IONE,S11
        SUB      ISMPL
        BGZ     TWO3
        LAC      IONE,S12
        SUB      ISMPL
        BGZ     CHORD1
        LAC      ISMPL,S8
        SACH     ISMPL
        LACK     60H
        ADD      ISMPL

```

```

      B      DONE
CHORD1: LAC   ISMPL,S9
      SACH   ISMPL
      LACK   50H
      ADD    ISMPL
      B      DONE
TWO3:  LAC   IONE,S10
      SUB    ISMPL
      BGZ    CHORD3
      LAC    ISMPL,S10
      SACH   ISMPL
      LACK   40H
      ADD    ISMPL
      B      DONE
CHORD3: LAC   ISMPL,S11
      SACH   ISMPL
      LACK   30H
      ADD    ISMPL
      B      DONE
LITTLE: LAC   IONE,S7
      SUB    ISMPL
      BGZ    SIX7
      LAC    IONE,S8
      SUB    ISMPL
      BGZ    CHORD5
      LAC    ISMPL,S12
      SACH   ISMPL
      LACK   20H
      ADD    ISMPL
      B      DONE
CHORD5: LAC   ISMPL,S13
      SACH   ISMPL
      LACK   10H
      ADD    ISMPL
      B      DONE
SIX7:  LAC   IONE,S6
      SUB    ISMPL
      BGZ    CHORD7
      LAC    ISMPL,S14
      SACH   ISMPL
      LAC    ISMPL
      B      DONE
CHORD7: LAC   ISMPL,S15
      SACH   ISMPL
      LACK   0FH
      AND    ISMPL
      B      DONE
OVER:  LACK   7FH
      B      DONE
UNDER: LACK   0FFH
      B      DONE
MINUS: LACK   33D
      SUB    ISMPL
      SACL   ISMPL
      SUB    IONE,S13
      BGE2   UNDER
      LAC    IONE,S9
      SUB    ISMPL
      BGZ    SMALL
      LAC    IONE,S11
      SUB    ISMPL

```

```

        BGZ      THREE2
        LAC      IONE,S12
        SUB      ISMPL
        BGZ      CORD01
        LAC      ISMPL,S8
        SACH     ISMPL
        LACK     0E0H
        ADD      ISMPL
        B        DONE
CORD01: LAC      ISMPL,S9
        SACH     ISMPL
        LACK     0D0H
        ADD      ISMPL
        B        DONE
THREE2: LAC      IONE,S10
        SUB      ISMPL
        BGZ      CORD03
        LAC      ISMPL,S10
        SACH     ISMPL
        LACK     0C0H
        ADD      ISMPL
        B        DONE
CORD03: LAC      ISMPL,S11
        SACH     ISMPL
        LACK     0B0H
        ADD      ISMPL
        B        DONE
SMALL:  LAC      IONE,S7
        SUB      ISMPL
        BGZ      SEVEN6
        LAC      IONE,S8
        SUB      ISMPL
        BGZ      CORD05
        LAC      ISMPL,S12
        SACH     ISMPL
        LACK     0A0H
        ADD      ISMPL
        B        DONE
CORD05: LAC      ISMPL,S13
        SACH     ISMPL
        LACK     90H
        ADD      ISMPL
        B        DONE
SEVEN6: LAC      IONE,S6
        SUB      ISMPL
        BGZ      CORD07
        LAC      ISMPL,S14
        SACH     ISMPL
        LACK     80H
        ADD      ISMPL
        B        DONE
CORD07: LAC      ISMPL,S15
        SACH     ISMPL
        LACK     70H
        ADD      ISMPL
DONE:   SACL     ISMPL
;
        LACKA    APITCH      ;GET ADDRESS OF PITCH
        TBLR     ITEMP
        LAC      PSCOPE      ;COMPARE WITH CURRENT ADDRESS
        SUB      ITEMP

```



```

;      BZ      FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
;      LACKA   AFMT1      ;GET ADDRESS OF 1ST FORMANT
;      TBLR    ITEMP
;      LAC     PSCOPE      ;COMPARE WITH CURRENT ADDRESS
;      SUB     ITEMP
;      BZ      FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
;      LACKA   AFMT2      ;GET ADDRESS OF 2ND FORMANT
;      TBLR    ITEMP
;      LAC     PSCOPE      ;COMPARE WITH CURRENT ADDRESS
;      SUB     ITEMP
;      BZ      FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
;      LACKA   AFMT3      ;GET ADDRESS OF 3RD FORMANT
;      TBLR    ITEMP
;      LAC     PSCOPE      ;COMPARE WITH CURRENT ADDRESS
;      SUB     ITEMP
;      BZ      FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
;      FLGDWN: LACK      7H      ;DRIVE = 0 ON CRC => FLAG DOWN
;                   B        INC
;
;      FLGUP:  LACK      87H      ;DRIVE = 1 ON CRC => FLAG UP
;
;      INC:    SACL      ITEMP
;                   OUT     ITEMP,PA6      ;UPDATE DRIVE ON CRC
;
;      LACK    OFBH      ;UNMASK CODEC INTERRUPT (DRIVE NOT CHANGED)
;      AND     ITEMP
;      SACL    ITEMP
;      OUT     ITEMP,PA6
;
;      OUT     ISMPL,PA3      ;OUTPUT SCOPE INFO. TO CODEC
;
;      LAC     PSCOPE      ;INCRIMENT POINTER
;      ADD     IONE
;      SACL    PSCOPE
;
;      LAC     CSCOPE      ;DECRIMENT COUNTER
;      SUB     IONE
;      SACL    CSCOPE
;      BNZ     ISROUT      ;ZERO YET?
;
;      FIN:    LACKA     ASCP1      ;IF YES THEN UPDATE COUNTER AND POINTER
;                   TBLR    PSCOPE
;                   LACK     (OUTB1+OUTB2)
;                   SACL    CSCOPE
;
;      ISROUT: LACKA     ACOUNT      ;STORE ISR VARIABLES IN PRAM
;                   TBLW    ICOUNT
;                   ADD     IONE
;                   TBLW    IPTR
;                   ADD     IONE
;                   TBLW    PSCOPE
;                   ADD     IONE
;                   TBLW    CSCOPE
;                   ADD     IONE
;                   TBLW    IFLAG
;
;      LACKA   AFFTSTG      ;RETURN FFT VARIABLES

```

```

        TBLR    ISMPL
        ADD     IONE
        TBLR    ITEMP
        ADD     IONE
        TBLR    ICOUNT
        ADD     IONE
        TBLR    IPTR
        ADD     IONE
        TBLR    PSCOPE
        ADD     IONE
        TBLR    CSCOPE
        ADD     IONE
        TBLR    IFLAG
;
        ZALS    ILOW          ;RETURN PRE-ISR STATE
        ADDH    IHIGH
        LST     ISTATE
        EINT
        RET
;
        ELSE
;
; ***** ISR FOR EVM BOARD *****
;
INPUT:  SST     ISTATE        ;STORE PRE-ISR STATE
        LDPK    1
        SACL    ILOW
        SACH    IHIGH
;
        LACKA   AEFTSTG      ;STORE FFT VARIABLES IN PRAM
        TBLW    ISMPL
        ADD     IONE
        TBLW    ITEMP
        ADD     IONE
        TBLW    ICOUNT
        ADD     IONE
        TBLW    IPTR
        ADD     IONE
        TBLW    PSCOPE
        ADD     IONE
        TBLW    CSCOPE
;
        LACKA   ACOUNT      ;RETURN ISR VARIABLES
        TBLR    ICOUNT
        ADD     IONE
        TBLR    IPTR
        ADD     IONE
        TBLR    PSCOPE
        ADD     IONE
        TBLR    CSCOPE
;
        LAC     ICOUNT        ;IF 0, PROCESSING GOING ON
        BZ      SOUT          ; SO --DON'T-- COLLECT DATA
;
        IN      ISMPL,PA2     ;GET BINARY COMPLEMENT DATA
        LAC     ISMPL,S12
        SACH    ISMPL        ;AND SHIFT IT RIGHT 4 PLACES
;
        LAC     ISMPL
        BLZ     NEG
        POSI:   SUB     IONE,S11 ;CONVERT TO 2'S COMPLEMENT

```

```

      B      GON
NEG:  ADD    IONE,S11
GON:  SACL   ISMPL
;
      LAC    IPTR      ;STORE INPUT DATA
      TBLR   ISMPL
      ADD    IONE      ;INCRIMENT POINTER
      SACL   IPTR
;
      LAC    ICOUNT    ;DECRIMENT INPUT COUNTER
      SUB    IONE
      SACL   ICOUNT
;
;OUTPUT RESULT
;
SOUT: LACKA  APITCH    ;GET ADDRESS OF PITCH
      TBLR   ITEMP
      LAC    PSCOPE    ;COMPARE WITH CURRENT ADDRESS
      SUB    ITEMP
      BZ     FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
      LACKA  AFMT1     ;GET ADDRESS OF 1ST FORMANT
      TBLR   ITEMP
      LAC    PSCOPE    ;COMPARE WITH CURRENT ADDRESS
      SUB    ITEMP
      BZ     FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
      LACKA  AFMT2     ;GET ADDRESS OF 2ND FORMANT
      TBLR   ITEMP
      LAC    PSCOPE    ;COMPARE WITH CURRENT ADDRESS
      SUB    ITEMP
      BZ     FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
      LACKA  AFMT3     ;GET ADDRESS OF 3RD FORMANT
      TBLR   ITEMP
      LAC    PSCOPE    ;COMPARE WITH CURRENT ADDRESS
      SUB    ITEMP
      BZ     FLGUP      ;IF THEY ARE THE SAME THEN FLAG UP
;
FLGDWN: LACK  10D      ;C7 = 0 ON AIB => FLAG DOWN
      B      INC
;
FLGUP:  LACK  138D     ;C7 = 1 ON AIB => FLAG UP
;
INC:    SACL   ISMPL
      OUT    ISMPL,PA0 ;UPDATE C7 ON AIB
;
      LAC    PSCOPE    ;GET NEXT DATA POINT FOR OUTPUT
      TBLR   ISMPL      ; FROM SCOPE BUFFER
      ADD    IONE
      SACL   PSCOPE
;
      LAC    ISMPL      ;RESULT IS A 16-BIT #, DON'T SHIFT LEFT
      ADD    IONE,S15    ;CONVERT TO BINARY
      SACL   ISMPL
      OUT    ISMPL,PA2   ;OUTPUT SCOPE INFO.
;
      LAC    CSCOPE    ;DECRIMENT COUNTER
      SUB    IONE
      SACL   CSCOPE
      BNZ    1SROUT     ;ZERO YET

```

```

;
FIN:   LACKA   ASCP1           ;IF YES THEN UPDATE COUNTER AND POINTER
      TBLR    PSCOPE
      LACK    (OUTB1+OUTB2)
      SACL    CSCOPE
;
ISROUT: LACKA   ACOUNT       ;STORE ISR VARIABLES IN PRAM
      TBLW    ICOUNT
      ADD     IONE
      TBLW    IPTR
      ADD     IONE
      TBLW    PSCOPE
      ADD     IONE
      TBLW    CSCOPE
;
      LACKA   AFFTSTG         ;RETURN FFT VARIABLES
      TBLR    ISMPL
      ADD     IONE
      TBLR    ITEMP
      ADD     IONE
      TBLR    ICCUNT
      ADD     IONE
      TBLR    IPTR
      ADD     IONE
      TBLR    PSCOPE
      ADD     IONE
      TBLR    CSCOPE
;
      ZALS    ILOW           ;RETURN PRE-ISR STATE
      ADDH    IHIGH
      LST     ISTATE
      EINT
      RET
;
      ENDIF
;
;
;----- DIVISION -----
;
DIVNEW: SACH    DDH
      SACL    DDL
      ZAC
      SACL    TEMP2
      LACK    1
      SACL    ONE
      LAC     ONE,S15
      SUB     ONE
      SACL    PLUS1
;
      SAR     ARO,DUMMY
      LARK    ARO,15
      LARP    0
;
      LAC     ONE,S15
      AND     DDH
      BNZ     LESS0
      ZALH    DDH
      ADDS    DDL
;
221:   ADD     TEMP2
      SUBC    DR

```

```

;      BA'Z      Z21
;
;      SACH      REM
;      AND       PLUS1
;
;      SACL      TEMP2
;      ZAC
;      SUB       TEMP2
;
;      LAR       ARO,DUMMY
;      RET
;
LESS0: LACK      0
      SUBH      DDH
      SUBS      DDL
;
ZZ2:   ADD       TEMP2
      SUBC      DR
      BANZ      Z22
;
;      SACH      REM
;      AND       PLUS1
;      LAR       ARO,DUMMY
;      RET
;
;
;----- PRAM DATA -----
;
;NOTE:  PRAM DATA MUST BE DECLARED IN THE FOLLOWING ORDER!!
;
FTABLE: DW 2928,1167,5711,-2379,763,5774,-3328,1650
;
WIND1:                                     ;GENERATE PITCH'S WINDOW
      I=1
      REPT      W1+1                       ;USES SYMETRY
      TRGEN1    %I
      I=I+1
      ENDM
;
WIND2:                                     ;GENERATE ENV'S WINDOW
      I=1
      REPT      W2+1                       ;USES SYMETRY
      TRGEN2    %I
      I=I+1
      ENDM
;
CNVRT: DW 0000H, 0002H, 0004H, 0006H, 0008H, 000AH, 000CH, 000EH
      DW 0010H, 0012H, 0014H, 0016H, 0018H, 001AH, 001CH, 001EH
      DW 0021H, 0025H, 0029H, 002DH, 0031H, 0035H, 0039H, 003DH
      DW 0041H, 0045H, 0049H, 004DH, 0051H, 0055H, 0059H, 005DH
      DW 0063H, 006BH, 0073H, 007BH, 0083H, 008BH, 0093H, 009BH
      DW 00A3H, 00ABH, 00B3H, 00BBH, 00C3H, 00CBH, 00D3H, 00DBH
      DW 00E7H, 00F7H, 0107H, 0117H, 0127H, 0137H, 0147H, 0157H
      DW 0167H, 0177H, 0187H, 0197H, 01A7H, 01B7H, 01C7H, 01D7H
      DW 01EFH, 020FH, 022FH, 024FH, 026FH, 028FH, 02AFH, 02CFH
      DW 02EFH, 030FH, 032FH, 034FH, 036FH, 038FH, 03AFH, 03CFH
      DW 03FFH, 043FH, 047FH, 04BFH, 04FFH, 053FH, 057FH, 05BFH
      DW 05FFH, 063FH, 067FH, 06BFH, 06FFH, 073FH, 077FH, 07BFH
      DW 081FH, 089FH, 091FH, 099FH, 0A1FH, 0A9FH, 0B1FH, 0B9FH
      DW 0C1FH, 0C9FH, 0D1FH, 0D9FH, 0E1FH, 0E9FH, 0F1FH, 0F9FH
      DW 105FH, 115FH, 125FH, 135FH, 145FH, 155FH, 165FH, 175FH

```

```

        DW    185FH, 195FH, 1A5FH, 1B5FH, 1C5FH, 1D5FH, 1E5FH, 1F5FH
;
SCOPE1: DW    03FFFH                ;SYNC PULSE
        DW    OUTB1-1    DUP (?)    ;PITCH'S DISPLAY BUFFER
SCOPE2: DW    OUTB2      DUP (?)    ;ENVELOPE'S DISPLAY BUFFER
;
        IF CRC
BUFF4: DW    (NPTS1/2)      DUP (?) ;INPUT BUFFER WHERE ISR COLLECTS DATA
        ELSE
BUFF4: DW    NPTS1          DUP (?) ;INPUT BUFFER WHERE ISR COLLECTS DATA
        ENDIF
;
BUFF3: DW    (OUTB1+NPTS2-NPTS1)  DUP (?) ;WHERE PITCH DUMPS WINDOWED DATA
BUFF1: DW    (NPTS1-NPTS2)      DUP (?) ;START OF PROCESSING BUFFER
BUFF2: DW    NPTS2              DUP (?) ;DATA THAT ENV USES
;
;NOTE: THE PROGRAM WILL FAIL IF THE NEXT LINE IS NOT PRESENT
;      IT IS NECESSARY FOR ENV'S AUTOCORRELATION ROUTINE
;
DW 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
;
THEEND: DW    0                  ;THE END
;
        IF CRC
        ORG    (2*FFTLOC)-2      ;FILL FROM END TO FFT WITH ZEROES
        ENDIF
;
        DW    0
        MAIN  ENDS
        END   START

```

Appendix B

TMS32010 source code listing Test demonstrating the CRC board to Sherbrooke stimulator interface.

```

PAGE 63,132
;
;----- SHERB-1.ASM -----
;
; COPYRIGHT (C) 1988 JON BOSLOY AND PETER A. BARSZCZEWSKI, CARLETON UNIVERSITY
; ALL RIGHTS RESERVED
;
; A PROGRAM FOR THE CRC BOARD WHICH TESTS THE SHERBROOKE INTERFACE
; CIRCUIT. IT OUTPUTS 'C3F0FD' TO THE SHERBROOKE INTERFACE.
;
;----- CROSS ASSEMBLER -----
;
IF1
    INCLUDE C:\ASM\TIMACG.ASM
ENDIF
;
;----- DATA MEMORY -----
;
DORG0
DATA    PSR
DATA    LOW16
DATA    HIGH8
DATA    TIME
;
;----- PROGRAM MEMORY -----
;
AORG
;
START:  DW      80H DUP (?)      ;DOWNLINE LOADER
;
        B        GO
        B        ISR
;
KLOW:   DW      0F0FDH           ;LOWER 16-BITS OF 24-BIT COMMAND WORD
KTIME:  DW      0001AH           ;TIME BETWEEN OUTPUT COMMANDS (46.5 USEC)
;
;----- INITIALIZATION -----
;
GO:     LACKA    KLOW             ;GET LOWER 16-BITS OF 24-BIT COMMAND WORD
        TBLR    LOW16
        LACK     0C3H            ;GET UPPER 8-BITS OF 24-BIT COMMAND WORD
        SACL     HIGH8
        LACKA    KTIME           ;GET COUNT-DOWN TIMER VALUE
        TBLR     TIME
        OUT      LOW16,PA5       ;OUTPUT LOWER 16-BITS
        OUT      HIGH8,PA5       ;OUTPUT UPPER 16-BITS
        OUT      TIME,PA5
        LACK     6               ;INITIALIZE GLUE CHIP - UNMASK RXC INTERRUPT
        SACL     PSR
        OUT      PSR,PA6
        EINT                    ;ENABLE INTERRUPTS
;
;----- MAIN PROGRAM -----
;
LOOP:   B        LOOP           ;INFINITE LOOP
;
;----- INTERRUPT SERVICE ROUTINE -----
;
ISR:    OUT      LOW16,PA5       ;1) OUTPUT LOWER 16-BITS OF 24-BIT COMMAND
        OUT      HIGH8,PA5       ;2) OUTPUT UPPER 8-BITS OF 24-BIT COMMAND

```



```
      OUT      TIME,PA5      ;3) OUTPUT COUNT DOWN TIME FOR NEXT INT
      LACK      7            ;4) MASK RXC INTERRUPT OF GLUE CHIP
      SACL      PSR
      OUT      PSR,PA6
      LACK      6            ;5) UNMASK RXC INTERRUPT OF GLUE CHIP
      SACL      PSR
      OUT      PSR,PA6
      EINT      ;6) ENABLE INTERRUPTS
      RET

;
;-----
;

      MAIN ENDS
      END START
```

Appendix C

TMS320E17 source code listing of the feature extraction system software for the E17-board and test bed.

