MASTER

Realization of a real-time 16 kbit/s speech coder and decoder on a single digital signal processor chip each

Kleuters, R.P.J.

Award date:
1987

Link to publication
REALIZATION OF A REAL-TIME 16 KBIT/S SPEECH CODER AND DECODER ON A SINGLE DIGITAL SIGNAL PROCESSOR CHIP EACH

by R.P.J. Kleuters

Coaching professor: prof. dr. J.C. Arnbak.
Coach: ir. A.P. Verlijsdonk.

The department of electrical engineering of the Eindhoven University of Technology disclaims any responsibility for the contents of training reports and graduation theses.
CONTENTS

SUMMARY 4

LIST OF ABBREVIATIONS AND SYMBOLS 5

1. INTRODUCTION 7

2. GENERAL DESIGN CONSIDERATIONS 9
   2.1. Introduction 9
   2.2. General principle of Subband Coding 9
   2.3. Capacities of the used digital signal processor 11
       2.3.1. Processor speed 11
       2.3.2. Program memory and data memory 12
       2.3.3. Arithmetical operations 12
       2.3.4. Fixed-point two's complement arithmetic 13
   2.4. Equipment for testing and realizing the software developments 13
   2.5. Design strategy 15

3. ANALYSIS/SYNTHESIS FILTER BANK 16
   3.1. Introduction 16
   3.2. Design considerations 17
   3.3. Implementation of the chosen filter bank algorithm 23
   3.4. Evaluation of the filter bank performance 29
   3.5. Improvement of subjectively perceived quality 31
   3.6. Conclusion 33

4. BIT-ALLOCATION 34
   4.1. Introduction 34
   4.2. Design considerations 34
   4.3. Implementation of the chosen bit-allocation algorithm 37
   4.4. Experimental check 39
5. CODING AND DECODING OF THE SUBBAND SAMPLES  
5.1. Introduction 41  
5.2. Design considerations 42  
5.3. Implementation of the chosen (de)coder algorithm 45  
5.4. Tests and results 51  
5.4.1. Performance of the coder(s) and decoder(s) 52  
5.4.2. Performance of the chosen bit-allocation method 53  
5.5. Conclusion 54  

6. MULTIPLEXING AND DEMULTIPLEXING OF THE CODED SUBBAND SAMPLES 55  
6.1. Introduction 55  
6.2. Design considerations 56  
6.3. Implementation of the chosen multiplexer and demultiplexer algorithm 58  
6.3.1. Multiplexer 59  
6.3.2. Demultiplexer 61  
6.4. Experimental confirmation of the proper working of the multiplexer and demultiplexer 63  

7. HARDWARE CONSIDERATIONS 65  
7.1. Introduction 65  
7.2. I/O of the transmitter DSP 65  
7.3. I/O of the receiver DSP 66  
7.4. Communication between transmitter and receiver system 67  
7.5. Outline of the hardware configuration that realizes a 16 kbit/s Subband Coder and Decoder 67  

8. UTILIZATION OF THE TRANSMITTER AND RECEIVER DSP 69  

9. SUGGESTIONS FOR (POSSIBLE) IMPROVEMENT OF THE SUBBAND CODER PERFORMANCE 71  

10. CONCLUSIONS 72
REFERENCES 74

APPENDIX A: Frequency response of bandlimiting filter 82

SOURCE FILE LISTINGS:

APPENDIX B: Main program SBC transmitter part 84

APPENDIX C: Subroutine QMF 100

APPENDIX D: Subroutine PCMAQB (coder) 113

APPENDIX E: Subroutine MULTIPLEXING 118

APPENDIX F: Main program SBC receiver part 126

APPENDIX G: Subroutine PCMAQB (decoder) 142

APPENDIX H: Subroutine INVQMF 147

APPENDIX I: Subroutine DEMULTIPLEXING 159
SUMMARY

This report describes the design, implementation and testing of a real-time 16 kbit/s speech coder and decoder. The algorithm used is simple enough for the coder and decoder to be implemented each on a single digital signal processor chip (TMS32010).

In the transmitter the up to 4 kHz bandlimited speech signal is first split into eight mutually exclusive subband signals, each having a spectral bandwidth of 500 Hz. The sampling rates of the subband signals are reduced from 8 kHz to 1 kHz (being the Nyquist rate). For this subband splitting and sampling rate reduction, a very efficient technique called Quadrature Mirror Filtering is used. Next, each subband signal is independently coded according to its perceptual contribution to the overall subjective quality. For coding the subband signals, Pulse Code Modulation, with backward adaptation of the quantization stepsize, is employed. Furthermore for each subband signal, the number of code bits is determined by a semi-adaptive bit-allocation algorithm. Finally, the coded subband samples together with the necessary side information and synchronization bits are multiplexed into a 16 kbit/s serial data stream.

In the receiver, the inverse operations (demultiplexing, decoding and speech signal reconstruction) take place. The quality performance of the implemented Subband Coder, in particular the intelligibility of the reconstructed speech, is acceptable for our purposes. There is only little difference with unprocessed up to 3 kHz bandlimited speech. Furthermore, the capacities of each digital signal processor are nearly completely utilized by the implemented Subband Coder and Decoder algorithm, which means that an appropriate type of digital signal processor has been chosen.
### LIST_OF_ABBREVIATIONS_AND_SYMBOLS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIB</td>
<td>Analog Interface Board; part of Texas Instruments development system.</td>
</tr>
<tr>
<td>A/D</td>
<td>Analog to Digital (conversion).</td>
</tr>
<tr>
<td>AQB</td>
<td>Adaptive Quantization Backward.</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathode Ray Tube.</td>
</tr>
<tr>
<td>D/A</td>
<td>Digital to Analog (conversion).</td>
</tr>
<tr>
<td>d_c</td>
<td>Stepsize adaption control factor.</td>
</tr>
<tr>
<td>d(n)</td>
<td>Log Δ(n).</td>
</tr>
<tr>
<td>Δ(n)</td>
<td>Stepsize for (de-)quantization of a subband sample.</td>
</tr>
<tr>
<td>DPCM</td>
<td>Differential PCM.</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor.</td>
</tr>
<tr>
<td>EVM</td>
<td>Evaluation Module; part of Texas Instruments development system.</td>
</tr>
<tr>
<td>F</td>
<td>Number of bits in a frame of the multiplexed signal.</td>
</tr>
<tr>
<td>FAW</td>
<td>Frame Alignment Word (in multiplexing operations).</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response (filter).</td>
</tr>
<tr>
<td>fₛ</td>
<td>Sampling rate.</td>
</tr>
<tr>
<td>γ</td>
<td>Stepsize adaption leakage factor.</td>
</tr>
<tr>
<td>I(n)</td>
<td>Two's complement B-bit code word; coded subband sample.</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output.</td>
</tr>
<tr>
<td>m</td>
<td>Number of bits in a FAW.</td>
</tr>
<tr>
<td>M(.)</td>
<td>Stepsize adaption parameter.</td>
</tr>
<tr>
<td>m(.)</td>
<td>Log M(.).</td>
</tr>
<tr>
<td>p</td>
<td>Bit error rate.</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer.</td>
</tr>
<tr>
<td>PCM</td>
<td>Pulse Code Modulation.</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Density Function.</td>
</tr>
<tr>
<td>PROM</td>
<td>Programmable Read Only Memory.</td>
</tr>
<tr>
<td>QMF</td>
<td>Quadrature Mirror Filter.</td>
</tr>
<tr>
<td>Qx</td>
<td>Fixed-point format in which a number is represented in the TMS32010. A number is represented with 1 sign bit, 15-x integer bits and x fractional bits.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory.</td>
</tr>
<tr>
<td>rms</td>
<td>Root mean square.</td>
</tr>
<tr>
<td>SBC</td>
<td>Subband Coder.</td>
</tr>
<tr>
<td>$s(n)$</td>
<td>8 kHz digital speech input to transmitter.</td>
</tr>
<tr>
<td>$\hat{s}(n)$</td>
<td>8 kHz reconstructed digital speech output from receiver.</td>
</tr>
<tr>
<td>$s(t)$</td>
<td>Analog speech input to transmitter.</td>
</tr>
<tr>
<td>$\hat{s}(t)$</td>
<td>Reconstructed analog speech output from receiver.</td>
</tr>
<tr>
<td>$t_1$</td>
<td>Mean time to acquire frame alignment.</td>
</tr>
<tr>
<td>$t_2$</td>
<td>Mean time between false indications of lost frame alignment.</td>
</tr>
<tr>
<td>$x(n)$</td>
<td>Quantizer input; uncoded subband sample.</td>
</tr>
<tr>
<td>$\hat{x}(n)$</td>
<td>Quantized subband sample.</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

In the Telecommunications Division of the Eindhoven University of Technology work is done on a cooperative project with the University of Dar-es-salaam. This project concerns the distribution of audio-visual information via satellite channels to be used for tele-education, information dissemination and news distribution to the rural population and to isolated institutions in Africa. To reduce costs, services suitable for standard narrowband (64 kbit/s) distribution channels are necessary. Examples are transmission of still pictures plus speech, teletext plus speech, silent-teletext and scribophone. For more information on the development project, see [1].

This report deals with the realization of a part of the above mentioned services, namely a real-time speech coder and decoder. In this project 16 kbit/s of a 64 kbit/s channel is reserved for the transmission of speech. Another design requirement is to keep the costs as low as possible. Considering the enormous advances in digital technology, coupled with the increasing economics of digital hardware, it was decided to implement the coder and decoder in software on one digital signal processor (DSP) chip each. A technique that can allow this and that can realize a fairly good quality of the reconstructed speech at the receive side is known as Subband Coding [2,3,4,5].

As the first work on the Subband Coder (SSC) project a study has been made about the kind of Subband Coding which best meets the requirements [6]. From this work a proposal resulted for the type of Subband Coding to be used. During further work [7,8] implementations of functional blocks of this proposal were designed and tested.

To realize a properly working Subband Coder and Decoder these implementation designs had to be adjusted and extended. To meet the DSP capacities the software also had to be optimized. Besides these alterations of the implementation designs and the software, the hardware communication between the transmitter and the receiver system had to be realized.
In this report the ultimate implementation design and hardware environment of a properly working real-time Subband Coder and Decoder realization are described.

First some general design considerations are discussed (chapter 2). This is followed by a description of the functional blocks of the Subband Coder and Decoder and the work that is done on the implementation design of them (chapter 3 to 6). After that the employed hardware will be considered (chapter 7). Finally the utilization of the transmitter and receiver DSP is discussed (chapter 8), followed by some suggestions for possible performance improvement (chapter 9) and the conclusions (chapter 10).
2. **GENERAL DESIGN CONSIDERATIONS**

2.1. **Introduction**

In this chapter some general aspects concerning the design of a Subband Coder are considered. First we will discuss the principle of Subband Coding. Then some features of the used digital signal processor are showed. This is followed by a description of the development equipment, and finally the design strategy is outlined.

2.2. **General principle of Subband Coding**

A general block diagram of a Subband Coder and Decoder is shown in Fig. 1.

Fig. 1: General block diagram of a Subband Coder and Decoder.

The digital input to the transmitter, s(n), is obtained by sampling a bandlimited (up to 4 kHz) analog speech signal at a rate of 8 kHz, followed by 12 bit linear PCM analog-digital conversion. This signal is filtered into a number of N subband signals, each with a different spectrum that is part of the baseband frequency range 0-4 kHz. After the bandsplitting the sampling rates, f_s, of these subband signals can be reduced (decimation). This is possible because
The spectrum of each subband signal is narrower than the spectrum of the input signal. The minimum sampling frequency is twice the bandwidth of a subband signal. Decimation by an integer factor M can be achieved by retaining only one out of every M filter output samples. Then the subband signals are independently quantized and PCM coded into a number of bits that is determined by the bit-allocation used. In any case the bit-allocation is so as to allow an ultimate bit rate of 16 kbit/s. Finally the different subband signals together with the necessary side and synchronization information will be multiplexed into a serial 16 kbit/s data stream, representing the output signal of the transmitter.

In the receiver the inverse actions will take place. First the incoming data stream is demultiplexed to recover the coded subband signals plus the side and synchronization information. Next the subband signals are decoded in the PCM decoders. The decoding takes place according to the bit-allocation used, which in an adaptive case can be extracted from the received side information. By inserting zero-valued samples between the subband samples, followed by a filtering identical to that in the transmitter, the sampling rate in the subband signals will be restored to the original sampling rate of 8 kHz. During the filtering interpolation takes place, which gives the inserted zero's an appropriate value. Increasing the sampling rate with an integer factor M is accomplished by filling in M-1 zero-valued samples between each pair of filter input samples. By combining these final obtained subband signals the original digital speech signal, $\xi(n)$, can be reconstructed.

The main advantages of Subband Coding relative to other coding techniques are:

- A low bit rate.
- The quantization noise generated within each subband remains confined to that channel and is independent of noise produced in other bands. In this way high level, low frequency quantization noise does not mask low level, high
frequency sounds, and vice versa.

Each subband signal may be coded independently according to the perceptual contribution that each band makes to the overall subjective quality. This together with the sampling rate reduction in the subbands is the explanation for the low bit rate that can be achieved.

By proper design a real-time implementation of the whole SBC principle is possible on two DSP's; one for the transmitter and one for the receiver.

These are enough reasons for using the Subband Coding technique for the realization of a 16 kbit/s speech coder and decoder that together deliver fairly good quality speech at a fairly low complexity (c.q. costs).

2.3. Capacities of the used DSP

The DSP used for the implementation of the SBC algorithm, is the TMS32010 from Texas Instruments. This TMS32010 is very suitable for speech processing applications. For an extensive documentation of the TMS32010, see [9]. As both the coder and decoder had to be implemented on one DSP chip each, it is not strange that the features and capacities of the TMS32010 have had a great influence on the design work. The most important ones will be discussed below.

2.3.1. Processor speed

The duration of one instruction cycle is 200 ns. Subband Coding is a real-time speech coding technique that in our case every 0.125 ms uses one input sample in the transmitter and creates one output sample in the receiver. So in the transmitter as well as in the receiver the whole coder resp. decoder program has to be passed through in 0.125 ms. So the longest cycle in either program may not count more than 625 instruction cycles.
2.3.2. Program memory and data memory

Program memory consists of up to 4K words of 16-bit width. In our case this concerns off-chip RAM. The available program memory is large enough and has been no constraint on the design work.

On the other hand data memory consists of 144 words of 16-bit width of RAM present on-chip. So data RAM is far too small to contain all variables for the coder or decoder. This problem can be solved by storing some data operands in off-chip program memory and then read them into on-chip data memory when needed. When they are not needed anymore they can be written back to program RAM if necessary. However this solution will slow down execution as reading from and writing to program memory each take 600 ns, whilst most instructions only need 200 ns. So it is necessary to carefully determine which variables are directly stored in data memory and which are stored in program memory. Variables that are not often used and tables for example can best be stored in program memory. For more information, see [7, pp. 39-40]. The available data memory has appeared to be a great constraint in the design of the SBC algorithm.

2.3.3. Arithmetical operations

Addition and subtraction are standard tools for the TMS32010 and will cost only one instruction cycle. Also multiplication can take place in 200 ns due to the presence of a 16×16-bit parallel multiplier.

However, the TMS32010 does not have an explicit divide instruction. A division therefore will have to be broken down into a series of subtractions and shifts with the consequence of a long execution time. So divisions have to be avoided as far as possible.
2.3.4. Fixed-point arithmetic

Computation on the TMS32010 is based on a fixed-point two's complement representation of numbers. Each 16-bit number is evaluated with a sign bit, i integer bits, and 15 minus i fractional bits. Thus the number:

```
0000 0010 1010 0000
```

has a value of 2.625. This particular number is said to be represented in a Q8 format (8 fractional bits). Its range is between -128 (1000 0000 0000 0000) and +127.996 (0111 1111 1111 1111). The fractional accuracy of a Q8 number is about 0.004 (one part in 2**8 or 256).

To reduce quantization noise on the one side and to avoid overflows on the other side, it is very important to select an appropriate representation (i.e. Q-format) for each variable and constant.

On the design work we have started from the principle that the transmitter DSP input and the receiver DSP output have been normalized to the range between -1 and +1 Volt, and therefore a Q15 format has been taken for the representation of these two signals.

2.4. Equipment for testing and realizing the software developments

Two development systems as shown in Fig. 2 are available, one for the transmitter DSP and one for the receiver DSP. Such development system consists of two printed circuit boards, called "Evaluation Module" (EVM) and "Analog Interface Board" (AIB), and a host computer, in our case a Personal Computer from IBM.

The basic purpose of the Evaluation Module and the Analog Interface Board used is to enable the user to develop programs for the TMS32010 digital signal processor and to run
these programs real-time. The programs can communicate with the analog "outside world" by means of A/D and D/A converters and bandlimiting/interpolation filters resident on the AIB. The program storage, the program design tools and the DSP itself are resident on the EVM board.

The two printed circuit boards of Texas Instruments are controlled by the host computer (IBM PC). This PC can communicate with the EVM via a serial interface. At the IBM PC the serial interface is controlled under BASIC. The EVM board is equipped with two serial interfaces of which only one, called "port 1", will be used. All serial interfaces are bi-directional. The EVM board together with the IBM PC form a program development system with which the user may enter assembler source files, assemble files, run them, execute them with breakpoints and single step programs.

The EVM and the AIB together can form a stand-alone system once these are programmed and started.

The EVM and AIB are well documented in [10] resp. [11]. In [7, pp. 24-32] the possibilities of the development system and the implications of this for a communications program resident in the IBM PC are discussed in more detail.

Later on in the project the development equipment has been extended with a so-called cross-assembler, which allows us to assemble source files on the IBM PC. This cross-assembler is described in [12] and [13].
2.5. Design strategy

When we look back at the general SBC principle described in 2.2., we can discern the following functional blocks:

- filtering
- bit-allocation
- coding and decoding
- multiplexing and demultiplexing

As a logical consequence the design work, implementation and testing has been done in a modular way. First the filtering has been realized and tested on proper working. After that the bit-allocation has been added. This was followed by adding the coding and decoding. Finally everything has been supplemented with the multiplexing and demultiplexing.
3. **ANALYSIS/SYNTHESIS_FILTER_BANK**

3.1. **Introduction**

An important and critical aspect of the SBC design is the filter bank and its interaction with the sampling rate reduction (decimation) in the transmitter and the sampling rate increase (interpolation) in the receiver. A good reconstruction of the speech signal, \( S(n) \), requires a subband splitting in the transmitter and subsequent subband combining in the receiver with an overall frequency response that in magnitude equals 1 as best as possible.

If we leave out of consideration effects such as quantization noise, which are no direct consequence for filtering, there are two effects that can cause the magnitude of the overall frequency response to be not exactly equal to 1 everywhere. Both effects concern the fact that the filters we deal with in practice are not ideal in their frequency response, but contain beside the passband also a transition band and a stop band.

The reduction of the subband sampling rates is necessary in order to maintain a low (16 kbit/s) bit rate in encoding these signals. This sampling rate reduction introduces aliasing terms [14, pp. 304-305] in each of the subband signals.

In the reconstruction process the subband signals are combined together. As interpolation in the receiver takes place with a filtering identical to the one in the transmitter, after reconstruction the original signal has twice passed the filter bank represented by the frequency responses \( H_1(f) \), \( H_2(f) \), ..., \( H_W(f) \). So exact reconstruction requires:

\[
|H_1(f)|^2 + |H_2(f)|^2 + ... + |H_W(f)|^2 = 1.
\]

In places were this requirement is not met, a ripple occurs in the response, called reconstruction ripple. Reconstruction ripple mainly occurs in the transition regions of the filter responses.
Thus concerning the filter bank development, aliasing and reconstruction ripple have to be suppressed or avoided as much as possible to accomplish a good speech signal reconstruction.

In the following the design, implementation and testing of the subband splitting and reconstruction method that has been chosen for our SBC realization is discussed.

3.2. Design considerations

Several methods exist for the splitting of the SBC input signal into a number of mutually exclusive subbands [6]. Some methods [15,16,17] are based on using a parallel bank of bandpass filters, as shown in Fig. 3.

![Diagram of parallel bank of bandpass filters](image)

Fig. 3: Parallel bank of bandpass filters for subband splitting.

These filters may be conveniently realized by linear-phase Finite Impulse Response (FIR) networks. However, to meet the design requirements as mentioned in 3.1., very sharp transition band filters, i.e. very high order networks (ca. 200-tap) with a strong stop band attenuation (ca. 45 dB) are needed. Such filters would have a complexity that is far too high, excessive delay and possible performance limitations due to finite precision wordlengths if realized directly.
A more convenient alternative [18,19,20,21] is the use of a tree-structured filter bank. Such a tree configuration works by successively splitting the signal into two subbands at each branching point, using a high-pass/low-pass filter combination. The output from each intermediate filter is downsampled to the appropriate Nyquist rate for the signal and then applied to the next branching level for further spectral division, see Fig. 4.

Fig. 4: Tree-structured filter bank for uniformly spaced subbands.

Besides this uniform subband splitting also octave-spaced subbands are possible when using the approach as shown in Fig. 5.

Fig. 5: Tree-structured filter bank for octave-spaced subbands.
If we would realize the tree-structured filter bank with the usual FIR filters, without taking any special measures, again very high order networks are needed. However for the bandsplitting a technique called "Quadrature Mirror Filtering" can also be used. Quadrature Mirror Filters (QMF's) have special phase and magnitude characteristics which allow the splitting of a band into two equal width subbands and, upon reconstruction, provide for the cancellation of aliasing effects that occur during downsampling. Furthermore, as each band is splitted into two symmetrical parts, it is not very difficult to see to it that reconstruction ripple is strongly suppressed. Because of these properties, lower order filters (32 to 12 tap), which have fairly wide transition widths, can be used to cover the entire speech band of interest without any spectral gaps in the total response. Therefore, and also for some other reasons to be discussed later, a tree-structured QMF bank has been chosen for the realization of our SBC.

Fig. 6 reviews the basic configuration for a two-band SBC design that will be used for the explanation of the QMF bandsplitting and derivation of its design requirements.

The transmitter input signal $s(n)$ is divided into two equally spaced frequency bands by low-pass and high-pass filters, $h_l(n)$ and $h_h(n)$, respectively. Each subband signal is reduced in sampling rate by a factor of two, i.e. if $f_s$ is the sampling rate of the input signal, $f_s/2$ is the sampling rate of the subband signals. In practice the subband signals are then encoded and multiplexed for transmission, which in this (modular) design stage will be left out of consideration.

In the receiver the subband signals are interpolated back to their original sampling rates with the aid of similar low-pass and high-pass filters. The sum of the two interpolated subband signals, $s(n)$, is the reconstructed version of the input signal, $s(n)$. 
To obtain the aliasing cancellation property, the filters $h_L(n)$ and $h_u(n)$ must be symmetrical FIR designs with even numbers of taps, i.e.,

$$h_L(n) = h_u(n) = 0 \quad \text{for } n < 0 \text{ and } n > N-1$$  \hfill (1)

where $N$ (even) is the number of taps. The symmetry property implies that

$$h_L(n) = h_L(N-1-n), \quad n = 0, 1, 2, \ldots, N/2-1, \text{ and}$$  \hfill (2a)

$$h_u(n) = -h_u(N-1-n), \quad n = 0, 1, 2, \ldots, N/2-1$$  \hfill (2b)
Furthermore it is required that

\[ h_u(n) = (-1)^n h_r(n) \quad n=0,1,2, \ldots, N-1 \]  \hspace{1cm} (3)

which is the mirror image relationship of the filters. With the above constraints, the aliasing cancellation property of the QMF bank can be verified easily. A derivation is given in the appendix of (20).

To suppress reconstruction ripple as much as possible, the filters \( h_r(n) \) and \( h_u(n) \) ideally must satisfy also the condition

\[ |H_r(f)|^2 + |H_u(f)|^2 \equiv 1 \]  \hspace{1cm} (4)

where \( H_r(f) \) and \( H_u(f) \) are the Fourier transforms of \( h_r(n) \) and \( h_u(n) \) respectively. This also can be seen from the derivation in the appendix of (20).

