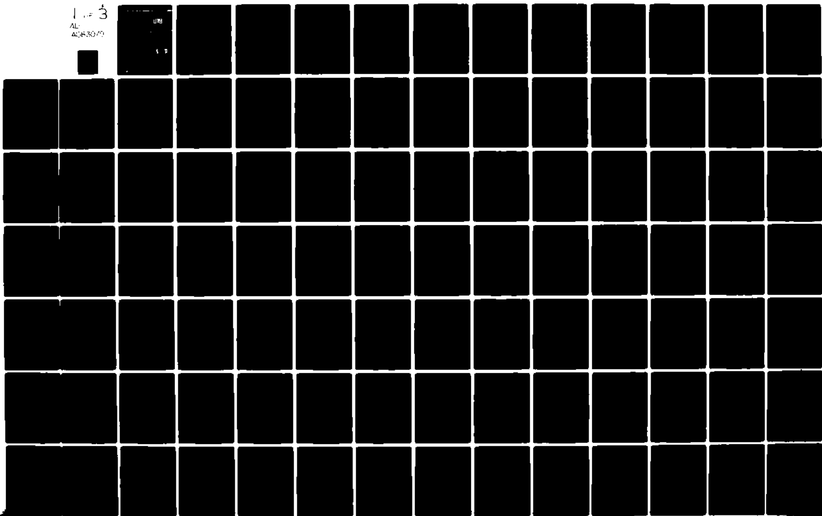
AD-A083 079 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR--ETC(U)
FEB 80 R VISWANATHAN, J WOLF, L COSELL DCA100-79-C-0003
UNCLASSIFIED BBN-4327-VOL-1 NL

1 of 3
AD
A083079



Bolt Beranek and Newman Inc.

Report No. 4327

LEVEL

Design and Real-Time Implementation of a Baseband I for Speech Transmission Over 9600 Bps Noisy Chann

Volume I: Description

Final Report

February 1980

Prepared for:
Defense Communications Agency

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

Field	Entry
1. REPORT NUMBER	BBN Report No. 4327
2. GOVT ACCESSION NO.	
3. RECIPIENT'S CATALOG NUMBER	9
4. TITLE (and Subtitle)	6 DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR SPEECH TRANSMISSION OVER 9600 BPS NOISY CHANNELS. Volume I. Description.
5. TYPE OF REPORT & PERIOD COVERED	Final Report. Nov [redacted]78 — Feb [redacted]80
6. PERFORMING ORG. REPORT NUMBER	BBN Report No. 4327
7. AUTHOR(s)	10 R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins and W. Russell
8. CONTRACT OR GRANT NUMBER(s)	15 DCA100-79-C-0003
9. PERFORMING ORGANIZATION NAME AND ADDRESS	Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02138
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	14 BBN-4327-Vol-1
11. CONTROLLING OFFICE NAME AND ADDRESS	Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305
12. REPORT DATE	11 Feb [redacted]980
13. NUMBER OF PAGES	243 (12) 255
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	
15. SECURITY CLASS. (of this report)	Unclassified.
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

DISTRIBUTION STATEMENT A

Approved for public release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

speech coding, 9600 bps speech transmission, voice-excited coder, baseband coder, linear prediction, high-frequency regeneration, digital voice terminal, real-time speech coder.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design and development of a real-time baseband LPC speech coder that transmits high-quality speech over a 9600 bps synchronous channel with bit-error rates of up to 1%. Presented are the results of our investigation of a number of aspects of the baseband LPC coder with the goal of maximizing the quality of the transmitted speech. Important among these aspects are: baseband width, baseband coding, → next page

cont'd.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (cont'd.)

high-frequency regeneration, and error-protection of important transmission parameters. The report also includes the system design, detailed documentation, and program listings of the MAP-300 real-time implementation of the optimized speech coder.

This report is bound in two volumes. Volume I contains the text of the report, and Volume II contains the program listings of the MAP-300 speech coder implementation.

ACCESSION for		
NTIS	White Section	☑
DDC	Buff Section	☐
UNANNOUNCED		☐
JUSTIFICATION		
BY		
DISTRIBUTION/AVAILABILITY CODES		
Dist.	AVAIL. and/or SPECIAL	
A		

Report No. 4327

DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR SPEECH TRANSMISSION OVER 9600 BPS NOISY CHANNELS

Final Report

Volume I: Description

Authors: R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins, and W. Russell

February 1980

Prepared for:
The Defense Communications Agency

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

List of Figures

List of Figures (continued)

List of Figures (continued)

List of Tables

ACKNOWLEDGMENTS

The authors wish to thank: J. Makhoul for significantly contributing to the writing of the proposal which led to this project; M. Berouti, for his work on entropy coding (Section 2.8.4) and speech enhancement preprocessor (Section 2.15.2); A. W. F. Huggins for conducting the formal subjective speech-quality test reported in Section 2.12; and G. Moran, the COTR of this project, for suggesting the engineering channel-error performance criterion stated in Section 2.13.

1. INTRODUCTION

The purpose of this work was the design and development of a real-time speech coding system that produces high quality speech at a data rate of 9600 bits per second (bps). The input speech of the coder has a bandwidth of 3200 Hz. The encoder and decoder of the speech coder operate independently, with the encoder mapping the analog input signal into an output binary sequence and the decoder mapping the binary sequence into the corresponding analog output speech. In addition to the requirement that the speech coder in general produce speech of very good quality in the sense that it has a very high degree of user acceptance, there are several specific requirements on the coder performance as given below:

1. Noisy channel: Produce good quality speech under conditions of a transmission bit error rate of up to 1%.
2. Acoustic background noise: Produce intelligible speech under conditions of acoustic background noise with a sound pressure level (SPL) of 60 dB re 20 micronewtons per square meter.
3. Tandem operation with CVSD coder: Perform satisfactorily in tandem with a CVSD speech coder operating at a data rate of 16 kbps. The tandem link should provide speech intelligibility with minimal degradation compared with a single link of 16 kbps CVSD coder alone.

For the speech coding algorithm, we chose the baseband residual coder, also known as the voice-excited coder. The speech coder was developed in two distinct phases. During the algorithm development phase, a number of aspects of the baseband residual coder were investigated in detail and optimized to the requirements stated above. In the subsequent implementation phase, the final optimized speech coder algorithm was implemented as a real-time full-duplex system on a CSP Inc. MAP-300 signal processing computer and associated hardware. The complete system was then delivered to Defense Communications Engineering Center, Reston, VA, where it was demonstrated.

The optimized speech coder algorithm may be summarized as follows. In the transmitter, the analog input speech is lowpass filtered at 3.2 kHz, sampled at 6.621 kHz, and divided into frames of about 27.2 ms duration. Spectral shape is characterized by an 8-pole LPC analysis using the autocorrelation method. The speech is inverse-filtered, and the resulting residual is lowpass filtered to 1/3 its original bandwidth, and down-sampled to 1/3 its original sampling rate. The extracted baseband residual is then coded by a 1-tap adaptive predictive coder that removes redundancy due to the pitch periodicity. The LPC parameters, energy, pitch, pitch tap, and baseband residual samples are quantized, coded, partially error-protected, and transmitted.

In the receiver, the baseband residual is reconstituted by a pitch-synthesis filter, and an approximation to the fullband residual is produced by a high-frequency regeneration method called perturbed spectral folding. This residual signal then forms the excitation for the linear prediction 8-pole spectrum synthesis filter, whence it is D/A converted and lowpass filtered at 3.2 kHz to produce the analog output speech signal.

In the body of this final report, we present: the results of our work on the development and optimization of the 9600 bps speech coding algorithm (Chapter 2); detailed documentation of the MAP-300 real-time implementation of the speech coding algorithm (Chapter 3); and instructions on the installation and use of these programs (Chapter 4). Contained in appendices are: the Equipment Description of the Speech Processor Interface (Appendix A); function descriptions of BBN-written MAP-300 modules (Appendix B); documentation of the MAP-300 buffers and scalars used in the speech coder (Appendices C and D); and listings of the MAP-300 and PDP-11 (FORTRAN) programs that constitute the real-time speech coder system (Appendix E, which makes up Volume II of this report).

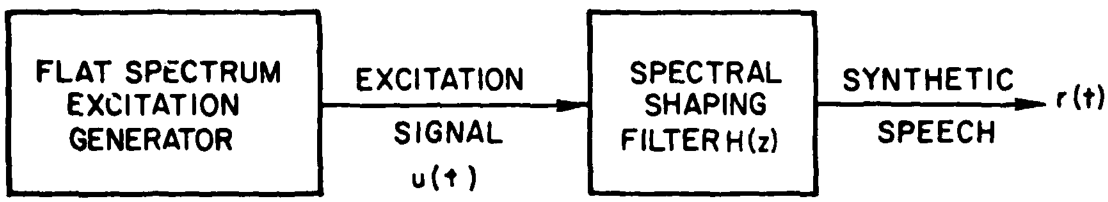
2. SPEECH-CODING ALGORITHM DEVELOPMENT AND OPTIMIZATION

2.1 INTRODUCTION

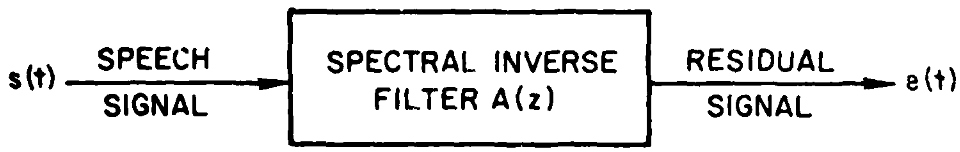
In this chapter, we describe the results of our work on the development and optimization of a 9600 bps speech-coding algorithm. The organization of the chapter is as follows. Section 2.2 provides our rationale for choosing the voice-excited or baseband coder (BBC) for further development and speech-quality optimization, to meet the requirements of this project stated in Chapter 1. A block-diagram description of the BBC system is given in Section 2.3. In Section 2.4, we briefly review previous work on BBC coders. The next six sections deal with different aspects of the BBC system that we considered in our investigation: sampling rate of the input speech (Section 2.5); estimation and coding of the spectral parameters (Section 2.6); extraction of the baseband from the fullband signal (Section 2.7); methods of quantizing and coding the baseband (Section 2.8); five techniques for regenerating the highband from the transmitted baseband (Section 2.9); and the speech-quality effect of short-term dc bias at each of several places within the coder (Section 2.10). While Sections 2.6, 2.8, and 2.9 contain some results of speech-quality optimization with respect to specific parameters, we present in Section 2.11 the results of overall optimization of speech quality of the 9600 bps BBC system, for the case of error-free transmission. Based on these results, we chose six BBC systems, which were then compared using a formal subjective speech quality test. The details of the six coders, the test, and the results are all given in Section 2.12. In Section 2.13, we treat the problem of transmission over a noisy channel with bit-error rates of up to 1% and present the results of our work to determine the amount and type of error-protection of the transmitted data, which leads to a graceful speech-quality degradation in the presence of channel bit-errors and which satisfies an engineering channel-error performance criterion stated in that section. Section 2.14 deals with the topic of acoustic background noise, while Section 2.15 treats the tandem operation of the optimized BBC system with a 16 kbps CVSD coder. Finally, in Section 2.16, we summarize the details of the optimized 9600 bps BBC system.

2.2 RATIONALE FOR CHOOSING THE BASEBAND CODER (BBC)

For the classes of speech coders we considered, the synthesis model employed at the receiver is shown in Fig. 2-1(a). In this figure, the synthetic or reconstructed speech signal r(t) is generated as the result of applying a time-varying flat-spectrum excitation signal u(t) as input to a time-varying spectral shaping filter H(z). The spectral envelope of the synthetic speech is completely determined by the spectral shaping filter. The parameters of the synthesis model, i.e., the excitation and the



(a)



(b)

Fig. 2-1 General synthesis model for most speech coders operating in the 2-16 kbps range.

(a) Implementation at the receiver

(b) Analysis at the transmitter to obtain the model parameters of the excitation signal u(t) filter, must be computed and transmitted periodically by the transmitter. Those parameters that represent the speech spectrum, denoted spectral parameters, are computed, quantized and transmitted every 10-30 ms. The type and frequency of transmission of excitation parameters, on the other hand, depends on the type of coder employed.

Figure 2-1(b) shows the analysis to be performed at the transmitter, when the receiver employs the above synthesis model. Notice that A(z) is proportional to 1/H(z). Although some coders may not explicitly compute the residual signal e(t) as shown in Fig. 2-1(b), it is convenient for our purpose here to consider the excitation model for u(t) as being based on the residual e(t). There are three excitation models, which lead to three classes of speech coders: residual-excited (or waveform) coders, pitch-excited coders, and baseband (or voice-excited) coders. Residual-excited coders represent one extreme, since they use the model u(t) = e(t) and transmit every sample of the residual waveform. Examples of these coders are: adaptive predictive coders (APC), adaptive transform coders (ATC), and sub-band coders. High-quality speech can be transmitted using any of these coders at 16 kbps [2]. Pitch-excited coders represent the other extreme in that they employ a binary pulse/noise source, one that is a sequence of pulses separated by the pitch period for voiced sounds and a random noise sequence for unvoiced sounds. The linear predictive coder (LPC) is an example of a pitch-excited coder that is used for transmitting speech in the range of 2-4 kbps. Baseband coders (BBC) represent a compromise between these two extremes. Since the low frequencies of the residual carry the most important information about the voicing characteristics of the speech, baseband coders transmit a low-frequency part (or baseband) of the residual e(t) and regenerate at the receiver the fullband excitation signal u(t) from the transmitted baseband.

Decreasing the data rates of residual-excited coders to below 10 kbps results in speech that is noisy, in general. On the other hand, increasing the data rate of a pitch-excited coder beyond 4 kbps does not substantially improve the speech quality. Since a baseband coder makes efficient use of the available 9.6 kbps data rate by transmitting only the important part of the frequency band, we chose to investigate BBC algorithms in this project. A BBC system can transmit the baseband of either the residual signal or the speech signal. Since our work on another government contract showed that baseband residual coders produced, in general, better speech quality than baseband speech coders [3], our work in this project was performed exclusively on baseband residual coders.

2.3 BLOCK DIAGRAM OF A BBC SYSTEM

Figure 2-2 shows the overall block diagram of a BBC system based on the linear prediction method of spectral analysis. In the figure, we assume that the bandwidth of the speech signal is W Hz and that the width of the baseband is B Hz (B<W). In the work reported here, we have assumed that the ratio W/B is an integer, denoted by L and called the "number of bands". The terms in parentheses below the signal path in Fig. 2-2 represent the Nyquist frequency (half the sampling rate) at each of those points. Along the lower branch of the transmitter in Fig. 2-2(a), LPC parameters are extracted, quantized, and encoded. Along the upper branch, the linear prediction residual is computed by inverse-filtering the speech signal with the filter A(z) given by

$$A(z) = 1 + \sum_{k=1}^{p} a(k)\ z^{-k}, \qquad (2.1)$$

where a(k), $1 \leq k \leq p$, are the predictor parameters and p is the LPC order. A baseband is then extracted from the residual. Baseband extraction usually takes the form of a lowpass filter. The extracted baseband signal is then decimated (down-sampled) to its Nyquist rate, quantized, and coded. The encoded LPC parameters and baseband signal are multiplexed and transmitted via a communication channel.

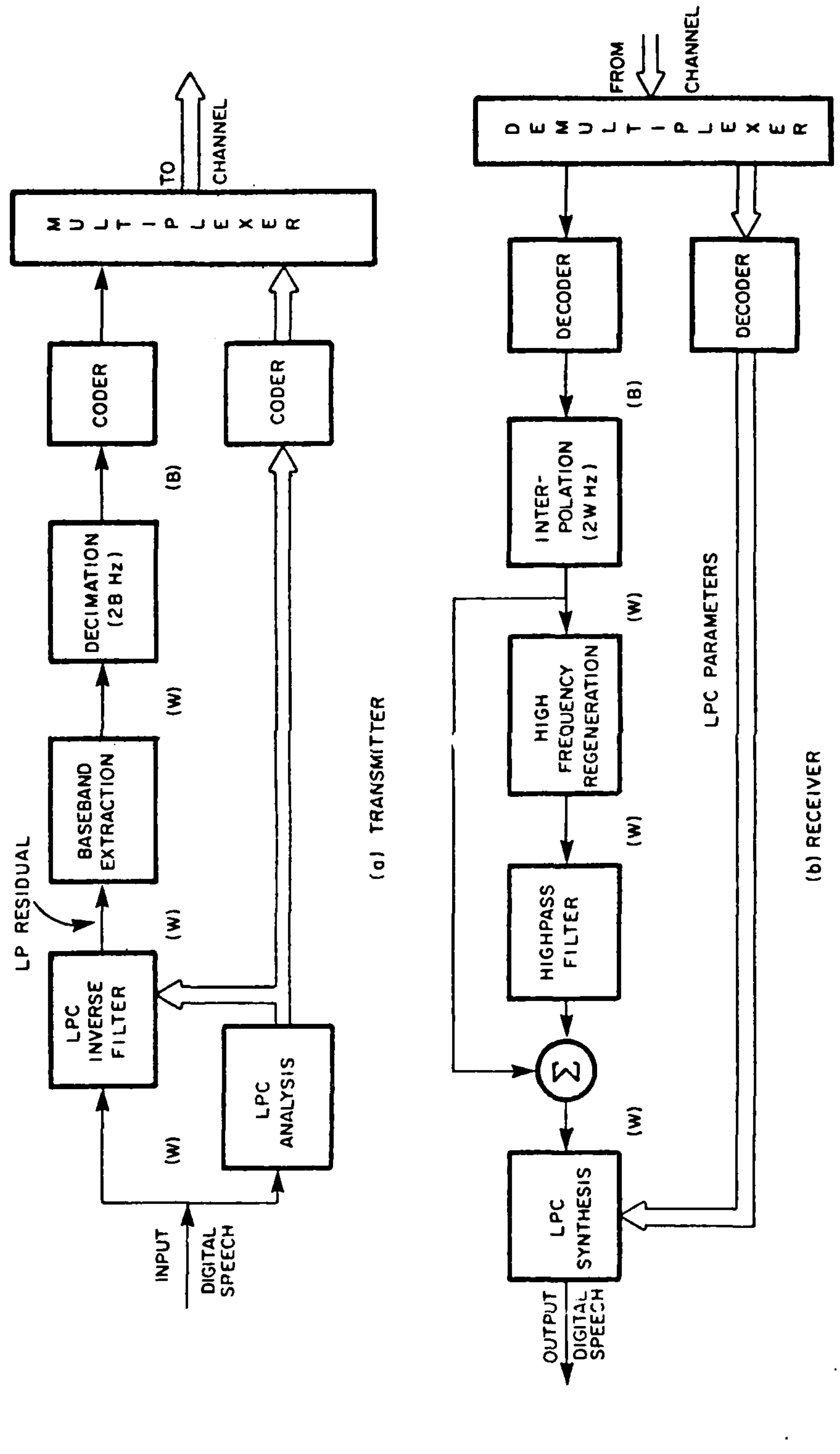


Fig. 2-2 Block diagram of a baseband residual coder

The receiver of the BBC coder is depicted in Fig. 2-2(b). The received data are demultiplexed and decoded into the baseband signal (top branch) and LPC parameters (bottom branch). High-frequency regeneration (HFR) operates on the baseband to produce a signal with a fullband width W. The HFR generally involves some nonlinear operation on the baseband (such as waveform rectification), followed by spectral flattening of the distorted signal for the purpose of LPC synthesis. The baseband part of the nonlinearly regenerated fullband signal is, in general, distorted relative to the received baseband. This problem is corrected by a procedure called baseband reintroduction, as shown in Fig. 2-2(b). The output of the HFR process is highpass filtered at a cutoff frequency of B Hz and added to the received baseband signal. The resulting signal is then applied to the all-pole LPC synthesizer $H(z) = 1/A(z)$, to produce the reconstructed output speech.

The data rate and the output speech quality of a BBC system, with a given input speech bandwidth W, are determined by the following items:

(a) estimation and coding of the LPC parameters;

(b) baseband width B, or equivalently the number of bands, $L=W/B$;

(c) coding of baseband residual;

(d) method of high-frequency regeneration; and

(e) type and amount of error protection of transmission parameters.

Before we consider these items in detail, we give a brief review of the literature on baseband or voice-excited coders.

2.4 PREVIOUS WORK

The idea of voice excitation was introduced by Schroeder and David [4-6] around 1960 in the context of developing a high-fidelity channel vocoder, as a way of overcoming the problems associated with binary pulse/noise excitation. In [6], a low-frequency band of the speech signal (250-940 Hz) is transmitted as the baseband. The higher frequencies (940-3650 Hz) are modeled by 17 vocoder channels, representing the spectral envelope for those frequencies. At the receiver, the excitation at higher frequencies is obtained by passing the baseband through a nonlinear distortion process followed by a spectral flattening process. The resulting excitation is applied to the channel synthesizer, and the outputs are added to the baseband to produce the synthetic speech signal. The nonlinear distortion and spectral flattening were realized by passing the baseband through a rectifier, followed by peak clipping. This general method, with minor variations, has been basically the only HFR method used in most investigations. A digital version of a voice-excited 16-channel vocoder operating at 9.6 kbps has been developed by Gold and Tierney [7].

The use of baseband excitation in LPC coders has been investigated by Un and Magill, who transmitted the baseband residual [8], and by Weinstein, who transmitted the baseband speech [9]. The first of these coders, called RELP (for residual-excited linear prediction), was designed to operate at a data rate of 9.6 kbps; the second coder, called VELP (for voice-excited linear prediction), transmitted at 8 kbps. Both coders were found to produce speech with a certain "roughness" and "hoarseness" [10]. These effects may be largely due to spectral aliasing caused by the nonlinear distortion performed on the baseband signal. Such aliasing can be eliminated by proper oversampling of the baseband, before the distortion is applied. With this modification, recently suggested by Esteban et al. [11], speech quality at 9.6 kbps has been reported to be good. Finally, a voice-excited LPC coder has also been considered for a low bit-rate transmission at 3.6 kbps by Atal et al. [12], but the results were not encouraging.

An observation on terminology is in order. Both RELP and VELP coders are voice-excited coders, since both conform to the spirit in which Schroeder and David introduced the idea of voice excitation. The terms "residual-excited" in RELP and "voice-excited" in VELP are thus confusing. This situation has prompted us to use the more obvious and clear terminology of baseband residual coder and baseband speech coder for RELP and VELP, respectively.

It is acknowledged that we have made extensive use of the results obtained in another government-sponsored BBN project [3,13] that started about six months before this project. (In that project, both baseband residual and baseband speech coders were investigated, and a speech sampling rate of 8 kHz was considered.) Specifically, most of the results reported in Sections 2.7, 2.9 (excluding Section 2.9.2) and 2.10 were obtained as part of the other project. The spectral folding method of high-frequency regeneration described in Section 2.9.2 was originally developed as part of a different government-sponsored BBN project [14].

2.5 SAMPLING RATE OF INPUT SPEECH

2.5.1 Sampling Rate

One of the requirements on the speech coder is that the bandwidth of the input speech of the coder be greater than or equal to 3.2 kHz. The audio signal interface provided by GTE Sylvania for the MAP-300 array processor provides lowpass filters with 3 dB cutoffs of 3.2 kHz and 3.8 kHz. Therefore, the input sampling rate FS may be chosen to have a value around 6.67 kHz or 8 kHz. This leads to three observations. First, a reasonable value for the LPC order (or number of spectral parameters) is 8 poles for FS=6.67 kHz and 10 poles for FS=8 kHz; the latter choice involves transmission and possible error protection of two additional parameters. Second, since we have assumed the ratio of fullband width W to baseband width B to be an integer L, reasonable values of L are 3 or 4 bands for 6.67 kHz and 4 or 5 bands for 8 kHz. Other values of L give either too small a B, which leads to poor speech quality, or too large a B, which leads to a coarse quantization of baseband samples for the given bit-rate of 9.6 kbps. As reported below in Section 2.9, the HFR schemes based on the spectral folding idea produce more "tonal noises" as L is increased. This result favors the choice of the lower sampling rate. Finally, the computational load is greater with the higher sampling rate. Although the choice of FS=8 kHz may potentially yield slightly higher speech quality, we chose FS=6.67 kHz based on the above considerations.

The exact value of the sampling rate has to be selected from the options provided by the real-time clock in the audio signal interface. The four candidate values around 6.67 kHz that we considered are: 384/60, 384/59, 384/58 and 384/57 kHz, where the value of 384 kHz is the primitive clock rate provided by the master oscillator within the interface, and the integer in the denominator in each case is the programmable divide-ratio. We decided against the divide-ratio of 60, since it would cause some aliasing, as it gives W=3.2 kHz, which is the 3 dB cutoff frequency of the lowpass filter. Our choice of the divide-ratio from the remaining three values was made as follows. For any chosen sampling rate, we must have an integer number of baseband samples per frame and preferably an integer number of data bits per frame to avoid additional buffering. Through detailed computations, we found that using the divide-ratio of 58 yields a variety of possible values for both frame size and bits per baseband sample and hence generates a number of 9.6 kbps coder realizations, which can be considered in the selection of the optimal system. Therefore, we chose the sampling rate as 384/58 (approximately 6.621) kHz. In all the simulations of the BBC coders on our PDP-10 computer, we used a sampling rate of 6.67 kHz (or 150-microsecond sampling period), since it is close to the chosen sampling rate and since all the simulation results can be simply carried over to the real-time system. For example, a frame size of 27 ms contains 180 speech samples at 6.67 kHz; the corresponding frame size for the real-time system is 27.1875 ms, which also has 180 speech samples.

2.5.2 Data Bases

We employed three data bases of 11-bit linear PCM speech in this project: a high-quality data base, an "office-noise" data base, and a CVSD data base. The high-quality data base has 12 sentences of duration of about 2-3 seconds each, with equal numbers of sentences from male and female talkers. This data base was obtained from an 8 kHz data base being used in another government contract at BBN. The signal-to-noise ratio of the speech in this data base is about 60 dB. The conversion of the sampling rate from 8 kHz to 6.67 kHz was achieved by a 5:1 interpolation followed by a 6:1 decimation. The office-noise data base has 10 sentences, which we digitized at 6.67 kHz directly from a sponsor-supplied audio tape recorded in an office-noise environment (with the acoustic background noise at a level of about 60 dB SPL re 20 micronewtons per square meter). The CVSD data base has 12 sentences of 16 kbps CVSD speech, which we digitized at 6.67 kHz from an audio tape provided by the sponsor. For the formal subjective speech-quality test described in Section 2.12, we used a different high-quality data base of 36 sentences, originally sampled at 10 kHz and digitally converted to the 6.67 kHz sampling rate.

2.6 ESTIMATION AND CODING OF SPECTRAL PARAMETERS

The spectral parameters in our case are the LPC parameters $a(k)$, $1 \le k \le p$, that determine the LPC inverse filter $A(z)$ in Eq. (2.1) used for computing the residual at the transmitter (Fig. 2-2(a)) and the synthesis filter $H(z)=1/A(z)$ at the receiver (Fig. 2-2(b)).

2.6.1 Estimation

In a residual-excited (fullband) coder, such as APC, the residual signal compensates for any lack of accuracy in estimating spectral (LPC) parameters. However, in a BBC coder where only a baseband is transmitted, the spectral accuracy at frequencies above the baseband depends exclusively on the accuracy of the LPC parameters in characterizing the spectrum. In this regard, the demands on spectral accuracy are similar to what one would find in a pitch-excited coder. In particular, this means that the number of LPC parameters (predictor order) p must be large enough to model the speech spectrum accurately, and the parameters must be quantized such that the spectrum is minimally distorted. We do not feel that the particular method of linear prediction used is an important issue in the BBC system, as long as the resulting all-pole filter $1/A(z)$ is stable. The autocorrelation method was chosen for its simplicity and because of the good results we have obtained with it in the past. We found that for a sampling rate of 6.67 kHz, a predictor order p of 8 was required for transmitting good-quality speech. A higher value of p, e.g. 10, did not produce any noticeable improvement in speech quality. Therefore, we chose the value of $p=8$ for use in all our simulation work.

While the spectral accuracy mentioned above is of a static nature, there is a dynamic variable which also affects the performance of a BBC system. This variable is the frame size, or the rate of transmission of LPC coefficients in frames/sec. For large frame sizes the spectral parameters attempt to describe a large speech region, which results in a residual with a non-flat short-term spectrum; also the predictor becomes less adaptive to input speech, thus leading to a decline in system performance. A frame size of about 20-30 ms has been found to give good results. Notice that the choice of the frame size involves a tradeoff between the dynamic spectral accuracy on the one hand, and the quantization accuracy of the baseband residual and/or the amount of error protection of the transmitted data on the other. If large frame sizes (or low frame rates) are to be used, parameter interpolation may be employed to provide a more frequent updating of the parameters of both the LPC inverse filter and the synthesis filter. This topic is discussed in Section 2.11.4.

2.6.2 Coding

The LPC parameters a(k) are to be quantized and binary-encoded. Our previous work has shown that optimal quantization of the LPC parameters can be accomplished by uniformly quantizing log area ratios (LARs), which are obtained by first converting predictor coefficients a(i) to reflection coefficients K(i) and then using the following logarithmic transformation [15,16]:

$$g(i)=10\log\,[1+K(i)]/[1-K(i)],\quad 1\leq i\leq p. \qquad (2.2)$$

The ranges of the 8 LARs obtained for the high-quality data base are given in Table 2-1. We employed a total of 33 bits and optimally allocated them among the individual coefficients using a method reported in [15,16]. The optimal bit allocation and step sizes are also given in Table 2-1.

A point regarding the computation of the residual is in order. Since the synthesizer at the receiver will use predictor coefficients obtained from the quantized LARs, it is important to use the same coefficient values in the inverse filter A(z) used in computing the residual; the resulting residual in the absence of quantization and with B=W will produce output speech identical to the input speech.

2.7 BASEBAND EXTRACTION

The baseband signal is obtained by lowpass filtering the linear prediction residual. The cutoff frequency of the lowpass filter is B=W/L. The filtered signal is then decimated to its Nyquist frequency of 2B Hz by retaining every Lth sample and discarding the others.

Elliptic, Butterworth, or finite impulse-response (FIR) filters can be used for this lowpass filtering operation. The

Coefficient Number	Range in dB		No. of Bits	Step Size (dB)
	Minimum	Maximum		
1	-16.854	3.498	5	0.636
2	- 4.199	16.473	5	0.646
3	-13.325	7.475	5	0.650
4	- 2.828	10.100	4	0.808
5	- 6.061	5.348	4	0.713
6	- 2.723	9.725	4	0.778
7	- 5.391	4.193	3	1.198
8	- 2.558	5.627	3	1.023

Table 2-1 Quantization data for log area ratios

first two types of filters produce a frequency-dependent phase delay at the output, while the linear-phase FIR filters produce a constant delay that can be easily compensated. To avoid any possible speech-quality degradations due to phase distortion, we chose to use a linear-phase FIR filter. In all our simulation work, we employed an FIR filter of order 201, which has a very sharp cutoff characteristic with a transition region of only about 110 Hz width and stop-band attenuation of over 50 dB.

For a linear-phase FIR filter of order (2M+1), the phase delay introduced at the filter output is equal to M samples, which is 15 ms with M=100. We compensated this constant delay by moving the filter impulse response left by M samples. The cost for the elimination of the phase delay is that for filtering a given (or current) frame of signal, it is necessary to have M samples from the past frame as well as M samples from the next or future frame. This was accomplished by providing a full frame of delay. Buffering of the past and the future frames of data was necessary for filtering the current frame. For each frame of input speech, the fullband residual was computed. Extraction of the baseband from this residual did not occur until the next frame. At that time the next frame fullband residual would also be available so that filtering could be done. Parameters of a frame (LPC coefficients and quantizer gain) were transmitted only after the baseband of that frame was extracted.

2.8 CODING OF BASEBAND RESIDUAL

The baseband residual can be coded by any of the known waveform coding techniques, such as log PCM, adaptive PCM, ADPCM, APC, ADM, CVSD, sub-band coding, or ATC. It has been reported that use of ADPCM or CVSD for coding the baseband speech does not provide any significant advantage over log PCM [9]. The reason may be that differential waveform coders derive their advantage from a high sample-to-sample correlation, which is usually not the case for the baseband signal. ADM coding has been used in [8], while sub-band coding has been used in [11], both for coding the baseband residual. Since the residual already has a flat spectrum, we do not believe that any differential coding technique would offer any distinct advantage. The same argument applies to sub-band coding, unless either noise spectral shaping or some band "skipping" is exercised.

In our work, we investigated the two techniques: Adaptive PCM (APCM) and APC. Below, we describe the two coding techniques and present the results of our investigation of entropy (or variable wordlength) encoding of the quantized baseband residual.

2.8.1 Adaptive PCM Coder

The APCM coder that we employed is shown in Fig. 2-3. This coder has a gain multipler G followed by an optimum (minimum mean-square error [17]) unit-variance quantizer. G is varied adaptively by setting its value equal to the reciprocal of the rms value of the baseband residual over the current frame. The value of G is quantized before it is used to multiply the baseband residual. Note that in Fig. 2-3, we show a double line emanating from the gain computation, implying that more than one value is computed per frame. This scheme, called segmented quantization, causes the quantizer to adapt more rapidly to local variations in the baseband signal, such as sudden increases in energy due to "pitch pulses" [18]. Therefore, in addition to computing and coding the gain over the whole frame, several incremental gain parameters are computed within the frame. The deviations of these parameters from the average gain G ("delta gain" parameters) are transmitted.

We found that the use of segmented quantization produced a perceivable improvement in speech quality only if the residual is quantized very coarsely (e.g., 1 bit/sample). Even at 2 bits/sample, the improvement in speech quality was only barely audible. Since all candidate 9.6 kbps systems considered in this project used at least 3 bits/sample, we decided not to employ

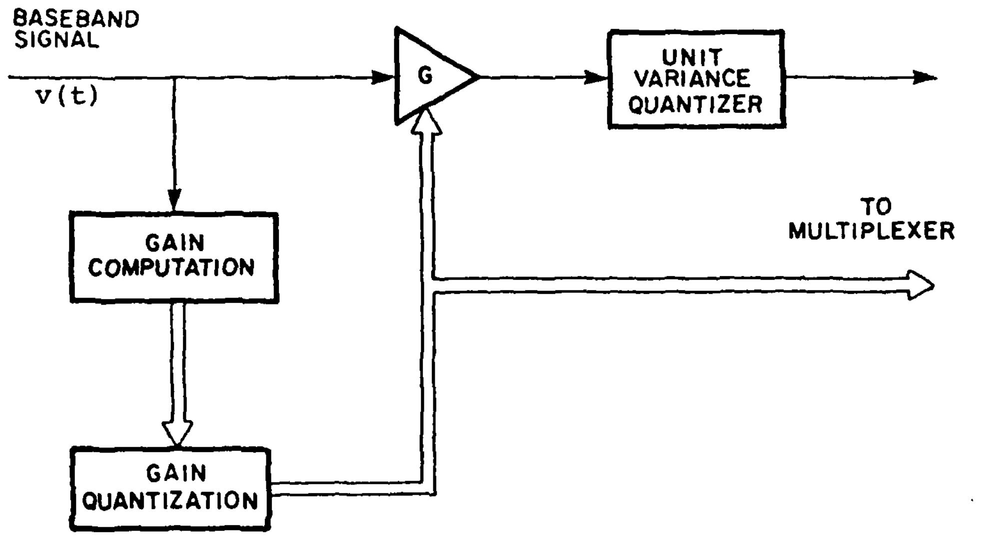


Fig. 2-3 Adaptive PCM coding of the baseband signal

segmented quantization in all our subsequent investigations. The quantizer gain G was transmitted in terms of the energy or mean-square value of the baseband residual. For the high-quality data base, the baseband energy had a range of -5 to 45 dB. We quantized the energy in dB using 6 bits. In designing the optimum nonuniform quantizer, we considered three distributions for the quantizer input: Gaussian, Laplacian, and Gamma. We chose a Laplacian quantizer because it produced a slightly higher speech quality than either of the other two.

In our optimization study, we allowed the use of a noninteger number of bits/sample for baseband residual quantization, e.g., 11 levels/sample. In this case, we combined the quantized levels in a block of typically 2-5 samples and coded them jointly using an integer number of bits. For the example of 11 levels/sample, we would code blocks of two samples using 7 bits per block, since a block corresponds to a total of 11 x 11 = 121 levels. The availability of the noninteger-bit option was important, since it significantly increased the number of 9.6 kbps systems considered in the optimization study described in Sections 2.11 and 2.12.

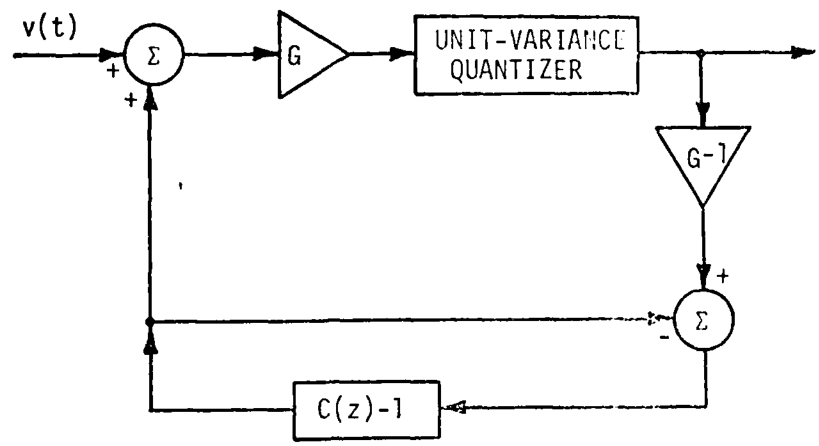
2.8.2 APC Coder

The APC coder shown in Fig. 2-4(a) has a feedback structure with an APCM quantizer placed in the forward path and an adaptive linear predictive filter placed in the feedback path around the quantizer. Since the signal to be coded is the baseband residual, the adaptive predictor in the APC feedback loop is a pitch predictor, which removes the redundancy due to quasi-periodicity during voiced segments. Notice that the quantizer gain G in Fig. 2-4(a) is computed from the "second residual" that is obtained by filtering the baseband residual with the pitch-inverse filter C(z). We considered the following two choices for C(z):

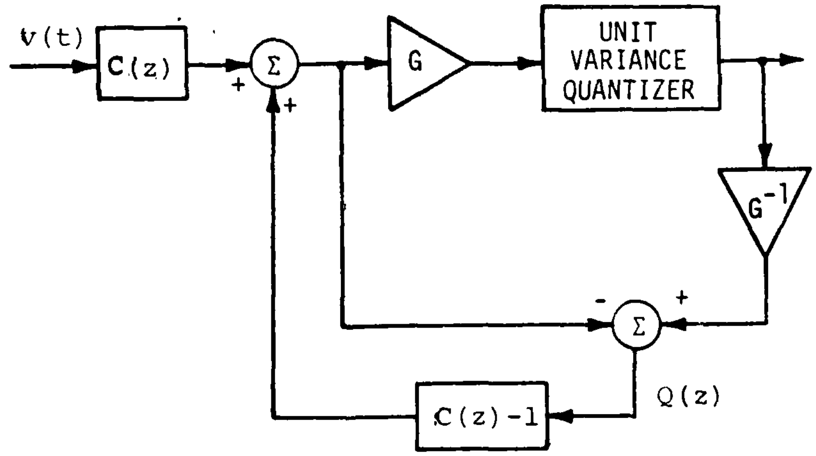
$$C(z) = 1 - c\, z^{-M} \quad \text{(1-tap)}, \tag{2.3}$$

$$C(z) = 1 - c1\, z^{-(M-1)} - c2\, z^{-M} - c3\, z^{-(M+1)} \quad \text{(3-tap)}, \tag{2.4}$$

where c, c1, c2 and c3 are pitch predictor coefficients and M is the pitch period in number of baseband samples. We computed M as the location of the peak of the autocorrelation function of a 40-ms interval of the baseband residual obtained from the current frame and the trailing part of the preceding frame. In our simulation on the PDP-10, we computed the autocorrelation function using two FFT operations by first calculating the spectrum of the baseband residual and then inverse transforming. For the range of pitch frequencies 50-450 Hz that we considered, M has a range 5-45 for



(a)



(b)

Fig. 2-4 Adaptive predictive coding of the baseband residual v(t). C(z) is the pitch-inverse filter.

(a) Initial implementation of the APC coder

(b) Implementation of the APC coder in the noise-feedback configuration. Q(z) is the quantization noise.

the 3-band coder, and a range 4-33 for the 4-band coder. Thus, M can be transmitted without quantization error in the two cases, using 6 bits and 5 bits, respectively.

We used the autocorrelation method of linear prediction for computing the pitch predictor coefficients. The necessary autocorrelation coefficients for this computation are already available from the above-described pitch estimation procedure. The autocorrelation normal equations for the 1-tap and 3-tap cases are given below:

<u>1-tap</u>:

$$c=R(M)/R(0). \tag{2.5}$$

<u>3-tap</u>:

$$\begin{bmatrix} R(0) & R(1) & R(2) \\ R(1) & R(0) & R(1) \\ R(2) & R(1) & R(0) \end{bmatrix} \begin{bmatrix} c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} R(M-1) \\ R(M) \\ R(M+1) \end{bmatrix}. \tag{2.6}$$

The 1-tap synthesis filter $1/C(z)$ is guaranteed to be stable, since c in Eq. (2.5) is less than 1 in magnitude. The range for the tap coefficient was found to be $0 \le c \le 1$. We quantized c linearly using 4 bits. On the other hand, the 3-tap case need not always be stable. The ranges for the 3 tap coefficients were found to be: $-0.75 \le c1 \le 0.5$, $0 \le c2 \le 1$ and $-0.33 \le c3 \le 0.49$. We quantized linearly c1 and c3 using 3 bits each, and c2 using 4 bits. The energy of the second residual was found to have a range of -18 to 36 dB. We quantized the energy in dB using 6 bits and transmitted it for computing the gain G at the receiver.

Since computing the gain G requires inverse filtering of the residual with C(z) as noted above, the second residual may be used as input to an alternate implementation of the APC coder as shown in Fig. 2-4(b). This alternate configuration is called the noise-feedback configuration [27], since it has an explicit feedback of the quantization noise Q(z). This configuration allows the monitoring of the relative contributions of the second residual and the quantization noise to the quantizer input [28,29]. Computationally, the two implementations in Fig. 2-4 require about the same number of multiplies and adds. In our simulation system, we employed the noise-feedback configuration.

Informal listening tests showed that the 1-tap and 3-tap APC coders produced a significant improvement in speech quality over the APCM coder. Results of an experiment comparing these three cases under a fixed bit-rate (9.6 kbps) constraint are given in Section 2.11.3.

2.8.3 Entropy Coding

We investigated variable wordlength entropy coding of the quantized values of the baseband residual. The main idea of entropy coding is to use a short code for high-probability levels and a longer code for lower probability levels, so that the average codelength is minimized. When using entropy coding, we employed a uniform quantizer, since it minimizes the mean-square error [19].