```

PAGE 63,132
;
;----- E17-3.ASM -----
;
; COPYRIGHT (C) 1988 PETER A. BARSZCZEWSKI, CARLETON UNIVERSITY
; ALL RIGHTS RESERVED.
;
;       IF1
;         INCLUDE C:\ASM\TIMACG.ASM
;         INCLUDE C:\ASM\W200.MAC
;         INCLUDE C:\ASM\W82.MAC
;       ENDIF
;
;----- CONSTANTS -----
;
;       .RADIX 10
;
;       TEST=0           ;CONDITIONAL ASSEMBLY TEST
;                       ;0(FALSE)-AUTONOMOUS E17   1(TRUE)-TEST BED
;
;       ; DISPLAY MESSAGE DURING ASSEMBLY SO USER KNOWS WHATS GOING ON
;
;       IF TEST
;         IF1
;       %OUT *           ASSEMBLING FOR E17 TEST BED
;       ENDIF
;       ELSE
;         IF1
;       %OUT *           ASSEMBLING FOR AUTONOMOUS E17 SYSTEM
;       ENDIF
;       ENDIF
;
;       FFTLOC=0AD1H     ;STARTING ADDRESS OF FFT
;
;       NPTS1=228D       ;NUMBER OF POINTS COLLECTED AND USED FOR PITCH EXTRACTION
;       M1=4D            ;ORDER
;       OUTB1=NPTS1/3    ;NUMBER OF POINTS IN PITCH'S OUTPUT BUFFER
;       W1=OUTB1         ;NUMBER OF POINTS IN PITCH'S WINDOW FUNCTION
;
;       JMI=4D           ;LOWER LIMIT FOR PEAK PICKER
;       JMA=41D          ;UPPER LIMIT FOR PEAK PICKER
;       I2M=47D          ;MAX NUMBER OF ZERO CROSSINGS FOR VOICED
;       IR1=4D           ;THRESHOLD FOR SIGNAL RMS
;
;       NPTS2=200D       ;NUMBER OF POINTS USED FOR FORMANT EXTRACTION
;       M2=10D           ;ORDER
;       OUTB2=64D        ;NUMBER OF POINTS IN ENVELOPE'S OUTPUT BUFFER
;       W2=NPTS2         ;NUMBER OF POINTS IN ENVELOPE'S WINDOW FUNCTION
;
;       IT=4D            ;FOR AUTO2'S BUFFERING SCHEME
;       NR=NPTS2/1T
;       BEND=NR+M2
;
;       NSPECT=32        ;# OF BYTES IN MAGNITUDE OF SPECTRUM
;
;       BIAS=33D         ;BIAS FOR LINEAR TO MU-LAW CONVERSION
;
;----- MACROS -----
;

```

```

DEFBFB MACRO      I,J1,J2
DATA      B&I
SAME      NA&J1
DATA      XD&I
SAME      NA&J2
ENDM

;

DEF MACRO      I,J
DATA      LOW&I
SAME      FK&J
DATA      HIGH&I
SAME      RX&J
ENDM

;

CALFFT MACRO
DW      0F800H
DW      FFTLOC
ENDM

;

TRGEN1 MACRO      I
DW      TRI&I
ENDM

;

TRGEN2 MACRO      I
DW      TSI&I
ENDM

;
;
----- PAGE 0 -----
;

DORG0

;

I=1
REPT      M2+1
J1=2*I-1
J2=2*I
DEFBFB    %I,%J1,%J2
I=I+1
ENDM

;
; WARNING:  AUTO2 USES LOCS 0-59D.  LOW0 STARTS AT LOCATION 60D.
;

RES      38D

;

I=0
J=1                      ;DEFINE FK's, RX's
REPT      M2+1
DEF      %I,%J
I=I+1
J=J+1
ENDM

;
; WARNING:  AUTO1 USES 0-75D DRAM - THE VARIABLES BELOW MUST SURVIVE
;          DPTR IS AT LOCATION 82D.
;

DATA      DPTR          ;GENERAL PURPOSE PRIMARY DATA POINTER
DATA      PTR2          ;GENERAL PURPOSE SECONDARY DATA POINTER
SAME      LGFLAG        ;LG,KTOA
DATA      COUNT         ;GENERAL PURPOSE COUNTER
DATA      TEMP          ;GENERAL PURPOSE TEMPORARY VARIABLE
SAME      Y             ;LG

```

```

DATA    SIG                ;GENERAL PURPOSE DATA SAMPLE
DATA    DR                 ;DIVIDE
SAME    Y1                 ;FILTER
DATA    DDH                ;DIVIDE
SAME    Y11                ;FILTER
DATA    DDL                ;DIVIDE
SAME    Y2                 ;FILTER
DATA    PLUS1              ;DIVIDE
SAME    Y21                ;FILTER
DATA    TEMP2              ;DIVIDE
SAME    Y22                ;FILTER
DATA    DUMMY              ;DIVIDE, LG
SAME    Y3                 ;FILTER
DATA    ZMASK              ;AUTO1, NORMAL, ZCD
SAME    Y31                ;FILTER
DATA    SCOUNT             ;AUTO1, NORMAL, NPIC
SAME    Y32                ;FILTER
DATA    XLOW               ;GETDAT, ZMEAN, AUTO1
SAME    RXT                ;MAGNTD
DATA    XHIGH              ;GETDAT, ZMEAN, AUTO1
SAME    IYT                ;MAGNTD

;
; WARNING: FC'S MUST BE CONTIGUOUS
;
DATA    DELTA              ;AUTO1, AUTO2
SAME    FC1                ;FILTER
DATA    SPTR2              ;AUTO1
SAME    FC2                ;FILTER
DATA    BASE               ;AUTO1
SAME    BB                 ;DECIS
SAME    FC3                ;FILTER
DATA    TEMPN              ;AUTO1, ZCD
SAME    SLM1               ;DECIS
SAME    FC4                ;FILTER
DATA    QTEMP              ;AUTO1
SAME    NRCNT              ;AUTO2
SAME    SL                 ;DECIS
SAME    FC5                ;FILTER
DATA    POWER              ;AUTO1
SAME    NRD                ;AUTO2
SAME    SLP1               ;DECIS
SAME    FC6                ;FILTER
DATA    DP1                ;AUTO1
SAME    LAGCNT             ;AUTO2
SAME    LL                 ;DECIS
SAME    FC7                ;FILTER
DATA    IR0                ;MAIN, DECIS
SAME    FC8                ;FILTER
DATA    MAXVAL             ;NPIC, PPIC
SAME    TOP                ;GETDAT, AUTO2
SAME    AP                 ;DECIS
DATA    ENRLO              ;AUTO1 (LOW 16 BITS OF ENERGY)
SAME    WIND               ;GETDAT, AUTO2
SAME    L                  ;DECIS
DATA    ENRHI              ;AUTO1 (HIGH 16 BITS OF ENERGY)
SAME    WPTR               ;GETDAT, AUTO2
SAME    ALI                ;DECIS
DATA    NZ                 ;ZCD, DECIS

;
; STORAGE FOR 11 AUTOCORRELATION COEFFICIENTS
;

```

```

      DATA   STR
      RES     100
;
; WARNING:  THE FOLLOWING IMPORTANT DATA MUST LAST FROM FRAME TO FRAME
;           AND NO, N1, N2, N3, NMIN, NMAX MUST BE CONTIGUOUS
;
      DATA   ONE           ;CONSTANT
      DATA   PITCH         ;PITCH              (DECIS)
      DATA   NO            ;DECISION VARIABLES  (DECIS)
      DATA   N1
      DATA   N2
      DATA   N3
      DATA   NMIN
      DATA   NMAX
;
;----- PAGE 1 -----
;
      DORG1
;
      DATA   ITEMP         ;FIRST LOCATION USED AS A TEMPORARY
;
      RES     113D          ;INPUT BUFFER IS 114 LOCATIONS.
      DATA   INBUFF        ;TOP OF THE INPUT BUFFER
;
; WARNING:  FRMT1, FRMT2, AND FRMT3 MUST BE CONTIGUOUS
;
      DATA   FRMT1         ;FORMANTS              (NPIC)
      DATA   FRMT2
      DATA   FRMT3
;
      DATA   ISMPL         ;10 LOCATIONS USED BY INPUT ISR ONLY
      DATA   PSCOPE
      DATA   CSCOPE
      DATA   IFLAG
      DATA   PREARO
      DATA   ISRARO
      DATA   ISTATE
      DATA   IONE
      DATA   IHIGH
      DATA   ILOW
;
;----- PRAM -----
;
      AORG           ;START PROGRAM SECTION
;
      START: B         GO
               B         INPUT
;
      DW          1CH      DUP (0) ;BUFFER FOR TBLW/OUT AND TBLR/IN OVERLAP
;
      ATABLE: ADDR FTABLE      ;FILTER CONSTANT TABLE
      AWIND1: ADDR WIND1       ;WINDOW FUNCTION TABLE FOR PITCH
      AWIND2: ADDR WIND2       ;WINDOW FUNCTION TABLE FOR FORMANTS
      ACNVRT: ADDR CNVRT        ;MU-LAW TO 2'S COMPL. CONVERSION TABLE
      ATARG:  ADDR TARGET       ;BASE ADDRESS CALA IN AUTO1
;
      ASCP1:  ADDR SCOPE1       ;SCOPE BUFFERS
      ASCP2:  ADDR SCOPE2
;
; THE PROCESSOR INITIALIZATION CONSTANTS

```

```

;
ASET1:  DW    1988H          ;SET PORT 1 AS CONTROL PORT
ASET2:  DW    1CFEH          ;THE HIGH CONTROL BITS (REMAIN CONSTANT)
ASENC:  DW    1888H          ;SERIAL ENCODE
AFLGDN: DW    1888H          ;KEEPS XF LATCH LOW
AFLGUP: DW    1C88H          ;SENDS XF LATCH HIGH
;
;
;----- INITIALIZATION -----
;
GO:      LDPK    0              ;INIT ON PAGE 0
        LACK    JMI            ;LOWER LIMIT FOR PEAK PICKER
        SACL    NMIN           ;
        LACK    JMA            ;UPPER LIMIT FOR PEAK PICKER
        SACL    NMAX           ;
        LACK    1              ;CONSTANT ON PAGE 0
        SACL    ONE
;
; LOAD SAMPLING ISR DATA
;
        LDPK    1              ;INIT ON PAGE 1
        SACL    IONE           ;CONSTANT ON PAGE 1
        ZAC      ;TOGGLE FLAG FOR DATA INPUT ON PAGE 1
        SACL    IFLAG
        LACKA    ASCP1          ;POINTER FOR DISPLAY OF SCOPE BUFFER
        TBLR     PSCOPE
        LACK     (OUTB1+OUTB2)  ;COUNTER FOR DISPLAY OF SCOPE BUFFER
        SACL     CSCOPE
        LACK     (INBUFF+128D)  ;INITIALIZE INPUT BUFFER POINTER/COUNTER
        SACL     ISRAR0         ; ON PAGE 1 (HENCE 128D OFFSET)
;
        LACKA    ASET1          ;INITIALIZE THE PROCESSOR HARDWARE
        TBLR     ITEMP
        OUT      ITEMP,PA0
        LACKA    ASET2          ;SET TIMING
        TBLR     ITEMP
        OUT      ITEMP,PA1
        LACKA    ASENC          ;SERIAL ENCODE MODE
        TBLR     ITEMP
        OUT      ITEMP,PA0
;
        EINT                  ;ENABLE INTERRUPTS
;
;
;----- THE MAIN PROGRAM -----
;
HERE:    LDPK    1
;
LOOP:    LACK    128D           ;FINISHED COLLECTING DATA WHEN COUNTER/POINTER
        SUB     ISRAR0         ; - PAGE 1 OFFSET
        BNZ     LOOP
;
        LDPK    0
;
        CALL    ZCD            ;COUNT NUMBER OF ZERO CROSSINGS
;
        CALL    AUTO2          ;11 LAG AUTO AND WINDOW
;
        CALL    NORMAL
;
;----- GETDAT STARTS DATA COLLECTION -----

```

```

;
;      CALL      GETDAT      ;FILTER, WINDOW AND DOWNSAMPLE PITCH'S INPUT
;
;-----
;
;      CALL      ZMEAN      ;CONVERT TO ZERO MEAN
;
;      LACK      4
;      CALL      AUTO1      ;4 LAG AUTO
;
;      ZALS      ENRLO      ;INPUT ENERGY (32 BITS)
;      ADDH      ENRHI
;      SUB      ONE,S15      ;THRESHOLD FOR CODEC'S 14 BIT SAMPLES - 2**15
;      BGEZ      AD63      ;ENERGY HIGH ENOUGH
;      ZAC
;      SACL      IRO      ;TELL DECIS UNVOICED
;      B      AD50
;
;      ; MOVE PREPARED DATA TO PAGE 1 SO THAT LG AND KTOA WILL NOT OVERWRITE IT
;
AD63:  LARK      ARO,(OUTB1-1)
      LARK      AR1,(OUTB1-1)+128D+5D
      LARP      0
LOOP9:  LAC      *,AR1
      SACL      *-,AR0
      BANZ      LOOP9
;
      LARK      ARO,128D      ;5 COEFS IN PRE-BUFFER
      LARP      0
      LAC      **+,AR0
      SACL      RX1
      LAC      **+,AR0
      SACL      RX2
      LAC      **+,AR0
      SACL      RX3
      LAC      **+,AR0
      SACL      RX4
      LAC      **+,AR0
      SACL      RX5
;
      LACK      0D      ;FLAG LG FOR 5TH ORDER AND RETURN HERE
      SACL      LGFLAG
      B      LG      ;RX'S TO FK'S
;
LGRET1: CALL      KTOA      ;FK'S TO NA'S
;
;      CALL      INVFIL      ;FILTER USING NA'S
;
;      LACK      255D
;      SACL      IRO      ;FAKE THIS FOR NOW
;
;      LACK      (OUTB1-1)
;      CALL      AUTO1      ;AUTOCORRELATE FILTERED SIGNAL
;
AD50:  CALL      DECIS      ;PITCH DECISION ROUTINE
;
;      LAC      IRO      ;IF AUTO1-75 HAS NOT EXECUTED THEN
;      BZ      CONT1      ; SCOPE BUFFER CONTAINS GARBAGE
;
;      IF TEST
;          LARK      ARO,128D

```



```

        LARK    AR1,(OUTB1-1)
        LACKA   ASCP1
        TBLR    DPTR
        LAC     DPTR
LOOP6:  LARP    0
        TBLW    **+,AR1
        ADD     ONE
        BANZ    LOOP6
    .
ENDIF
;
;
-----
;
CONT1:  LARK    AR0,RX1      ;OUTPUT OF NORMAL MUST BE MOVED BACK
        LARK    AR1,STR     ; SO LG CAN FIND IT
        LACK    11         ;NUMBER OF POINTS TO BE MOVED
        SACL    COUNT
        LARP    1
LOOP5:  LAC     **+,AR0      ;LOAD LOCATIONS 104-114
        SACL    **+,AR0      ;STORE IN RX'S
        MAR     **+,AR1      ;MOVE AR0 SO IT POINTS TO NEXT RX
        LAC     COUNT        ;MOVE 11 POINTS
        SUB     ONE
        SACL    COUNT
        BNZ     LOOP5
;
        LACK    1D          ;FLAG LG FOR 11TH ORDER AND RETURN HERE
        SACL    LGFLAG
        B       LG          ;RX'S TO FK'S
;
LGRET2: CALL    KTOA        ;FK'S TO NA'S
;
        LARK    AR0,128D+7D
        LARP    0
        LAC     ONE
        SACL    **+,AR0
        LAC     PITCH
        SACL    **+,AR0
        LAC     N0
        SACL    **+,AR0
        LAC     N1
        SACL    **+,AR0
        LAC     N2
        SACL    **+,AR0
        LAC     N3
        SACL    **+,AR0
        LAC     NMIN
        SACL    **+,AR0
        LAC     NMAX
        SACL    **+,AR0
;
        CALL    PREFFT      ;PREPARE NA'S AND DRAM FOR FFT
;
        CALFFT              ;DETERMINE FREQ. SPECTRUM USING 11 NA'S
;
        LDPK    0
        LARK    AR0,128D+7D
        LARP    0
        LAC     **+,AR0
        SACL    ONE
        LAC     **+,AR0

```

```

        SACL    PITCH
        LAC     **+,ARO
        SACL    N0
        LAC     **+,ARO
        SACL    N1
        LAC     **+,ARO
        SACL    N2
        LAC     **+,ARO
        SACL    N3
        LAC     **+,ARO
        SACL    NMIN
        LAC     **+,ARO
        SACL    NMAX
;
        CALL    MAGNTD          ;CALCULATE MAGNITUDE OF SPECTRUM
;
        IF TEST
            LARK    ARO,0          ;MOVE SPECTRUM TO PRAM FOR DISPLAY
            LARK    AR1,NSPECT-1
            LACKA    ASCP2
            TBLR    DPTR
            LAC     DPTR
WRITE:   LARP    0
            TBLW    *,ARO
            ADD     ONE
            TBLW    **+,AR1
            ADD     ONE
            BANZ    WRITE
        ENDIF
;
        CALL    NPIC          ;FIND FORMANTS IN SPECTRUM
;
        B       HERE          ;AND GO AGAIN
;
;
;----- ZERO CROSSINGS DETECTOR -----
;
ZCD:     LARK    ARO,(NPTS1/2)-2      ;LOOP COUNTER
        LARK    AR1,INBUFF+128D      ;INITIALIZE ZERO CROSSINGS COUNT
        LACK    80H                  ;MASK #1 - 0000 0000 1000 0000
        SACL    TEMPN
        LAC     TEMPN,S8              ;MASK #2 - 1000 0000 0000 0000
        SACL    ZMASK
        ZAC
        SACL    NZ
;
ZLOOP:   LARP    1
        LAC     *,S8,AR1
        SACH    TEMP                ;GET HIGH BYTE (STILL IN MU-LAW)
        LAC     TEMP
        XOR     *,AR1                ;OPPOSITE SIGN BITS WILL PRODUCE A ONE
        AND     TEMPN                ;GET RID OF GARBAGE
        BZ      NOZC1                ;NO POSITIVE
        LAC     NZ                    ;INCRIMENT THE ZERO COUNT
        ADD     ONE
        SACL    NZ
;
NOZC1:   LAC     **-,S8,AR1          ;GET LOW BYTE OF CURRENT WORD
        XOR     *,ARO                ;XOR WITH NEXT WORD'S HIGH BYTE
        AND     ZMASK                ;STRIP OFF GARBAGE
        BZ      NOZC2

```

```

        LAC      NZ          ; INCRIMENT THE ZERO COUNT
        ADD      ONE
        SACL     NZ
;
NOZC2:  BANZ     ZLOOP        ; LOOP UNTIL DONE
;
        RET
;
;
;----- PREFFT -----
;
; NOW: NA1 IN 0, NA2 IN 1, ... , NA11 IN 10
; DOUBLE NA'S UNTIL 16K < LARGEST < 32767
; MOVE NA1-NA11 TO NA12-NA22 AND FILL REST OF DRAM WITH 0'S
; MOVE DATA FROM PAGE 0 TO BUFF2 (PRAM) FOR THE FFT
;
PREFFT: LARK     ARO, NA11
        LARK     AR1, NA22
        LARP     0
SW1:    LAC      *, AR1        ; NA11, ..., NA1
        SACL     *-, ARO       ; NA22, ..., NA12
        BANZ     SW1
;
DSTART: LARK     ARO, NA22      ; MOVE BACKWARDS THRU ARRAY
        LARK     AR1, M2        ; DO M2+1 TIMES
DOUBLE: LARP     0
        LT       *, ARO        ; OLD NA22, ..., NA12
        LAC      *, ARO, S1     ; 2*OLD
        SACL     *, ARO        ; SAVE IT
        MPY      *-, AR1       ; OLD*NEW=OLD*(2*OLD)
        PAC
        BLZ      OVFL          ; --+ OR +- = -, ++ OR -- = +
        BANZ     DOUBLE
;
; MOVE NA12-NA22 TO NA1-NA11 IF NO OVERFLOW DURING PASS
;
        LARK     ARO, NA11
        LARK     AR1, NA22
SW2:    LARP     1
        LAC      *-, ARO       ; NA22, ..., NA12
        SACL     *, ARO        ; NA11, ..., NA1
        BANZ     SW2          ; BANZ DEC ARO
        B        DSTART
;
; GOOD RESULTS IN NA1-NA11 AS NO MOV IF OVERFLOW DURING LAST PASS
;
OVFL:   LARK     ARO, (2*M2+1)   ; 21 FOR M2=10
        LARK     AR1, M2        ; 10 FOR M2=10
;
; LOAD NA'S INTO FIRST REAL PART OF FFT
;
NALP:   LARP     0              ; USE ARO FIRST
        ZAC
        SACL     *-, AR1       ; 0 INTO 21, 19, 17, ..., 1
        LAC      *, ARO        ; (10) INTO 20, ..., (1) INTO 2, (0) INTO 0
        SACL     *-, AR1
        BANZ     NALP          ; AR1 DECREMENTED BY BANZ
;
; FILL REST OF MEMORY WITH ZEROES
;
        LARK     ARO, 22D       ; 22, 21, 127

```

```

        LARK    AR1,105D      ;106 ZEROS
        ZAC
;
ZACLP:  LARP    0
        SACL    *+,AR1
        BANZ    ZACLP
;
        RET
;
;
;----- MAGNTD -----
;
MAGNTD: LARK    ARO,0          ;POINTER TO INPUT
        LARK    AR1,0          ;POINTER TO OUTPUT
        LACK    NSPECT         ;NUMBER OF POINTS IN SPECTRUM
        SACL    COUNT
;
;MAGNITUDE OF SPECTRUM CALCULATION, RESULT STORED IN SCOPE BUFFER
;
STORE:  LAC     *+,ARO          ;PICK UP REAL
        ABS
        SACL    RXT             ;REAL
;
        LAC     *+,AR1          ;PICK UP IMAGINARY
        ABS
        SACL    IYT             ;IMAG
;
        SUB     RXT
        BGZ     YGX
;
XGY:    LAC     RXT              ;REAL >= IMAGINARY
        SUB     IYT,S2
        BGZ     POS1
;
        ZALH    RXT
        SUB     RXT,S13
        ADD     IYT,S15
        B       FINMAG
;
POS1:   ZALH    RXT
        ADD     IYT,S13
        B       FINMAG
;
YGX:    LAC     IYT              ;IMAGINARY > REAL
        SUB     RXT,S2
        BGZ     POS2
;
        ZALH    IYT
        SUB     IYT,S13
        ADD     RXT,S15
        B       FINMAG
;
POS2:   ZALH    IYT
        ADD     RXT,S13          ;ACC. HAS THE RESULT
;
FINMAG: SACH    RXT,S4           ;JUSTIFY IT FOR DISPLAY - SHIFT LEFT 4
        LAC     RXT,S14         ;SHIFT RIGHT 2 TOTAL
        SACH    *+,ARO
;
        LAC     COUNT
        SUB     ONE

```