The above filter requirement of eq. (4) cannot be met exactly except when \( N=2 \) and when \( N \) approaches infinity. However, it can be very closely approximated for modest values of \( N \). Filter designs which satisfy eq. (2a) and approximate the condition of eq. (4) and the lowpass characteristic can be obtained with the aid of an optimization program. In practice, "Hanning window" designs, optimized by the "Hooke and Jeeves algorithm" [6,21] will give satisfactory results.

Fig. 7 shows the frequency response characteristics for an \( N=32 \)-tap filter design, acquired with the above mentioned technique. As can be seen from Fig. 7b, the requirement of eq. (4) is satisfied to within \( 0.025 \) dB, which is more than satisfactory for good SBC performance.

The QMF technique discussed for the two-band SBC from Fig. 6a can be applied in the same way at each branching point in a SBC that employs a tree-structured filtering into more than two subbands.
Finally we have to sketch the form of tree-structured QMF bank that has been chosen. As we will see in chapter 4, some adaption in the bit-allocation is necessary to achieve satisfactory reconstruction speech quality. However, if the subbands are unequally spaced, this requires techniques of a very high complexity. Therefore we have chosen uniformly spaced subbands \([6, 18]\). Furthermore, the perceptual quality of the recovered speech will increase as the number of subbands increases. The number of subbands to be used in practice is a trade-off between recovered speech quality, processing complexity and delay in the filter bank. For our purposes an eight-band SBC has been found to represent a good compromise \([6]\).

A sketch of the ultimate filterbank design to be implemented in the transmitter and receiver is shown in Fig. 8.
3.3. **Implementation of the chosen filter bank algorithm**

Here again, the basic principle of the implementation of the eight-band tree-structured QMF bank is discussed for the case of a two-band SBC, see Fig. 6a. From the mirror image property of the QMF bank described by eq. (3), we note that the coefficients used for the upper and lower subband filters are identical, except for the signs of the odd numbered coefficients. This property can be used to save a factor of two in computation by sharing the computation between the
filters in the manner described in Fig. 9, where \( h_e(n) = h(n) \) and \( h_o(n) = (-1)^n h(n) \). The partial sums of products are accumulated separately for the even- and odd-filter coefficient values. The sum of these two partial sums then gives the lower subband signal, and their difference produces the upper subband signal. Since the subband sampling rates are one-half of the input sampling rate, an additional factor of two is gained by computing the sums of products indicated in Fig. 9 once for every other input sample. Thus, each sample is shifted two delays in the shift register of Fig. 9 before being used.

Because of this sampling rate reduction, the filter structure of Fig. 9 can be divided into two parts as shown in Fig. 10.

![Fig. 10: Polyphase QMF bank structure of a SBC transmitter.](image)

This structure is a two-band version of a more general class of multirate structures sometimes referred to as polyphase structures. As Fig. 10 shows, the input signal is separated into two sets by a commutator. Assuming that the commutator is in the lower position at time \( n = -1 \), the lower branch receives odd values of \( s(n) \), i.e. \( s(-1), s(1), s(3), \ldots \), and the upper branch receives even values of \( s(n) \), i.e. \( s(0), s(2), s(4), \ldots \). Both branches now operate at one-half of
the original sampling rate. Odd values of $s(n)$ are filtered at odd sample times in the lower branch with an $N/2$ tap filter of odd valued filter coefficients (cycle 0). Similarly, even valued samples of $s(n)$ are filtered in the upper branch with a $N/2$ tap filter of even filter coefficients. Furthermore double buffering is required when data computed in one cycle are needed in another cycle. The outputs of the even and odd filters are computed and stored in the left buffer for cycles 0 and 1. For taking the sum and difference the available data from the right buffer, which have been computed in the previous 0 and 1 cycles, are used. At the beginning of cycle 0 the data are transferred from the left buffer to the right buffer. The sum in the lower branch of Fig. 10 is computed in cycle 0, while the difference is computed in cycle 1.

A similar efficient polyphase structure can be generated for the QMF synthesis bank in the receiver. The resulting structure is shown in Fig. 11, that speaks for itself.

So, in this QMF analysis and synthesis implementation, that makes efficient use of the available DSP resources, the computation load is evenly distributed between even and odd time cycles of the input sampling rate. This also has been
one of the reasons for choosing the QMF technique. Table 1 reviews the two-cycle control structure that is used in the implementation of a two-band QMF splitting and reconstruction.

Table 1: Control structure for two-band QMF analysis and synthesis.

<table>
<thead>
<tr>
<th>Cycle 0</th>
<th>A. Transmitter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Double buffer</td>
</tr>
<tr>
<td></td>
<td>2. Create lower band output by summation</td>
</tr>
<tr>
<td></td>
<td>3. Input one sample of $s(n)$</td>
</tr>
<tr>
<td></td>
<td>4. FIR filter (lower branch)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B. Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Double buffer</td>
</tr>
<tr>
<td></td>
<td>2. Create lower branch filter input by subtraction</td>
</tr>
<tr>
<td></td>
<td>3. Input one subband sample and store in left buffer</td>
</tr>
<tr>
<td></td>
<td>4. FIR filter (lower branch)</td>
</tr>
<tr>
<td></td>
<td>5. Output one sample of $s(n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle 1</th>
<th>A. Transmitter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Create upper band output by subtraction</td>
</tr>
<tr>
<td></td>
<td>2. Input one sample of $s(n)$</td>
</tr>
<tr>
<td></td>
<td>3. FIR filter (upper branch)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B. Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Create upper branch filter input by summation</td>
</tr>
<tr>
<td></td>
<td>2. Input one subband sample and store in left buffer</td>
</tr>
<tr>
<td></td>
<td>3. FIR filter (upper branch)</td>
</tr>
<tr>
<td></td>
<td>4. Output one sample of $s(n)$</td>
</tr>
</tbody>
</table>
For the implementation of the eight-band SBC, this technique is repeated at each branching point of the filter tree. As the eight-band SBC is a multirate system with an ultimate sampling rate ratio of eight, it requires an eight-cycle control structure. For each cycle only one out of the eight possible paths in the QMF tree has to be passed through. A modulo-8 counter, called PATH, is used to assign the path in the QMF tree to be processed.

A sketch of the implemented QMF filter bank for the eight-band SBC (transmitter), including the subband processing sequence according to PATH, is shown in Fig. 12. For example, during cycle 3 (PATH=3) a sample of subband 5 (2000-2500 Hz) is created by respectively passing through HIGH1 in the first QMF stage, HIGH3 in the second stage and LOW7 in the third stage.

Fig. 12: QMF filter bank implementation for the eight-band SBC (transmitter)
the third stage. Also it has to be taken into account that after high-pass filtering the input signal into the 2-4 kHz part and sampling rate reduction, the 2-3 kHz part is obtained by high-pass filtering instead of low-pass filtering. This can be verified easily by an examination of the spectra. Similar circumstances occur at some other parts in the QMF tree. As can be seen from Fig. 12, for PATH=0 to 7 the subband processing sequence is: 0-500 Hz, 3500-4000 Hz, 1500-2000 Hz, 2000-2500 Hz, 500-1000 Hz, 3000-3500 Hz, 1000-1500 Hz, 2500-3000 Hz, being respectively subband 1, 8, 4, 5, 2, 7, 3, 6.

For the first QMF stage 32-tap FIR filtering is used (16 taps for the upper branch and 16 taps for the lower branch). For the second and third stage 16- resp. 12-tap FIR filtering is used. The coefficients, as obtained by the Hooke and Jeeves optimization method [21] are depicted in Table 2.

Table 2: Filter coefficients for 12-, 16- and 32-tap

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>12-tap</strong></td>
<td></td>
</tr>
<tr>
<td>h(0) = h(11) = -0.3809699E-2</td>
<td></td>
</tr>
<tr>
<td>h(1) = h(10) = 0.1885659E-1</td>
<td></td>
</tr>
<tr>
<td>h(2) = h(9) = -0.2710326E-2</td>
<td></td>
</tr>
<tr>
<td>h(3) = h(8) = -0.8469594E-1</td>
<td></td>
</tr>
<tr>
<td>h(4) = h(7) = 0.8846992E-1</td>
<td></td>
</tr>
<tr>
<td>h(5) = h(6) = 0.4843894E+0</td>
<td></td>
</tr>
</tbody>
</table>

| **16-tap**  |              |
| h(0) = h(15) = 0.1050167E-2 |
| h(1) = h(14) = -0.5054526E-2 |
| h(2) = h(13) = -0.2589756E-2 |
| h(3) = h(12) = 0.2764140E-1 |
| h(4) = h(11) = -0.9666376E-2 |
| h(5) = h(10) = -0.9039223E-1 |
| h(6) = h(9) = 0.9779817E-1 |
| h(7) = h(8) = 0.4810284E+0 |

| **32-tap**  |              |
| h(0) = h(31) = 0.6910579E-3 |
| h(1) = h(30) = -0.1403793E-2 |
| h(2) = h(29) = -0.1268303E-2 |
| h(3) = h(28) = 0.4234195E-2 |
| h(4) = h(27) = 0.1414246E-2 |
| h(5) = h(26) = -0.9458318E-2 |
| h(6) = h(25) = -0.1303859E-3 |
| h(7) = h(24) = 0.1798145E-1 |
| h(8) = h(23) = -0.4187483E-2 |
| h(9) = h(22) = -0.3123862E-1 |
| h(10) = h(21) = 0.1456844E-1 |
| h(11) = h(20) = 0.5294745E-1 |
| h(12) = h(19) = -0.9980243E-1 |
| h(13) = h(18) = -0.9980243E-1 |
| h(14) = h(17) = 0.1285579E+0 |
| h(15) = h(16) = 0.4664053E+0 |
These coefficients are best represented in Q16 format (see 2.3.4.). In the beginning all time varying signals in the QMF filter bank were decided to be represented in the same format as the input signal s(n), i.e. in Q15 format. Later, as will be discussed in 3.5., these representations have been adapted according to the properties of speech spectra. For reasons mentioned in 2.3.2., all filter coefficients (only 30, thanks to the symmetry property, eq. (2), of the QMF’s) are permanently available in data memory. The same holds for the delay variables of the first QMF stage and furthermore for all buffer variables. However, the delay variables of the second and third QMF stage are stored in program memory. See also [7, pp. 39-40]. Appendix C contains a listing of the source file of the transmitter QMF bank, appendix H that of the receiver QMF bank.

3.4. Evaluation of the filter bank performance

Proper working of the analysis and synthesis filter bank implementation has been tested with the configuration as shown in Fig. 13. The analog input, s(t), is bandlimited and subsequently converted to s(n) by 8 kHz sampling followed by A/D conversion. Next the transmitter DSP, loaded with the QMF analysis
program, creates for each 8 kHz sample a 1 kHz subband sample. Each subband sample is transported in a parallel way from the transmitter DSP to the receiver DSP. In the receiver DSP, loaded with the QMF synthesis program, for each incoming subband sample a 8 kHz output sample, s(n), is produced. By D/A conversion followed by bandlimiting, s(n) is converted to the analog output s(t).

How the bandlimiting and the A/D and D/A conversion are performed, will be discussed in chapter 7. The hardware realization and synchronization of the parallel communication between transmitter and receiver system will not be discussed, as it is not functional for the ultimate SBC realization.

The transmitter DSP program is controlled by interrupts. After each 8 kHz interrupt, created by the A/D converter, an interrupt service routine is executed. In this interrupt service routine a sample s(n) is read in and a created subband sample is written out. When the interrupt service routine has finished, the QMF program (for one out of eight paths) is carried out followed by a wait cycle for the next interrupt. QMF program plus wait cycle together form the main routine.

The receiver DSP program is also controlled by 8 kHz interrupts from the regained system clock, which in our circumstances is retained from the transmitter system (see also chapter 7). In the interrupt service routine a subband sample is read in and a reconstructed speech sample, s(n), is written out. In the subsequent main routine, the inverse QMF program (for one out of eight paths) is carried out followed by a wait cycle for the next interrupt.

So far the same QMF principle also had been realized during previous work [7,8]. Therefore objective measurements of the filter bank performance give the same satisfactory results (for the reconstruction a "flat" curve in the band of interest) as are extensively discussed and illustrated in [7] and [8].

A subjective test has been carried out by applying an analog
speech signal from a tape recorder to the transmitter input, and listening to the reconstructed speech signal via the audio output of the receiver system. By short-circuit of $s(n)$ and $g(n)$, processed speech could be easily compared with unprocessed speech. The intelligibility of processed and unprocessed speech was quite the same. However, processed speech appeared to suffer more from quantization noise, due to the 16-bit representation used in the DSP's.

3.5. Improvement of the subjectively perceived quality

Till now, each time varying signal in the QMF tree has been represented in Q15 format. Watching a long-term speech spectrum as shown in Fig. 14, it is not difficult to see that in general the amplitude range will not be the same for each time varying signal in the QMF tree. So, as mentioned in 2.3.4., the quantization noise can be reduced by choosing an appropriate representation format for each signal in the QMF tree.

Extensive testing, being a very time consuming affair, resulted in the signal representation as depicted in Fig. 15. The signals in the receiver have to be represented in cor-

![Fig. 14: Long-term spectrum of speech based on measurements by Beranek, Dunn and White [15].](image-url)
responding formats. This representation has been implemented, performing a reconstructed speech quality that differs not or only marginally from unprocessed speech. This adjustment of the subband signal representation will also turn out to be advantageous for the encoding of the subband signals (chapter 5), as the dynamic ranges to be covered are better specified.

Fig. 15: Signal representation in (transmitter) QMF filter bank.
3.6. Conclusion

The QMF technique has proved to be very suitable for the implementation of the eight-band filter bank. We do not exaggerate unduly if we call $\hat{s}(n)$ a delayed replica of $s(n)$. However, the encoding of the subband signals into a 16 kbit/s bit stream, left out of consideration till now, will cause a performance degradation by adding an amount of quantization noise to the replica of $s(n)$. 
4. BIT-ALLOCATION

4.1. Introduction

As discussed in 2.2., after the subband splitting each subband signal is coded independently into a number of bits. The ultimate bit rate has to be 16 kbit/s. So, if we reserve 1 kbit/s for side and synchronization information (see also chapter 6), 15 bits per ms are left to divide over the eight 1 kHz subband signals. This division of 15 bits over the eight subbands is determined by the bit-allocation. For realizing a proper quality it is desirable [6] that the number of bits, allocated to a subband, agrees with its perceptual contribution to the overall subjective quality. As can be seen from Fig. 14, this contribution will not be the same for each speaker. The same holds for one speaker at various moments. In other words, the bit-allocation, which is optimal at one certain moment and for one certain speaker, is not necessary the best at another moment and/or for another speaker.

In this chapter the design and implementation of the bit-allocation algorithm that has been chosen for our purpose is discussed. Also described is an experiment to test the functioning of the implemented algorithm.

4.2. Design considerations

Three methods exist [6] for allocating the bits to the several subbands.

For the first one, called fixed-bit-allocation, the number of bits allocated to a subband is the same at each moment for each speaker. This method will not perform an acceptable quality [3,23].

The second method, called dynamic bit-allocation, is a fully adaptive bit-allocation algorithm, where the power is measured in each band and the bits are successively allocated to the subbands [23]. Proper implementation of this method (in combination with the right coding algorithm) will lead to
very acceptable results. However, this method requires a lot of side information and delay and furthermore is far too complex to fit the DSP device capabilities. Therefore a compromise solution, called semi-adaptive bit-allocation, has been chosen for the realization of the SBC. A semi-adaptive bit-allocation scheme is, compared to a fully adaptive bit-allocation method, relatively simple to implement, yet gives a significant improvement in performance over a fixed-bit-allocation scheme. Furthermore the required side information is reduced greatly and, as we shall soon see, the semi-adaptive algorithm does not introduce any additional delay over and above that of the QMF transmitter filter bank.

The first QMF filter outputs are used to obtain energy estimates for the bit-allocation algorithm. The spectral envelope estimate, approximated by the short-term-average magnitude, is computed during a window, determined by the average period for which speech signals are stationary. This is done for the speech in each of the subbands, 0-2 kHz (L) and 2-4 kHz (H). At time intervals, dictated by framing considerations (chapter 6), the ratio between average magnitude estimates for the two bands is used to form a three way decision as to whether the speech is voiced, unvoiced or intermediate. The following voicing decision scheme, recommended in the literature [2,3,4], is used:

\[
\begin{align*}
L/H & > 20 \quad \text{voiced} \\
1.5 & < L/H < 20 \quad \text{intermediate} \\
L/H & < 1.5 \quad \text{unvoiced}
\end{align*}
\]

Hysteresis is included in the decision-making process to prevent rapid strategy changes due to marginal decisions. If the voiced strategy is already in use, L/H must be less than 10 to change back to the intermediate strategy, and for the unvoiced strategy in use, L/H must be greater than 3 to revert back to the intermediate strategy. The frequency range 200-3200 Hz is seen as the band of
interest [6,20] in a speech signal. The 0-200 Hz part contains a lot of power that however is not important for the intelligibility of speech. This part can even better be omitted by the bandlimiting filter (Fig. 13) before the QMF splitting takes place, since it has a wrong influence on the voicing decision and causes a lot of aliasing, both resulting in an inefficient coding of the subband signals. Also the part above 3200 Hz, containing only very little power (Fig. 14), is unimportant for speech intelligibility and may be omitted by the bandlimiting filter (as this part contains very little energy, the filtering required at this edge is not as sharp as at the 200 Hz edge). To cover the band of interest as best as possible, the bit-allocation assignment for the three possible voicing strategies as shown in Table 3 is used [2,3,4].

<table>
<thead>
<tr>
<th>Band (kHz)</th>
<th>0.0</th>
<th>0.5</th>
<th>1.0</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>3.0</th>
<th>3.5</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voiced</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Intermediate</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Unvoiced</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

As can be seen from this table, no bits are allocated to the 3-3.5 kHz subband (i.e. 3000-3200 Hz). Using these bits to improve the coding of the other subband signals, results in a better reconstructed speech quality than inclusion of coding of the 3000-3200 Hz part will do. By consequence it is not necessary to split (and in the receiver to reconstruct) the 3-4 kHz band in the QMF filter bank. However, the 3000-3200 Hz part may not be omitted by the bandlimiting filter, since it contributes to the voicing characteristic of the speech signals. Ultimately the voicing strategy used also has to be transmitted to the receiver. As there are only three possibilities, the side information required is modest.
Fig. 16: Semi-adaptive bit-allocation in QMF filter bank.

The filter bank design, extended with the semi-adaptive bit-allocation to be implemented, is sketched in Fig. 16.

4.3. Implementation of the chosen bit-allocation algorithm

The energy summation and the voicing decision itself are straight implementations of the design discussed in 4.2. (using appropriate representation formats for the variables, all being directly stored in data RAM), and therefore need no further explanation. What really matters in the implementation of the bit-allocation algorithm is the synchronisation with the rest of the SBC program.

For reasons to be discussed in chapter 6, a frame length of 128 bits (8 ms) is used in multiplexing the coded subband signals. Speech may be considered to be in a stationary state during a period of 8 ms. Therefore, each time a new frame starts, the voicing strategy is updated in agreement with an 8 ms energy measurement. As this energy measurement takes place in the first QMF stage, we have to take into account the processing delay time in the second and third
stage of the QMF filter bank. This delay time is calculated
to be 11.5 ms. As the energy measurement takes 8 ms, after
the voicing decision it will last another 3.5 ms before the
first samples corresponding with the voicing decision enter
the PCM coders. Therefore the voicing decision takes place
3.5 ms before a new frame starts, at which the new strategy
is taken over (buffering is required). As the 3-4 kHz
subband is not transmitted and therefore not further split­
ted in the QMF filter bank, a lot of processing time is
saved by executing such voicing decision, where otherwise
the 3-4 kHz bandsplitting would occur.
To perform this synchronization a down-counter, called
FRAME, is used in addition to the modulo-8 counter PATH,
introduced in 3.3. PATH determines the subband to be proces­
sed for an 8 kHz cycle. As discussed earlier, for PATH=0 to
7 this is the subband sequence 1,8,4,5,2,7,3,6. For one
frame this sequence has to be passed through eight times,
kept up by FRAME. PATH is updated for each 8 kHz input,
FRAME is updated each time just before a subband 1 sample is
coded. At the beginning of a frame, the down-counter FRAME
is loaded with the value 7. Each time when PATH=0, FRAME is
adjusted. When FRAME=0 and PATH=0 the voicing strategy for
the PCM coding is updated and a new frame is started. When
FRAME=3 and PATH=5, being ca. 3.5 ms before a new frame
start, a new voicing decision is made.
In the receiver the same synchronization principle is used.
Here, no voicing decision has to take place. The voicing
strategy for the PCM decoding is retained from the side
information (available in the frame alignment word, chapter
6). As in the receiver the PCM decoding is done before the
QMF reconstruction, FRAME is updated each time that PATH=7.
A new frame is started and the voicing strategy is updated
when FRAME=0 and PATH=7.
The whole synchronization is depicted in Fig. 17.
The semi-adaptive bit-allocation software has been added to the filter bank programs of appendix C and H.

4.4. Experimental check

After the DSP programs had been extended with the semi-adaptive bit-allocation algorithms, some testing has been done with the same configuration as shown in Fig. 13. Again an analog speech signal was applied to the transmitter input and listening to the reconstructed speech signal took place via the audio output of the receiver. Although the PCM
coding and decoding had not yet been implemented, the perceived quality of the reconstructed speech was not as good as in the case discussed in 3.5. This is a consequence of not transmitting the 3-4 kHz band. However, the quality, for our purpose particularly the intelligibility, was still fairly satisfactory. Therefore, during all further work this quality has been taken to be the maximum quality that can be achieved with our 16 kbit/s speech coder.

This time, also an analog signal, corresponding with the voicing strategy used, was retained from the transmitter system (AIB) and applied to an oscilloscope. In this way the changing of the voicing strategy for different "types" of speech has been confirmed. And indeed changes do not occur within a 8 ms period. Mostly the voicing strategy is constant for more than one 8 ms period, so the choice of 8 ms as a period during which speech signals may be considered stationary is adequate.

Furthermore, by setting breakpoints and single stepping (2.4.) also the synchronization has been checked.

The suitability of the semi-adaptive bit-allocation and the used bit assignments, shown in Table 3, can (apart from 0 bits allocated to the 3-4 kHz band) only be tested if the PCM coding and decoding has been implemented, and therefore will be discussed in 5.4.2.
5. CODING_AND_DECODING_OF_THE_SUBBAND_SAMPLES

5.1. Introduction

Before coding the different subband signals, an incoming sample or difference between an incoming sample and an estimation value of it (in case of differential PCM = DPCM) has to be quantized. Quantization takes place by allocating one out of \(2^B\) quantization levels to the quantizer input, where \(B\) is the number of bits assigned for the coding of a subband. Next, the quantized value is coded into \(B\) bits.

Design of an efficient encoding scheme requires some knowledge of the statistics of the signal to be coded. If we had an a priori knowledge of the statistics of the samples, a nearly optimum quantization scheme would consist of:

- A quantizer matched to the probability density function (PDF) of the samples to be quantized.
- A predictor optimized for the given autocorrelation function of the signal.

In digital speech-encoding systems, we have only a small amount of a priori knowledge of the statistics which, in addition, usually change with time.

- The long-period mean level differs from speaker to speaker.
- At a given mean level, the instantaneous level changes because of variations in speech sounds (c.q. split speech sounds).
- The correlations (as far as present) between successive samples change because of variations in speech sounds (c.q. split speech sounds).

To overcome these problems of unknown statistics, adaptive quantization and (in case of DPCM) adaptive prediction schemes can be used.

We will now discuss the PCM coder and decoder design that best suits our purpose. Then the implementation and testing of it is described.
Also for the coding of the subband signals several methods are possible. Many of them are extensively discussed in [6, 22, 23, 24, 25].