Entropy coding is attractive, since it minimizes the bit-rate at a given signal-to-quantization-noise (S/Q) ratio, or, equivalently, for a given bit-rate it allows us to maximize the S/Q ratio. The drawbacks of entropy coding are: (1) variable bit-rate, (2) possible need to update the coding tables periodically, which may be quite costly when the number of quantization levels is large, and (3) in the case of a noisy channel, a single bit-error causes a misinterpretation of all subsequent codes at the receiver. In our work we investigated: (1) the possibility of having a fixed set of variable length codes that is applicable to all speakers, all sentences, and all input signal-to-noise ratios, and (2) the self-synchronizing property, which is necessary for satisfactory performance over a noisy channel. Our findings were that it is possible to use a fixed set of codes for all inputs with a minimal change in bit-rate. However, the codes were quite different from the set of self-sync codes we considered. In particular, for a

31-level quantizer in a 3-band system, the codelengths for the innermost 9 levels were, from most negative to most positive: 5,4,3,3,2,3,4,4,5 bits, and the average transmission rate for the baseband residual was about 7733 bps. These codes do not exhibit the self-sync property. In order to insure having the self-sync property, we attempted to replace the above codes with a set having the codelengths: 8,6,4,2,1,3,5,7,9 bits. This second set of codes is{0,10,110, etc.}, which clearly has the self-sync property [20]. Because of the great mismatch between this last set of codes and the optimal ones, we observed an increase of 1.2 bits in the average codelength per sample, or a total of 2667 bps increase in bit-rate, which produced an average transmission rate of 10.4 kbps for the baseband residual alone. We also performed similar experiments for the 2- and 4-band cases. In all cases, the increase in bit-rate was not acceptable. Therefore, because of the lack of the self-sync property, we decided against the use of variable length codes in a 9.6 kbps BBC system.

In all our subsequent work, we therefore considered exclusively a nonuniform Laplacian quantizer and fixed wordlength binary encoding.

2.9 HIGH-FREQUENCY REGENERATION

The HFR problem is stated as follows: Given the baseband residual, generate high frequencies such that the fullband excitation signal has a flat spectrum and proper harmonic structure for voiced sounds. It is well known that if the baseband has either the voice fundamental or at least two adjacent harmonics, a waveform containing all the original harmonics of voiced input speech can be generated by feeding the baseband signal to an instantaneous, zero-memory, nonlinear device. The spectral shape of the regenerated harmonic structure may be quite arbitrary and must be flattened to produce a suitable excitation function. Schroeder and his associates investigated several analog nonlinear distortion devices for these purposes. In one study, they developed an ingenious "zig-zag" nonlinear circuit which performs nonlinear distortion and also "spreads" (or flattens) the spectral envelope by substantially increasing the number of zero-crossings of the baseband signal [4,5]. In another investigation, HFR was accomplished by rectifying the baseband signal, applying it to the bandpass filters of the channel vocoder synthesizer, and peak-clipping the filter outputs [6]. Almost all recent implementations have been digital [7-12]. Below, we describe the well-known waveform rectification method and several other methods that have been developed recently at BBN [3,13,14].

2.9.1 Rectification

Figure 2-5 shows a digital implementation of HFR by rectification and spectral flattening. The received baseband signal is interpolated in two stages to a sampling rate of 4W Hz. The higher sampling rate is necessary to minimize the spectral aliasing from the subsequent nonlinear distortion, which can cause roughness and "hollowness" in the output speech [11]. The two-stage interpolation, rather than the single-stage interpolation shown by the dashed-line box in Fig. 2-5, has two advantages: (1) The combined order of the two lowpass filters required in the two-stage process can be made substantially less than the order of the lowpass filter in the single-stage case, for the same overall filter characteristics. For instance, a high-order FIR filter for stage 1 and a low-order Butterworth filter for stage 2 would be quite adequate. (2) The two-stage process provides without additional computation the baseband signal at the full sampling rate of 2W Hz, which is required for baseband reintroduction (see Fig. 2-2(b). (Notice that in the single-stage case, the required baseband signal is obtained by down-sampling the interpolated signal by a factor of 2.) The interpolated baseband signal is then nonlinearly distorted by passing it through a waveform rectifier. It has been found that the degree of rectification (ranging from half-wave to full-wave rectification) does not affect the output

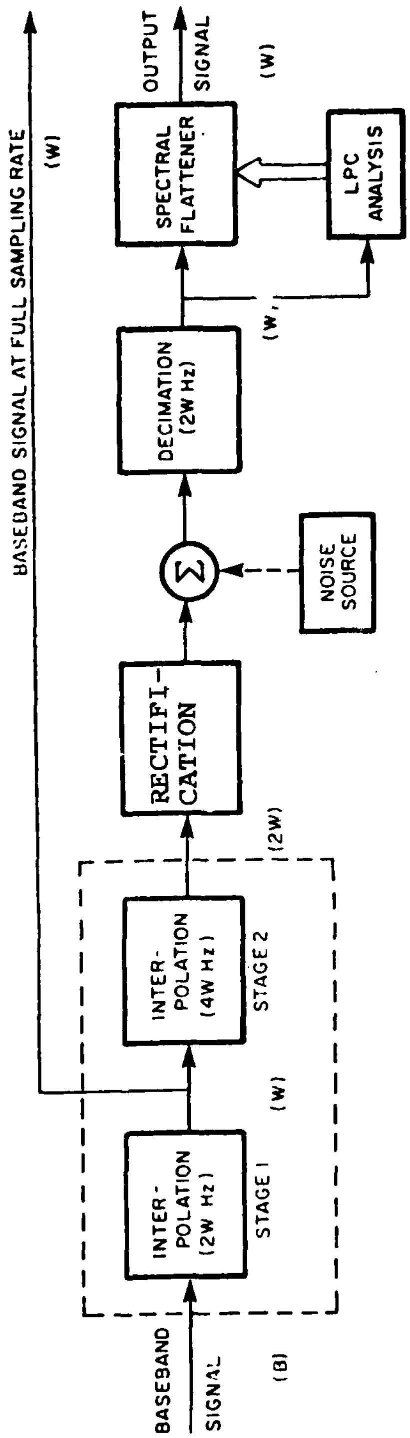


Fig. 2-5 High-frequency regeneration using waveform rectification

speech quality [14]. The distorted signal is decimated to 2W Hz and then spectrally flattened to form the output of the HFR process. While spectral flattening can be done in a number of ways, Fig. 2-5 shows the adaptive LPC inverse filtering method. The spectral flattener is placed after the decimation (instead of before), so that a lower order LPC filter would suffice. Most schemes to date have added some noise to the distorted signal as shown by the dashed lines in Fig. 2-5, to compensate for the lack of high frequencies in fricatives. With the use of the adaptive LPC spectral flattening, we have found that the noise source is unnecessary. Finally, the spectrally flattened output is highpass filtered at a cutoff frequency of B Hz (not shown in Fig. 2-5) to obtain the highband, which is then combined with the baseband (at the full sampling rate of 2W Hz). The combination procedure involves appropriate scaling and adding of the two flat-spectrum (highband and baseband) signals such that the resulting signal has a flat overall spectrum rather than a stair-case spectrum [3].

Our experimental investigations of the rectification method have produced the following two results: (1) Removing the short-term dc bias from the signal before rectification and from the decimated signal (after rectification) used for computing LPC flattener parameters improves the output speech quality by reducing the amount of roughness; and (2) the improved rectification method still produces audible roughness in the form of background noises.

The filtering operations required for the rectification method (see Fig. 2-5) introduce additional frames of delay (see Section 2.7) and make the HFR process computationally rather expensive.

2.9.2 Spectral Folding

Recently, Makhoul and Berouti introduced a simple HFR method called spectral folding [14]. In this method, which is illustrated in Fig. 2-6(a), the process of high-frequency regeneration reduces to inserting L-1 zeros after every baseband sample, where L=W/B. This upsampling operation aliases the baseband into the highband, as shown schematically in Fig. 2-6(b) for the case L=4. Notice that spectral folding, unlike the rectification method, preserves the baseband. Therefore, the additional steps of highpass filtering and baseband reintroduction are not required. Also, the spectral folding method does not perform spectral flattening since the baseband residual is assumed to have a generally flat spectrum.

Clearly, the spectral folding method is quite simple. However, it produces audible "tonal noises," which increase with the number of bands L and with the pitch of the speaker. A modification to the above method shown in Fig. 2-6(c) largely eliminates a specific tonal noise in the output speech at multiples

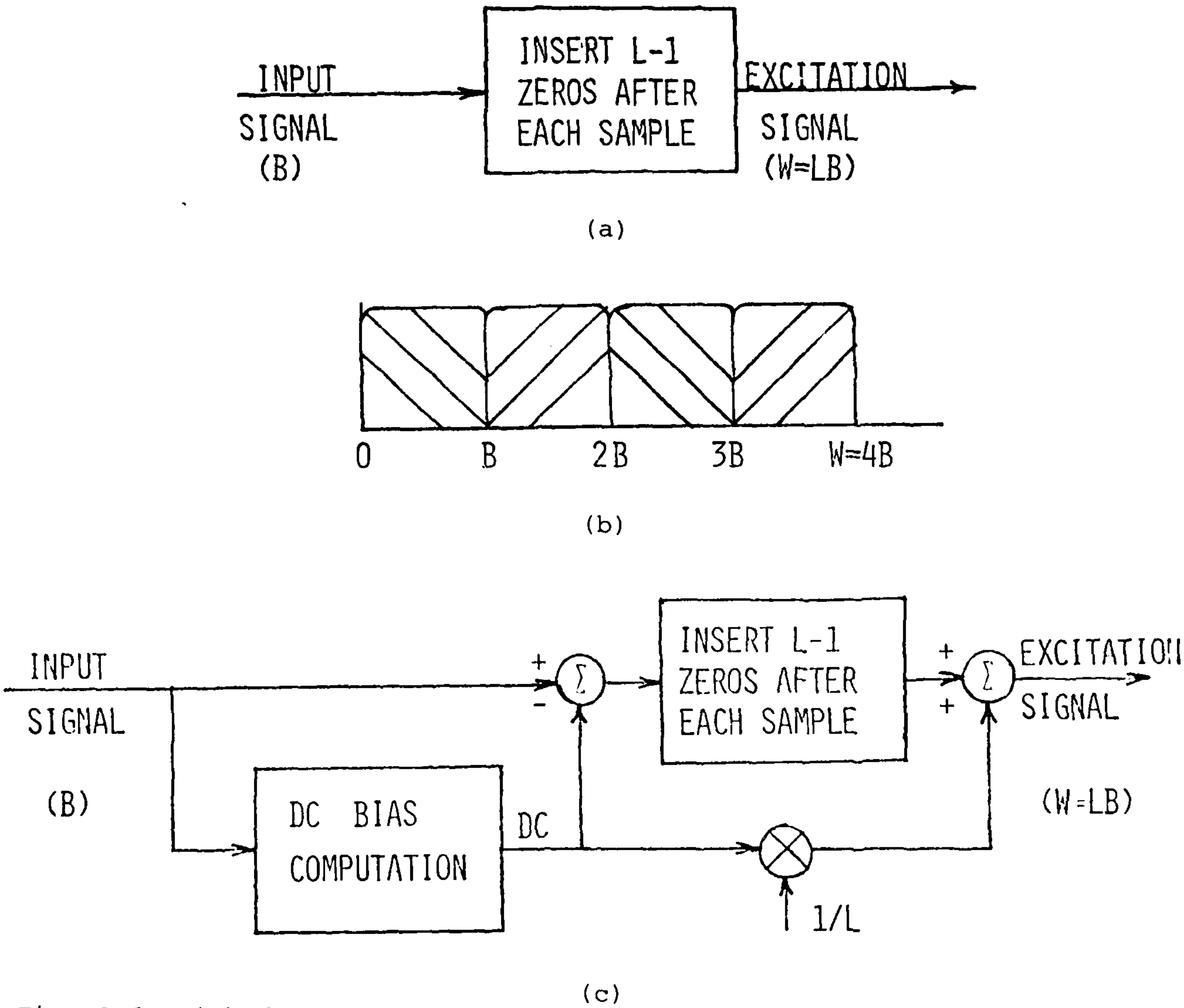


Fig. 2-6 High-frequency regeneration by spectral folding

(a) Basic method

(b) Frequency-domain interpretation for the 4-band example (L=4)

(c) A modification to the basic method, which removes dc bias prior to upsampling and restores it afterwards. The scaling by 1/L keeps the output dc value same as the input dc value.

of 2B Hz. The modification requires the subtraction of the short-term dc in the baseband residual before inserting zeros, and restoring the original dc after the insertion, as shown in Fig. 2-6(c).

We also investigated a related method called spectral translation [14]. In this method, the baseband is copied into, rather than aliased into, higher bands. The method requires the use of heterodyning and bandpass filtering. We found that the speech produced by the spectral translation method did not differ appreciably from that produced by the spectral folding method. Therefore, the extra computation required for spectral translation is not worthwhile.

The simplicity of the spectral folding method motivated us to develop methods of reducing or masking the tonal noises at the cost of greater complexity. This work was performed as part of another government contract [13]. The three HFR methods developed in this work are described below.

2.9.3 Spectral Folding with Preflattening

For proper generation of the high-frequency portion of the synthesizer excitation signal, the spectral folding method requires that the baseband spectrum be flat. However, this is not always so. Since the high-frequency part generated by spectral folding is made up of folded versions of the baseband, a nonflat baseband spectrum results in a nonflat high-frequency spectrum. This might, in part, be responsible for the tonal noises produced by the spectral folding method.

The method of spectral folding with preflattening, depicted in Fig. 2-7, overcomes the above problem by spectrally flattening the baseband prior to spectral folding. The upper branch in Fig. 2-7 extracts the baseband at the full sampling rate of 2W Hz, and the lower branch extracts the highband; the two are added to obtain the fullband output signal. In the lower branch, the baseband signal is first spectrally flattened by the LPC inverse-filtering method and then spectrally folded by upsampling. The scaling that follows is done to compensate for the energy loss due to inverse filtering. Without the baseband reintroduction shown in Fig. 2-7, the baseband portion of the excitation signal is also modified by the preflattener, which was found to cause substantial roughness in the output speech.

Informal listening tests have shown that this HFR method reduces tonal noises at the expense of "pinging sounds". The order of the LPC flattener represents a tradeoff between the amounts of tonal noises and pinging sounds. That is, as the filter order is increased, tonal noises are reduced and pinging sounds are enhanced. Our experiments have shown that a fourth-order flattener offers the best compromise.

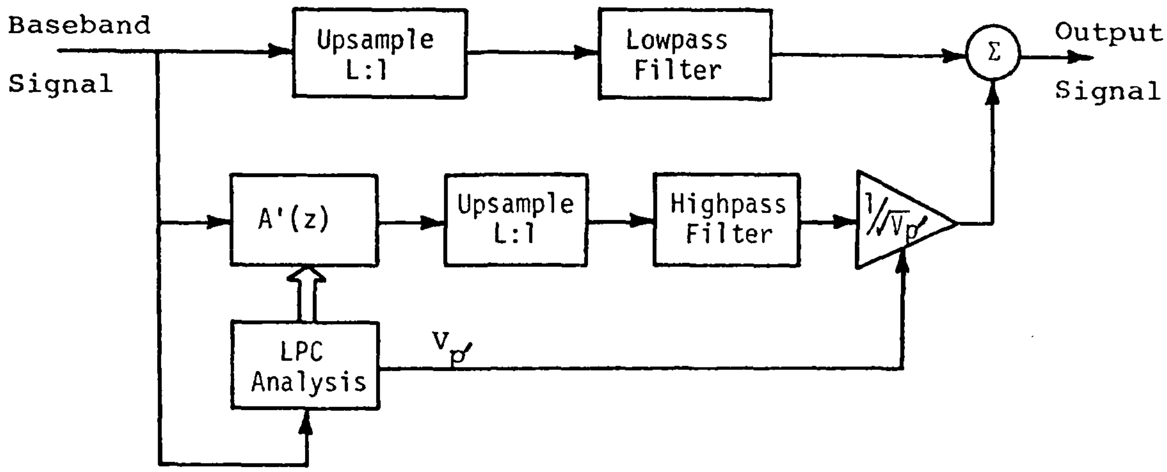
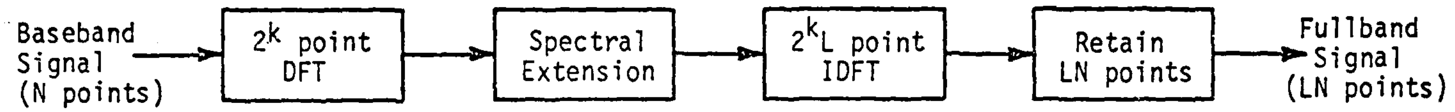


Fig. 2-7 Block diagram description of the high-frequency regeneration method of spectral folding with preflattening. A'(z) is the LPC inverse filter used for spectrally flattening the baseband signal, prior to spectral folding. $V_{p'}$ is the normalized error-signal energy computed as part of the LPC analysis.

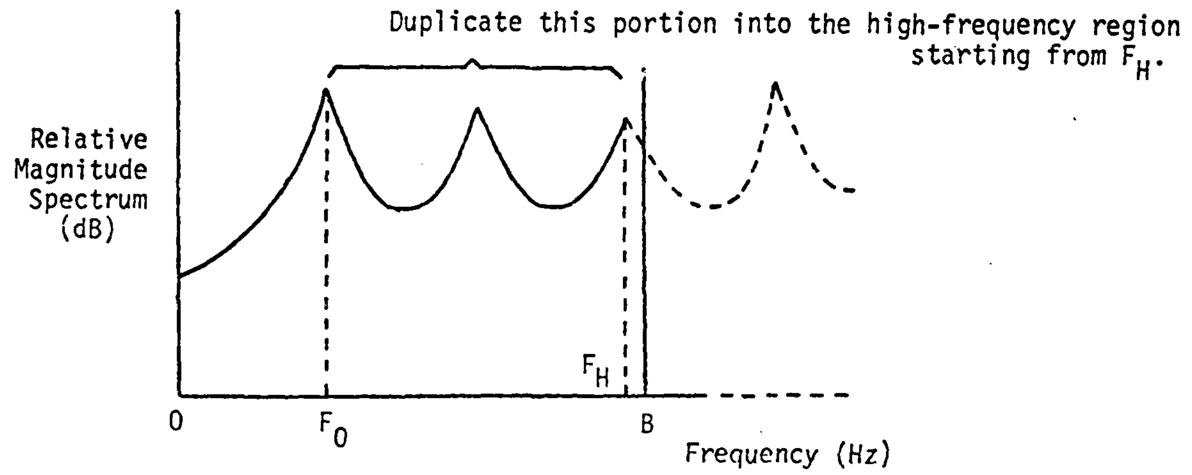
2.9.4 Frequency-Domain Method

The spectral folding method causes, in general, a break in the harmonic structure at multiples of the folding frequency B. This is because the baseband width B is not an integer multiple of the fundamental frequency. This problem of harmonic interruption can be resolved by carrying out spectral folding in the frequency domain, as shown in Fig. 2-8. The discrete Fourier transform (DFT) of the baseband residual is computed first, and the DFT coefficients between the first and the highest pitch harmonics are duplicated over and over to obtain the fullband, thus keeping the harmonic structure intact. In our investigation, we computed the first harmonic as the lowest maximum of the baseband magnitude spectrum. We then computed the highest harmonic in the baseband as follows. The baseband magnitude spectrum was reversed in frequency and cross-correlated with the unreversed original spectrum. The frequency corresponding to the first (lowest) local maximum of the cross-correlation function was used as the estimate of the highest pitch harmonic in the baseband. When the cross-correlation function mentioned above did not have a local maximum within a predetermined value of the lag, we declared the frame unvoiced and duplicated the entire baseband into the high-frequency region.

In our experimental investigation of several different versions of the frequency-domain HFR method, we found that the



(a)



(b)

Fig. 2-8 Frequency-domain method of high-frequency regeneration.

(a) Block diagram description of the method, where $2^k \geq N$. (IDFT denotes Inverse DFT).

(b) Interpretation in the magnitude spectrum.

DFT-based method reduced tonal noises, but at the expense of certain background "tinkling noises". Recently, the frequency-domain HFR approach has been implemented using the discrete cosine transform instead of the DFT [21], but the tinkling noises were still perceived in the output speech. We feel that these background noises may be due to improper phase extension.

2.9.5 Perturbed Spectral Folding

Simple spectral folding exhibits spectral regularity in the way it reflects the baseband into the higher bands. This regularity might be responsible for some of the tonal noises. The method we call perturbed spectral folding breaks up this spectral regularity. This method is illustrated in Fig. 2-9. In the lower branch of Fig. 2-9, the nonzero samples of the spectrally folded (or upsampled) baseband residual are randomly perturbed; perturbation is performed by simply interchanging the sample with an adjacent (zero) sample. The perturbation procedure is as follows. No perturbation is performed if the magnitude of the nonzero samples is larger than a preselected threshold X; this avoids perturbing samples corresponding to pitch pulses. Nonzero samples with magnitudes less than this threshold are perturbed with a variable probability, with the smaller samples having a higher probability of being perturbed. The perturbed signal is then highpass filtered and added to the received baseband residual at the full sampling rate of 2W Hz (top branch in Fig. 2-9).

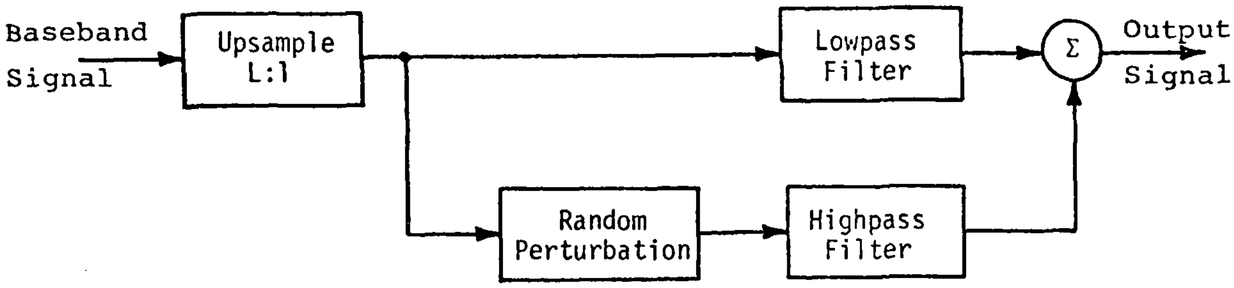


Fig. 2-9 Block diagram description of the perturbed spectral folding method of high-frequency regeneration. The top branch provides the received baseband at the full sampling rate, while the bottom branch produces the high-frequency region by randomly perturbing the non-zero samples of the spectrally-folded input.

A flow-chart of the perturbation scheme is shown in Fig. 2-10. The values of the thresholds X and C, and the parameter D were chosen for best speech quality. The optimized values are: X=35, C=0.5, and D=0.7. The X-value of 35 was also found to be approximately the long-term rms value of the baseband residual.

An interpretation of the perturbed spectral folding method is shown in Fig. 2-11, for the case L=4. The result of regular spectral folding is shown in Fig. 2-11(a), while the result of the perturbed spectral folding is shown in Fig. 2-11(b). The dashed line shows that the third nonzero sample was perturbed to the left by one sample. The result of perturbed folding may be viewed as the sum of the regular spectral folding output and the additive noise shown at the bottom of the figure. Perceptually, we have found that this additive noise has the effect of masking the tonal noises. The additive noise also causes slight roughness.

Of the different HFR methods we considered, our informal listening tests have indicated that perturbed spectral folding produces the best overall speech quality. This result was also borne out in a formal subjective speech quality test, which is described in Section 2.12.

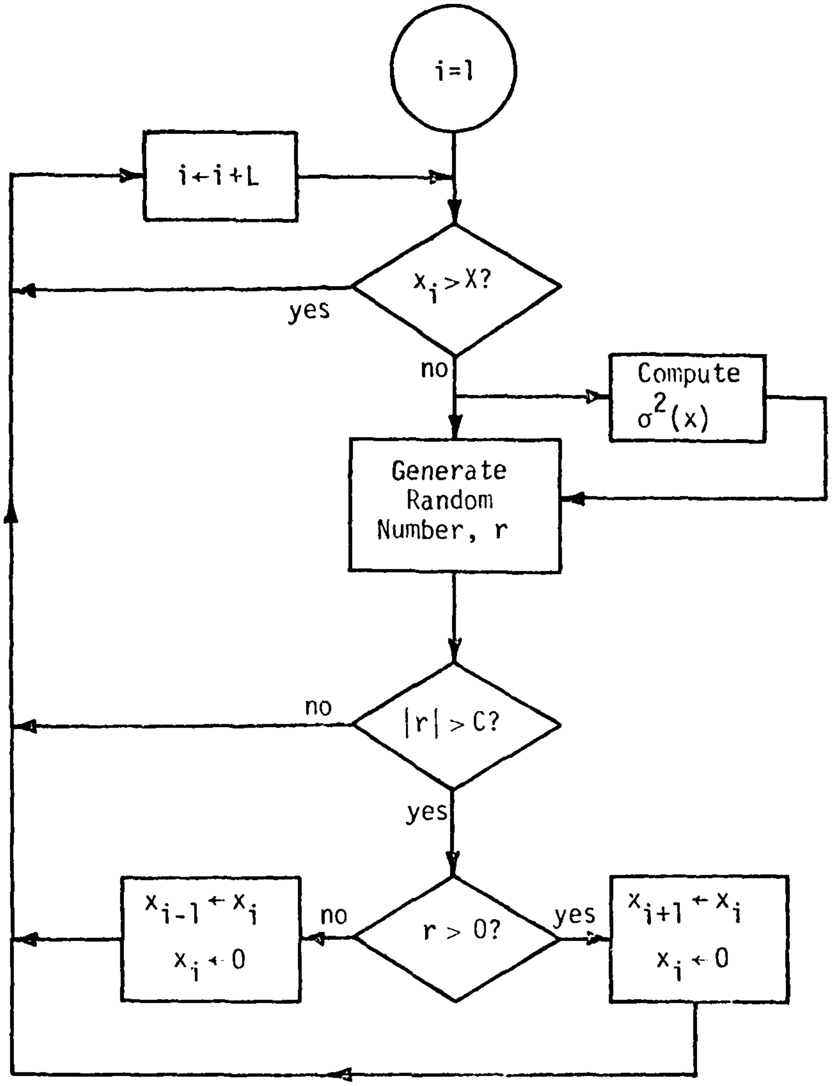


Fig. 2-10 Flow-chart of the scheme to randomly perturb the nonzero samples of the spectrally-folded signal. In our experiments, we generated the random numbers using a zero-mean Gaussian distribution with a variance σ^2, where $\sigma^2(x)$ is equal to 0 for $|x| > X$, and $D(1-|x|/X)$ for $|x|<X$. The quantities X, C, and D are the parameters of the HFR scheme, which we optimized experimentally.

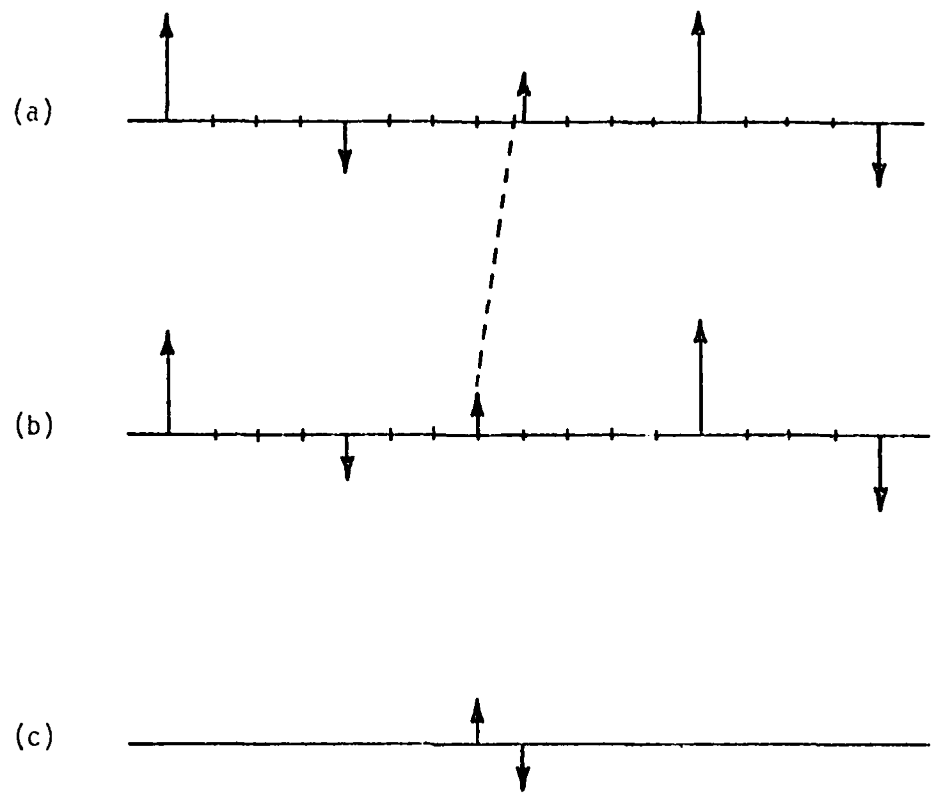


Fig. 2-11 Interpretation of the perturbed spectral folding method

(a) Spectral folding method

(b) Perturbed spectral folding method. Perturbation of a sample is shown by a dashed line, running between (a) and (b).

(c) Additive noise due to perturbation. Notice that adding the time signals in (a) and (c) results in the signal (b).

2.10 SHORT-TERM DC BIAS

We have found through our experiments that the presence of a nonzero short-term (or frame-by-frame) dc bias at each of several points along the signal path in the BBC coder in Fig. 2-2 could cause speech quality degradations at the coder output. We summarize below our main observations.

1. Removing the short-term dc bias from the input speech before LPC analysis (lower branch in Fig. 2-2(a)) was found to yield reduced background noise in the output speech.

2. If dc is removed before LPC analysis <u>and</u> if the quantizer gain G (see Section 2.8) is computed as the reciprocal of the standard deviation (rather than the rms value) of the baseband residual, then dc must be removed from the baseband residual before quantizing it; otherwise, continuous overloading of the quantizer (or clipping) can occur in some low-energy regions of speech. If the rms value is used for computing G, then dc removal from the baseband residual is not required.

3. Removing dc from the baseband residual prior to HFR was found to improve the output speech quality, as mentioned in Section 2.9. For spectral-folding-based HFR methods, highpass filtering the baseband residual prior to HFR, with the cutoff frequency of the filter being below 100 Hz, prevents not only the dc value but also

low-frequency components (due to room acoustics or audio equipment) from being reflected into the higher bands. The speech data base used in the formal subjective speech-quality test (Section 2.12) has such low-frequency components. Without the highpass filtering mentioned above, the speech processed through the coders contained objectionable roughness and noises. These distortions were eliminated when we used the following second-order Butterworth highpass filter with its cutoff at about 73.3 Hz:

$$HP(z) = \frac{0.864\ (1-z^{-1})^2}{1 - 1.708\ z^{-1} + 0.746\ z^{-2}} . \tag{2.7}$$

With the use of the highpass filter, frame-by-frame dc removal is no longer necessary in the receiver. Also, since restoring the dc value after spectral folding (see Fig. 2-6(c)) is not required in this approach, the lowpass filtering to obtain the baseband at the full sampling rate (top branch in Figs. 2-7 and 2-9) can be done efficiently by exploiting the presence of zero samples at the filter input.

2.11 PARAMETER OPTIMIZATION STUDY (ERROR-FREE CHANNEL)

As we explained in the preceding sections, there are several parameters that affect the performance of the BBC system. Important among these parameters are: input sampling rate FS, frame size or equivalently transmission frame rate of coder parameters, number of bands L, number of quantization levels per baseband sample, number of LPC parameters (or LARs) p, number of taps of the pitch predictor, and number of bits per frame allocated for error protection of important parameters. We reported earlier that we chose FS=6.67 kHz (Section 2.5) and that we obtained good results using: p=8, 33 bits for quantizing the 8 LARs (Section 2.6 and Table 2-1), and 6 bits for quantizing the gain G of the baseband quantizer (Section 2.8). Below, we report the results of our optimization study involving the remaining parameters. The objective of this study was to maximize the output speech quality for the high-quality input speech and under a fixed bit-rate of 9.6 kbps for the case of error-free channels. (The topic of transmission over noisy channels is treated in Section 2.13, and the issue of acoustic background noise at the coder input is considered in Section 2.14.) Due to the fixed bit-rate constraint, the focus of this study was to optimize the tradeoff between the various parameters. As a remark that may be helpful for the reader in sorting out the results given below, we point out that the APC coding of the baseband and the new HFR schemes reported in Sections 2.9.3-2.9.5 were available towards the later part of our optimization study. Also, informal listening tests were used to make all quality judgments reported in this section.

2.11.1 Number of Bands (Baseband Width)

The data rate for transmitting the baseband is the product of the number of samples/second and the number of bits/sample. A smaller baseband width allows for more accurate coding of the baseband (for a given bit rate), which increases the signal-to-quantization-noise noise (S/Q) ratio at low frequencies, but simultaneously expands the high-frequency region that must be regenerated, which would contribute further distortion to the signal, and vice versa. Towards finding reasonable values of baseband width or number of bands L for use in subsequent optimization experiments, we considered two sets of 9.6 kbps BBC coders with L=2,3,4 and 5, one set using the rectification method and the other using the spectral folding method. We found that the rectification method produced speech quality which improved gradually when L was increased from 2 to 4 and degraded (rough and "hollow speech") substantially when L was set to 5. A similar speech-quality behavior was observed for the spectral folding method with the exceptions that the peak quality was achieved for L=3 and that the reduction of speech quality (excessive tonal noises) from L=4 to 5 was noticeably more than for the rectification method. Also, considering the speech-quality variability over male and female speakers, the rectification method produced little variability, while the spectral folding method produced better quality (less tonal noises) for males than for females. Based on these results, we chose to investigate only 3-band and 4-band coders in all our subsequent work.

2.11.2 Tradeoff Study

During our parameter optimization study, we ran extensive tests involving tradeoffs between the various coder parameters. Rather than describe all those tests, we have chosen to describe below a test involving six selected BBC systems, all using APCM for coding the baseband. These six systems are defined in Table 2-2. There are three 3-band coders and three 4-band coders. The various parameters are: frame size or duration (FD)=24.3 ms or 28.8 ms for L=3, and FD=25.2 ms or 28.8 ms for L=4; number of levels per baseband sample (NL)=8,10 and 11 for L=3, and NL=16,20, and 25 for L=4. For cases involving noninteger bits per baseband sample, we employed the block-coding scheme described in Section 2.8.1; the numbers in parentheses in Table 2-2 correspond to the average bits/sample. (Notice that a frame contains an integer number of such blocks of baseband samples.) The last row in Table 2-2 gives the left-over or extra bits/frame, which can be used for error protection and synchronization. As another variable, we considered HFR by spectral folding and rectification. The results of our comparative speech-quality judgments of the 12 BBC systems are given below.

ITEM	3-BAND			4-BAND		
	1	2	3	4	5	6
FRAME SIZE (MS)	24.3	24.3	28.8	25.2	25.2	28.8
FRAME RATE (APPROX.)	41.2	41.2	34.7	39.7	39.7	34.7
AVAILABLE BITS/FRAME	233	233	276	241	241	276
SPEECH SAMPLES PER FRAME	162	162	192	168	168	192
BASEBAND SAMPLES/FRAME	54	54	64	42	42	48
BASEBAND LEVELS (BITS) PER FRAME	8 (3)	10 (10/3)	11 (7/2)	16 (4)	20 (13/3)	25 (14/3)
TOTAL BASEBAND BITS/FRAME	162	180	224	168	182	224
LARS & GAIN PER FRAME	39	39	39	39	39	39
TOTAL COMMITTED BITS/FRAME	201	219	263	207	221	263
EXTRA BITS PER FRAME	32	14	13	34	20	13

Table 2-2 Description of the six 9.6 kbps BBC coders tested for determining the speech-quality tradeoff between the various coder parameters

For either HFR method, the low-frame-rate systems (3 and 6) produced at the output occasional faint "chirps" and "pops", but less background noise, relative to the corresponding (same number of bands) high-frame-rate systems (1,2,4 and 5). For the spectral folding method, the best 3-band coder was System 3 and the best 4-band coder was System 5. System 3 was noticeably better than System 5, because the latter system produced much more tonal noises. For the rectification method, the best 3-band and 4-band coders were, respectively, System 2 and System 6, with the latter system being slightly better than the former. For both HFR methods, the speech-quality differences between Systems 2 and 3 or between Systems 5 and 6 were relatively small. The best spectral folding (3-band) system and the best rectification (4-band) system produced different types of distortion: low-level tonal noises in the 3-band system and perceptible roughness in the 4-band system. Subjects differed in their preference among the two systems. The results reported above for the spectral folding scheme were found generally to carry over to the spectral-folding-type schemes described in Section 2.9.

2.11.3 Baseband Coding: APCM vs. APC

From our experimental investigations, we found that the APC coder produced a significant improvement in speech quality relative to the APCM coder, for all of the BBC systems we tested. In one experiment, we compared 3 BBC systems, which were identical except for the method of baseband coding. These systems employed APCM, 1-tap APC, and 3-tap APC. The number of quantizer levels/sample for each system was chosen such that its data rate was about 9.6 kbps. The average segmental (or short-term) S/Q ratios for the baseband residual coding in the three cases were found to be 16.6 dB (APCM), 20.7 dB (1-tap APC) and 20.3 (3-tap APC). Thus, use of APC produced substantial increases in S/Q ratio. Informal listening tests showed that the two APC coders produced a significant improvement in speech quality over the APCM coder and that the 1-tap APC system yielded slightly better speech quality than the 3-tap system.

2.11.4 Parameter Interpolation

We investigated the use of parameter interpolation in low-frame-rate systems. It was anticipated that a low transmission frame rate might be required to provide more of the available bit-rate for error protection and for improved baseband residual quantization. However, low-frame-rate systems have a tendency to cause roughness or discontinuities in the output speech. This effect is due to the extended averaging of spectral parameters and quantizer gain due to the large frame size. Interpolation provides a method for more frequent updating of these parameters and potentially reducing these adverse affects [16,30].

We implemented and tested linear interpolation of LARs and the quantizer gain in dB. These parameters were updated twice each frame (i.e., one interpolation was performed to obtain parameters for the midframe update). Application of interpolation as described above requires several additional considerations. For example, observe that up to one frame of delay and hence buffering is required at the receiver since the <u>next</u> frame parameters must be available to perform the interpolation. Also, the residual or the excitation signal to the synthesis filter must correspond to or match the filter coefficients. This implies that the interpolated LPC coefficients must be used at the transmitter to extract the residual. Thus, a frame of delay (and buffering) is required at the transmitter also.

Initial tests on uncoded parameters extracted every 28.8 ms showed that interpolation (once per frame, at the center) did reduce roughness, but the resulting speech was "muffled" and somewhat lacking in clarity. A shorter analysis interval (a little more than half the transmission frame of 28.8 ms) was found to be more acceptable; the location of this analysis interval with respect to the start of the frame was experimentally optimized. We also found that the transmission of an extra quantizer gain parameter per frame (corresponding to the second half of the 28.8 ms frame) reduced the "muffled" effect. Since interpolation adds more complexity and since it provided only a marginal speech quality improvement, we decided against its use in the real-time implementation of the BBC system, unless demand for increased error protection called for a relatively low transmission frame rate. Fortunately, as reported below in Section 2.13, such a demand did not arise.

2.12 SUBJECTIVE SPEECH-QUALITY EVALUATION

2.12.1 Six BBC Systems

As reported in Sections 2.9 and 2.11, different HFR schemes produced different types of distortions: roughness, tonal noises, pinging sounds, and tinkling noises. The informal relative speech-quality judgments of these schemes varied over the listeners, the speakers, and the speech material. The relative quality ratings reported in the preceding sections represented merely our best informal estimates. To decide on the HFR scheme to be used in our final system, we felt that it was necessary to perform a more formal evaluation, using a testing procedure that explicitly takes into account the above-mentioned variations over subjects, speakers, and speech material. Such a testing procedure had been previously developed at BBN as part of a government contract [16]. A specially constructed randomized-order for generating the test stimuli as well as analysis programs for processing the subjective ratings were readily available for a set of six systems. Therefore, we decided to narrow our choices of BBC systems down to six. After the HFR process, the baseband coding issue is the most important one. Based on these and other considerations, we chose six 9.6 kbps BBC systems denoted as SPF, FT1, PF1, FT2, WFR, and PF2. The important characteristics of these systems are shown in Table 2-3. The frame size is 24.0 ms for WFR and PF2, 24.3 ms for FT2, and 24.75 ms for the remaining three systems. Although we tested these coders in the absence of channel bit-errors, each coder has a small number of extra bits per frame reserved for error protection of important transmission parameters.

2.12.2 Description of the Speech-Quality Test

A brief description of the testing procedure is given below. For more details, the reader is referred to [16]. Subjects rated the perceived quality of each system for a set of 36 test sentences, made up of six sentences, each read by six speakers chosen so as to represent the full range of speaker variables found

CODER ID	L, # OF BANDS	# OF LEVELS PER SAMPLE	HFR METHOD	BASEBAND CODING
SPF	3	8	SPECTRAL FOLDING	1-TAP APC
FT1	3	8	SPECTRAL FOLDING WITH PREFLATTENING	1-TAP APC
PF1	3	8	PERTURBED SPECTRAL FOLDING	1-TAP APC
FT2	3	10	SPECTRAL FOLDING WITH PREFLATTENING	APCM
WFR	4	22	RECTIFICATION	APCM
PF2	4	22	PERTURBED SPECTRAL FOLDING	APCM

Table 2-3 Description of the six 9.6 kbps BBC systems included in the subjective speech-quality test

in a group of 20 speakers. The sentences used are given in Table 2-4, and the average pitch values of the speakers are given in Table 2-5. The first four test sentences in Table 2-4 are phoneme-specific, in the sense that each contains all and only the phonemes of a particular type (glides, nasals, fricatives, and stops), together with vowels. The last two sentences are "general" sentences, which contain several consonant clusters and unstressed syllables. The 216 stimulus sentences (6 systems x 6 speakers x 6 sentences) were presented in a counterbalance sequence effects. The first six blocks of six stimuli each were repeated at the end (without the subjects' knowledge) to permit estimation of the reliability and drift of the judgments. Six subjects were used, three of whom were experienced with the systems under test, and three of whom were naive. Quality judgments were made on an 8-point scale, with 8 representing best quality and 1 representing worst quality. The absolute values of the mean quality judgments have no meaning, since the ratings given to a particular system are strongly affected by the context supplied by the quality of the other systems included in the test.