```

        SACL    COUNT
        BNZ     STORE
;
        RET
;
;----- PEAK PICKER FOR PITCH -----
;
PPIC:   LARP    0
        LACK    128D          ;TOP OF BUFFER
        SACL    COUNT        ;FOR LATER
        ADD     NMIN
        SACL    DPTR         ;ADDRESS OF FIRST POINT TO SEARCH
        LAC     NMAX
        SUB     NMIN
        SACL    TEMP
        LAR     ARO,TEMP      ;LOOP COUNTER
        LAR     AR1,DPTR     ;BUFFER POINTER
        LAC     ONE,S15
        SACL    MAXVAL        ;INITIALIZE MAXVAL TO -32768
;
FPEAK:  LARP    1
        LAC     *+,ARO
        SACL    TEMP
        LAC     MAXVAL
        SUB     TEMP
        BGEZ    LOWER        ;IF (MAXVAL - TEMP) >= 0, THEN LOWER
HIGHER: LAC     TEMP
        SACL    MAXVAL        ;MAXVAL = CURRENT DATA POINT
        SAR     AR1,DPTR
LOWER:  BANZ    FPEAK         ;LOOP UNTIL DONE
;
        LAC     DPTR          ;ADDRESS OF PEAK
        SUB     COUNT         ;PUT INTO RANGE 1 ... (OUTB1)
        SUB     ONE           ;PUT INTO RANGE 0 ... (OUTB1) - 1
        RET
;
;----- NULL PICKER FOR FORMANTS -----
;
NPIC:   LACK    3D            ;NUMBER OF FORMANTS TO PICK
        SACL    COUNT
;
        ZAC                     ;ZERO PREVIOUS FORMANT NULLS
        LDPK    1
        SACL    FRMT1
        SACL    FRMT2
        SACL    FRMT3
        LDPK    0
;
        LACK    (NSPECT-3)    ;COUNTER
        SACL    SCOUNT
        LARK    AR1,1         ;POINTER TO 2 POINT IN SPECTRUM
        LARK    ARO,128D+FRMT1 ;LOCATION OF FIRST FORMANT
        LARP    1
;
CHKMIN: MAR     *+,AR1        ;"SHIFT" BUFFER (POINT TO NEW C)
        LAC     MAXVAL        ;CHECK FOR MAX POINT
        SUB     *,AR1
        BGEZ    SPICK        ;IF C > MAXVAL THEN
        LAC     *,AR1        ;MAXVAL = C

```

```

      SACL    MAXVAL
;
; LOOKING AT POINTS :  A B C (RESPECTIVELY)
;                      IF A > B < C THE B IS A MIN
;
SPICK:  LAC    *-,AR1          ;GET C (POINT TO B)
        SUB    *+,AR1          ;SUB B FROM C (POINT TO C)
        BLEZ   ENDPK           ;IF C <= B NO MIN FOUND
        MAR    *-,AR1          ; (POINT TO B)
        LAC    *-,AR1          ;GET B (POINT TO A)
        SUB    *+,AR1          ;SUB A FROM B (POINT TO B)
        MAR    *+,AR1          ; (POINT TO C)
        BGEZ   ENDPK           ;IF B >= A NO MIN FOUND
;
; WE HAVE PEAK AT B, CHECK FOR NULL CONDITION OF -2.5 dB FROM PREV MAXVAL
;
        LT     MAXVAL
        MPYK   1535D           ;CALC 0.75 OF MAXVAL
        PAC
        SACH   TEMP,X4         ;12 BIT MULT, SHIFT ACC <- X4
        LAC    TEMP,S1         ;SUB NULLVAL FROM 0.75*MAXVAL
        MAR    *-,AR1          ; (POINT TO B)
        SUB    *+,AR1          ;I.E. -2.5 dB FROM MAX VAL (POINT TO C)
        BLZ    BADPK           ;IF RESULT POS MIN IS OK:TAG IT
;
        LACKA  ASCP2           ;GET ADDRESS OF SCOPE BUFFER
        TBLR   TEMP
        LAC    TEMP
        SAR    AR1,TEMP         ;GET OFFSET
        ADD    TEMP,S1         ;ADD OFFSET*2
        LARP   0
        SACL   *+,AR1          ;SAVE POINTER TO PEAK
;
        LAC    COUNT           ;INC PEAK COUNTER
        SUB    ONE
        BLEZ   DONEPK          ;IF FOUND THREE GOOD PEAKS
        SACL   COUNT
;
BADPK:  ZAC          ;BUG OUT, ELSE
        SACL   MAXVAL   ;RESET MAXVAL, BUT CHECK IF AT END OF BUFFER
;
ENDPK:  LAC    SCOUNT
        SUB    ONE
        SACL   SCOUNT
        BNZ    CHKMIN       ;LOOP BACK
;
DONEPK: RET
;
;----- DECIS -----
;
DECIS:  LACK    IR1           ;SIGNAL RMS THRESHOLD
        SUB    IR0           ;SIGNAL RMS
        BGEZ   DE253         ;UNVOICED IF RMS THRESHOLD >= SIGNAL RMS
;
        LACK    IZM          ;ZERO CROSSINGS THRESHOLD
        SUB    NZ            ;ZERO CROSSINGS COUNT
        BLZ    DE253         ;UNVOICED IF ZC THRESHOLD < ZC COUNT
;
        CALL   PPIC          ;FIND PEAK IN RANGE OF NMIN TO NMAX
        SACL   L             ;SAVE PEAK LOCATION

```

```

        LARP      0
;
; TEST FOR POSSIBLE DOUBLING
;
        LAC      L,S14
        SACH     TEMP          ;TEMP = L/4
        LAC      L
        SUB      TEMP
        SACL     NMAX          ;NMAX = L - L/4
        SUB      NMIN
        BLZ      DE226         ;SKIP TEST IF NMAX < NMIN
        CALL     PPIC          ;FIND PEAK ;N RANGE OF NMIN TO NMAX
        SACL     LL            ;SAVE PEAK LOCATION
;
        LACK     128D          ;ADDRESS OF START OF SCOPE BUFFER
        ADD      L
        SACL     DPTR
        LAR      ARO,DPTR      ;SCOPE(L)
        LT       *,ARO
        LACK     128D
        ADD      LL
        SACL     DPTR
        LAR      ARO,DPTR      ;SCOPE(LL)
        LAC      *,S12,ARO
        MPYK     -3481D        ;-0.85
        APAC
        BLEZ     DE226         ;NO DOUBLING IF SCOPE(LL) <= 0.85*SCOPE(L)
        LAC      LL
        SACL     L             ;DOUBLING, SO TAKE PEAK AT LL
;
;PARABOLIC INTERPOLATION TO FIND HEIGHT AND POSITION OF 'TRUE' PEAK
;
DE226:  LACK     128D          ;ADDRESS OF START OF SCOPE BUFFER
        ADD      L
        SACL     DPTR
        LAR      ARO,DPTR
        LAC      *-,ARO
        SACL     SL            ;SCOPE(L), POINT TO L-1
        LAC      *+,ARO
        SACL     SLM1          ;SCOPE(L-1), POINT TO L
        MAR      *+,ARO        ;POINT TO L+1
        LAC      *,ARO
        SACL     SLP1          ;SCOPE(L+1)
;
        LAC      SL
        SUB      SLM1
        BLEZ     DE253         ;UNVOICED IF SCOPE(L) <= SCOPE(L-1)
        LAC      SL
        SUB      SLP1
        BLEZ     DE253         ;UNVOICED IF SCOPE(L) <= SCOPE(L+1)
;
        LAC      SL,S15        ;-AA = (2*SCOPE(L) - SCOPE(L-1) -
        SUB      SLM1,S14      ;      SCOPE(L+1))/2
        SUB      SLP1,S14
        SACH     DR,S1         ;-AA > 0 ALWAYS
;
        LAC      SLP1,S14
        SUB      SLM1,S14
        SACH     BB            ;BB = (SCOPE(L+1) - SCOPE(L-1))/4
        LT       BB
        MPY      BB

```

```

PAC                                ;BB*BB
CALL DIVNEW
SACL TEMP                            ;BB*BB/AA = -(BB*BB/(-AA))
LAC SL
SUB TEMP
SACL AP                            ;AP = SCOPE(L) - BB*BB/AA
;
LAC BB,S7                          ;BB*(2**7)
CALL DIVNEW
SACL TEMP                            ;BB*(2**7)/AA = -(BB*(2**7)/(-AA))
LAC L,S7                            ;L*(2**7)
SUB TEMP
SACL ALI                            ;ALI = (L - BB/AA)*(2**7)
;
LAR ARO,DPTR
LAC *,ARO
LT L
MPYK -300
APAC
SACL TEMP                            ;SCOPE(0) - 300*L
LT TEMP
LAC AP,S12
MPYK -1638                          ;-0.4
APAC
BGEZ DE820                          ;PEAK OK IF AP >= 0.4*(SCOPE(0) - 300*L)
LAC N1
BZ DE253                            ;IF N1 = 0, THEN UNVOICED
LAC AP,S12
MPYK -1229                          ;-0.3
APAC
BGEZ DE820                          ;PEAK OK IF AP >= 0.3*(SCOPE(0) - 300*L)
;
DE253: ZAC
SACL NO                            ;MARK AS UNVOICED
B DE250
;
DE820: LAC ONE,S6                    ;0.5*(2**7) = 2**6
LT ALI                            ;(L - BB/AA)*(2**7)
MPYK 3
APAC                                ;((L - BB/AA)*3 + 0.5)*(2**7)
SACL TEMP
LAC TEMP,S9                        ;((L - BB/AA)*3 + 0.5)*(2**16)
SACH NO                            ;NO = TRUNC((L - BB/AA)*3 + 0.5)
;
DE250: LAC N1,S12
SUB N3,S12
ABS
LT N3
MPYK -1638                          ;-0.4
APAC
BGZ DE999                          ;ABS(N1 - N3) - 0.4*N3 > 0
LAC N1,S15
ADD N3,S15
SACH N2                            ;N2 = (N1 + N3)/2
;
;IF N3 = 0, N2 <> 0, & NO AND N1 ARE CLOSE, EXTRAPOLATE FOR FIRST VOICED
;TERM USING LINEAR EXTRAPOLATION
;
DE999: LAC N3
BNZ DE891                          ;IF N3 <> 0 THEN NO EXTRAPOLATION
LAC N2

```



```

      BZ      DE891      ;IF N2 = 0 THEN NO EXTRAPOLATION
      LAC     NO,S12
      SUB     N1,S12
      ABS
      LT      N1
      MPYK    -819      ;-0.2
      APAC
      BGZ     DE891      ;IF ABS(NO - N1) > 0.2*N1 THEN NO EXTRAPOLATION
      LAC     N1,S1
      SUB     NO
      SACL    N2      ;N2 = 2*N1 - NO (LINEAR EXTRAPOLATION)
;
;UPDATE FRAMES
;
DE891: DMOV   N2      ;N3 = N2
      DMOV   N1      ;N2 = N1
      DMOV   NO      ;N1 = NO
;
;DETERMINE LIMITS OF NEXT FRAME IF PRESENT FRAME IS VOICED
;
      LACK   JMI
      SACL   NMIN      ;NMIN = JMI
      LACK   JMA
      SACL   NMAX      ;NMAX = JMA
      LAC    NO
      BZ     DE888      ;IF PRESENT FRAME UNVOICED, THEN LIMITS SET
;
      LT     ONE
      LAC    L
      MPYK   -7
      APAC
      SACL   NMIN      ;NMIN = L - 7
      MPYK   -JMI
      APAC
      BGEZ   DE886      ;IF NMIN >= JMI THEN OK
      LACK   JMI
      SACL   NMIN      ;OTHERWISE NMIN = JMI
;
DE886: LACK   7
      ADD    L
      SACL   NMAX      ;NMAX = L + 7
      MPYK   -JMA
      APAC
      BLE2   DE888      ;IF NMAX <= JMA THEN OK
      LACK   JMA
      SACL   NMAX      ;OTHERWISE NMAX = JMA
;
DE888: LAC    N3      ;CURRENT PITCH
      BNZ    VOICED    ;IF NOT ZERO, THEN VOICED
      LACK   0      ; ELSE PITCH=0 FOR INPUT
      B      STRPTC    ;STORE PITCH IN DRAM
;
VOICED: LACK   3
      SACL   DR      ;FOR DISPLAY, DIVIDE PITCH BY 3
      LAC    N3
      CALL   DIVNEW
      SACL   TEMP      ;TEMP = -(N3/3)
      LACKA  ASCP1      ;GET ADDRESS OF TOP OF SCOPE BUFFER
      TBLR   DPTR
      LAC    DPTR
      SUB     TEMP      ;PITCH = SCOPE1 + N3/3

```

```

      ADD      ONE      ;FOR DISPLAY PURPOSES
;
STRPTC: SACL   PITCH    ;STORE IN PRAM - FFT WIPES OUT DRAM
      RET
;
;
;----- GETDAT -----
;      FILTER, DOWNSAMPLE, DECIMATE BY THREE, WINDOW
;
GETDAT: LACKA   AWIND1
      TBLR     WPTR      ;POINTER TO WINDOW
;
      LACKA   ACNVRT     ;GET STARTING ADDRESS OF CONVERSION TABLE
      TBLR     TOP
;
      LACKA   ATABLE     ;READ IN FILTER COEFFICIENTS
      TBLR     TEMP
      LAC      TEMP
      TBLR     FC1
      ADD      ONE
      TBLR     FC2
      ADD      ONE
      TBLR     FC3
      ADD      ONE
      TBLR     FC4
      ADD      ONE
      TBLR     FC5
      ADD      ONE
      TBLR     FC6
      ADD      ONE
      TBLR     FC7
      ADD      ONE
      TBLR     FC8
;
      ZAC      ;ZERO FILTER
      SACL     Y1
      SACL     Y11
      SACL     Y2
      SACL     Y21
      SACL     Y22
      SACL     Y3
      SACL     Y31
      SACL     Y32
      SACL     XLOW
      SACL     XHIGH
;
      LACK     W1/2      ;COUNT
      SACL     COUNT
      LARK     ARI,INBUFF+128D ;ADDRESS OF INPUT BUFFER ON PAGE 1
      LARK     ARO,0D     ;OUTPUT
      LARP     1
;
; CONVERT U-LAW TO LINEAR, FILTER, DOWNSAMPLE, AND WINDOW
; DO FIRST 2 POINTS SEPERATELY SO THAT ISR CAN BE STARTED
;
      LAC      *.S8,ARO   ;GET HIGH BYTE FIRST
      SACH     TEMP
      LACK     07FH       ;GET INDEX IN CONVERSION TABLE, WHICH IS
      AND      TEMP       ; THE MAGNITUDE OF THE HIGH BYTE
      ADD      TOP        ;ADD STARTING ADDRESS OF CONVERSION TABLE
      TBLR     SIG        ;READ IN CONVERTED POSITIVE SAMPLE

```

```

        LACK    80H
        AND     TEMP          ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F1B
        ZAC     SIG          ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F1B:    CALL    FILTER        ;FILTER 1ST POINT
;
        LAC     WPTR          ;GET WINDOW SAMPLE
        TBLR    WIND
        ADD     ONE
        SACL    WPTR
        LT      WIND
        MPY     SIG          ;SIG NOW WINDOWED
        PAC
        SACH    *,X1,ARG
;
        ZALS    XLOW          ;RUNNING SUM TO BE USED LATER
        ADDH    XHIGH
        ADD     *+,AR1
        SACL    XLOW
        SACH    XHIGH
;
        LACK    7FH          ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     *,AR1        ; THE MAGNITUDE OF THE LOW BYTE
        ADD     TOP          ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG          ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     *~,AR1       ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F2B
        ZAC     SIG          ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F2B:    CALL    FILTER        ;FILTER 2ND POINT
;
; START DATA COLLECTION '!!!!
;
        LDPK     1
        LACK     (INBUFF+128D) ;INITIALIZE INPUT BUFFER POINTER/COUNTER
        SACL     ISRARG       ; WHICH STARTS ISR COLLECTING DATA
        LDPK     0
;
        B        ENTER       ;BRANCH INTO MIDDLE OF LOOP
;
; LOOPING STARTS HERE
;
FLOOP:  LAC      *,S8,ARG     ;GET HIGH BYTE FIRST
        SACH     TEMP
        LACK     07FH        ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND      TEMP        ; THE MAGNITUDE OF THE HIGH BYTE
        ADD      TOP          ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR     SIG          ;READ IN CONVERTED POSITIVE SAMPLE
        LACK     80H
        AND      TEMP        ;WAS IT A NEGATIVE SAMPLE ?
        BZ       F1
        ZAC      SIG          ;YES THEN NEGATE CONVERTED SAMPLE
        SUB      SIG
        SACL     SIG
F1:     CALL     FILTER        ;FILTER 1ST POINT
;
        LAC      WPTR        ;GET WINDOW SAMPLE

```

```

        TBLR    WIND
        ADD     ONE
        SACL    WPTR
        LT      WIND
        MPY     SIG                ;SIG NOW WINDOWED
        PAC
        SACH    *,X1,AR0
;
        ZALS    XLOW                ;RUNNING SUM TO BE USED LATER
        ADDH    XHIGH
        ADD     *+,AR1
        SACL    XLOW
        SACH    XHIGH
;
        LACK    7FH                ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     *,AR1              ; THE MAGNITUDE OF THE LOW BYTE
        ADD     TOP                ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG                ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     *-,AR1            ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F2
        ZAC     SIG                ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F2:      CALL    FILTER            ;FILTER 2ND POINT
;
ENTER:   LAC     *,S0,AR1          ;GET HIGH BYTE FIRST
        SACH    TEMP
        LACK    07FH              ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     TEMP              ; THE MAGNITUDE OF THE HIGH BYTE
        ADD     TOP                ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG                ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     TEMP              ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F3
        ZAC     SIG                ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F3:      CALL    FILTER            ;FILTER 3RD POINT
;
        LACK    7FH                ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     *,AR1              ; THE MAGNITUDE OF THE LOW BYTE
        ADD     TOP                ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG                ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     *-,AR0            ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F4
        ZAC     SIG                ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F4:      CALL    FILTER            ;FILTER 1ST POINT
;
        LAC     WPTR              ;GET WINDOW SAMPLE
        TBLR    WIND
        ADD     ONE
        SACL    WPTR
        LT      WIND
        MPY     SIG                ;SIG NOW WINDOWED
        PAC
        SACH    *,S1,AR0
;

```

```

        ZALS    XLOW          ;RUNNING SUM TO BE USED LATER
        ADDH    XHIGH
        ADD     *+,AR1
        SACL    XLOW
        SACH    XHIGH
;
        LAC     *,S8,AR1      ;GET HIGH BYTE FIRST
        SACH    TEMP
        LACK    07FH          ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     TEMP          ; THE MAGNITUDE OF THE HIGH BYTE
        ADD     TOP           ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG           ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     TEMP          ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F5
        ZAC     TEMP          ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F5:     CALL    FILTER         ;FILTER 2ND POINT
;
        LACK    7FH          ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     *,AR1         ; THE MAGNITUDE OF THE LOW BYTE
        ADD     TOP           ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG           ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     *-,AR1        ;WAS IT A NEGATIVE SAMPLE ?
        BZ      F6
        ZAC     TEMP          ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
F6:     CALL    FILTER         ;FILTER 3RD POINT
;
        LAC     COUNT
        SUB     ONE
        SACL    COUNT
        BNZ     FLOOP
;
        RET
;
;
;----- CONVERT SIGNAL TO ZERO MEAN -----
;
ZMEAN:  LACK    OUTB1
        SACL    DR
        ZALS    XLOW
        ADDH    XHIGH
        CALL    DIVNEW        ;MEAN = (RUNNING SUM) / (NUMBER OF POINTS)
        SACL    TEMP
        ZAC     TEMP
        SUB     TEMP
        SACL    TEMP          ;SAVE MEAN
;
        LARK    ARO,(OUTB1-1) ;LOOP THROUGH OUTPUT BUFFER
        LARP    0D           ;USE PAGE 0
;
MLOOP:  LAC     *,ARO
        SUB     TEMP          ;SUBTRACT MEAN
        SACL    *,ARO
        BANZ    MLOOP
;
        RET

```

```

;
;
;----- PITCH'S AUTOCORRELATION -----
;
AUTO1:  SACL    TEMP            ;NUMBER OF LAGS
;
;       LACK    128D
;       SACL    SPTR2
;
;       LAC     ONE,S14         ;INITIALIZE MASK
;       SACL    ZMASK
;       ZAC     ;INITIALIZE NORMALIZATION LOOP COUNT
;       SACL    SCOUNT
;
; COMPUTE SUM OF SQUARES OF POINTS - WHICH GIVES R[0] AND ENERGY
;
;       LARK    AR1,(OUTB1-1)   ;76 POINTS
;       ZAC     ;ACC := 0
;       MPYK    0               ;P := 0
;
;       LARP    1
SQU:    LTA     *,AR1           ;S(I)
;       MPY     *,AR1           ;S(I)
;       BANZ    SQU
;       APAC
;
;       SACL    XLOW            ;SAVE SUM OF SQUARES
;       SACH    XHIGH
;       SACL    ENRLO           ;ENERGY USED LATER BY DECIS
;       SACH    ENRHI
;
; NORMALIZE R[0] TO CALCULATE SCOUNT FOR LATER USE IN NORMALIZATION
;
;       BNZ     GT0             ;R(0) = 0?
;       LACK    1
;       SACL    XLOW
GT0:    LAC     XHIGH           ;ALREADY NORMALIZED?
;       AND     ZMASK
;       BNZ     NFIN           ;NORM FIN IF R(0) = 01XXXXX
;
NORM00: LAC     ONE             ;INCRIMENT NORMALIZATION LOOP COUNT
;       ADD     SCOUNT
;       SACL    SCOUNT
;
;       ZALS    XLOW
;       ADDH    XHIGH
;       ADDH    XHIGH           ;DOUBLE IT
;       ADDS    XLOW
;       SACL    XLOW           ;SAVE IT
;       SACH    XHIGH
;
;       LAC     XHIGH
;       AND     ZMASK
;       BZ     NORM00          ;KEEP ON GOING
;
; DETERMINE ZMASK TO ADJUST NORMALIZED RESULTS FOR DISPLAY
;
NFIN:   LAC     XHIGH
;       SACL    DR
;       LAC     ONE,S15
;       SUB     ONE            ; 32767

```