Concerning the quantization, there are two possibilities, fixed and adaptive quantizers. For reasons like unknown mean level and variations of the instantaneous level, a nonadaptive quantizer will not provide an acceptable quality in our situation. Therefore adaptive quantization has been chosen, in which the $2^B$ possible quantization levels are not fixed in time. For the adaption two schemes are possible. With the first one, forward adaption, the adaption value is calculated from samples of the input signal. As this has to take place in each subband, this method is far too complex and furthermore requires serial buffering (extra delay) and a lot of side information for transmitting the adaption values to the receiver. The second scheme, backward adaption, is more suitable. Here, the adaption value is calculated from quantized samples. In the receiver the same can be done and therefore no side information is required. Furthermore no extra delay above the filtering delay time occurs, since the buffering required is done in parallel. Hence for our SBC design a backward adaptive quantization scheme (AQ8) has been chosen.

For the application of DPCM the following has to be taken into account. With DPCM the complexity of the coder algorithms, especially when the prediction is made adaptive, increases. Furthermore, as there is little or no correlation between the subband outputs [2], the predictor coefficients will be close to zero [26]. So DPCM will perform little or no quality improvement, while the complexity increases. For these reasons it has been decided to use "simple" PCM coding without prediction.

We will now discuss the backward adaptive PCM method, chosen for the (de-)coding of the subband signals, in more detail. See also [6, 7, 15, 26].

A general scheme of a PCMAQ8 coder is depicted in Fig. 18.
For the quantization of the input signal $x(n)$ a quantizer with a midriser characteristic, as shown in Fig. 19, is used. As can be seen from this figure, a value of $x(n)$ in the range $[I(n)\Delta(n); (I(n)+1)\Delta(n)]$ results in a quantized value $R(n)=(I(n)+0.5)\Delta(n)$, where:

$$
\Delta(n)=\text{stepsize (i.e. spacing between quantized levels)}
$$

at moment $n$.

$$
I(n)=\text{two's complement value representing one out of }2^B
$$

quantization levels, i.e. the $B$-bit code word, at moment $n$.

The output of the quantizer is the $R(n)$ representing code word $I(n)$. The stepsize adaption strategy used is based on the one-word stepsize memory approach proposed by Jayant, Flanagan and Cummiskey [25]. The robust stepsize adaption is based on the relation:

$$
\Delta(n)=[\Delta(n-1)]^y \times M(\|I(n-1)\|) \tag{5}
$$

where $\Delta(n)$ is the stepsize used for the encoding at the $n$th time sample and $\Delta(n-1)$ is the stepsize that was used for the $(n-1)$th time sample. The value of $\Delta(n-1)$ is raised to a power $y$, where $y<1$ is a coder parameter (to be discussed in more detail in 5.3.). It is then multiplied by a (positive) scale factor $M(.)$ which is a function of the previous code word $I(n-1)$ to give the stepsize estimate $\Delta(n)$. In general,
if the previous code word $I(n-1)$ indicates that an upper (absolute) quantizer level was used in encoding, a value of $M(.) > 1$ is used to increase the size of the new stepsize $\Delta(n)$. If $I(n-1)$ indicates that a lower (absolute) amplitude level was used by the quantizer, a value of $M(.) < 1$ is used to reduce the estimation of the new stepsize $\Delta(n)$. Thus the stepsize adaption algorithm is constantly attempting to adjust the stepsize $\Delta(n)$ such that it tracks the rms level of the signal and scales the quantizer characteristic to span the amplitude range of the signal. For practical reasons the stepsize must be limited to the range:

$$\Delta_{\min} \leq \Delta(n) \leq \Delta_{\max}$$  \hspace{1cm} (6)$$

to prevent $\Delta(n)$ to grow beyond the limits of the number representation adopted. The ratio $\Delta_{\max}/\Delta_{\min}$ determines the dynamic range that the coder can handle. In our case a ratio of 2048, $\approx 66$ dB, is taken, being within the range of the digital arithmetic. The actual values of $\Delta_{\min}$ and $\Delta_{\max}$ must be different for each subband (each subband signal is coded
with its own PCMAQB coder), to match properly the dynamic range characteristics of the coders to that of the subband signals, as shown in Fig. 15.

The proportion of the amplitude range that is spanned by the quantizer at a particular time (i.e. for a particular stepsize $\Delta(n)$) determines its "loading". If the range of the quantizer is too small relative to the signal range, the quantizer will overload and clip the signal. If it is too large, the quantizer stepsize will be too large, and this will result in an excessive quantization error or noise (often referred to as granular noise). Thus the proper loading of the quantizer is an important factor in maintaining a good reproduction of the signal. The loading is controlled by the choice of the parameters $\gamma$ and $M(.)$.

In the receiver the same stepsize adaptation algorithm is used as in the transmitter, since this adaptation takes place according to the coded subband samples. A block diagram of a PCMAQB decoder is depicted in Fig. 20. By consequence of the midriser characteristic of the dequantizer an input $I(n)$ results in the dequantized value $\hat{x}(n)=(I(n)+0.5)\Delta(n)$.

\[ I(n) \rightarrow \text{B-bit dequantizer} \rightarrow \hat{x}(n) \]

Fig. 20: Block diagram of a PCMAQB decoder.

5.3. Implementation of the chosen (de)coder algorithm

By taking the logarithm, eq. (5) can be written in the form

\[ d(n)=\gamma d(n-1)+m(\lfloor I(n-1)/\Delta(n) \rfloor) \]  

(7)

where:
\[ d(n) = \log \Delta(n) \]
\[ m(.) = \log M(.) \]

The stepsize adaptation is then implemented by the circuit shown in Fig. 21.

\[ \text{Fig. 21: Step size adaption circuit.} \]

The first table lookup converts values of \( I(n-1) \) to \( m(\|I(n-1)\|) \), and the second table lookup realizes the exponential conversion from \( d(n) \) to \( \Delta(n) \). However, as for the quantization the value \( 1/\Delta(n) \) is needed, in the transmitter an exponential conversion from \( d(n) \) to \( 1/\Delta(n) \) has been implemented. Thus it is seen that the adaption circuit consists of two table lookups and a first-order recursive digital filter which can easily be implemented. The advantage of the method of using table lookups is, that it can be done relatively fast and it can perform functions difficult to calculate in the DSP.

An extra dc input \( (1-\gamma) d_c \) is also applied to the circuit in Fig. 21, and it is used to control the loading of the quantizer. Thus eq. (7) has been modified to the form:

\[ d(n) = \gamma d(n-1) + m(\|I(n-1)\|) + (1-\gamma) d_c \]

where for \( d_c \) has been chosen the practical value of approximately

\[ d_c \approx \log(\Delta_{max}/10) \]
Now it is the right moment to discuss the parameter $\gamma$. The adaptation leakage factor $\gamma$, chosen to be $\gamma = 0.99$, forces the realignment of the stepsizes between transmitter and receiver after channel errors occur. Realignment will also occur when the stepsize reaches its maximum or minimum value according to eq. (6), even if $\gamma$ were chosen to be 1. However, a long time may pass before this maximum or minimum is achieved. Since the cancellation of aliasing in the QMF bank depends strongly on the exact tracking of the stepsizes in each subband, it is therefore preferable to use a value of $\gamma < 1$ to dissipate any effects of channel errors more rapidly. Another effect of the leakage due to $\gamma$ is that the log of the stepsize, that is $d(n)$ in Fig. 21, tends to decay to zero in the absence of the inputs $m(\|I(n-1)\|)$ and $(1-\gamma)d_\epsilon$. By adding the term $(1-\gamma)d_\epsilon$ the stepsize toward which $d(n)$ decays can be set to any arbitrary level.

We will now discuss the implementation of the table lookups (stored in program memory) in more detail. In the first table lookup $I(n-1)$ is converted to $m(\|I(n-1)\|)$ according to:

$$ I(n-1) \rightarrow \|I(n-1)\| \rightarrow M(\|I(n-1)\|) \rightarrow m(\|I(n-1)\|) \quad (10) $$

For each subband the same table is used. An appropriate base for taking the logarithms has been found to be $10^9$. Choices for the values of $M(.)$ for different numbers of bits to which the input signal has to be quantized are depicted in Table 4. Experiments have shown that small deviations from the presented values have very little influence on the performance of the adaptive coder. The optimal representation format for the $m(.)$ values is the Q18 format. The exponential table lookup is not the same for each subband. Each subband has its own dynamic range spanned up by $\Delta_{\text{min}}$ and $\Delta_{\text{max}}$. The ratio $\Delta_{\text{max}}/\Delta_{\text{min}}$ is the same (2048) for each subband. The values chosen for $\Delta_{\text{max}}$ and $\Delta_{\text{min}}$, which have been adapted to the dynamic ranges of the subband signals, are depicted in Table 5. The six different tables have been
obtained by taking the logarithm of $\Delta_{\text{max}}$ and $\Delta_{\text{min}}$ and dividing the range between $d_{\text{min}}$ and $d_{\text{max}}$ into 127 uniformly spaced parts, resulting in 128 entries (convenient for our purpose). For each entry the exponential conversion to $A$ (in the transmitter to $1/A$) has been stored in the table. The tables, created in this way, can be implemented very efficiently. If each table is represented in its own optimal format, only one table with 128 entries is needed to represent all six tables.

---

**Table 4: PCM ADPCM coder parameters.**

<table>
<thead>
<tr>
<th>B=</th>
<th>4</th>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>0.9</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>$M_2$</td>
<td>0.9</td>
<td>1.0</td>
<td>1.9</td>
</tr>
<tr>
<td>$M_3$</td>
<td>0.9</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>0.9</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>$M_5$</td>
<td>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_6$</td>
<td>1.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_7$</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_8$</td>
<td>2.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Table 5: $\Delta_{\text{min}}$ and $\Delta_{\text{max}}$ for different subbands.**

<table>
<thead>
<tr>
<th>subband</th>
<th>$f$(Hz)</th>
<th>$\Delta_{\text{min}}$</th>
<th>$\Delta_{\text{max}}$</th>
<th>format for $\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0-500</td>
<td>4.882813E-4</td>
<td>1</td>
<td>Q15</td>
</tr>
<tr>
<td>2</td>
<td>500-1000</td>
<td>4.882813E-4</td>
<td>1</td>
<td>Q15</td>
</tr>
<tr>
<td>3</td>
<td>1000-1500</td>
<td>2.441406E-4</td>
<td>0.5</td>
<td>Q16</td>
</tr>
<tr>
<td>4</td>
<td>1500-2000</td>
<td>2.441406E-4</td>
<td>0.5</td>
<td>Q16</td>
</tr>
<tr>
<td>5</td>
<td>2000-2500</td>
<td>3.051758E-5</td>
<td>0.0625</td>
<td>Q19</td>
</tr>
<tr>
<td>6</td>
<td>2500-3000</td>
<td>3.051758E-5</td>
<td>0.0625</td>
<td>Q19</td>
</tr>
<tr>
<td>7</td>
<td>3000-3500</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>3500-4000</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
For example, $\Delta_{\text{min}}$ and $\Delta_{\text{max}}$ for subband 1 are optimally represented in Q15 format, resulting in the integer values to be implemented of $16$ resp. $32767$. For subband 5 $\Delta_{\text{min}}$ and $\Delta_{\text{max}}$ are optimally represented in Q19 format, which also results in the integer values of $16$ resp. $32767$. The same holds for intermediate values of $\Delta$. So for each subband this table has to be interpreted with its own representation format and its own input range $d_{\text{min}}$ to $d_{\text{max}}$. The entries of this table, $d(n)$, are for each subband optimally represented in Q14 format. So this way of implementing allows us to work with six exponential table lookups, matched to the dynamic ranges of the different subband signals, for the price of only one.

Since the six PCMAQB coders, as mentioned above, have a lot in common, they have been all brought together in one PCMAQB coder subroutine, that in some points is passed through differently for the different subband signals. To accomplish this, tables in program memory are used for storing constants as $(1-\gamma)d_{\Delta}$, $d_{\text{min}}$, $d_{\text{max}}$ and the bit-assignments, and for updating variables as $d(n)$ and $l(n)$. These constants and variables are different for each subband and therefore each table contains one entry for each subband. To synchronize with the rest of the SBC program, again the variable PATH is used to determine the subband to be coded. The same holds for the PCMAQB decoders in the receiver.

The essential points in the coding and decoding subroutines, see appendix D resp. G, are discussed below. For a listing of the table lookups is referred to the program memory initializations in appendix B and F.

After the QMF routine, in the transmitter, has delivered a subband sample to the PCMAQB coding routine, the following takes place. First the subband sample is identified using PATH (=0 to 7). This means reading in the number of code bits, dictated by the voicing strategy in use, and reading in the dynamic range (i.e. $d_{\text{min}}$ and $d_{\text{max}}$) for $d(n)$. Then the stepsize adaption algorithm is executed to determine the new
value of $1/\Delta(n)$ to quantize with. This is done by reading in the previous coded result for that subband, $I(n-1)$, followed by the determination of $m(I(n-1))$, $(1-\gamma)d_n$ and $\gamma d(n-1)$. Adding the former three values according to eq. (8) will lead to the new value of $d(n)$. This new value of $d(n)$ is checked on lying within the dynamic range spanned by $d_{\text{min}}$ and $d_{\text{max}}$; if not, saturation to $d_{\text{min}}$ or $d_{\text{max}}$ takes place. The remaining value of $d(n)$ is saved in its table in program memory. Then $d(n)$ is rounded to one of the entry points of the exponential table lookup, ultimately leading to $1/\Delta(n)$. Having this value of $1/\Delta(n)$ the quantization takes place by multiplying it with the subband sample delivered by the QMF routine and taking the integer part of the product. According to the midriser characteristic of the quantizer this results in the two's complement code word $I(n)$. After checking the range of it, dictated by the number of code bits, and possible saturation the code word $I(n)$ is saved in its table in program memory. Later the multiplexer, to be discussed in chapter 6, will use this table of code words $I(n)$ to create a 16 kbit/s data stream.

In the receiver, the demultiplexer will recover the table of code words $I(n)$ from the 16 kbit/s data stream. This table will be used by the PCMAQB decoder to reconstruct the uncoded subband signals. When this decoding routine is called, first the stepsize $\Delta(n)$ is determined, in the same way as described for the transmitter. Then the subband sample to be decoded, according to PATH, is read in. After adding 0.5 to this value of $I(n)$, multiplication with $\Delta(n)$ takes place, resulting in a dequantized subband sample according to the midriser characteristic. After checking the range of this reconstructed subband sample and possible saturation, it is delivered to the inverse QMF bank for further processing. For more details about the coding and decoding algorithm is referred to the source files, listed in appendix D and G.
5.4. Tests and results

For testing the implemented PCMAQB coders and decoders, again the test configuration as shown in Fig. 13 has been used. Only the DSP programs used in this stage differ from those used in Fig. 13. The main routine in the transmitter DSP has been extended with the PCMAQB coding module. After the QMF program has finished, the PCMAQB program is passed-through. Then the wait cycle follows. The same holds for the main routine in the receiver DSP. However, here the PCMAQB decoding is passed-through before the inverse QMF program is called. The configuration of the DSP programs, also including the extensions for the semi-adaptive bit-allocation, is sketched in Fig. 22.

![Fig. 22: DSP configuration for testing the PCMAQB coders and decoders.](image)

Again tests with speech signals have been carried out, which is necessary due to the adaptive schemes in the SBC algorithm. In all tests the analog speech input was applied to the transmitter system input, and listening to the recon-
structured speech signal took place via the audio output of the receiver system. For some kind of objective checking, the reconstructed speech signal was also visualized on a memory oscilloscope. Furthermore, the reconstructed speech quality could easily be compared with the maximum quality that can be achieved (up to 3 kHz bandlimited speech quality, see 4.4.) by skipping the PCMAQ8 coding and decoding (by means of a hardware switch connected with the DSP's).

5.4.1. Performance of the coders and decoders

To confirm the proper working of the coding and decoding itself, the influence of the bit-allocation assignments was eliminated by coding each subband signal with 4 bits. This was allowed, by the fact that the communication between transmitter and receiver system still was performed in a parallel way and not serial. The performance of this 24 kbit/s SBC was rather good. The reconstructed speech differed only very little when compared with uncoded, up to 3 kHz bandlimited, speech.

The loading of the (de-)quantizers was subjectively checked by switching off (in software) the saturation actions in the coding and decoding, since many saturation actions indicate an improper loading of the (de-)quantizers. This resulted in some quality degradation, but not so much as to conclude that the (de-)quantizers were improperly loaded.

Also, the proper loading of the (de-)quantizers has been confirmed objectively, as the behaviour of the reconstructed speech signal showed no excessive clipping on the memory oscilloscope. Finally, also the correctness of the code words in the tables of the transmitter and receiver programs was verified.

After the proper working of the coder and decoder principle had been confirmed in this way, the coding into 16 kbit/s, i.e. the influence of a bit-allocation different from 4 bits per subband sample, could be tested.
5.4.2. Performance of the chosen bit-allocation method

First the semi-adaptive bit-allocation method with the assignments as shown in Table 3, (eliminated in the former testing) was restored. Then the same tests, as described in 5.4.1., have been carried out. The intelligibility of the reconstructed speech was fairly satisfactory. However, a quality degradation, compared to uncoded up to 3 kHz bandlimited speech, was noticeable. Furthermore, tests with several listeners have been carried out to compare the semi-adaptive bit-allocation with fixed-bit-allocation and also to try out other bit assignments. Notwithstanding the fact, that in some cases differences were difficult to perceive, the semi-adaptive bit-allocation with the assignments from Table 3 appeared to provide the best overall performance. However, the difference with fixed-bit-allocation was less than expected from the argument given in 4.2. Finally the loading of the (de-)quantizers has been investigated, which proved to be proper. This is not only a consequence of a convenient stepsize adaption algorithm but also due to an appropriate representation of the subband signals (see 3.5.) and the use of "six different" exponential table lookups (5.3.). In the development work, preceding the ultimate realization of the coding and decoding described above, also tests have been carried out, where all subband signals were represented in the same Q15 format and were coded using the same exponential table lookup for each subband signal. This resulted in a reconstructed speech signal that excessively suffered from audible clipping and "clicks" due to an improper loading of the (de-)quantizers. Also a dynamic range for the exponential table lookups smaller than 66 dB has been tried out, resulting again in an improper loading of the (de-)quantizers. By the way, due to this dynamic range, the different values of \( \Delta_{\text{min}} \) are such, that in periods of silence \( (I(n)=0 \text{ and } \Delta(n)=\Delta_{\text{min}}; \text{reconstructed subband sample } = (0+0.5) \cdot \Delta_{\text{min}}^{1/2} \) \( \Delta_{\text{min}} \) causes no audible output upon reconstruction.
5.5. Conclusion

The 16 kbit/s coding and decoding of the subband samples, using 2, 3 or 4 bit PCMAQB and a semi-adaptive bit-allocation algorithm, results in adding an amount of quantization noise to the replica of \( s(n) \) (3.6.). However, the perceived quality of the reconstructed speech signal, especially the intelligibility, is considered acceptable for our applications. At this stage the actual Subband Coding and Decoding has been realized. The only thing left to be described is the realization of the multiplexing and demultiplexing to perform the serial 16 kbit/s communication.
6. MULTIPLEXING AND DEMULTIPLEXING OF THE CODED SUBBAND SAMPLES

6.1. Introduction

To create, after quantization and coding, the ultimate serial 16 kbit/s data stream, the subband signals together with the side information and synchronization bits are multiplexed. This multiplexing has to be performed in a controlled way; by demultiplexing it has to be possible to recover the coded subband samples, the side information and the synchronization bits. This can be accomplished by multiplexing the data to be transmitted into a repetitive framed sequence.

Important features, coupled with the multiplexing and demultiplexing are:

- Synchronization between transmitter and receiver, i.e. frame alignment.
  i) The time for alignment with 99% probability must not seriously disrupt the speech communication. It is usually a compromise between the time taken to confirm the presence of the framing pattern and the risk of incorrectly aligning to random imitations of it.
  ii) False indications of lost frame alignment for an error rate of ca. 1:10³ must not occur too frequently.
- Frame organization.
  The composition of a frame in the 16 kbit/s serial data stream has to match the framing, i.e. the subband processing sequences and the voicing strategy changes (Fig. 17) in the "actual" (without (de)multiplexing) Subband Coder and Decoder algorithms. This to avoid excessive buffering.

The bit synchronization in the demultiplexer is assumed to be located in a (higher order) part outside the Subband Decoder algorithm, where the demultiplexing of the Subband Coder signal (16 kbit/s) and another signal (see 1.) has to be done. Therefore the bit synchronization will not be described here.
In this chapter the design, implementation and testing of the multiplexer and demultiplexer for our SBC will be discussed.

6.2. Design considerations

Based upon recommendations from literature [3,27] and previous work [6,8,28] it has been decided to use the following strategies for the framing and synchronization.

To acquire frame alignment, when the first bits are received or alignment has been lost, it is for the demultiplexer to search for and recognize the Frame Alignment Word (FAW), present in a fixed position in each frame. Then the demultiplexer has to lock its timing counters into the correct phase relationship with the incoming signal, and examine the FAW in two successive confirmatory frames. Non-recognition of the FAW in its expected position in either of these two frames causes a recommencement of the search. This is done to safeguard against false alignment by imitations of the FAW within the signal.

The minimum time that is necessary to acquire and confirm frame alignment is, therefore, between 2 and 3 frame periods, depending on the point within the frame that a valid signal is applied and the search commences. However, the incoming signal contains an essentially random occurrence of ones and zeros in the information digit time-slots, and there is a probability of these digits imitating the FAW. The probability of an imitation in any position is $2^{-m}$ for a random bit stream, $m$ being the number of bits in the FAW. Such imitations of the FAW will lead to an increase of the time required to acquire frame alignment, due to the greater number of false starts; the false alignment being rejected at the first or second confirmatory frame.

The time, $t_i$, to acquire frame alignment with a 99% probability of not being exceeded can be estimated using the following simplified formula for the alignment strategy described above:
where:

$F$ is the total number of bits in a frame of the multiplexed signal.

$m$ is the number of bits in the FAW.

To avoid false alarms due to bit errors as much as possible, frame alignment is considered to have been lost when 4 consecutive FAW's are incorrectly received in their predicted positions. The probability of random digital errors causing this condition to be fulfilled is approximately $(mp)^4$ when $p$, the bit error rate, is low; better than, say, $1:10^4$. It follows that the mean time between false losses of frame alignment for a given error rate is,

$$t_i = F / (16(mp)^4) \text{ milliseconds}$$

To meet the requirements i) and ii) in 6.1., $t_i$ has to be small (order of ms) and $t_L$ has to be large (order of hours). Appropriate values for the frame length $F$ and the FAW length $m$ have been chosen to be 128 resp. 8 bits. In that case the frame duration is 8 ms and the information bit rate is 15 bits per ms. These choices have already been used in designing the bit-allocation and synchronization for the "actual" Subband Coding and Decoding (chapter 4).

With respect to the organization of a frame, the following can be said. As mentioned above, for synchronization reasons the first 8 bits of such an 128 bits frame contain a FAW. As the side information, required for our SBC design, only concerns the bit-allocation (3 possibilities) used for the coding of the subband samples during a whole frame period, this is inserted in the FAW. So three FAW's are possible, each being composed in a special way to reduce the probability of imitating it:

$$t_i = F / (16(mp)^4) \text{ milliseconds}$$

where:

$F$ is the total number of bits in a frame of the multiplexed signal.

$m$ is the number of bits in the FAW.
The remaining 120 bits are all used for transmission of the coded subband samples. These 120 bits are filled in, taking into account the way of subband processing in the "actual" SBC, see Fig. 17.

6.3. Implementation of the chosen multiplexer and demultiplexer algorithm

Before describing the implementation of the multiplexer and demultiplexer themselves, we will discuss the application of the 16 kHz clock signal, needed to create a 16 kbit/s bit stream. As discussed earlier, the "actual" SBC algorithms are controlled by an 8 kHz interrupt signal, updating the timing variables PATH and FRAME. Therefore, to complete the overall SBC system, the DSP programs have been adapted as follows.

The transmitter DSP is controlled by interrupts, from now on a 16 kHz clock signal. Each 16 kHz interrupt created by the A/D converter (see chapter 7) causes an execution of the interrupt service routine. In this interrupt service routine a sample of $s(n)$ is read in, followed by a call of the multiplex subroutine to create one bit to be written out. The main routine is passed-through for every other 16 kHz interrupt (let say the odd ones), realizing the same 8 kHz control as before. In this main routine, thus once interrupted by a 16 kHz interrupt, the QMF filtering and PCMAQD coding is carried out, followed by a wait cycle for the next odd interrupt.