1. Why were you away a year, Roy?

2. Nanny may know my meaning.

3. His vicious father has seizures.

4. Which tea-party did Baker go to?

5. The little blankets lay around on the floor.

6. The trouble with swimming is that you can drown.

Table 2-4 A set of six sentences used in the subjective speech-quality test

Speaker (Initials)	Male/ Female	Ave. Fundamental (Hz)
DK	M	95
JB	M	118
DD	M	139
AR	F	167
RS	F	209
PF	F	232

Table 2-5 A set of six speakers used in the subjective speech-quality test

2.12.3 Results of the Speech-Quality Test

Mean quality judgments for each system are plotted in Fig. 2-12, and the means and their standard deviations for each system, speaker, sentence, and subject are given in Table 2-6. At the bottom of the table, results of t-tests are given. To calculate the t-values, the ratings were arranged as paired samples. To compare two systems A and B, each rating for system A by each subject and for each speaker and sentence was subtracted from the equivalent rating for system B by the same subject and for the same speaker and sentence. The distribution of difference scores was then tested to see if the mean differed from zero. With an N of 216, t-values *correspond closely to z-scores* (i.e. number of standard deviations from the mean in a normal distribution). Since a difference of 3 standard deviations occurs with probability $P < 0.001$, it can be seen that most of the obtained differences were very highly significant. In particular, the systems fell into three groups: (PF1 and FT1), (SPF, FT2, and PF2), and (WFR). The differences within groups were insignificant (except that PF1 was just significantly better than FT1, $P < 0.05$), but the differences between groups were highly reliable. When we averaged the quality judgments separately for male and female speakers, we found that the systems SPF, FT2 and PF2 exhibited the most variability between the two mean-quality scores and the system WFR produced the least

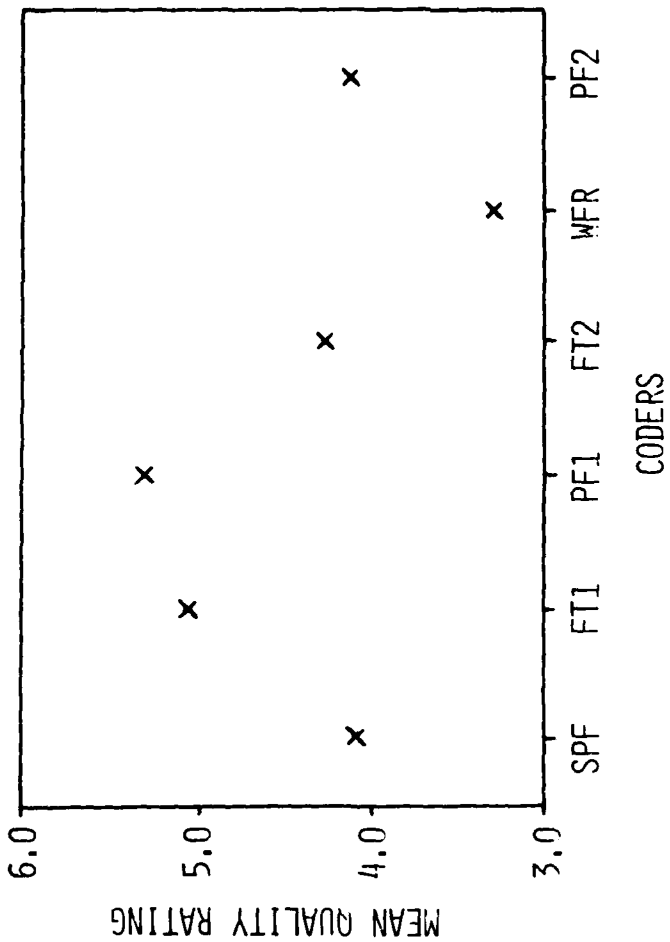


Fig. 2-12 Mean quality ratings of the six 9.6 kbps BBC systems tested

		SPF	FT1	PF1	FT2	WFR	PF2
SYSTEMS	AV:	4.04	5.04	5.29	4.28	3.30	4.13
(N=216)	SD:	1.67	1.69	1.64	1.68	1.67	1.67
		AR	JB	DK	DD	RS	PF
SPEAKERS	AV:	3.83	4.16	4.64	5.12	4.26	4.06
(N=216)	SD:	1.75	1.88	1.73	1.64	1.68	1.79
		1	2	3	4	5	6
SENTENCE	AV:	4.61	4.37	4.00	4.45	4.57	4.07
(N=216)	SD:	1.69	1.67	1.84	1.74	1.93	1.80
SUBJECTS	AV:	4.74	4.74	4.09	4.75	3.38	4.38
(N=216)	SD:	1.40	1.76	1.72	1.27	2.06	2.00
RELIABILITIES	:	.924	.908	.919	.916	.948	.929

T-Tests	SPF	FT1	PF1	FT2	WFR	PF2
SPF		-7.54	-9.01	-1.75	5.68	-0.65
FT1			-2.14	6.54	13.43	7.14
PF1				7.19	14.36	8.49
FT2					7.31	1.18
WFR						-6.91

Table 2-6 Mean 8-point quality ratings and standard deviations for each system, speaker, sentence, and subject (top), and results of t-tests between each pair of systems (bottom). Negative values of T indicate that the system corresponding to the column-label yielded better quality than the system corresponding to the row-label.

variability, with the remaining two systems FT1 and PF1 producing moderate amounts of variability.

From the results presented above, we have the following two conclusions. (1) The perturbed spectral folding method is the best HFR method. (2) APC coding of the baseband residual noticeably improves the perceived speech quality relative to that obtained using APCM coding.

2.13 TRANSMISSION OVER A NOISY CHANNEL

One of the requirements of this project has been to design the 9.6 kbps BBC system to tolerate channel bit-error rates of up to 1%. As an engineering criterion suggested by the COTR, we set our overall objective as follows: The speech quality of the error-protected 9.6 kbps coder at 1% channel bit-errors should be about the same or better than the speech quality of the same coder when it is operated without error protection over a channel with an error rate of 0.1%. Our study was concerned with optimizing 1) the tradeoff between the voice data bandwidth and the error-protection bandwidth, and 2) the amount of error protection for individual transmission parameters. We used the Hamming (7,4) code to protect certain parameter bits but did not protect the coded baseband residual.

Several tests were made to examine how each transmitted parameter, when subjected to 1% channel error, would independently affect the output speech. Results of these tests showed that in coders using APCM, channel errors on the baseband residual samples caused the speech to have a continuously rough or "raspy" character. With the use of APC, the roughness in the output speech was reduced considerably. One reason for this improvement provided by APC relative to APCM is that adequately error-protected and transmitted pitch and pitch tap(s) in the APC system produce the proper pitch periodicity in the output speech; in the APCM system, on the other hand, the channel bit-errors on the unprotected residual samples can distort the pitch periodicity. Errors in the LPC coefficients and quantizer gain caused discrete effects such as "pops" and "clicks" in the speech. The block-coding technique required with the provision of noninteger bits/sample (see Section 2.8.1) was found to produce more noticeable speech quality distortions under channel bit-errors than the integer bits/sample case, since a single bit-error results in the erroneous decoding of a block of baseband residual samples.

Subsequently, we investigated BBC systems using APC for baseband residual coding and obtained the following results. (1) The output speech from coders without any error protection had a "ringing" or reverberant character. Providing error protection for the pitch and pitch predictor tap(s) was found to reduce this effect substantially. (2) Considering the two frame rates of 37 and 40 frames/sec, we found that the lower-frame-rate system produced higher quality speech because of the additional error protection it provided. (3) For a similar reason, the 1-tap APC case yielded better speech quality than the 3-tap case.

Of the various coders we tested, the 1-tap APC-coded low-frame-rate system provided the highest quality output speech. The details of this coder are given in Table 2-7. For this coder, 30 bits per frame were available for error protection (i.e., 40 data bits were protected with ten Hamming codewords). The first three LAR coefficients (coded with 5 bits each) and the pitch predictor tap (4 bits) were protected completely. The three most significant bits (msb) of the fourth, fifth and sixth LAR coefficients, one msb of the seventh and the eighth coefficients, and the of energy and pitch were also protected. This error-protection bandwidth of 30 bits/frame or about 1.1 kbps constitutes about 11.5% of the channel bandwidth of 9.6 kbps. The coder described in Table 2-7 was found to satisfy the above-mentioned engineering error-performance criterion. Also, for input speech from the high-quality data base, the output speech quality of the BBC system was found to be quite good. Thus, the particular allocation of the total channel bandwidth of 9.6 kbps

Frame Size (ms)	Total Bits per frame	Number of samples per frame		Bits per Baseband sample	Bits/frame for parameters				Error Protection	Sync
		speech	Base-band		8 LARs	Gain	Pitch	Pitch Tap		
27.1875 @6.621 kHz	261	180	60	3	33	6	6	4	30	2

(a)

Item	Log Area Ratios								Gain	Pitch	Pitch Tap	Total Bits
	1	2	3	4	5	6	7	8				
Bits for Coding	5	5	5	4	4	4	3	3	6	6	4	49
Bits (msb) Protected	5	5	5	3	3	3	1	1	5	5	4	40

(b)

Table 2-7 Description of the best 9.6 kbps system resulting from our error-protection study.

(a) Values of various coder parameters
(b) Bit allocation for quantization and error protection

that the optimized coder provides for voice data and for error protection adequately satisfies the requirements of this project dealing with the noiseless and noisy channel performances of the coder.

2.14 ACOUSTIC BACKGROUND NOISE

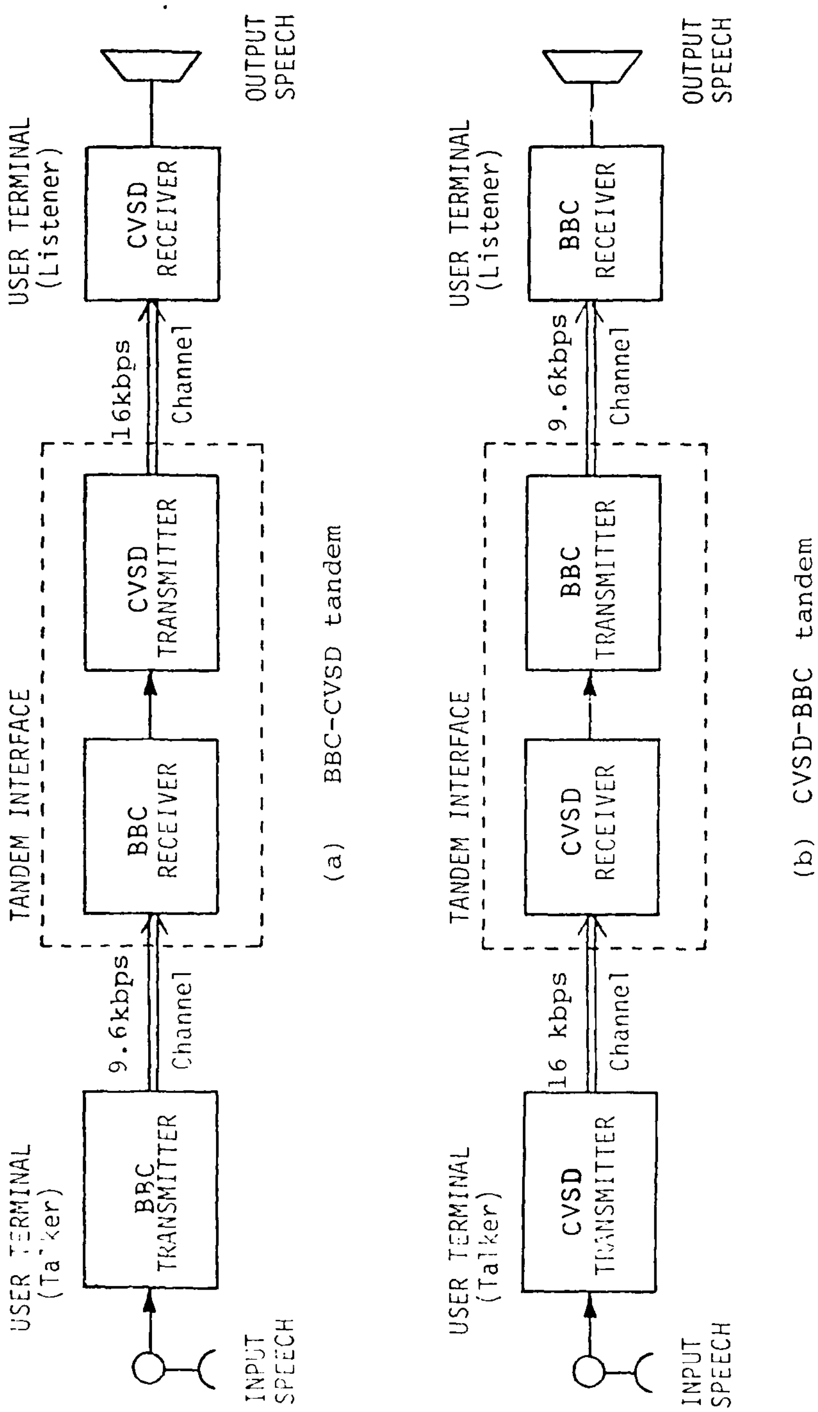
We processed the sentences from the office-noise data base described in Section 2.5.2, using the optimized, error-protected coder (Table 2-7). We found that the output speech quality and intelligibility were good. Interestingly, the quality of the output speech in this case was found to be closer to that of the input speech than was observed using speech from our high-quality data base. Therefore, the speech-enhancement preprocessor that we originally proposed [1] to use at the coder input is not necessary.

2.15 TANDEMING WITH CVSD

It is envisioned that future large secure digital voice communication networks will include digital links of different data rate capacities and user or subscriber terminals equipped with different speech coders we considered. In particular, such a network might include 16 kbps links and CVSD coders on the one hand, and 9.6 kbps links and the optimized BBC coder developed in this project on the other. The need for a tandem interface arises when a user having a BBC coder wants to communicate with a CVSD user. In this case, the tandem interface consists of a BBC decoder followed by a CVSD encoder, as illustrated in Fig. 2-13(a). Similarly, a CVSD-BBC tandem connection is shown in Fig. 2-13(b). An ideal, distortion-free tandem operation, such that the tandem connection be no worse than the poorer of the two coders, is desirable. We have found that the BBC coder produces output speech that has better quality than the CVSD speech provided by the sponsor. Therefore, with ideal coupling, the overall performance should be no worse than that of the CVSD coder alone. A specific requirement of this project stated in Chapter 1 is that the tandem-link should provide minimal degradation in speech intelligibility relative to the 16 kbps CVSD coder.

2.15.1 BBC-CVSD Tandem

Recently, several investigations have studied the tandem connection between a 16 kbps CVSD coder and a 2400 bps LPC pitch-excited vocoder [22-25]. For proper operation of the CVSD coder, its input signal amplitude should change fairly smoothly. If the signal is "peaky" in that its peak-to-rms ratio is high, then the CVSD coder will have increased slope overload noise. In fact, the output signal of the pitch-excited LPC vocoder is peaky. To improve the tandem performance of the LPC-CVSD link, several researchers have suggested various forms of phase modification [22,23]. Such phase modification is not necessary in a BBC



(a) BBC-CVSD tandem

(b) CVSD-BBC tandem

Fig. 2-13 Tandem operation of BBC and CVSD coders.

application, because unlike the impulse sequence used in the LPC vocoder, the excitation signal of the BBC coder is not minimum-phase. Consequently, as we have verified, the output from the BBC coder is not peaky. We believe that a BBC-CVSD tandem-link will behave quite similarly to a single CVSD link.

2.15.2 CVSD-BBC Tandem

To evaluate the CVSD-BBC tandem link performance, we processed the sentences from the CVSD data base (Section 2.5.2) using the optimized BBC coder (Table 2-7). The output speech had a slight additional roughness, but otherwise was found to have quality and intelligibility approximately similar to the CVSD speech.

We investigated the effect on the tandem-link performance of using a speech-enhancement preprocessor to enhance the CVSD speech prior to processing through the BBC coder. The details of the speech-enhancement algorithm (previously developed at BBN) we used are given in [26]. The enhancement procedure requires an estimate of the noise spectral amplitude, which is usually obtained during the silence region in the beginning of a sentence. However, for CVSD speech, the quantization noise is signal-dependent, and during non-speech regions it was found to be negligible. Hence, we took the approach of assuming the noise to be white with a constant spectral amplitude, and we tried different values for this constant with the goal of maximizing the enhancement for the sentences from the data base. We then processed the CVSD speech through the optimized 9.6 kbps BBC coder with and without the speech enhancement preprocessing, and compared the speech quality of the resulting coder outputs. The coder output speech quality improved only marginally as a result of the enhancement. Therefore, we judged that the substantially additional computation required for the enhancement process was not worthwhile.

2.16 OPTIMIZED 9600 BPS CODER

In this section, we summarize the details of the optimized 9600 bps BBC system. Before we proceed with the summary, we discuss issues dealing with simplification of two aspects of the coder for facilitating its real-time implementation.

2.16.1 HFR Implementation

We explored two methods of simplifying the implementation of the perturbed spectral folding technique, which had been judged to be the best HFR scheme in our formal subjective speech quality test described above. The first method was concerned with reducing the rate of random number generation. In the original perturbation scheme, a random number was generated for each baseband sample, and the decision to perturb the sample was based on the magnitude of the random number. We modified the scheme so that for each perturbed sample, a random number was drawn, giving the number of samples tc pass before the next perturbation. Clearly this modification reduces the rate of random number generation, but it was found to be less effective in masking tonal noises. In the second method of simplification, we obviated the need for on-line generation of random numbers by storing a prefabricated sequence of random numbers and using them cyclically; this simplification did not produce any additional speech quality degradation. Therefore, we decided to simplify the HFR scheme by the second method.

2.16.2 Filter Order

In our simulation of the optimized coder, we employed 3 FIR filters, each of order 201. One filter (lowpass) was used at the transmitter for extracting the baseband and two filters (one lowpass and one highpass) were used at the receiver as part of the HFR scheme (see Fig. 2-9). For the real-time implementation of the BBC system, we had to reduce the order of the three FIR filters to 75. Fig. 2-14 shows the amplitude response of the lowpass filter that we chose to implement. The filter has a transition width of about 188 Hz and a stop-band attenuation of about 35 dB. When we used the lower order filters in our simulation, the output speech quality suffered only a slight loss relative to the case involving order 201 filters.

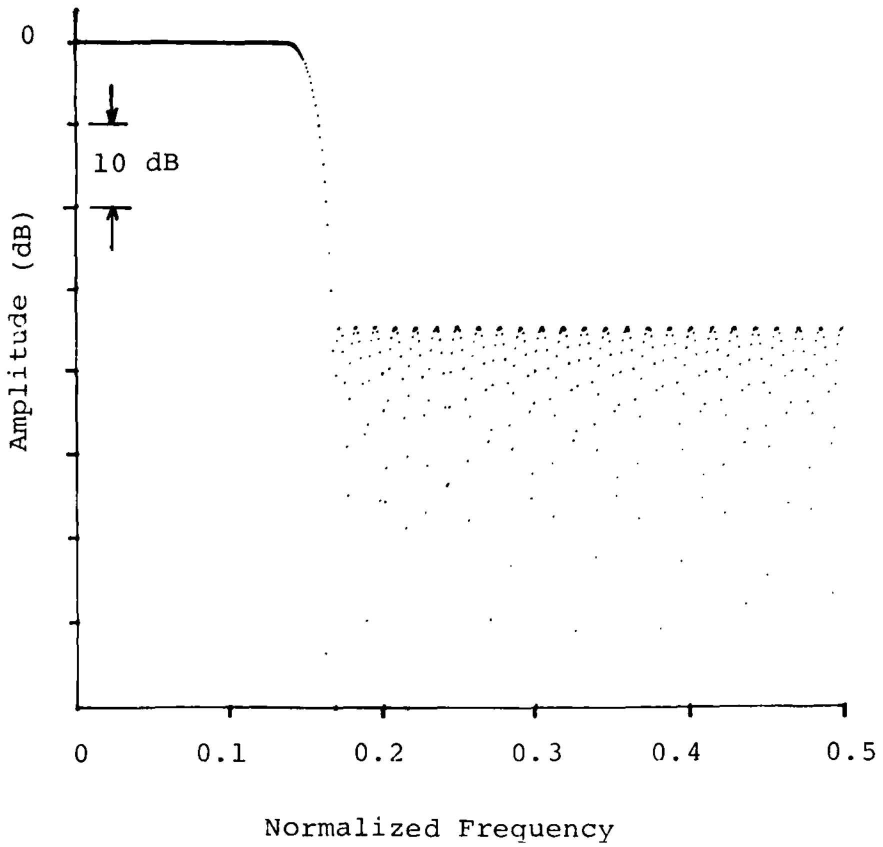


Fig. 2-14 Amplitude response of the 75-th order FIR lowpass filter chosen for use in the real-time BBC system. (Normalized frequency of 1 corresponds to the full sampling rate of 2W Hz.)

2.16.3 Optimized Coder

A block diagram of the optimized coder is shown in Fig. 2-15. Table 2-8 provides information regarding the coding and error protection of parameter data of the BBC system. At the transmitter, the analog input speech is lowpass filtered at 3.2 kHz and sampled at 384/58 (or about 6.621) kHz. Referring to Fig. 2-15(a), the sampled speech s(t) is analyzed over 180 speech samples (once every 27.1875 ms). The LPC analysis consists of removing the short-term dc bias from the speech samples, Hamming windowing, and using the autocorrelation method of linear prediction to compute 8 reflection coefficients. The reflection coefficients are quantized using log area ratios (LARs), with a total of 33 bits as indicated in Table 2-8. Predictor coefficients are obtained from the quantized LARs and are used in the inverse filter A(z) to compute the residual e(t). e(t) is decimated 3:1 to obtain the baseband residual by lowpass filtering e(t) with a 75-th order FIR filter having its stop-band edge at about 1.1 kHz and retaining every 3rd filtered sample. Pitch analysis of the baseband consists of computing the autocorrelation function of v(t) for lags 5-45, from an interval of 90 baseband samples (60 from the current frame and 30 from the last frame), determining the pitch value M as the peak of this function, and computing the pitch tap c using Eq. (2.5). The pitch tap is linearly quantized using 4 bits

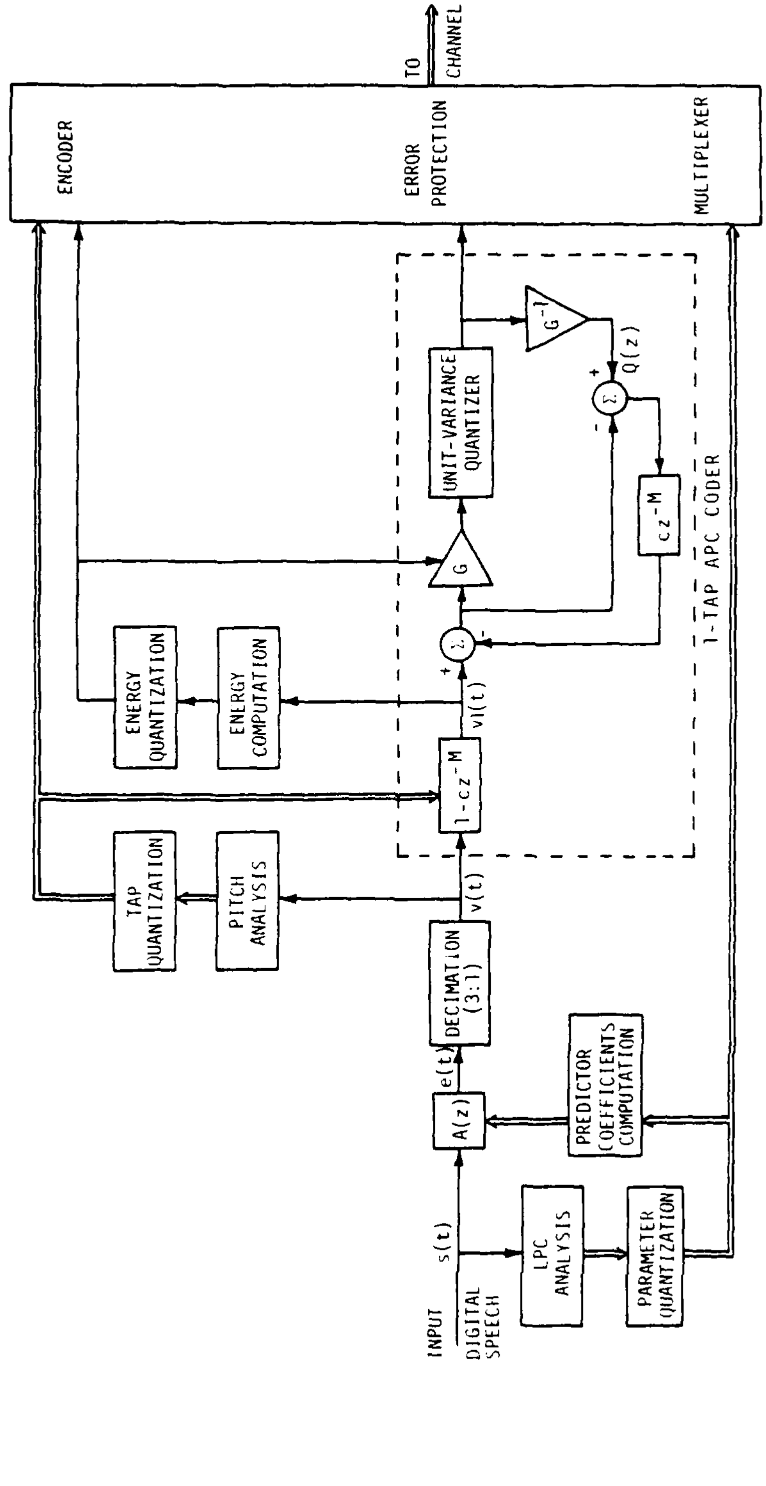


Fig. 2-15(a) The transmitter of the optimized 9.6 kbps BBC system

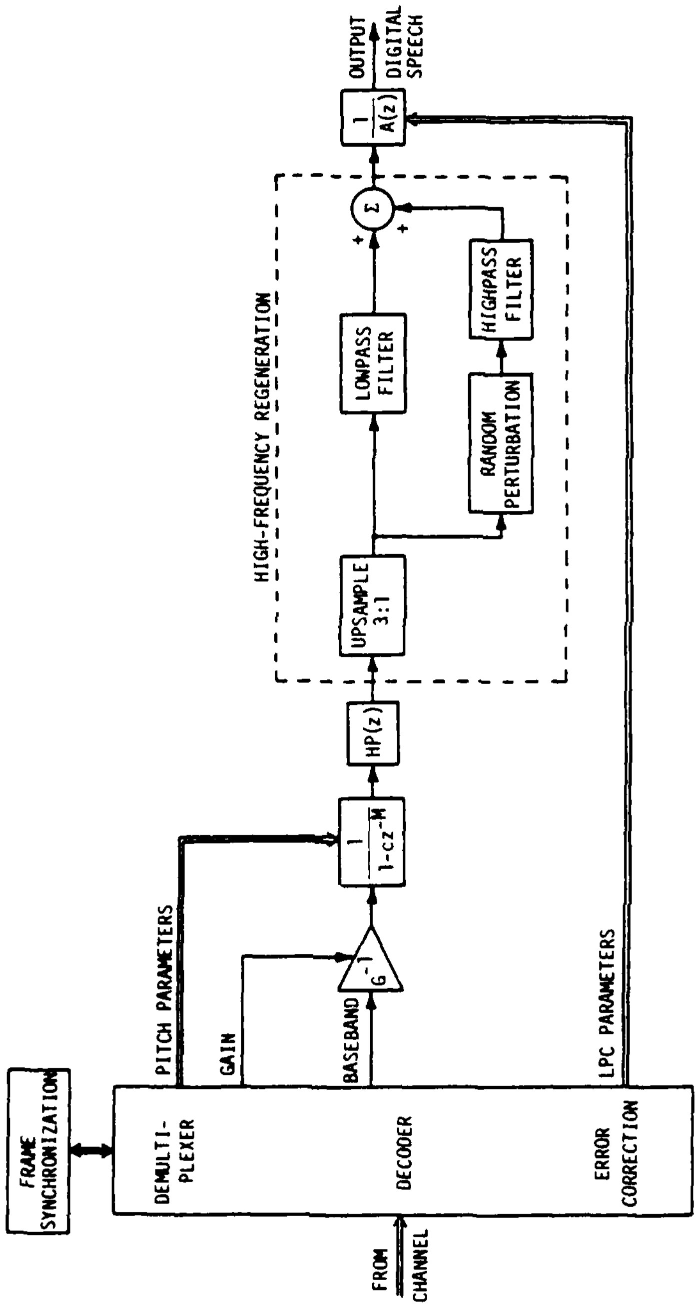


Fig. 2-15(b) The receiver of the optimized 9.6 kbps BBC system

Parameter		Min	Max	Step Size	# of Bits	# of Bits Protected
Second residual energy (dB)		-18.0	36.0	0.84	6	5
Pitch (# of baseband samples)		5	45	No Quantization	6	5
Pitch tap		0.0	1.0	0.0645	4	4
Log Area Ratios (dB)	1	-16.854	3.498	0.636	5	5
	2	- 4.199	16.473	0.646	5	5
	3	-13.325	7.475	0.650	5	5
	4	- 2.828	10.100	0.808	4	3
	5	- 6.061	5.348	0.713	4	3
	6	- 2.723	9.725	0.778	4	3
	7	- 5.391	4.193	1.198	3	1
	8	- 2.558	5.627	1.023	3	1
Total bits per frame					49	40

Table 2-8 Quantization and error-protection of parameter data for the optimized 9.6 kbps BBC system.

(see Table 2-8). The quantized 1-tap filter $(1-c\ z^{-M})$ is used to compute from v(t) the second residual vl(t). The energy of vl(t) over the current frame is computed and quantized using 6 bits (see Table 2-8). The gain G within the 1-tap APC coder is set equal to the reciprocal of the quantized energy. The second residual vl(t) is then coded using the 1-tap APC coder, which contains a 3-bit minimum-mean-square-error nonuniform Laplacian unit-variance quantizer. The symmetric quantizer has its input boundaries at 0, 0.531, 1.248 and 2.371, and the output values 0.233, 0.830, 1.666, and 3.075. The quantized data, APC-coded samples, LARs, pitch tap, and energy of the second residual, and the unquantized pitch are all binary encoded, error protected using 10 Hamming (7,4) codes (see Table 2-8), multiplexed with one synchronization bit, and transmitted over the channel. There is one unused bit, since the BBC system uses 260 bits out of the available 261 bits per frame.

At the receiver shown in Fig. 2-15(b), the received data are demultiplexed, decoded, and error-corrected. The decoded baseband samples are divided by the quantizer gain G and filtered by the pitch-synthesis filter $1/(1-cz^{-M})$. The output of the pitch filter is highpass filtered with a second-order Butterworth filter HP(z) given in Eq. (2.7). High-frequency regeneration is performed next using the perturbed spectral folding method described in detail in Section 2.9.5. The lowpass and highpass filters within the HFR process are both 75-th order FIR filters with the stop-band edge at about 1.1 kHz. The regenerated fullband excitation signal is then applied to the linear prediction all-pole filter 1/A(z) to produce the digital speech output. This digital output is passed through a D/A converter and an analog lowpass filter with its cutoff at 3.2 kHz to produce the analog output speech.

The COTR was supplied with an audio demonstration tape in July 1979. The tape contained the recordings of the output speech obtained from the simulation of the above-described 9.6 kbps BBC system. The four sections on the tape successfully demonstrated the performance of the optimized coder, respectively, for high-quality input speech, in office-noise environment, over a noisy channel at 1% bit-errors, and in tandem with a 16 kbps CVSD coder. In each of these cases, the coder performance met and surpassed the requirements stated in Chapter 1.

3. REAL-TIME IMPLEMENTATION

3.1 OVERVIEW

This section describes the function, components, and design of the real-time speech coder system. The other sections in this chapter describe in more detail the operation of the speech coder system, the hardware and software used to construct it, the real-time performance of the completed system, and the rationale for aspects of the system design.

3.1.1 System Function

The speech coder system is a full duplex terminal of a complete communication system. It is intended to be connected via a 9600 bps digital I/O link through a communication channel to an identical system. The speech coder system functions simultaneously as both a transmitter and a receiver.

3.1.2 System Components

The speech coder system contains both hardware and software elements. The hardware elements include a CSP Inc. MAP-300 array processor attached to a PDP-11, a handset including microphone and earphone, a hookswitch, amplifiers and low-pass filters, and digital line interfaces. The software elements include MAP-300 programs, which comprise the real-time software, and the program that runs on the PDP-11, which is used only to load and initialize the MAP software. Figure 3-1 is a block diagram of the system.

3.1.3 System Design

The real-time speech coder consists of six separate foreground processes. The Transmitter requires an analog-in process, an analysis process, and a digital-out process. The Receiver requires a digital-in process, a synthesis process, and an analog-out process. Since all of these processes make use of the MAP-300 CSPU to some extent, a mechanism for scheduling them is necessary. Part of this mechanism is contained in the MAP hardware interrupt structures and the SNAP-II executive program. The rest is implemented using flags in conjunction with a background process running in the CSPU. Figure 3-2 is a diagram of these processes. The processes share data buffers and communicate the status of these buffers through flags. Since the four I/O processes are continuous, pairs of double buffers are used to enable the necessary sharing.

Each I/O process includes an interrupt service routine, which handles the flags and transfers data to or from the shared buffers. Because of constraints imposed by the MAP architecture, two levels of double buffering are provided within each I/O process to maintain acceptable system performance under real-time exception conditions.

AD-A083 079 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR--ETC(U)
FEB 80 R VISWANATHAN, J WOLF, L COSELL DCA100-79-C-0003
UNCLASSIFIED BBN-4327-VOL-1 NL

2 of 3
AD
A083079

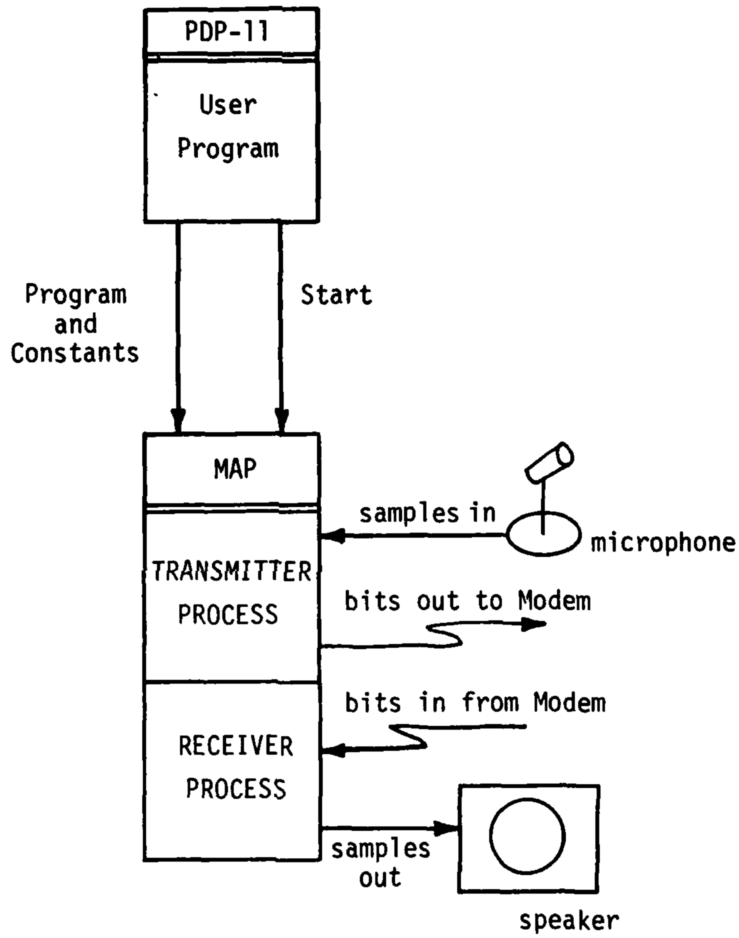


Figure 3-1. System Components

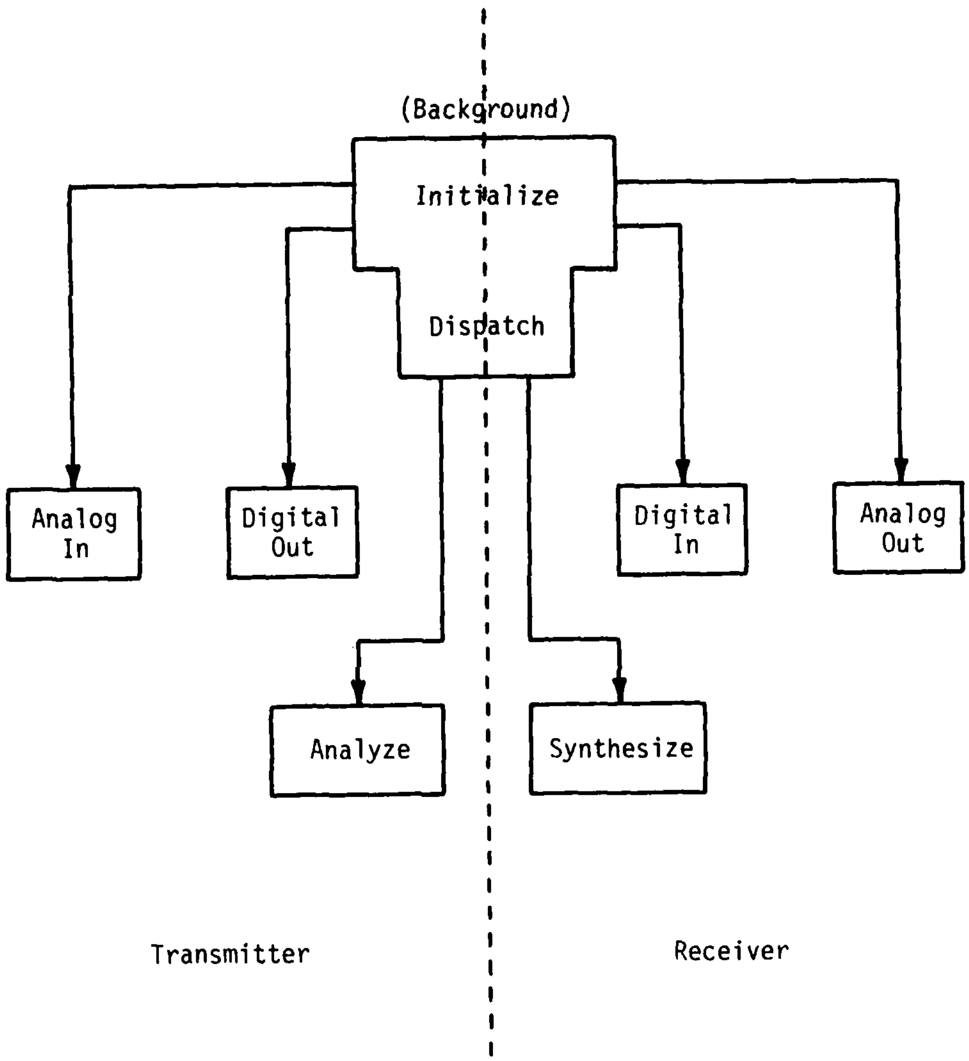


Figure 3-2. System Structure

The background process is shown in Fig. 3-3. The ANALYZE and SYNTHESIZE modules are logically asynchronous; they respectively belong to the independent Transmitter and Receiver. However, the modules must be controlled by a sequential machine, the CSPU. The control strategy for allowing each module as much flexibility as possible results in a structure that executes a module only if that module's input and output buffers are available.

The Initialization module is the first module executed, and it is executed only once. It sets all buffer flags to indicate empty, loads and starts the programs in the various I/O Scroll processors, and enables interrupts from these processors.

Control then passes to the basic loop of the background process. This loop executes the ANALYZE and SYNTHESIZE modules when the required I/O buffers are available. For example, the SYNTHESIZEA module will be executed only when the RBITSB buffer is full and the RSINKA buffer is empty.

The ANALYZEA and ANALYZEB modules are functionally equivalent, differing only in their input and output buffers. The structure of the SNAP-II programming environment (specifically, the use of prebound functions for run-time efficiency) does not allow changing the buffers used in a particular function. Therefore, two separate functions must be created to deal with the two sets of

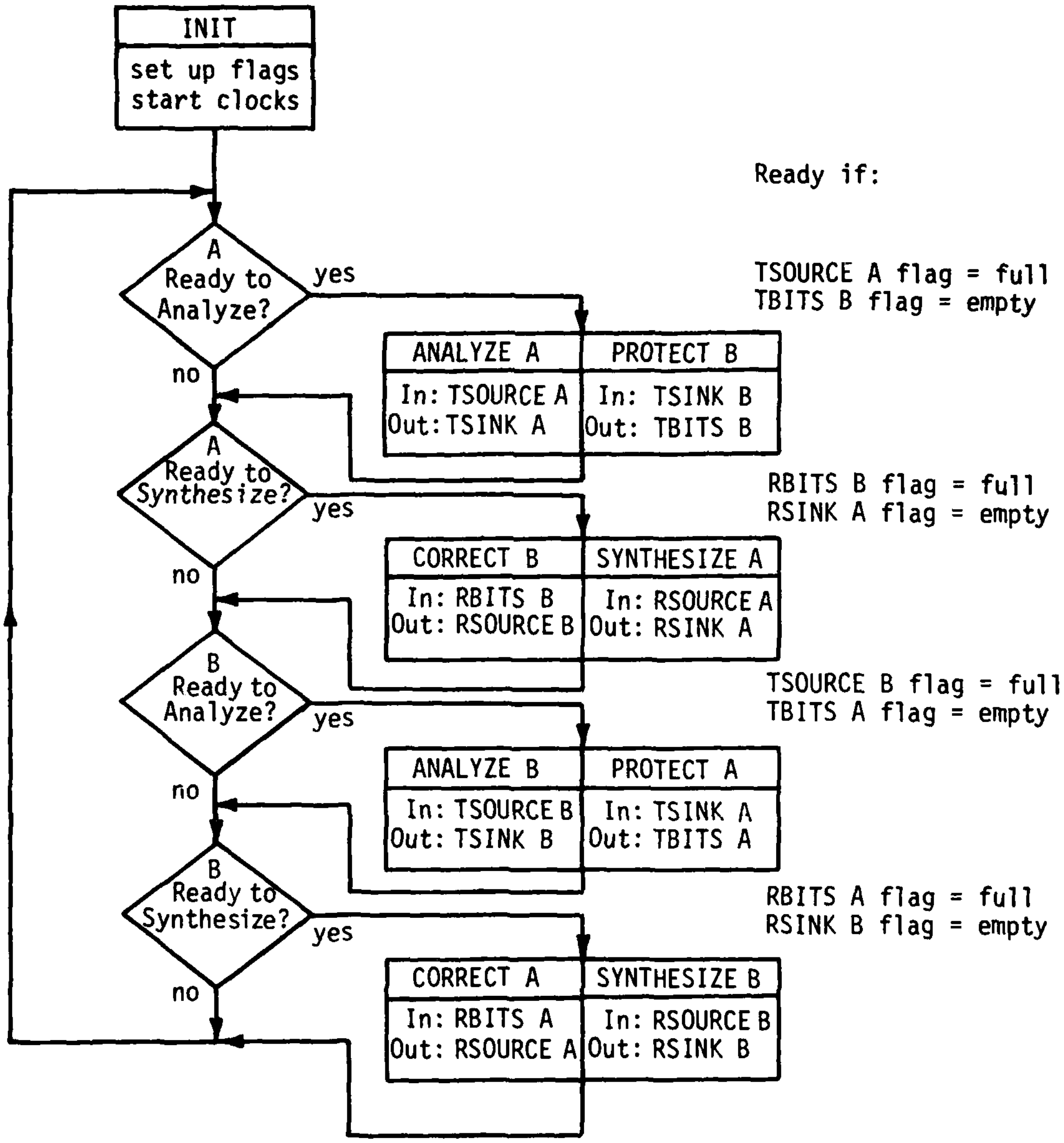


Figure 3-3. Background Process Loop

TSOURCE and TBITS buffers, A and B. Similarly, two separate SYNTHESIZE functions are required.

All of the foreground processes are loosely coupled. Execution of neither the ANALYZE nor the SYNTHESIZE module is connected rigidly to any I/O process.