```

        SACL    ZMASK
        LAC     ZMASK,S14
        CALL    DIVNEW
        SACL    ZMASK          ;ZMASK = [32767*2**14/R(0)]
        ZAC
        SUB     ZMASK
        SACL    ZMASK
;
        LT      XHIGH
        MPY     ZMASK
        PAC
        SACH    XHIGH,S1      ;R(0*)[ ]/2**15 = 32767/2 = 16383 = CONSTANT
;
        LAR     ARO,SPTR2      ;SAVE THE FIRST POINT
        LARP    0
        LAC     XHIGH
        SACL    **
        SAR     ARO,SPTR2
;
; CALCULATE POWERS TO BE USED LATER FOR NORMALIZATION
;
        LAC     SCOUNT
        BZ      CONT5          ;SCOUNT = 0 -> NO NORMALIZATION REQUIRED
        SUB     ONE,S4         ;ACC = (SCOUNT-16)
        BGZ     PWR2          ;SCOUNT > 16 -> POWER=2**(SCOUNT-16)
        BZ      CONT5          ;SCOUNT = 16 -> NO POWER REQUIRED
                                ;SCOUNT < 16 -> POWER=2**(SCOUNT)
;
PWR1:   LAC     SCOUNT          ;I = SCOUNT
        B       PWR
;
PWR2:   LAC     SCOUNT
        SUB     ONE,S4         ;I = SCOUNT-16
;
; POWER = 2**I AND QTEMP = 2**I - 1
;
PWR:    SUB     ONE            ;I=I-1 FOR BANZ
        SACL    TEMPN
        LAR     ARI,TEMPN
        LARP    1
        LAC     ONE
        SACI    POWER          ;ACC := 1
PLOOP:  ADD     POWER          ;DOUBLE 2 I TIMES
        SACL    POWER
        BANZ    PLOOP
        SUB     ONE            ;ACC := 2**I - 1
        SACL    QTEMP          ;CREATE MASK (ALL 1'S)
;
; OUTER LOOP FOR NUMBER OF LAGS IN TEMP
;
CONT5:  LACK    (OUTB1-2)
        SACL    DELTA          ;1'ST IS DONE
;
        LACKA   ATARG          ;BASE ADDRESS FOR CALA
        TBLR    BASE
;
OUTER:  LARK    ARI,(OUTB1-1)   ;INDEX THROUGH DATA POINTS
        LAR     ARO,DELTA      ;DELTA = 80 FIRST TIME AROUND
;
        MPYK    0              ;P := 0
        LARP    0              ;USE ARO FIRST

```

```

;
; SPECIAL FAST ALGORITHM FOR INNER LOOP SUMMATION.
; MUST DETERMINE WHERE TO ENTER IN-LINE CODE
;
      LAC      ONE          ;CALCULATE WHERE TO START LOOPING
      ADD      DELTA
      SACL     DP1          ;DP1 := DELTA + 1 = NUMBER OF LOOPS
      LAC      DP1,S13
      SACH     TEMPN        ;(DP1 MOD 8)
      LAC      TEMPN,S4     ;16 * (DP1 MOD 8)
      SUB      DP1,S1       ;OFFSET := -2[DP1-8*(DP1 MOD 8)]
                          ;:= -2[REM OF DP1/8]
;
      BNZ      REM          ;IF [REM = 0] THEN OFFSET = -16
      ZAC
      SUB      ONE,S4       ;OFFSET = -16
REM:   ADD      BASE        ;ADD OFFSET TO BASE ADDRESS (TARGET)
      SACL     TEMPN        ;YOU CAN'T ZERO ACCUM. SO SAVE
      CALA     CALA         ;JUMP TO INDEXED ADDRESS
;
; INNER MULT./ACC. LOOP - IN-LINE CODE USED FOR SPEED.
;
AUTLP:
      REPT     7
      LTA      *-,AR1
      MPY      *-,ARO
      ENDM
      LTA      *,AR1        ;DON'T DECREMENT ARO
      MPY      *-,ARO
TARGET: BANZ    AUTLP        ;BANZ DOES FINAL DECREMENT ON ARO
      APAC     APAC         ;FINAL SUM
;
      SUB      TEMPN        ;ADJUST THE ACCUMULATOR BY PREVIOUS CONTENTS
      SACL     XLOW         ;STORE RESULTS
      SACH     XHIGH
      POP
                          ;CLEAN UP THE STACK FROM CALA
;
; NORMALIZE THE RESULTS (A SPECIAL FAST ALGORITHM)
;
      LAC      SCOUNT      ;GET NORMALIZATION COUNT
      BZ       NORM0        ;ALREADY NORMALIZED.
      SUB      ONE,S4       ;SHIFT = SCOUNT - 16
      BGZ      NORM2        ;SHIFT > 16
      BZ       NORM3        ;SHIFT = 16
      ADD      ONE
      BZ       NORM4        ;SHIFT = 15      (SPECIAL CASE)
;
; CASE 1: 1 <= SHIFT COUNT <= 14
;
NORM1: LT       POWER        ;GET 2**I IN T REG
      MPY      XLOW          ;LOW (SIGN EXTENSION!) * 2**I
      PAC
      SACH     TEMPN         ;/2**16
      ZALS     TEMPN         ;LOW * 2**(I-16)
      AND      QTEMP         ;ZAP SIGN BIT EXTENSION IN LOW WORD
      MPY      XHIGH         ;HIGH * 2**I
      APAC     APAC          ;+ LOW * 2**(I-16)
      SACL     XHIGH
      B        NORM0
;
; CASE 2: SHIFT COUNT > 16 (COMPUTE XLOW * 2**(SCOUNT-16))
;

```



```

NORM2: LT      POWER      ;GET 2**I IN T REG
      MPY      XLOW       ;XLOW * 2**(SCOUNT-16)
      PAC
      SACL     XHIGH
      B        NORM0
;
; CASE 3:  SHIFT COUNT = 16  (TRANSFER LOW BYTE TO HIGH BYTE)
;
NORM3: LAC      XLOW       ;GET LOW
      SACL     XHIGH      ;INTO HIGH
      B        NORM0
;
; CASE 4:  SHIFT COUNT = 15  (SPECIAL CASE)
;
NORM4: LAR      AR0,SCOUNT
      ZAC
NLP2:  ADDH     XHIGH
      ADDS     XLOW
      SACL     XLOW
      SACH     XHIGH
      BANZ     NLP2
;
; CASE 0:  ALREADY NORMALIZED  (ADJUST RESULTS FOR DISPLAY)
;
NORM0: LT      XHIGH      ; R(I)/2 < R(I)*[ ]/2**15 < R(I)
      MPY      ZMASK      ;AS 2**14 < [ ] < 2**15-1
      PAC
      SACH     XHIGH,S1
;
      LAR      AR0,SPTR2
      LARP     0
      LAC      XHIGH
      SACL     *+,AR0
      SAR      AR0,SPTR2
;
      LAC      DELTA      ;DECRIEMENT LAG POINTER
      SUB      ONE
      SACL     DELTA
;
      LAC      TEMP      ;DECRIEMENT LAG COUNT
      SUB      ONE
      SACL     TEMP
;
      BNZ      OUTER
      RET
;
;
;----- ENV'S AUTOCORRELATION -----
;
AUTO2: LACK     NR/2
      SACL     NRD      ;NUMBER OF POINTS IN EACH BUFFER/2 = 25
;
      LACK     IT
      SACL     NRCNT     ;NUMBER OF BUFFERS TO DO = [N / NR] = 4
;
;R(DELTA) := 0, DELTA = 0, ... ,M2
;
      ZAC      ;ZERO THE ACCUMULATOR
      LARK     AR0,LOW0
      LARK     AR1,(2*M2)+1
;

```

```

AUTOL:  LARP      0                ;ZERO R(DELTA)'S
        SACL      *+,AR1
        BANZ      AUTOL
;
;POINT TO BUFFERSTART
;
        LACK      INBUFF-M2/2+129D
        SACL      DPTR                ;INPUT DATA POINTER
;
        LACKA     AWIND?
        TBLR      WPTR                ;POINTER TO WINDOW
        LACK      M2-1                ; M2-1
        ADD       NRD,S1              ; + NUMBER OF POINTS IN THE BUFFER
        ADD       WPTR                ; + START OF WINDOW TABLE
        SACL      WPTR                ; = POINTER TO END OF FIRST BUFFER IN PRAM
;
;AUTOCORRELATION ALGORITHM
;
;FOR I = 1 TO IT (WHERE IT = N/NR, AND NR = NUMBER OF POINTS PER ITERATION)
;  FOR DELTA = 0 TO M2 (WHERE M2 = NUMBER OF LAGS)
;    R(DELTA) = R(DELTA) + SUM(I), WHERE SUM(I) = SUM(X(N)*X(N+DELTA))
;    NEXT DELTA
;  NEXT I
;
;FOR I = 1 TO IT, ** READ IN A BUFFERFUL OF POINTS **
;
        LARP      0                ;USE ARO
AGAIN:   LARK      ARO,(BEND-1)      ;OUTPUT BUFFER (READ IN BACKWARDS)
        LAC       DPTR              ;END OF DATA IN TABLE MEMORY
        SUB       NRD                ;NEXT BUFFER
        SACL      DPTR
        LAR       AR1,DPTR          ;INPUT BUFFER
;
;NOTE: REDUCE AVERAGE READ TIME TO 4.4 FROM 6 CYCLES VIA INLINE CODE
;
WLOOP:   LARP      1
        LACKA     ACNVRT            ;GET STARTING ADDRESS OF CONVERSION TABLE
        TBLR      SIG
        LACK      7FH              ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND       *,AR1            ; THE MAGNITUDE OF THE LOW BYTE
        ADD       SIG              ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR      SIG              ;READ IN CONVERTED POSITIVE SAMPLE
        LACK      80H
        AND       *,ARO            ;WAS IT A NEGATIVE SAMPLE ?
        BZ        CONT8
        ZAC       ;YES THEN NEGATE CONVERTED SAMPLE
        SUB       SIG
        SACL      SIG
CONT8:   LAC       WPTR              ;WINDOW THE SAMPLE
        TBLR      WIND
        SUB       ONE              ;INCRIMENT WINDOW POINTER
        SACL      WPTR
        LT        WIND
        MPY       SIG
        PAC
        SACH      *-,AR1
;
        LAC       *+,S8,ARO        ;GET HIGH BYTE SECOND (READ IN BACKWARDS)
        SACH      TEMP
        LACKA     ACNVRT            ;GET STARTING ADDRESS OF CONVERSION TABLE
        TBLR      SIG

```

```

        LACK    07FH          ;GET INDEX IN CONVERSION TABLE, WHICH IS
        AND     TEMP          ;GET THE MAGNITUDE OF THE HIGH BYTE
        ADD     SIG           ;ADD STARTING ADDRESS OF CONVERSION TABLE
        TBLR    SIG           ;READ IN CONVERTED POSITIVE SAMPLE
        LACK    80H
        AND     TEMP          ;WAS IT A NEGATIVE SAMPLE ?
        BZ      CONT7
        ZAC     ZAC           ;YES THEN NEGATE CONVERTED SAMPLE
        SUB     SIG
        SACL    SIG
CONT7:   LAC     WPTR          ;WINDOW THE SAMPLE
        TBLR    WIND
        SUB     ONE           ;INCRIMENT WINDOW POINTER
        SACL    WPTR
        LT      WIND
        MPY     SIG
        PAC
        SACH    *,ARO
;
        BANZ    WLOOP         ;LOOP FOR ALL SAMPLES COLLECTED
;
;SETUP FOR INNER LOOP
;
        ZAC
        SACL    DELTA         ;DELTA := 0
        LACK    M2+1
        SACL    LAGCNT        ;LAGCOUNT := M2+1
;
        LARK    ARO,LOW0      ;POINT TO R(0)
        LARP    0
;
;DO ALL LAGS (0 TO M2) FOR THIS BUFFER, THEN DO NEXT BUFFER.
;
LAG:     ZALS    **+,ARO       ;LOAD LOW R(DELTA)
        ADDH    **-,ARO       ;LOAD HIGH R(DELTA)
        SAR     ARO,TEMP      ;SAVE ARO
;
;R(DELTA) := R(DELTA) + SUM[X(N)*X(N+DELTA)]
;
        LARK    ARO,0         ;STARTS AT ZERO
        LAR     AR1,DELTA     ;STARTS AT DELTA, DELTA = 0, ... ,M2
;
        LT      **+,AR1
        MPY     **+,ARO
        REPT    NR-1
        LTA     **+,AR1
        MPY     **+,ARO
        ENDM
        APAC                ;FINAL ADD
;
        LAR     ARO,TEMP      ;RESTORE ARO
        SACL    **+,ARO       ;SAVE LOW R(DELTA)
        SACH    **+,ARO       ;SAVE HIGH R(DELTA) AND ON TO NEXT
;
        LAC     DELTA
        ADD     ONE           ;DELTA := DELTA + 1
        SACL    DELTA
;
        LAC     LAGCNT
        SUB     ONE           ;DECREMENT LAG COUNT
        SACL    LAGCNT

```

```

        BGZ      LAG              ;ONTO NEXT LAG FOR THIS BUFFERFUL
;
        LACK     M2-1             ;UPDATE WINDOW POINTER
        ADD      NRD,S2
        ADD      WPTR
        SACL     WPTR
;
        LAC      NRCNT
        SUB      ONE              ;DECREMENT BUFFER COUNT
        SACL     NRCNT
        BGZ      AGAIN           ;OUTER LOOP, PER BUFFER
        RET
;
;----- R(I)'S COMPUTED. NOW HAVE TO NORMALIZE THE THEM -----
;
NORMAL: LARK     AR0,LOW0         ;NORMALIZE THE RX'S
        LARK     AR1,STR         ;SPECIAL STORAGE LOCATION
        LACK     M2              ;COUNTER
        SACL     TEMP
        LAC      ONE,S14         ;ACC := 1
        SACL     ZMASK           ;MASK := 010000..00
        ZAC      0               ;ACC := 0
        SACL     SCOUNT          ;SCOUNT := 0
        LARP     0               ;USE AR0 FIRST
;
        MAR      *+,AR0          ;SKIP LOW
        LAC      *-,AR0          ;GET HIGH, POINTERS BACK
        AND      ZMASK
        BZ       NORMM
        RET                    ;NO NORM NEEDED
;
;DOUBLE R(0) UNTIL ACC = 01XXXXXXX.XXXXX (32 BITS)
;
NORMM:  LACK     1                ;| INCREMENT SHIFT COUNTER
        ADD      SCOUNT          ;| SCOUNT := SCOUNT + 1
        SACL     SCOUNT          ;| AND STORE IT.
        ZALS     *+,AR0          ;ACC := R(0)
        ADDH     *-,AR0
        ADDS     *+,AR0          ;ACC := ACC + R(0) = 2*R(0)
        ADDH     *-,AR0
        SACL     *+,AR0          ;R(0) := 2*R(0)
        SACH     *-,AR0
;
        LAC      *-,AR0          ;LOAD HIGH R(0) TO TEST, POINT TO LOW0
        AND      ZMASK           ;IS BIT 15 1?
        BZ       NORMM          ;NO. DOUBLE R(0) AGAIN
;
;NOW DO OTHERS R(I), I = 1,...,M2
;
        MAR      *+,AR0          ;SKIP OVER R(0) TO POINT TO R(1)
        LAC      *+,AR1          ;SAVE HIGH IN SPECIAL LOCATION FOR LATER
        SACL     *+,AR0
;
;NORMALIZATION
;
NORMS:  LAC      SCOUNT          ;GET SHIFT COUNT
        SACL     COUNT          ;COUNT := SCOUNT
NORM:   ZALS     *+,AR0          ;LOAD R(I) LOW
        ADDH     *-,AR0          ;AND R(I) HIGH
        ADDS     *+,AR0          ;ADD R(I) LOW

```

```

        ADDH    *-,ARO          ; ADD R(I) HIGH [DOUBLE R(I)!!]
        SACL    *+,ARO          ; STORE R(I) LOW
        SACH    *-,ARO          ; AND R(I) HIGH
;
        LAC     COUNT           ;|
        SUB     ONE             ;COUNT := COUNT - 1
        SACL    COUNT           ;|
        BNZ     NORM            ;UNTIL COUNT = 0
;
        MAR     *+,ARO          ;SKIP OVER R(I) PAIR JUST DONE
        LAC     *+,AR1          ;SAVE HIGH IN SPECIAL LOCATION FOR LATER
        SACL    *+,ARO
;
        LAC     TEMP
        SUB     ONE
        SACL    TEMP
        BNZ     NORMS          ;AND DO NEXT R(I) UNTIL M2 DONE
;
        RET
;
;
;----- THE ISR ROUTINE -----
;
INPUT:  SST     ISTATE          ;STORE PRE-ISR STATE
        LDPK    1
        SACL    ILOW
        SACH    IHIGH
;
; INPUT A SAMPLE, OR NOT?  ISRAR0 IS COUNTER/POINTER.  DOES NOT USE LOC 0.
;
        LACK    128D           ;BUFFER FULL WHEN COUNTER\POINTER
        SUB     ISRAR0         ; = PAGE 1 OFFSET
        BZ      SOUT
;
        SAR     AR0,PREAR0      ;SAVE PRE-ISR AR0
        LAR     AR0,ISRAR0      ;LOAD ISR'S AR0
        LARP    0
;
        IN      ISMPL,PA1       ;GET MU-LAW SAMPLE
        LACK    OFFH           ;ONE'S COMPLIMENT INPUT SAMPLE
        XOR     ISMPL
        SACL    ISMPL
;
; STORE 2 SAMPLES PER WORD, FIRST IN HIGH BYTE, SECOND IN LOW BYTE
;
        LAC     IONE
        XOR     IFLAG          ;TOGGLE FLAG
        SACL    IFLAG
        BZ      ILWBYT        ;IF ZERO THEN INSERT LOW BYTE
;
IHGHPT: LAC     ISMPL           ;WRITE HIGH BYTE ONLY TO INPUT BUFFER AND
        SACL    *              ; DON'T INCRIMENT INPUT POINTER
        B       RSTORE
;
ILWBYT: LACK    OFFH           ;MASK OFF GARBAGE FROM LOW BYTE
        AND     ISMPL
        ADD     *,S8           ;SHIFT HIGH BYTE OVER AND ADD TO LOW BYTE
        SACL    *-            ;DECRIMENT COUNTER AND POINTER
;
RSTORE: SAR     AR0,ISRAR0      ;SAVE ISR'S AR0
        LAR     AR0,PREAR0      ;LOAD PRE-ISR AR0

```

```

;
; GET RESULTS FOR SCOPE DISPLAY
;
SOUT:  LAC      CSCOPE      ;DECRIMENT COUNTER
        SUB      IONE
        SACL     CSCOPE
        BNZ      CONT      ;ZERO YET?
;
        LACKA    ASCP1      ;IF YES THEN UPDATE COUNTER AND POINTER
        TBLR     PSCOPE
        LACK     (OUTB1+OUTB2)
        SACL     CSCOPE
;
CONT:   LAC      PSCOPE      ;IS SCOPE POINTER AT THE PITCH LOCATION ?
        LDPK     0
        SUB      PITCH
        LDPK     1
        BZ       FLGUP
;
        LAC      PSCOPE      ;IS SCOPE POINTER AT THE FIRST FORMANT ?
        SUB      FRMT1
        BZ       FLGUP
        LAC      PSCOPE      ;IS SCOPE POINTER AT THE SECOND FORMANT ?
        SUB      FRMT2
        BZ       FLGUP
        LAC      PSCOPE      ;IS SCOPE POINTER AT THE THIRD FORMANT ?
        SUB      FRMT3
        BZ       FLGUP
;
FLGDN:  LACKA    AFLGDN      ;SEND THE XF LATCH LOW
        B        INC
FLGUP:  LACKA    AFLGUP      ;SEND THE XF LATCH HIGH
INC:    TBLR     ISMPL
        OUT      ISMPL,PA0
;
        LAC      PSCOPE      ;GET OUTPUT SAMPLE
        TBLR     ISMPL
        ADD      IONE
        SACL     PSCOPE
;
; CONVERT SCOPE BUFFER'S 2'S COMPLEMENT NUMBER TO SIGN MAGNITUDE TO MU-LAW
;
        LAC      ISMPL      ;POSITIVE OR NEGATIVE?
        BLZ      NEGOUT
;
POSOUT: LAC      ISMPL,S14    ;ISMPL IS A 16-BIT 2'S COMPL. NUMBER
        SACH     ISMPL      ;CONVERT TO A 14-BIT 2'S COMPL. NUMBER
        LACK     BIAS       ;ADD MU-LAW BIAS OF 33D
        ADD      ISMPL
        B        DONE
;
NEGOUT: ABS                      ;TAKE MAGNITUDE
        SACL     ISMPL
        LAC      ISMPL,S14    ;ISMPL IS A 16-BIT 2'S COMPL. NUMBER
        SACH     ISMPL      ;CONVERT TO A 14-BIT 2'S COMPL. NUMBER
        LACK     BIAS       ;ADD MU-LAW BIAS OF 33D
        ADD      ISMPL
        ADD      IONE,S15     ;ADD SIGN
;
DONE:   SACL     ISMPL      ;OUTPUT SCOPE INFO. TO CODEC
        OUT      ISMPL,PA1

```