The receiver DSP program is controlled by 16 kHz interrupts from the regained system clock, for which again the transmitter system clock has been used. In the interrupt service routine the demultiplex subroutine is called to read in one bit from the 16 kbit/s stream and furthermore a reconstructed speech sample, $\hat{s}(n)$, is written out. In the main routi-
ne, to be passed-through for every other interrupt, the PCMAQ8 decoding and the QMF reconstruction takes place, followed by a wait cycle for the next odd interrupt. However, when the receiver is not in alignment $g(n)$ is set to zero, without calling the PCMAQ8 decoding and QMF reconstruction routines.

Furthermore, considering the 16 kHz transmitter input, $s(n)$, only one out of two inputs is used in the main routine. In the receiver, the reconstructed output from the inverse QMF filtering is a sample sequence with a rate of 8 kHz. To reduce the aperture effect [14, pp. 302-304] this sequence is interpolated to obtain the ultimate output signal, $g(n)$, with a sampling rate of 16 kHz.

To accomplish the whole synchronization, beside the variables PATH and FRAME a variable called STATE is used. STATE is a modulo-16 counter, being updated for every 16 kHz interrupt. So, for every odd value of STATE, PATH is updated and the main routine (in transmitter or receiver) is passed-through. As discussed in 4.3., FRAME is updated for every sequence of PATH from 0 to 7, which has to be done eight times for one frame period in the "actual" Subband Coding or Decoding.

Now the multiplexer and demultiplexer implementations are discussed in more detail.

6.3.1. Multiplexer

Apart from timing considerations, the multiplex algorithm is not very complicated. Each time the multiplex routine is called, a variable is checked to determine the sort of information, i.e. from the FAW or a certain subband, to be transmitted. A second variable is checked to determine which bit from that FAW or subband has to be transmitted. The transmission of each bit is done via a send buffer. Each time before the first bit of a FAW or coded subband sample has to be transmitted, the send buffer is loaded with that FAW or subband sample code word. The FAW is retained from
the bit-allocation decision network, while the coded subband samples are retained from the "I(n) table" (see 5.3.) in program memory. The variable that indicates the bit to be transmitted is updated for each bit sent. The variable that indicates the sort of information to be transmitted is updated after the last bit of the send buffer has been sent.

To match the way of subband processing in the QMF splitting and PCMAQB coding (Fig. 17) first the bits from subband 1 are sent, followed by sending the bits from resp. subband 4, 5, 2, 3 and 6. This sequence is repeated eight times for one frame. The sending of these eight sequences is preceded by sending the FAW of the frame. Furthermore, to ensure that each subband sample is retained from the "I(n) table" on the right moment (i.e. after it has been created and before the next sample of that subband is created) the multiplexer frame periods are delayed three 16 kHz interrupts (being the minimum possible) with respect to the frame periods in the "actual" transmitter SBC. The subband processing in the QMF splitting and PCMAQB coding for one frame period, and its corresponding multiplexer frame for the three voicing strategies resp. voiced, intermediate and unvoiced, are depicted in Fig. 23. In this figure, the numbers in the upper line represent the subband processed in the QMF splitting and PCMAQB coding between two odd 16 kHz interrupt service routines. The numbers in the next line represent the subband

![Frame composition in multiplexer with respect to a subband processing frame.](image-url)
from which a bit is sent in a 16 kHz interrupt service routine for the voiced bit-allocation strategy; a "F" represents a FAW bit. The same holds for the third and fourth line, but then for the intermediate resp. unvoiced bit-allocation strategy. The source file of the multiplex subroutine is listed in appendix E.

6.3.2. Demultiplexer

The demultiplex algorithm is more complex, as it also has to take care of the frame alignment. Each time the demultiplex routine is called, it is determined whether the receiver is in alignment or not. When the receiver is not in alignment, either a search for a FAW takes place or, when a FAW already has been detected, the confirmatory stage is executed. For the FAW search, a received bit is inserted in the receive buffer (a shift register), which is subsequently checked on containing a FAW. If a FAW is detected, then in the next demultiplex call jumping to the confirmatory stage takes place, otherwise again a FAW search is executed. In the confirmatory stage, a received bit is inserted in the receive buffer and, if it is the moment to expect a FAW (128 bits after the previous FAW), the receive buffer is checked on containing a FAW. In the demultiplex calls, jumping to this confirmatory stage repeats until alignment is definitively confirmed or denied. When denied, the next demultiplex call concerns a FAW search. When confirmed, the receiver timing is set by loading the synchronization variables STATE, PATH and FRAME with the appropriate values, and in the next demultiplex call demultiplexing takes place according to the "in alignment situation".

When the receiver is in alignment, the inverse of the multiplex algorithm is executed. Again two variables are used to determine which bit for which subband or FAW is to be received. The received subband samples are stored in the "I(n) table" (see 5.3.) in program memory. The side information
retained from a received FAW is delivered to the voicing strategy buffer. The only action without a corresponding action in the multiplexer, is the alignment check when a FAW is received. When 4 consecutive FAW's are incorrectly received in their predicted positions, the "in alignment situation" is left and the next demultiplex call results in a FAW search.

The demultiplexer frame for the three possible voicing strategies, matching the subband processing in the PCMAQB decoding and QMF reconstruction for a corresponding frame period, is depicted in Fig. 24. As the subband samples first have to be received before they can be processed, in this figure the numbers in the upper three lines represent the subband or FAW (F) for which a bit is received for the resp. bit-allocation strategies: voiced, intermediate and unvoiced. The numbers in the last line represent the subband processed in the PCMAQB decoding and QMF reconstruction ("actual" Subband Decoding). As can be seen from this figure, to process each subband sample (retained from the "I(n) table") on the right moment, the frame periods in the "actual" Subband Decoding are delayed eleven 16 kHz interrupts (being the minimum possible) with respect to the demultiplexer frame periods.

The source file of the demultiplex subroutine is listed in
6.4. Experimental confirmation of the proper working of the multiplexer and demultiplexer

With the configuration from Fig. 13, the 8 kHz A/D and D/A conversions replaced by 16 kHz A/D and D/A conversions and the DSP programs updated to perform the functioning depicted in Fig. 25, experiments have been carried out. Again analog speech was applied to the transmitter system input and the audio output of the receiver system was used to listen to the reconstructed speech.

After starting up the transmitter and receiver program, frame alignment in the receiver was acquired very fast (hardly to perceive). When in alignment, the perceived quality obtained with this “serial 16 kbit/s SBC” was exactly the same as that obtained for the “parallel 16 kbit/s SBC” (5.4.2.). Furthermore, proper functioning in situations that cause loss of frame alignment and frame realignment has been
confirmed by imitating these situations (interrupting the communications and/or synchronization).

Also, for these situations, the necessity of an adaption leakage factor $\gamma < 1$ in the PCMAQB (de-)coding, eq. (8), has been proved. Choosing this factor equal to 1 resulted in a lasting mis-adaption between the PCMAQB coding and decoding stepsizes, leading to unintelligible reconstructed speech.

So, also the multiplexer and demultiplexer implementations may be considered to function properly. With that, as these were the last modules to be added, the description of the SBC realization concerning the DSP implementations has finished. Therefore Fig. 25 is the block diagram of the implemented Subband Coder and Decoder. All the necessary source files are listed in appendix B to E for the transmitter DSP, and in appendix F to I for the receiver DSP.
7. HARDWARE_CONSIDERATIONS

7.1. Introduction

So far the hardware environment of the DSP's, also necessary for a proper working SBC realization, has been left undis­
cussed.
In this chapter we will discuss the essentials of the band­
limiting, the A/D and D/A conversion and furthermore the serial transmission from transmitter to receiver DSP.
For detailed information about the A/D and D/A converters, and for a hardware scheme of the AlB (pin connections etc.) is referred to the AlB book [11].

7.2. I/O_of_the_transmitter_DSP

The 16 kHz 12-bit linear PCM digital input, $s(n)$, to the transmitter DSP, and the 16 kHz interrupt signal that con­
trols the DSP program (6.3.) are created in the following way.
The analog speech signal, $s(t)$, is bandlimited, for which in our case a standard telephone-channel filter (appendix A) has been used. Then the signal is applied to J2 of the AlB, which is via a sample-and-hold-circuit the A/D converter input. Via the software in the DSP program, this A/D conver­
ter is programmed to operate with a sampling rate of 16 kHz, and to deliver beside the $s(n)$ signal also a 16 kHz inter­
rupt signal to the DSP. In short the A/D converter works as follows. Upon receipt of a start-of-conversion signal (pro­
grammed to be a 16 kHz clock signal) the A/D conversion starts, and after the conversion has been completed, an end­
of-conversion signal is created and applied to the interrupt pin of the transmitter DSP. This causes an interrupt (16 kHz) to read in the A/D converted sample and to control the DSP program. The end-of-conversion signal is also used for taking a new sample in the sample-and-hold-circuit. Further­
more, the analog speech signal is supposed to match the dynamic range of the A/D converter, which can be achieved by
means of an amplifier.

The serial 16 kbit/s output stream from the transmitter can be retained from pin 38 of port P1 (a buffered AIB output port). This port P1 is programmed via the DSP program to operate in the sample delay mode, to ensure periodicity in the 16 kbit/s output stream. This means that an output bit from the transmitter DSP is first stored in a primary buffer. By means of a pulse from the same 16 kHz start-of-conversion signal as mentioned above, it is transferred to a secondary buffer, being the interface with the "outside world". As the primary buffer is filled via the DSP program far before a start-of-conversion pulse occurs, it always contains stable data on the moment of transfer.

7.3. I/O_of_the_receiver_DSP

In the receiver system, the clock signal that controls the DSP program is not created on the AIB itself as is the case for the transmitter system. In the receiver system an externally regained 16 kHz system clock (to be discussed in 7.4.) controls the synchronization.

The serial 16 kbit/s input stream to the receiver system has to be delivered to pin 12 of port P1. Via the DSP program this AIB input port is programmed to operate in the asynchronous receive mode. By means of a pulse from the regained system clock a bit from the input stream is clocked into an input buffer. As the regained system clock is applied to the interrupt pin of the DSP, this pulse also creates an interrupt to read in the buffered input bit and to control the DSP program.

The 16 kHz digital output samples, $x(n)$, from the receiver DSP are delivered to the D/A converter that is programmed to operate in the transparent mode. Each sample is directly D/A converted without double buffering. Next, the D/A converted signal is bandlimited with a up to 4.7 kHz bandlimiting filter resident on the AIB. Finally, the analog output is available via J3, or, when an audio output is desired, via J4 (AIB outputs).
7.4. **Communication between transmitter and receiver system**

As mentioned in 6.1., in the receiver the system clock is assumed to be regained in a (higher order) part outside the subband decoder algorithm. Therefore, for our (testing) purposes this clock signal has been retained from the transmitter system (hard-wired clock). The signal taken, is the 16 kHz end-of-conversion signal. An end-of-conversion pulse occurs ca. 25 μs after a start-of-conversion pulse used in the transmitter system to clock-out a bit of the 16 kbit/s stream. So, when this end-of-conversion signal is used as a regained system clock in the receiver, always stable data are clocked into the input buffer of the receiver system. Furthermore in this way the transmitter and receiver DSP programs are controlled by the same 16 kHz clock.

In the transmitter system, the end-of-conversion signal is retained from pin 9 of U40 on the AIB, and is via a 50 Ω line driver (SN745140N) applied to a BNC connector. In the receiver system, this end-of-conversion signal is via a BNC connector applied to pin 13 of port P1.

If both pin 38 of port P1 in the transmitter system and pin 12 of port P1 in the receiver system have been connected with a BNC connector too, the communication between transmitter and receiver system is realized by means of two coaxial cables. One for the 16 kHz system clock and one for the 16 kbit/s serial data stream.

With a frequency counter the system clock frequency has been measured to be 15.974 kHz.

7.5. **Outline of the hardware configuration that realizes a 16 kbit/s Subband Coder and Decoder**

In Fig. 26 a simple diagram of the SBC hardware and its connections is shown. Furthermore, for the AIB's, to perform the functioning as described in this chapter its jumpers have to be set according to the settings depicted in Table 6.
Fig. 26: SBC hardware configuration.

<table>
<thead>
<tr>
<th>Jumper Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transmitter</strong></td>
<td></td>
</tr>
<tr>
<td>E1 4-5</td>
<td>Connects J2 input jack to A/D (bypasses filter)</td>
</tr>
<tr>
<td>E2 don't care</td>
<td></td>
</tr>
<tr>
<td>E3 don't care</td>
<td></td>
</tr>
<tr>
<td>E4 not connected</td>
<td>Leaves Vcc pin on target socket open</td>
</tr>
<tr>
<td>E5 2-3</td>
<td>Connects A/D end-of-conversion signal to interrupt pin of TMS32010 emulator socket</td>
</tr>
<tr>
<td>E6 1-2</td>
<td>Connects sample and hold to A/D input</td>
</tr>
<tr>
<td><strong>Receiver</strong></td>
<td></td>
</tr>
<tr>
<td>E1 don't care</td>
<td></td>
</tr>
<tr>
<td>E2 1-2</td>
<td>Connects output filter to J3 output jack</td>
</tr>
<tr>
<td>3-4</td>
<td>Connects D/A converter to output filter</td>
</tr>
<tr>
<td>E3 1-2</td>
<td>Connects analog output to audio amplifier</td>
</tr>
<tr>
<td>E4 not connected</td>
<td>Leaves Vcc pin on target socket open</td>
</tr>
<tr>
<td>E5 2-3</td>
<td>Connects regained system clock to interrupt pin of TMS32010 emulator socket</td>
</tr>
<tr>
<td>E6 don't care</td>
<td></td>
</tr>
</tbody>
</table>
At all stages in the design of the Subband Coder, a prime consideration was the optimum use of the DSP resources. Table 7 illustrates the allocation of processing time for a 0.125 ms period (in which one 8 kHz speech sample is processed) and the utilization of program and data memory. The brackets ( ) indicate the percentage use of the total available resource. Concerning the processing time, the worse case situations are depicted. Table 8 shows the decoder utilization for the inverse processes to restore the original speech signal.

**Table 7: Transmitter DSP utilization**

<table>
<thead>
<tr>
<th>function</th>
<th>processing time</th>
<th>program memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>instruction cycles/0.125 ms</td>
<td>RAM locations</td>
</tr>
<tr>
<td>Initialization</td>
<td>-</td>
<td>391</td>
</tr>
<tr>
<td>Main program</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>Interrupt service</td>
<td>68</td>
<td>34</td>
</tr>
<tr>
<td>QMF subroutine</td>
<td>217</td>
<td>668</td>
</tr>
<tr>
<td>PCMAQB subroutine</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>Multiplex subroutine</td>
<td>80</td>
<td>282</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td><strong>494 (79%)</strong></td>
<td><strong>1501 (37%)</strong></td>
</tr>
</tbody>
</table>

Data memory (number of RAM locations)=125 (87%)

---

B. UTILIZATION OF THE TRANSMITTER AND RECEIVER DSP

---
### Table 8: Receiver DSP utilization

<table>
<thead>
<tr>
<th>Function</th>
<th>Processing time</th>
<th>Program memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>instruction cycles/0.125 ms</td>
<td>RAM locations</td>
</tr>
<tr>
<td>Initialization</td>
<td>-</td>
<td>391</td>
</tr>
<tr>
<td>Main program</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>Interrupt service</td>
<td>74</td>
<td>40</td>
</tr>
<tr>
<td>Inverse QMF subroutine</td>
<td>230</td>
<td>675</td>
</tr>
<tr>
<td>PCMAQB subroutine</td>
<td>117</td>
<td>118</td>
</tr>
<tr>
<td>Demultiplex subroutine</td>
<td>105</td>
<td>396</td>
</tr>
<tr>
<td><strong>total</strong></td>
<td><strong>548 (88%)</strong></td>
<td><strong>1643 (40%)</strong></td>
</tr>
</tbody>
</table>

Data memory (number of RAM locations) = 129 (90%)

Considering processor speed and data RAM utilization, we may conclude that the DSP capacities are almost completely exploited. Using one DSP for the transmitter and one for the receiver, a more complex SBC algorithm can most probably not be realized.
9. SUGGESTIONS FOR (POSSIBLE) IMPROVEMENT OF THE SUBBAND CODER PERFORMANCE

So far, the bandlimiting in the transmitter and receiver occurs via resp. a standard telephone-channel filter and a lowpass filter resident on the AIB. The performance of the SBC will improve when bandlimiting filters, matching the band of interest 200-3200 Hz (4.2.) more properly, are used. These bandlimiting filters can be realized in an analog or digital form. During previous work [7, pp. 45-58] such a digital bandlimiting filter has been developed. However, adding this algorithm to our SBC program will largely exceed the DSP capacity for both transmitter and receiver. And therefore the application of digital bandlimiting filters will require another two DSP's, one for the transmitter and one for the receiver.

One of the reasons for not choosing DPCM for the coding of the subband signals, was the fact that little or no correlation exists between the subband samples. However, according to [29] some correlation exists in the first two subbands (0-500 Hz and 500-1000 Hz). In coding, whitening these two subband signals using DPCM in stead of PCM will probably improve the SBC performance. It is questionable, however, whether the DSP's can cope with the resulting increase of complexity.

Besides the literature referred to so far, also the references [30] to [60] have been studied. Here, some alternative SBC algorithms are presented. However, our SBC implementation still remains the best compromise between quality and complexity for our purposes.
The Subband Coding technique, using DMF filters for the bandsplitting and reconstruction, PCMAQB for the coding and decoding of the subband signals and furthermore applying a semi-adaptive bit-allocation strategy, has proved to be a very appropriate method for the realization of a 16 kbit/s speech coder and decoder on one DSP each. Reviewing the design constraints, the quality performance of the implemented SBC, especially the intelligibility of the reconstructed speech, is fairly satisfactory. The capacities of the DSP's are almost completely utilized. So, also the TMS32010, selected for implementing the SBC algorithm, has showed to be a suitable choice.

For professional applications (i.e. without development systems) in future, the Subband Coder will consist of the following main parts:

- a digital or analog bandlimiting filter (200-3200 Hz).
- an amplifier to accomplish the matching of the dynamic ranges of the input speech signal and the A/D converter.
- a 16 kHz sample and hold circuit.
- a 16 kHz 12 bit linear PCM A/D converter.
- a TMS32010 DSP; program memory is partly present in on-chip or off-chip PROM (burnt-in with the Subband Coder program) and partly in off-chip RAM (for tables and variables, not fixed in time).
- a power supply.

The main parts of the Subband Decoder will be:

- a TMS32010 DSP; program memory is partly present in on-chip or off-chip PROM (burnt-in with the Subband Decoder program) and partly in off-chip RAM (for tables and variables, not fixed in time).
- a 16 kHz 12 bit linear PCM D/A converter.
- a digital or analog bandlimiting filter (200-3200 Hz).
-an audio amplifier+loudspeaker.
-a power supply.

Since both Subband Coder and Decoder will be part of a satellite communication system (see 1.), their 16 kHz system clocks can be retained from the (64 kHz) master clock available in the transmitter and receiver of the satellite link.
REFERENCES


[7] J.H. Bolt,
"Speech coding into 16 kbit/s with subband coding on a
digital signal processor",
Graduation Work Report, Telecommunications division of
the Eindhoven University of Technology, the Netherlands

[8] P.J.J. Chitamu,
Practical Training Report, Telecommunications Division
of the Eindhoven University of Technology, the Nether­
lands, July 1986.

Digital Signal Processor Products, Texas Instruments,
SPRU001A, USA 1983.

Digital Signal Processor Products, Texas Instruments,
SPRU005A, USA 1985.

Digital Signal Processor Products, Texas Instruments,
USA 1984.

[12] TMS32010 Assembly Language Programmer's Guide,
Digital Signal Processor Products, Texas Instruments,
SPRU002B, USA 1985.

[13] TMS32010 Crossware Installation Guide,
Digital Signal Processor Products, Texas Instruments,
SPDU049, USA 1986.

[14] A.B. Carlson,
Communication Systems, second edition,
[15] R.E. Crochiere,  
"On the design of Sub-band Coders for low-bit-rate speech communication",  
_Bi_Si_Ti_I_1_, vol. 56, pp. 747-770, May-June 1977.

[16] R.E. Crochiere,  
"Digital coding of speech in sub-bands",  

[17] R.E. Crochiere, S.A. Webber and J.L. Flanagan,  
"Digital coding of speech in sub-bands",  
_Bi_Si_Ti_I_1_, vol. 55, pp. 1069-1085, Oct. 1976.

[18] A.J. Barabell and R.E. Crochiere,  
"Sub-band coder design incorporating quadrature filters and pitch prediction",  

[19] D. Esteban and C. Galand,  
"Application of quadrature mirror filters to split band voice coding schemes",  

[20] R.E. Crochiere,  
"Digital signal processor: Sub-band coding",  
_Bi_Si_Ti_I_1_, vol. 60, pp. 1633-1653, Sept. 1981.

[21] J.D. Johnston,  
"A filter family designed for use in quadrature mirror filter banks",  

[22] V. Gupta and K. Virupaksha,  
"Performance evaluation of adaptive quantizers for a 16-kbit/s subband coder",  
[23] C. Galand and D. Esteban, 
"16 kbps sub-band coder incorporating variable overhead information",

[24] P. Noll,
"A comparative study of various quantization schemes for speech encoding",

[25] N.S. Jayant,
"Digital coding of speech waveforms: PCM, DPCM and DM quantizers",

[26] J.D. Johnston and R.E. Crochiere,
"An all-digital commentary grade subband coder",

[27] E.R. Brigham, M.J. Snaith and D.M. Wilcox,
"Multiplexing for a digital main network",

[28] L.J.D.C. Voorbraak,
"Synchronisatieschakeling voor de subband coder",
Project Report, Telecommunications Division of the Eindhoven University of Technology, the Netherlands, 25 April 1985.

[29] P. Hamel, J. Sourmagne and A. Le Guyader,
"A new dynamic bit allocation scheme for sub-band coding",
[30] T.A. Ramstad,
"Sub-band coder with a simple adaptive bit-allocation algorithm; a possible candidate for digital mobile telephony?",

[31] C. Galand and D.J. Esteban,
"16 kbps real time QMF sub-band coding implementation",

"Real-time implementation of sub-band coding on a programmable integrated circuit",

[33] D. Esteban and C. Galand,
"Multiport implementation of real time 16 kbps coder",

[34] F.S. Yeoh and C.S. Xydeas,
"Split-band coding of speech signals using a transform technique",

[35] R.E. Crochiere and L.R. Rabiner,
"Recent developments in the design and implementation of digital decimators, interpolators and narrow band filters",

[36] P. Cummiskey, N.S. Jayant and J.L. Flanagan,
"Adaptive quantization in differential PCM coding of speech",
[37] J. Menez, J.F. Galliano and C. Galand,  
"Comparative study of 16 kbps adaptive predictive coders",  

[38] J. Max,  
"Quantization for minimum distortion",  

[39] J.D. Gibson, S.K. Jones and J.L. Melsa,  
"Sequentially adaptive prediction and coding of speech signals",  

[40] T.P. Barnwell,  
"Subband coder design incorporating recursive quadrature filters and optimum ADPCM coders",  

[41] R.W. Schafer and L.R. Rabiner,  
"A digital signal processing approach to interpolation",  

[42] P.C. Millar,  
"Mirror filters with minimum delay responses for use in subband coders",  

[43] M.B. Donvito and B.W. Schoenherr,  
"Subband coding with silence detection",  

[44] F.K. Soong, R.V. Cox and N.S. Jayant,  
"Subband coding of speech using backward adaptive prediction and bit-allocation",  
[45] J.H. Derby and C.R. Galand,
"Multirate subband coding applied to digital speech interpolation",
IEEE_trans_\_ASSP, vol. 4, pp. 43.3.1-43.3.4, 1985.

[46] K.K. Paliwal and T. Svendsen,
"A study of three coders (Subband, RELP and MPE) for speech with adaptive white noise",
IEEE_trans_\_ASSP, vol. 4, pp. 43.5.1-43.5.4, 1985.