3.2 SYSTEM OPERATION

The BBC speech coder system functions as one terminal of a full duplex digital voice connection. It accepts voice input, digitizes it, and processes it. The processed speech is transmitted as a sequence of bits to a similar terminal. The system also accepts a sequence of bits representing speech transmitted from a remote terminal and processes this sequence to obtain synthetic voice output.

3.2.1 Transmitter

The Transmitter is shown in Fig. 3-4. The low-pass filtered input speech is fed into an A/D converter contained in an I/O Scroll processor (the ADAM, or Analog Data Acquisition Module), a component processor of the MAP. The program running in this scroll controls the sampling of the speech data, transferring 180 samples (one frame) into each of two buffers, alternately. The sampling rate is 6.621 kHz, so each frame is about 27.2 ms long.

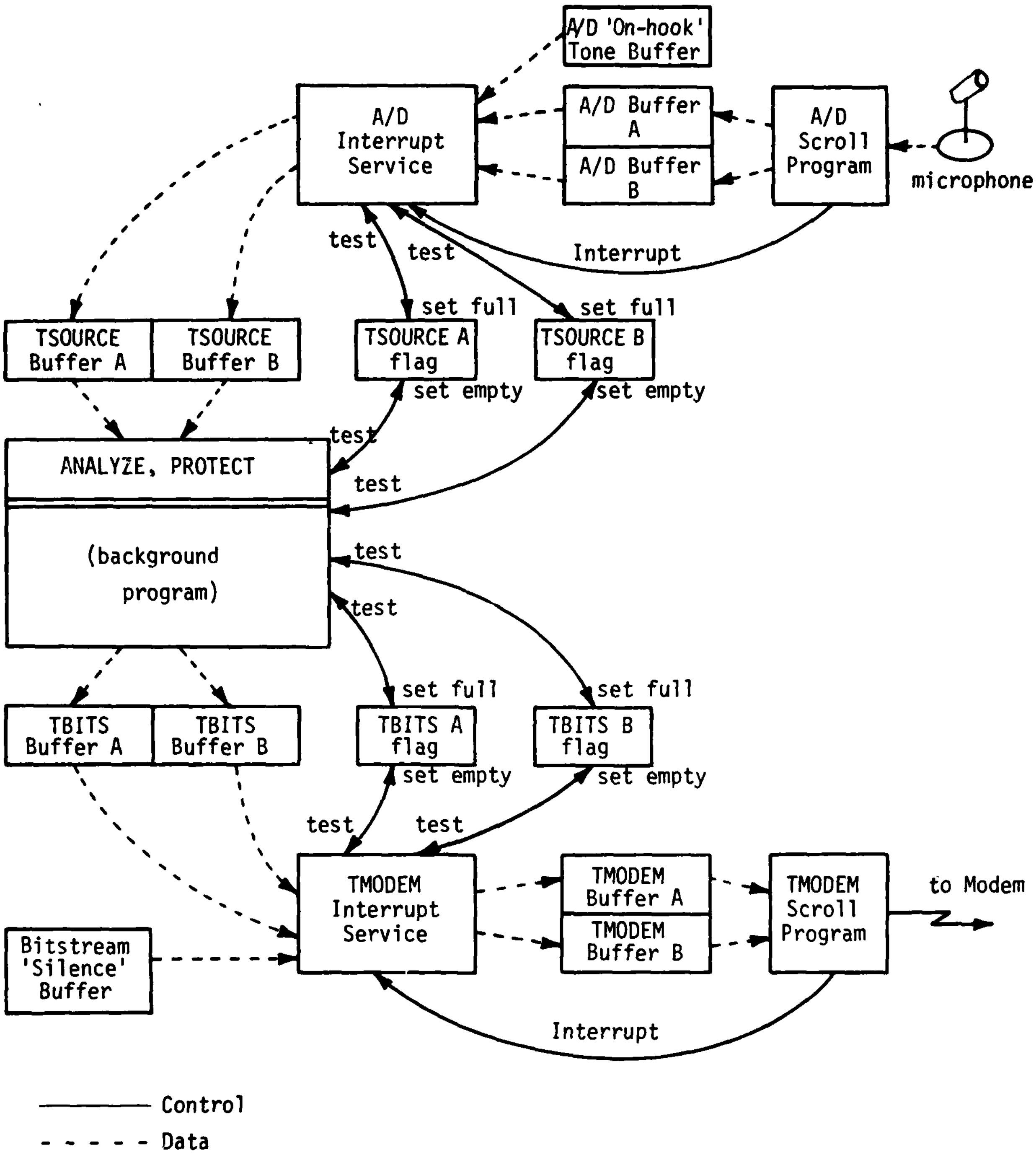


Figure 3-4. Transmitter Process

When the ADAM fills one of these buffers, it generates an interrupt to the main MAP processor, the CSPU. The A/D Interrupt Service routine, which is activated by this interrupt, transfers the data from the just-filled A/D buffer to the current one of the two TSOURCE buffers (if the current one is empty), and sets the corresponding TSOURCEA or TSOURCEB flag to indicate that the buffer is now full. If the current TSOURCE Buffer is not empty, the A/D buffer contents are discarded.

The data in the TSOURCE buffers is used as input by the ANALYZE program module. The Background Process (Section 3.1.3) tests the TSOURCE flags to determine when data is available to the ANALYZE module. After processing a TSOURCE buffer, ANALYZE clears the corresponding flag to indicate that the buffer is empty. The ANALYZE module processes the data, producing coded speech, which is then written into one of the (empty) TSINK buffers. The ANALYZE module runs in the CSPU and AP at background level and is therefore asynchronous with the interrupt service routines.

After ANALYZE fills a TSINK buffer, the PROTECT module transforms the data from this buffer into an error-protected bitstream. This bitstream is stored in one of the (empty) TBITS buffers, and the corresponding TBITS flag is set to indicate full. The PROTECT operation logically follows ANALYZE, in that data must be processed by ANALYZE before it is available to PROTECT.

However, the PROTECT module is actually executed concurrently with the next execution of the ANALYZE module. In fact, the PROTECT module appears as part of the ANALYZE function list. This means that the first time the ANALYZE and PROTECT modules are executed, the PROTECT module will have no meaningful data on which to operate. This problem is circumvented by initializing the contents of the TSINK buffers to coded silence.

The data processed by the PROTECT module will eventually be output to the modem by the TMODEM program, running in another I/O Scroll processor. This program takes data, in the form of a bitstream, from the TMODEM buffers and puts it out to the modem. When the TMODEM buffer is emptied, the TMODEM scroll program generates an interrupt to the CSPU and begins taking data from the other TMODEM buffer.

This interrupt causes execution of the TMODEM Interrupt Service routine. This routine transfers data from a full TBITS buffer to the just-emptied TMODEM buffer and clears the corresponding TBITS flag to indicate empty. If the TBITS buffer is not full, a buffer corresponding to coded silence is transferred to the TMODEM buffer.

3.2.1.1 ADAM Scroll Program (ADPROG)

Speech input is performed by the ADAM, an IOS-2 scroll processor that contains an analog multiplexer, sample/hold, and A/D converter. The ADAM receives both the input speech signal and the input sampling clock from the Speech Processor Interface (SPI). The signal is in the range -5 to +5 volts, and the sampling rate is 6.621 kHz.

The ADAM program (ADPROG) is shown in Fig. 3-5. The speech samples are double-buffered in A/D input buffers named TADBA and TADBB. (All MAP buffers and most scalars are prefixed with "T" or "R" to indicate that they are used in the speech coder Transmitter or Receiver respectively.) The A/D input buffers are 180 samples long, which corresponds to a frame length of about 27.2 ms. The A/D samples are written into memory in short (16-bit) floating point format. (The ADAM and AOM are capable of working only in 16-bit floating or 16-bit fixed point formats.)

ADPROG sets the ADAM multiplexer to Channel 1 and sets the F1 flag to initiate sampling. ADPROG maintains a pointer into the A/D input buffer, which is initialized to TADBA-1. When a sample is converted, this address pointer is incremented and used to transfer the A/D sample to MAP memory. The address pointer is compared to the buffer end address; when the end has been reached, a line 1

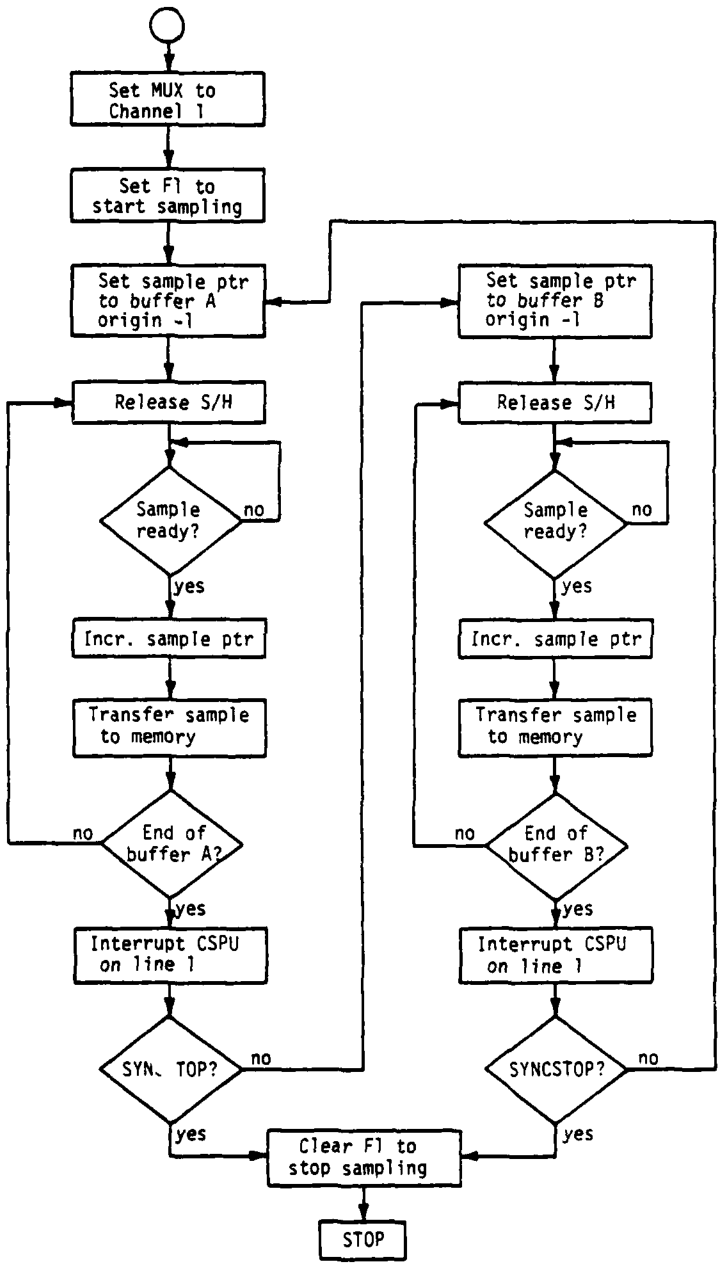


Figure 3-5. ADPROG: ADAM Scroll Program

interrupt is passed to the CSPU to signal the filling of the buffer, and the SYNCSTOP register is checked. SYNCSTOP is a register that is set by the CSPU to signal the ADAM to stop sampling. In normal operation, it remains clear, so ADPROG resets the address pointer to the other A/D input buffer (TADBB), and speech sample input proceeds without interruption. Interrupt 1 from the ADAM is used to signal the filling of each A/D buffer and the switch to the other A/D buffer.

3.2.1.2 A/D Interrupt Service (ADAMINT)

ADAMINT is activated by each ADAM line 1 interrupt, signifying the filling of another A/D input buffer. ADAMINT, like the other three speech coder input/output interrupt service routines, maintains a pair of integer scalar "pointer offsets" to keep track of its input and output buffer relationships. These pointer offsets, which take on the values -2 and 0, are used to reference small tables of address pointers. Thus, for example, ADPO ("A/D Pointer Offset") references ADBPTR, a table of pointers to the two A/D input buffers, and TSRPO ("TSOURCE Pointer Offset") references the tables TSRBPTR and TSRFPTR, which point to TSOURCE buffer-copying subroutines and buffer-status flags.

The operation of ADAMINT is shown in Fig. 3-6. When it receives a buffer-filled (line 1) interrupt, ADAMINT switches ADPO

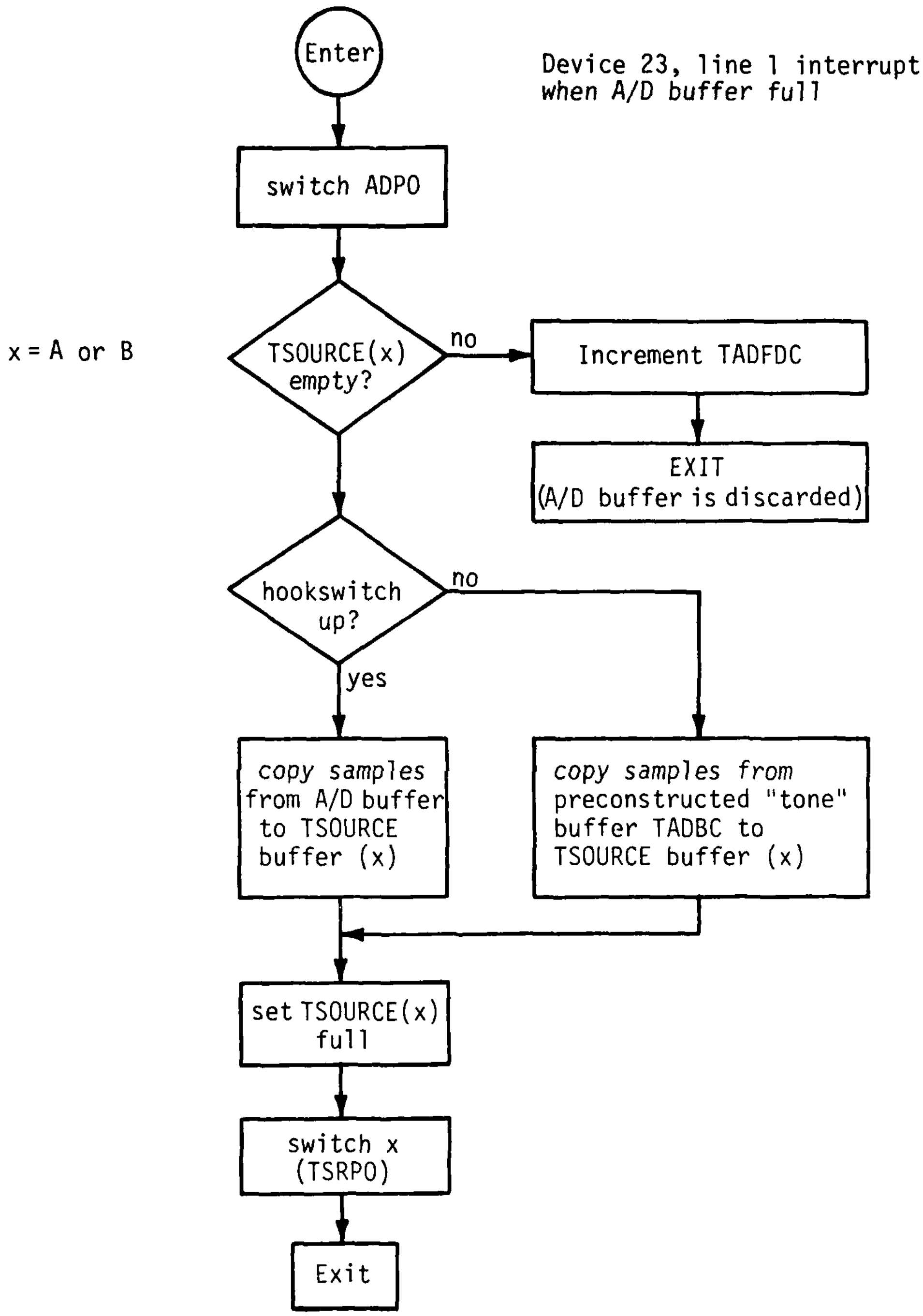


Figure 3-6. A/D Interrupt Service Routine

to the buffer just filled by the ADAM. Then it checks the status of the next TSOURCE buffer to be filled; if it is available (empty) and the handset hookswitch is up (indicating that the handset is in service), ADAMINT copies the new A/D buffer to the current TSOURCE buffer. If the TSOURCE buffer is empty but the hookswitch is down, a preconstructed "tone" buffer is copied instead to the TSOURCE buffer. (This simulated input is intended to indicate to the person using the remote speech coder that this coder is in operation, but the handset is resting in the holder.) In either case, the TSOURCE buffer flag is set to nonzero to indicate full, TSRPO is switched, and the routine exits. If, on the other hand, the TSOURCE flag shows that the current TSOURCE buffer is not empty and therefore unavailable, ADAMINT simply exits, effectively discarding the new A/D data. An "A/D Frame Discard Counter" (TADFDC) is incremented by one to keep track of the fact that an A/D buffer was discarded.

3.2.1.3 ANALYZE Module

The ANALYZE Module is shown in Fig. 3-7. It consists of two control functions and five processing modules.

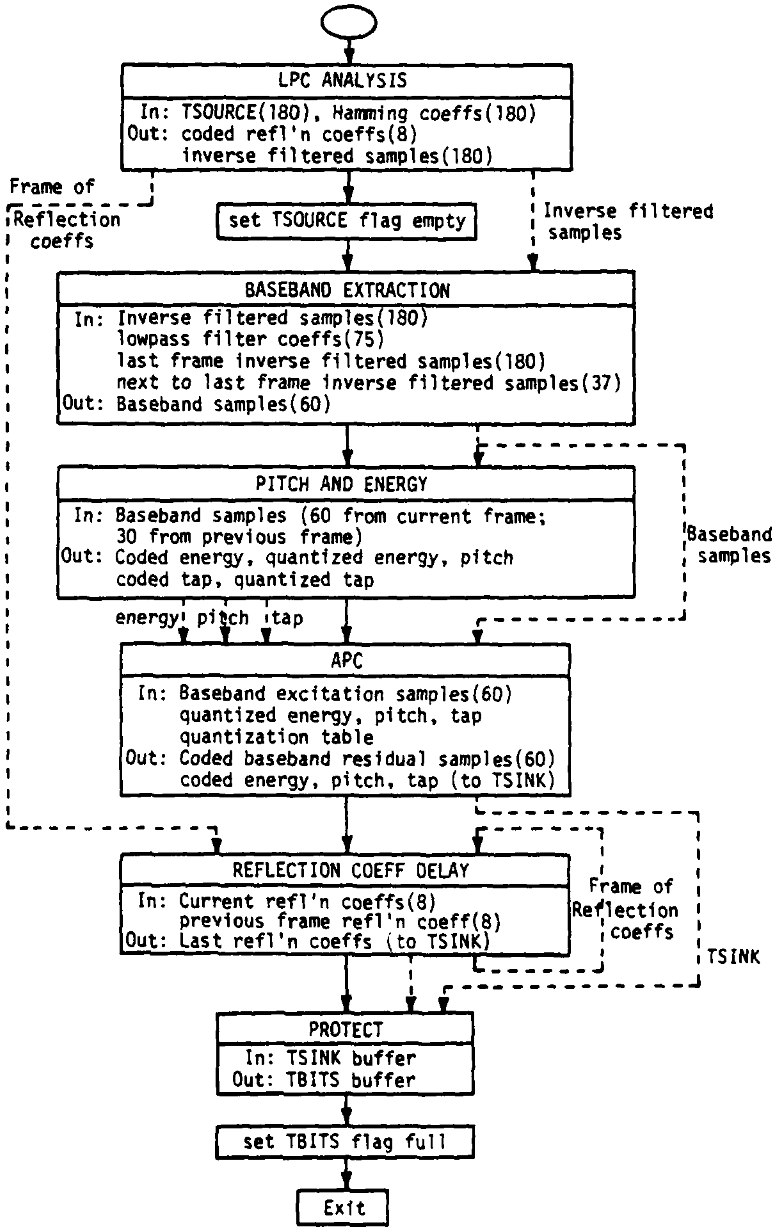


Figure 3-7. ANALYZE Flow Chart

3.2.1.3.1 LPC Analysis Module (Figure 3-8)

Input: TSOURCE buffer of 180 samples
Hamming window coefficients (180)
8 coded reflection coefficients from previous frame

Output: 8 coded reflection coefficients
180 inverse filtered samples
8 coded reflection coefficients from previous frame

Method: Perform LPC analysis using the autocorrelation method, to derive quantized LPC coefficients and coded reflection coefficients, then inverse filter the input samples and the last 8 samples of the previous frame by the quantized LPC coefficients.
Delay the coded reflection coefficients by one frame.

The LPC Analysis Module consists of the following sub-modules:

Remove DC; Multiply by Hamming window

Input: TSOURCE buffer (180)(type: short real)
Hamming window coefficients (180)

Output: Windowed samples (DC removed) (180)

Method: X´(I)=(X(I)-(SUM(X(I))/N))* HAMMING COEFF(I)
where N = number of samples in frame = 180

Autocorrelate

Input: Windowed samples (DC removed) (180)

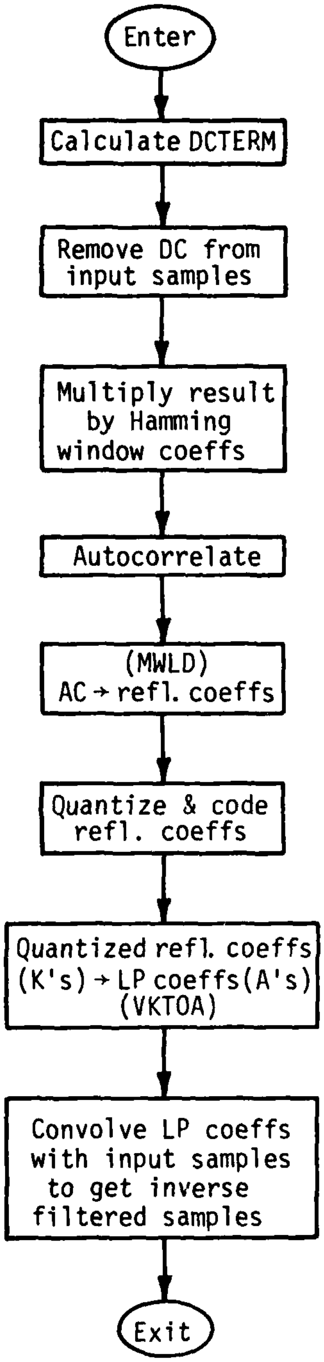
Output: Autocorrelation values (9)

Method: Autocorrelate by padding input with 8 zeros.

Delay Reflection Coefficients

Input: 8 coded reflection coefficients (from previous frame)

Output: 8 delayed coded reflection coefficients



$$\text{DCTERM} = \frac{1}{N}\sum_{i=0}^{N-1} x(i), \quad N=180$$

$x'(i) = (x(i) - \text{DCTERM})*\text{Hamming window coeff}(i)$

$x(i) = i^{th}$ input sample in current frame

$x'(i) = i^{th}$ output sample (DC removed)

SNAP-II Library Function

$A_m(m) = K_m$

$A_m(j) = A_{m-1}(j) + K_m A_{m-1}(m-j), \quad 1 \le j \le m-1$

Figure 3-8. LPC Analysis Module

Compute, Quantize and Code Reflection Coefficients

Input: Autocorrelation values (9)

Output: Quantized reflection coefficients (8)
Coded reflection coefficients (fixed point) (8)

Method: Use modified version of SNAP-II expanded library function 'MWLD' to generate reflection coefficients and quantize and code them, using table lookup

Compute LP Coefficients

Input: Quantized reflection coefficients (8) (current)

Output: LP coefficients (9)

Method: Use recursion formula: $A_m(m) = K(m)$
$A_m(i) = A_{m-1}(i) + K(m)A_{m-1}(m-i)$
for $1<i<m-1$; $m=1,2,\ldots 8$
$A(0)=1$

Compute Inverse Filtered Samples

Input: LP coefficients (9)
TSOURCE buffer (180)
Last 8 samples from previous frame TSOURCE

Output: Inverse filtered samples (180)

Method: Convolve TSOURCE buffer (plus 8 samples from last frame) with LP coefficients.

3.2.1.3.2 TSOURCE Flag Control Module

When the LPC Analysis module has completed, the appropriate TSOURCE flag is set to empty to indicate that the associated TSOURCE buffer is again available.

3.2.1.3.3 Baseband Extraction Module (Figure 3-9)

Input: 180 inverse filtered samples (from LPC ANALYSIS module)
180 inverse filtered samples from the previous frame
37 inverse filtered samples from the second previous frame.
75 low-pass filter coefficients

Output: Current frame of 60 filtered and downsampled baseband residual samples corresponding in time to the previous frame of input samples.

Method: Use demultiplexing convolution routine (from standard library) on array consisting of last 37 samples of (n-2)nd frame, 180 samples of (n-1)st frame, and first 37 samples of nth (current) frame

3.2.1.3.4 Pitch and Energy Determination Module (Figure 3-10)

Input: 60 samples of baseband residual from current frame
Last 30 samples of baseband residual from previous frame
Quantization/coding table

Output: Coded gain
Quantized gain
Pitch
Coded tap parameter
Quantized tap parameter

Method: Find location and value of peak autocorrelation. Remove pitch from baseband residual.
Find energy of pitch-removed baseband residual, use it to find coded and quantized gain.

This module consists of the following sub-modules:

Determine Pitch and Tap (Figure 3-11)

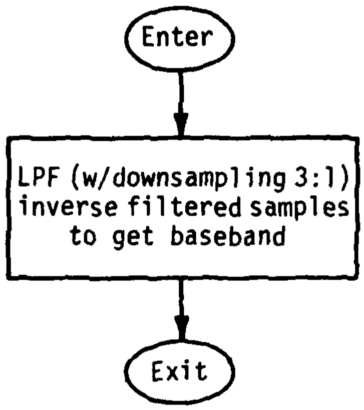


Figure 3-9. Baseband Extraction Module

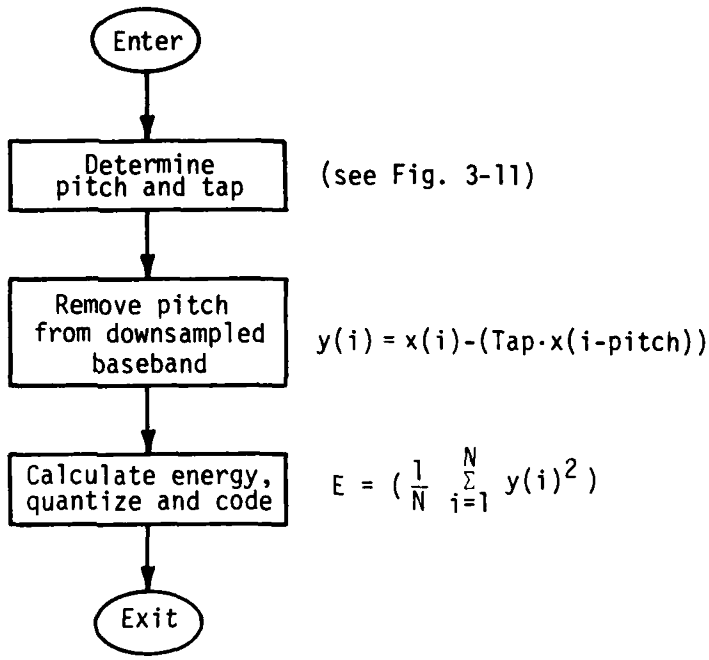


Figure 3-10. Pitch and Energy Determination Module

Input: 60 baseband residual samples from current frame
Last 30 baseband residual samples from previous frame
Quantization/coding table

Output: Pitch
Quantized tap
Coded tap

Method: Compute autocorrelation R(I), I=0,38 of current frame baseband plus half of previous frame baseband
Find positive maximum of R(I)'s between I=5 and I=38. Set pitch equal to value of I at max(R(I)). Set tap equal to max. (R(I))/R(0). Quantize and code tap via table lookup.

Remove Pitch from Baseband

Input: Baseband residual samples from current frame (60)
Baseband residual samples from previous frame (60)
Pitch
Quantized tap

Output: Pitch-removed baseband samples (60)

Method: y(i) = x(i) - (Tap)(x(i-pitch))

Calculate Energy, Quantize and Code

Input: Pitch-removed baseband residual samples(60)
Quantization/coding table

Output: Coded gain
Quantized gain, 1/quantized gain

Method: Energy = (SUM(Y(I)**2))/N, Gain = SQRT(Energy)
Quantized gain, coded gain, by table lookup. Square root and inverse operations performed in same table look-up.

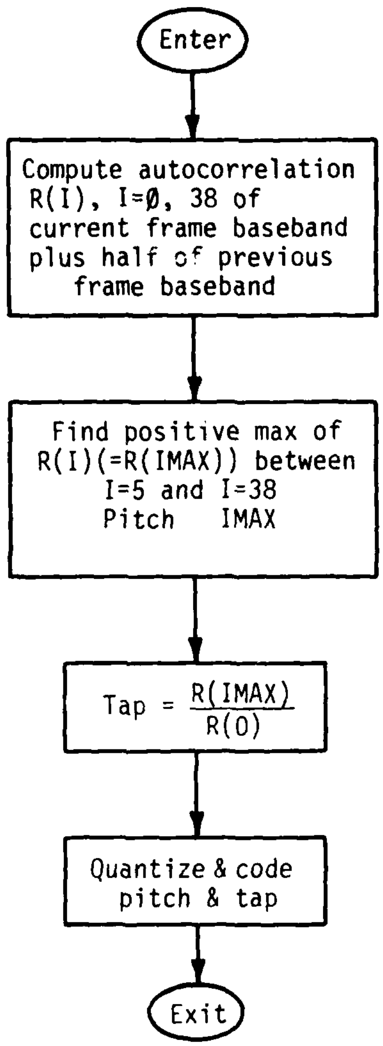


Figure 3-11. Determine Pitch and Tap

3.2.1.3.5 APC (Figure 3-12)

Input: 60 pitch-removed baseband residual samples
Coded and quantized gain, tap and pitch, inverse of quantized gain.
Quantization table for baseband
8 coded reflection coefficients

Output (to TSINK): Coded energy, pitch and tap,
8 coded reflection coefficients.
60 coded baseband residual samples

Method: Perform APC on baseband, removing pitch redundancies and performing quantization within the APC loop.

3.2.1.3.6 Error-protection and Bitstreaming (PROTECT)

The PROTECT module takes as input a TSINK buffer containing quantized and coded analysis parameters and baseband residual samples and produces as output a TBITS buffer, in which:

(a) certain high-order data bits have been grouped together and protected with (7,4) Hamming codewords;

(b) the data to be transmitted has been "bitstreamed", one bit per half-word, in the form for use by the TMODEM scroll program;

(c) histogram information of the coded analysis parameters has been recorded.

The format of the TSINK buffer is shown below, along with the number of bits per parameter and the number of high-order bits protected by the Hamming code (K1-K8 denote the 8 reflection coefficients).

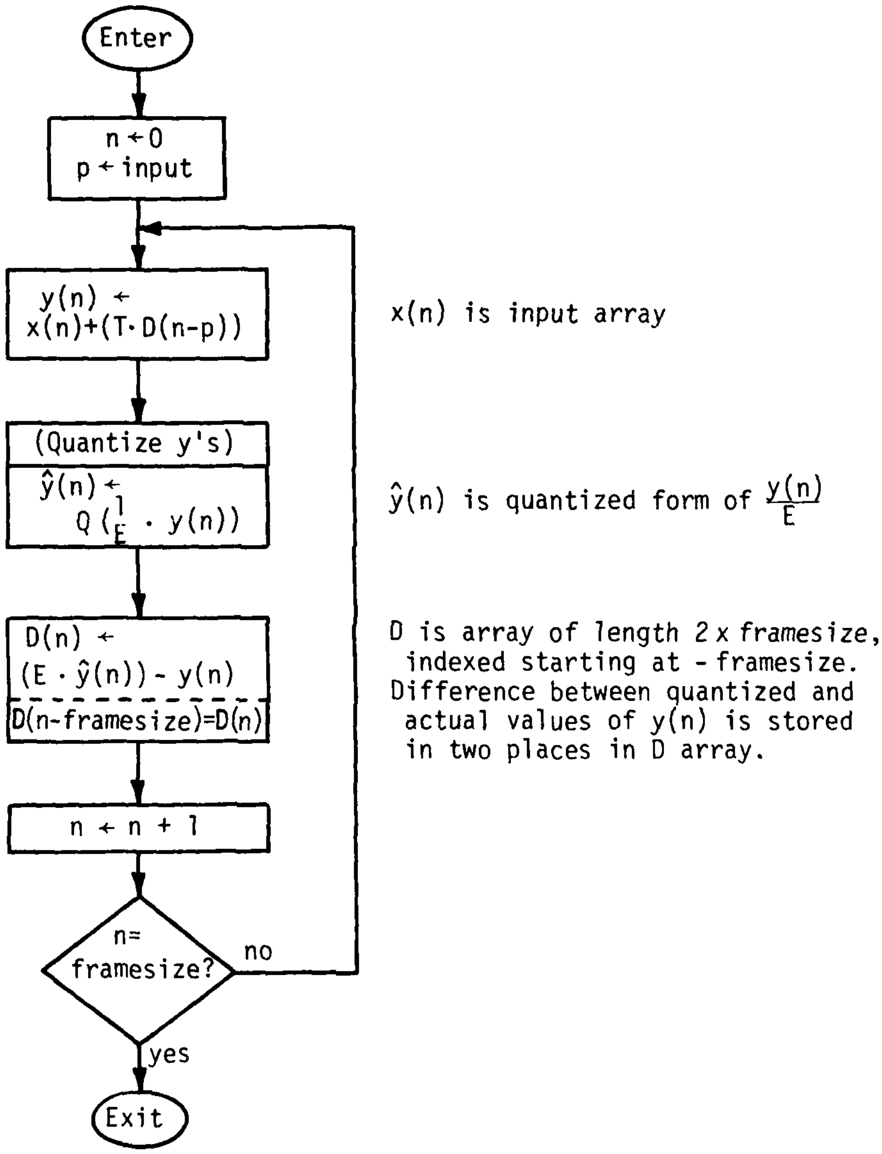


Figure 3-12. APC Module

Word	Parameter	No. of bits	Bits protected
TSINK+0	PITCH	6	5
1	TAP	4	4
2	GAIN	6	5
3-5	K1-K3	5	5
6-8	K4-K6	4	3
9-10	K7-K8	3	1
11-71	BB1-BB60	3	0

A total of 40 high-order bits of the transmitted parameters are protected using ten Hamming codewords. The format of the TBITS buffer is described in detail in the program listing of PROTECT in Appendix E. There are 259 speech coder data bits and one sync bit, leaving one bit unused in the 261-bit frame. Although two bits could have been used for frame synchronization, the simplicity and effectiveness of the single-bit scheme described in Sections 3.2.1.4 and 3.2.2.2.1 was felt to be preferable to a more complex implementation.

A histogram-gathering function was included in the PROTECT module for the purpose of gathering statistics on the effectiveness of the quantization tables for the analysis parameters. The PDP-11 host program can be modified to include code to (1) initialize the histogram buffers defined on Bus 1 and (2) upon command, stop the speech coder, transfer the histogram data from the MAP to the host, and display it.

3.2.1.3.7 TBITS Flag Control Module

After the coded, protected parameters have been written into the operational TBITS buffer, the corresponding flag is set to full.

3.2.1.4 TMODEM Interrupt Service (TMODEMINT)

The TMODEM Interrupt Service routine is shown in Fig. 3-13. When activated by a Line 1 interrupt from the modem IOS-2 scroll, this routine updates the TMODEM pointer offset to point to the current (just emptied) TMODEM buffer, then checks the current TBITS buffer flag to see if there is new bitstream data ready to be transmitted. If the flag is set to full, the data is copied from the TBITS buffer to the current TMODEM buffer, the TBITS flag is set to empty, and the TBITS pointer offset is updated to point to the other TBITS buffer/flag. If, on the other hand, the current TBITS buffer flag indicates not-full, bitstream data corresponding to a frame of silence is copied (from buffer TBTC) to the current TMODEM buffer, the "fake frame" is counted by incrementing TMFFC, and the routine exits without having changed the TBITS pointer offset.

In either case, only 259 words (bits) of data are copied to the TMODEM buffer. The data bit of the first word of each TMODEM buffer is not copied, as it is the synchronization bit: a 0 in

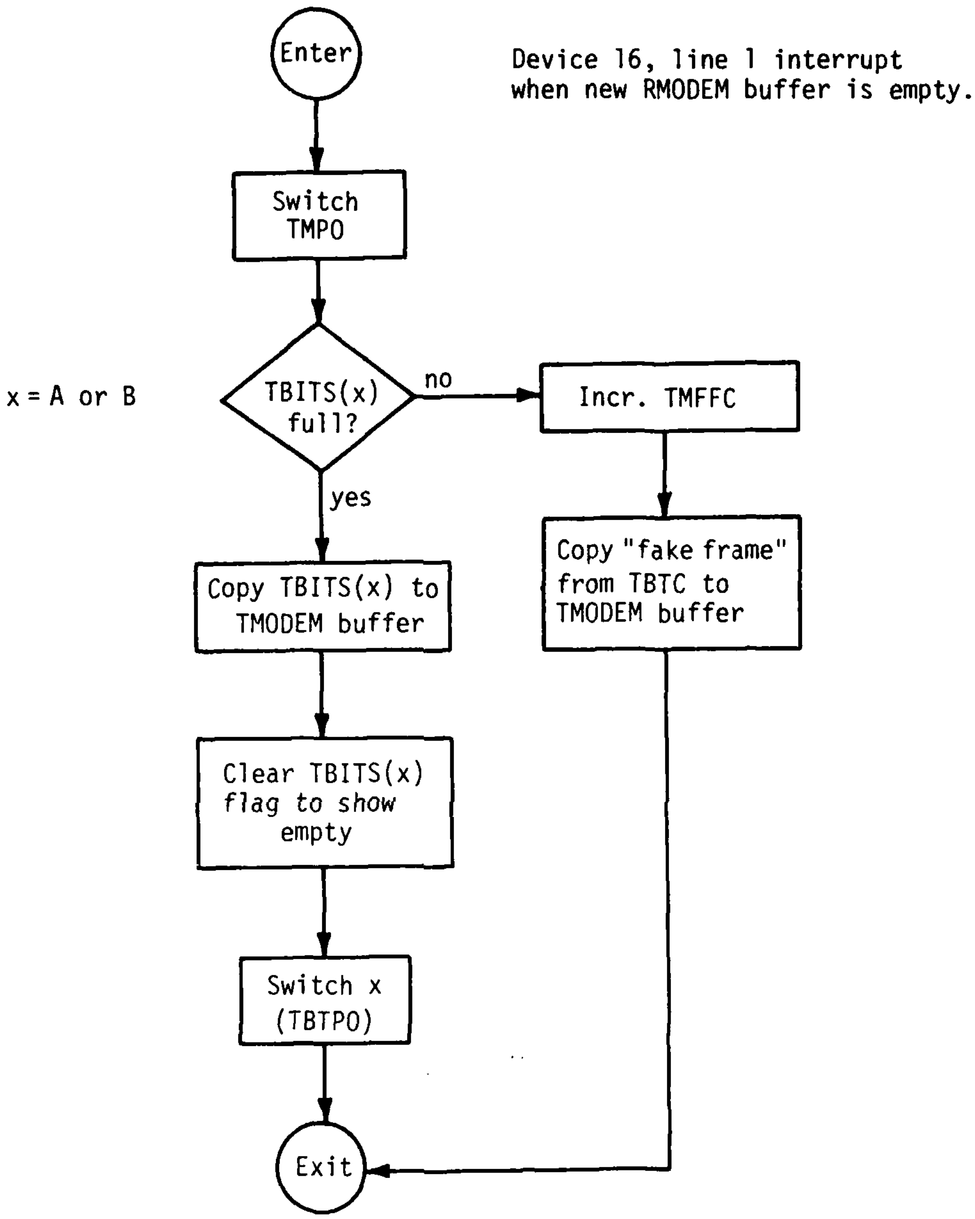


Figure 3-13. TMODEM Interrupt Service Routine (TMODEMINT)

TMODEMA and a 1 in TMODEMB. Since data is transmitted alternately from TMODEMA and TMODEMB, the data frames at the receiver have the sync bit alternating between 0 and 1 in successive frames. The last word of each buffer is unused and contains 0.

3.2.1.5 TMODEM and RMODEM I/O Scroll Program (RTPROG)

The digital data input and output is performed by an IOS-2SM Scroll processor that has been augmented by a modem interface. This interface also contains the dual-rate clock for timing both the modem data and the speech sample input and output.

The TMODEM and RMODEM Scroll program (RTPROG) is shown in Fig. 3-14. Although the TMODEM output and RMODEM input processes are logically distinct, they are performed by this single program. After setting the rates of the modem-data and speech-sample clocks and initializing buffer switches and address pointers, the program enters a basic loop that checks the two peripheral flags for data ready from the modem receiver interface (P1 set) or for the modem transmitter interface ready to accept data (P2 set).

When a new datum is available from the interface, it is stored in RMODEM buffer A, B, or C, depending on the value of a buffer switch in register R0 and an address pointer in register R1. The transfer clears flag P1. If the input buffer is now full, the buffer switch and address pointer are changed to point to the next buffer and the CSPU is interrupted on line 2.

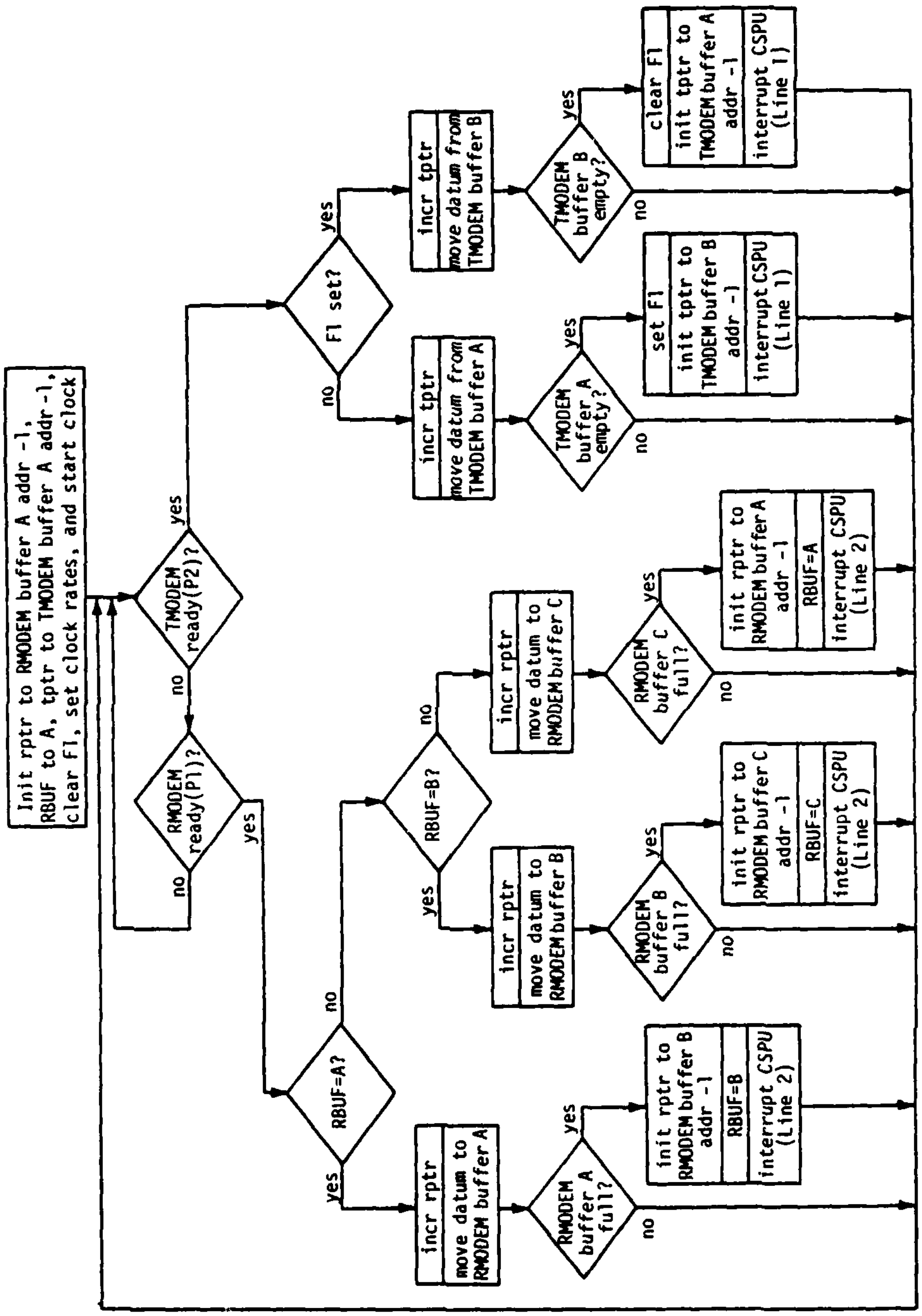


Figure 3-14. TMODEM and RMODEM Scroll Program Flow Chart

When the interface becomes ready to accept a datum (to be transmitted), a word is transferred from TMODEM buffer A or B, depending on the value of flag F1, used as a buffer switch, and an address pointer in register R2. The transfer clears flag P2. If the output buffer is now empty, the buffer switch and address pointer are changed to indicate the other TMODEM buffer and the CSPU is interrupted on line 1.