```

;
ZALS      ILOW      ;RETURN PRE-ISR STATE
ADDH      IHIGH
LST       ISTATE
;
EINT      ;ENABLE INTERRUPTS
RET       ;RETURN
;
;
----- LOW PASS FILTER -----
;
;          5TH ORDER BI-QUAD FIR LOW PASS FILTER
;          WITH CORNER FREQUENCY AT 1.1 KHZ
;
;
FILTER: ZAC      ;1ST STAGE - ORDER 1
LT        Y11
MPY       FC1
LTA       SIG
MPY       FC2
LTA       Y11
SACH      Y1,S4
DMOV      Y1      ;Y11 := Y1
;
;
ZAC      ;2ND STAGE - ORDER 2
MPY      FC5
LTA      Y22
MPY      FC4
LTD      Y21      ;Y22 := Y21
MPY      FC3
LTA      Y21
SACH      Y2,S4
DMOV      Y2      ;Y21 := Y2
;
;
ZAC      ;3RD STAGE - ORDER 2
MPY      FC8
LTA      Y32
MPY      FC7
LTD      Y31      ;Y32 := Y31
MPY      FC6
APAC
SACH      Y3,S4
DMOV      Y3      ;Y31 := Y3
;
;
SACH      SIG,S4      ;FILTER OUTPUT
;
RET
;
;
----- THE LG ALGORITHM -----
;
;
LG:      LARP      0      ;ARP = 0; DP = 0
;
;
LAC      RX1      ;R(1) *2**15
SACL     DR        ;DIVISOR = R(1) *2**15
;
;
LAC      RX2,S15   ;DIVIDEND = R(2) *2**30
CALL     DIVNEW    ;(R(2)/R(1)) *2**15
SACL     FK1       ;FK(1) = (-R2/R1) *2**15
;
;
LAC      RX2      ;R(2) *2**15
SACL     XD1       ;D(1) = R(2) *2**15
;

```

```

;
LAC      RX1
SACL     DR                ;DIVISOR=R(1) *2**15
;
LT        RX2              ;R(2) *2**15
MPY       RX2              ;(R(2))**2 *2**30
PAC
CALL      DIVNEW           ;DIVIDEND = (R(2)**2 *2**30
ADD        RX1             ;((R(2)**2)/(R(1))) *2**15
SACL      XD2              ;(R(1) - (R(2)**2/(R(1)))) *2**15
;
;D(2) = (R(1) - (R(2)**2/R(1))) *2**15
;
LG1 MACRO      I,J
LAC      RX&I,S15
SACH      B1,X1
MPYK      0
LARK      ARO,FK1
LARK      AR1,XD1
CALL      LG&J&S
ENDM
;
J=1
I=3
REPT      M1-1
LG1      %I,%J
I=I+1
J=J+1
ENDM
;
LAC      LGFLAG            ;WHERE TO RETURN? - STACK TOO DEEP FOR RET
BZ       LGRET1            ;      ORDER OF 4 FOR PITCH
;
REPT      (M2-1)-(M1-1)
LG1      %I,%J
I=I+1
J=J+1
ENDM
;
B        LGRET2            ;      ORDER OF 10 FOR ENVELOPE
;
LG2 MACRO J
MAR      *+,AR1
MAR      *+,ARO
LG&J&S:
LTA      *+,ARO
MAR      *+,AR1
SACH      Y,X1
LAC      *+,AR1,S15
MPY       Y
APAC
SACH      *~,AR1,X1
LAC      Y,S15
MPY       *~,AR1
DMOV      *+,AR1
ENDM
;
J=M2-1
REPT      M2-1
LG2      %J
J=J-1
ENDM
;
LGOS:    APAC

```



```

SACH      Y,X1
MAR        +,AR1
MAR        +,AR1
LAC        *,ARO
SACL       DR          ;DIVISOR = D(I) *2**15
LAC        Y,S15       ;DIVIDEND= Y *2**30
CALL       DIVNEW      ;(Y/D(I)) *2**15
SACL       *,AR1       ;FK(I) = -(Y/D(I)) *2**15
LAC        *-,ARO,S15  ;D(I) *2**30
LT         *,AR1       ;Y *2**15
MPY        Y          ;(Y * FK(I)) *2**30
APAC              ;(D(I) + Y * FK(I)) *2**30
DMOV       +,AR1
MAR        +,AR1
MAR        +,AR1
SACH       *,ARO,X1    ;D(I+1) = (D(I) + Y * FK(I)) *2**15
RET

;
;
;----- DIVISION -----
;
DIVNEW: SACH DDH
SACL      DDL
ZAC
SACL      TEMP2
LACK      1
SACL      ONE
LAC       ONE,S15
SUB       ONE
SACL      PLUS1
;
SAR       ARO,DUMMY
LARK      ARO,15
LARP      0
;
LAC       ONE,S15
AND       DDH
BNZ       LESS0
ZALH     DDH
ADDS     DDL
;
221: ADD    TEMP2
SUBC     DR
BANZ     221
;
SACH     REM
AND      PLUS1
;
SACL     TEMP2
ZAC
SUB      TEMP2
;
LAR      ARO,DUMMY
RET
;
LESS0: LACK 0
SUBH    DDH
SUBS    DDL
;
222: ADD    TEMP2
SUBC     DR

```

```

        BANE      222
;
;      SACH      REM
;      AND       PLUS1
;      LAR       ARO,DUMMY
;      RET
;
;
;----- TRANSFORM K'S TO A'S -----
;
;W A R N I N G : NA'S MUST BE IN CONTIGUOUS ARRAY
;
;      ST1B MACRO I,IP,IQ,MN2
;      LARK      ARO,NA&IQ      ;ADDR OF NA(IQ)
;      LARK      AR1,NA&IP      ;ADDR OF NA(IP)
;      LAC       FK&I          ;PASS FK'I IN TEMP
;      SACL      TEMP
;      LACK      MN2-1          ;NUMBER OF LOOPS INTO ACC FOR CALL
;      CALL      LPI           ;CALL CALC SUB
;      ENDM
;
;      ST2B MACRO XIFLAG,I,P1,P2
;
;      IFE      XIFLAG          ;THIS DONE ONLY FOR EVEN VALUES OF I
;      LAC      NA&P1,S15        ;NA&P1*2**15
;      LT       NA&P1
;      MPY      FK&I            ;NA&P1*(FK&I*2**15)
;      APAC     NA&P1*(1+FK&I)*2**15
;      SACH     NA&P1,X1
;      ENDIF
;
;      LAC      FK&I,S13        ;[FK&I*2**15]*2**13
;      SACH     NA&P2          ;[FK&I*2**28]/2**16=FK&I*2**12
;      ENDM
;
;LPI:   SACL     COUNT          ;SAVE COUNT
;      LARP     1              ;USE AR1 FIRST
;
;LPT:   LAC      *,ARO,S15      ;NA&IP*2**15
;      LT       TEMP          ;FK&I*2**15
;      MPY      *,AR1          ;P := FK&I*NA&IQ*2**15
;      LTA      *,AR1          ;T := NA&IP,
;      SACH     *,ARO,X1        ;ACC := [NA&IP+FK&I*NA&IQ]*2**15
;      SACH     *,ARO,X1        ;NA&IP := [NA&IP+FK&I*NA&IQ]*2**15
;
;      LAC      *,ARO,S15      ;NA&IQ*2**15
;      MPY      TEMP          ;NA&IP STILL IN T REG
;      APAC     TEMP          ;P := NA&IP*FK&I*2**15
;      SACH     *-,AR1,X1      ;ACC := [NA&IQ+FK&I*NA&IP]*2**15 (OLD)
;      SACH     *-,AR1,X1      ;NA&IQ := [NA&IQ+FK&I*NA&IP]*2**15 (OLD)
;
;      LAC      COUNT
;      SUB      ONE            ;DECREMENT LOOP COUNT
;      SACL     COUNT
;
;      BGZ      LPT
;      RET
;
;-----
;
;KTOA:  LAC      ONE,S12

```

```

;      SACL      NA1          ;NA1 := 2**12 - 1 IN OUR CONVENTION
;
;      LT        FK1
;      MPY        FK2          ;FK1*FK2*2**30
;      PAC
;      ADD        FK1,S15      ;[FK1+FK1*FK2*2**15]*2**15
;      SACH      TEMP
;      LAC        TEMP,S14
;      SACH      NA2          ;NA2 :=[FK1+FK1*FK2]*2**12
;
;      LAC        FK2,S13
;      SACH      NA3          ;NA3 := FK2*2**12
;
;      I=3
;      REPT      (M1-2)        ;TRANSFORM FIRST FK'S FOR PITCH AND ENV.
;      MN2=(I-1)/2+1
;      IQ=I
;      IP=2
;      ST1B      %I,%IP,%IQ,%MN2
;      XIFLAG=2*(I/2)-I
;      P1=MN2+1
;      P2=I+1
;
;      ST2B %XIFLAG,%I,%P1,%P2
;
;      I=I+1
;      ENDM
;
;      LAC      LGFLAG          ;IF FLAG=1 THEN ENV
;      BZ      RETURN          ;IF FLAG=0 THEN PITCH
;
;CONTINU:
;      REPT      (M2-2)-(M1-2) ;TRANSFORM THE REMAINING K'S FOR ENV ONLY
;      MN2=(I-1)/2+1
;      IQ=I
;      IP=2
;      ST1B      %I,%IP,%IQ,%MN2
;      XIFLAG=2*(I/2)-I
;      P1=MN2+1
;      P2=I+1
;
;      ST2B %XIFLAG,%I,%P1,%P2
;
;      I=I+1
;      ENDM
;
;RETURN: RET
;
;----- INVERSE FILTER -----
;      SIG NA6-NA9 + DUMMY IN 22: COEFFS NA2-NA5
;
;INVFIL: 2AC          ;ZERO INVFIL'S DELAY LINE
;      SACL      FC8
;      SACL      FC7
;      SACL      FC6
;      SACL      FC5
;
;      LAC      NA5          ;GET INVFIL'S FILTER COEFFICIENTS
;      SACL      FC4
;      LAC      NA4

```

```

        SACL    FC3
        LAC     NA3
        SACL    FC2
        LAC     NA2
        SACL    FC1
;
        LACK    OUTB1          ; COUNTER
        SACL    COUNT
        LARK    AR1,128D+5D    ; INPUT POINTER
        LARK    ARO,0          ; OUTPUT POINTER
        LARP    1
;
FLP:    LAC     *,S12,ARO      ; IX(I)
        MPYK    0              ; P := 0
        LTD     FC8            ; LOAD S, ACC, SHIFT
        MPY     FC4            ; MULT X COEFF
        LTD     FC7            ; LOAD S, ACC, SHIFT
        MPY     FC3            ; MULT X COEFF
        LTD     FC6            ; LOAD S, ACC, SHIFT
        MPY     FC2            ; MULT X COEFF
        LTD     FC5            ; LOAD S, ACC, SHIFT
        MPY     FC1            ; MULT X COEFF
        APAC    ; FINAL ADD P TO ACC
        SACH    *+,X4,AR1      ; OUTPUT OF CONVOLUTION
        LAC     *+,AR1         ; CURRENT IX
        SACL    FC5            ; INTO DELAY LINE
;
        LAC     COUNT
        SUB     ONE
        SACL    COUNT
        BNZ     FLP
;
        RET
;
;
; ----- PRAM DATA -----
; NOTE: PRAM DATA MUST BE DECLARED IN THE FOLLOWING ORDER!
;
FTABLE: DW 2928,1167,5711,-2379,763,5774,-3328,1650
;
WIND1:                                ; GENERATE PITCH'S WINDOW
        I=1+3
        REPT    W1
        TRGEN1  %I
        I=I+1
        ENDM
;
WIND2:                                ; GENERATE ENV'S WINDOW
        I=1
        REPT    W2
        TRGEN2  %I
        I=I+1
        ENDM
;
; NOTE: THE PROGRAM WILL FAIL IF THE NEXT LINE IS NOT PRESENT
; IT IS NECESSARY FOR ENV'S AUTOCORRELATION ROUTINE
;
DW 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
;
CNVRT:  DW 0000H, 0002H, 0004H, 0006H, 0008H, 000AH, 000CH, 000EH
        DW 0010H, 0012H, 0014H, 0016H, 0018H, 001AH, 001CH, 001EH

```

```

    DW 0021H, 0025H, 0029H, 002DH, 0031H, 0035H, 0039H, 003DH
    DW 0041H, 0045H, 0049H, 004DH, 0051H, 0055H, 0059H, 005DH
    DW 0063H, 006BH, 0073H, 007BH, 0083H, 008BH, 0093H, 009BH
    DW 00A3H, 00AEH, 00B3H, 00BBH, 00C3H, 00CBH, 00D3H, 00DBH
    DW 00E7H, 00F7H, 0107H, 0117H, 0127H, 0137H, 0147H, 0157H
    DW 0167H, 0177H, 0187H, 0197H, 01A7H, 01B7H, 01C7H, 01D7H
    DW 01EFH, 020FH, 022FH, 024FH, 026FH, 028FH, 02AFH, 02CFH
    DW 02EFH, 030FH, 032FH, 034FH, 036FH, 038FH, 03AFH, 03CFH
    DW 03FFH, 043FH, 047FH, 04BFH, 04FFH, 053FH, 057FH, 05BFH
    DW 05FFH, 063FH, 067FH, 06BFH, 06FFH, 073FH, 077FH, 07BFH
    DW 081FH, 089FH, 091FH, 099FH, 0A1FH, 0A9FH, 0B1FH, 0B9FH
    DW 0C1FH, 0C9FH, 0D1FH, 0D9FH, 0E1FH, 0E9FH, 0F1FH, 0F9FH
    DW 105FH, 115FH, 125FH, 135FH, 145FH, 155FH, 165FH, 175FH
    DW 185FH, 195FH, 1A5FH, 1B5FH, 1C5FH, 1D5FH, 1E5FH, 1F5FH
;
IF TEST
;
SCOPE1: DW OUTB1 DUP (0) ;PITCH'S DISPLAY BUFFER
SCOPE2: DW OUTB2-1 DUP (0, ;ENVELOPE'S DISPLAY BUFFER
END1: DW 0
;
ELSE
;
SCOPE1: DW 03FFH ;SYNC PULSE
        DW (OUTB1-1) DUP (0) ;EMPTY BUFFER
SCOPE2: DW 00FFH ;SYNC PULSE
        DW (OUTB2-2) DUP (0) ;EMPTY BUFFER
END1: DW 0
;
ENDIF
;
        ORG (2*FFTLLOC)-2
;
END2: DW 0 ;THE END
;
        MAIN ENDS
        END START

```

Appendix D

TMS320C25 source code listing of the feature extraction system software for the SWDS.

```

PAGE 63,132
LOG=0          ;CONDITIONAL ASSEMBLY TEST VARIABLE
               ;0(FALSE)-NO DATA LOGGING 1(TRUE)-DATA LOGGING
;
;----- C25-3F.ASM -----
;
; COPYRIGHT (C) MAY 1988, PETER A. BARSZCZEWSKI, CARLETON UNIVERSITY,
; ALL RIGHTS RESERVED
;
;----- MEMORY MAP -----
;
; PAGE 0      : ( 96-127) ISR AND IMPORTANT DATA
; PAGE 4 & 5   : ( 0-246) DUPLICATE INPUT ARRAY FOR MAC IN AUTOCORRELATION
; PAGE 6      : ( 0-127) SCRATCH DATA VARIABLES
; PAGE 7 & 8   : ( 0-245) PROCESSING BUFFER (DOWNSAMPLED AND WINDOWED DATA)
;              : (246-253) 8 LAG DELAY LINE FOR LOW-PASS FILTER
; PAGE 9 & A   : ( 0-245) 246 POINT CLEAN INPUT BUFFER
; PAGE B      : ( 0-122) 123 POINTS BEING COLLECTED BY THE ISR
; PAGE C      : ( 0- 81) 82 POINT OUTPUT BUFFERS FOR DIAGNOSTICS
;              : ( 82-122) 41 POINT LPF SIGNAL FROM LAST FRAME
;
;----- CROSS-ASSEMBLER -----
;
;      IF1
;      INCLUDE C:\ASM\TI25A.ASM
;      ENDIF
;
;----- CONSTANTS -----
;
;      .RADIX 10
;
;      IF LOG
;      IF1
%OUT *          DATA LOGGING IS ON          *
;      ENDIF
;      ELSE
;      IF1
%OUT *          DATA LOGGING IS OFF         *
;      ENDIF
;      ENDIF
;
PAGE0=0        ;STARTING ADDRESSES OF PAGES FOR INDIRECT ADDRESSING
PAGE4=200H
PAGE5=280H
PAGE6=300H
PAGE7=380H
PAGE8=400H
PAGE9=480H
PAGEA=500H
PAGEB=580H
PAGEC=600H
PAGED=680H
;
PRAM=0FF00H    ;STARTING ADDRESS OF BLOCK B0 IN PRAM
FFTLOC=00D36H ;STARTING ADDRESS OF FFT
;
NPTS=246      ;NUMBER OF DATA POINTS COLLECTED PER SPEECH FRAME
NPTS2=NPTS/3  ;NUMBER OF DOWNSAMPLED POINTS FOR PITCH
M1=4          ;LPC ORDER FOR SIFT

```

```

M2=10          ;LPC ORDER FOR FORMANT EXTRACTION
OUTB1=50       ;NUMBER OF POINTS IN PITCH'S OUTPUT BUFFER
OUTB2=32       ;NUMBER OF POINTS IN ENVELOPE'S OUTPUT BUFFER
;
JMI=4          ;LOWER LIMIT FOR PITCH PEAK PICKER (1.5 MSEC = 667 HZ.)
JMA=41         ;UPPER LIMIT FOR PITCH PEAK PICKER (15.375 MSEC = 65 HZ.)
IZM=84         ;MAX NUMBER OF ZERO CROSSINGS (2.73 CROSSINGS/MSEC)
ENERGY=07A1H   ;ENERGY THRESHOLD (5.0 * 10**5) / 2**8
RANGE=8        ;RANGE THAT PITCH IS TRACKED (F0 - RANGE, ... , F0 + RANGE)
;
;
;----- MACROS -----
;
DEFBFB MACRO    I,J1,J2
DATA            B&I
SAME            NA&J1
DATA            XD&I
SAME            NA&J2
ENDM

DEF MACRO        I,J
DATA            LOW&I
SAME            FK&J
DATA            HIGH&I
SAME            RX&J
ENDM

CALFFT MACRO
DW              OFE80H
DW              FFTLOC
ENDM

;
;
;----- PAGE 0 - ISR AND IMPORTANT DATA PAGE -----
;
DORG0

;
DATA            DRR              ;SERIAL PORT DATA RECEIVE REGISTER
DATA            DXR              ;SERIAL PORT DATA TRANSMIT REGISTER
DATA            TIM              ;TIMER REGISTER
DATA            PRD              ;PERIOD REGISTER
DATA            IMR              ;INTERRUPT MASK REGISTER
DATA            GREG             ;GLOBAL MEMORY ALLOCATION REGISTER
;
RES            90D              ;PAGE 0 DATA LOCATIONS 96 TO 125 ONLY
;
DATA            ICOUNT           ;ISR - INPUT DATA COUNTER
DATA            ISTATE1         ;ISR - STATUS REGISTER ST0 STORAGE
DATA            ISTATE2         ;ISR - STATUS REGISTER ST1 STORAGE
DATA            IONE            ;ISR - CONSTANT
DATA            IHIGH           ;ISR - HIGH ACCUMULATOR STORAGE
DATA            ILOW           ;ISR - LOW ACCUMULATOR STORAGE
;
DATA            F0              ;FEATURE ADDRESS STORAGE - MUST BE IN ORDER
DATA            F1
DATA            F2
DATA            F3
;
DATA            F1A             ;FOR TRACKING F1
DATA            F1B
DATA            F1C

```



```

DATA F2A ;FOR TRACKING F2
DATA F2B
DATA F2C
DATA F3A ;FOR TRACKING F3
DATA F3B
DATA F3C
;
DATA N1B ;DECISION VARIABLES FOR DECIS
DATA N2B
DATA N3B
DATA NMINB
DATA NMAXB
DATA ENRHIB
DATA ENRLOB
DATA NZB
;
;----- PAGE 6 - SCRATCH VARIABLES PAGE -----
;
DORG6
;
I=1
REPT M2+1
J1=2*I-1
J2=2*I
DEFBF %I,%J1,%J2
I=I+1
ENDM
;
I=0
J=1 ;DEFINE FK's, RX's
REPT M2+1
DEF %I,%J
I=I+1
J=J+1
ENDM
;
DATA N0 ;THE NEXT 9 LOCS MUST BE CONTIGUOUS
DATA N1 ;DECIS
DATA N2 ;DECIS
DATA N3 ;DECIS
DATA NMIN ;DECIS
DATA NMAX ;DECIS
DATA ENRHI ;DECIS, AUTO (HIGH 16 BITS OF ENERGY)
DATA ENRLO ;DECIS, AUTO (LOW 16 BITS OF ENERGY)
DATA NZ ;DECIS
DATA L ;DECIS
DATA ALI ;DECIS
DATA LL ;DECIS
DATA AP ;DECIS
DATA BB ;DECIS
DATA SLM1 ;DECIS
DATA SL ;DECIS
DATA SLP1 ;DECIS
DATA Y ;LG
DATA DUMMY ;LG
DATA LGFLAG ;LG,KTOA
DATA WPTR ;GETDAT,WINDAT
DATA WIND ;GETDAT,WINDAT
DATA INDEX ;WINDAT
DATA LAST ;WINDAT
;

```