[47] J.H. Rothweiler,
"Polyphase quadrature filters - A new subband coding technique",

[48] N.S. Jayant,
"Coding speech at low bit rates",

"New Quadrature mirror filter structures",

[50] C.R. Galand and D.J. Esteban,
"Design and evaluation of parallel quadrature mirror filters",

[51] H.J. Nussbaumer,
"Complex quadrature mirror filters",

[52] P.L. Chu,
"Quadrature mirror filter design for an arbitrary number of equal bandwidth channels",
[53] M.G. Bellanger, G. Bonnerot and M. Coudreuse, 
"Digital filtering by polyphase network: application to sample-rate alteration and filter banks",

[54] P.C. Millar,
"Recursive quadrature mirror filters - criteria specification and design method",

[55] M.J.T. Smith and T.P. Barnwell,
"Exact reconstruction techniques for tree-structured subband coders",

[56] E.F. Deprettere and P. Kroon,
"Compression and quantization of speech",

[57] P.C. Millar,
"A reduced delay 8-band sub-band coder",

[58] M.R. Sambur,
"Speech algorithm advances promise toll-quality medium-band digitized speech",

[59] J.L. Flanagan and others,
"Speech coding",

[60] R.E. Crochiere and L.R. Rabiner,
Multirate digital signal processing,
APPENDIX A

Frequency response of bandlimiting filter
APPENDIX B

Main program SBC transmitter part
THIS MAIN PROGRAM USES THE SUBROUTINES:
- QMF FILTERING
- PCMAGB CODING
- MULTIPLEING

TO REALIZE A 16 KBIT PER SECOND SUBBAND CODER.

SYSTEM CLOCK RATE: 16 kHz

REPORTS FOR DETAILS:
- PT REPORT BY P. CHITAMU
- FINAL REPORT BY J. BOLT
- FINAL REPORT BY R. KLEUTERS

AUTHOR: ROLAND KLEUTERS
DATE: 25-5-1987

****************************************************

DATA MEMORY INITIALIZATION

default page=0

****************************************************

GENERAL VARIABLES DATA MEMORY PAGE 1

CLOCK EQU 0 CLOCK RATE (16 kHz)
MODE EQU 1 MODE FOR ANALOG INTERFACE BOARD
AR00 EQU 2 STORAGE PLACE FOR AUXILIARY REGISTER 0
AR01 EQU 3 STORAGE PLACE FOR AUXILIARY REGISTER 1
ACH EQU 4 STORAGE PLACE FOR HIGH PART OF ACCUMULATOR
ACL EQU 5 STORAGE PLACE FOR LOW PART OF ACCUMULATOR
TREG EQU 6 STORAGE PLACE FOR T REGISTER
STATU EQU 7 STORAGE PLACE FOR STATUS WORD OF TMS32010

from now on default page

GENERAL VARIABLES

MSTAI EQU 0 MASK FOR STATE MOD 16
SWITCH EQU 1 COUNTS FIRST 4 INTERRUPTS; WHEN 0 MUX IS ON (Q0)
FRAME EQU 2 COUNTER FOR FRAME LENGTH (Q0)
ONE EQU 3 CHECK BIT (Q0)
STRAT EQU 4 CONTAINS 0, 1, or 2: VOICED, INTER. or UNYVO. (Q0)
SSTRAT EQU 5 BUFFERING (Q0)
STATE EQU 6 TREE POINTER (Q0)
RESO EQU 7 INTERMEDIATE CALCULATION RESULTS FOR MAIN ROUTINE
RESI EQU 8 INTERMEDIATE CALCULATION RESULTS FOR MAIN ROUTINE
PATH EQU 9 CONTAINS STATE/2=CHANNEL TO PROCESS (Q0)
FIRST QMF STAGE VARIABLES

BUFFER VARIABLES OF LOW1 (0-2 kHz) AND HIGH1 (2-4 kHz)  FORMAT=Q16
A1 EQU 10
B1 EQU 11
C1 EQU 12

INPUT TO (FILTER PART OF) FIRST QMF STAGE  FORMAT=Q15
INPF EQU 13

DELAY VARIABLES OF LOW1 (0-2 kHz)  FORMAT=Q15
IF2 EQU 14  AFTER FILTERING INPUT STORED IN IF2, SO
IF3 EQU 15  IF1 NOT NEEDED
IF4 EQU 16
IF5 EQU 17
IF6 EQU 18
IF7 EQU 19
IF8 EQU 20
IF9 EQU 21
IF10 EQU 22
IF11 EQU 23
IF12 EQU 24
IF13 EQU 25
IF14 EQU 26
IF15 EQU 27
IF16 EQU 28

DELAY VARIABLES OF HIGH1 (2-4 kHz)  FORMAT=Q15
IF18 EQU 29  AFTER FILTERING INPUT STORED IN IF18, SO
IF19 EQU 30  IF17 NOT NEEDED
IF20 EQU 31
IF21 EQU 32
IF22 EQU 33
IF23 EQU 34
IF24 EQU 35
IF25 EQU 36
IF26 EQU 37
IF27 EQU 38
IF28 EQU 39
IF29 EQU 40
IF30 EQU 41
IF31 EQU 42
IF32 EQU 43

SECOND QMF STAGE VARIABLES

BUFFER VARIABLES OF LOW2 (0-1 kHz) AND HIGH2 (1-2 kHz)  FORMAT=Q15
A2 EQU 44
B2 EQU 45
C2 EQU 46
* BUFFER VARIABLES OF LOW3 (3-4 kHz) AND HIGH3 (2-3 kHz)  FORMAT=Q17
  A3  EQU 47
  B3  EQU 48
  C3  EQU 49

* INPUT TO (FILTER PART OF) SECOND QMF STAGE
  INPS  EQU 50

* DELAY VARIABLES OF LOW2,HIGH2,LOW3 AND HIGH3 (TIME SHARING)
  IS2  EQU 51
  IS3  EQU 52
  IS4  EQU 53
  IS5  EQU 54
  IS6  EQU 55
  IS7  EQU 56
  IS8  EQU 57

* BUFFER VARIABLES OF LOW4 (0-500 Hz) AND HIGH4 (500-1000 Hz)  FORMAT=Q15
  A4  EQU 58
  B4  EQU 59
  C4  EQU 60

* BUFFER VARIABLES OF LOW5 (1500-2000 Hz) AND HIGH5 (1000-1500 Hz)  FORMAT=Q15
  A5  EQU 61
  B5  EQU 62
  C5  EQU 63

* BUFFER VARIABLES OF LOW7 (2000-2500 Hz) AND HIGH7 (2500-3000 Hz)  FORMAT=Q18
  A7  EQU 64
  B7  EQU 65
  C7  EQU 66

* INPUT TO (FILTER PART OF) THIRD QMF STAGE
  INPT  EQU 67

* DELAY VARIABLES OF LOW4,HIGH4,LOW5,HIGH5,LOW7 AND HIGH7 (TIME SHARING)
  IT2  EQU 68
  IT3  EQU 69
  IT4  EQU 70
  IT5  EQU 71
  IT6  EQU 72

* VOICING DECISION VARIABLES

ENLOW  EQU 73  ENERGY IN 0-2 kHz BAND (Q14)
ENHIGH  EQU 74  ENERGY IN 2-4 kHz BAND (Q16)
FIRST QMF STAGE COEFFICIENTS  FORMAT=Q16

CF0  EQU  75
CF1  EQU  76
CF2  EQU  77
CF3  EQU  78
CF4  EQU  79
CF5  EQU  80
CF6  EQU  81
CF7  EQU  82
CF8  EQU  83
CF9  EQU  84
CF10 EQU  85
CF11 EQU  86
CF12 EQU  87
CF13 EQU  88
CF14 EQU  89
CF15 EQU  90

SECOND QMF STAGE COEFFICIENTS  FORMAT=Q16

CS0  EQU  91
CS1  EQU  92
CS2  EQU  93
CS3  EQU  94
CS4  EQU  95
CS5  EQU  96
CS6  EQU  97
CS7  EQU  98

THIRD QMF STAGE COEFFICIENTS  FORMAT=Q16

CT0  EQU  99
CT1  EQU  100
CT2  EQU  101
CT3  EQU  102
CT4  EQU  103
CT5  EQU  104

PCMAQB VARIABLES

INPCM  EQU  105  INPUT TO PCMAQB CODER
CALC1 EQU  106  GENERAL VARIABLE TO CALCULATE WITH
CALC2 EQU  107  GENERAL VARIABLE TO CALCULATE WITH
NBIT  EQU  108  CONTAINS NR. OF BITS TO QUANTIZE TO (Q0)
I0    EQU  109  LOWEST POSSIBLE INPUT VALUE OF EXPONENT TABLE (Q14)
IX0   EQU  110  HIGHEST POSSIBLE INPUT VALUE OF EXPONENT TABLE(Q14)
STEP  EQU  111  1/(DISTANCE BETWEEN INPUTS OF EXPONENT TABLE) (Q7)
GAMMA EQU  112  GAMMA VALUE OF FORMULA (0.99) (Q15)
MULTIPLEXER VARIABLES

COUNT EQU 113  COUNT FOR BITS TO SEND
BAND EQU 114  FLAG, CURRENT SUBBAND PROCESSING
DATA EQU 115  STORES OUTPUT BITS FOR S16OUT
S16OUT EQU 116  VARIABLE TO WRITE OUT SERIAL DATA FROM INSIDE
DEMO EQU 117  NEW (FAW or SUBBAND) ACCESS MARK

PROGRAM MEMORY INITIALIZATION

AORG 0
B START  BRANCH TO MAIN ROUTINE
B INTRO  BRANCH TO INTERRUPT SERVICE ROUTINE
AORG 8  OTHERWISE ERRORS WITH TBLW'S

GENERAL CONSTANTS

CCLOCK DATA 312  CLOCK RATE (16 kHz)
MMODE DATA 24  MODE FOR ANALOG INTERFACE BOARD
MMSTAT DATA >000F  MASK FOR STATE MOD 16

SECOND QMF STAGE VARIABLES

DELAY VARIABLES OF LOW2 (0-1 kHz)  FORMAT=Q14
IIS2 DATA 0  AFTER FILTERING INPUT STORED IN IIS2, SO
IIS3 DATA 0  IIS1 NOT NEEDED
IIS4 DATA 0
IIS5 DATA 0
IIS6 DATA 0
IIS7 DATA 0
IIS8 DATA 0

DELAY VARIABLES OF HIGH2 (1-2 kHz)  FORMAT=Q14
IIS10 DATA 0  AFTER FILTERING INPUT STORED IN IIS10, SO
IIS11 DATA 0  IIS9 NOT NEEDED
IIS12 DATA 0
IIS13 DATA 0
IIS14 DATA 0
IIS15 DATA 0
IIS16 DATA 0

DELAY VARIABLES OF LOW3 (3-4 kHz)  FORMAT=Q16
IIS18 DATA 0  AFTER FILTERING INPUT STORED IN IIS18, SO
IIS19 DATA 0  IIS17 NOT NEEDED
IIS20 DATA 0
IIS21 DATA 0
IIS22 DATA 0
IIS23 DATA 0
IIS24 DATA 0
* DELAY VARIABLES OF HIGH3 (2-3 kHz) * 
  IX26 DATA 0 AFTER FILTERING INPUT STORED IN IX26, SO
  IX27 DATA 0 IX25 NOT NEEDED
  IX28 DATA 0
  IX29 DATA 0
  IX30 DATA 0
  IX31 DATA 0
  IX32 DATA 0
 *
* THIRD QMF STAGE VARIABLES *
* *
* DELAY VARIABLES OF LOW4 (0-500 Hz) *
  IXT2 DATA 0 AFTER FILTERING INPUT STORED IN IXT2, SO
  IXT3 DATA 0 IXT1 NOT NEEDED
  IXT4 DATA 0
  IXT5 DATA 0
  IXT6 DATA 0
 *
* DELAY VARIABLES OF HIGH4 (500-1000 Hz) *
  IXT8 DATA 0 AFTER FILTERING INPUT STORED IN IXT8, SO
  IXT9 DATA 0 IXT7 NOT NEEDED
  IXT10 DATA 0
  IXT11 DATA 0
  IXT12 DATA 0
 *
* DELAY VARIABLES OF LOW5 (1000-2000 Hz) *
  IXT14 DATA 0 AFTER FILTERING INPUT STORED IN IXT14, SO
  IXT15 DATA 0 IXT13 NOT NEEDED
  IXT16 DATA 0
  IXT17 DATA 0
  IXT18 DATA 0
 *
* DELAY VARIABLES OF HIGH5 (1500-2000 Hz) *
  IXT20 DATA 0 AFTER FILTERING INPUT STORED IN IXT20, SO
  IXT21 DATA 0 IXT19 NOT NEEDED
  IXT22 DATA 0
  IXT23 DATA 0
  IXT24 DATA 0
 *
* DELAY VARIABLES OF LOW7 (2000-2500 Hz) *
  IXT26 DATA 0 AFTER FILTERING INPUT STORED IN IXT26, SO
  IXT27 DATA 0 IXT25 NOT NEEDED
  IXT28 DATA 0
  IXT29 DATA 0
  IXT30 DATA 0
 *
* DELAY VARIABLES OF HIGH7 (2500-3000 Hz) *
  IXT32 DATA 0 AFTER FILTERING INPUT STORED IN IXT32, SO
  IXT33 DATA 0 IXT31 NOT NEEDED
  IXT34 DATA 0
  IXT35 DATA 0
  IXT36 DATA 0
 *
FIRST QMF STAGE COEFFICIENTS  FORMAT=Q16

CCF0  DATA  45
CCF1  DATA  -92
CCF2  DATA  -63
CCF3  DATA  277
CCF4  DATA  93
CCF5  DATA  -620
CCF6  DATA  -9
CCF7  DATA  1178
CCF8  DATA  -274
CCF9  DATA  -2047
CCF10 DATA  955
CCF11 DATA  3470
CCF12 DATA  -2579
CCF13 DATA  -6541
CCF14 DATA  8425
CCF15 DATA  30566

SECOND QMF STAGE COEFFICIENTS  FORMAT=Q16

CCS0  DATA  69
CCS1  DATA  -331
CCS2  DATA  -170
CCS3  DATA  1812
CCS4  DATA  -633
CCS5  DATA  -5924
CCS6  DATA  6409
CCS7  DATA  31525

THIRD QMF STAGE COEFFICIENTS  FORMAT=Q16

CCT0  DATA  -250
CCT1  DATA  1236
CCT2  DATA  -178
CCT3  DATA  -5551
CCT4  DATA  5798
CCT5  DATA  31745

(1-GAMMA)\*DC FOR 6 SUBBANDS  FORMAT=Q22

D0D1 DATA  -10486  SUBBAND 1
DUM1 DATA  0
D0D4 DATA  -13642  SUBBAND 4
D0D5 DATA  -23112  SUBBAND 5
D0D2 DATA  -10486  SUBBAND 2
DUM2 DATA  0
D0D3 DATA  -13642  SUBBAND 3
D0D6 DATA  -23112  SUBBAND 6
* D(N) FOR 6 SUBBANDS
  * DN1 DATA -13563 SUBBAND 1
  * DUM3 DATA 0
  * DN4 DATA -14794 SUBBAND 4
  * DN5 DATA -18495 SUBBAND 5
  * DN2 DATA -13563 SUBBAND 2
  * DUM4 DATA 0
  * DN3 DATA -14794 SUBBAND 3
  * DN6 DATA -18495 SUBBAND 6

* BIT ALLOCATION FOR VOICED STRATEGY (0)
  * BAV011 DATA 4 SUBBAND 1
  * DUM5 DATA 0
  * BAV014 DATA 2 SUBBAND 4
  * BAV015 DATA 2 SUBBAND 5
  * BAV012 DATA 4 SUBBAND 2
  * DUM6 DATA 0
  * BAV013 DATA 3 SUBBAND 3
  * BAV016 DATA 0 SUBBAND 6

* BIT ALLOCATION FOR INTERMEDIATE STRATEGY (1)
  * BAINT1 DATA 4 SUBBAND 1
  * DUM7 DATA 0
  * BAINT4 DATA 2 SUBBAND 4
  * BAINT5 DATA 2 SUBBAND 5
  * BAINT2 DATA 3 SUBBAND 2
  * DUM8 DATA 0
  * BAINT3 DATA 2 SUBBAND 3
  * BAINT6 DATA 2 SUBBAND 6

* BIT ALLOCATION FOR UNVOICED STRATEGY (2)
  * BAUNV1 DATA 2 SUBBAND 1
  * DUM9 DATA 0
  * BAUNV4 DATA 3 SUBBAND 4
  * BAUNV5 DATA 2 SUBBAND 5
  * BAUNV2 DATA 3 SUBBAND 2
  * DUM10 DATA 0
  * BAUNV3 DATA 3 SUBBAND 3
  * BAUNV6 DATA 2 SUBBAND 6

* STORAGE OF CODED SUBBAND SAMPLES
  * COD1 DATA 0 SUBBAND 1
  * DUM11 DATA 0
  * COD4 DATA 0 SUBBAND 4
  * COD5 DATA 0 SUBBAND 5
  * COD2 DATA 0 SUBBAND 2
  * DUM12 DATA 0
  * COD3 DATA 0 SUBBAND 3
  * COD6 DATA 0 SUBBAND 6

* SATURATION LEVELS FOR QUANTIZER
  * SAT1 DATA 1 2 BITS
  * SAT2 DATA 3 3 BITS
  * SAT3 DATA 7 4 BITS
* QUANTIZER CONSTANTS

\[ \text{STEP DATA 19637} \quad \frac{1}{\text{DISTANCE BETWEEN INPUTS OF EXPONENT TABLE}} (Q7) \]

\[ \text{GAMMA DATA 32440} \quad \text{GAMMA=0.99} \quad (Q15) \]

* LOGTABLE 2 BIT CASE

\[ \text{LOG20 DATA 18268} \]
\[ \text{LOG21 DATA -4626} \]
\[ \text{LOG22 DATA -4626} \]
\[ \text{LOG23 DATA 18268} \]
\[ \text{DUM13 DATA 0} \]
\[ \text{DUM14 DATA 0} \]
\[ \text{DUM15 DATA 0} \]
\[ \text{DUM16 DATA 0} \]
\[ \text{DUM17 DATA 0} \]
\[ \text{DUM18 DATA 0} \]
\[ \text{DUM19 DATA 0} \]
\[ \text{DUM20 DATA 0} \]
\[ \text{DUM21 DATA 0} \]
\[ \text{DUM22 DATA 0} \]

* LOGTABLE 3 BIT CASE

\[ \text{LOG30 DATA 11540} \]
\[ \text{LOG31 DATA 0} \]
\[ \text{LOG32 DATA 0} \]
\[ \text{LOG33 DATA -4626} \]
\[ \text{LOG34 DATA -4626} \]
\[ \text{LOG35 DATA 0} \]
\[ \text{LOG36 DATA 0} \]
\[ \text{LOG37 DATA 11540} \]
\[ \text{DUM23 DATA 0} \]
\[ \text{DUM24 DATA 0} \]
\[ \text{DUM25 DATA 0} \]
\[ \text{DUM26 DATA 0} \]

* LOGTABLE 4 BIT CASE

\[ \text{LOG40 DATA 24918} \]
\[ \text{LOG41 DATA 19728} \]
\[ \text{LOG42 DATA 13377} \]
\[ \text{LOG43 DATA 5189} \]
\[ \text{LOG44 DATA -2999} \]
\[ \text{LOG45 DATA -2999} \]
\[ \text{LOG46 DATA -2999} \]
\[ \text{LOG47 DATA -2999} \]
\[ \text{LOG48 DATA -2999} \]
\[ \text{LOG49 DATA -2999} \]
\[ \text{LOG50 DATA -2999} \]
\[ \text{LOG51 DATA -2999} \]
\[ \text{LOG52 DATA 5189} \]
\[ \text{LOG53 DATA 13377} \]
\[ \text{LOG54 DATA 19728} \]
\[ \text{LOG55 DATA 24918} \]

*
FORMT=Q14

SUBBAND 1
IX10 DATA -13563
XX1127 DATA 0      SUBBAND 1
DUM27 DATA 0
DUM28 DATA 0
XX140 DATA -14796  SUBBAND 4
XX14127 DATA -1233  SUBBAND 4
XX150 DATA -18495   SUBBAND 5
XX15127 DATA -4932  SUBBAND 5
XX120 DATA -13563   SUBBAND 2
XX12127 DATA 0      SUBBAND 2
DUM29 DATA 0
DUM30 DATA 0
XX130 DATA -14796  SUBBAND 3
XX13127 DATA -1233  SUBBAND 3
XX160 DATA -18495   SUBBAND 6
XX16127 DATA -4932  SUBBAND 6

EXPONENT TABLE OF CODER
FORMATS ARE:
  SUBBAND 1  Q4
  SUBBAND 2  Q4
  SUBBAND 3  Q3
  SUBBAND 4  Q3
  SUBBAND 5  Q0
  SUBBAND 6  Q0

CEXPO DATA 32767
  DATA 30859
  DATA 29060
  DATA 27367
  DATA 25772
  DATA 24271
  DATA 22856
  DATA 21525
  DATA 20270
  DATA 19089
  DATA 17977
  DATA 16929
  DATA 15943
  DATA 15014
  DATA 14139
  DATA 13315
  DATA 12539
  DATA 11809
  DATA 11121
  DATA 10473
  DATA 9862
  DATA 9288
  DATA 8746
  DATA 8237
  DATA 7757
  DATA 7305
  DATA 6879
  DATA 6478
  DATA 6101
  DATA 5745
  DATA 5411
  DATA 5095

LOWEST AND HIGHEST POSSIBLE INPUTS OF EXPONENT TABLE

DATA -13563  SUBBAND 1
DATA 0        SUBBAND 1
DATA 0
DATA 0        SUBBAND 4
DATA 0
DATA 0        SUBBAND 3
DATA 0
DATA 0        SUBBAND 3
DATA 0
DATA -14796  SUBBAND 4
DATA -1233   SUBBAND 4
DATA -18495  SUBBAND 5
DATA -4932   SUBBAND 5
DATA -13563  SUBBAND 2
DATA 0       SUBBAND 2
DATA 0
DATA 0
DATA -14796  SUBBAND 3
DATA -1233   SUBBAND 3
DATA -18495  SUBBAND 6
DATA -4932   SUBBAND 6
DATA 4798
DATA 4519
DATA 4256
DATA 4008
DATA 3774
DATA 3554
DATA 3347
DATA 3152
DATA 2968
DATA 2795
DATA 2632
DATA 2479
DATA 2335
DATA 2199
DATA 2070
DATA 1950
DATA 1836
DATA 1729
DATA 1628
DATA 1534
DATA 1444
DATA 1360
DATA 1281
DATA 1206
DATA 1136
DATA 1070
DATA 1007
DATA 949
DATA 893
DATA 841
DATA 792
DATA 746
DATA 703
DATA 662
DATA 623
DATA 587
DATA 553
DATA 520
DATA 490
DATA 462
DATA 435
DATA 409
DATA 385
DATA 363
DATA 342
DATA 322
DATA 303
DATA 286
DATA 269
DATA 253
DATA 238
DATA 225
DATA 211
DATA 199
DATA 188
DATA 177
DATA 166
DATA 157
DATA 148
DATA 139
DATA 131
DATA 123
DATA 116
DATA 109
DATA 103
DATA 97
DATA 91
DATA 86
DATA 81
DATA 76
DATA 72
DATA 68
DATA 64
DATA 60
DATA 56
DATA 53
DATA 50
DATA 47
DATA 44
DATA 42
DATA 39
DATA 37
DATA 35
DATA 33
DATA 31
DATA 29
DATA 27
DATA 26
DATA 24
DATA 23
DATA 22
DATA 20
DATA 19
DATA 18
DATA 17
DATA 16

*   ...
* MAIN ROUTINE
*   ...
*   ...
* INITIALIZATIONS
*   ...
* START DINT
*   ...
* INITIALIZE STATUS BITS
LARP 0
LDPK 0
SOVM
* INITIALIZE GENERAL VARIABLES OF DATA MEMORY PAGE 1
  LDPK 1
  LACK CCLOCK
  TBLR CLOCK       CLOCK RATE (16 kHz)
  LACK MMODE
  TBLR MODE       MODE FOR ANALOG INTERFACE BOARD
  LDPK 0