3.2.1.6 Transmitter Input/Output Buffer Initial Sequence

The several layers of shared input/output buffers in the speech coder transmitter and receiver require that the modules that reference them be initialized to do so in the proper order. The start-up sequence for the transmitter is described below.

The ADAM scroll program starts by writing speech samples into buffer TADBA (and then proceeds to write into TADBB, etc.). At the first ADAM interrupt, ADAMINT copies from TADBA to TSRA (TSOURCE A buffer) and sets the flag TSRFA The background process executes ANLZA first, since TSRFA=1 and TBTFB is initially 0. The execution of ANLZA includes PROTCT(B), which writes the first bitstream data into buffer TBTB and then sets the flag TBTFB to 1. The TMODEMINT interrupt routine is initialized to expect its first bitstream data in buffer TBTB, so that is the first data copied into a TMODEM buffer for output by the TMODEM scroll program.

Meanwhile, the TMODEM scroll program has started by transmitting (initialized) data from buffer TMDMA. It then sends an interrupt and proceeds to read from TMDMB, and so forth. The initial TMODEM interrupt activates TMODEMINT, whose first task is to provide new data for the just-emptied TMDMA buffer. TMODEMINT is initialized to check the flag TBTFB, as described above. At the end of the first frame, however, ANLZA has not yet executed and there is as yet no data in TBTB, so TMODEMINT copies "dummy" bitstream data instead from TBTC into TMDMA and exits. The next time TMODEMINT is activated, it again checks TBTFB and (since ANLZA has run by now) finds it set, so it copies the new bitstream data from TBTB to the (just-emptied) TMDMB buffer. Thereafter all data buffers are alternately filled and emptied without conflict.

3.2.2 Receiver

The Receiver is similar in structure to the Transmitter. The Receiver is shown in Fig. 3-15.

Data is accepted from the modem and put into one of three RMODEM buffers by the RMODEM program running in an I/O scroll. When this program fills a buffer, it generates an interrupt to the CSPU and begins filling the next buffer (A,B,C,... in rotation).

This interrupt activates the RMODEM Interrupt Service routine, which checks frame synchronization and, if correct, transfers the

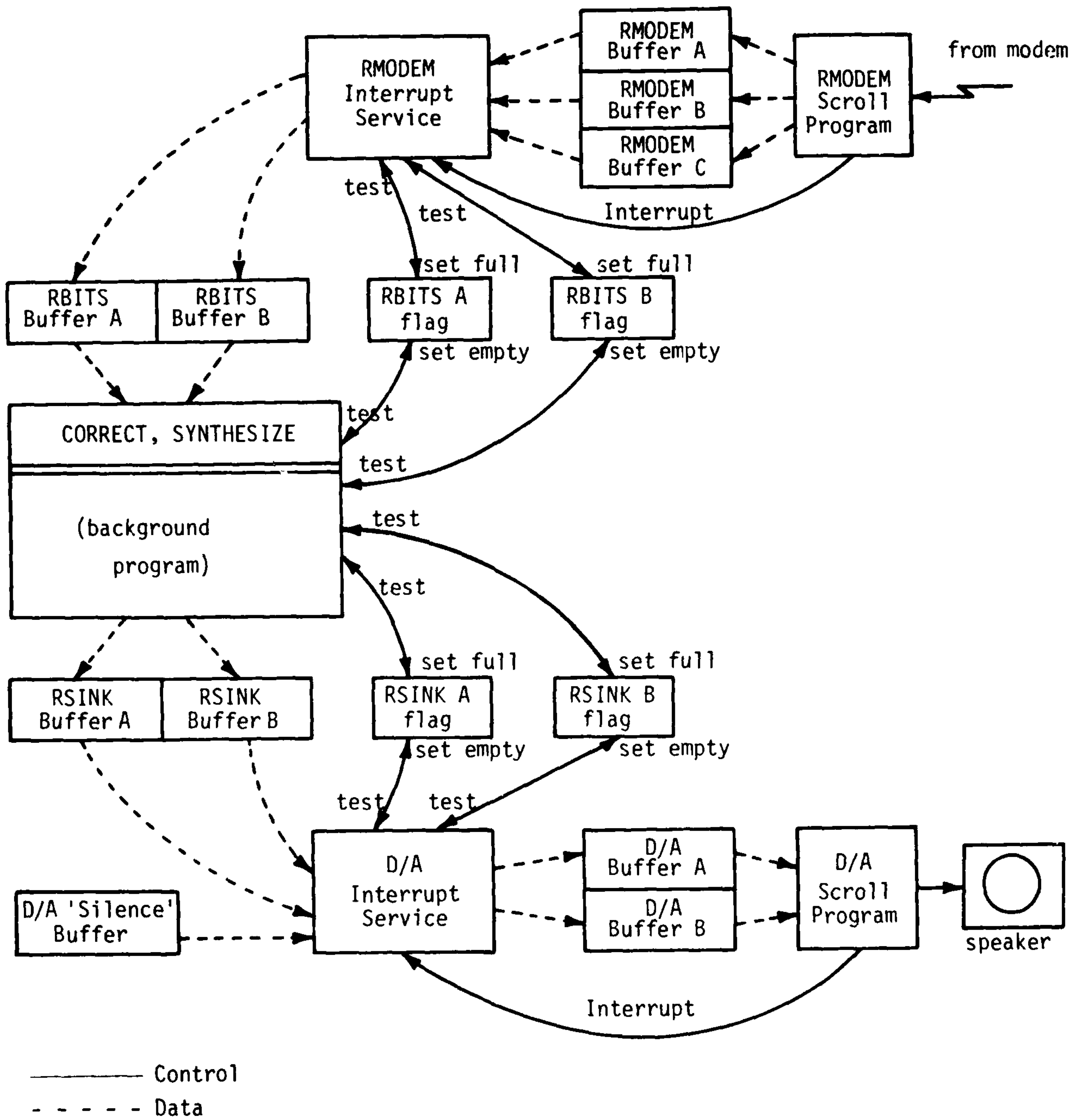


Figure 3-15. Receiver Process

new data to an empty RBITS buffer and sets the corresponding RBITS flag to indicate full. The data that is actually transferred does not correspond exactly to the just-filled RMODEM buffer. Rather, it is a "frame" of data completely contained in the new RMODEM buffer and the RMODEM buffer previously filled. A frame of data begins with a sync bit. Since this bit may occur anywhere within a single RMODEM buffer, two full buffers (and therefore three buffers all together) are required to guarantee that a complete frame is available.

The CORRECT module, running in the CSPU at background level, empties the full RBITS buffer, performs error correction and decoding of the incoming bitstream, copies these decoded parameters into an (empty) RSOURCE buffer, and clears the RBITS flag to signify empty.

The SYNTHESIZE module, running in the CSPU at background level, empties the (full) RSOURCE buffer, performs the processing necessary to synthesize speech, and puts the output speech samples into an (empty) RSINK buffer, setting the corresponding RSINK flag to indicate full. The SYNTHESIZE module, although logically following the CORRECT module, in fact contains CORRECT and is executed concurrently with it. That is, SYNTHESIZE operates on the data supplied by the previous execution of CORRECT. This means that SYNTHESIZE will execute the first time before any meaningful data has been made available to it by CORRECT. This problem is circumvented by initializing RSOURCE to contain data that will synthesize silence.

The data from the RSINK buffer is transferred to an empty D/A buffer by the D/A Interrupt Service routine. This routine also sets the corresponding RSINK flag to indicate empty.

This Interrupt Service routine is activated by an interrupt from the D/A Scroll program, running in another I/O scroll (the AOM, or Analog Output Module). This scroll program transfers data from a D/A buffer to the D/A converter, interrupts the CSPU when the buffer is empty, and begins transferring data from the other D/A buffer.

3.2.2.1 RMODEM Scroll Program

The RMODEM scroll program is described in section 3.2.1.5.

3.2.2.2 RMODEM Interrupt Service (RMODEMINT)

The RMODEM Interrupt Service routine is shown in Fig. 3-16. It is activated by a line 2 interrupt from the modem IOS-2 scroll, which indicates that the next RMODEM buffer is full. The routine updates the RMODEM pointer offset to point to the just-filled buffer. Then it copies the on-hook bit from the first word in the current RMODEM buffer to the integer scalar RONHK, where it is

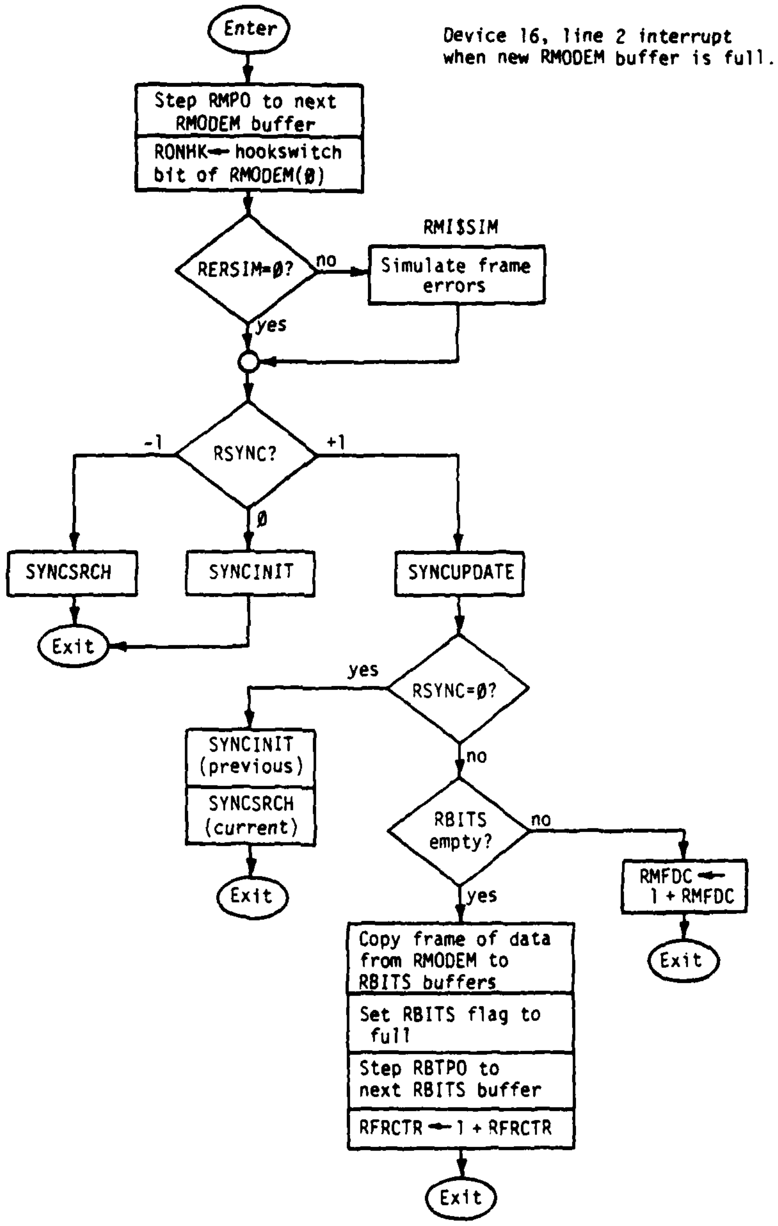


Figure 3-16. RMODEM Interrupt Service Module (RMODEMINT).

saved for use by the ADAM interrupt service routine (Section 3.2.1.2).

Then the integer scalar RERSIM is tested. This is normally zero, but if it is set to nonzero (by a command from the host program), a channel error simulation routine (RMI$SIM) is called. RMI$SIM uses the value in RERSIM as the number of errors to be introduced per frame. Since there are 261 bits per frame, each error contributes about 0.4% to the error rate. This error simulation is not intended to be an accurate simulation of an errorful channel.

The next set of operations deals with frame synchronization. Since the receiver has no prior knowledge about the position of the frame boundaries in the input data stream, it must infer the boundary from the received data pattern. The first bit of each transmitted frame is the sync bit, which alternates between 0 and 1 in successive frames; this pattern allows the receiver to detect and maintain frame synchronization. The synchronization routines are described in section 3.2.2.2.1 below.

RMODEMINT tests the value of RSYNC, an integer scalar that maintains the receiver's sync-state. If RSYNC=0 (as it is when the speech coder is initialized), nothing is known about sync, so the SYNCINIT routine is called to initialize the sync-search process with the current RMODEM buffer. If RSYNC<0, then the receiver is still searching for sync, so the SYNCSRCH routine is called to continue the search using the new data in the current RMODEM buffer. In both of these cases, RMODEMINT exits immediately. If RSYNC>0, then the receiver has gained frame-sync, so the SYNCUPDATE routine is called to check that the current RMODEM buffer continues to show the expected sync-bit value. Upon the return from SYNCUPDATE, RSYNC is checked again; if it has been set to zero, then the receiver has lost sync and must again search the incoming bitstream for the sync-bit pattern; therefore SYNCINIT and SYNCSRCH are called to restart sync-searching with the previous and current RMODEM buffers.

If RSYNC is still greater than zero after SYNCUPDATE, then synchronization is confirmed, and the receiver makes use of the data for speech output. RMODEMINT checks the current RBITS buffer flag; if empty, the most recent frame of data is copied from the RMODEM buffers to the current RBITS flag, the RBITS flag is set to full, and the RBITS pointer offset (RBTPO) is switched to the next RBITS buffer. Note that in general, the current frame will straddle the previous and current RMODEM buffers. It is for this reason that there are three RMODEM buffers, two to hold the current frame while the third is being filled by the RMODEM scroll program.

If, for some reason, the RBITS buffer is not empty (ready to accept new data), the new data frame is effectively discarded. A frame discard counter (RMFDC) is incremented to take note of this, and the routine exits without updating RBTPO.

3.2.2.2.1 Synchronization Routines

The SYNCINIT subroutine (Fig. 3-17) initializes the state of the frame synchronization section of the speech coder. Its primary function is to copy the data bit of each word of the current RMODEM buffer into RSSPF, the "previous frame" memory used in SYNCSRCH, and to clear the counts in the RSSSS buffer. SYNCINIT also zeros the gain word of both RSOURCE buffers and the entire previous RBITS buffer, so that when the speech coder receiver resumes operation after regaining sync, the incorrect information in these buffers will not produce unwanted transients in the speech output.

The SYNCSRCH subroutine (Fig. 3-18) scans the incoming RMODEM buffers (one per call), building up statistics on the frame-to-frame data patterns in each bit position of the frame until it detects a large enough number (ACQTHR) of consecutive bit alternations in a single bit position to allow it to identify that position as carrying as the sync bit. When that occurs, it sets the value of RBOFO ("beginning-of-frame-offset") to the distance from the start of the buffer to the sync-bit position and RSYNC to

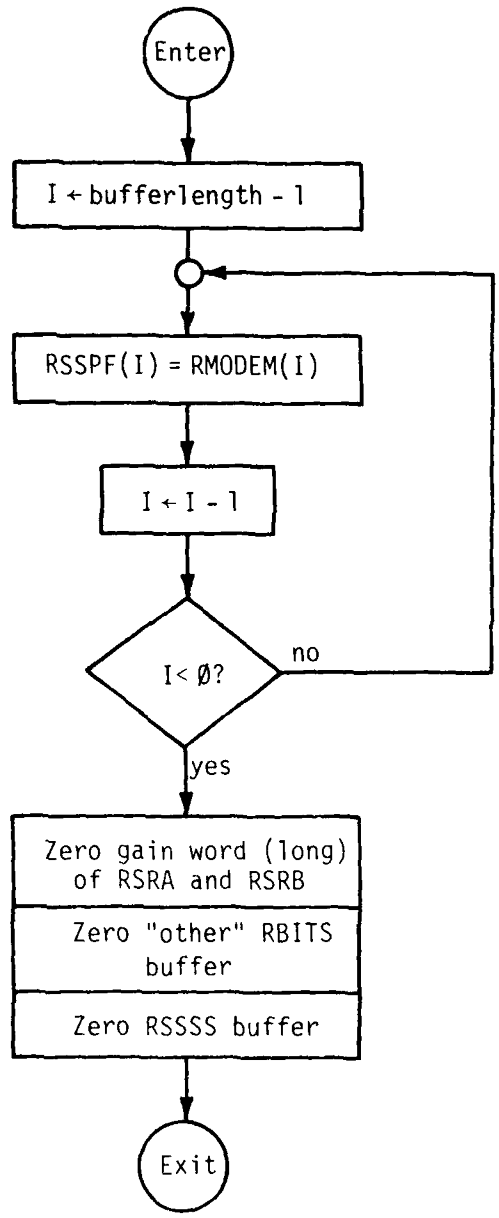


Figure 3-17. Frame Synchronization Initialization Module (SYNCINIT)

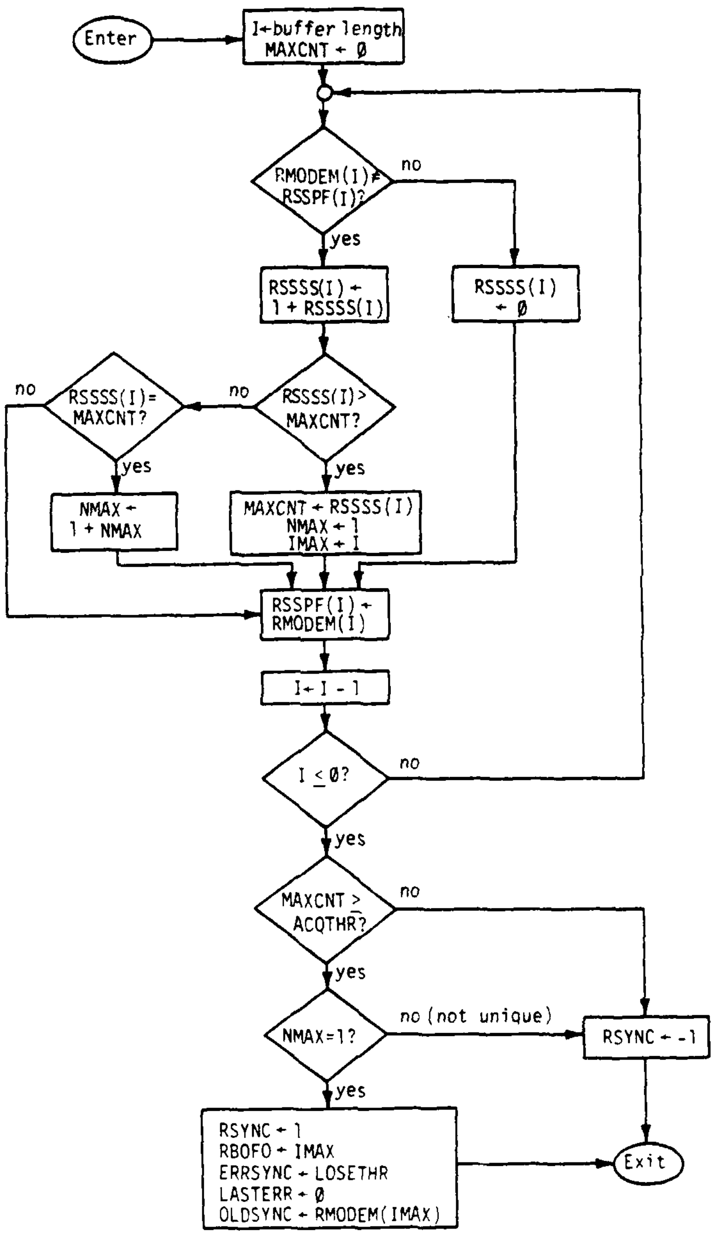


Figure 3-18. Frame Synchronization Search Module (SYNCSRCH).

+1 to cause RMODEMINT to start passing input frames to the speech coder synthesizer on the next RMODEM interrupt. SYNCSRCH also initializes three variables that are used in subsequent calls to SYNCUPDATE.

The SYNCUPDATE subroutine (Fig. 3-19) is used once sync has been acquired to check the sync-bit position of each incoming RMODEM buffer to see if it has the expected value. It is the function of SYNCUPDATE to declare sync lost if the declared sync bit does not show the expected frame-to-frame alternation. Since channel errors may cause the sync bit to be incorrect, the loss-of-sync criterion must be more stringent than a single incorrect sync bit. The criterion is a sufficient number (LOSETHR) of incorrect sync bits without two consecutive correct sync bits. A simpler criterion, such as LOSETHR consecutive sync bit errors, would be unsatisfactory since the high-order bit of many speech coder parameters (e.g., energy) may have the same value for many consecutive frames, and a constant bit would compare "correctly" with an alternating bit sequence every other frame.

The performance of the synchronization routines is controlled by two thresholds, ACQTHR and LOSETHR. It is necessary to set ACQTHR high enough so that it is very unlikely that SYNCSRCH will detect sync on the wrong bit position. The value of LOSETHR must be high enough that SYNCUPDATE is unlikely to declare loss-of-sync

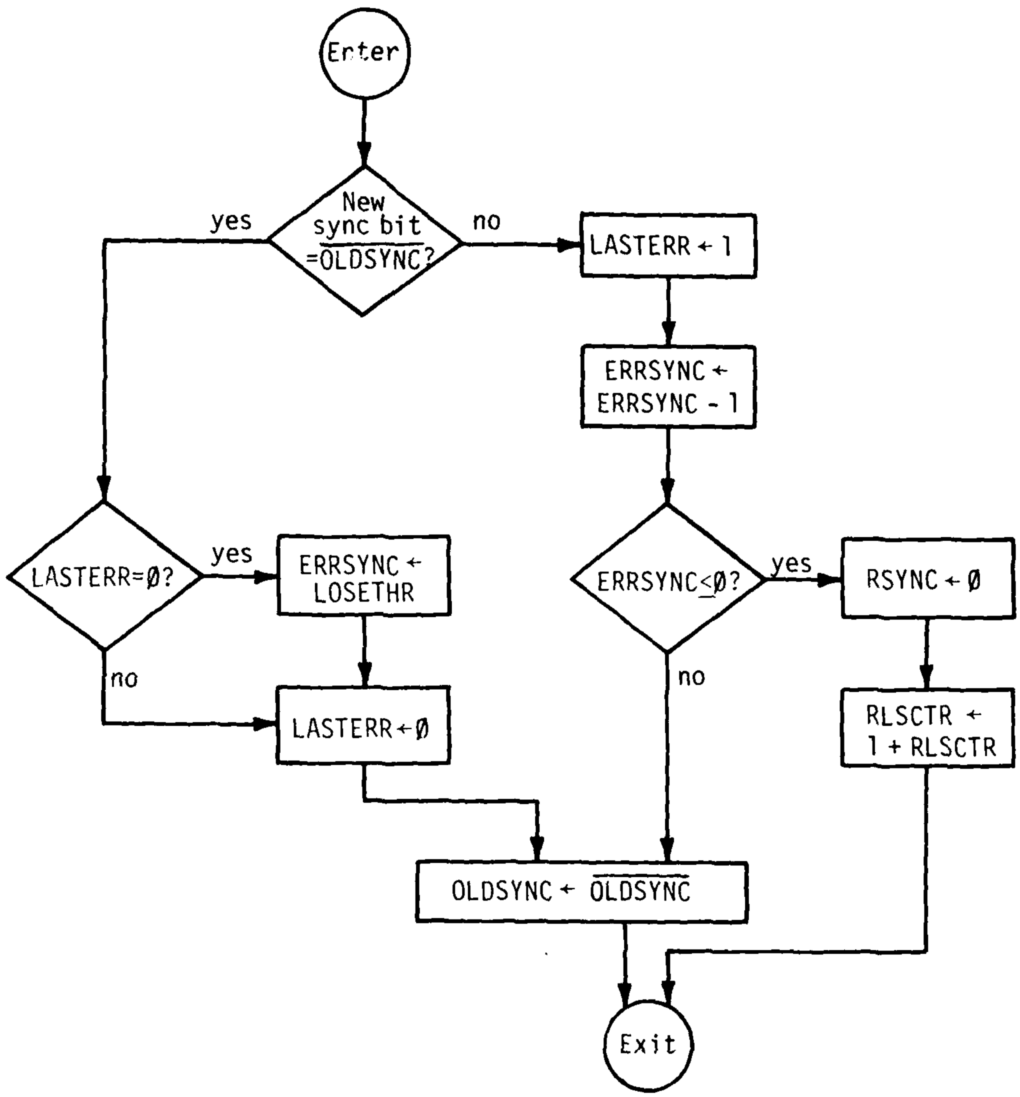


Figure 3-19. Frame Synchronization Update Module (SYNCUPDATE)

due to channel errors falling on the sync bit, but small enough that if sync should be lost for some reason, the receiver does not synthesize too much "garbage" before detecting the loss-of-sync. Of course, while the receiver is searching for sync, RMODEMINT does not pass data to the synthesizer, so silence is output. For a 1% channel bit error rate, the values of ACQTHR=10 and LOSETHR=4 give a probability of acquiring false sync of about 0.0002 and an expected time to spontaneous loss-of-sync of about 10 hours, respectively.

3.2.2.3 SYNTHESIZE Module

The SYNTHESIZE module is shown in Fig. 3-20.

The Synthesize Module contains three processing modules and two control functions.

3.2.2.3.1 Unbitstreaming and Error Correction (CORRECT)

The CORRECT module takes as input an RBITS buffer containing a frame of bitstream data input from the modem and produces as output an RSOURCE buffer containing error-corrected and decoded floating point values of analysis/synthesis parameters and baseband residual samples, ready for immediate use by the speech coder synthesizer.

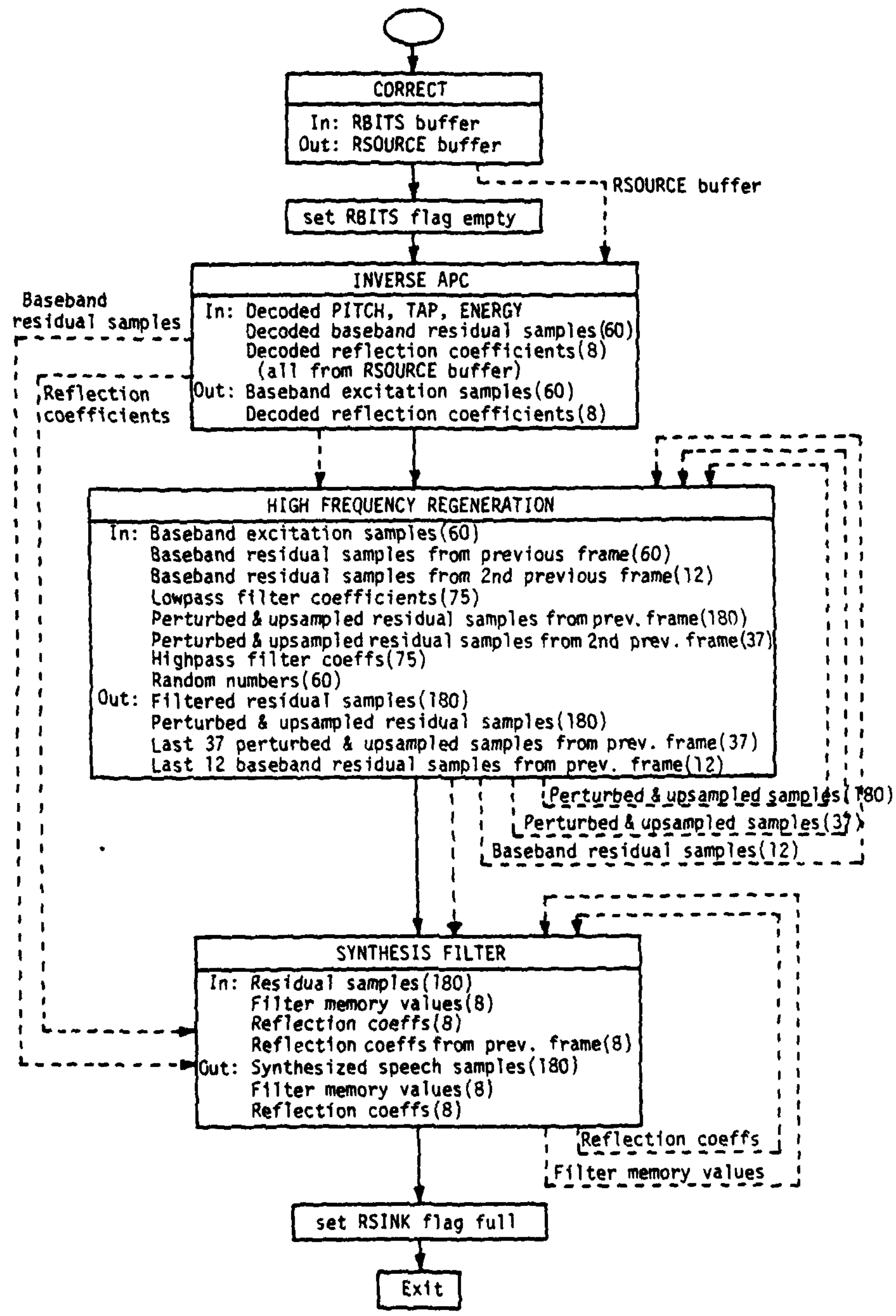


Figure 3-20. SYNTHESIZE Flow Chart

The format of the RBITS buffer is the same as the TBITS buffer, as described under PROTECT in the program listing in Appendix E. The data bits are regrouped, and the 7-bit codewords are error-corrected by lookup in an inverse Hamming code table. The bits are then grouped into coded parameter values and decoded by lookup in parameter-specific decoding tables of floating-point values. The format of the RSOURCE buffer is similar to the TSINK buffer, except that the half-word coded TSINK values are replaced by full-word (32-bit) floating point values.

3.2.2.3.2 Inverse APC Module (Figure 3-21)

Input: Decoded pitch
Decoded tap
Decoded gain
Decoded baseband residual samples (60)
Decoded reflection coefficients (8)
(All in one buffer)

Output: Baseband residual samples (60)
Decoded reflection coefficients (8)

Method: First separate parameters into buffers and scalars. Then multiply samples by Gain, and use sample delay loop of Pitch length to restore pitch period redundancies to residual samples. The reflection coefficients are simply copied out of the RSOURCE buffer in order to free it.

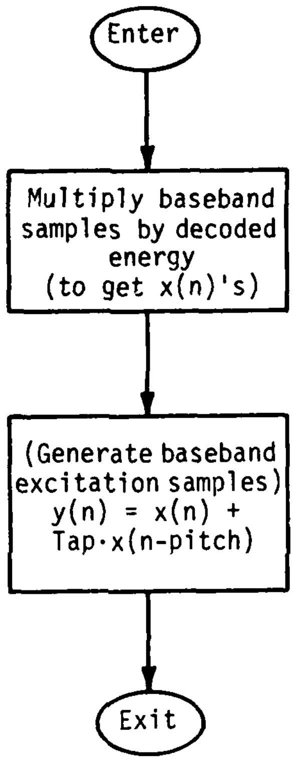


Figure 3-21. Inverse APC Module

3.2.2.3.3 RBITS Flag Control

This function sets the appropriate RBITS flag to empty.

3.2.2.3.4 High Frequency Regeneration Module (Figure 3-22)

Input:
- Baseband residual samples (60)
- Baseband residual samples from previous frame (high-passed)(60)
- Last 12 baseband residual samples from 2nd previous frame (high-passed)
- Low-pass filter coefficients (75)
- Perturbed and upsampled residual samples from previous frame (180)
- Last 37 perturbed and upsampled residual samples from 2nd previous frame
- High-pass filter coefficients (75)
- Array of random numbers (60)

Output:
- 180 filtered residual samples
- 180 perturbed & upsampled residual samples
- Last 37 upsampled & perturbed residual samples from previous frame
- 60 baseband residual samples (high-passed)
- Last 12 baseband residual samples from previous frame

Method:
- Use a second-order Butterworth high-pass filter to remove very low frequencies from baseband residual.
- Low-pass filter (upsampled) baseband residual.
- Perturb upsampled residual, high-pass filter, and add filter outputs. Due to the delay of the high- and low-pass filters, the filter output corresponds to the input from the previous frame.

This module consists of the following sub-modules:

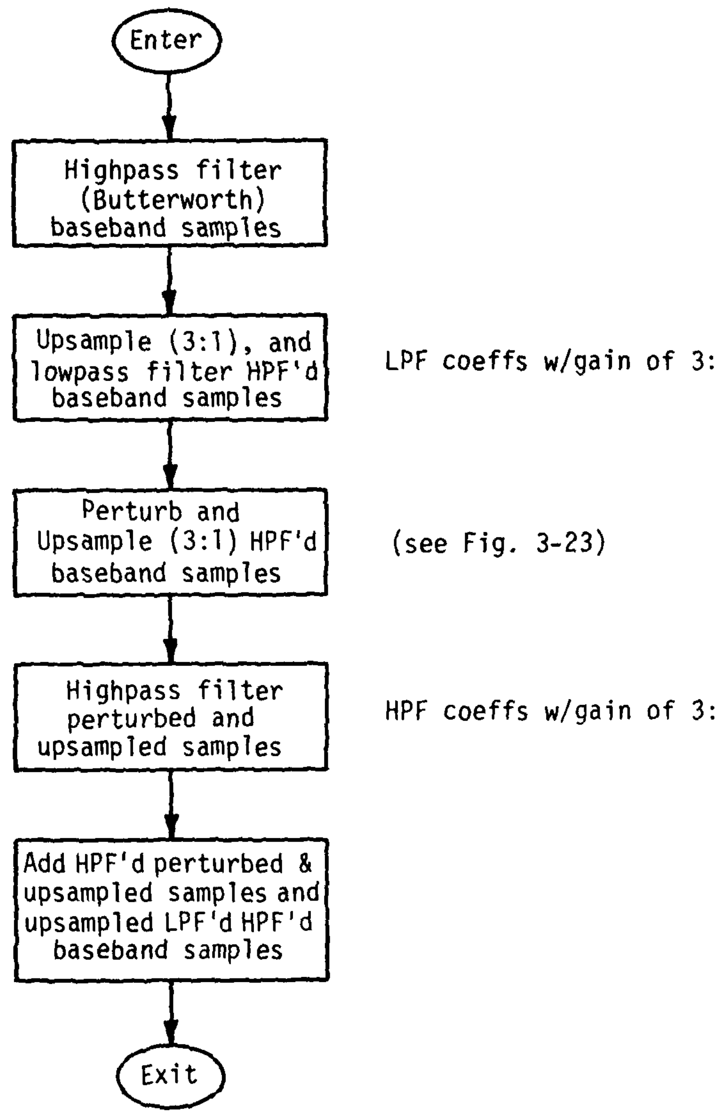


Figure 3-22. High Frequency Regeneration Module

High-pass Filter Baseband Samples

Input: Baseband residual samples (60)
Filter memory (4)
Filter coefficients (4)

Output: HPF'd baseband residual samples (60)
Filter memory (4)

Method: 2 pole, 2 zero Butterworth filter

Upsample and Lowpass Filter

Input: HPF'd baseband residual samples (current frame) (60)
HPF'd baseband residual samples (prev. frame) (60)
HPF'd baseband residual samples (2nd prev. frame) (last 12)
Lowpass filter coefficients (75)

Output: LPF'd upsampled baseband residual samples (180)

Method: Upsample (3:1) and LPF, with filter centered on prev. frame
LPF coefficients are pre-multiplied by upsample factor (3).
Output corresponds to prev. frame of inputs.

Upsample and Perturb (Figure 3-23)

Input: HPF'd baseband residual samples (60)
Random number array (60)

Output: Upsampled and perturbed residual samples (180)

Method: Upsample (3:1) and exchange each input sample with either its left neighbor, right neighbor, or not at all, according to the array of random numbers.

High-pass Filter Upsampled and Perturbed Samples

Input: Upsampled and perturbed residual samples (current frame) (180)
Upsampled and perturbed residual samples (prev. frame (180)
Upsampled and perturbed residual samples (2nd prev. frame) (last 37)

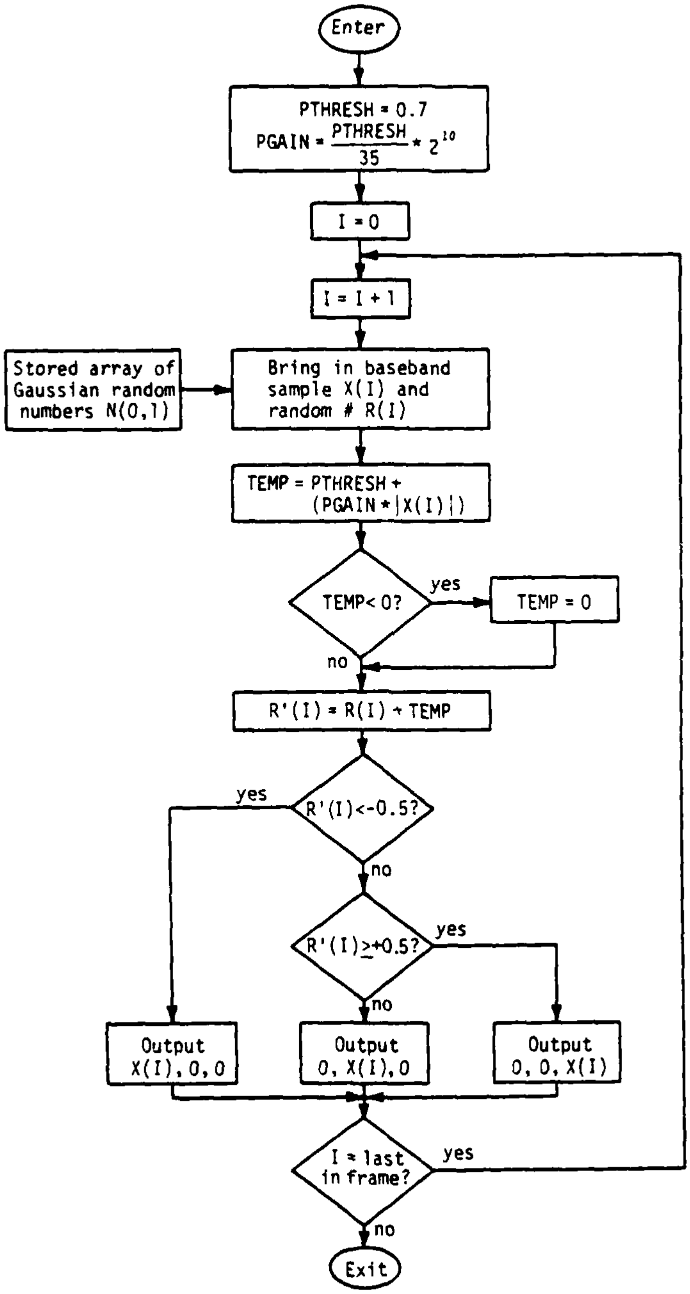


Figure 3-23. Perturbed Upsampling (1:3)

High-pass Filter Coefficients (75)

Output: HPF'd Upsampled and Perturbed Samples (180)

Method: HPF, with filter centered on prev. frame.
HPF coefficients are pre-multiplied by upsample factor (3). Output corresponds to prev. frame of inputs.

Add Low-pass and High-pass Residuals

Input: LPF'd upsampled baseband residual samples (180)
HPF'd upsampled and perturbed residual samples (180)

Output: Excitation samples (180)

3.2.2.3.5 Synthesis Filter Module (Figure 3-24)

Input: 180 excitation samples
8 filter memory values from previous execution of this module.
8 reflection coefficients
8 reflection coefficients from previous frame

Output: 180 synthesized speech samples
8 filter memory values
8 current reflection coefficients

Method: Use lattice form filter to perform all-pole filtering of excitation to obtain synthetic speech. Store synthetic speech in RSINK buffer. Delay reflection coefficients by one frame before using in filter.

3.2.2.3.6 RSINK Flag Control

When the synthetic speech samples have been sent to the current RSINK buffer, the current RSINK flag is set to indicate full.