```

DATA   BSTRT3           ;NPIC - THESE MUST BE IN THIS ORDER
SAME   A1
DATA   BSTRT2           ;NPIC
SAME   A2
DATA   BSTRT1           ;NPIC
SAME   A3
;
DATA   ONE              ;GENERAL PURPOSE CONSTANT
DATA   TOP              ;CONSTANT PTR TO TOP OF PROCESSING BUFFER
DATA   DPTR             ;GENERAL PURPOSE PRIMARY DATA POINTER
DATA   TEMP             ;GENERAL PURPOSE TEMPORARY VARIABLE
DATA   XHIGH            ;GETDAT,AUTO
DATA   XLOW             ;GETDAT,AUTO
DATA   DR               ;DIVIDE - DIVISOR
DATA   DDH              ;DIVIDE - DIVIDEND HIGH
DATA   DDL              ;DIVIDE - DIVIDEND LOW
DATA   PLUS1            ;DIVIDE
DATA   TEMP2            ;DIVIDE
DATA   COUNT            ;AUTO
DATA   ZMASK            ;AUTO
DATA   SCOUNT          ;AUTO
DATA   MAXVAL           ;NPIC, PPIC
DATA   RXT              ;MAGNTD
DATA   IYT              ;MAGNTD
;
DATA   Y1               ;DELAY LINE FOR LP FILTERING OF INPUT DATA
DATA   Y11
DATA   Y2
DATA   Y21
DATA   Y22
DATA   Y3
DATA   Y31
DATA   Y32
;
DATA   RS1              ;TEMPORARY LOCATION FOR AUTO RESULTS
DATA   RS2
DATA   RS3
DATA   RS4
DATA   RS5
DATA   RS6
DATA   RS7
DATA   RS8
DATA   RS9
DATA   RS10
DATA   RS11
;
;----- PRAM -----
;
AORG                      ;START PROGRAM SECTION
;
START: B      GO
      B      ISR
;
      DW      (20H-4) DUP(?) ;FIRST 20H OF PRAM IS RESERVED
;
;----- INITIALIZATION -----
;
GO:   DINT          ;DISABLE INTERRUPTS
      SOVM          ;SET OVERFLOW MODE
      SSXM          ;SET SIGN EXTENSION MODE
      SPM           0 ;SET P REG. OUTPUT MODE TO NO SHIFT

```

```

        CNFP                                ;SET BLOCK B0 AS PRAM - FOR AUTO
        LARP      0                        ;AUXILARY REGISTER 0
        LDPK      0                        ;PAGE 0 - ISR AND IMPORTANT DATA PAGE
;
        LRLK      AR0,PAGEC                ;OUTPUT BUFFER IS ON PAGE C
        LALK      03FFFH                  ;SYNC. PULSE IN THE OUTPUT BUFFER
        SACL      *
;
        LACK      JMI                      ;INITIALIZE DECIS SEARCH RANGE
        SACL      NMINB
        LACK      JMA
        SACL      NMAXB
        ZAC                               ;ZERO DECISION VARIABLES AND FEATURE LOCS.
        LARK      AR0,PAGE0+FO
        RPTK      7
        SACL      **+,AR0
;
        LRLK      AR0,PAGE7+NPTS           ;ZERO THE LOW-PASS FILTER DELAY LINE
        RPTK      7
        SACL      **+,AR0
;
; INITIALIZE SAMPLING AND ISR DATA
;
        LACK      1
        SACL      IONE                     ;ISR - CONSTANT ON PAGE 0
;
        LALK      OFFC1H
        SACL      IMR                     ;SET INTERRUPT MASK REGISTER
;
        OUT       IONE,PA3                 ;DISABLE PORT 3 ON AIB (SEE P-1 OF AIB MANUAL)
;
        LALK      624                      ;SET THE SAMPLING RATE TO 8 KHZ
        SACL      ICOUNT                   ; N = CLKOUT/(SAMPLING RATE) - 1
        OUT       ICOUNT,PA1               ;   = (5 MHZ / 8 KHZ) - 1 = 624
;
        LACK      10                      ;SET THE OUTPUT MODE (CDA1=1, CAD1=1)
        SACL      ICOUNT
        OUT       ICOUNT,PA0
;
        LRLK      AR6,PAGEC                ;ISR - POINTER TO SCOPE BUFFER
        LRLK      AR7,OUTB1+OUTB2-1        ;ISR - COUNTER FOR SCOPE BUFFER DISPLAY
;
; FILL THE FIRST BUFFER
;
        IF LOG
        ELSE
            EINT
        ENDIF
;
        LRLK      AR5,PAGE9+NPTS/2         ;ISR - INPUT BUFFER
        LACK      NPTS/2                   ;ISR - INITIALIZE INPUT BUFFER COUNTER (WHICH
        SACL      ICOUNT                   ;   IS ALSO A FLAG TO START COLLECTING)
        IF LOG
        WAIT2: CALL    ISR
        ELSE
        WAIT2: IDLE
        ENDIF
;
        LAC       ICOUNT
        BNZ       WAIT2
;
        LRLK      AR5,PAGEB                ;ISR - INPUT BUFFER STARTS ON PAGE B

```

```

        LACK    NPTS/2      ;ISR - INITIALIZE INPUT BUFFER COUNTER (WHICH
        SACL    ICOUNT      ;      IS ALSO A FLAG TO START COLLECTING)
;
        LDPK    6
        LARK    AR0,(NPTS2/2)-1      ;NUMBER OF OUTPUT POINTS
        LRLK    AR1,PAGEC+OUTB1+OUTB2 ;OUTPUT (PROCESSING) BUFFER
        LRLK    AR2,PAGE9+NPTS/2      ;INPUT (CLEAN) BUFFER
        CALL    GETDAT      ;FILTER AND DOWNSAMPLE PITCH'S INPUT
;
;----- THE MAIN PROGRAM -----
;
HERE:    LDPK    0      ;ON TO PAGE 0 FOR BUFFER SWAPING ETC.
        IF LOG
WAIT:    CALL    ISR
        ELSE
WAIT:    IDLE      ;FINISH FILLING THE INPUT BUFFER
        ENDIF
        LAC     ICOUNT
        BNZ     WAIT
;
; REARRANGE THE INPUT BUFFER
;
        LARP    0      ;MOVE THE BOTTOM HALF OF THE INPUT BUFFER
        LRLK    AR0,PAGE9      ; UP TO THE TOP HALF
        RPTK    (NPTS/2)-1
        BLKD    (PAGE9+NPTS/2),*+,AR0
;
        LRLK    AR0,(PAGE9+NPTS/2)      ;MOVE THE JUST COLLECTED DATA TO THE
        RPTK    (NPTS/2)-1      ; BOTTOM HALF OF THE INPUT BUFFER
        BLKD    PAGEB,*+,AR0
;
; RESTART THE ISR COLLECTING DATA
;
        LRLK    AR5,PAGEB      ;INITIALIZE ISR'S INPUT BUFFER POINTER
        LACK    NPTS/2      ;INITIALIZE INPUT BUFFER COUNTER (WHICH
        SACL    ICOUNT      ; IS ALSO A FLAG TO START COLLECTING)
;
        ZAC
        LARK    AR0,F0      ;ZERO THE FEATURES FROM THE LAST FRAME
        RPTK    4-1      ;FEATURES MUST BE STORED IN ORDER
        SACL    *+,AR0
;
;----- PITCH EXTRACTION CALCULATIONS -----
;
PSTRT:   LDPK    6      ;POINT TO TEMPORARY VARIABLES PAGE
        LACK    1      ;CONSTANT ON PAGE 6
        SACL    ONE
;
        LARK    AR0,NPTS-2      ;LOOP COUNTER
        LRLK    AR1,PAGE9      ;STARTING ADDRESS OF PROCESSING BUFFER
        LARK    AR2,PAGE0+N2B      ;STORAGE FOR ZERO CROSSING COUNT
        CALL    ZCD      ;COUNT NUMBER OF ZERO CROSSINGS
;
        LARP    0      ;MOVE FILTERED DATA FROM LAST FRAME TO
        LRLK    AR0,PAGE7      ; PROCESSING BUFFER
        RPTK    NPTS2/2-1
        BLKD    PAGEC+OUTB1+OUTB2,*+,AR0
;
        LARK    AR0,(NPTS2/2)-1      ;NUMBER OF OUTPUT POINTS

```

```

LRLK  AR1,PAGE7+NPTS2/2      ;OUTPUT (PROCESSING) BUFFER
LRLK  AR2,PAGE9+NPTS/2      ;INPUT (CLEAN) BUFFER
CALL  GETDAT                ;FILTER AND DOWNSAMPLE PITCH'S INPUT
;
LARP  0                      ;MOVE FILTERED DATA FOR NEXT FRAME
LRLK  ARO,PAGEC+OUTB1+OUTB2
RPTK  NPTS2/2-1
BLKD  PAGE7+NPTS2/2,*+,ARO
;
LRLK  ARO,(NPTS2-1)         ;NUMBER OF POINTS
LRLK  AR1,PAGE7             ;POINTER TO TOP OF INPUT (CLEAN) BUFFER
LRLK  AR2,PAGE7+NPTS2       ;POINTER TO TOP OF OUTPUT (PROCESSING) BUFFER
CALL  PREEMF                ;PRE-EMPHASIZE WITH OPTIMAL MU
;
LRLK  ARO,(NPTS2-1)         ;NUMBER OF POINTS
LRLK  AR1,PAGE7+NPTS2       ;POINTER TO TOP OF INPUT (CLEAN) BUFFER
LRLK  AR2,PAGE7             ;POINTER TO TOP OF OUTPUT (PROCESSING) BUFFER
LACK  3                     ;USE EVERY THIRD POINT OF THE HAMMING WINDOW
CALL  WINDOW                ;APPLY WINDOW FUNCTION
;
LRLK  ARO,PAGE6+RS1         ;POINTER TO OUTPUT ARRAY
LRLK  AR1,PAGE7             ;POINTER TO INPUT ARRAY (PROCESSING BUFFER)
LARK  AR2,M1                ;NUMBER OF LAGS
LACK  (NPTS2-1)             ;NUMBER OF POINTS IN INPUT ARRAY
CALL  AUTO                  ;GET 4 LAGS FROM DOWNSAMPLED SIGNAL
;
LARP  0                      ;SAVE ENERGY ON PAGE 0
LARK  ARO,PAGE0+ENRHIB
LAC   ENRHI
SACL  *+,ARO
LAC   ENRLO
SACL  *+,ARO
;
LRLK  ARO,PAGE6+RS1         ;5 R'S TO WHERE LG CAN FIND THEM, RX1-RX5
LRLK  AR1,PAGE6+RX1
LARK  AR2,M1
ADLP: LAC  *+,AR1
SACL  *+,AR1
MAR   *+,AR2
BANZ  ADLP,ARO
;
LACK  0D                    ;FLAG LG FOR 4TH ORDER AND RETURN HERE
SACL  LGFLAG
CALL  LG                    ;5 RX'S TO 5 FK'S
;
CALL  KTOA                  ;5 FK'S TO 5 NA'S
;
LRLK  ARO,PAGE7+NPTS2       ;STARTING ADDRESS OF PROCESSING BUFFER
LARK  AR1,(NPTS2-1)         ;NUMBER OF POINTS
CALL  INVFIL                ;INVERSE FILTER PROCESSING BUFFER WITH NA'S
;
LRLK  ARO,(NPTS2-1)         ;NUMBER OF POINTS
LRLK  AR1,PAGE7+NPTS2       ;POINTER TO TOP OF INPUT BUFFER
LRLK  AR2,PAGE7             ;POINTER TO TOP OF OUTPUT BUFFER
LACK  3                     ;USE EVERY THIRD POINT OF THE HAMMING WINDOW
CALL  WINDOW                ;APPLY WINDOW FUNCTION
;
LRLK  ARO,PAGE7             ;POINTER TO OUTPUT ARRAY
LRLK  AR1,PAGE7             ;POINTER TO INPUT ARRAY
LARK  AR2,(OUTB1-1)         ;NUMBER OF LAGS
LACK  (NPTS2-1)             ;NUMBER OF POINTS IN INPUT ARRAY

```

```

CALL    AUTO                ;AUTOCORRELATE FILTERED SIGNAL
;
LARP    0                    ;MOVE PITCH BUFFER TO DISPLAY BUFFER
LRLK    AR0,PAGEC
RPTK    (OUTB1-1)
BLKD    PAGE7,*+,AR0
;
AD50:   LRLK    AR1,PAGE7      ;STARTING ADDRESS OF PITCH BUFFER
LARK    AR2,PAGE0+FO          ;LOCATION OF PITCH RESULT
CALL    DECIS                ;PITCH DECISION ROUTINE
;
IF LOG
OUT      NZ,PA8
OUT      ENRHI,PA9
OUT      AP,PA10
LDPK     0
OUT      FO,PA11
LDPK     6
ENDIF
;
;
;----- FORMANT EXTRACTION CALCULATIONS -----
;
FSTRT:  LRLK    AR0,(NPTS-1)    ;NUMBER OF POINTS
LRLK    AR1,PAGE9              ;POINTER TO TOP OF INPUT (CLEAN) BUFFER
LRLK    AR2,PAGE7              ;POINTER TO TOP OF OUTPUT (PROCESSING) BUFFER
CALL    PREEMF                ;PRE-EMPHASIZE SIGNAL WITH OPTIMAL MU
;
LRLK    AR0,(NPTS-1)    ;NUMBER OF POINTS
LRLK    AR1,PAGE7      ;POINTER TO TOP OF INPUT BUFFER
LRLK    AR2,PAGE7      ;POINTER TO TOP OF OUTPUT BUFFER
LACK    1              ;USE EVERY POINT OF THE HAMMING WINDOW
CALL    WINDOW         ;APPLY WINDOW FUNCTION
;
LRLK    AR0,PAGE6+RS1   ;POINTER TO OUTPUT ARRAY
LRLK    AR1,PAGE7       ;POINTER TO INPUT ARRAY
LARK    AR2,M2          ;NUMBER OF LAGS
LACK    (NPTS-1)        ;NUMBER OF POINTS IN INPUT ARRAY
CALL    AUTO            ;GET 10 LAGS FROM INPUT SIGNAL
;
LRLK    AR0,PAGE6+RS1   ;11 R'S TO WHERE LG CAN FIND THEM, RX1-RX11
LRLK    AR1,PAGE6+RX1
LARK    AR2,M2
LARP    0
ADLP2:  LAC      *+,AR1
SACL    *+,AR1
MAR     *+,AR2
BANZ    ADLP2,AR0
;
LACK    1D              ;FLAG LG FOR 11TH ORDER AND RETURN HERE
SACL    LGFLAG
CALL    LG              ;RX'S TO FK'S
;
CALL    KTOA            ;IK'S TO NA'S
;
LRLK    AR0,PAGE6+NA1   ;INPUT ARRAY
LRLK    AR1,PAGE7       ;PROCESSING BUFFER FOR THE FFT
CALL    PREFFT          ;PREPARE NA'S AND DRAM FOR FFT
;
LRLK    AR0,PAGE7       ;COPY PADDED ARRAY TO PRAM FOR FFT
LARK    AR1,(128-1)

```

```

; LALK FFTBUF
; LARP 0
LOOPZ: TBLW *+,AR1
; ADD ONE
; BANZ LOOPZ,AR0
;
; CALFFT ;DETERMINE FREQ. SPECTRUM USING 11 NA'S
; ADDR FFTBUF ;ADDRESS OF INPUT BUFFER
;
; LDPK 6 ;CHANGE PAGE POINTER BACK TO 6
; LACK 1 ;RE-INITIALIZE THE CONSTANT ONE
; SACL ONE
;
; LRLK AR0,PAGE6 ;LOCATION OF INPUT BUFFER
; LARK AR1,(OUTB2-1) ;COUNTER
; LRLK AR2,PAGEC+OUTB1 ;OUTPUT FOR DISPLAY
; CALL MAGNTD ;CALCULATE MAGNITUDE OF SPECTRUM
;
; LRLK AR0,PAGEC+OUTB1 ;LOCATION OF INPUT DATA
; LARK AR1,OUTB2-3 ;LOOP COUNTER
; LRLK AR2,PAGE6+N1 ;LOCATION OF CURRENT NULLS
; LARK AR3,3-1 ;NUMBER OF NULLS TO BE CALCULATED
; CALL NPIC ;FIND FORMANTS IN SPECTRUM
;
; LARK AR2,PAGE0+F0 ;POINTER TO FEATURE STORAGE
; LRLK AR3,PAGE6+N1 ;POINTER TO LOCATION OF CURRENT NULLS
; CALL FTRACK ;APPLY MEDIAN SMOOTHING TO FORMANT TRACKS
;
; IF LOG
; LDPK 0
; OUT F1,PA12
; OUT F2,PA13
; OUT F3,PA14
; ENDIF
; B HERE ;DONE - PROCESS THE NEXT FRAME
;
; ----- THE ISR ROUTINE -----
; AR5 : INPUT PTR AR6 : OUTPUT PTR AR7 : OUTPUT COUNTER
;
; ISR: SST ISTATE1 ;STORE ST0
; SST1 ISTATE2 ;STORE ST1
; LDPK 0 ;USE PAGE 0
; SACL ILOW ;STORE ACCUMULATOR
; SACH IHIGH
;
; OUTPUT RESULT - CANNOT FLAG THE PITCH OR FORMANTS
;
; LARP 6 ;AR6 POINTS TO THE OUTPUT BUFFER
; OUT *+,PA2,AR7 ;OUTPUT SCOPE INFO, IT'S A 16-BIT #
; BANZ SIN,AR5 ;IF COUNTER < 0 THEN UPDATE OUTPUT REGS.
; LRLK AR6,PAGEC ;OUTPUT BUFFER
; LRLK AR7,OUTB1+OUTB2-1 ;COUNTER
;
; INPUT SAMPLE
;
; SIN: LAC ICOUNT ;IF ICOUNT=0, PROCESSING STILL GOING ON
; BZ ISROUT ; SO --DON'T-- COLLECT DATA
; SUB IONE ;ELSE DECREMENT INPUT COUNTER
; SACL ICOUNT ; AND COLLECT ANOTHER SAMPLE
;

```

```

        IN      *,PA2,AR5      ;GET 16-BIT 2'S COMPLIMENT DATA
        LAC     *,S13,AR5      ;SHIFT RIGHT 3 PLACES (BETTER ACCURACY)
        SACH    *+,AR6        ;STORE 13-BIT 2'S COMPLIMENT DATA
;
ISROUT: ZALS    ILOW          ;RETURN PRE-ISR STATE
        ADDH    IHIGH
        LST1    ISTATE2      ;RESTORE ST1
        LST     ISTATE1      ;RESTORE ST0
        IF LOG
        ELSE
            EINT
        ENDIF
        RET
;
;
;----- ZERO CROSSINGS DETECTOR -----
;      ARO : NUMBER OF POINTS      AR1 : INPUT BUFFER PTR
;      AR2 : ADDRESS FOR ZERO CROSSING COUNT AR3 : ZERO CROSSING COUNT
;
ZCD:    LARK    AR3,0          ;INITIALIZE ZERO CROSSINGS COUNT
        LARP    1
;
ZLOOP:  LAC     *+,AR1        ;CHECK FOR CHANGE OF SIGN BETWEEN CURRENT
        XOR     *,AR0        ; AND NEXT DATA POINTS
        SACL    TEMP        ;EXTEND SIGN
        LAC     TEMP
        BGEZ    NOZC        ;IF NOT NEGATIVE, THEN NO ZERO CROSSING
        LARP    3
        MAR     *+,AR0        ;INCREMENT ZERO CROSSINGS COUNT
;
NOZC:   BANZ    ZLOOP,AR1     ;LOOP UNTIL DONE
        SAR     AR3,TEMP      ;STORE ZERO CROSSINGS COUNT
        LARP    2
        LAC     TEMP
        SACL    *,AR2
        RET
;
;
;----- LOW PASS FILTER -----
; 5TH ORDER BI-QUAD FIR LOW PASS FILTER WITH CORNER FREQUENCY AT 0.9 KHZ
; (EXPECTS ARP=2 !!!!)
;
FILTER: ZAC                                ;1ST STAGE - ORDER 1
        LT      Y11
        MPY     NA1
        LTA     *+,AR2
        MPY     NA2
        LTA     Y11          ;ACC := (Y11*NA1 + S[1]*NA2) * 2**14
        SACH    Y1,S2
        DMOV    Y1          ;Y11 := Y1
;
        ZAC                                ;2ND STAGE - ORDER 2
        MPY     NA5
        LTA     Y22
        MPY     NA4
        LTD     Y21          ;Y22 := Y21
        MPY     NA3
        LTA     Y21          ;ACC := (Y11*NA5 + Y22*NA4 + Y21*NA3) * 2**14
        SACH    Y2,S2
        DMOV    Y2          ;Y21 := Y2
;

```



```

      ZAC                      ;3RD STAGE - ORDER 2
      MPY      NA8
      LTA      Y32
      MPY      NA7
      LTD      Y31          ;Y32 := Y31
      MPY      NA6
      APAC                      ;ACC := (Y21*NA8 + Y32*NA7 + Y31*NA6) * 2**14
      SACH      Y3,S2
      DMOV      Y3          ;Y31 := Y3
;
      SACH      TEMP,S2      ;FILTER OUTPUT
;
      RET
;
;----- GETDAT -----
;      FILTER AND DOWNSAMPLE (DECIMATE BY THREE)
;      AR0 : NUMBER OF POINTS - 1          AR1 : OUTPUT ARRAY
;      AR2 : INPUT ARRAY
;
GETDAT: LARP      4
      LALK      FTABLE          ;ADDRESS OF COEFFICIENT TABLE
      LRLK      AR4,(NA1+PAGE6) ;ADDRESS OF FILTER COEFS IN DRAM
      RPTK 7D
      TBLR      *+,AR4
;
      LRLK      AR4,(Y1+PAGE6) ;LOAD DELAY LINE FROM LAST FRAME
      RPTK 7D
      BLKD      PAGE7+NPTS,*+,AR4
;
      LARP      2
;
; FILTER AND DOWNSAMPLE THE INPUT BUFFER
;
FLOOP1: CALL      FILTER          ;FILTER 1ST POINT
;
      CALL      FILTER          ;FILTER 2ND POINT
;
      CALL      FILTER          ;FILTER 3RD POINT
;
      LARP      1
      LAC      TEMP
      SACL      *+,AR0          ;STORE 3RD POINT IN OUTPUT ARRAY
;
      BANZ      FLOOP1,AR2      ;LOOP TILL DONE
;
      LARP      4
      LRLK      AR4,PAGE7+NPTS ;STORE DELAY LINE FOR NEXT FRAME
      RPTK 7D
      BLKD      PAGE6+Y1,*+,AR4
;
      RET
;
;----- INVERSE FILTER -----
;      NA6-NA9 : DELAY LINE      NA2-NA5 : COEFS      "NA10" : DUMMY
;      A0 : OUTPUT POINTER      A1 : INPUT POINTER      A2 : COUNTER
;
INVFIL: ZAC                      ;ZERO THE DELAY LINE
      SACL      NA9
      SACL      NA8

```