* INITIALIZE GENERAL VARIABLES OF DATA MEMORY PAGE 0
  LACK MMSTAT
  TBLR MSTAT     MASK FOR STATE MOD 16
  LACK 5         NO. OF WAIT CYCLES BEFORE MULTIPLEXER START
  SAcl SWITCH    COUNTER FOR MULTIPLEXER ON or OFF, NOW OFF
  ZAC
  SAcl FRAME     STATUS FOR CODER
  LACK 1
  SAcl ONE       '1' FOR LOGICAL MANIPULATIONS
  SAcl STRAT    BY DEFAULT LOAD INTERMEDIATE STRATEGY
  SAcl SSTRAT   ALSO INTERMEDIATE STRATEGY FOR BUFFER
  ZAC
  SAcl STATE     INIT TREE POINTER

* INITIALIZE PCMAIN VARIABLES
  LACK SSFEP
  TBLR STEP     1/(DISTANCE BETWEEN INPUTS OF EXPONENT TABLE)
  LACK GGAMMA
  TBLR GAMMA    GAMMA VALUE OF FORMULA (.99)

* INITIALIZE MULTIPLEXER VARIABLES
  ZAC
  SAcl BAND     STATUS FOR MULTIPLEXER
  LACK >FF
  SAcl DEMO     NEW ACCESS MARK

* LOAD QMF COEFFICIENTS INTO DATA RAM
  LARK 1,CTS
  LARK 0.29
  LACK CCTS
  LOAD LARP 1
  TBLR +,-0
  SUB ONE
  BANZ LOAD

* INITIALIZE ANALOG INTERFACE BOARD (AIB)
  LDPK 1
  OUT MODE,0      PROGRAM AIB
  OUT CLOCK,1     SET CLOCK RATE OF AIB
  LDPK 0
  OUT RES0,2      CLEAR ADIS (PROTECTION AGAINST RESET INTERRUPTS)
  OUT RES0,3      CLEAR AD2S (PROTECTION AGAINST RESET INTERRUPTS)

* INITIALIZATION AND VARIABLE LOADING DONE, NOW BEGIN EINT
DISCARD FIRST TWO INTERRUPTS TO GET A DEFINED STARTING POINT

ACR KQ  ZALS STATE
BZ ACKNO
LACK 1
XOR STATE
BZ ACKNO
ZAC
SACL STATE INIT TREE POINTER

AORTA

ODD CYCLE OF 16 kHz CLOCK MUST JUST HAVE PASSED
WAITEV LAC STATE
AND ONE
BNZ WAITEV
EVEN CYCLE OF 16 kHz CLOCK HAS PASSED (BIT 0 OF STATE=0)
WAIT LAC STATE
AND ONE
BZ WAIT
ODD CYCLE OF 16 kHz CLOCK HAS JUST PASSED (BIT 0 OF STATE=1)

INPUT SAMPLE RATE REDUCED FROM 16 kHz (2*8 kHz) TO 8 kHz BY OMITTING
EVERY EVEN INPUT SAMPLE

DURING THE PROCESSING OF A SUBBAND SAMPLE STATE CAN CHANGE AS A
RESULT OF AN INTERRUPT, SO DEFINE PATH=STATE/2
LAC STATE,15
SACH PATH

ACTUAL SUBBAND CODING
CALL QMF
CALL PMAGB
B WAITEV

INTERRUPT SERVICE ROUTINE

INTRO DINT

SAVE STATUS OF TMS32010
SST STATU
MPY ONE T REGISTER TO P REGISTER
LDPK 1
SAR 0,AR00
SAR 1,AR01
SACH ACK
SACL ACL
PAC
SACL TRES
LDPK 0
STATE UPDATE
LAC STATE
ADD ONE
AND MSTAT STATE MOD 16
SACL STATE

SAMPLE INPUT (16 kHz = 2*8 kHz)
IN IMPF,2 ONE OF TWO SAMPLES IS USED LATER

CHECK IF MULTIPLEXER IS ON (SWITCH=0); IF NOT DECREMENT SWITCH
AND RETURN TO PLACE OF INTERRUPT
ZALS SWITCH
B2 MIXER
SUB ONE
SACL SWITCH
B RSSTAT

RUN THROUGH MULTIPLEXER
MIXER CALL MUX

RESTORE STATUS OF TMS32010
RSSTAT LD PK 1
LAR 0,AR00
LAR 1,AR01
ZALX ACH
ADDS ACL
LT TREG
LST STATU

RETURN TO PLACE OF INTERRUPT AND CONTINUE
EINT
RET

CODER PROGRAM DONE

======================================
APPENDIX C

Subroutine QMF
QMF FILTERING

THIS SUBROUTINE INCORPORATES:
- QMF FILTER TREE
- ENERGY MEASUREMENT
- VOICING STRATEGY DECISION
- TAKE OVER OF NEW VOICING STRATEGY
FOR THE PCM CODING

INPUT: 8 kHz SAMPLES (1 STREAM)
OUTPUT: FAW + 1 kHz SAMPLES (8 STREAMS)

SUBBAND PROCESSING: 1, 8, 4, 5, 2, 7, 3, 6

BIT ALLOCATION:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICED</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UNVOICED</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FAW's:
- VOICED >76
- INTERMEDIATE >96
- UNVOICED >CC

AUTHOR: ROLAND KLEUTERS
DATE: 29-5-1987

MACRO'S

MACRO TO REALIZE FILTERING IN LOW BRANCH OF FIRST QMF STAGE

```
MACRO I
ZAC
LT XF16
MPY CF0
LTD XF15
MPY CF2
LTD XF14
MPY CF4
LTD XF13
MPY CF6
LTD XF12
MPY CF8
LTD XF11
MPY CF10
LTD XF10
MPY CF12
LTD XF9
MPY CF14
LTD XF8
MPY CF15
LTD XF7
MPY CF13
LTD XF6
MPY CF11
LTD XF5
MPY CF9
```
MACRO TO REALIZE FILTERING IN HIGH BRANCH OF FIRST QMF STAGE

FLTFHI $MACRO X

ZAC
LT XF32
MPI CF1
LTD XF31
MPI CF3
LTD XF30
MPI CF5
LTD XF29
MPI CF7
LTD XF28
MPI CF9
LTD XF27
MPI CF11
LTD XF26
MPI CF13
LTD XF25
MPI CF15
LTD XF24
MPI CF14
LTD XF23
MPI CF12
LTD XF22
MPI CF10
LTD XF21
MPI CF8
LTD XF20
MPI CF6
LTD XF19
MPI CF4
LTD XF18
MPI CF2
LTD IMPF
MPI CF0
APAC
SACH :.5,1
LAC IMPF
SACL XF18
$END

8 kHz INPUT TO FIRST QMF STAGE

8 kHz INPUT TO FIRST QMF STAGE
MACRO TO REALIZE FILTERING IN LOW BRANCH OF SECOND QMF STAGE
FLTSLO $MACRO X
ZAC
LT IS8
MPY CS0
LTD IS7
MPY CS2
LTD IS6
MPY CS4
LTD IS5
MPY CS6
LTD IS4
MPY CS7
LTD IS3
MPY CS5
LTD IS2
MPY CS3
LTD IMPS 4 kHz INPUT TO SECOND QMF STAGE
MPY CS1
APAC
SACH :.S.,I
$END

MACRO TO REALIZE FILTERING IN HIGH BRANCH OF SECOND QMF STAGE
FLTSHI $MACRO X
ZAC
LT IS8
MPY CS1
LTD IS7
MPY CS3
LTD IS6
MPY CS5
LTD IS5
MPY CS7
LTD IS4
MPY CS6
LTD IS3
MPY CS4
LTD IS2
MPY CS2
LTD IMPS 4 kHz INPUT TO SECOND QMF STAGE
MPY CS0
APAC
SACH :I.IS.,I
$END

MACRO TO REALIZE FILTERING IN LOW BRANCH OF THIRD QMF STAGE
FLTLO $MACRO X
ZAC
LT IT6
MPY CT0
LTD IT5
MPY CT2
LTD IT4
MPY CT4
LTD IT3
MPY CT5
LTD IT2
MPY CT3
LTD INFT  2 kH: INPUT TO THIRD QMF STAGE
MPY CT1
APAC
SACH :X.5:1
$END

* # MACRO TO REALIZE FILTERING IN HIGH BRANCH OF THIRD QMF STAGE
FLTTHI $MACRO I
  ZAC
  LT XT6
  MPY CT1
  LTD XT5
  MPY CT3
  LTD XT4
  MPY CT5
  LTD XT3
  MPY CT4
  LTD XT2
  MPY CT2
  LTD INFT  2 kH: INPUT TO THIRD QMF STAGE
  MPY CTO
  APAC
  SACH :X.5:1
$END

# # MACRO TO LOAD DELAY VARIABLES OF SECOND QMF STAGE
LOADDS $MACRO I
  LACK :X.5:
  LARK 0,1S2
  TBLR ++
  ADD ONE
  TBLR ++
  ADD ONE
  TBLR ++
  ADD ONE
  TBLR ++
  ADD ONE
  TBLR ++
  ADD ONE
  TBLR ++
  ADD ONE
  TBLR +
$END

# # MACRO TO SAVE DELAY VARIABLES OF SECOND QMF STAGE
SAVEDS $MACRO I
  LACK :X.5:
  LARK 0,1S2
  TBLW ++
  ADD ONE
  TBLW ++
  ADD ONE
  TBLW ++
  ADD ONE
  TBLW ++
  ADD ONE
  TBLW ++
ADD ONE
TBLW ++
ADD ONE
TBLW *
$END

MACRO TO LOAD DELAY VARIABLES OF THIRD QMF STAGE
LOADDT $MACRO I
LACK:i.s;
LARK 0,IT2
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR *
$END

MACRO TO SAVE DELAY VARIABLES OF THIRD QMF STAGE
SAVEDT $MACRO I
LACK:i.s;
LARK 0,IT2
TBLW ++
ADD ONE
TBLW ++
ADD ONE
TBLW ++
ADD ONE
TBLW ++
ADD ONE
TBLW *
$END

MACRO DEFINITION DONE; MAIN PART OF SUBROUTINE BEGINS

QMF NOP ENTRY POINT

SELECT LOW1 OR HIGH1 FOR FIRST QMF STAGE
LAC ONE BIT 0=0 or 1
AND PATH
BZ LOW1
B HIGH1

FIRST QMF STAGE:

LOW1 0kHz-2kHz
HIGH1 2kHz-4kHz

LOW1 DM0V A1 DOUBLE BUFFER
OUTPUT CALCULATION
LAC C1,14
ADD B1,14
SACH INPS 4 kHz INPUT TO SECOND QMF STAGE

FILTER 0-2 kHz
FLTFLO A1

ENERGY SUMMATION, SUM X(n) SQUARED
LT INPS
MPY INPS
PAC ACCU CONTAINS INPS*INPS IN Q28 FORMAT
ADD ENLOW,14 ACCU CONTAINS ENLOW IN Q28 FORMAT
SACL RES0 MULTIPLY ACCU WITH 4
SACH RES1
ADD RES0
ADDH RES1
ADD RES0
ADDH RES1
ADD RES0
ADDH RES1 ACCU CONTAINS ENLOW IN Q30 FORMAT
SACH ENLOW ENLOW IN Q14 FORMAT

SELECT LOW2 OR HIGH2 FOR SECOND QMF STAGE
LAC ONE,1
AND PATH
BZ LOW2
B HIGH2

HIGH1 NOP NO DOUBLE BUFFER

OUTPUT CALCULATION
ZALH C1
SUBH B1
SACH INPS 4 kHz INPUT TO SECOND QMF STAGE

FILTER 2-4 kHz
FLTFHI C1

ENERGY SUMMATION, SUM X(n) SQUARED
LT INPS
MPY INPS
PAC ACCU CONTAINS INPS*INPS IN Q32 FORMAT
ADDH ENHIGH ACCU CONTAINS ENHIGH IN Q32 FORMAT
SACH ENHIGH ENHIGH IN Q16 FORMAT

SELECT LOW3 OR HIGH3 FOR SECOND QMF STAGE
LAC ONE,1
AND PATH
BZ LOW3
B HIGH3
SECOND QMF STAGE:

- **LOW2**
  - 0kHz-1kHz
  - OUTPUT CALCULATION
    - LAC C2.15
    - ADD B2.15
    - SACH INPT 2 kHz INPUT TO THIRD QMF STAGE
  - LOAD DELAY VARIABLES OF LOW2
    - LOADD US2
  - FILTER 0-1 kHz
    - FLTSLO A2
  - SAVE DELAY VARIABLES OF LOW2
    - SAVEDS US2
  - SELECT LOW4 OR HIGH4 FOR THIRD QMF STAGE
    - LAC ONE,2
    - AND PATH
    - BZ LOW4
    - B HIGH4

- **HIGH2**
  - NO DOUBLE BUFFER
  - OUTPUT CALCULATION
    - LAC C2.15
    - SUB B2.15
    - SACH INPT 2 kHz INPUT TO THIRD QMF STAGE
  - LOAD DELAY VARIABLES OF HIGH2
    - LOADDUS US10
  - FILTER 1-2 kHz
    - FLTSHI C2
  - SAVE DELAY VARIABLES OF HIGH2
    - SAVEDUS US10
  - SELECT LOWS OR HIGHS FOR THIRD QMF STAGE
    - LAC ONE,2
    - AND PATH
    - BZ LOWS
    - B HIGHS

- **LOW3**
  - NO DOUBLE BUFFER
  - OUTPUT CALCULATION
    - ZALH C3
    - ADDH B3
    - SACH INPT 2 kHz INPUT TO THIRD QMF STAGE
LOAD DELAY VARIABLES OF LOW3
LOADDS IXIS18

FILTER 3-4 kHz
FLTSLD A3

SAVE DELAY VARIABLES OF LOW3
SAVEDS IXIS18

THE 3-4 kHz SUBBAND IS NOT CODED, SO FURTHER SPLITTING IS NOT NECESSARY

VOICING DECISION?
FRAME HAS TO BE 3 AND PATH HAS TO BE 5; INTERRUPT 75
LACK 3
XOR FRAME
BWZ NOVODE FRAME=3?
LACK 5
XOR PATH FRAME=5?
BZ VOIDEC

NOVODE RET NOT INTERRUPT 75; NO VOICING DECISION; QMF DONE

HIGH3 NOP NO DOUBLE BUFFER

OUTPUT CALCULATION
2ALH C3
SUBH B3
SACH INPT 2 kHz INPUT TO THIRD QMF STAGE

LOAD DELAY VARIABLES OF HIGH3
LOADDS IXIS26

FILTER 2-3 kHz
FLTSHI C3

SAVE DELAY VARIABLES OF HIGH3
SAVEDS IXIS26

SELECT LOW7 OR HIGH7 FOR THIRD QMF STAGE
LAC ONE,2
AND PATH
BZ LOW7
B HIGH7

--------------------------------------------------------------------------------------------------
THIRD QMF STAGE:

LOW4 0-500 Hz
HIGH4 500-1000 Hz
LOW5 1500-2000 Hz
HIGH5 1000-1500 Hz
LOW7 2000-2500 Hz
HIGH7 2500-3000 Hz

MAKE A VOICING DECISION
TAKE OVER NEW VOICING STRATEGY

--------------------------------------------------------------------------------------------------

LOAD4 DMOV A4 DOUBLE BUFFER
OUTPUT CALCULATION
LAC C4,15
ADD B4,15
SACH INPCM 1 kHz INPUT TO PCM CODER

LOAD DELAY VARIABLES OF LOW4
LOADDT XI2

FILTER 0-500 Hz
FLTTL0 A4

SAVE DELAY VARIABLES OF LOW4
SAVEDT XI2

QMF DONE;
BEFORE EXIT, UPDATE FRAME AND TAKE OVER NEW STRATEGY IF END OF FRAME
LAR 0,FRAME
BANN SFrame
LAC 7 END OF FRAME; PREPARE NEXT FRAME
SAI FRAME
DMOV STRAT TAKE OVER NEW VOICING STRATEGY
RET
SFramE SAR 0,FRAME NOT END OF FRAME; PREPARE NEXT DATA BLOCK
RET

HIGH4 NOP NO DOUBLE BUFFER

OUTPUT CALCULATION
LAC C4,15
SUB B4,15
SACH INPCM 1 kHz INPUT TO PCM CODER

LOAD DELAY VARIABLES OF HIGH4
LOADDT XI8

FILTER 500-1000 Hz
FLTTTHI C4

SAVE DELAY VARIABLES OF HIGH4
SAVEDT XI8

QMF DONE
RET

LOW5 DMOV A5 DOUBLE BUFFER

OUTPUT CALCULATION
ZALH C5
ADDAH B5
SACH INPCM 1 kHz INPUT TO PCM CODER

LOAD DELAY VARIABLES OF LOW5
LOADDT XI14

FILTER 1500-2000 Hz
FLTTL0 A5
SAVE DELAY VARIABLES OF LOW5
SAVEDT XXT14

QMF DONE
RET

HIGH5 NOP NO DOUBLE BUFFER

OUTPUT CALCULATION
ZALH C5
SUBH B5
SACH INPCM 1 kHz INPUT TO PCM CODER

LOAD DELAY VARIABLES OF HIGH5
LOADDT XXT20

FILTER 1000-1500 Hz
FLTTHI C5

SAVE DELAY VARIABLES OF HIGH5
SAVEDT XXT20

QMF DONE
RET

LOW7 DMOV A7 DOUBLE BUFFER

OUTPUT CALCULATION
ZALH C7
ADDH B7
SACH INPCM 1 kHz INPUT TO PCM CODER

LOAD DELAY VARIABLES OF LOW7
LOADDT XXT26

FILTER 2000-2500 Hz
FLTTL0 A7

SAVE DELAY VARIABLES OF LOW7
SAVEDT XXT26

QMF DONE
RET

HIGH7 NOP NO DOUBLE BUFFER

OUTPUT CALCULATION
ZALH C7
SUBH B7
SACH INPCM 1 kHz INPUT TO PCM CODER

LOAD DELAY VARIABLES OF HIGH7
LOADDT XXT32

FILTER 2500-3000 Hz
FLTTHI C7
SAVE DELAY VARIABLES OF HIGH?
SAVEDT XIT32

QMF DONE
RET

MAKE A VOICING DECISION
VOIDEC NOP

IF 20ENHIGH-ENLOW LESS THAN OR EQUAL TO ZERO THEN VOICED
  LT ENHIGH
  MPYK +20
  PAC
  SUB ENLOW,2
  BLEZ SETVOI

IF 3ENHIGH-2ENLOW GREATER THAN OR EQUAL TO ZERO THEN UNVOICED
  LT ENHIGH
  MPYK +3
  PAC
  SUB ENLOW,2
  SUB ENLOW,2
  BGEZ SETUN

IF 10ENHIGH-ENLOW LESS THAN OR EQUAL TO ZERO AND PREVIOUS VOICING
  WAS VOICED THEN VOICED
  LAC STRAT
  BNZ PREUNV
  LT ENHIGH
  MPYK +10
  PAC
  SUB ENLOW,2
  BLEZ SETVOI

IF 3ENHIGH-ENLOW GREATER THAN OR EQUAL TO ZERO AND PREVIOUS VOICING
  WAS UNVOICED THEN UNVOICED
  PREUNV LACK 2
  IOR STRAT
  BNZ SETINT
  LT ENHIGH
  MPYK 3
  PAC
  SUB ENLOW,2
  BGEZ SETUN

ELSE NEW STRATEGY IS INTERMEDIATE STRATEGY

SET INTERMEDIATE STRATEGY AND CLEAR VOICING DECISION BUFFERS
SETINT LACK 1
SACL STRAT
ZAC
SACL ENLOW
SACL ENHIGH
RET
  QMF DONE
SET VOICED STRATEGY AND CLEAR VOICING DECISION BUFFERS

SET VOI ZAC
SACL STRAT
SACL EMLOW
SACL ENHIGH
RET QMF DONE

SET UNVOICED STRATEGY AND CLEAR VOICING DECISION BUFFERS

SET UN ZAC
SACL STRAT
ZAC
SACL EMLOW
SACL ENHIGH
RET QMF DONE

QMF FILTERING DONE

#=========================================================================================

APPENDIX D

Subroutine PCMAQ8 (coder)
PCMAQB Coding

This subroutine incorporates:
- Backward stepsize adaptation of the quantizer
- Quantization of a sample of the subband being processed
- 2, 3 or 4 bit PCM coding of that quantized sample

Input: FAW + 1 kHz samples (8 streams)
Output: FAW + 1 kHz samples (8 streams)

Bit allocation according to SSRT:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voiced</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Intermediate</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Unvoiced</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Author: Roland Kleuters
Date: 29-5-1987
DETERMINE 1/STEP SIZE i.e. 1/DELTA(n) BY BACKWARD STEPSIZE ADAPTATION

DETERMINE d(n)=LOG(DELTA(n))

READ IN THE PREVIOUS CODED RESULT I(n-1) FROM TABLE
READIN LACK COD1
ADD PATH
TBLR CALC1 CALCI CONTAINS THE PREVIOUS CODED RESULT I(n-1)

DETERMINE m(I(n-1))=LOG(M(I(n-1))) FROM TABLES
LAC NBIT
SUB ONE,1
SA CL CALC2
LACK LOG2
ADD CALC2,4
ADD CALC1
TBLR RESI RESI CONTAINS m(I(n-1))=LOG(M(I(n-1)))

DETERMINE (1-GAMMA)*DC FROM TABLE
LACK OBD1
ADD PATH
TBLR CALC1 CALCI CONTAINS (1-GAMMA)*DC

DETERMINE GAMMA*d(n-1) FROM TABLE
LAC DNI
ADD PATH
TBLR RESO RESO TO NAINS d(n-1)
LT GAMMA
MPY RESO
PAC
SACH CALC2,1 CALC2 CONTAINS GAMMA*d(n-1)

DETERMINE GAMMA*d(n-1)+m(I(n-1))+1-GAMMA)*DC
ZALH CALC2
ADD RESI,12
ADD CALCI,8
SACH CALC1 CALCI CONTAINS GAMMA*d(n-1)+m(I(n-1))+
(1-GAMMA)*DC, i.e. d(n)

CONTROL THE LIMITATION OF d(n) TO ITS RANGE AND SAVE d(n) IN TABLE
LAC CALCI
SUB XI27
BE Z SATDNH d(n) GREATER THAN MAXIMUM POSSIBLE VALUE OF d(n)?
LAC IO
SUB CAL CI
BE Z SATDL d(n) LESS THAN MINIMUM POSSIBLE VALUE OF d(n)?
B SAVEDN NO SATURATION NEEDED
SATDNH LAC XI27 SATURATE d(n) TO MAXIMUM POSSIBLE VALUE
SA CL CALC1
B SAVEDN
SATDL LAC IO SATURATE d(n) TO MINIMUM POSSIBLE VALUE
SA CL CALC1
SAVED
LACK DNI
ADD PATH
TBLW CALCI

SAVED
LACK DNI
ADD PATH
TBLW CALCI

DETERMINE 1/DELTAn=EXP(-d(n))

ROUND d(n) TO ONE OF THE ENTRY POINTS OF THE EXPONENT TABLE
ZALH CALCI
SUB X0
SACH CALCI
CALCI CONTAINS d(n)-X0
LT CALCI
MPY STEP
PAC
SACH CALCI
CALCI CONTAINS (d(n)-X0)\*STEP
LAC CALCI,11
SACH CALCI
CALCI CONTAINS INTEGER[(d(n)-X0)\*STEP]=ENTRY POINT

DETERMINE THE OUTPUT OF EXPONENT TABLE
LACK CEXP0
ADD CALCI
TBLR CALCI
CALCI CONTAINS OUTPUT OF EXPONENT TABLE=1/DELTAn

QUANTIZE THE PCMABg CODER INPUT AND CODE THE RESULT (DETERMINE I(n))