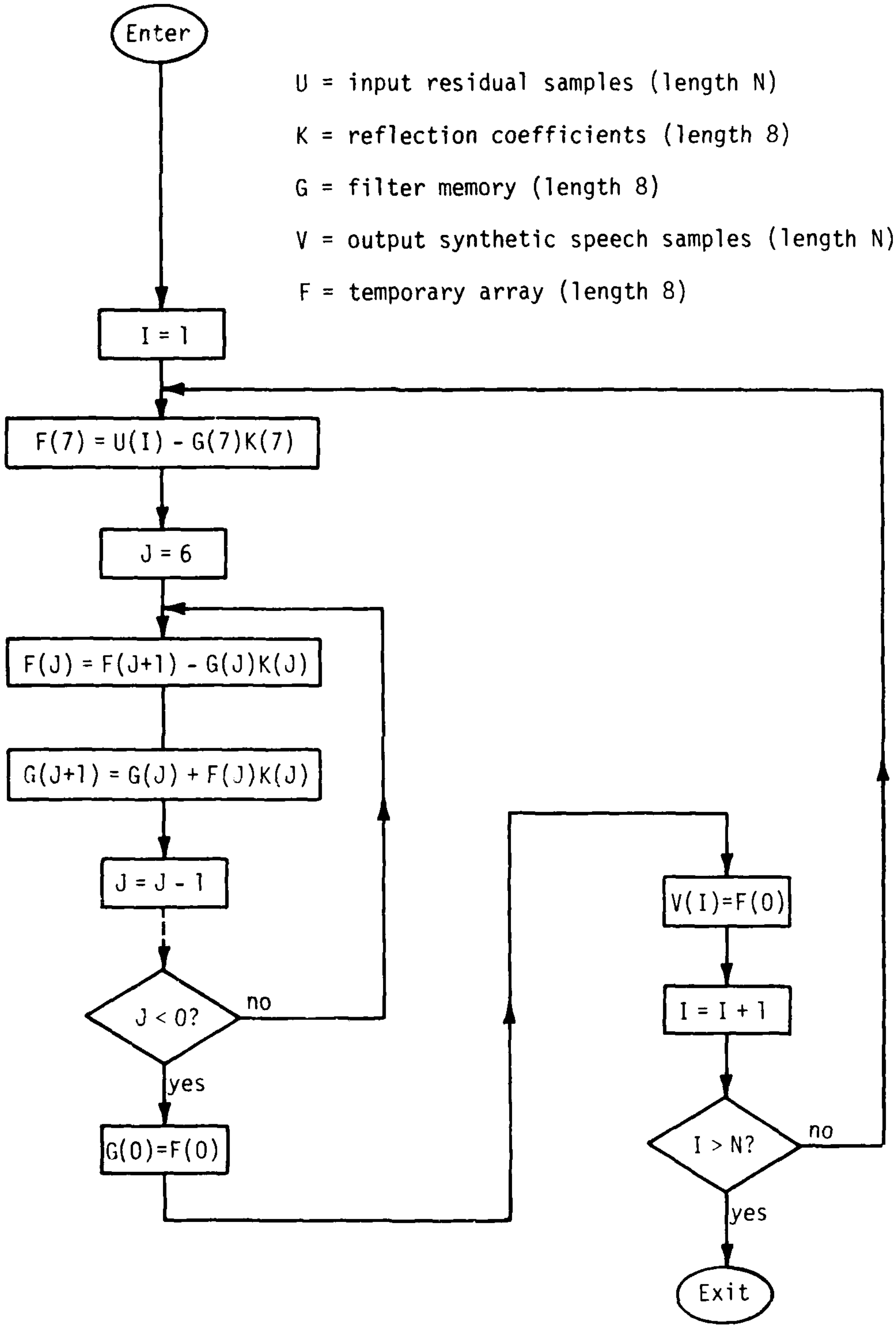


Figure 3-24. Synthesis Filter Module

3.2.2.4 D/A Interrupt Service Routine (AOMINT)

AOMINT is activated by each AOM line 1 interrupt, signifying the emptying of a D/A output buffer by the AOM scroll program. Its operation, as illustrated by Fig. 3-25, is analogous to that of the TMODEMINT routine, the only difference being the copying of synthesis output data from RSINK buffers to D/A buffers for output to the AOM. If an RSINK buffer is not available (full), a buffer of silence is output in its place. This happens whenever the speech coder is not receiving data from the remote speech coder and during frame synchronization or resynchronization.

3.2.2.5 D/A Scroll Program (DAPROG)

Speech output is performed by the AOM, an IOS-2 scroll processor that contains two D/A converters. The AOM gets its sample output clock from the SPI, and it sends the D/A output signal to the SPI for subsequent low-pass filtering and output. The signal is in the range -5 to +5 volts, and the sample rate is 6.621 kHz, the same as for the A/D input.

The AOM program (DAPROG) is illustrated in Fig. 3-26. In "single-channel" mode, the AOM outputs an internally-generated value on D/A Channel 0 while it converts samples from MAP memory on D/A Channel 1. The Channel 0 signal is a ramp that is reset each

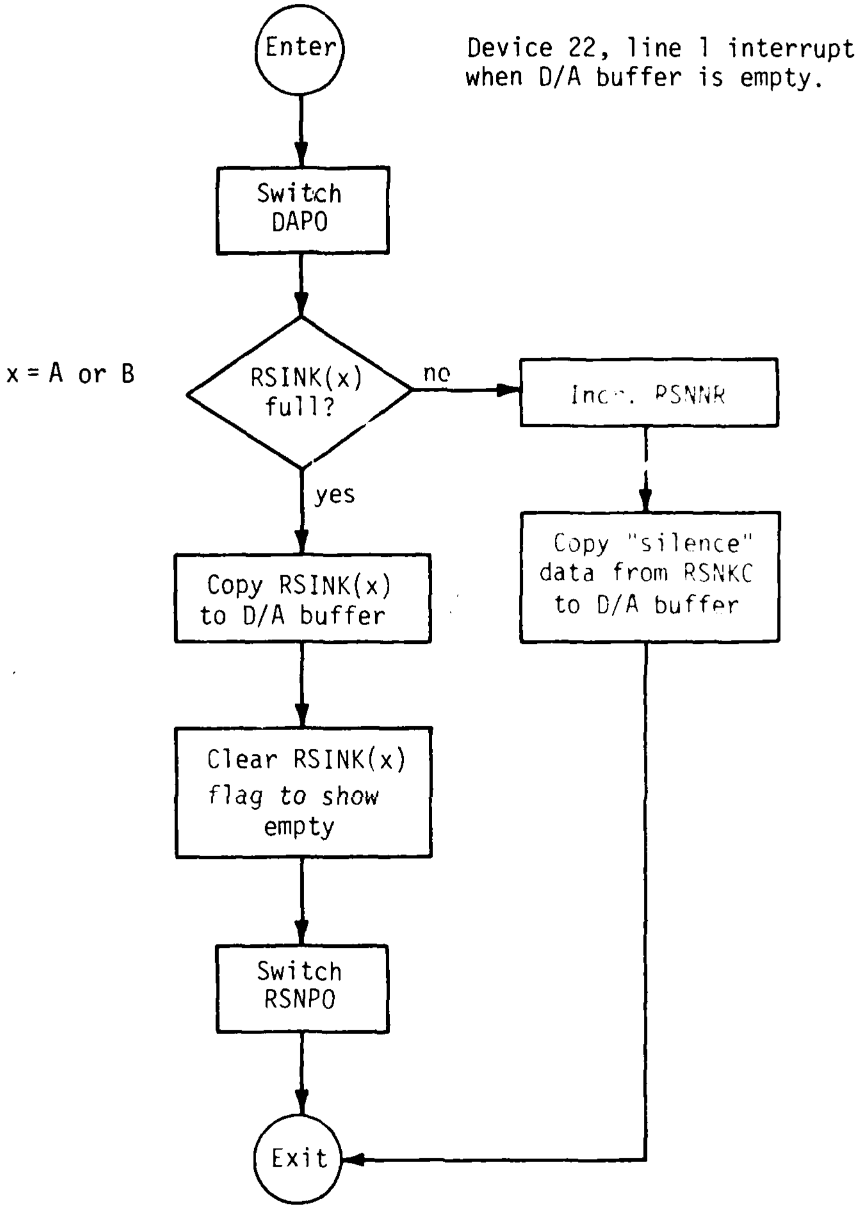


Figure 3-25. D/A Interrupt Service Routine (AOMINT)

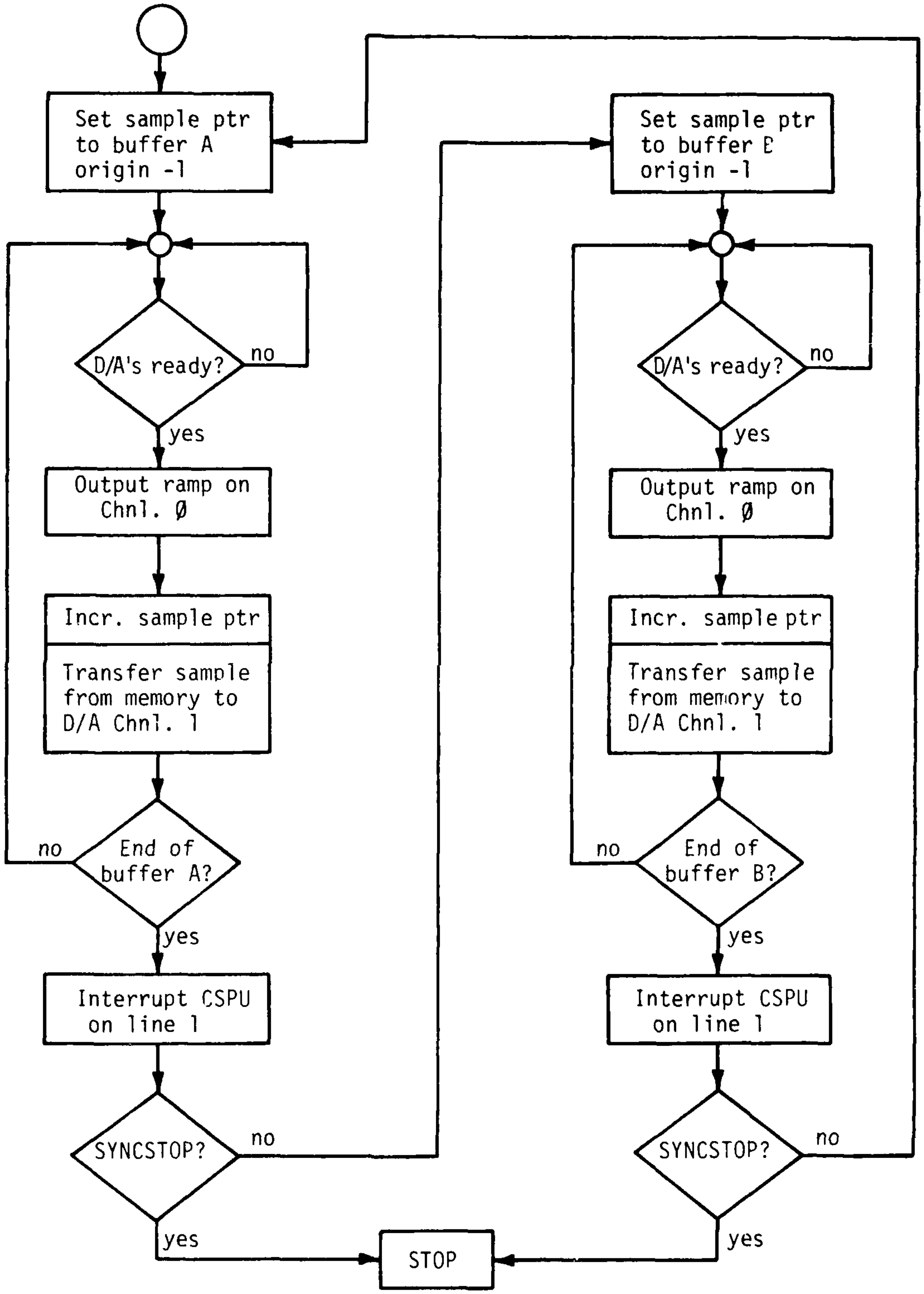


Figure 3-26. DAPROG: AOM Scroll Program

frame time; it is irrelevant to speech coder operation, but it may be used for synchronization or horizontal sweep on an external oscilloscope.

The operation of DAPROG is analogous to that of ADPROG. Output begins from buffer RDABA, and when the end of that buffer is reached, a line 1 interrupt is sent to the CSPU, and buffer RDABB is selected. Reaching the end of RDABB produces another line 1 interrupt and a switch to RDABA, and so forth.

3.2.2.6 Receiver Input/Output Buffer Initial Sequence

The buffer start-up sequence for the speech coder receiver is described below.

The RMODEM scroll program starts by writing received data words into buffer RMDMA, sending an interrupt when it is filled. Upon the initial interrupt, RMODEMINT begins by using RMDMA for frame-sync initialization. Since frame-sync has not yet been acquired, RMODEMINT does not output any data. Successive RMODEM interrupts cause RMODEMINT to use buffers RMDMB, RMDMC, RMDMA, etc. for further frame-sync searching until, after at least 12 input frames, sync has been found. Then RMODEMINT will copy a frame of bitstream data from the RMODEM buffers to buffer RBTA and set the flag RBTFA=1. The background process will execute SYNZB first, since RBTFA=1 and RSNFB is initialized to 0. SYNZB writes the first synthesized speech data in buffer RSNKB and sets the flag RSNFB=1. The AOMINT interrupt routine is initialized to expect the first synthetic speech data in buffer RSNKB, so when it is next activated, that buffer is copies into a D/A buffer for output by the AOM scroll program.

Meanwhile, the AOM scroll program has started by reading data (initialized to silence) from buffer RDABA, sending an interrupt, going on to RDABB, etc. These AOM interrupts activate AOMINT, whose task is to copy new synthesized speech data from RSINK buffers (initially RSNKB) to the just-emptied D/A buffers (initially RDABA). On its first several (usually 13) activations, AOMINT finds RSNKB empty, because the receiver has not yet gained frame-sync, so AOMINT copies a buffer of silence from buffer RSNKC to the D/A buffers instead. On the next activation after SYNZB has run, AOMINT finds RSNKB full, so it copies it to the D/A buffer, and thereafter all data buffers are alternately filled and emptied without conflict.

3.3 SYSTEM HARDWARE

The 9600 bps speech coder system is implemented on a MAP-300 array processing computer, which is manufactured by CSP Inc., of Billerica, Mass. Section 3.3.1 describes the configuration of MAP-300 equipment that is necessary for the speech coder system.

Section 3.3.2 and Appendix A describe the additional audio signal interface and IOS-2SM scroll modifications for connection to a modem that complete the system.

3.3.1 MAP-300 Hardware

The MAP-300 configuration that was specified by Defense Communications Agency for the implementation of the speech coder system is listed below.

1	1030	MAP-300 Processor
1	2030	8K x 32 MOS Master Memory, 500 nsec, Bus 1
1	2050	16K x 32 MOS Slave Memory, 500 nsec, Bus 1
1	2203	8K x 32 MOS Master Memory, 300 nsec, Bus 2
1	2020	2K x 32 Bipolar Memory, Bus 3
1	3110	PDP-11 Interface
1	4020	Model 2SM I/O Scroll
2	4040	Bus Switch (for Model 2SM I/O Scroll)
1	5120	Analog Data Acquisition Module
1	5130	Analog Output Module
1	6100	Expansion Chassis
1	6200	Auxiliary Power Supply

The Bus 1 memory was originally specified at 8K words, but subsequent familiarization with the CSPI software led to a strong recommendation by all three 9600 bps speech coder contractors that additional Bus 1 memory be added to accommodate:

1. New releases of CSPI software that require more than the original 8K.

2. New special-purpose software written by each contractor for the implementation of their speech coder systems.

3. "Prebound functions", which are address-processor modules in which the arguments are bound before execute-time for greater efficiency.

4. Data buffers.

The Bus 3 bipolar memory was intended for use for data buffers in the most time-critical portions of the speech coder system. Although it was specified at 125 ns by the manufacturer, it actually ran much slower. Only toward the end of the contract was the manufacturer able to make it run as fast as 180 ns. The effect of this speed discrepancy is to make our speech coder algorithms run slower than originally designed; this required an additional reprogramming effort to produce a real-time system.

3.3.2 Audio and Modem Interface Hardware

In addition to the MAP-300 equipment listed above, two items were needed to enable the MAP-300 to function as a complete, stand-alone speech coder system: an audio signal interface and a digital data interface to a modem. These two items were designed and built by the GTE Sylvania Electronic Systems Group in Needham, Mass., which is also one of the three 9600 bps speech coder contractors. Identical interfaces were provided for all three MAP-300 systems so that they would be interchangeable at the hardware level.

The audio signal interface consists of a handset, tape input and output jacks, and circuitry for the amplification, equalization, and filtering necessary for interfacing speech input and output signals to the MAP-300 A/D and D/A converters.

The modem interface consists of modifications to the MAP-300 IOS-2SM scroll processor for the purpose of transferring data from MAP memory to the modem and also data from the modem into MAP memory. Two real-time clocks derived from a single master oscillator are also provided for controlling the modem data rate and the speech sampling rates.

The audio and modem interfaces are described in more detail in the Equipment Description included as Appendix A.

3.4 SYSTEM SOFTWARE

The speech coder software consists of two distinct sets of modules. The first set is made up of modules that run in the MAP-300, including CSPI-supplied as well as BBN-written programs. Section 3.4.1 describes the MAP-300 modules that must be loaded into the MAP-300 processor before the speech coder can be started.

The second set of software modules contains routines that run in the PDP-11. These routines make use of the CSPI-supplied SNAP-II software system to initialize and start the MAP-300 speech coder programs. Section 3.4.2 describes these FORTRAN routines.

3.4.1 MAP-300 Software Components

All MAP-300 processing is done in conjunction with Release 3.5 of the CSPI-supplied SNAP-II software system. For the most part, BBN-written MAP-300 programs take the form of new AP or CSPU functions, callable via the standard SNAP-II calling procedure. In addition, BBN-written interrupt service routines and input/output scroll programs have been added to the executive.

CSPI-supplied MAP-300 software necessary for speech coder operation is described in section 3.4.1.1. MAP-300 speech coder software written by BBN is described in Section 3.4.1.2.

Figure 3-27 shows the MAP-300 memory organization, including the location of CSPI and BBN software, as well as the location of buffers defined in the speech coder system.

3.4.1.1 CSPI-Supplied MAP-300 Software

The following CSPI-supplied MAP-300 software modules are required for speech coder operation:

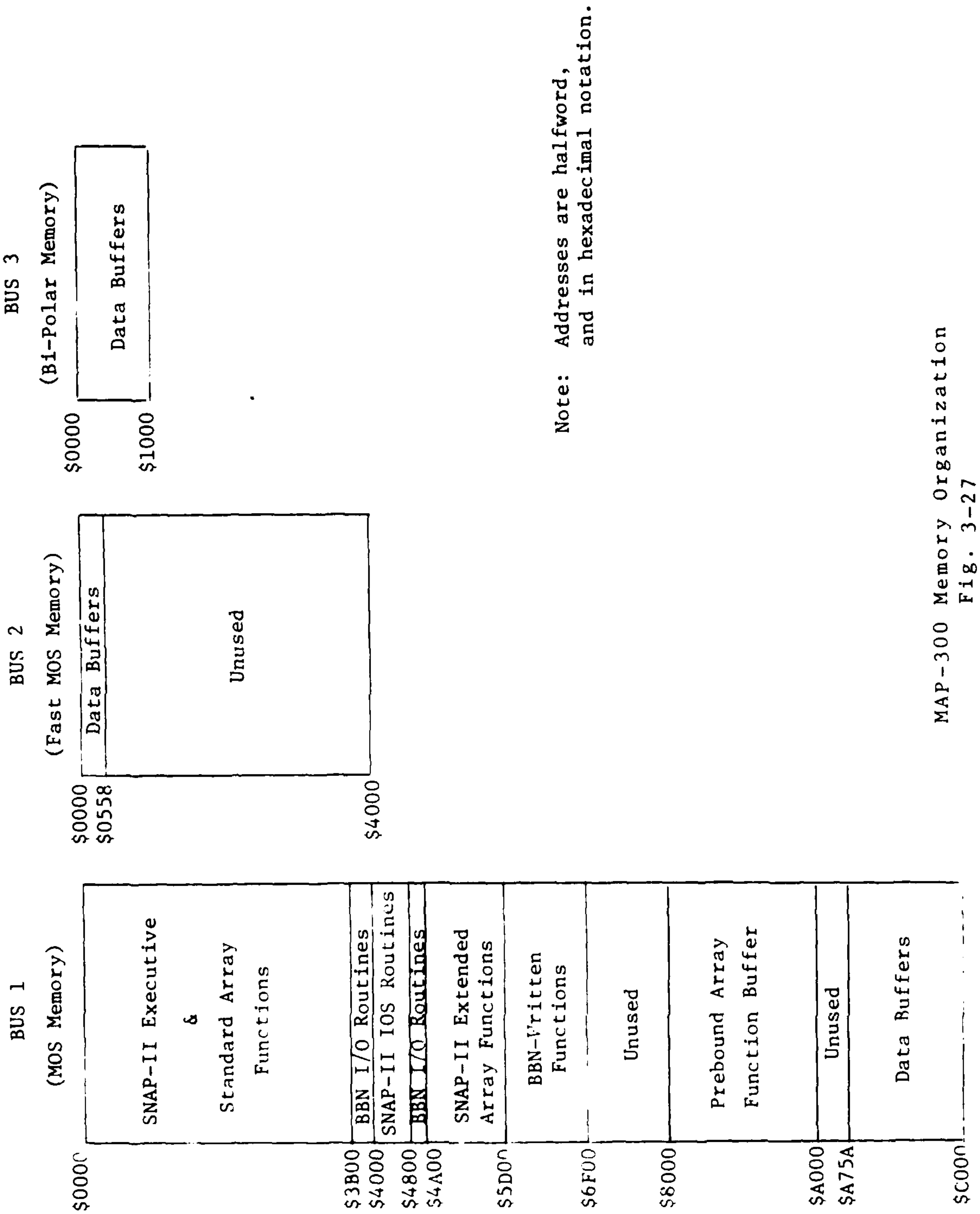
SNAP-II Software System Release 3.5 Model 8300-RSX11M.
(This package includes the SNAP-II Executive and the Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package Release 0.1 Model 8400-RSX11M.
(This package includes the SNAP-II IOS Modules.)

These software modules are described in CSPI documentation, specifically:

"SNAP-II Reference Manual" (JB6000-017-01)

"Release Note, SNAP-II Release 3:
Executive Features and Expanded Array Function Set"



MAP-300 Memory Organization
Fig. 3-27

(DW6000-021-04)

"SNAP-II Input/Output Scroll Package, User's Manual" (JW8400-001-01).

3.4.1.2 BBN-written MAP-300 Software

BBN-written additions and modifications to the SNAP-II software system are contained in three separate files. (For all BBN-written MAP-300 modules, a file extension of .MSO designates "MAP source", .MOB designates "MAP object", and .MLI designates "MAP listing".) The files with first name "BBN300" contain added SNAP-II functions (array and non-array) for algorithmic and system support processing as well as tables for decoding and quantization. "BBNIOS" contains PROTECT and CORRECT, ADAM, AOM, and IOS-2 programs, and CSPU routines that respond to interrupts from these devices. These programs and routines are described in section 3.2 of this report. "BBNPAT" contains a patch to Release 3.5 of the SNAP-II executive to allow for proper operation of the SNAP-II "prebinding" process when using a prebinding buffer located above X'7FFF'. Assembly-listings of these three files appear in Appendix E of this report.

BBN-written SNAP-II functions are enumerated in Fig. 3-28 and functionally described in Appendix B of this report. Each function has been assigned a Function Control Block (FCB) number. An entry has been made for each new FCB in the Function Dispatch Table

(FDT), indicating the location of the APU, APS and/or CSPU module(s) that implement the associated function.

Several unused CSPI-supplied functions, normally available with Release 3.5 of the SNAP-II software system, have been disabled to provide room in the FDT for BBN-written functions. The following SNAP-II functions are unavailable once the BBN-written MAP-300 software has been loaded and executed:

DPRE	(FCB# 191)	MWLD	(FCB# 225)
FFT2D	(FCB# 212)	ADMRB	(FCB# 229)
CMML	(FCB# 220)	VHIST	(FCB# 230)
CMINV	(FCB# 224)	VRANL	(FCB# 235)

These functions can be made available by reloading the SNAP-II software system without loading BBN-written MAP-300 software.

In addition, the standard SNAP-II interrupt routines for interrupts from Device 16 (Lines 1 and 2), Device 22 (Line 1), and Device 23 (Line 1) have been modified to transfer control to BBN-written interrupt routines. The standard interrupt routines for these devices can be similarly accessed by reloading the SNAP-II Executive without loading BBN-written MAP-300 software.

3.4.2 PDP-11 Software Components

The proper operation of the speech coder is defined and controlled by a group of programs running in the PDP-11. These programs include a set of SNAP-II support subroutines and a MAP-300 driver, supplied by CSPI, as well as a number of FORTRAN programs written by BBN that make use of the CSPI-supplied routines to perform the specific task of defining and controlling the operation of the speech coder in the MAP.

Function Name	Name Expansion	Contained in File:	FCB
CORECT(I)	Perform error correction	BBNIOS	122
DCOR(Y,U,V)	Discrete correlation	BBN300	191
DEAL(Y,A,U,B,V)	"Deal" buffer contents to buffers and scalars	BBN300	190
ENRG(A,B,C,W,D)	Compute, code and quantize energy	BBN300	199
IADINT(0)	Simulate A/D interrupt	BBNIOS	124
IDAINT(0)	Simulate D/A interrupt	BBNIOS	127
IRMINT(0)	Simulate RMODEM interrupt	BBNIOS	126
ITMINT(0)	Simulate TMODEM interrupt	BBNIOS	125
MPGSC(GFLAG,SETCLR)	G-flag set/clear	BBNIOS	123
MPIFF(IA,IB,FLID)	If(IA.NE.0 & IB.EQ.0) Conditional function list execution	BBN300	105
MPMBS(Y,A,N)	Move buffer to scalar	BBN300	111
MWLF(Y,A,U,V)	Matrix(Wiener-Levinson-Durbin) solution, with quantized & coded output	BBN300	135
PROTCT(I)	Perform error protection	BBNIOS	121
PRTRB(Y,A,U,B,V)	Upsample (3:1) with perturbation	BBN300	134
PTAP(Y,A,U,B,V,C,W)	Compute pitch & tap and do pitch removal	BBN300	212
VAPC(Y,A,U,B,V,C,D)	APC	BBN300	150
VIAPC(Y,A,U,B)	Inverse APC	BBN300	196
VKTOA(Y,U)	Convert reflection coefficients to linear predictor coefficients	BBN300	133
VLTSY(Y,U,V,W)	Lattice synthesis filter	BBN300	132

Figure 3-28
BBN-written SNAP-II functions

CSPI-supplied PDP-11 programs are described in section 3.4.2.1. PDP-11 programs written by BBN are described in section 3.4.2.2.

3.4.2.1 CSPI-Supplied PDP-11 Software

The following CSPI-supplied PDP-11 software modules are required for speech coder operation:

SNAP-II Software System Release 3.5 Model 8300-RSX11M. (This package includes the SNAP-II Host Support Packages for Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package Release 0.1 Model 8400-RSX11M (This package includes the SNAP-II IOS Host Support Package.)

DEC RSX-11M I/O Driver Model 8901

These software modules are described in CSPI documentation, including those listed in Section 3.4.1.1 of this report, as well as:

"Installation Procedure for Release 3 SNAP-II Software System on the DEC RSX-11M System" (LL8901-004-03)

"MAP Software Interface Description for the DEC RSX-11M System" (ST8901-000-02).

3.4.2.2 BBN-written PDP-11 Software

A set of BBN-written FORTRAN programs performs the multiple tasks of defining and initializing MAP-300 buffers and scalars and of defining and executing the sequence of functions in the MAP that perform the actual speech coding operations. These FORTRAN programs are organized around a single mainline program (DCA96M), which calls, in turn, subroutines to configure MAP buffers and scalars (DCA96C), initialize these buffers and scalars (DCA96I), define function lists for various speech coder tasks (DCA96F), and interact with the user in controlling the execution of these function lists (DCA96E). These subroutines are described in sections 3.4.2.2.1 through 3.4.2.2.4 below.

3.4.2.2.1 Buffer and Scalar Configuration (DCA96C)

The task of defining MAP-300 buffers and scalars to the SNAP-II executive is performed by subroutine DCA96C. Upon entry, this routine first initializes the SNAP-II executive via the MPOPN function. All SNAP-II buffers are then configured using the MPCLB function, with buffer ID numbers, sizes, and addresses defined symbolically within the DCA96C routine. In general, transmitter buffer names begin with "T", while receiver buffer names begin with "R". Buffer sizes are stored in variables named by appending an "S" in front of the buffer name. Similarly, buffer addresses are stored in variables named by appending "A" in front of the buffer name. The buffers used in the speech coder system are listed in Appendix C. All real scalars and integer scalars are then defined by simply assigning scalar ID numbers to the associated symbolic scalar names. Transmitter scalar names generally begin with "T", while receiver scalar names begin with "R". Real and integer scalars used in the speech coder system are listed in Appendix D.

All buffer and scalar names (and certain buffer sizes and addresses) are included in FORTRAN labeled common blocks. All other FORTRAN subroutines in the speech coder system can then reference these symbolic buffer and scalar names in their calls to SNAP-II functions.

3.4.2.2.2 Buffer and Scalar Initialization (DCA96I)

MAP-300 buffers and scalars are set to their initial values by subroutine DCA96I. Only certain buffers and scalars require such initialization. Appendices C and D indicate the proper initial contents of MAP buffers, real scalars, and integer scalars.

File DCA96S contains a set of FORTRAN subroutines called by DCA96I and used to calculate the initial contents of certain buffers. Included are subroutines that generate a square-wave of a given frequency and amplitude (SQWAV), a Hamming window of a given size (HAMMNG), a 75 point symmetric low or high pass filter with passband and stopband edges at 925 Hz and 1111 Hz and with a stopband rejection of -35 dB (LPFHPF), and a sequence of Gaussian random numbers with an average value of 0 and a standard deviation of 1 (RANDOM).

3.4.2.2.3 Control Structure Definition (DCA96F)

Subroutine DCA96F defines a set of function lists that specify the operation of the speech coder system. Function list ID numbers are defined symbolically. The FORTRAN variables containing the function list ID numbers are included in a labeled common block so that they can be referenced by other subroutines.

As part of the function list definition task, subroutine DCA96F produces pre-bound versions of all of the SNAP-II array functions used in the speech coder system. Pre-binding is a SNAP-II operation that causes function parameter information, normally communicated to the function at execution time, to be "bound" to the function at some earlier time (in this case at system initialization time). Approximately 8000 halfwords are required on Bus 1 to store the pre-bound versions of the speech coder array functions.

Function lists ANLZA, ANLZB, SYNZA, and SYNZB are defined for the analysis and synthesis sections of the speech coder. (Since the system is double-buffered, the analysis and synthesis function lists are each defined twice to allow for the different sets of input and output buffers.) These function lists specify the sequence of MAP-300 array and non-array functions, operating on the system buffers and scalars previously defined, that implement the analysis and synthesis processes described in sections 3.2.1.3 and 3.2.2.3 of this report.

The analysis and synthesis function lists described above are used in the specifications of several other function lists that define the outer structure of the speech coder system. Function list DCALP defines the real-time speech coder loop, executing function lists ANLZA, SYNZA, ANLZB, and SYNZB when buffer status flags indicate that the input buffers are full and the output buffers are empty for each of these processes. Function list DCARTS defines the start-up sequence for the real-time speech coder system, starting the Modem, ADAM, and AOM Scroll processors, and repeatedly executing the DCALP function list described above. This repeated execution is contingent upon the contents of MAP integer scalar RUN being non-zero; hence, the speech coder can be stopped by setting RUN to zero. Function list DRSTRT restarts the real-time speech coder by setting RUN to non-zero and reinitiating the repeated execution of the real-time speech coder loop.

Two other function list definitions are included in this subroutine to support non-real-time file-to-file speech coder operation, speech coder timing operation, and speech coder oscilloscope display operation (for detailed internal timing). These modes of speech coder operation are not supported, and can not be invoked, in the delivered speech coder system. They were used for system development and debugging use under the RT-11 operating system, and are included here to aid in future additions to the speech coder system. Function list DCAFFT specifies the speech coder loop with explicit calls to functions that execute the A/D, RMODEM, D/A, and TMODEM interrupt routines. Function list DCATIM repeatedly invokes a sequence of six DCAFFT function list executions (one for each possible combination of input and output buffers), with software simulation of the transmitter-to-receiver communication path occurring after each DCAFFT execution.

All SNAP-II functions are called via PDP-11 support subroutines that perform the actual communication with the MAP driver. These subroutines are contained in library SNPLIB for CSPI-supplied functions, and in file BBNHSP for BBN-written functions.

3.4.2.2.4 System Software Execution (DCA96E)

The execution of the speech coder system is controlled by subroutine DCA96E. This subroutine starts the real-time speech coder system by first loading the Modem, ADAM, and AOM Scrolls with their respective programs, and then executing the real-time speech coder start-up function list (DCARTS) described in the preceding section. It then interacts with the user, responding to single-character commands to halt the speech coder ('Q'), enable/disable error simulation ('E'), enable/disable error correction ('C'), cause the speech coder to lose sync artificially ('L'), type out a group of speech coder state scalars ('T'), and suspend the controlling RSX-11M task, allowing the speech coder to continue executing in the MAP-300 ('S').

Provisions are included in this subroutine for user selection of other speech coder modes of operation, including file-to-file, timing, and oscilloscope display modes. However, this user selection process is bypassed in the delivered speech coder system, and real-time speech coder execution mode is forced. As described in the previous section, these other operating modes are not supported, and cannot be invoked, in the delivered speech coder system. They were intended for system development and debugging use, and are included here to aid in future additions to the speech coder system. File DCA96D contains dummy versions of various system-dependent subroutines, which constitute the unsupported software portions of these unsupported modes of speech coder operation.

3.5 SYSTEM TIMING PERFORMANCE

The speech coder system introduces a total delay of 11 frames, corresponding to 299.2 milliseconds, between voice input and synthesized voice output (not including any delays in transmission). This consists of 5 frames of delay introduced by the transmitter portion of the system and 6 frames of delay contributed by the receiver portion. In the transmitter, a frame of delay each is produced by the buffering in the ADAM scroll program, the ANALYZE processing module, and the TMODEM scroll program. In addition, the ANALYZE module introduces a frame of processing delay due to the baseband extraction module (see Section 3.2.1.3.3) and a frame of "folding" delay due to the concurrent execution of the ANALYZE module and the PROTECT module (see Section 3.2.1).

In the receiver, two frames of delay are contributed by the buffering in the RMODEM scroll program, and a frame each is produced by buffering in the SYNTHESIZE processing module and the AOM scroll program. In addition, the SYNTHESIZE module introduces a frame of processing delay due to the high frequency regeneration module (see Section 3.2.2.3.4) and a frame of "folding" delay due to the concurrent execution of the SYNTHESIZE module and the CORRECT module (see Section 3.2.2).

The observed internal timing of the speech coder modules is given in Fig. 3-29. This figure reflects data obtained by running the speech coder in a loop without the ADAM, AOM, and MODEM scroll processors and observing the states of the MAP-300 RA (APU RUN) and G-flags with an oscilloscope. The speech coder was provided with

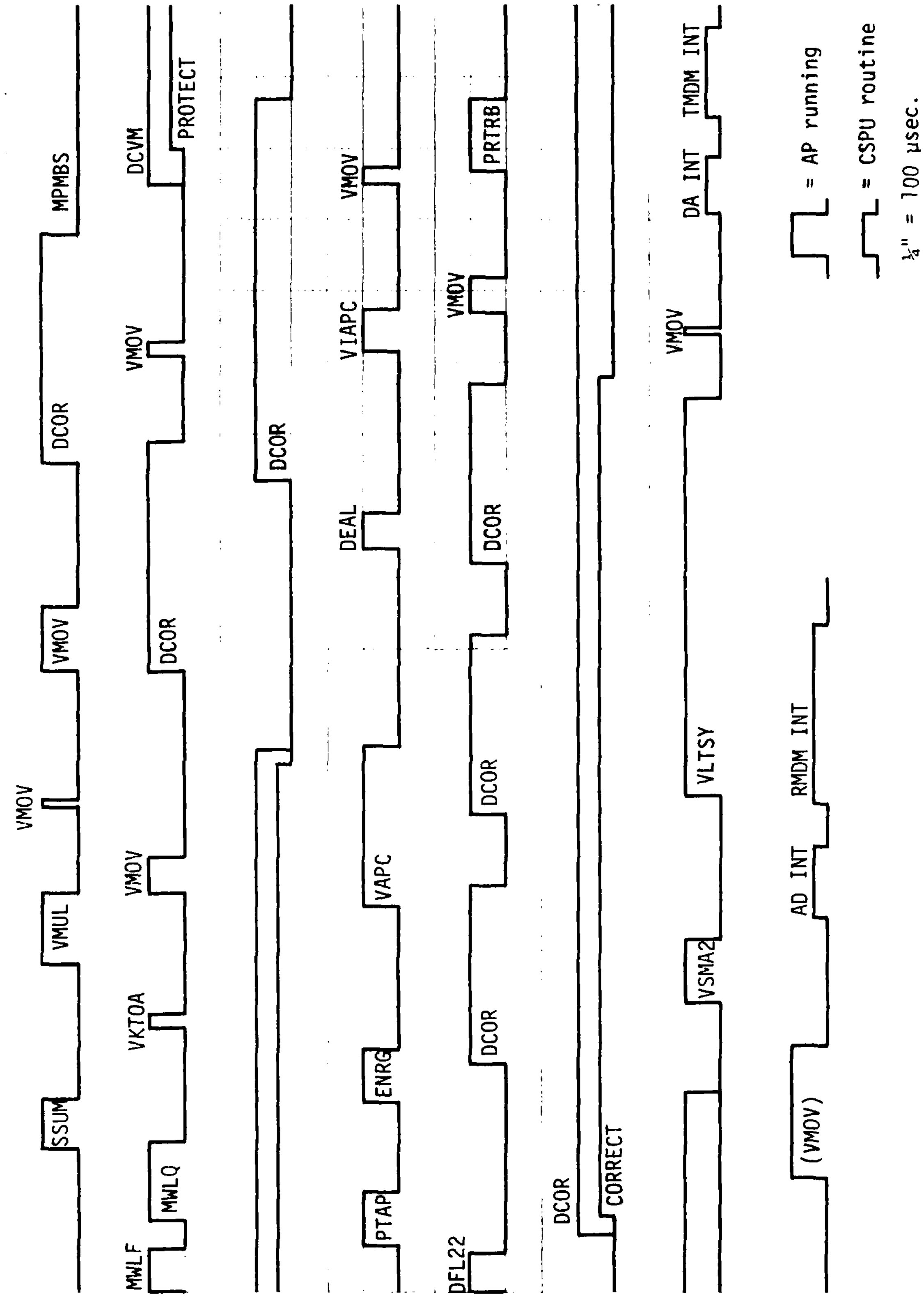


Figure 3-29 MAP-300 Timing

simulated A/D input data, and ADAM, AOM, and MODEM scroll interrupts were simulated by SNAP-II calls to the appropriate interrupt routines. Modem data transmission was simulated by a SNAP-II "VMOV" operation from TMODEM to RMODEM buffers. After deducting time for processing that does not occur in the real-time version of the speech coder (SNAP-II overhead for interrupt routines, SNAP-II "VMOV" for simulation of the transmission), Fig. 3-29 shows a total processing time of 25.6 milliseconds for a 27.2 millisecond frame of data, or about 0.94 times real time. (This is the worst case total processing time. If the ADAM, AOM, or MODEM scroll interrupts were to occur while the AP was running, some or all of the interrupt routine execution time would be hidden behind the concurrent AP routine.)

3.6 SYSTEM TIMING CONSIDERATIONS

A significant portion of the execution time required by the speech coding system is devoted to overhead tasks. Since the MAP-300 APS and APU program memories are small compared to the amount of code required by the speech coding system, program modules must be loaded into these memories at execution time. For most modules, these loading times can be minimized by using the "pre-binding" capability of SNAP-II, which binds parameters to functions at initialization time, rather than during execution. Some SNAP-II functions, notably the FFT, cannot be prebound, so the binding time must be included in the execution overhead.

Since the function queue management is performed by the CSPU, it can be done while the APU and APS are executing other functions. In order to take advantage of this parallel processing, however, the APU and APS functions must be long enough to mask the overhead tasks, which is not always the case. Moreover, some of the overhead tasks cannot be overlapped with APU/APS execution. This non-overlappable overhead typically takes 200 micro-seconds per function.

The CSPU has a more general purpose architecture than the other processors in the MAP, and, as such, is an attractive possibility for performing non-array portions of algorithms. However, the CSPU is quite slow compared to the rest of the MAP, having a typical instruction execution time of 2 to 3 microseconds. It is also kept fairly busy performing SNAP-II tasks such as the processor loading and queue management discussed above. Therefore, we have found that it is not always efficient to have the CSPU perform those tasks, such as table look-ups, for which it seems, on the surface, best suited. Finally, if a particular task, such as error protection, must be performed by the CSPU, the execution of the task should be overlapped with a suitably long APU/APS function if possible.

We found the documentation of the overhead times and the function execution times to be inadequate and inaccurate. As explained in Section 3.5, we used an oscilloscope to display internal MAP conditions,s, such as the APU run flag, in order to obtain accurate per-function timing information. In this manner, we found that the SNAP-II DCVM function (time-domain convolution) function, in particular, required 2 to 3 times as long for APU/APS execution as the times given in the SNAP-II manual. The manuals also give little indication of the additional overhead time costs associated with a function. We conclude that any new MAP implementation efforts should be begun with timing tests on the SNAP functions intended to be used.

3.7 ARCHITECTURAL CONSTRAINTS

The implementation of the speech coder on the MAP-300 brought out several aspects of the MAP architecture that are not well matched to real-time speech coding applications. This section discusses these architectural constraints.

The MAP appears to have been designed with a certain kind of "array processing" in mind. A primary premise of this kind of array processing appears to be that the computations to be performed do not depend on the data being operated on, that the flow of control of the computation process be data-independent. The design of the Arithmetic Processor (AP) as a separate Arithmetic Processor Addresser (APS) and Arithmetic Processor Unit

(APU) exemplifies this philosophy. The APS computes addresses for operands, but it is unable to examine or modify those operands; the APU performs computations on operands, but it is unable to know about or modify the addressing sequence; there is little communication between these processors.

Examples of computations that fit this philosophy are the DFT, convolution/correlation, and digital filtering. A restricted class of data-dependent operations that do not alter the addressing sequence, such as vector-clipping or the "perturbed upsampling" used in the BBN speech coder are also not difficult to implement. Other computations, such as the pitch-APC loop, where the relative addresses of the operands depend on a piece of data, or nested loops, where the iteration counts are data dependent, require efficiency-robbing processor synchronization or other esoteric techniques. Computations requiring even more data-dependence of control, such as sorting or table-lookup, become impractical. Although speech processing applications use many operations from the first class, they also include the other kinds of operations, often in the form of heuristics. These can be difficult to accomplish efficiently on the MAP.

Another architectural feature is the small program memories of the MAP processors. They must be loaded before each separate operation, and during this loading time, they are idle. If the processing they do is long with respect to this idle time, then this overhead is small. The limited stationarity of speech signals, however, generally dictates relatively short data buffers (100-250 samples), so the AP execution times are often short enough that the intermodule overhead is significant.

The MAP architecture also affects the approach used for software implementation. The system of small processor memories that must be loaded for each program module to be executed dictates the need for an executive program to do this, as well as to bind arguments to program modules and to respond to interrupts. Rather than write our own special purpose executive, we elected to implement the speech coder within the SNAP-II software system supplied by the manufacturer. This general purpose system brought with it a new set of constraints, but we felt these were outweighed by SNAP-II's features.

The MAP architecture uses asynchronous processors and memories. This division-of-the-labor is a two-edged sword; it can be quite efficient in execution, but the separation and asynchrony creates an environment in which interprocessor interaction is difficult to debug. The MAP simulator program supplied by the manufacturer is not a satisfactory approach to developing these aspects of a program because it omits some of the interprocessor flags that must be used to effect the interaction; it is not a faithful simulation of the more difficult aspects of the MAP architecture.

One final point concerns the paucity of communication paths between the Scroll processors and the CSPU. For a real-time application subject to exceptional events (such as clock drift or interruption of the received bitstream), it should be possible to control the input/output processes in such a way to be responsive to these events. The CSPU's control of a Scroll processor, however, is limited to starting and stopping it, and since the Scroll processor can only transfer data, not examine it, there is no way for it to be responsive to external conditions. In the present speech coder implementation, it was necessary to adopt a system of pairs of double buffers at each I/O port in order to put a layer of control (the Scroll interrupt routines) between the signal processing modules and the blindly-executing Scroll processes.