```

        SACL    NA7
        SACL    NA6
        LARP    0
;
FLP:    LAC     *,S12,ARO      ;IX(I)
        MPYK    0             ;P := 0
        LTD     NA9           ;LOAD S, ACC, SHIFT
        MPY     NA5           ;MULT X COEFF
        LTD     NA8           ;LOAD S, ACC, SHIFT
        MPY     NA4           ;MULT X COEFF
        LTD     NA7           ;LOAD S, ACC, SHIFT
        MPY     NA3           ;MULT X COEFF
        LTD     NA6           ;LOAD S, ACC, SHIFT
        MPY     NA2           ;MULT X COEFF
        APAC    ;FINAL ADD P TO ACC
        SACH    TEMP,S4       ;TEMP. STORAGE OF CONVOLUTION OUTPUT
;
        LAC     *,ARO         ;IX(I) INTO DELAY LINE
        SACL    NA6
        LAC     TEMP          ;SAVE OUTPUT OVER INPUT
        SACL    *+,AR1
;
        BANZ    FLP,ARO       ;REPEAT UNTIL DONE
;
        RET
;
;----- PEAK PICKER FOR PITCH -----
;      ARO : NUMBER OF POINTS TO SEARCH      AR1 : INPUT BUFFER PTR
;      RETURNS PEAK LOCATION IN ACCUMULATOR IN RANGE : 0 .. OUTB1 - 1
;
PPIC:   LARP    1
        LAC     ONE,S15
        SACL    MAXVAL        ;INITIALIZE MAXVAL TO -32768
;
FPEAK:  LAC     *+,ARO         ;CURRENT DATA POINT
        SACL    TEMP
        SUB     MAXVAL
        BLZ     LOWER         ;IF ( CURRENT - MAXVAL < 0 ) THEN LOWER
HIGHER: LAC     TEMP
        SACL    MAXVAL        ;MAXVAL = CURRENT DATA POINT
        SAR     AR1,DPTR      ;SAVE THE ADDRESS OF MAXVAL.
LOWER:  BANZ    FPEAK,AR1     ;LOOP UNTIL DONE
;
        LAC     DPTR          ;ADDRESS ONE LOCATION PAST THE PEAK
        SBLK    PAGE7         ;PUT INTO RANGE 1 .. OUTB1
        SUB     ONE           ;PUT INTO RANGE 0 .. OUTB1 - 1
        RET
;
;----- DECIS -----
;      PERFORMS PITCH DECISION (V/UV) BY USING A SET HEURISTIC RULES.
;
DECIS:  LARP    0             ;RESTORE DECISION PARAMETERS ON PAGE6
        LRLK    ARO,PAGE6+N1
        RPTK    7
        BLKD    PAGE0+N1B,*+,ARO
;
        ZAC     ;ZERO VARIABLES (MAINLY FOR TESTING)
        SACL    AP
        SACL    L

```

```

        SACL    LL
        SACL    SLM1
        SACL    SLP1
        SACL    SL
;
; INITIAL FRAME TESTS
;
        ZALS    ENRLO           ;INPUT ENERGY (32 BITS)
        ADDH    ENRHI
        SBLK    ENERGY,S8      ;THRESHOLD FOR 13-BIT SAMPLES
        BLZ     DE253           ;IF FRAME ENERGY < THRESHOLD THEN UNVOICED
;
        LACK    IZM             ;ZERO CROSSINGS THRESHOLD
        SUB     NZ               ;ZERO CROSSINGS COUNT
        BLZ     DE253           ;IF ZC THRESHOLD < ZC COUNT THEN UNVOICED
;
; FIND AUTOCORRELATION PEAK WITHIN THE RANGE (NMIN, ...NMAX) FROM LAST FRAME
;
        LALK    PAGE7           ;ADDRESS OF TOP OF INPUT BUFFER
        ADD     NMIN
        SACL    DPTR            ;ADDRESS OF FIRST POINT TO SEARCH
        LAC     NMAX
        SUB     NMIN
        SACL    TEMP
        LAR     ARO,TEMP        ;COUNTER := NMAX - NMIN
        LAR     ARI,DPTR        ;ADDRESS POINTER
        CALL    PPIC            ;FIND PEAK WITHIN RANGE OF NMIN TO NMAX
        SACL    L               ;L := PEAK LOCATION
;
; TEST FOR POSSIBLE DOUBLING - I.E. TRUE PEAK IS LESS THAN HIGHEST PEAK
;
        LAC     L,S14
        SACH    TEMP            ;TEMP := L/4
        LAC     L
        SUB     TEMP
        SACL    NMAX            ;NMAX := L - L/4
        SUB     NMIN
        BLZ     DE226           ;IF NMAX < NMIN THEN SKIP TEST
;
        LALK    PAGE7           ;ADDRESS OF TOP OF INPUT BUFFER
        ADD     NMIN
        SACL    DPTR            ;ADDRESS OF FIRST POINT IN SEARCH RANGE
        LAC     NMAX
        SUB     NMIN
        SACL    TEMP
        LAR     ARO,TEMP        ;COUNTER := NMAX - NMIN
        LAR     ARI,DPTR        ;ADDRESS POINTER
        CALL    PPIC            ;FIND PEAK WITHIN NEW RANGE OF NMIN TO NMAX
        SACL    LL              ;LL := NEW PEAK LOCATION
;
        LARP    0
        LALK    PAGE7           ;ADDRESS OF START OF AUTOCORRELATION BUFFER
        ADD     L
        SACL    DPTR
        LAR     ARO,DPTR
        LT      *,ARO           ;SCOPE(L)
        LALK    PAGE7
        ADD     LL
        SACL    DPTR
        LAR     ARO,DPTR
        LAC     *,S12,ARO       ;SCOPE(LL)

```

```

        MPYK    -3481D        ;-0.85
        APAC
        BLEZ    DE226        ;IF SCOPE(LL) > 0.85*SCOPE(L)
        LAC     LL           ;      THEN L := LL (DOUBLING OCCURED)
        SACL    L
;
; PARABOLIC INTERPOLATION - TO FIND HEIGHT AND POSITION OF 'TRUE' PEAK
;
DE226:  LALK    PAGE7        ;ADDRESS OF START OF SCOPE BUFFER
        ADD     L
        SACL    DPTR
        LAR     ARO,DPTR
        LAC     *-,ARO
        SACL    SL           ;SL := SCOPE(L)
        LAC     *+,ARO
        MAR     *+,ARO
        SACL    SLM1        ;SLM1 := SCOPE(L-1)
        LAC     *+,ARO
        SACL    SLP1        ;SLP1 := SCOPE(L+1)
;
        LAC     SL
        SUB     SLM1
        BIEZ    DE253        ;IF SCOPE(L) <= SCOPE(L-1) THEN UNVOICED
        LAC     SL
        SUB     SLP1
        BLEZ    DE253        ;IF SCOPE(L) <= SCOPE(L+1) THEN UNVOICED
;
        LAC     SL,S15        ;-AA := ( 2*SCOPE(L)
        SUB     SLM1,S14      ;      - SCOPE(L-1)
        SUB     SLP1,S14      ;      - SCOPE(L+1) ) * 2**14 / 2
        SACH    DR,S1        ;-AA > 0 ALWAYS
;
        LAC     SLP1,S14
        SUB     SLM1,S14
        SACH    BB           ;BB := ( SCOPE(L+1) - SCOPE(L-1) ) / 4
        LT      BB
        MPY     BB
        PAC     BB*BB
        CALL    DIVNEW
        SACL    TEMP        ;TEMP := -BB*BB/(-AA) = BB*BB/AA
        LAC     SL
        SUB     TEMP
        SACL    AP          ;AP := SCOPE(L) - BB*BB/AA (PEAK HEIGHT)
;
        LAC     BB,S7        ;BB*(2**7)
        CALL    DIVNEW
        SACL    TEMP        ;BB*(2**7)/AA = -(BB*(2**7)/(-AA))
        LAC     L,S7        ;L*(2**7)
        SUB     TEMP
        SACL    ALI        ;ALI := (L - BB/AA)*(2**7) (PEAK POSITION)
;
        LAC     ONE,S6
        LT      ALI
        MPYK    3
        APAC     ;((L - BB/AA)*3 + 0.5)*(2**7)
        SACL    TEMP
        LAC     TEMP,S9      ;((L - BB/AA)*3 + 0.5)*(2**7)
        SACH    NO          ;NO := TRUNC((L - BB/AA)*3 + 0.5)
;
; FINAL FRAME TEST - COMPARING PEAK (AP) TO THE SLOPE OF THE LINE
; 16383 - 300*L WHERE L = PEAK POSITION

```

```

;
    LRLK    ARO,PAGE7      ;SCOPE(0) = 16383
    LAC     *,ARO
    LT      L
    MPYK    -300
    APAC
    SACL    TEMP           ;16383 - 300*L
    LT      TEMP
    LAC     AP,S12
    MPYK    -1638          ;-0.4
    APAC     ;IF AP >= 0.4 * (16383 - 300*L)
    BGEZ    DE891          ;      THEN PEAK O.K.
    LAC     N1
    BZ      DE253          ;IF N1 = 0 THEN LAST FRAME WAS UNVOICED
    LAC     AP,S12
    MPYK    -1229          ;-0.3
    APAC     ;IF AP >= 0.3 * (16383 - 300*L)
    BGEZ    DE891          ;      THEN PEAK O.K.
;
; MARK AS UNVOICED
;
DE253:  ZAC                     ;MARK AS UNVOICED
        SACL    NO
;
; USE MEDIAN SMOOTHING TO SMOOTH THE PITCH CONTOUR
;
DE891:  DMOV    N2             ;N3 := N2
        DMOV    N1             ;N2 := N1
        DMOV    N0             ;N1 := N0
;
        LAC     N1
        SACL    A1
        LAC     N2
        SACL    A2
        LAC     N3
        SACL    A3
;
    LRLK    ARO,PAGE6+A1      ;POINTER TO TOP OF ARRAY TO BE SORTED
    LARK    AR1,3-1          ;NUMBER OF POINTS IN ARRAY - 1
    CALL    SORT
;
    LARP    2
    LAC     A2                ;A2 IS THE MEDIAN
    SACL    *,ARO             ;F0 := MEDIAN
;
; DETERMINE LIMITS OF NEXT FRAME (L - RANGE, ... , L + RANGE)
;
    LACK    JMI
    SACL    NMIN              ;NMIN := JMI
    LACK    JMA
    SACL    NMAX              ;NMAX := JMA
    LAC     N1
    BZ      DE888             ;IF PRESENT FRAME UNVOICED THEN LIMITS SET
;
    LT      ONE
    LAC     L
    MPYK    -RANGE
    APAC
    SACL    NMIN              ;NMIN := L - RANGE
    MPYK    -JMI
    APAC

```

```

        BGEZ    DE886      ;IF NMIN < JMI THEN
        LACK    JMI        ;      NMIN := JMI
        SACL    NMIN
;
DE886:  LACK    RANGE
        ADD     L
        SACL    NMAX      ;NMAX := L + RANGE
        MPYK    -JMA
        APAC
        BLEZ    DE888      ;IF NMAX > JMA THEN
        LACK    JMA        ;      NMAX := JMA
        SACL    NMAX
;
DE888:  LARK    ARO,PAGE0+N1B ;SAVE DECISION PARAMETERS ON PAGE 0
        RPTK    4
        BLKD    PAGE6+N1,*,ARO
;
        RET
;
;
;----- PRE-EMPHASIS -----
;      ARO : NUMBER OF POINTS      AR1 : INPUT BUFFER PTR
;                                  AR2 : OUTPUT BUFFER PTR
;
PREEMF: SAR     ARO,RS8      ;SAVE AR REGS. SO AUTO DOESN'T DESTROY THEM
        SAR     AR1,RS9
        SAR     AR2,RS10
;
        LRLK    ARO,PAGE6+RS1 ;POINTER TO OUTPUT ARRAY
        LAR     AR1,RS9      ;POINTER TO INPUT ARRAY
        LARK    AR2,1        ;NUMBER OF LAGS
        LAC     RS8          ;NUMBER OF POINTS IN INPUT ARRAY
        CALL    AUTO         ;GET 1 LAG FROM INPUT SIGNAL
;
        LAR     ARO,RS8      ;RESTORE AR REGS.
        LAR     AR1,RS9
        LAR     AR2,RS10
;
        ZAC
        SACL    LAST        ;LAST := 0
        SUB     RS2
        SACL    RS2         ;MU := -RS2 = -((R[1]/R[0]) * 32768)
;
        LARP    1
;
PRLP:  LT      LAST          ;T REG = LAST SAMPLE
        MPY     RS2          ;P REG = - (MU*LAST) * 2**14
        LAC     *,S14,AR2    ;ACC = CURRENT SAMPLE * 2**14
        SACH    LAST,S2      ;STORE LAST FOR NEXT ITERATION
        APAC
        SACH    *,S2,ARO      ;OUTPUT SAMPLE := CURRENT - MU*LAST
        BANZ    PRLP,AR1     ;LOOP UNTIL DONE
;
        RET
;
;
;----- WINDOW -----
;      ARO : NUMBER OF POINTS - 1      AR1 : INPUT ARRAY
;      ACC. : INDEX THRU WINDOW ARRAY      AR2 : OUTPUT ARRAY
;
WINDOW: SACL    INDEX        ;INDEX := STEP SIZE THRU WINDOW FUNCTION

```

```

        SAR    ARO,DR          ;DR    := # OF POINTS - 1
        SAR    AR2,TEMP        ;TEMP  := POINTER TO OUTPUT ARRAY
        LALK    WARRAY
        SACL    WPTR           ;WPTR  := PRAM POINTER TO WINDOW FUNCTION
;
; APPLY WINDOW FUNCTION BY USING TABLE LOOK-UP
;
        LARP    1
WLOOP:  LAC     WPTR
        TBLR    WIND           ;TABLE LOOK-UP WINDOW SAMPLE
        ADD     INDEX
        SACL    WPTR
;
        LT      WIND           ;T REG = WIND * 2**15
        MPY     *+,AR2         ;P REG = S[I] * WIND * 2**15
        PAC
        SACH    *+,S1,ARO      ;OUTPUT SAMPLE := S[I] * WIND
;
        BANZ    WLOOP,AR1      ;LOOP UNTIL DONE
;
; ZERO MEAN THE WINDOWED SIGNAL (SO MEAN DOESN'T EFFECT LATER RESULTS)
;
        LAR     ARO,DR         ;ARO   := # OF POINTS
        LAR     AR2,TEMP        ;AR2   := POINTER TO TOP OF OUTPUT ARRAY
        LAR     AR3,TEMP        ;AR3   := POINTER TO TOP OF OUTPUT ARRAY
;
        LAC     DR
        SACL    TEMP           ;TEMP  := # OF POINTS - 1
        ADD     ONE
        SACL    DR             ;DR    := (# OF POINTS - 1) + 1
;
        ZAC
        LARP    3
        RPT     TEMP           ;FOR I := 0 TO (# OF POINTS - 1)
        ADD     *+,AR3         ;    ACC. := ACC. + S[I]
;
        CALL    DIVNEW
        SACL    TEMP           ;TEMP  := - (RUNNING SUM / # OF POINTS)
;
        LARP    2
MLOOP:  LAC     *,AR2           ;FOR I := 0 TO (# OF POINTS - 1)
        ADD     TEMP           ;    S[I] := S[I] - MEAN
        SACL    *+,ARO
        BANZ    MLOOP,AR2
;
        RET
;
;
;----- PREFFT -----
;   ARO : POINTER TO INPUT ARRAY      AR1 : POINTER TO OUTPUT FOR FFT
;
PREFFT: SAR    ARO,DPTR        ;DPTR  POINTS TO NA1-NA11
        LAR     AR3,DPTR
        LACK    M2+1
        ADD     DPTR
        SACL    TEMP           ;TEMP  POINTS TO NA12-NA22
        LAR     AR4,TEMP
        LARK    AR2,M2
        LARP    3
SW1:    LAC     *+,AR4          ;COPY NA1-NA11 TO NA12-NA22
        SACL    *+,AR2         ;NA1,...,NA11
                                ;NA12,...,NA22

```

```

        BANC      SW1,AR3
;
; DOUBLE NA'S UNTIL 3FFF < LARGEST < 7FFF
; NOTE: NA1 := 1 = 2**12-1 = FFF - THUS NA'S CANNOT BE SHIFTED
;           MORE THAN THREE TIMES
;
DSTART: LAR      AR3,TEMP      ;MOVE FORWARDS THRU ARRAY
        LARK     AR4,M2       ;DO M2+1 TIMES
DOUBLE: LT      *,AR3         ;OLD NA12,...,NA22
        LAC      *,S1,AR3     ;2*OLD
        SACL     *,AR3       ;SAVE IT
        MPY      **+,AR4      ;OLD*NEW-OLD*(2*OLD)
        PAC
        BLZ      OVFL        ;--*+ OR +*- = -, ++* OR -*- = +
        BANC     DOUBLE,AR3
;
        LAR      AR3,DPTR     ;COPY NA12-NA22 TO NA1-NA11 IF NO OVERFLOW
        LAR      AR4,TEMP
        LARK     AR2,M2
        LARP     4
SW2:    LAC      **+,AR3       ;NA1 ...,NA22
        SACL     **+,AR2      ;NA1 ...,NA11
        BANC     SW2,AR4
        LARP     3            ;USE AR3 AT THE TOP OF DSTART
        B        DSTART
;
; GOOD RESULTS IN NA1-NA11 IF OVERFLOW DURING LAST PASS,
; LOAD NA'S INTO FIRST REAL PART OF FFT
; PAD THE REST OF THE PAGE WITH ZEROES
;
OVFL:   LAR      AR3,DPTR     ;INPUT
        LARK     AR2,M2      ;COUNTER
        LARP     3           ;USE AR3 FIRST
NALP:   LAC      **+,AR1      ; (0) INTO 0, (1) INTO 2, ..., (10) INTO 20
        SACL     **+,AR1
        ZAC
        SACL     **+,AR2      ; 0 INTO 1, ..., 17, 19, 21
        BANC     NALP,AR3     ;DECRIMENT COUNTER
;
        LARP     1
        ZAC      ; FILL REST OF PAGE WITH ZEROES
        RPTK     128-(2*M2)-1
        SACL     **+
;
        RET
;
;===== MAGNTD =====
; AR0 : INPUT BUFFER (REAL,COMPLEX)   AR1 : NUMBER OF COMPLEX POINTS
; AR2 : OUTPUT BUFFER
;
MAGNTD: LARP     0
;
STORE:  LAC      **+,AR0      ;PICK UP REAL
        ABS
        SACL     RXT
        LAC      **+,AR2      ;PICK UP IMAGINARY
        ABS
        SACL     IYT
        SUB      RXT
        BGZ      YGX

```



```

;
XGY:  LAC      RXT          ;REAL >= IMAGINARY
      SUB      IYT,S2
      BGZ      POS1        ;IF (RXT <= 4*IYT)
      ZALH     RXT          ; THEN MAG = (RXT - RXT/8) + IYT/2
      SUB      RXT,S13
      ADD      IYT,S15
      B        FINMAG
POS1:  ZALH     RXT          ; ELSE MAG = RXT + IYT/8
      ADD      IYT,S13
      B        FINMAG
;
YGX:  LAC      IYT          ;IMAGINARY > REAL
      SUB      RXT,S2
      BGZ      POS2        ;IF (IYT <= 4*RXT)
      ZALH     IYT          ; THEN MAG = (IYT - IYT/8) + RXT/2
      SUB      IYT,S13
      ADD      RXT,S15
      B        FINMAG
POS2:  ZALH     IYT          ; ELSE MAG = IYT + RXT/8
      ADD      RXT,S13
;
FINMAG: SACH     *+,S2,AR1    ;JUSTIFY MAG FOR DISPLAY - SHIFT LEFT 2
;
      BANZ     STORE,ARO     ;AND AGAIN
;
      RET
;
;
;----- NULL PICKER FOR FORMANTS -----
;  ARO : INPUT BUFFER PTR          AR1 : NUMBER OF POINTS
;  AR2 : PTR TO FORMANT RESULTS    AR3 : NUMBER OF FORMANTS PICKED
;
NPIC:  LARP      0
      SAR      ARO,TOP      ;SAVE TOP OF BUFFER FOR NORMALIZING
      LAC      *+,ARO       ;INITIALIZE BUFFER
      SACL     BSTRT1      ;FIRST POINT
      LAC      *+,ARO
      SACL     BSTRT2      ;SECOND POINT
      LAC      *+,ARO
      SACL     BSTRT3      ;THIRD POINT
      B        SPICK       ;SKIP BUFFER SHIFT ON 1ST PASS
;
;  SHIFT BUFFER HERE
;
CHKMIN: DMOV     BSTRT2      ;NEW 1ST POINT = OLD 2ND POINT
      DMOV     BSTRT3      ;NEW 2ND POINT = OLD 3RD POINT
      LAC      *+,AR2      ;NOW LOAD IN THE NEW POINT
      SACL     BSTRT3      ;GET NEW 3RD POINT
;
;  CHECK FOR A NULL POINT
;
SPICK:  LAC      BSTRT3
      SUB      BSTRT2
      BLEZ     ENDPK        ;IF 3RD POINT <= 2ND POINT THEN NO MIN FOUND
      LAC      BSTRT1
      SUB      BSTRT2
      BLEZ     ENDPK        ;IF 1ST POINT <= 2ND POINT THEN NO MIN FOUND
;
;  SAVE FORMANT LOCATION (INDEX CORRESPONDS TO FREQUENCY)
;

```