DETERMINE INPUT/DELTAn AND ROUND TO THE GREATEST INTEGER BELOW
LT CALCI
MPY INPCM
PAC
SACH CALCI
CALCI CONTAINS INPUT/DELTAn (QB)
LAC CALCI,14
SACH CALCI
CALCI CONTAINS INTEGER(INPUT/DELTAn) IN TWO'S COMPLEMENT CODE, i.e. THE CODED RESULT I(n)

CONTROL THE LIMITATION OF I(n) TO THE RANGE ACCORDING TO THE NUMBER OF CODE BITS AND SAVE I(n) IN TABLE OF CODED SUBBAND SAMPLES
LACK SATI
ADD MBIT
SUB ONE,1
TBLR RESO
RESO CONTAINS THE MAXIMUM POSSIBLE VALUE OF I(n)
LAC CALCI
SUB RESO
BGZ SATINH
I(n) GREATER THAN MAXIMUM POSSIBLE VALUE OF I(n)?
ZAC
SUB RESO
SUB ONE
SACL RESI
RESI CONTAINS THE MINIMUM POSSIBLE VALUE OF I(n)
SUB CALCI
BGZ SATINL
I(n) LESS THAN MINIMUM POSSIBLE VALUE OF I(n)?
LACK CODI
SAVE I(n) WITHOUT SATURATION
ADD PATH
TBLW CALCI
RET
SATINH LACK COD1 SATURATE I(n) TO MAXIMUM POSSIBLE VALUE AND SAVE
ADD PATH
TBLW RESO
RET
SATINL LACK CODI SATURATE I(n) TO MINIMUM POSSIBLE VALUE AND SAVE
ADD PATH
TBLW RESI
RET

PCMA9B CODING DONE
#=================================================================================
APPENDIX E

Subroutine MULTIPLEXING
MULTIPLEXING

THIS SUBROUTINE INCORPORATES:
- MULTIPLEXING AND
- SENDING OF PCM CODED DATA

INPUT: FAW + 1 kHz SAMPLES (8 STREAMS)
OUTPUT: 16 KBPS BITSTREAM

SUBBAND PROCESSING: 1, 8, 4, 5, 2, 7, 3, 6

BIT ALLOCATION:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICED</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UNVOICED</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FAW's:
- VOICED > 76
- INTERMEDIATE > 96
- UNVOICED > 26

AUTHOR: ROLAND KLEUTERS
DATE: 29-5-1987

MACRO

MACRO TO SEND ONE BIT
SEND

MUI MOP ENTRY POINT

SELECT FAW or ONE OF THE SUBBANDS FROM WHICH DATA HAS TO BE SENT
FAM SUB
LACK 4
SUB BAND
B62 FWSB62
BZ NEXT4
ADD ONE
BZ NEXT5
B NEXT6
SUBBAND 1 (0-500 Hz)
SUBBAND 2 (500-1000 Hz)
SUBBAND 3 (1000-1500 Hz)

FWSBGZ
SUB ONE
BZ NEXT1
SUB ONE
BZ NEXT2
SUB ONE
BZ NEXT3
SUB ONE

* SEND FAW BITS

NFAM ZALS DEMO NEW ACCESS CHECK
BZ FSIG

* INITIALIZATION FOR A NEW FAW ACCESS
ZAC
SACL DEMO NO NEW ACCESS ANYMORE
LACK 7
SACL COUNT 8 FAW BITS TO SEND

* LOAD FAW INTO SENDBUFFER (=DATA)
ZALS SSRTAT
BZ SVOICE
XOR ONE
BZ SINTER
SUNVDI LACK >CC CODE FOR UNVOICED STRATEGY
SACL DATA PRESSET UNVOICED STRATEGY
B FSIG
SVOICE LACK >76 CODE FOR VOICED STRATEGY
SACL DATA PRESET VOICED STRATEGY
B FSIG
SINTER LACK >96 CODE FOR INTERMEDIATE STRATEGY
SACL DATA PRESET INTERMEDIATE STRATEGY

* SEND A FAW BIT
FSIG SEND
LAR 0,COUNT
BANZ CONT

* ALL FAW BITS SENT, PREPARE NEXT SUBBAND
LACK >FF
SACL DEMO NEW ACCESS MARK
LACK 1
SACL BAND NEXT SUBBAND TO SEND (0-500 Hz)
RET

* NOT ALL FAW BITS SENT, PREPARE NEXT BIT
CONT SAR 0,COUNT
RET

* SEND SUBBAND BITS
A BIT OF SUBBAND 1 (0-500 Hz) HAS TO BE SENT

NEIT1 ZALS DEMO NEW ACCESS CHECK
BZ LOOP1

INITIALIZATION FOR A NEW SUBBAND 1 ACCESS
ZAC
SACL DEMO NO NEW ACCESS ANYMORE
LACK CODE
TBLR DATA LOAD SUBBAND 1 DATA INTO SEND BUFFER
LACK 2
XOR SSTRAT UNVOICED?
BNZ VOINI

FURTHER INITIALIZATION FOR UNVOICED STRATEGY
LACK 1
SACL COUNT 2 BITS TO SEND
LAC DATA,6 ADJUST DATA ACCORDING TO BITSELECTOR
SACL DATA DATA=0000 0000 DDDD 0000
B LOOP1

FURTHER INITIALIZATION FOR VOICED AND INTERMEDIATE STRATEGY
VOINI LACK 3
SACL COUNT 4 BITS TO SEND
LAC DATA,4 ADJUST DATA ACCORDING TO BITSELECTOR
SACL DATA DATA=0000 0000 DDDD 0000

SEND A SUBBAND 1 BIT
LOOP1 SEND
LAR 0,COUNT
BANZ CONTI

ALL SUBBAND 1 BITS SENT, PREPARE NEXT SUBBAND
LACK 0FF
SACL DEMO NEW ACCESS MARK
LACK 4
SACL BAND NEXT SUBBAND TO SEND (1500-2000 Hz)
RET

NOT ALL SUBBAND 1 BITS SENT, PREPARE NEXT BIT
CONTI SAR 0,COUNT
RET

A BIT OF SUBBAND 2 (500-1000 Hz) HAS TO BE SENT

NEIT2 ZALS DEMO NEW ACCESS CHECK
BZ LOOP2
**INITIALIZATION FOR A NEW SUBBAND 2 ACCESS**

ZAC  
SACL DEMO NO NEW ACCESS ANYMORE  
LACK COD2:  
TBLR DATA LOAD SUBBAND 2 DATA INTO SEND BUFFER  
LAC SSTRAT VOICED?  
BZ INUN2

**FURTHER INITIALIZATION FOR VOICED STRATEGY**

LACK 3  
SACL COUNT 4 BITS TO SEND  
LAC DATA,4 ADJUST DATA ACCORDING TO BITSELECTOR  
SACL DATA DATA=0000 0000 DDDD 0000

**FURTHER INITIALIZATION FOR INTERMEDIATE AND UNVOICED STRATEGY**

INUN2 LACK 2  
SACL COUNT 3 BITS TO SEND  
LAC DATA,5 ADJUST DATA ACCORDING TO BITSELECTOR  
SACL DATA DATA=0000 0000 DDDD 0000

**SEND A SUBBAND 2 BIT**

LOOP2 SEND  
LAR 0,COUNT  
BANZ CONT2

**ALL SUBBAND 2 BITS SENT, PREPARE NEXT SUBBAND**

LACK >FF  
SACL DEMO NEW ACCESS MARK  
LACK 3  
SACL BAND NEXT SUBBAND TO SEND (1000-1500 Hz)  
RET

**NOT ALL SUBBAND 2 BITS SENT, PREPARE NEXT BIT**

CONT2 SAR 0,COUNT  
RET

**A BIT OF SUBBAND 3 (1000-1500) HAS TO BE SENT**

**INITIALIZATION FOR A NEW SUBBAND 3 ACCESS**

ZAC  
SACL DEMO NO NEW ACCESS ANYMORE  
LACK COD3  
TBLR DATA LOAD SUBBAND 3 DATA INTO SEND BUFFER  
LACK 1  
XOR SSSTRAT INTERMEDIATE?  
BZ WOUN3
FURTHER INITIALIZATION FOR INTERMEDIATE STRATEGY
LACK 1
SACL COUNT 2 BITS TO SEND
LAC Data A,6 ADJUST DATA ACCORDING TO BITSELECTOR
SACL DATA DATA=0000 0000 DDOO 0000
B LOOP3

FURTHER INITIALIZATION FOR VOICED AND UNVOICED STRATEGY
VOUN3 LACK 2
SACL COUNT 3 BITS TO SEND
LAC Data A,5 ADJUST DATA ACCORDING TO BITSELECTOR
SACL DATA DATA=0000 0000 DDOO 0000

SEND A SUBBAND 3 BIT
LOOP3 SEND
LAR 0,COUNT
BANZ CONT3

ALL SUBBAND 3 BITS SENT AND PERHAPS ALSO END OF 15 BIT DATABLOCK
CHECK FOR END OF FRAME
LACK >FF
SACL DEMO NEW ACCESS MARK
LACK 7
IOD FRAME
BZ ENDFRM

NOT END OF FRAME, PREPARE NEXT SUBBAND
LAC SSTRAT VOICED?
BZ ENDFRM
LACK 6
SACL BAND NEXT SUBBAND TO SEND (2500-3000 Hz)
RET

NOT ALL SUBBAND 3 BITS SENT, PREPARE NEXT BIT
CONT3 SAR 0,COUNT
RET

A BIT OF SUBBAND 4 (1500-2000 Hz) HAS TO BE SENT

INITIALIZATION FOR A NEW SUBBAND 4 ACCESS
ZAC
SACL DEMO NO NEW ACCESS ANYMORE
LACK COD4
TBLR DATA LOAD SUBBAND 4 DATA INTO SEND BUFFER
LACK 2
IOD SSTRAT UNVOICED?
BNZ VOIN4
* FURTHER INITIALIZATION FOR UNVOICED STRATEGY
  \[ \text{LACK 2} \]
  \[ \text{SAACL COUNT} \quad 3 \text{ BITS TO SEND} \]
  \[ \text{LAC DATA,5} \quad \text{ADJUST DATA ACCORDING TO BITSELECTOR} \]
  \[ \text{SAACL DATA} \quad \text{DATA}=0000 \quad 0000 \quad 0000 \quad 0000 \]

* FURTHER INITIALIZATION FOR VOICED AND INTERMEDIATE STRATEGY
  \[ \text{VOIN4 LACK 1} \]
  \[ \text{SAACL COUNT} \quad 2 \text{ BITS TO SEND} \]
  \[ \text{LAC DATA,6} \quad \text{ADJUST DATA ACCORDING TO BITSELECTOR} \]
  \[ \text{SAACL DATA} \quad \text{DATA}=0000 \quad 0000 \quad 0000 \quad 0000 \]

* SEND A SUBBAND 4 BIT
  \[ \text{LOOP4 SEND} \]
  \[ \text{LAR} \quad 0, \text{COUNT} \]
  \[ \text{BANZ CONT4} \]

* ALL SUBBAND 4 BITS SENT PREPARE NEXT SUBBAND
  \[ \text{LACK} \quad \text{FF} \]
  \[ \text{SAACL DEMO} \quad \text{NEW ACCESS MARK} \]
  \[ \text{LACK 5} \]
  \[ \text{SAACL BAND} \quad \text{NEXT SUBBAND TO SEND (2000-2500 Hz)} \]
  \[ \text{RET} \]

* NOT ALL SUBBAND 4 BITS SENT, PREPARE NEXT BIT
  \[ \text{CONT4 SAR} \quad 0, \text{COUNT} \]
  \[ \text{RET} \]

* A BIT OF SUBBAND 5 (2000-2500) HAS TO BE SENT

* NEXT5 ZALS DEMO \quad \text{NEW ACCESS CHECK}
  \[ \text{B2 LOOP5} \]

* INITIALIZATION FOR A NEW SUBBAND 5 ACCESS
  \[ \text{ZAC} \]
  \[ \text{SAACL DEMO} \quad \text{NO NEW ACCESS ANYMORE} \]
  \[ \text{LACK CUDS} \]
  \[ \text{TBLR DATA} \quad \text{LOAD SUBBAND 5 DATA INTO SEND BUFFER} \]
  \[ \text{LACK 1} \]
  \[ \text{SAACL COUNT} \quad 2 \text{ BITS TO SEND FOR ALL STRATEGIES} \]
  \[ \text{LAC DATA,6} \quad \text{ADJUST DATA ACCORDING TO BITSELECTOR} \]
  \[ \text{SAACL DATA} \quad \text{DATA}=0000 \quad 0000 \quad 0000 \quad 0000 \]

* SEND A SUBBAND 5 BIT
  \[ \text{LOOP5 SEND} \]
  \[ \text{LAR} \quad 0, \text{COUNT} \]
  \[ \text{BANZ CONT5} \]

* ALL SUBBAND 5 BITS SENT, PREPARE NEXT SUBBAND
  \[ \text{LACK} \quad \text{FF} \]
  \[ \text{SAACL DEMO} \quad \text{NEW ACCESS MARK} \]
  \[ \text{LACK 2} \]
  \[ \text{SAACL BAND} \quad \text{NEXT SUBBAND TO SEND (500-1000 Hz)} \]
  \[ \text{RET} \]
NOT ALL SUBBAND 6 BITS SENT, PREPARE NEXT BIT

CONT5 SAR 0, COUNT
RET

A BIT OF SUBBAND 6 (2500-3000 Hz) HAS TO BE SENT
ONLY ACCESSED BY UNVOICED AND INTERMEDIATE STRATEGY

NEXT6 ZALS DEMO NEW ACCESS CHECK
    BZ LOOP6

INITIALIZATION FOR A NEW SUBBAND 6 ACCESS
    ZAC
    SACL DEMO NO NEW ACCESS ANYMORE
    LACK COD6
    TBLR DATA LOAD SUBBAND 6 DATA INTO SEND BUFFER
    LACK 1
    SACL COUNT 2 BITS TO SEND FOR ALL POSSIBLE STRATEGIES
    LAC DATA, 6 ADJUST DATA ACCORDING TO BITSELECTOR
    SACL DATA DATA=0000 0000 0000 0000

SEND A SUBBAND 6 BIT
LOOP6 SEND
    LAR 0, COUNT
    BAWZ CONT6

ALL SUBBAND 6 BITS SENT AND ALSO END OF 15 BIT DATABLOCK
CHECK FOR END OF FRAME
    LACK >FF
    SACL DEMO NEW ACCESS MARK
    LACK 7
    XOR FRAME
    BNZ NENDF

END OF FRAME, PREPARE NEXT FRAME
ENDFRM ZAC
    SACL BAND NEXT TO SEND=FAM
RET

NOT END OF FRAME, PREPARE NEXT DATABLOCK
NENDF LACK 1
    SACL BAND NEXT SUBBAND TO SEND (0-500 Hz)
RET

NOT ALL SUBBAND 6 BITS SENT, PREPARE NEXT BIT
CONT6 SAR 0, COUNT
RET

MULTIPLEXING DONE
APPENDIX F

Main program SBC receiver part
DECODER PROGRAM

THIS MAIN PROGRAM USES THE SUBROUTINES:
- DEMULTIPLEXING
- PCMABE DECODING
- INVFIR FILTERING

TO REALIZE A 16 kBIT PER SECOND SUBBAND DECODER.

SYSTEM CLOCK RATE: 16 kHz

REPORTS FOR DETAILS:
- PT REPORT BY P. CHITAMU
- FINAL REPORT BY J. BOLT
- FINAL REPORT BY R. KLEUTERS

AUTHOR: ROLAND KLEUTERS
DATE: 29-5-1987

DATA MEMORY INITIALIZATION

default page=0

GENERAL VARIABLES DATA MEMORY PAGE 1

CLOCK EQU 0
MODE EQU 1
AR00 EQU 2
AR01 EQU 3
ACH EQU 4
ACL EQU 5
TREG EQU 6
STATU EQU 7

MSTAT EQU 0
SWITCH EQU 1
FRAME EQU 2
ONE EQU 3
STRAT EQU 4
SSTRAT EQU 5
STATE EQU 6
RES0 EQU 7
RES1 EQU 8
PATH EQU 9
DUTP EQU 10
BOUTP1 EQU 11
BOUTP2 EQU 12

MASK FOR STATE MOD 16
>FF=SEARCH MODE, ELSE=RECEIVE MODE; 0=DECODER ON (Q0)
COUNTER FOR FRAMELENGTH (Q0)
CHECK BIT (Q0)
CONTAINS 0, 1, or 2: VOICED, INTERM. or UMVO. (Q0)
BUFFERING (Q0)
TREE POINTER (Q0)
INTERMEDIATE CALCULATION RESULTS FOR MAIN ROUTINE
INTERMEDIATE CALCULATION RESULTS FOR MAIN ROUTINE
CONTAINS STATE/2=CHANNEL TO PROCESS (Q0)
8 kHz OUTPUT SAMPLE OF TM532010 (Q15)
BUFFERED 8 kHz OUTPUT SAMPLE OF TM532010 (Q15)
16 kHz INTERPOLATION OF TM532010 OUTPUT (Q15)
FIRST INVQMF STAGE VARIABLES

BUFFER VARIABLES OF LOWI (0-2 kHz) AND HIGHI (2-4 kHz) FOR = QI RESP. Q16
A1 EQU 13
B1 EQU 14
C1 EQU 15

INPUT TO (FILTER PART OF) FIRST INVQMF STAGE FORMAT=Q15
INPF EQU 16

BUFFER VARIABLES OF LOWI (0-2 kHz)
FORMAT=Q15
XF2 EQU 17  AFTER FILTERING INPUT STORED IN XF2, SO
XF3 EQU 18  XF1 NOT NEEDED
XF4 EQU 19
XF5 EQU 20
XF6 EQU 21
XF7 EQU 22
XF8 EQU 23
XF9 EQU 24
XF10 EQU 25
XF11 EQU 26
XF12 EQU 27
XF13 EQU 28
XF14 EQU 29
XF15 EQU 30
XF16 EQU 31

DELAY VARIABLES OF HIGN (2-4 kHz)
FORMAT=Q15
XF18 EQU 32  AFTER FILTERING INPUT STORED IN XF18, SO
XF19 EQU 33  XF17 NOT NEEDED
XF20 EQU 34
XF21 EQU 35
XF22 EQU 36
XF23 EQU 37
XF24 EQU 38
XF25 EQU 39
XF26 EQU 40
XF27 EQU 41
XF28 EQU 42
XF29 EQU 43
XF30 EQU 44
XF31 EQU 45
XF32 EQU 46

SECOND INVQMF STAGE VARIABLES

BUFFER VARIABLES OF LOW2 (0-1 kHz) AND HIGH2 (1-2 kHz) FORMAT=Q14
A2 EQU 47
B2 EQU 48
C2 EQU 49
• BUFFER VARIABLES OF LOW3 (3-4 kHz) AND HIGH3 (2-3 kHz)  FORMAT=Q17
A3 EQU 50
B3 EQU 51
C3 EQU 52

• INPUT TO (FILTER PART OF) SECOND INVQMF STAGE
INPS EQU 53

• DELAY VARIABLES OF LOW3, HIGH3, LOW3 AND HIGH3 (TIME SHARING)
XS2 EQU 54
XS3 EQU 55
XS4 EQU 56
XS5 EQU 57
XS6 EQU 58
XS7 EQU 59
XS8 EQU 60

• THIRD INVQMF STAGE VARIABLES

• BUFFER VARIABLES OF LOW4 (0-500 Hz) AND HIGH4 (500-1000 Hz)  FORMAT=Q14
A4 EQU 61
B4 EQU 62
C4 EQU 63

• BUFFER VARIABLES OF LOW5 (1500-2000 Hz) AND HIGH5 (1000-1500 Hz)  FORMAT=Q15
A5 EQU 64
B5 EQU 65
C5 EQU 66

• BUFFER VARIABLES OF LOW7 (2000-2500 Hz) AND HIGH7 (2500-3000 Hz)  FORMAT=Q18
A7 EQU 67
B7 EQU 68
C7 EQU 69

• INPUT TO (FILTER PART OF) THIRD INVQMF STAGE
INPT EQU 70

• DELAY VARIABLES OF LOW4, HIGH4, LOW5, HIGH5, LOW7 AND HIGH7 (TIME SHARING)
XT2 EQU 71
XT3 EQU 72
XT4 EQU 73
XT5 EQU 74
XT6 EQU 75

• FIRST INVQMF STAGE COEFFICIENTS  FORMAT=Q16

• CF0 EQU 76
CF1 EQU 77
CF2 EQU 78
CF3 EQU 79
CF4 EQU 80
CF5 EQU 81
CF6 EQU 82
CF7 EQU 83
CF8 EQU 84
CF9 EQU 85
CF10 EQU 86
CF11 EQU 87
CF12 EQU 88
CF13 EQU 89
CF14 EQU 90
CF15 EQU 91

* * * * *
SECOND INVQMF STAGE COEFFICIENTS FORMAT=Q16
* * * * *

CS0 EQU 92
CS1 EQU 93
CS2 EQU 94
CS3 EQU 95
CS4 EQU 96
CS5 EQU 97
CS6 EQU 98
CS7 EQU 99

* * * * *
THIRD INVQMF STAGE COEFFICIENTS FORMAT=Q16
* * * * *

CT0 EQU 100
CT1 EQU 101
CT2 EQU 102
CT3 EQU 103
CT4 EQU 104
CT5 EQU 105

* * * * *
PCM&Q VARIABLES
* * * * *

OUTPCM EQU 106 OUTPUT OF PCM&Q DECODER
CALC1 EQU 107 GENERAL VARIABLE TO CALCULATE WITH
CALC2 EQU 108 GENERAL VARIABLE TO CALCULATE WITH
NBIT EQU 109 CONTAINS NR. OF BITS TO QUANTIZE TO
X0 EQU 110 LOWEST POSSIBLE INPUT VALUE OF EXPONENT TABLE
X127 EQU 111 HIGHEST POSSIBLE INPUT VALUE OF EXPONENT TABLE
STEP EQU 112 1/(DISTANCE BETWEEN INPUTS OF EXPONENT TABLE)
GAMMA EQU 113 GAMMA VALUE OF FORMULA
V3FFF EQU 114 >3FFF; HELPCONSTANT

* * * * *
DEMULTIPLIER VARIABLES
* * * * *

POINT EQU 115 FLAG: >FF=SEARCH FAW, 0=SKIP 128 BITS
ERROR EQU 116 CONTAINS NUMBER OF ERROR ALLOWANCES
COUNT EQU 117 MULTIPURPOSE COUNTER
BAND EQU 118 FLAG, CURRENT SUBBAND PROCESSING
DATA EQU 119 STORES INPUT BITS VIA SIGIN
SIGIN EQU 120 VARIABLE TO READ IN SERIAL DATA FROM OUTSIDE
DEMO EQU 121 NEW (FAW or SUBBAND) ACCESS MARK
PROGRAM MEMORY INITIALIZATION

AORG 0
B START BRANCH TO MAIN ROUTINE
B INTRO BRANCH TO INTERRUPT SERVICE ROUTINE
AORG 8 OTHERWISE ERRORS WITH TABLES

GENERAL CONSTANTS

CCLOCK DATA 312 CLOCK RATE (16 kHz)
MMODE DATA 7 MODE FOR ANALOG INTERFACE BOARD
MMSTAT DATA >000F MASK FOR STATE MOD 16

SECOND INVQMF STAGE VARIABLES

DELAY VARIABLES OF LOW2 (0-1 kHz)  FORMAT=Q14
IXS2 DATA 0 AFTER FILTERING INPUT STORED IN IXS2, SO
IXS3 DATA 0 IXS1 NOT NEEDED
IXS4 DATA 0
IXS5 DATA 0
IXS6 DATA 0
IXS7 DATA 0
IXS8 DATA 0

DELAY VARIABLES OF HIGH2 (1-2 kHz)  FORMAT=Q14
IXS10 DATA 0 AFTER FILTERING INPUT STORED IN IXS10, SO
IXS11 DATA 0 IXS9 NOT NEEDED
IXS12 DATA 0
IXS13 DATA 0
IXS14 DATA 0
IXS15 DATA 0
IXS16 DATA 0