4. SOFTWARE OPERATING PROCEDURES

The installation and execution of the speech coder system software are accomplished using the RSX-11M operating system as well as MAP-300 support software supplied by CSPI. It is assumed that the RSX-11M operating system, the MAP-300 device drivers, the MAP-300 loaders, and the SNAP-II software system have all previously been installed prior to the speech coder system installation.

After the speech coder system software has been installed, the speech coder can be executed by using the procedure given in Section 4.1. The software installation procedure is given in Section 4.2.

4.1 SOFTWARE EXECUTION PROCEDURE

The speech coder system execution procedure consists of two major steps. First, both MAP-300 processors in the full-duplex speech coder system must be loaded from the host PDP-11 with identical copies of the speech coder MAP-300 software. Second, two RSX-11M tasks must be run to initialize and start the two MAP-300 speech coders.

The following files must be disk-resident before speech coder execution can be attempted:

```
BBN96.MBN       (MAP Binary file)
BBN96A.TSK      (Host Task for MAP #A)
BBN96B.TSK      (Host Task for MAP #B)
```

If these files do not exist, they must be created according to the speech coder system software installation procedure given in Section 4.2.

The following dialog indicates the proper command sequence for full-duplex speech coder system execution. User commands are underscored to differentiate them from computer responses.

```
>SET /UIC=[5,200]
>ALLOCATE MA:            **allocate MAP #A**
>ALLOCATE MP:            **allocate MAP #B**
>INSTALL BBN96A
>INSTALL BBN96B
>RUN [200,200]AMPLD      **load MAP #A**
OBJECT INPUT?(carriage return)
BINARY INPUT?BBN96.MBN
TT2 -- STOP

>RUN MPLD                **load MAP #B**
OBJECT INPUT?(carriage return)
BINARY INPUT?BBN96.MBN
TT0 -- STOP

>RUN BBN96A              **start MAP #A**
BBN 9600 BPS MAP-300 VOCODER SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS...
EXECUTING DCA96 SYSTEM...

VOCODER IS IN OPERATION.

COMMANDS ARE:
 S:   SUSPEND TASK (LEAVING VOCODER RUNNING)
 Q:   QUIT (HALT VOCODER)
 E N: SIMULATE N ERRORS PER FRAME
```

```
 C N: ERROR CORRECT IF N=0
 L:   CAUSE RCVR TO LOSE SYNC
 T:   TYPE OUT VOCODER STATE

ENTER COMMAND: S
BBN96A -- PAUSE (VOCODER CONTINUING)

>RUN BBN96B              **START MAP #B**
BBN 9600 BPS MAP-300 VOCODER SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS...
EXECUTING DCA96 SYSTEM...

VOCODER IS IN OPERATION.

COMMANDS ARE:
 S:   SUSPEND TASK (LEAVING VOCODER RUNNING)
 Q:   QUIT (HALT VOCODER)
 E N: SIMULATE N ERRORS PER FRAME
 C N: ERROR CORRECT IF N=0
 L:   CAUSE RCVR TO LOSE SYNC
 T:   TYPE OUT VOCODER STATE

ENTER COMMAND: S
BBN96B -- PAUSE (VOCODER CONTINUING)

>
```

At this point in the command sequence, the full-duplex speech coder system should be in operation. Host tasks BBN96A and BBN96B can be resumed (for interaction with the speech coder software) by invoking the RSX-11M "RESUME" command. The speech coder system can be halted either by responding with "Q" to the "ENTER COMMAND:" typeout from BBN96A and BBN96B or by aborting the BBN96A and BBN96B tasks through the use of the RSX-11M "ABORT" command.

4.2 SOFTWARE INSTALLATION PROCEDURE

The speech coder system software installation procedure consists of two major steps. First, the speech coder MAP-300 software must be converted from MAP object format to MAP binary format. Second, two RSX-11M tasks (for initializing and starting the two MAP-300 processors) must be task-built.

The following files must be disk-resident before speech coder software installation can be attempted:

```
BBNMAP.MOB                (MAP object file)
             (This  file  consists  of  the  following
             concatenated   files,   with   intervening
             "FFFF" lines deleted:
                               S300EX.MOB
                               EAF300.MOB
                               IOS300.MOB
                               BBN300.MOB
                               BBNIOS.MOB
                               BBNPAT.MOB)
DCA96C.FOR             (HOST FORTRAN module)
DCA96D.FOR             (       "           )
DCA96E.FOR             (       "           )
DCA96F.FOR             (       "           )
DCA96I.FOR             (       "           )
DCA96M.FOR             (       "           )
DCA96S.FOR             (       "           )
BBNHSP.FOR             (       "           )
```

If these files do not exist, they must be recopied to disk from the "BBN 9600 BPS Vocoder S/W" magnetic tape delivered with the speech coder system.

The MAP-300 speech coder software should be converted from MAP object format to MAP binary format using the following command sequence:

```
>RUN [200,200]MPLD
OBJECT INPUT?BBNMAP.MOB
BINARY OUTPUT?BBN96.MBN
LOAD MAP?(Y OR N) N
TT2 -- STOP 2
>
```

The two RSX-11M tasks (for initializing and starting the two MAP-300 processors) should be generated next. First, all FORTRAN speech coder modules listed above must be compiled. Then the two tasks must be task-built, with each task including the compiled FORTRAN object modules, the CSPI-supplied SNAP host support library modules, and the appropriate CSPI-supplied MAP device driver for MAP #A or MAP #B. The following command sequence will accomplish this procedure:

```
>F4P DCA96C=DCA96C.FOR/CO:35
>F4P DCA96D=DCA96D.FOR/CO:35
>F4P DCA96E=DCA96E.FOR/CO:35
>F4P DCA96F=DCA96F.FOR/CO:35
>F4P DCA96I=DCA96I.FOR/CO:35
>F4P DCA96M=DCA96M.FOR/CO:35
>F4P DCA96S=DCA96S.FOR/CO:35
>F4P BBNHSP=BBNHSP.FOR/CO:35
>TKB                         **build host task for MAP #A**
TKB>BBN96A/FP/CP=DCA96M,DCA96C,DCA96D
TKB>DCA96E,DCA96F,DCA96I,DCA96S,BBNHSP
TKB>[200,200]ASPLIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=10
TKB>//
>TKB                         **build host task for MAP #B**
```

```
TKB>BBN96B/FP/CP=DCA96M,DCA96C,DCA96D
TKB>DCA96E,DCA96F,DCA96I,DCA96S,BBNHSP
TKB>[200,200]SNPLIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=10
TKB>//
>
```

The speech coder system software is now installed. The speech coder can be executed by using the procedure given in Section 4.1.

REFERENCES

1. Proposal for "Speech Algorithm Optimization at 9600 Bits per Second", BBN Proposal No. P78-ISD-76, Submitted to Defense Communications Agency, June 1978.

2. J. L. Flanagan et al., "Speech Coding", IEEE Trans. Commun., Vol. COM-27, pp. 710-737, April 1979.

3. R. Viswanathan, W. Russell and J. Makhoul, "Voice-Excited LPC Coders for 9.6 KBPS Speech Transmission", Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, Washington, D.C., pp. 558-561, April 1979.

4. M. R. Schroeder, "Recent Progress in Speech Coding at Bell Telephone Laboratories", in Proc. IIIrd Int. Congr. Acoust., Stuttgart, Germany, 1959.

5. M. R. Schroeder and E. E. David, Jr., "A Vocoder for Transmitting 10 kc/s Speech Over a 3.5 kc/s Channel", Acustica, Vol. 10, pp. 35-43, 1960.

6. E. E. David, M. R. Schroeder, B. F. Logan, and B. F. Prestigiacomo, "New Applications of Voice-Excitation of Vocoders", in Proc. Stockholm Speech Commun. Seminar, Royal Inst. Technol., Stockholm, Sweden, 1962.

7. B. Gold and J. Tierney, "Digitized Voice-Excited Vocoder for Telephone-Quality Inputs, Using Bandpass Sampling of the Baseband Signals", J. Acoust. Soc. Amer., Vol. 37, pp. 753-754, April 1965.

8. C. K. Un and D. T. Magill, "The Residual-Excited Linear Prediction Vocoder with Transmission Rate Below 9.6 kbits/s", IEEE Trans. Comm., Vol. COM-23, pp. 1466-1474, December 1975.

9. C. J. Weinstein, "A Linear Prediction Vocoder with Voice Excitation", Proc. EASCON'75, pp. 30A-30G, September 29-October 1, 1975, Washington, D. C.

10. C. J. Weinstein, "Comments on VELP and RELP", ARPA NSC Note 51, 1975.

11. D. Esteban, C. Galand, D. Mauduit, and J. Menez, "9.6/7.2 KBPS Voice Excited Predictive Coder (VEPC)", Proc. IEEE Int'l Conf. Acoustics, Speech, Signal Processing, Tulsa, OK, pp. 307-311, April 1978.

12. B. S. Atal, M. R. Schroeder, and V. Stover, "Voice-Excited Predictive Coding System for Low Bit-Rate Transmission of Speech", Int'l Conf. Communications, San Francisco, CA, June 16-18, 1975.

13. A. Higgins, R. Viswanathan and W. Russell, "New High-Frequency Regeneration (HFR) Techniques for Voice-Excited Speech Coders", J. Acoust. Soc. Amer., Vol. 66, S22 (Abstract), November 1979.

14. J. Makhoul and M. Berouti, "High-Frequency Regeneration in Speech Coding Systems", Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, Washington, D. C., pp. 428-431, April 1979.

15. R. Viswanathan and J. Makhoul, "Quantization Properties of Transmission Parameters in Linear Predictive Systems", IEEE Trans. Acoustics, Speech, and Signal Processing, Vol. ASSP-23, pp. 309-321, June 1975.

16. R. Viswanathan, J. Makhoul and A. W. F. Huggins, "Speech Compression and Evaluation", Final Report, BBN Report No. 3794, Bolt Beranek and Newman Inc., Cambridge, MA, April 1978.

17. J. Max "Quantizing for Minimum Distortion", IRE Trans. Info. Theory, Vol. IT-6, pp. 7-122, March 1960.

18. T. Tremain, J. W. Fussell, R. A. Dean, B. M. Abzug, M. D. Cowing and P. W. Boudra, Jr., "Implementation of Two Real-Time Narrowband Speech Algorithms", Proc. EASCON'78, pp. 698-708, September 1978.

19. H. Gish and J. N. Pierce, "Asymptotically Efficient Quantizing", IEEE Trans. Info. Theory, Vol. IT-14, No. 5, pp. 676-683, 1968.

20. J. Makhoul and M. Berouti, "Adaptive Noise Spectral Shaping and Entropy Coding in Predictive Coding of Speech", IEEE Trans. ASSP, Vol. ASSP-27, pp. 63-73, February 1979.

21. M. Berouti and J. Makhoul, "An Adaptive-Transform Baseband Coder", Presented at the 97th Meeting of the Acoustic Society of America, Cambridge, MA, June 1979.

22. R. Crochiere, D. Goodman, L. Rabiner, and M. Sambur, "Analysis of the Performance of a Tandem Connection of a 2.4 Kbps LPC Vocoder to a 16 Kbps CVSD Coder", Bell Syst. Tech. J., Vol. 56, No. 9, pp. 1701-1722, November 1977.

23. T. P. Barnwell, R. W. Schafer, and A. M. Bush, "Evaluation of LPC/CVSD Tandem Connections", Proc. IEEE Int'l. Conf. Acoustics, Speech, Signal Processing, Tulsa, OK, pp. 326-329, April 1978.

24. L. Rabiner, M. Sambur, R. Crochiere, and D. Goodman, "Analysis of the Performance of a Tandem Connection of a 16 Kbps CVSD Coder to a 2.4 Kbps LPC Vocoder", Bell Syst. Tech. J., Vol. 56, No. 9, pp. 1723-1741, November 1977.

25. L. E. Bergeron, "A Spectral Enhancement Procedure for the Wideband/Narrowband Tandem", Proc. IEEE Int'l. Conf. on Acoustics, Speech, and Signal Processing, Tulsa, OK, pp. 330-333, April 1978.

26. M. Berouti, R. Schwartz and J. Makhoul, "Enhancement of Speech Corrupted by Acoustic Noise", Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, Washington, D. C., April 1979.

27. P. Noll, "On Predictive Quantizing Schemes", Bell Syst. Tech. J., Vol. 57, pp. 1499-1532, May-June 1978.

28. B. S. Atal and M. R. Schroeder, "Predictive Coding of Speech Signals and Subjective Error Criteria", IEEE Trans. ASSP, Vol. ASSP-27, pp. 247-254, June 1979.

29. R. Viswanathan, W. Russell, A. Higgins, M. Berouti and J. Makhoul, "Speech-Quality Optimization of 16 kb/s Adaptive Predictive Coders", To be presented at the IEEE Int'l Conf. Acoustics, Speech, and Signal Processing, Denver, CO, April 1980.

30. J. Makhoul, R. Viswanathan, L. Cosell, and W. Russell, "Natural Communications with Computers: Speech Compression Research at BBN", Report No. 2976, Vol. II, Bolt Beranek and Newman Inc., Cambridge, MA, December 1974.

APPENDIX A: EQUIPMENT DESCRIPTION OF GTE AUDIO AND MODEM INTERFACES

This Appendix consists of relevant portions of the GTE-supplied Equipment Description describing the audio signal interface and the IOS-2SM scroll modifications for the modem modem interface. Note that the original description of the IOS-2SM modem interface design is incorrect; a March 19 addendum (included as the last page of this Appendix) describes the modified design.

DCA 9600 BPS
OPTIMIZATION STUDY

EQUIPMENT DESCRIPTION

March 7, 1979

GTE SYLVANIA INCORPORATED
Electronic Systems Group
Eastern Division
77 "A" Street
Needham Heights, Massachusetts 02194

The equipment required to support the DCA 9600 BPS Speech Processing Study is comprised of two major elements: an audio interface assembly and a modified CSPI IOS-2SM I/O scroll.

The audio interface assembly provides the amplification, equalization, and filtering necessary to interface the handset to the MAP-300 A/D and D/A converters, and also serves as the common junction point for all digital signals between the MAP-300 and an external modem. A switch on the front panel permits the selection of either of two sets of filters, thereby permitting a choice of cutoff frequency. Filters having cutoff frequencies of 3200 Hz and 3800 Hz are provided. The filters are of the plug-in type, thereby enabling the user to install other filters with different cutoff frequencies of his choice if so desired.

The handset provided with the equipment uses a dynamic microphone which has been designed to GTE Sylvania specifications and has been optimized for use in speech processing applications. The handset connects directly to the front panel of the audio interface assembly and may be stored on the hookswitch, which is also located on the front panel. When "on hook," the audio circuits (both receiving and transmitting) for the handset are disabled. A 25-foot extension cable for use with the handset is also provided.

A pair of telephone jacks located on the rear panel of the audio interface unit may be used to connect a tape recorder or test equipment for test and measurement purposes. The audio circuits for the tape recorder are always active and are unaffected by the operation of the hookswitch. A single connector, also located on the rear panel, is used to connect the audio interface assembly to the A/D - D/A converters of the MAP-300.

The CPSI IOS-2SM scroll has been modified by GTE Sylvania to provide a means for interfacing to any modem employing an EIA RS-449/RS-423 interface. The interface design is such that an EIA RS-232-C interface may also be used, and the line drivers and receivers have been selected such that the protective networks for RS-449/RS-232 interoperation (refer to EIA Industrial Bulletin No. 12) are not required. In addition to the modem interface, the modified I/O scroll includes a programmable real-time clock which generates the timing signals for speech sampling and the modem data. The I/O scroll is connected to the audio interface assembly by means of a single cable.

The data and speech sampling rates are set by issuing a single 16-bit control word from the IOS-2SM. The most significant byte of the control word determines the data rate, whereas the least significant byte determines the speech sampling rate. Once the real-time clock has been so programmed, it is not necessary to issue any other control words unless it is desired to change either of the clock rates, or unless power is removed from the MAP-300. It should be noted that the entire control word must be issued whenever changing rates, even if only one rate is to be changed.

Data transfers between the MAP-300 and the modem take place via the IOS-2SM data bus. The data transfer process is interrupt driven, with timing based on the transmit data clock, for output transfers, and the modem receiver clock, for input transfers. The interrupts will, therefore, occur at the data rate, thereby allowing sufficient time for the IOS-2SM to acknowledge each interrupt and perform the appropriate data transfer. The interrupts are controlled by a two-phase clock such that simultaneous interrupts can never occur. Furthermore, the interrupts are mutually exclusive so that the IOS-2SM can receive only one interrupt at a time. Interrupt number 1 is used for receive data and interrupt number 2 is used for transmit data.

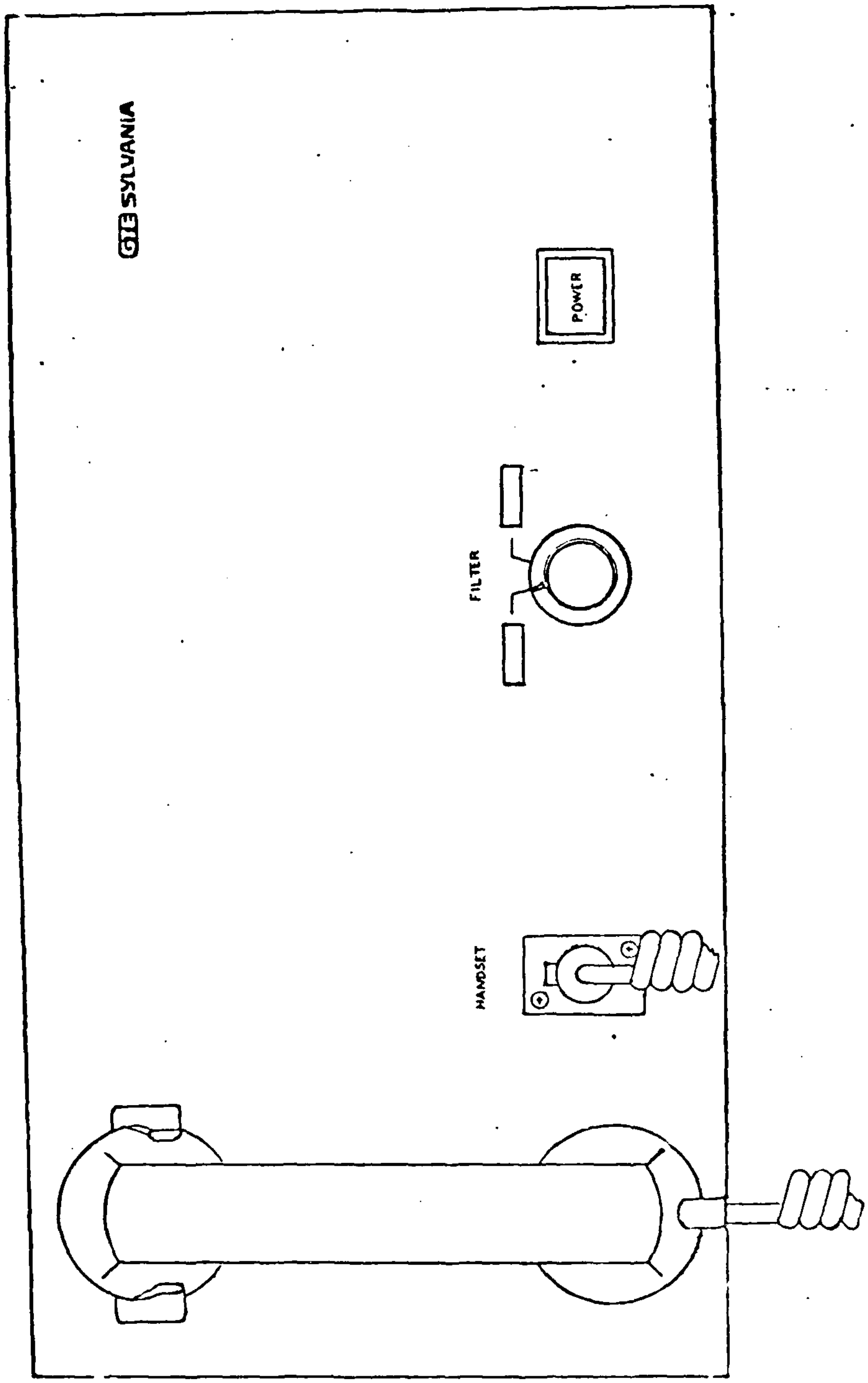
Upon receiving interrupt number 1, the processor should perform an input data transfer, acknowledging the interrupt after the transfer is complete. The input data word will contain the current modem data sample (least significant bit), RS-449 interface status data (bits 17-22), and a hookswitch status bit (bit 23). The receive interrupts are timed by the modem receive data clock; hence, the maximum response time to input the data and acknowledge the interrupt is approximately one-half of a period at the modem data rate. (The one-half period results from the fact that the processor must also supply transmit data.)

Upon receiving interrupt number 2, the processor should perform an output data transfer, acknowledging the interrupt after the transfer is complete. The output data word must contain the next data bit in the least significant bit position, and the appropriate RS-449 interface control bits in bit positions 17-19. The transmit interrupts are timed by the transmit data clock; hence, the maximum response time to output the data and acknowledge the interrupt is approximately one-half of a period at the transmit data rate. In order to insure the correct timing of the transmit data at the modem interface, the output data is double buffered.

Hence, while the MAP-300 is transferring transmit data bit n, data bit n-l appears at the modem interface.

The audio interface will be tested according to the attached table. These tests will insure that all amplifiers and filters are performing to the designed specifications. The loop tests to be performed for both the handset and tape audio paths will insure that the entire audio interface functions properly with the MAP-300, and that the overall loop gains and frequency response requirements are met.

The real time clock portion of the modified IOS-2SM scroll will be tested by programming several different line and speech sampling rates. A sufficient number of possible rates will be programmed to insure proper functioning of all RTC control bits. The modem interface will be tested by first outputting various control and data bit patterns from the MAP-300, and insuring that the correct bit patterns appear at the modem interface connector. Once the output has been determined to function properly, the modem input will be tested by inputting various control and data bit patterns, looping through the MAP-300, and observing the bit patterns at the interface connector. A digital loop test will then be performed whereby the MAP-300 will output a digital data stream which will be looped back at the modem interface connector and fed back into the processor where the resulting input will be compared to the original output.

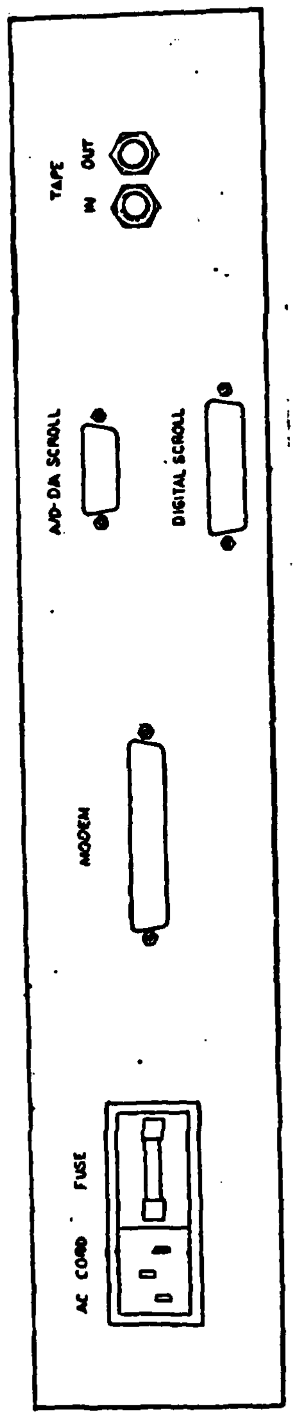


FRONT PANEL

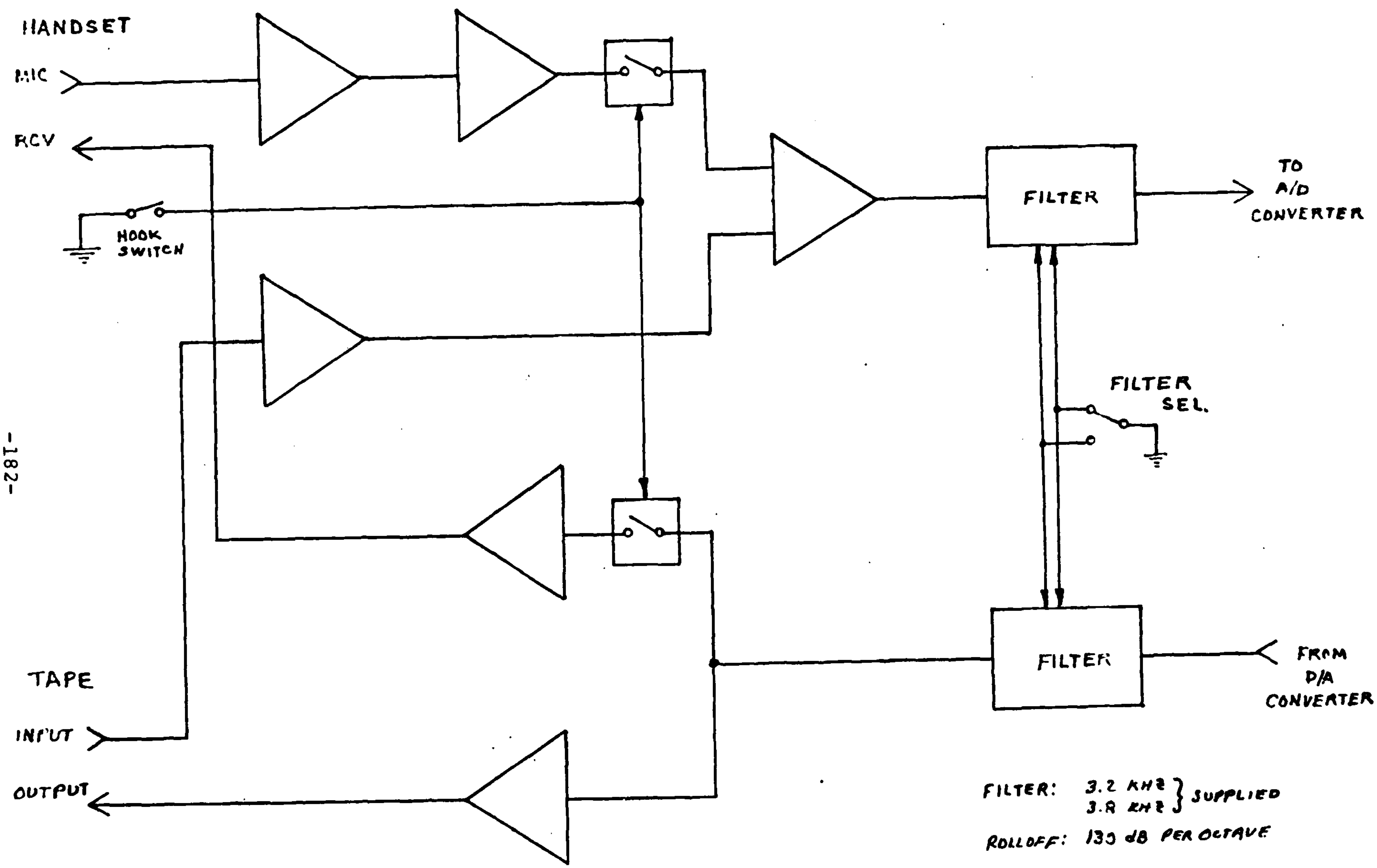
AD-A083 079 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2
DESIGN AND REAL-TIME IMPLEMENTATION OF A BASEBAND LPC CODER FOR--ETC(U)
FEB 80 R VISWANATHAN, J WOLF, L COSELL DCA100-79-C-0003
UNCLASSIFIED BBN-4327-VOL-1 NL

3 of 3
AD-
A083079

END
DATE
FILMED
5-80
DTIC



REAR PANEL



AUDIO INTERFACE - BLOCK DIAGRAM

TABLE 1 - AUDIO INTERFACE SPECIFICATIONS

<u>Handset Interface</u>

Input impedance	150 ohms, balanced
Output impedance	150 ohms, balanced
Gain, input to A/D converter	61 dB
Gain, D/A converter to output	-12 dB

<u>Tape Recorder Interface</u>

Input impedance	600 ohms, balanced
Output impedance	600 ohms, balanced
Gain, input to A/D converter	0 dB
Gain, D/A converter to output	0 dB

<u>MAP-300 Interface</u>

Output impedance - A/D converter	3000 ohms
Signal level - A/D converter	±5 V pk-pk
Input impedance - D/A converter	3000 ohms
Signal level - D/A converter	±5 V pk-pk

<u>Filter Characteristics</u>

Cutoff frequency	3.2 KHz / 3.8 KHz (switch selectable)
Rolloff	130 dB per octave

31–24 (MSB)	23–16 (LSB)
TRANSMIT DATA CLOCK	SAMPLE (SPEECH) CLOCK

RTC FREQUENCY CONTROL WORD FORMAT (CONTROL)

31–24 (MSB)	23							16 (LSB)
NOT USED	OFF HOOK	SI	IC	CS	DM	RR	SØ	RD

INPUT DATA WORD FORMAT

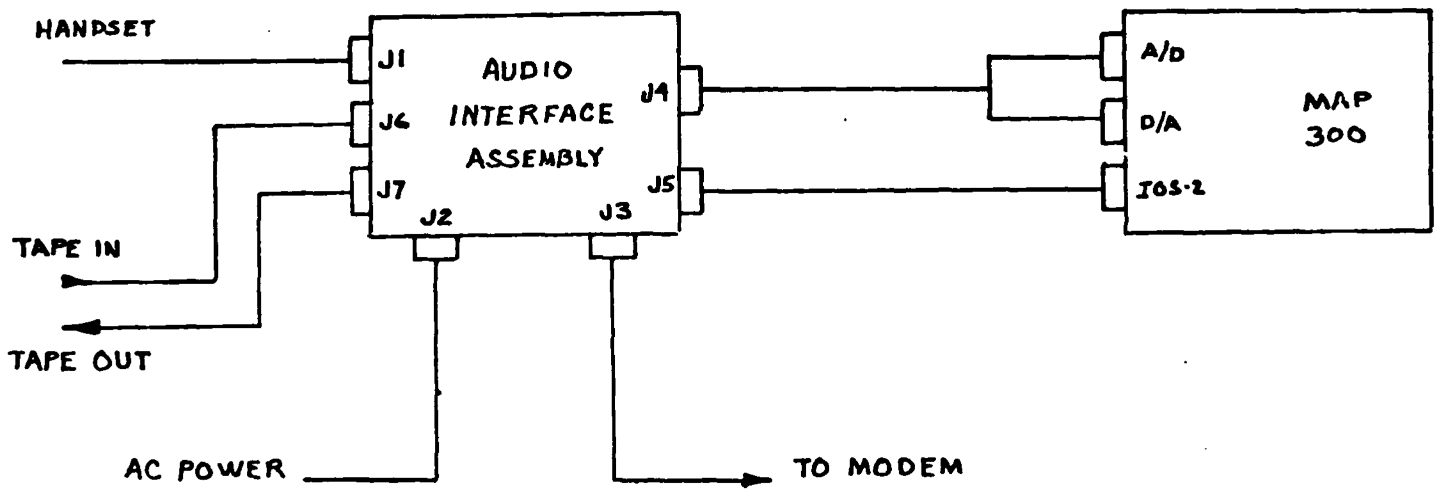
31–20 (MSB)	19			16 (LSB)
NOT USED	TR	RS	SR	SD

OUTPUT DATA WORD FORMAT

J3 - MODEM INTERFACE CONNECTIONS (RS-449)

PIN	SIGNAL
2	SI - Signal Rate Indicator
4	SD - Send Data
6	RD - Receive Data
7	RS - Request to Send
8	RT - Receive Timing
9	CS - Clear to Send
11	DM - Data Mode
12	TR - Terminal Ready
~~12~~ 13	RR - Receiver Ready
15	IC - Incoming Call
16	SR - Signal Rate Selector
17	TT - Terminal Timing
19	SG - Signal Ground
20	RC - Receive Common
33	SQ - Signal Quality
37	SC - Send Common

L. Bergeron 9/6/79



INTERCONNECTION DIAGRAM

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING DECIMAL	HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
0	00	00000000	256	0.750	1.500
1	01	00000001	255	0.753	1.506
2	02	00000010	254	0.756	1.512
3	03	00000011	253	0.759	1.518
4	04	00000100	252	0.762	1.524
5	05	00000101	251	0.765	1.530
6	06	00000110	250	0.768	1.536
7	07	00000111	249	0.771	1.542
8	08	00001000	248	0.774	1.548
9	09	00001001	247	0.777	1.555
10	0A	00001010	246	0.780	1.561
11	0B	00001011	245	0.784	1.567
12	0C	00001100	244	0.787	1.574
13	0D	00001101	243	0.790	1.580
14	0E	00001110	242	0.793	1.587
15	0F	00001111	241	0.797	1.593
16	10	00010000	240	0.800	1.600
17	11	00010001	239	0.803	1.607
18	12	00010010	238	0.807	1.613
19	13	00010011	237	0.810	1.620
20	14	00010100	236	0.814	1.627
21	15	00010101	235	0.817	1.634
22	16	00010110	234	0.821	1.641
23	17	00010111	233	0.824	1.648
24	18	00011000	232	0.828	1.655
25	19	00011001	231	0.831	1.662
26	1A	00011010	230	0.835	1.670
27	1B	00011011	229	0.838	1.677
28	1C	00011100	228	0.842	1.684
29	1D	00011101	227	0.846	1.692
30	1E	00011110	226	0.850	1.699
31	1F	00011111	225	0.853	1.707
32	20	00100000	224	0.857	1.714
33	21	00100001	223	0.861	1.722
34	22	00100010	222	0.865	1.730
35	23	00100011	221	0.869	1.738
36	24	00100100	220	0.873	1.745
37	25	00100101	219	0.877	1.753
38	26	00100110	218	0.881	1.761
39	27	00100111	217	0.885	1.770
40	28	00101000	216	0.889	1.778
41	29	00101001	215	0.893	1.786

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING DECIMAL	COUNTER SETTING HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	OUTPUT RATE - KHZ SPEECH
42	2A	00101010	214	0.897	1.794
43	2B	00101011	213	0.901	1.803
44	2C	00101100	212	0.906	1.811
45	2D	00101101	211	0.910	1.820
46	2E	00101110	210	0.914	1.829
47	2F	00101111	209	0.919	1.837
48	30	00110000	208	0.923	1.846
49	31	00110001	207	0.928	1.855
50	32	00110010	206	0.932	1.864
51	33	00110011	205	0.937	1.873
52	34	00110100	204	0.941	1.882
53	35	00110101	203	0.946	1.892
54	36	00110110	202	0.950	1.901
55	37	00110111	201	0.955	1.910
56	38	00111000	200	0.960	1.920
57	39	00111001	199	0.965	1.930
58	3A	00111010	198	0.970	1.939
59	3B	00111011	197	0.975	1.949
60	3C	00111100	196	0.980	1.959
61	3D	00111101	195	0.985	1.969
62	3E	00111110	194	0.990	1.979
63	3F	00111111	193	0.995	1.990
64	40	01000000	192	1.000	2.000
65	41	01000001	191	1.005	2.010
66	42	01000010	190	1.011	2.021
67	43	01000011	189	1.016	2.032
68	44	01000100	188	1.021	2.043
69	45	01000101	187	1.027	2.053
70	46	01000110	186	1.032	2.065
71	47	01000111	185	1.038	2.076
72	48	01001000	184	1.043	2.087
73	49	01001001	183	1.049	2.098
74	4A	01001010	182	1.055	2.110
75	4B	01001011	181	1.061	2.122
76	4C	01001100	180	1.067	2.133
77	4D	01001101	179	1.073	2.145
78	4E	01001110	178	1.079	2.157
79	4F	01001111	177	1.085	2.169
80	50	01010000	176	1.091	2.182
81	51	01010001	175	1.097	2.194
82	52	01010010	174	1.103	2.207
83	53	01010011	173	1.110	2.220

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

DECIMAL	COUNTER SETTING HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
84	54	01010100	172	1.116	2.233
85	55	01010101	171	1.123	2.246
86	56	01010110	170	1.129	2.259
87	57	01010111	169	1.136	2.272
88	58	01011000	168	1.143	2.286
89	59	01011001	167	1.150	2.299
90	5A	01011010	166	1.157	2.313
91	5B	01011011	165	1.164	2.327
92	5C	01011100	164	1.171	2.341
93	5D	01011101	163	1.178	2.356
94	5E	01011110	162	1.185	2.370
95	5F	01011111	161	1.193	2.385
96	60	01100000	160	1.200	2.400
97	61	01100001	159	1.208	2.415
98	62	01100010	158	1.215	2.430
99	63	01100011	157	1.223	2.446
100	64	01100100	156	1.231	2.462
101	65	01100101	155	1.239	2.477
102	66	01100110	154	1.247	2.494
103	67	01100111	153	1.255	2.510
104	68	01101000	152	1.263	2.526
105	69	01101001	151	1.272	2.543
106	6A	01101010	150	1.280	2.560
107	6B	01101011	149	1.289	2.577
108	6C	01101100	148	1.297	2.595
109	6D	01101101	147	1.306	2.612
110	6E	01101110	146	1.315	2.630
111	6F	01101111	145	1.324	2.648
112	70	01110000	144	1.333	2.667
113	71	01110001	143	1.343	2.685
114	72	01110010	142	1.352	2.704
115	73	01110011	141	1.362	2.723
116	74	01110100	140	1.371	2.743
117	75	01110101	139	1.381	2.763
118	76	01110110	138	1.391	2.783
119	77	01110111	137	1.401	2.803
120	78	01111000	136	1.412	2.824
121	79	01111001	135	1.422	2.844
122	7A	01111010	134	1.433	2.866
123	7B	01111011	133	1.444	2.887
124	7C	01111100	132	1.455	2.909
125	7D	01111101	131	1.466	2.931

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING DECIMAL	HEXADECIMAL	BINARY	DIVIDE RATIO	OUTPUT RATE - KHZ LINE	SPEECH
126	7E	01111110	130	1.477	2.954
127	7F	01111111	129	1.488	2.977
128	80	10000000	128	1.500	3.000
129	81	10000001	127	1.512	3.024
130	82	10000010	126	1.524	3.048
131	83	10000011	125	1.536	3.072
132	84	10000100	124	1.548	3.097
133	85	10000101	123	1.561	3.122
134	86	10000110	122	1.574	3.148
135	87	10000111	121	1.587	3.174
136	88	10001000	120	1.600	3.200
137	89	10001001	119	1.613	3.227
138	8A	10001010	118	1.627	3.254
139	8B	10001011	117	1.641	3.282
140	8C	10001100	116	1.655	3.310
141	8D	10001101	115	1.670	3.339
142	8E	10001110	114	1.684	3.368
143	8F	10001111	113	1.699	3.398
144	90	10010000	112	1.714	3.429
145	91	10010001	111	1.730	3.459
146	92	10010010	110	1.745	3.491
147	93	10010011	109	1.761	3.523
148	94	10010100	108	1.778	3.556
149	95	10010101	107	1.794	3.589
150	96	10010110	106	1.811	3.623
151	97	10010111	105	1.829	3.657
152	98	10011000	104	1.846	3.692
153	99	10011001	103	1.864	3.728
154	9A	10011010	102	1.882	3.765
155	9B	10011011	101	1.901	3.802
156	9C	10011100	100	1.920	3.840
157	9D	10011101	99	1.939	3.879
158	9E	10011110	98	1.959	3.918
159	9F	10011111	97	1.979	3.959
160	A0	10100000	96	2.000	4.000
161	A1	10100001	95	2.021	4.042
162	A2	10100010	94	2.043	4.085
163	A3	10100011	93	2.065	4.129
164	A4	10100100	92	2.087	4.174
165	A5	10100101	91	2.110	4.220
166	A6	10100110	90	2.133	4.267
167	A7	10100111	89	2.157	4.315

PROGRAMMABLE OSCILLATOR OUTPUT RATES

OSCILLATOR FREQUENCY = 1.536 MHZ

	COUNTER SETTING		DIVIDE	OUTPUT RATE - KHZ	
DECIMAL	HEXADECIMAL	BINARY	RATIO	LINE	SPEECH
168	A8	10101000	88	2.182	4.364
169	A9	10101001	87	2.207	4.414
170	AA	10101010	86	2.233	4.465
171	AB	10101011	85	2.259	4.518
172	AC	10101100	84	2.286	4.571
173	AD	10101101	83	2.313	4.627
174	AE	10101110	82	2.341	4.683
175	AF	10101111	81	2.370	4.741
176	B0	10110000	80	2.400	4.800
177	B1	10110001	79	2.430	4.861
178	B2	10110010	78	2.462	4.923
179	B3	10110011	77	2.494	4.987
180	B4	10110100	76	2.526	5.053
181	B5	10110101	75	2.560	5.120
182	B6	10110110	74	2.595	5.189
183	B7	10110111	73	2.630	5.260
184	B8	10111000	72	2.667	5.333
185	B9	10111001	71	2.704	5.408
186	BA	10111010	70	2.743	5.486
187	BB	10111011	69	2.783	5.565
188	BC	10111100	68	2.824	5.647
189	BD	10111101	67	2.866	5.731
190	BE	10111110	66	2.909	5.818
191	BF	10111111	65	2.954	5.908
192	C0	11000000	64	3.000	6.000
193	C1	11000001	63	3.048	6.095
194	C2	11000010	62	3.097	6.194
195	C3	11000011	61	3.148	6.295
196	C4	11000100	60	3.200	6.400
197	C5	11000101	59	3.254	6.508
198	C6	11000110	58	3.310	6.621
199	C7	11000111	57	3.368	6.737
200	C8	11001000	56	3.429	6.857
201	C9	11001001	55	3.491	6.982
202	CA	11001010	54	3.556	7.111
203	CB	11001011	53	3.623	7.245
204	CC	11001100	52	3.692	7.385
205	CD	11001101	51	3.765	7.529
206	CE	11001110	50	3.840	7.680
207	CF	11001111	49	3.918	7.837
208	D0	11010000	48	4.000	8.000
209	D1	11010001	47	4.085	8.170

OSCILLATOR FREQUENCY = 1.536 MHZ

COUNTER SETTING			DIVIDE	OUTPUT RATE - KHZ	
DECIMAL	HEXADECIMAL	BINARY	RATIO	LINE	SPEECH
210	D2	11010010	46	4.174	8.348
211	D3	11010011	45	4.267	8.533
212	D4	11010100	44	4.364	8.727
213	D5	11010101	43	4.465	8.930
214	D6	11010110	42	4.571	9.143
215	D7	11010111	41	4.683	9.366
216	D8	11011000	40	4.800	9.600
217	D9	11011001	39	4.923	9.846
218	DA	11011010	38	5.053	10.105
219	DB	11011011	37	5.189	10.378
220	DC	11011100	36	5.333	10.667
221	DD	11011101	35	5.486	10.971
222	DE	11011110	34	5.647	11.294
223	DF	11011111	33	5.818	11.636
224	E0	11100000	32	6.000	12.000
225	E1	11100001	31	6.194	12.387
226	E2	11100010	30	6.400	12.800
227	E3	11100011	29	6.621	13.241
228	E4	11100100	28	6.857	13.714
229	E5	11100101	27	7.111	14.222
230	E6	11100110	26	7.385	14.769
231	E7	11100111	25	7.680	15.360
232	E8	11101000	24	8.000	16.000
233	E9	11101001	23	8.348	16.696
234	EA	11101010	22	8.727	17.455
235	EB	11101011	21	9.143	18.286
236	EC	11101100	20	9.600	19.200
237	ED	11101101	19	10.105	20.211
238	EE	11101110	18	10.667	21.333
239	EF	11101111	17	11.294	22.588
240	F0	11110000	16	12.000	24.000
241	F1	11110001	15	12.800	25.600
242	F2	11110010	14	13.714	27.429
243	F3	11110011	13	14.769	29.538
244	F4	11110100	12	16.000	32.000
245	F5	11110101	11	17.455	34.909
246	F6	11110110	10	19.200	38.400
247	F7	11110111	9	21.333	42.667
248	F8	11111000	8	24.000	48.000
249	F9	11111001	7	27.429	54.857
250	FA	11111010	6	32.000	64.000
251	FB	11111011	5	38.400	76.800

COUNTER SETTING			DIVIDE	OUTPUT RATE - KHZ	
DECIMAL	HEXADECIMAL	BINARY	RATIO	LINE	SPEECH
252	FC	11111100	4	48.000	96.000
253	FD	11111101	3	64.000	128.000
254	FE	11111110	2	96.000	192.000
255	FF	11111111	1	192.000	384.000

GTE SYLVANIA

Electronic Systems Group
Eastern Division
77 "A" Street
Needham Heights, Mass. 02194
617 449-2000

March 19, 1979

Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA

Attention: Mr. Jared Wolf

Subject: 9600 BPS Optimization Study

Reference: GTE Sylvania letter JFM-79-119 dated 12 March 1979

Enclosure: "IOS-2/Peripheral Communication" Photostat from C.S.P.I. Document No. JB4020-000-02

Gentlemen:

Per reference letter, GTE Sylvania, Inc. submitted to you the equipment description for the MAP-300 speech interface unit for subject program. As an addendum to that material, we are submitting the enclosed document entitled IOS-2/Peripheral Communication. As stated in this document, flags P1 and P2 can be used for IOS-2 program control conditioned by peripheral requests. Our digital logic for the speech interface unit will use these flags to dictate transmit/receive requests. The flags will be set when each digital I/O clock turns over and will be reset when the 16-bit address/control data lines are exercised (read/write). Since no interrupt lines will be tied to the CSPU, it is your responsibility to handle these flags so as to maintain real time synchronism and framing of the digital data.