```

        SAR    ARO,DPTR      ;GET SCOPE BUFFER POINTER
        LAC    DPTR
        SUB    ONE,S1        ;MOVE CURRENT POINTER BACK 2 PLACES
        SUB    TOP           ;SUB TOP OF BUFFER
        SACL   *+,AR3        ;SAVE POINTER TO PEAK
        BANZ   ENDPK,AR1     ;DONE ALL THREE FORMANTS ?
        B      DONEPK       ; YES - THEN EXIT
;
ENDPK:  LARP    1
        BANZ   CHKMIN,ARO    ;DONE ALL BUFFER SHIFTS ?
;                               YES - THEN EXIT
DONEPK: RET
;
;
;----- BUBBLE SORT -----
; ARO : POINTER TO TOP OF ARRAY   AR1 : NUMBER OF POINTS IN ARRAY - 1
;
SORT:   SAR    ARO, TOP      ;INITIALIZATION
        SAR    AR1,COUNT
        LAC    COUNT
;
LP1:    SUB    ONE           ;REPEAT
        SACL   COUNT        ; ADJUST COUNTER FOR BANZ
        LAR    AR1,COUNT    ; LOOP COUNTER := (L - 1)
        LAC    TOP         ; COUNT := POINTER TO A(0)
        SACL   COUNT
        LAR    ARO, TOP     ; I := 1
;
LP2:    LAC    *+,ARO        ; FOR I := 1 TO (L - 1)
        SUB    *,ARO
        BLEZ   NEXT1        ; IF A(I) > A(I + 1) THEN
        LAC    *-,ARO       ; BEGIN
        SACL   TEMP         ; TEMP := A(I + 1)
        SAR    ARO,COUNT    ; COUNT := POINTER TO A(I)
        LAC    *+,ARO       ; A(I + 1) := A(I)
        SACL   *-,ARO
        LAC    TEMP        ; A(I) := TEMP
        SACL   *+,AR1       ; END
NEXT1:  LARP    1
        BANZ   LP2,ARO
;
        LAC    COUNT        ; ADJUST COUNT INTO [0 .. N-1] RANGE
        SUB    TOP
        BGZ    LP1          ;UNTIL (COUNT-1) = 0
;
        RET                ;RETURN
;
;----- FORMANT TRACKING ALGORITHMS -----
; AR2 : POINTER TO FEATURES   AR3 : POINTER TO CURRENT NULLS
;
FTRACK: LDPK    0            ;DELAY LINE OF NULLS IS ON PAGE 0
        LARP    3            ;USE CURRENT NULLS POINTER
;
        DMOV    F1B          ;F1C := F1B
        DMOV    F1A          ;F1B := F1A
        LAC     *+,AR3
        SACL    F1A          ;F1A := NEW PEAK N1
;
        DMOV    F2B          ;F2C := F2B

```

```

        DMOV    F2A            ;F2B := F2A
        LAC     **+,AR3
        SACL    F2A            ;F2A := NEW PEAK N2
;
        DMOV    F3B            ;F3C := F3B
        DMOV    F3A            ;F3B := F3A
        LAC     **+,AR2
        SACL    F3A            ;F3A := NEW PEAK N3
        LDPK    6
;
; CHECK IF FRAME IS VOICED OR UNVOICED
;
        LAC     **+,AR3
        BNZ     TF1            ;IF F0 (CURRENT PITCH) = 0 THEN
        LARP    2              ; FRAME IS UNVOICED AND ALL FORMANTS
        ZAC     ;              ARE ZERO
        SACL    **+,AR2
        SACL    **+,AR2
        SACL    "
        RET
;
; TRACK F1
;
TF1:    LARK    AR3,PAGE0+F1A    ;POINTER TO DELAYED NULLS
        LAC     **+,AR3          ;LOAD F1 NULL ARRAY INTO SORTING ARRAY
        SACL    A1
        LAC     **+,AR3
        SACL    A2
        LAC     **+,AR0          ;USE AR0 FOR SORT
        SACL    A3
;
        LRLK    AR0,PAGE6+A1    ;POINTER TO TOP OF ARRAY TO BE SORTED
        LARK    AR1,3-1         ;NUMBER OF POINTS IN ARRAY - 1
        CALL    SORT            ;SORT ARRAY FROM LOWEST TO HIGHEST
;
        LARP    2
        LAC     A2              ;A2 IS THE MEDIAN
        SACL    **+,AR3         ;F1 := MEDIAN
;
; TRACK F2
;
        LAC     **+,AR3          ;LOAD F2 NULL ARRAY INTO SORTING ARRAY
        SACL    A1
        LAC     **+,AR3
        SACL    A2
        LAC     **+,AR0          ;USE AR0 FOR SORT
        SACL    A3
;
        LRLK    AR0,PAGE6+A1    ;POINTER TO TOP OF ARRAY TO BE SORTED
        LARK    AR1,3-1         ;NUMBER OF POINTS IN ARRAY - 1
        CALL    SORT            ;SORT ARRAY FROM LOWEST TO HIGHEST
;
        LARP    2
        LAC     A2              ;A2 IS THE MEDIAN
        SACL    **+,AR3         ;F2 := MEDIAN
;
; TRACK F3
;
        LAC     **+,AR3          ;LOAD F3 NULL ARRAY INTO SORTING ARRAY
        SACL    A1
        LAC     **+,AR3

```

```

SACL      A2
LAC      *,ARO          ;USE ARO FOR SORT
SACL      A3

;
LRLK     ARO,PAGE6+A1   ;POINTER TO TOP OF ARRAY TO BE SORTED
LARK     AR1,3-1        ;NUMBER OF POINTS IN ARRAY - 1
CALL     SORT           ;SORT ARRAY FROM LOWEST TO HIGHEST

;
LARP     2
LAC      A2             ;A2 IS THE MEDIAN
SACL     *              ;F3 := MEDIAN

;
RET

;
;
;----- AUTOCORRELATION -----
; ARO - POINTER TO OUTPUT ARRAY      AR1 - POINTER TO INPUT/DRAM ARRAY
; AR2 - NUMBER OF LAGS              AR3,AR4 - GP REGISTERS
; ACC - NUMBER OF POINTS IN INPUT ARRAY
;
AUTO:    SACL      COUNT          ;NUMBER OF POINTS IN INPUT ARRAY
;
; TRANSFER INPUT ARRAY FROM DRAM TO PRAM
;
LARP     4
SAR      AR1,TEMP
LAR      AR4,TEMP        ;ADDRESS OF TOP OF INPUT ARRAY
RSXM
LALK     PRAM            ;TURN OF SIGN EXTENSION FOR LONG LOAD
SSXM
RPT      COUNT
TBLW     **+,AR4         ;PERFORM TRANSFER

;
; COMPUTE SUM OF SQUARES - WHICH GIVES UNNORMALIZED R[0] = ENERGY
;
LARP     1
SAR      AR1,TEMP        ;SAVE TOP OF DRAM ARRAY PTR
MAR      **+,AR4         ;INCRIMENT DRAM ARRAY PTR FOR NEXT ITERATION
LAR      AR4,TEMP        ;LOAD ADDRESS OF TOP OF DRAM
MPYK     0               ;INITIALIZE MULTIPLIER
ZAC
RPT      COUNT
MAC      PRAM,**+,AR4    ;MULTIPLY ACCUMULATE = SUM OF SQUARES
APAC     ;GIVES INITIAL (UNNORMALIZED) R[0]
SACH     ENRHI
SACL     ENRLO           ;R[0] := ENERGY USED LATER BY DECIS

;
; NORMALIZE R[0] TO CALCULATE SCOUNT AND ZMASK FOR NORMALIZING THE REST OF
; THE AUTO RESULTS, XLOW AND XHIGH ARE ALREADY IN ACCUMULATOR
;
LARK     AR4,0           ;ZERO EXPONENT
RPTK     31D            ;SHIFT R[0] TILL A "1" IN BIT 30
NORM     **+,AR4
SACH     XHIGH          ;STORE SHIFTED R[0]
SACL     XLOW
SAR      AR4,SCOUNT     ;FINAL RESULT FOR SCOUNT

;
SACH     DR              ;DETERMINE ZMASK
LALK     32767D,S14      ;R[0] WILL BE NORMALIZED TO CONSTANT
CALL     DIVNEW
ABS      ;ZMASK = (32767 * 2**14 / R(0))

```

```

;          SACL    ZMASK          ;FINAL RESULT FOR ZMASK
;
;          LARP    0              ;NORMALIZE R[0] TO A CONSTANT
;          LT      XHIGH          ;LOAD SHIFTED R[0]
;          MPY     ZMASK          ;(SHIFTED R[0])*ZMASK/2**15 = 32767/2 = 3FFF
;          PAC
;          SACH    *+,S1,AR2      ;FINAL NOMALIZED R[0] = 3FFF
;
;          BANZ    OUTER,AR1      ;CHECK LAG COUNTER FIRST TIME AROUND
;          B       NFIN
;
; R[0] IS DONE, NOW DO THE REST
;
OUTER: SAR      AR1,TEMP
      MAR      *+,AR4          ;INCRIMENT DRAM ARRAY PTR FOR NEXT ITERATION
      LAR      AR4,TEMP        ;POINT TO TOP OF DRAM ARRAY + LAG
      LAC      COUNT          ;DECRIMENT THE # OF MULT/ACC. COUNTER
      SUB      ONE
      SACL     COUNT
      MPYK     0              ;INITIALIZE MULTIPLIER
      ZAC
INNER: RPT      COUNT
      MAC      PRAM,*+,AR4      ;MULTIPLY ACCUMULATE GIVES R[LAG]
      APAC
      SACH     XHIGH          ;STORE RESULTS
      SACL     XLOW
;
; NORMALIZE THE AUTOCORRELATION RESULT
;
      LT      SCOUNT          ; LOAD SHIFT COUNT INTO T REGISTER
      LAC     SCOUNT
      SUB     ONE,S4
      BLZ     NORM2
;
NORM1: LACT     XLOW          ; CASE 1: 16 <= SHIFT COUNT <= 31
      SACL     XHIGH
      B       NORM0
;
NORM2: LACT     XHIGH          ; CASE 2: 0 <= SHIFT COUNT <= 15
      SACL     XHIGH
      ZALH     XHIGH
      RSXM
      ADDT     XLOW
      SSXM
      SACH     XHIGH
;
NORM0: LARP     0              ;PERFORM FINAL NORMALIZATION TO CONSTANT
      LT      XHIGH          ; R[I]/2 < R[I]*( )/2**15 < R[I]
      MPY     ZMASK          ;AS 2**14 < ( ) < 2**15-1
      PAC
      SACH     *+,S1,AR2      ;STORE RESULT
;
      BANZ    OUTER,AR1      ;LOOP FOR NUMBER OF LAGS
;
NFIN: RET
;
;----- THE LG ALGORITHM -----
;
LG:   LARP     0              ;ARP = 0; DP = 0
;

```

```

LAC      RX1          ;R(1) *2**15
SACL     DR           ;DIVISOR = R(1) *2**15
;
IAC      RX2,S15      ;DIVIDEND = R(2) *2**30
CALL     DIVNEW       ;(R(2)/R(1)) *2**15
SACL     FK1          ;FK(1)=(-R2/R1) *2**15
;
LAC      RX2          ;R(2) *2**15
SACL     XD1          ;D(1) = R(2) *2**15
;
LAC      RX1
SACL     DR           ;DIVISOR=R(1) *2**15
;
LT       RX2          ;R(2) *2**15
MPY      RX2          ;(R(2))**2 *2**30
PAC      ;DIVIDEND = (R(2))**2 *2**30
CALL     DIVNEW       ;((R(2)**2)/(R(1))) *2**15
ADD      RX1          ;(R(1) - (R(2)**2/(R(1)))) *2**15
SACL     XD2          ;D(2) = (R(1) - (R(2)**2/R(1))) *2**15
;
LG1      RO          I,J
LAC      RX&I,S15
SACH     B1,X1
MPYK     0
LRLK     ARO,PAGE6+FK1
LRLK     AR1,PAGE6+XD1
CALL     LG&J&S
ENDM
;
J=1
I=3
REPT     M1-1
LG1      %I,%J
I=I+1
J=J+1
ENDM
;
LAC      LGFLAG      ;WHERE TO RETURN? - STACK TOO DEEP FOR RET
BZ       OUTLG       ;      ORDER OF 4 FOR PITCH
;
REPT     (M2-1)-(M1-1)
LG1      %I,%J
I=I+1
J=J+1
ENDM
;
OUTLG:   RET          ;      ORDER OF 10 FOR ENVELOPE
;
LG2      MACRO J
MAR      **+,AR1
MAR      **+,ARO
LG&J&S:
LTA      **+,ARO
MAR      **+,AR1
SACH     Y,X1
LAC      **+,AR1,S15
MPY      Y
APAC
SACH     *-,AR1,X1
LAC      Y,S15
MPY      *-,AR1

```

```

        DMOV    **+,AR1
        ENDM

;
        J=M2-1
        REPT    M2-1
        LG2     %J
        J=J-1
        ENDM

;
LGOS:   APAC
        SACH    Y,X1
        MAR     **+,AR1
        MAR     **+,AR1
        LAC     *,AR0
        SACL    DR                ;DIVISOR = D(I) *2**15
        LAC     Y,S15             ;DIVIDEND= Y *2**30
        CALL    DIVNEW            ;(Y/D(I)) *2**15
        SACL    *,AR1             ;FK(I) = -(Y/D(I)) *2**15
        LAC     *-,AR0,S15        ;D(I) *2**30
        LT      *,AR1             ;Y *2**15
        MPY     Y                 ;(Y * FK(I)) *2**30
        APAC
        DMOV    **+,AR1          ;(D(I) + Y * FK(I)) *2**30
        MAR     **+,AR1
        MAR     **+,AR1
        SACH    *,AR0,X1         ;D(I+1) = (D(I) + Y * FK(I)) *2**15
        RET

;
;----- TRANSFORM K'S TO A'S -----
;
;  W A R N I N G :  NA'S MUST BE IN CONTIGUOUS ARRAY
;
        ST1B    MACRO    I,IP,IQ,MN2
        LRLK    ARO,PAGE6+NA&IQ    ;ADDR OF NA(IQ)
        LRLK    AR1,PAGE6+NA&IP    ;ADDR OF NA(IP)
        LARK    AR2,(MN2-2)        ;NUMBER OF LOOPS INTO ACC FOR CALL
        LAC     FK&I               ;PASS FK'I IN TEMP
        CALL    LPI                ;CALL CALC SUB
        ENDM

;
        ST2B    MACRO    XIFLAG,I,P1,P2
;
        IFE     XIFLAG             ;THIS DONE ONLY FOR EVEN VALUES OF I
        LAC     NA&P1,S15          ;NA&P1*2**15
        LT      NA&P1
        MPY     FK&I               ;NA&P1*[FK&I*2**15]
        APAC
        SACH    NA&P1,X1           ;NA&P1*(1+FK&I)*2**15
        ENDIF

;
        LAC     FK&I,S13           ;[FK&I*2**15]*2**13
        SACH    NA&P2              ;[FK&I*2**26]/2**16=FK&I*2**12
        ENDM

;
LPI:    SACL    TEMP
        LARP    1
LPT:    LAC     *,AR0,S15          ;NA&IP*2**15
        LT      TEMP              ;FK&I*2**15
        MPY     *,AR1              ;P := FK&I*NA&IQ*2**15
        LTA     *,AR1              ;T := NA&IP, ACC := [NA&IP+FK&I*NA&IQ]*2**15

```

```

        SACH    *+,AR0,X1      ;NA&IP := [NA&IP+FK&I*NA&IQ]*2**15
        LAC     *,AR0,S15      ;NA&IQ*2**15, NA&IP STILL IN T REG
        MPY     TEMP           ;P := NA&IP*FK&I*2**15
        APAC    ;ACC := [NA&IQ+FK&I*NA&IP]*2**15
        SACH    *-,AR2,X1      ;NA&IQ := [NA&IQ+FK&I*NA&IQ]*2**15 (OLD)
        BANZ    LPT,AR1        ;DECRIMENT LOOP COUNT
        RET

;
;-----
;
KTOA:   LAC     ONE,S12
        SUB     ONE
        SACL    NA1            ;NA1 := 2**12-1 = 0FFFH = 1 IN OUR CONVENTION
;
        LT      FK1
        MPY     FK2            ;FK1*FK2*2**30
        PAC
        ADD     FK1,S15        ;[FK1+FK1*FK2*2**15]*2**15
        SACH    TEMP
        LAC     TEMP,S14
        SACH    NA2            ;NA2 := [FK1+FK1*FK2]*2**12
;
        LAC     FK2,S13
        SACH    NA3            ;NA3 := FK2*2**12
;
        I=3
        REPT    (M1-2)         ;TRANSFORM FIRST FK'S FOR PITCH AND ENV.
        MN2=(I-1)/2+1
        IQ=I
        IP=2
        ST1B    %I,%IP,%IQ,%MN2
        XIFLAG=2*(I/2)-I
        P1=MN2+1
        P2=I+1
;
        ST2B    %XIFLAG,%I,%P1,%P2
;
        I=I+1
        ENDM
;
        LAC     LGFLAG          ;IF FLAG=1 THEN ENV
        BZ      RETURN          ;IF FLAG=0 THEN PITCH
;
CONTINU:
        REPT    (M2-2)-(M1-2)   ;TRANSFORM THE REMAINING K'S FOR ENV ONLY
        MN2=(I-1)/2+1
        IQ=I
        IP=2
        ST1B    %I,%IP,%IQ,%MN2
        XIFLAG=2*(I/2)-I
        P1=MN2+1
        P2=I+1
;
        ST2B    %XIFLAG,%I,%P1,%P2
;
        I=I+1
        ENDM
;
RETURN: RET
;
;

```



```

;----- DIVISION -----
;
DIVNEW: SACH   DDH
        SACL   DDL
        ZAC
        SACL   TEMP2
        LAC    ONE,S15
        SUB    ONE
        SACL   PLUS1
;
        LAC    ONE,S15
        AND    DDH
        BNZ    LESS0
        ZALH   DDH
        ADDS   DDL
;
        REPT   16D
        ADD    TEMP2
        SUBC   DR
        ENDM
;
        AND    PLUS1
        SACL   TEMP2
        ZAC
        SUB    TEMP2
        RET
;
LESS0:  LACK   0
        SUBH   DDH
        SUBS   DDL
;
        REPT   16D
        ADD    TEMP2
        SUBC   DR
        ENDM
;
        AND    PLUS1
        RET
;
;----- FILTER AND WINDOW COEFFICIENTS -----
;
; LOW-PASS FILTER COEFFICIENTS * 2**14
;
FTABLE: DW      11712,  4672,  22846, -9517,  3056,  23099, -13315, 6600
;
; 746 POINT HAMMING WINDOW * 2**15
;
WARRAY: DW      2621,  2626,  2641,  2666,  2700,  2744,  2798,  2862
        DW      2935,  3018,  3110,  3212,  3324,  3445,  3575,  3714
        DW      3863,  4020,  4186,  4362,  4546,  4738,  4939,  5148
        DW      5366,  5591,  5825,  6066,  6314,  6571,  6834,  7104
        DW      7382,  7666,  7956,  8253,  8557,  8866,  9181,  9501
        DW      9827,  10158, 10494, 10834, 11179, 11528, 11882, 12239
        DW      12599, 12963, 13330, 13700, 14073, 14447, 14824, 15203
        DW      15584, 15966, 16349, 16732, 17117, 17502, 17887, 18272
        DW      18656, 19040, 19423, 19805, 20185, 20564, 20941, 21316
        DW      21688, 22058, 22425, 22789, 23150, 23507, 23860, 24209
        DW      24554, 24895, 25231, 25561, 25887, 26208, 26523, 26832
        DW      27135, 27432, 27723, 28007, 28284, 28554, 28818, 29074
        DW      29323, 29564, 29797, 30023, 30240, 30449, 30650, 30843

```

```

DW      31027, 31202, 31366, 31526, 31674, 31814, 31944, 32065
DW      32176, 32278, 32371, 32453, 32527, 32590, 32644, 32688
DW      32723, 32747, 32762, 32767, 32762, 32747, 32723, 32688
DW      32644, 32590, 32527, 32453, 32371, 32278, 32176, 32065
DW      31944, 31814, 31674, 31526, 31368, 31202, 31027, 30843
DW      30650, 30449, 30240, 30023, 29797, 29564, 29323, 29074
DW      28818, 28554, 28284, 28007, 27723, 27432, 27135, 26832
DW      26523, 26208, 25887, 25561, 25231, 24895, 24554, 24209
DW      23860, 23507, 23150, 22789, 22425, 22058, 21688, 21316
DW      20941, 20564, 20185, 19805, 19423, 19040, 18656, 18272
DW      17887, 17502, 17117, 16732, 16349, 15966, 15584, 15203
DW      14824, 14447, 14073, 13700, 13330, 12963, 12599, 12239
DW      11882, 11528, 11179, 10834, 10494, 10158, 9827, 9501
DW      9181, 8866, 8557, 8253, 7956, 7666, 7382, 7104
DW      6834, 6571, 6314, 6066, 5825, 5591, 5366, 5148
DW      4939, 4738, 4546, 4362, 4186, 4020, 3863, 3714
DW      3575, 3445, 3324, 3212, 3110, 3018, 2935, 2862
DW      2798, 2744, 2700, 2666, 2641, 2626
;
FFTBUF: DW      128      DUP (?)      ;TEMPORARY BUFFER FOR THE FFT
;
;-----
;
ENDING: DW 0
        MAIN      ENDS
        END        START

```

END

17|09.9|0

FIN