This change in design will neither impact the cost nor delivery time of the speech interface units.

Please refer any questions to Mr. Larry Bergeron, Project Engineer, or Mr. John McGowan, Contracts, at (617) 449-2000, extensions 3526 or 2009, respectively.

Very truly yours,

L. E. Bergeron
Project Engineer

db

APPENDIX B: BBN-WRITTEN SNAP-II FUNCTION DESCRIPTIONS

This Appendix contains a complete user-level description for each MAP-300 SNAP-II function written by BBN and used in the speech coder implementation. The descriptions are ordered alphabetically by function name.

Functions which were used in the speech coder implementation and are not described here were supplied by CSPI as part of Release 3.5 of the SNAP-II Software System(see Section 3.4.1.1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: CORECT(I)
NAME EXPANSION: Perform Error Correction

FCB #: 122
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: CORRECT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal I	-2,0	Integer	I=-2=>Buffer A I=0 =>Buffer B

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Takes as input one of the RBITS buffers containing a frame of bitstream data from the modem and produces as output an RSOURCE buffer containing error-corrected and decoded floating point values of analysis/synthesis parameters and baseband residual samples.

The argument I specifies which pair of RBITS/RSOURCE buffers are used. I=-2 means RBTA/RSRA, and I=0 means RBTB/RSRB.

The tables used for decoding each parameter and the baseband residual begin at location $5D00. Their definitions are at the beginning of the file BBN300.MSO.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DCOR(Y,U,V)
NAME EXPANSION: Discrete Correlation

FCB #: 191
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DCRU$
APS MODULE NAME: DCRS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Correlation
Buffer U	1-63	Real	Input: Reference, Kernal
Buffer V	1-63	Real	Input

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: YBS must equal VBS -UBS+1 for correct operation.

FUNCTION DESCRIPTION:

$$Y(m) = \sum_{K=0}^{UBS-1} U(K) + V(K+M) \qquad 0 \le m \le VBS-UBS+1$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DEAL(Y,A,U,B,V)
NAME EXPANSION: Deal Buffer Contents to Buffers and Scalars

FCB #: 190
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DEALU$
APS MODULE NAME: DEALS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	real	Output: baseband samples *gain
Real Scalar A	0-191	real	Output: received pitch
Buffer U	1-63	real(8)	Output: received reflection coeffs (K's)
Real Scalar B	0-191	real	Output: received tap
Buffer V	1-63	real	Input

NUMBER OF OUTPUT SAMPLES: YBS + 1 + 8 + 1
NOTES:

FUNCTION DESCRIPTION:

Takes decoded values from V buffer, in order pitch, tap, gain, K(1)-K(8), BB(1)-BB(YBS), and moves them to more useful positions. Also, multiplies the baseband samples by gain, which includes the received gain as well as the gain for the Butterworth filter to be used later.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ENRG(A,B,C,W,D)
NAME EXPANSION: Compute, Code & Quantize Energy (Gain)

FCB #: 199
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: ENU$
APS MODULE NAME: ENS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Real Scalar A	0-191	Real	Output: Quantized Gain
Real Scalar B	0-191	Real	Output: Inverse of Quantized Gain
Real Scalar C	0-191	Long Fixed (in left halfword of Real Scalar)	Output: Coded Gain
Buffer W	1-63	Real	Input: Baseband with Pitch Removed
Real Scalar D	0-191	Real	Input: 1/WBS (where WBS is 'W' Buffer Size)

NUMBER OF OUTPUT SAMPLES: 3 Output Scalars
NOTES:

FUNCTION DESCRIPTION:

First, the energy is computed by summing the squares of the samples in Buffer W, and dividing by the number of samples: energy = D * SUM(W(I)**2). Then, the energy is coded and quantized by comparing it linearly to a quantization threshold table, stored internally. Coded gain (gain = sqrt(energy)) is generated by adding an increment (equivalent to fixed point 1 in left halfword of 32-bit fullword) to an accumulator for each threshold table element searched. Quantized gain is generated from an internal gain quantization value table accessed in parallel with the energy threshold table. Inverse of quantized gain is similarly generated from an internal inverse gain quantization value table accessed in parallel with the energy threshold table.

Hardware flags WI and FWI are used throughout for APU/APS synchronization.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: IADINT
NAME EXPANSION: Simulate A/D Interrupt

FCB #: 124
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: ADAMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the ADAMINT A/D Interrupt Service module in exactly the same manner as it is used to respond to A/D interrupts, so it may be used to simulate such an interrupt. See Sec. 3.2.1.2.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: IDAINT
NAME EXPANSION: Simulate D/A Interrupt

FCB #: 127
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: AOMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS

None

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the AOMINT D/A Interrupt Service module, so it may be used to simulate such an interrupt. See Sec. 3.2.2.5.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: IRMINT
NAME EXPANSION: Simulate Receiver Modem Interrupt

FCB #: 126
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: RMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the RMODEMINT Receiver Modem Interrupt Service module, so it may be used to simulate such an interrupt. See Sec. 3.2.2.2.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ITMINT
NAME EXPANSION: Simulate Transmitter Modem Interrupt

FCB #: 125
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: TMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS

None

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the TMODEMINT Transmitter Modem Interrupt Service module, so it may be used to simulate such an interrupt. See Sec. 3.2.1.5.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPGSC(GFLAG,SETCLR)
NAME EXPANSION: G-Flag Set/Clear

FCB #: 123
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPGSC

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal GFLAG	0-3	Integer	Selects G-flag
Literal SETCLR	0-1	Integer	SETCLR=0=>Set flag SETCLR=1=>Clear flag

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

MPGSC sets or clears one of the four G-flags. This is useful for program timing using an external oscilloscope.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPIFF(IA,IB, FLID)
NAME EXPANSION: If (IA.NE.0)& (IB.EQ.0) Conditional Function List Execution

FCB #: 105
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPIFF$

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Integer Scalar IA	0-127	Integer	Input
Integer Scalar IB	0-127	Integer	Input
Literal FLID	0-63	Integer	Input: Function List ID

NUMBER OF OUTPUT SAMPLES: -
NOTES: Function list 'FLID' must be previously defined.

FUNCTION DESCRIPTION:

Function list 'FLID' is executed if and only if Integer Scalar IA is not equal to zero, and Integer Scalar IB is equal to zero.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPMBS(Y,A,N)
NAME EXPANSION: Move Buffer to Scalar

FCB #: 111
ARRAY OR NON-ARRAY FUNCTION: Non-Array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPMBS$

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Input
Real Scalar A	0-191	Real	ID of First Output Scalar
Literal N	1-(191-A)	Integer	Input: Number of Samples to move

NUMBER OF OUTPUT SAMPLES: N
NOTES: -

FUNCTION DESCRIPTION:

'N' samples are moved from Buffer Y to the Real Scalar Table, beginning with Real Scalar A. This function differs from MPTBS in that the Buffer Y base address is not changed.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPXBM(FCBNO,Y,A,U,V)
NAME EXPANSION: Execute Bound Version of MWLF

FCB #: -
ARRAY OR NON-ARRAY FUNCTION: (Host Support Only)

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal FCBNO	0-255	Integer	Input: FCB Number of prebound MWLF.
Buffer Y	Same as parameters to MWLF		
Real Scalar A			
Buffer U			
Buffer V			

NUMBER OF OUTPUT SAMPLES: -
NOTES: -

FUNCTION DESCRIPTION:

The pre-bound version of MWLF, residing at FCB number 'FCBNO', is executed. This function differs from MPXBF in that the buffer and scalar ID parameters are inserted into the function control block.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MWLF(Y,A,U,V)
NAME EXPANSION: Matrix(Weiner-Levinson-Durbin) Solution, With Quantized & Coded Output
FCB #: 135
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: MWLF$APU
APS MODULE NAME: MWLF$APS
CSPU MODULE NAME: MWLQ$SSM

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	real	Reflection coeff output
Real Scalar A	0-191	real	Threshold value
Buffer U	1-63	real	Coded, quantized reflection coeffs, multiplexed as 16 bit fixed, short float numbers in two halves of real element
Buffer V	1-63	real	Autocorrelation input

NUMBER OF OUTPUT SAMPLES: 8
NOTES: Special intermediate CSPU support used.

FUNCTION DESCRIPTION:

Obtain solution of Linear prediction matrix equation using Weiner-Levinson-Durbin method. Quantize and code 8 resulting reflection coefficients. After solution is found, special support causes APS portion of module to be modified and whole module continues to perform quantization and coding. The coded outputs are stored as integers in the first half of each real sample in the U array. The quantized values are stored as short floating point numbers in the second half of each real sample in the U array.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PROTCT (I)
NAME EXPANSION: Perform Error Protection

FCB #: 121
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: PROTECT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal I	-2,0	Integer	I=-2 => Buffer A I=0 => Buffer B

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

Takes as input one of the TSINK buffers containing quantized and coded analysis parameters and baseband residual samples and writes error-protected and bitstreamed data in one of the TBITS buffers. Also accumulates histogram statistics for the analysis parameters.

The argument I specifies which pair of TSINK/TBITS buffers are used. I=-2 means TSNKA/TBTA, and I=0 means TSNKB/TBTB.

See Section 3.2.1.4.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PRTRB(Y,A,U,B,V)
NAME EXPANSION: Upsample (3:1) With Perturbation

FCB #: 134
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PRTRB$
APS MODULE NAME: P2120$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Perturbed and Upsampled Data
Real Scalar A	0-191	Real	Input: Perturbation Threshold Constant
Buffer U	1-63	Real	Input: Downsampled Data
Real Scalar B	0-191	Real	Input: Perturbation Gain Factor
Buffer V	1-63	Real	Input: Sequence of Gausian Random Numbers with Mean 0, Standard Deviation 1

NUMBER OF OUTPUT SAMPLES: 3*UBS
NOTES: UBS≤VBS
3*UBS≤YBS

FUNCTION DESCRIPTION:

Buffer U is upsampled (3:1), with each data sample perturbed positionally in the upsampled output (Buffer Y) by -1,0, or +1 depending on the values in random buffer V. The function is described as follows:

```
FOR I=0,(UBS-1):
    Y(3I), Y(3I+1), Y(3I+2) =

    U(I),0,0     IF   WEIGHT<-.5
    0,U(I),0     IF           -.5 ≤ WEIGHT<+.5
    0,0,U(I)     IF                   +.5 ≤ WEIGHT

WHERE:  WEIGHT = {V(I)*MAX[0,(SA+(SB*ABS(U(I))))]}
```

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PTAP(Y,A,U,B,V,C,W)
NAME EXPANSION: Compute Pitch and Tap and Do Pitch Removal

FCB #: 212
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PTU$
APS MODULE NAME: PTS$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Baseband with Pitch Removed
Real Scalar A	0-191	Real	Output: Pitch (Index of maximum Baseband Excitation Autocorrelation)
Buffer U	1-63	Real	Input: Autocorrelation of Baseband Excitation-Pitch Calculation Part (W(5)-W(38))
Real Scalar B	0-191	Real	Output: Quantized Tap
Buffer V	1-63	Real	Input: Downsampled Baseband Excitation
Real Scalar C	0-191	Long Fixed*	Output: Coded Tap
Buffer W	1-63	Real	Input: Autocorrelation of Baseband Excitation - whole thing

*(in left halfword of real scalar)

NUMBER OF OUTPUT SAMPLES: YBS Samples Output to Buffer Y, 3 Output Scalars
NOTES: YBS=VBS; U(0)-U(33)=W(5)-W(38); UBS=34
APS input program is modified using value calculated in APU program (pitch).

FUNCTION DESCRIPTION:

Pitch (defined to be the index of the maximum W(I) for I=5 to I=38) is computed by adding 5 to the index of the maximum U buffer element. This pitch is then converted to a 16-bit fixed point value, and is written into the APS input program for its future use in generating pitch-offset V buffer element addresses.

Tap (defined to be max(U(I))/W(0)) is computed and then coded and quantized by comparing it linearly to a quantization threshold table, stored internally. Coded tap is generated by adding an increment (equivalent to fixed point 1 in left halfword of 32-bit fullword) to an accumulator for each threshold table element searched. Quantized tap is generated from an internal quantization value table accessed in parallel with the threshold table.

Finally, pitch removal is accomplished according to the following equation: Y(I) = V(I) - QTAP*V(I-PITCH) (where QTAP is quantized tap).

Hardware flags WI and FWI are used throughout for APU/APS synchronization.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VAPC(Y,A,U,B,V,C,D)
NAME EXPANSION: APC

FCB #: 150
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VAPC$
APS MODULE NAME: AAPC$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Integer	Output
Real Scalar A	0-191	Real	Pitch (in) (1)
Buffer U	1-63	Real	Quantization error (in/out)
Real Scalar B	0-191	Real	Quantized tap (in) (3)
Buffer V	1-63	Real	Input Residual
Real Scalar C	0-191	Integer	Coded tap (in) (10)
Real Scalar D	0-191	Integer	Baseband quantization information (7)

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: This routine modifies its APS code. Note that B,C,D, are each collections of contiguous scalars.

FUNCTION DESCRIPTION:

Generate buffer of coded parameters and residual samples to be protected. Perform adaptive predictive coding on residual samples, and code output values.

B scalar is beginning of 3 scalars: quantized tap, quantized gain, and the inverse of quantized gain. C is beginning of 10 scalars: coded tap, coded gain, and 8 coded reflection coefficients. D is beginning of 7 scalars: 3 baseband quantization thresholds and 4 quantized values.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VIAPC(Y,A,U,B)
NAME EXPANSION: Inverse APC

FCB #: 196
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: APCIU$
APS MODULE NAME: APCIS$
CSPU MODULE NAME: ---

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output residual samples
Real Scalar A	0-191	Real	Pitch
Buffer U	1-63	Real	Decoded residual samples
Real Scalar B	0-191	Real	Tap

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: This routine modifies its APU code.

FUNCTION DESCRIPTION:

Y(n)=Y(n-YBS)=X(n)+tap*Y(n-pitch)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VKTOA(Y,U)
NAME EXPANSION: Convert Reflection Coefficients to Linear Predictor Coefficients

FCB #: 133
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VKTOA$
APS MODULE NAME: V1100K$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: 9 Linear Prediction Coefficients
Buffer U	1-63	Real	Input: 8 Reflection Coefficients

NUMBER OF OUTPUT SAMPLES: 9
NOTES: Number of input samples and output samples fixed.

FUNCTION DESCRIPTION:

The recursion equations for the conversion are:

$$A_m(m) = K(m)$$

$$A_m(L) = A_{m-1}(L) + K_m \cdot A_{m-1}(m-L) \qquad \begin{matrix} 1 \le m \le 8 \\ 1 \le L < m \end{matrix}$$

where subscripts indicate iteration, A is LP coefficient and K is reflection coefficient, and A(0) = 1.0

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VLTSY(Y,U,V,W)
NAME EXPANSION: Lattice Synthesis Filter

FCB #: 132
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VLTSY$
APS MODULE NAME: V3200$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output synthetic speech samples
Buffer U	1-63	Real(8)	Filter memory (G's)
Buffer V	1-63	Real(8)	Reflection coefficients (K's)
Buffer W	1-63	Real	Input samples

NUMBER OF OUTPUT SAMPLES: WBS
NOTES: Exactly 8 elements from each U and V are used. Contents of U are changed.

FUNCTION DESCRIPTION:

Performs lattice form all-pole filter, using reflection coefficients as filter coefficients.

Implements the following Fortran code:

```
        DO 20 I=1, WBS
        F(7)=W(I)-G(7)*K(8)
        DO 10 J=6,0, -1
        F(J)=F(J+1)-G(J)*K(J+1)
10      G(J+1)=G(J)+F(J)*K(J+1)
        G(0)=F(0)
20      Y(I)=F(0)
```

APPENDIX C: MAP-300 BUFFERS

The table contained in this Appendix describes the characteristics of each data buffer used in the MAP-300 speech coder implementation. The table entry labeled "BID" indicates the buffer identification number (for SNAP-accessable buffers) or is blank (for non-SNAP buffers). The entries labeled "Written by:" and "Read by:" indicate subroutines, function lists, or program modules which so access the buffer. (The A/D, D/A, transmitter modem, and receiver modem interrupt routines are specified in this table by "IADINT", "IDAINT", "ITMINT", and "IRMINT", respectively.)

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TBTA	-	TBITS A Buffer	-	PROTCT	ITMINT	1	LNG FXD	261	42842	1	-	2
TBTB	-	TBITS B Buffer	-	PROTCT	ITMINT	1	LNG FXD	261	43104	1	-	2
TBTC	-	TBITS C Buffer	-	DCA96I	ITMINT	1	LNG FXD	261	43366	1	BITSTREAM EQUIVALENT OF "SILENCE" (CODED ENERGY = 0)	1
RBTA	-	RBITS A Buffer	-	IRMINT	CORECT	1	LNG FXD	261	43628	1	-	2
RBTB	-	RBITS B Buffer	-	IRMINT	CORECT	1	LNG FXD	261	43890	1	-	2
TADBA	-	A/D Input from ADAM	-	ADPROG (ADAM Scroll Program)	IADINT	1	SHRT FLT	180	44152	1	-	2
TADBB	-	A/D Input from ADAM	-	ADPROG (ADAM Scroll Program)	IADINT	1	SHRT FLT	180	44332	1	-	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TADRC	-	A/D "On-Hook" Tone Data	-	DCA96I	IADINT	1	SHRT FLT	180	44512	1	Approx. 225 Hz square wave (6 cycles per buffer)	1
TSRA	1	TSOURCE A Buffer	-	IADINT	ANLZ (LPC)	1	SHRT FLT	180	44692	1	-	2
TSRB	2	TSOURCE B Buffer	-	IADINT	ANLZ (LPC)	1	SHRT FLT	180	44872	1	-	2
THAMW	3	Hamming Window Coefficients	-	DCA96I	ANLZ (LPC)	1	LNG FLT	180	240	2	Hamming Window Coefficients	1
TWSR	4	Windowed TSRx	←180→ ←8→ 0's ←TWSR→ ←TWSRZ→	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	180	794	3	(Zeros)	2
TWSRZ	6	TWSR with 8 zeros appended on right.	(see TWSR)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	188	794	3	Zeros	2
TACV	7	Autocorrelation Values	(identical to TLPC)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	9	1594	3	-	2

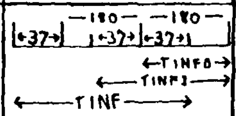
MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TRFER	8	Output from MWLF: Reflection Coefficients and Error Terms	←16→ |←8→|←8→| Reflection ←coefs→←Errors→ ←TRFER→	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	16	1170	3	-	2
TCQRF	9	Output from MWLF: Coded & Quantized Reflection coefficients (Interleaved TCRF, TQRF)	each sample (32 bits): ←16 bits→←16 bits→ ←TCRF→ ←TQRF→	ANLZ (LPC)	ANLZ (LPC) (APC)	1	LNG FLT	8	1202	3	zeros	2
TCRF	10	Coded Reflection Coefficients	(see TCQRF)	ANLZ (LPC)	ANLZ (LPC)	2	LNG FLT	8	1202	3	(zeros)	2
TQRF	11	Quantized Reflection Coefficients	(see TCQRF)	ANLZ (LPC)	ANLZ (LPC)	2	LNG FXD	8	1203	3	(zeros)	2
TLPC	(7)	Linear Prediction Coefficients	(identical to TACV)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	9	1594	3	-	2
TLPCR	5	Reverse Buffer of TLPC	(overlays TLPC)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	9	1594	3	-	2
TSR	13	Current TSOURCE x data in LNG FLT format	←180→ |←8→| |←8→| ←TSR→ ←TSRM→ ←TSRH→ ←TSRL→	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	180	1234	3	(zeros)	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TSRM	14	TSR with 8 memory samples in front	(see TSR)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	188	1218	3	zeros	2
TSRF	15	First 8 samples of TSRM	(see TSR)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	8	1218	3	(zeros)	2
TSRL	16	Last 8 samples of TSRM	(see TSR)	ANLZ (LPC)	ANLZ (LPC)	1	LNG FLT	8	1578	3	(zeros)	2
TINFO	17	Inverse Filtered Samples (current frame)		ANLZ (LPC)	ANLZ (BBEXT)	1	LNG FLT	180	434	3	(zeros)	2
TINF1	18	TINFO plus 37 previous frame elements in front	(see TINFO)	ANLZ (LPC)	ANLZ (BBEXT)	1	LNG FLT	217	360	3	zeros	2
TINF	19	254 Inverse Filtered Samples centered on previous frame	(see TINFO)	ANLZ (LPC)	ANLZ (BBEXT)	1	LNG FLT	254	0	3	-	2
TLPFB	20	LPF Coefficients to get Baseband Excitation	-	DCA96I	ANLZ (BBEXT)	1	LNG	75	1612	3	LPC coefficients for 75 point symmetric filter, passband edge at 925 Hz, stopband edge at 1111 Hz, stopband rejection of -35dB	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TBEO	21	Downsampled Baseband Excitation	-30-|-30-| — 60 — |—38— ←TBE1→←—TBEO—→←O's→ ←—TBE—→ ←—TBEZ—→	ANLZ (BBEXT)	ANLZ (APC	1	LNG FLT	60	1882	3	(zeros)	2
TBE1	22	Previous frame's TBEO	(see TBEO)	ANLZ (BBEXT)	ANLZ (BBEXT)	1	LNG FLT	60	1762	3	-	2
TBE	23	Last half of TBE1, all of TBEO	(see TBEO)	ANLZ (BBEXT)	ANLZ (PEDET)	1	LNG FLT	90	1822	3	(zeros)	2
TBEZ	24	TBE, plus 38 zeros	(see TBEO)	ANLZ (BBEXT)	ANLZ (PEDET)	1	LNG FLT	128	1822	3	zeros	2
TPAC	28	Baseband Excitation Pitch Autocorrelation Buffer	|-5-|←— 34 —→| ←- TBPCP -→ ←— TPAC —→	ANLZ (PEDET)	ANLZ (PEDET)	1	LNG FLT	39	2078	3	-	2
TBPCP	29	Baseband Excitation Autocorrelation: Pitch Calculation Part	(see TPAC)	ANLZ (PEDET)	ANLZ (PEDET)	1	LNG FLT	34	2088	3	-	2
TBPR	(40)	Baseband Excitation with Pitch Removed	(identical to RBR)	ANLZ (PEDET)	ANLZ (PEDET)	1	LNG FLT	60	736	2	-	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TBRDR	33	Baseband Residual Difference Buffer (Right Half)	|←60→|←60→| ←TBRDL→ ←TBRDR→	ANLZ (APC)	ANLZ (APC)	1	LNG FLT	60	120	2	zeros	2
TBRDL	-	Baseband Residual Difference Buffer (Left Half)	(see TBRDR)	ANLZ (APC)	ANLZ (APC)	1	LNG FLT	60	0	2	zeros	2
TSNKA	35	TSINK A Buffer	-	ANLZ (APC)	PROTCT	1	LNG FXD	71	45052	1	-	2
TSNKB	36	TSINK B Buffer	-	ANLZ (APC)	PROTCT	1	LNG FXD	71	45124	1	zeros	2
TSNKC	-	(Unused)	-	-	-	1	LNG FXD	71	45196	1	-	-
TMDMA	-	TMODEM A Buffer	-	ITMINT	RTMODEM (Modem Scroll Program)	1	LNG FXD	261	45268	1	Bitstream equivalent of "silence" (coded energy = 0), sync bit = 0	2
TMDMB	-	TMODEM B Buffer	-	ITMINT	RTMODEM (Modem Scroll Program)	1	LNG FXD	261	45530	1	Bitstream equivalent of "silence" (coded energy = 0), sync bit = 1	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RMDMA	-	RMODEM A Buffer	-	RTMODEM (Modem Scroll Program)	IRMINT	1	LNG FXD	261	45792	1	-	2
RMDMB	-	RMODEM B Buffer	-	RTMODEM (Modem Scroll Program)	IRMINT	1	LNG FXD	261	46054	1	-	2
RMDMC	-	RMODEM C Buffer	-	RTMODEM (Modem Scroll Program)	IRMINT	1	LNG FXD	261	46316	1	-	2
RSSPF	-	Sync Search: Previous Frame Buffer	-	IRMINT	IRMINT	1	LNG FXD	261	46578	1	-	2
RSSSS	-	Sync Search: Sum Sync Buffer	-	IRMINT	IRMINT	1	LNG FXD	261	46840	1	-	2
RSRA	37	RSOURCE A Buffer	-	CORECT	SYNZ (APCI)	1	LNG FLT	71	47102	1	-	2
RSRB	38	RSOURCE B Buffer	-	CORECT	SYNZ (APCI)	1	LNG FLT	71	47244	1	zeros	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RRFO	39	Current Frame Reflection Coefficients	-	SYNZ (APCI)	SYNZ (SNFL)	1	LNG FLT	8	600	2	-	2
RBR	40	Baseband Residual	-60- | -60- ←RBE→ ←RBR→ ←TBPR→	SYNZ (APCI)	SYNZ (APCI)	1	LNG FLT	60	736	2	-	2
RBE	41	Baseband Excitation	(see RBR)	SYNZ (APCI)	SYNZ (APCI) (HFR)	1	LNG FLT	60	616	2	zeros	2
RHBEO	43	HPF'd Baseband Excitation	—60— —60— -12- | -12- | -12- ←RHBEO→ ←RHBE1→ ←RHBE→	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	60	2300	3	(zeros)	2
RHBE1	44	RHBEO plus 12 previous frame elements in front	(see RHBEO)	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	72	2276	3	zeros	2
RHBE	45	84 HPF'd Baseband Excitation Samples centered on Previous Frame	(see RHBEO)	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	84	2156	3	-	2
RLPU	-	LPF Coefficients (with gain of 3 for up-sample) to get up-sampled Baseband Excitation	RLPU(0) RLPU(1) RLPU(2) RLPU3(0) | RLPU2(0) | RLPU1(0) etc.	DCA96I	SYNZ (HFR)	1	LNG FLT	75	2420	3	LPC coefficients (with gain of 3) for 75 point symmetric filter, passband edge at 925 Hz, stopband edge at 1111 Hz, stopband rejection of -35 dB.	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RLPU1	46	Every 3rd sample of RLPU starting with 3rd	(see RLPU)	DCA96I	SYNZ (HFR)	3	LNG FLT	25	2424	3	(LPF Coefficients)	1
RLPU2	47	Every 3rd sample of RLPU starting with 2nd	(see RLPU)	DCA96I	SYNZ (HFR)	3	LNG FLT	25	2422	3	(LPF Coefficients)	1
RLPU3	48	Every 3rd sample of RLPU starting with 1st	(see RLPU)	DCA96I	SYNZ (HFR)	3	LNG FLT	25	2420	3	(LPF Coefficients)	1
RUBE	49	Upsampled, LPF'd Baseband Excitation	RUBE(0) RUBE(1) RUBE(2) / RUBE1(0) RUBE2(0) RUBE3(0) ...etc.	SYNZ (HFRP)	SYNZ (HFR)	1	LNG FLT	180	3514	3	-	2
RUBE1	50	Every 3rd sample of RUBE starting with 1st	(see RUBE)	SYNZ (HFR)	SYNZ HFR)	3	LNG FLT	60	3514	3	-	2
RUBE2	51	Every 3rd sample of RUBE starting with 2nd	(see RUBE)	SYNZ (HFR)	SYNZ (HFR)	3	LNG FLT	60	3516	3	-	2
RUBE3	52	Every 3rd sample of RUBE starting with 3rd	(see RUBE)	SYNZ (HFR)	SYNZ (HFR)	3	LNG FLT	60	3518	3	-	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RRND	53	Random Number Array	-	DCA96I	SYNZ (HFR)	1	LNG FLT	60	856	2	Gaussain Random Number sequence, Mean = 0 Standard Deviation = 1	1
RUPB0	54	Current Frame of Upsampled and Perturbed Baseband Samples	—180— —180— -37- -37- -37- ←RUPB0→ ←RUPB1→ ←RUPB→	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	180	3154	3	(zeros)	2
RUPB1	55	RUPB0 plus 37 previous frame elements in front	(see RUPB0)	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	217	3080	3	zeros	2
RUPB	56	254 Upsampled and Perturbed Baseband Samples centered on previous frame	(see RUPB0)	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	254	2720	3	-	2
RHPU	57	HPF Coefficients (with gain of 3 for upsample) to get upsampled, perturbed samples	-	DCA96I	SYNZ (HFR)	1	LNG FLT	75	2570	3	HPF Coefficients (with gain of 3) for 75 point symmetric filter, passband edge at 1111 Hz, stopband edge at 925 Hz, stopband rejection of -35 dB.	1
RHUP	58	HPF'd, Upsampled, Perturbed Samples	(identical to RFES)	SYNZ (HFR)	SYNZ (HFR)	1	LNG FLT	180	976	2	-	2
RFES	(58)	Filtered Excitation Samples	(identical to RHUP)	SYNZ (HFR)	SYNZ (SNFL)	1	LNG FLT	180	976	2	-	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RRF1	59	Previous Frame's RRFO (BID #39)	-	SYNZ (SNFL)	SYNZ (SNFL)	1	LNG FLT	8	1336	2	zeros	2
RSFM	60	Synthesis Filter Memory	-	SYNZ (SNFL)	SYNZ (SNFL)	1	LNG FLT	8	1352	2	zeros	2
RSNKA	61	Synthesized Speech	-	SYNZ (SNFL)	IDAINT	1	SHRT FLT	180	47386	1	-	2
RSNKB	62	Synthesized Speech	-	SYNZ (SNFL)	IDAINT	1	SHRT FLT	180	47566	1	-	2
RSNKC	-	D/A "Silence" Data	-	DCA96I	IDAINT	1	SHRT FLT	180	47746	1	zeros	1
RDABA	-	D/A Output to AOM	-	IDAINT	DAPROG (AOM Scroll Program)	1	SHRT FLT	180	47926	1	zeros	2
RDABB	-	D/A Output to AOM	-	IDAINT	DAPROG (AOM Scroll Program)	1	SHRT FLT	180	48106	1	zeros	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
-	12	(Unused)	-	-	-	-	-	-	-	-	-	-
-	25	(Unused)	-	-	-	-	-	-	-	-	-	-
-	26	(Unused)	-	-	-	-	-	-	-	-	-	-
-	27	(Unused)	-	-	-	-	-	-	-	-	-	-
-	30	(Unused)	-	-	-	-	-	-	-	-	-	-
-	31	(Unused)	-	-	-	-	-	-	-	-	-	-
-	32	(Unused)	-	-	-	-	-	-	-	-	-	-

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
-	34	(Unused)	-	-	-	-	-	-	-	-	-	-
-	42	(Unused)	-	-	-	-	-	-	-	-	-	-
-	63	Used as temporary buffer for initialization and other functions	-	-	-	-	-	-	-	-	-	-

APPENDIX D: MAP-300 SCALARS

The tables contained in this Appendix describe the characteristics of each real and each integer user-defined scalar referenced in the MAP-300 speech coder implementation. The table entry labeled "SID" or "ISID" indicates the real or integer scalar identification number. The entries labeled "Written by:" and "Read by:" indicate subroutines, function lists, or program modules which so access the scalar. (The A/D, D/A, transmitter modem, and receiver modem interrupt routines are specified in these tables by "IADINT", "IDAINT", "ITMINT", and "IRMINT", respectively.)

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TDCN	50	Negative of current frame D.C. term	-	ANLZ (LPC)	ANLZ (LPC)	-	2
TFSZI	51	$-\frac{1}{\text{Framesize}}$	-	DCA96I	ANLZ (LPC) (PEDET)	$-\frac{1.}{180.}$	1
TKTHR	52	*Threshold value* for MWLF (see CSPI MWLD documentation)	-	DCA96I	ANLZ (LPC)	.9995	1
-	53	*(Unused)*	-	-	-	-	-
-	54	(Unused)	-	-	-	-	-
TPTC	55	Pitch (index of maximum baseband autocorrelation - between 5 and 38)	-	ANLZ (PEDET)	ANLZ (PEDET) (APC)	-	2
-	56	*(Unused)*	-	-	-	-	-

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TE	57	Energy of Pitch-removed Baseband excitation	-	ANLZ (PEDET)	ANLZ (PEDET)	-	2
TQTAP	58	Quantized Tap Coefficient	Must be consecutive	ANLZ (PEDET)	ANLZ (PEDET) (APC)	-	2
TQG	59	Quantized Gain (Gain = $\sqrt{TE}$)	Must be consecutive	ANLZ (PEDET)	ANLZ (APC)	-	2
TQGI	60	Inverse of Quantized Gain (1/TQG)	Must be consecutive	ANLZ (PEDET)	ANLZ (APC)	-	2
TPT1	61	Baseband Quantization Threshold 1	Must be consecutive	DCA96I	ANLZ (APC)	0.531	1
TBT2	62	Baseband Quantization Threshold 2	Must be consecutive	DCA96I	ANLZ (APC)	1.248	1
TBT3	63	Baseband Quantization Threshold 3	Must be consecutive	DCA96I	ANLZ (APC)	2.371	1

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	*SID*	*Description*	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TBQ0	64	Baseband Quantization Value 0	Must be consecutive	DCA96I	ANLZ (APC)	0.233	1
TBQ1	65	Baseband Quantization Value 1	Must be consecutive	DCA96I	ANLZ (APC)	0.830	1
TBQ2	66	Baseband Quantization Value 2	Must be consecutive	DCA96I	ANLZ (APC)	1.666	1
TBQ3	67	Baseband Quantization Value 3	Must be consecutive	DCA96I	ANLZ (APC)	3.075	1
RTAP	68	Decoded Tap Coefficient	-	SYNZ (APCI)	SYNZ (APCI)	-	2
-	69	(Unused)	-	-	-	-	-
RBWC1	70	Coefficient 1 for Butterworth Filter	Must be consecutive	DCA96I	SYNZ (HFR)	-2.0000	1

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RBWC2	71	Coefficient 2 for Butterworth Filter	Must be Consecutive	DCA96I	SYNZ (HFR)	1.0000	1
RBWC3	72	Coefficient 3 for Butterworth Filter	Must be Consecutive	DCA96I	SYNZ (HFR)	-1.7085	1
RBWC4	73	Coefficient 4 for Butterworth Filter	Must be Consecutive	DCA96I	SYNZ (HFR)	.7459	1
RBWM1	74	Memory 1 for Butterworth Filter Y(UBS-2)	Must be Consecutive	SYNZ (HFR)	SYNZ (HFR)	0.	2
RBWM2	75	Memory 2 for Butterworth Filter Y(UBS-1)	Must be Consecutive	SYNZ (HFR)	SYNZ (HFR)	0.	2
RBWM3	76	Memory 3 for Butterworth Filter U(UBS-2)	Must be Consecutive	SYNZ (HFR)	SYNZ (HFR)	0.	2
RBWM4	77	Memory 4 for Butterworth Filter U(UBS-1)	Must be Consecutive	SYNZ (HFR)	SYNZ (HFR)	0.	2

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RPC1	78	Perturbation Threshold Constant	-	DCA96I	SYNZ (HFR)	.7	1
RPC2	79	Perturbation Gain Factor Constant	-	DCA96I	SYNZ (HFR)	$-\frac{.7}{35.}$ * 2.**10	1
TCTAP	80	Coded Tap (Integer in left halfword)	Must be Consecutive	ANLZ (PEDET)	ANLZ (APC)	-	2
TCG	81	Coded Gain (Integer in left halfword)	Must be Consecutive	ANLZ (PEDET)	ANLZ (APC)	-	2
TCRF1	82	Previous Frame Coded Reflection Coefficient (1) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
TCRF2	83	Previous Frame Coded Reflection Coefficient (2) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
TCRF3	84	Previous Frame Coded Reflection Coefficient (3) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TCRF4	85	Previous Frame Coded Reflection Coefficient (4) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
TCRF5	86	Previous Frame Coded Reflection Coefficient (5) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
TCRF6	87	Previous Frame Coded Reflection Coefficient (6) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
TCRF7	88	Previous Frame Coded Reflection Coefficient (7) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
TCRF8	89	Previous Frame Coded Reflection Coefficient (8) (Integer in left halfword)	Must be Consecutive	ANLZ (LPC)	ANLZ (APC)	-	2
-	90	(Unused)	-	-	-	-	-
RPTC	91	Decoded Pitch	-	SYNZ (APCI)	SYNZ (APCI)	-	2

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
T5	92	Constant 5	-	DCA96I	ANLZ (PEDET)	5.	1
-	93	(Unused)	-	-	-	-	-
TDFSI	94	Inverse of Downsampled Framesize	-	DCA96I	ANLZ (PEDET)	$\frac{1.}{60.}$	1
-	95-191	(Unused)	-	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TSRFA	50	TSRA Buffer Status Flag ($0 \Rightarrow$ Empty)	-	ANLZ, IADINT	DCALP, IADINT	0	2
TSRFB	51	TSRB Buffer Status Flag ($0 \Rightarrow$ Empty)	-	ANLZ, IADINT	DCALP, IADINT	0	2
TBTFA	52	TBTA Buffer Status Flag ($0 \Rightarrow$ Empty)	-	ANLZ, ITMINT	DCALP, ITMINT	0	2
TBTFB	53	TBTB Buffer Status Flag ($0 \Rightarrow$ Empty)	-	ANLZ, ITMINT	DCALP, ITMINT	0	2
TIPTC	54	(Unused)	-	-	-	-	-
-	55	(Unused)	-	-	-	-	-
-	56	(Unused)	-	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RBTFA	57	RBTA Buffer Status Flag (0⇒Empty)	-	SYNZ, IRMINT	DCALP, IRMINT	0	2
RBTFB	58	RBTB Buffer Status Flag (0⇒Empty)	-	SYNZ, IRMINT	DCALP, IRMINT	0	2
RSNFA	59	RSNA Buffer Status Flag (0⇒Empty)	-	SYNZ, IDAINT	DCALP, IDAINT	0	2
RSNFB	60	RSNB Buffer Status Flag (0⇒Empty)	-	SYNZ, IDAINT	DCALP, IDAINT	0	2
RIPTC	61	(Unused)	-	-	-	-	-
RUN	62	System Run Flag (0⇒ Stop)	Must be Consecutive	DCARTS, DCA96E	DCARTS	0	2
-	63	(Left unused for MPITM)	Must be Consecutive	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics'	Written by:	Read by:	Initial Value	Usage
ADPO	64	TADBA/B Buffer Pointer (0⇒B; -2⇒A)	-	IADINT	IADINT	0	2
TSRPO	65	TSRA/B Buffer Pointer (0⇒B; -2⇒A)	-	IADINT	IADINT	-2	2
TBTPO	66	TBTA/B Buffer Pointer (0⇒B; -2⇒A)	-	ITMINT	ITMINT	0	2
TMPO	67	TMDMA/B Buffer Pointer (0⇒B; -2⇒A)	-	ITMINT	ITMINT	0	2
RMPO	68	RMDMA/B/C Buffer Pointer (0⇒C; -2⇒B; -4⇒A)	-	IRMINT	IRMINT	0	2
RBTPO	69	RBTA/B Buffer Pointer (0⇒B; -2⇒A)	-	IRMINT	IRMINT	-2	2
RSNPO	70	RSNKA/B Buffer Pointer (0⇒B; -2⇒A)	-	IDAINT	IDAINT	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
DAPO	71	RDABA/B Buffer Pointer (0=>B; -2=>A)	-	IDAINT	IDAINT	0	2
TADFDC	72	A/D Frame Discard Counter	-	IADINT	DCA96E	0	2
TMFFC	73	TMODEM Fake Frame Counter	-	ITMINT	DCA96E	0	2
RMFDC	74	RMODEM Frame Discard Counter	-	IRMINT	DCA96E	0	2
RSNNR	75	RSINK Data-Not-Ready Counter	-	IDAINT	DCA96E	0	2
TFRCTR	76	Transmitter Frame Counter	-	IADINT	DCA96E	0	2
RFRCTR	77	Receiver Frame Counter	-	IRMINT	DCA96E	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RLSCTR	78	Receiver Lost-Sync Counter	-	IRMINT	DCA96E	0	2
RONHK	79	Local Handset On-Hook State (0⇒Handset Off-Hook)	-	IRMINT	IADINT, DCA96E	0	2
RSYNC	80	State of Receiver Sync 0⇒Lost Sync -1⇒Searching Sync +1⇒Found Sync	-	IRMINT	IRMINT, DCA96E	0	2
RBOFO	81	Beginning-of-Frame Offset (Sync Bit Position in RMODEM Buffer)	Must be consecutive	IRMINT	IRMINT, DCA96E	-	2
-	82	(Left unused for MPITM)	Must be consecutive	-	-	-	-
-	83-122	(Unused)	-	-	-	-	-
RNOCOR	123	Error-Correction switch (0⇒Error Correction)	Must be consecutive	DCA96E	CORECT	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
-	124	(Left unused for MPITM)	Must be consecutive	-	-	-	-
RERSIM	125	Set to non-zero to cause channel error simulation	Must be consecutive	DCA96E	IRMINT	0	2
-	126	(Left unused for MPITM)	Must be consecutive	-	-	-	-
VSTATE	127	Debugging variable (Unused in real-time vocoder)	-	-	-	-	-