DELAY VARIABLES OF LOW3 (3-4 kHz)  FORMAT=Q16
IXS18 DATA 0 AFTER FILTERING INPUT STORED IN IXS18, SO
IXS19 DATA 0 IXS17 NOT NEEDED
IXS20 DATA 0
IXS21 DATA 0
IXS22 DATA 0
IXS23 DATA 0
IXS24 DATA 0

DELAY VARIABLES OF HIGH3 (2-3 kHz)  FORMAT=Q16
IXS26 DATA 0 AFTER FILTERING INPUT STORED IN IXS26, SO
IXS27 DATA 0 IXS25 NOT NEEDED
IXS28 DATA 0
IXS29 DATA 0
IXS30 DATA 0
IXS31 DATA 0
IXS32 DATA 0
### THIRD INVQMF STAGE VARIABLES

#### DELAY VARIABLES OF LOW4 (0-500 Hz)  FORMAT=Q14

<table>
<thead>
<tr>
<th>IIT2</th>
<th>DATA 0</th>
<th>AFTER FILTERING INPUT STORED IN IIT2, SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT3</td>
<td>DATA 0</td>
<td>XXTI NOT NEEDED</td>
</tr>
<tr>
<td>IIT4</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT5</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT6</td>
<td>DATA 0</td>
<td></td>
</tr>
</tbody>
</table>

#### DELAY VARIABLES OF HIGH4 (500-1000 Hz)  FORMAT=Q14

<table>
<thead>
<tr>
<th>IIT8</th>
<th>DATA 0</th>
<th>AFTER FILTERING INPUT STORED IN IIT8, SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT9</td>
<td>DATA 0</td>
<td>XXTI7 NOT NEEDED</td>
</tr>
<tr>
<td>IIT10</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT11</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT12</td>
<td>DATA 0</td>
<td></td>
</tr>
</tbody>
</table>

#### DELAY VARIABLES OF LOW5 (1500-2000 Hz)  FORMAT=Q14

<table>
<thead>
<tr>
<th>IIT14</th>
<th>DATA 0</th>
<th>AFTER FILTERING INPUT STORED IN IIT14, SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT15</td>
<td>DATA 0</td>
<td>XXTI3 NOT NEEDED</td>
</tr>
<tr>
<td>IIT16</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT17</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT18</td>
<td>DATA 0</td>
<td></td>
</tr>
</tbody>
</table>

#### DELAY VARIABLES OF HIGH5 (1000-1500 Hz)  FORMAT=Q14

<table>
<thead>
<tr>
<th>IIT20</th>
<th>DATA 0</th>
<th>AFTER FILTERING INPUT STORED IN IIT20, SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT21</td>
<td>DATA 0</td>
<td>XXTI9 NOT NEEDED</td>
</tr>
<tr>
<td>IIT22</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT23</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT24</td>
<td>DATA 0</td>
<td></td>
</tr>
</tbody>
</table>

#### DELAY VARIABLES OF LOW6 (2000-2500 Hz)  FORMAT=Q17

<table>
<thead>
<tr>
<th>IIT26</th>
<th>DATA 0</th>
<th>AFTER FILTERING INPUT STORED IN IIT26, SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT27</td>
<td>DATA 0</td>
<td>XXTI5 NOT NEEDED</td>
</tr>
<tr>
<td>IIT28</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT29</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT30</td>
<td>DATA 0</td>
<td></td>
</tr>
</tbody>
</table>

#### DELAY VARIABLES OF HIGH6 (2500-3000 Hz)  FORMAT=Q17

<table>
<thead>
<tr>
<th>IIT32</th>
<th>DATA 0</th>
<th>AFTER FILTERING INPUT STORED IN IIT32, SO</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIT33</td>
<td>DATA 0</td>
<td>XXTI1 NOT NEEDED</td>
</tr>
<tr>
<td>IIT34</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT35</td>
<td>DATA 0</td>
<td></td>
</tr>
<tr>
<td>IIT36</td>
<td>DATA 0</td>
<td></td>
</tr>
</tbody>
</table>

### FIRST INVQMF STAGE COEFFICIENTS  FORMAT=Q16

<p>| CCF0  | DATA 45  |
| CCF1  | DATA -92 |
| CCF2  | DATA -83 |
| CCF3  | DATA 277 |
| CCF4  | DATA 93  |
| CCF5  | DATA -620|
| CCF6  | DATA -9  |
| CCF7  | DATA 1178|</p>
<table>
<thead>
<tr>
<th>CCF8</th>
<th>DATA</th>
<th>-274</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCF9</td>
<td>DATA</td>
<td>-2047</td>
</tr>
<tr>
<td>CCF10</td>
<td>DATA</td>
<td>955</td>
</tr>
<tr>
<td>CCF11</td>
<td>DATA</td>
<td>3470</td>
</tr>
<tr>
<td>CCF12</td>
<td>DATA</td>
<td>-2579</td>
</tr>
<tr>
<td>CCF13</td>
<td>DATA</td>
<td>-5541</td>
</tr>
<tr>
<td>CCF14</td>
<td>DATA</td>
<td>8425</td>
</tr>
<tr>
<td>CCF15</td>
<td>DATA</td>
<td>30566</td>
</tr>
</tbody>
</table>

---

**SECOND INVQMF STAGE COEFFICIENTS FORMAT=Q16**

<table>
<thead>
<tr>
<th>CCS0</th>
<th>DATA</th>
<th>69</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCS1</td>
<td>DATA</td>
<td>-331</td>
</tr>
<tr>
<td>CCS2</td>
<td>DATA</td>
<td>-170</td>
</tr>
<tr>
<td>CCS3</td>
<td>DATA</td>
<td>1812</td>
</tr>
<tr>
<td>CCS4</td>
<td>DATA</td>
<td>-633</td>
</tr>
<tr>
<td>CCS5</td>
<td>DATA</td>
<td>-5924</td>
</tr>
<tr>
<td>CCS6</td>
<td>DATA</td>
<td>6409</td>
</tr>
<tr>
<td>CCS7</td>
<td>DATA</td>
<td>31525</td>
</tr>
</tbody>
</table>

---

**THIRD INVQMF STAGE COEFFICIENTS FORMAT=Q16**

<table>
<thead>
<tr>
<th>CCT0</th>
<th>DATA</th>
<th>-250</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCT1</td>
<td>DATA</td>
<td>1236</td>
</tr>
<tr>
<td>CCT2</td>
<td>DATA</td>
<td>-178</td>
</tr>
<tr>
<td>CCT3</td>
<td>DATA</td>
<td>-5551</td>
</tr>
<tr>
<td>CCT4</td>
<td>DATA</td>
<td>5798</td>
</tr>
<tr>
<td>CCT5</td>
<td>DATA</td>
<td>31745</td>
</tr>
</tbody>
</table>

---

**PCMAQ8's TABLES**

**D(N) FOR 7 SUBBANDS FORMAT=Q22**

<table>
<thead>
<tr>
<th>O6D1</th>
<th>DATA</th>
<th>-10486</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUM1</td>
<td>DATA</td>
<td>0</td>
</tr>
<tr>
<td>O6D4</td>
<td>DATA</td>
<td>-13642</td>
</tr>
<tr>
<td>O6D5</td>
<td>DATA</td>
<td>-23112</td>
</tr>
<tr>
<td>O6D2</td>
<td>DATA</td>
<td>-10486</td>
</tr>
<tr>
<td>DUM2</td>
<td>DATA</td>
<td>0</td>
</tr>
<tr>
<td>O6D3</td>
<td>DATA</td>
<td>-13642</td>
</tr>
<tr>
<td>O6D6</td>
<td>DATA</td>
<td>-23112</td>
</tr>
</tbody>
</table>

---

**D(N) FOR 7 SUBBANDS FORMAT=Q14**

<table>
<thead>
<tr>
<th>DN1</th>
<th>DATA</th>
<th>-13563</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUM3</td>
<td>DATA</td>
<td>0</td>
</tr>
<tr>
<td>DN4</td>
<td>DATA</td>
<td>-14796</td>
</tr>
<tr>
<td>DN5</td>
<td>DATA</td>
<td>-18495</td>
</tr>
<tr>
<td>DN2</td>
<td>DATA</td>
<td>-13563</td>
</tr>
<tr>
<td>DUM4</td>
<td>DATA</td>
<td>0</td>
</tr>
<tr>
<td>DN3</td>
<td>DATA</td>
<td>-14796</td>
</tr>
<tr>
<td>DN6</td>
<td>DATA</td>
<td>-18495</td>
</tr>
<tr>
<td>Allocation for Voiced Strategy (1)</td>
<td>Format=Q0</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>BAVO11 Data 4</td>
<td>Subband 1</td>
<td></td>
</tr>
<tr>
<td>BAVO14 Data 2</td>
<td>Subband 4</td>
<td></td>
</tr>
<tr>
<td>BAVO15 Data 2</td>
<td>Subband 5</td>
<td></td>
</tr>
<tr>
<td>BAVO12 Data 4</td>
<td>Subband 2</td>
<td></td>
</tr>
<tr>
<td>BAVO13 Data 3</td>
<td>Subband 3</td>
<td></td>
</tr>
<tr>
<td>BAVO16 Data 0</td>
<td>Subband 6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Allocation for Intermediate Strategy (1)</th>
<th>Format=Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAINT1 Data 4</td>
<td>Subband 1</td>
</tr>
<tr>
<td>BAINT4 Data 2</td>
<td>Subband 4</td>
</tr>
<tr>
<td>BAINT5 Data 2</td>
<td>Subband 5</td>
</tr>
<tr>
<td>BAINT2 Data 3</td>
<td>Subband 2</td>
</tr>
<tr>
<td>BAINT8 Data 0</td>
<td></td>
</tr>
<tr>
<td>BAINT3 Data 2</td>
<td>Subband 3</td>
</tr>
<tr>
<td>BAINT6 Data 2</td>
<td>Subband 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Allocation for Unvoiced Strategy (2)</th>
<th>Format=Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAUV1 Data 2</td>
<td>Subband 1</td>
</tr>
<tr>
<td>BAUV4 Data 3</td>
<td>Subband 4</td>
</tr>
<tr>
<td>BAUV5 Data 2</td>
<td>Subband 5</td>
</tr>
<tr>
<td>BAUV2 Data 3</td>
<td>Subband 2</td>
</tr>
<tr>
<td>BAUV8 Data 0</td>
<td></td>
</tr>
<tr>
<td>BAUV3 Data 3</td>
<td>Subband 3</td>
</tr>
<tr>
<td>BAUV6 Data 2</td>
<td>Subband 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Storage of Coded Subband Samples (Input PCM/DEcoder)</th>
<th>Format=Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>COD1 Data 0</td>
<td>Subband 1</td>
</tr>
<tr>
<td>COD4 Data 0</td>
<td>Subband 4</td>
</tr>
<tr>
<td>COD5 Data 0</td>
<td>Subband 5</td>
</tr>
<tr>
<td>COD2 Data 0</td>
<td>Subband 2</td>
</tr>
<tr>
<td>COD12 Data 0</td>
<td></td>
</tr>
<tr>
<td>COD3 Data 0</td>
<td>Subband 3</td>
</tr>
<tr>
<td>COD6 Data 0</td>
<td>Subband 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Storage of Previous Coded Subband Samples (Stepsize Adaptation)</th>
<th>Format=Q0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCOD1 Data 0</td>
<td>Subband 1</td>
</tr>
<tr>
<td>PCOD4 Data 0</td>
<td>Subband 4</td>
</tr>
<tr>
<td>PCOD5 Data 0</td>
<td>Subband 5</td>
</tr>
<tr>
<td>PCOD2 Data 0</td>
<td>Subband 2</td>
</tr>
<tr>
<td>PCOD12 Data 0</td>
<td></td>
</tr>
<tr>
<td>PCOD3 Data 0</td>
<td>Subband 3</td>
</tr>
<tr>
<td>PCOD6 Data 0</td>
<td>Subband 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantizer Constants</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSSTEP Data 19637</td>
<td>1/(Distance between inputs of exponent table) (Q7)</td>
</tr>
<tr>
<td>GAMMA Data 32440</td>
<td>GAMMA=0.99 (Q15)</td>
</tr>
<tr>
<td>VV3FFF Data 3FFF</td>
<td>3FFF; HELPCONSTANT</td>
</tr>
</tbody>
</table>
*  LOGTABLE 2 BIT CASE  
  LOG60  DATA  18268
  LOG61  DATA  -4626
  LOG62  DATA  -4626
  LOG63  DATA  18268
  DUM15  DATA  0
  DUM16  DATA  0
  DUM17  DATA  0
  DUM18  DATA  0
  DUM19  DATA  0
  DUM20  DATA  0
  DUM21  DATA  0
  DUM22  DATA  0
  DUM23  DATA  0
  DUM24  DATA  0

*  LOGTABLE 3 BIT CASE  
  LOG630  DATA  11540
  LOG631  DATA  0
  LOG632  DATA  0
  LOG633  DATA  -4626
  LOG634  DATA  -4626
  LOG635  DATA  0
  LOG636  DATA  0
  LOG637  DATA  11540
  DUM25  DATA  0
  DUM26  DATA  0
  DUM27  DATA  0
  DUM28  DATA  0

*  LOGTABLE 4 BIT CASE  
  LOG640  DATA  24918
  LOG641  DATA  19728
  LOG642  DATA  13377
  LOG643  DATA  5189
  LOG644  DATA  -2999
  LOG645  DATA  -2999
  LOG646  DATA  -2999
  LOG647  DATA  -2999
  LOG648  DATA  -2999
  LOG649  DATA  -2999
  LOG650  DATA  -2999
  LOG651  DATA  -2999
  LOG652  DATA  5189
  LOG653  DATA  13377
  LOG654  DATA  19728
  LOG655  DATA  24918

*  LOWEST AND HIGHEST POSSIBLE INPUTS OF EXPONENT TABLE  
  XX10  DATA  -13563  SUBBAND 1
  XX1127  DATA  0  SUBBAND 1
  DUM29  DATA  0
  DUM30  DATA  0
  XX40  DATA  -14796  SUBBAND 4
  XX4127  DATA  -1233  SUBBAND 4
  XX50  DATA  -18495  SUBBAND 5
  XX5127  DATA  -4932  SUBBAND 5
  XX20  DATA  -13563  SUBBAND 2
SUBBAND 2
DUM31 DATA 0
DUM32 DATA 0

SUBBAND 3
XX31 DATA -14796

SUBBAND 4
XX60 DATA -18495

SUBBAND 5
XX6127 DATA -4932

SUBBAND 6

EXponent TABLE OF DECODER

FORMATS ARE : SUBBAND 1 Q15
SUBBAND 2 Q15
SUBBAND 3 Q16
SUBBAND 4 Q16
SUBBAND 5 Q19
SUBBAND 6 Q19

CEXPO DATA 16
DATA 17
DATA 18
DATA 19
DATA 20
DATA 22
DATA 23
DATA 24
DATA 26
DATA 27
DATA 29
DATA 31
DATA 33
DATA 35
DATA 37
DATA 39
DATA 42
DATA 44
DATA 47
DATA 50
DATA 53
DATA 56
DATA 60
DATA 64
DATA 68
DATA 72
DATA 76
DATA 81
DATA 86
DATA 91
DATA 97
DATA 103
DATA 109
DATA 116
DATA 123
DATA 131
DATA 139
DATA 148
DATA 157
DATA 166
DATA 177
DATA 188
DATA 199
DATA 211
DATA 225
DATA 238
DATA 253
DATA 269
DATA 286
DATA 303
DATA 322
DATA 342
DATA 363
DATA 385
DATA 409
DATA 425
DATA 462
DATA 490
DATA 520
DATA 553
DATA 587
DATA 623
DATA 662
DATA 703
DATA 746
DATA 792
DATA 841
DATA 893
DATA 949
DATA 1007
DATA 1070
DATA 1136
DATA 1206
DATA 1281
DATA 1360
DATA 1444
DATA 1534
DATA 1628
DATA 1729
DATA 1836
DATA 1950
DATA 2070
DATA 2199
DATA 2335
DATA 2479
DATA 2632
DATA 2795
DATA 2968
DATA 3152
DATA 3347
DATA 3554
DATA 3774
DATA 4008
DATA 4256
DATA 4519
DATA 4798
DATA 5095
DATA 5411
DATA 5745
DATA 6101
DATA 6478  
DATA 6879  
DATA 7305  
DATA 7757  
DATA 8237  
DATA 8746  
DATA 9288  
DATA 9862  
DATA 10473  
DATA 11121  
DATA 11809  
DATA 12539  
DATA 13315  
DATA 14139  
DATA 15014  
DATA 15943  
DATA 16929  
DATA 17977  
DATA 19089  
DATA 20270  
DATA 21525  
DATA 22856  
DATA 24271  
DATA 25772  
DATA 27367  
DATA 29060  
DATA 30859  
DATA 32767  

# MAIN ROUTINE

# INITIALIZATIONS

START DINT

INITIALIZE STATUS BITS
LARP 0
LDPK 0
SDVM

INITIALIZE GENERAL VARIABLES OF DATA MEMORY PAGE 1
LDPK 1
LACK CCLOCK
TBLR CLOCK CLOCK RATE (16 kHz)
LACK MNMODE
TBLR MNODE NODE FOR ANALOG INTERFACE BOARD
LDPK 0

INITIALIZE GENERAL VARIABLES OF DATA MEMORY PAGE 0
LACK MNSTAT
TBLR MNSTAT MASK FOR STATE MOD 16
LACK YFF
SACL SWITCH DEMULTIPLEXER IN SEARCH MODE; DECODER OFF
ZAC
SACL FRAME  STATUS FOR DECORER
LACK 1
SACL ONE  '1' FOR LOGICAL MANIPULATIONS
SACL STRAT  BY DEFAULT LOAD INTERMEDIATE STRATEGY
SACL STRAT  ALSO INTERMEDIATE STRATEGY FOR BUFFER
ZAC
SACL STATE  INIT TREE POINTER
SACL DOUTP  ZERO 8 kHz OUTPUT SAMPLE OF TMS2010
SACL DOUTP1  ZERO BUFFERED 8 kHz OUTPUT SAMPLE OF TMS2010
SACL DOUTP2  ZERO 16 kHz INTERPOLATED OUTPUT OF TMS2010

* INITIALIZE PCMAD6 VARIABLES
  LACK SSTEP
  TBLR STEP  1/(DISTANCE BETWEEN INPUTS OF EXPONENT TABLE)
  LACK GAMMA
  TBLR GAMMA  GAMMA VALUE OF FORMULA (0.99)
  LACK VV3FFF
  TBLR V3FFF  >3FF; HELPCONSTANT

* INITIALIZE DEMULTIPLEXER VARIABLES
  LACK OFF
  SACL POINT  SEARCH FIRST FAW

* LOAD INVQRF COEFFICIENTS INTO DATA RAM
  LARK 1,CT5
  LARK 0,29
  LACK CCT5
  LOAD LARP 1
  TBLR 0,-0
  SUB ONE
  BANZ LOAD

* INITIALIZE ANALOG INTERFACE BOARD (AIB)
  LDPK 1
  OUT NODE,0  PROGRAM AIB
  OUT CLOCK,1  SET CLOCK RATE OF AIB
  LDPK 0
  OUT DUTP,2  CLEAR ADIS (PROTECTION AGAINST RESET INTERRUPTS)
  OUT DUTP,3  CLEAR AD2S (PROTECTION AGAINST RESET INTERRUPTS)

* INITIALIZATION AND VARIABLE LOADING DONE, NOW BEGIN
  EINT

* ...
* ...
* AORTA
* ...
* ODD CYCLE OF 16 kHz CLOCK MUST JUST HAVE PASSED
  WAIT EV LAC STATE
  AND ONE
  BANZ WAIT
  EVEN CYCLE OF 16 kHz CLOCK HAS PASSED (BIT 0 OF STATE=0)
  WAIT LAC STATE
  AND ONE
  BZ WAIT
  ODD CYCLE OF 16 kHz CLOCK HAS JUST PASSED (BIT 0 OF STATE=1)
*
# DURING THE PROCESSING OF A SUBBAND SAMPLE STATE CAN CHANGE AS A
# RESULT OF AN INTERRUPT, SO DEFINE PATH=STATE/2
  LAC STATE,IS
  SACL PATH
#
# CHECK IF DECODER IS ON (i.e. IN SYNCHRONIZATION)
  ZALS SWITCH
  BZ ACTSUB   SWITCH=0?
#
# DECODER IS NOT ON; OUTPUT A ZERO AND WAIT FOR NEXT CYCLE
  ZAC
  SACL OUTP
  B WAITEV
#
# DECODER IS ON; ACTUAL SUBBAND DECODING
  ACTSUB CALL PCMAGB
  CALL INVQMF
  B WAITEV
#
#------------------------------------------------------------------
# INTERRUPT SERVICE ROUTINE
#------------------------------------------------------------------
# INTRO DINT
#
# SAVE STATUS OF TMS32010
  SST STATU
  MPY ONE T REGISTER TO P REGISTER
  LDPK 1
  SAR 0,AR00
  SAR 1,AR01
  SACH ACH
  SACL ACL
  PAC
  SACL TRES
  LDPK 0
#
# STATE UPDATE
  LAC STATE
  ADD ONE.
  AND MSTAT     STATE MOD 16
  SACL STATE
#
# RUN THROUGH DEMULTIPLEXER
  CALL DEMUX
#
# CREATE A 16 kHz OUTPUT SAMPLE OF THE TMS32010
# OUTPUT IS 8 kHz INVQMF TREE OUTPUT IF CYCLE IS ODD
# OUTPUT IS INTERPOLATION OF 8 kHz INVQMF TREE OUTPUT IF CYCLE IS EVEN
  LAC STATE
  AND ONE
  BZ INTPOl
  LAC OUTP     CYCLE IS ODD; OUTPUT IS INVQMF OUTPUT
  SACL BOUTP1
  OUT BOUTP2,2
  B RSSTAT
INTFOL LAC BOUTP1,15 CYCLE IS EVEN; OUTPUT IS INTERPOLATION
ADD BOUTF2,15
SACH BOUTP2
OUT BOUTP2,2
DMOV BOUTP1

* RESTORE STATUS OF TMS32010
RSSTAT LDPK 1
LAR 0,AR00
LAR 1,AR01
ZALH ACH
ADDS ACL
LT TRES
LST STATU

* RETURN TO PLACE OF INTERRUPT AND CONTINUE
EINT
RET

* DECODER PROGRAM DONE

==============================================================================
APPENDIX G

Subroutine PCMAQB (decoder)
*DECODING ENTRY POINT*

THIS SUBROUTINE INCORPORATES:
- BACKWARD STEPSIZE ADAPTATION OF THE QUANTIZER
- -2, 3 OR 4 BIT PCM DECODING OF A CODED SAMPLE INTO A QUANTIZED SAMPLE OF THE SUBBAND BEING PROCESSED

INPUT: FAW + 1 kHz SAMPLES (8 STREAMS)
OUTPUT: 1 kHz SAMPLES (8 STREAMS)

BIT ALLOCATION ACCORDING TO SSTRAT:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICED</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UNVOICED</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

AUTHOR: ROLAND KLEUTERS
DATE: 29-5-1987

---

PCMAQB NOP ENTRY POINT

READ IN FROM TABLE THE NUMBER OF CODE BITS FOR THE SUBBAND IN PROCESSION
LACK LANO
ADD PATH
ADD SSTRAT, 3
TBLR NBIT NBIT CONTAINS NUMBER OF CODE BITS

READ IN THE LOWEST AND HIGHEST POSSIBLE INPUT VALUE OF THE EXPONENT TABLE
LACK LXXO
ADD PATH, 1
TBLR IO IO CONTAINS LOWEST POSSIBLE INPUT OF EXP. TABLE
ADD ONE
TBLR LL27 LL27 CONTAINS HIGHEST POSSIBLE INPUT OF EXP. TABLE

CHECK IF NUMBER OF CODE BITS (NBIT) GREATER THAN ZERO
LAC NBIT
BGZ READIN DECODE?

NO DECODING HAS TO TAKE PLACE (NBIT=0)
ZAC ZERO "CODED" RESULT; I(n):=0
SAIC CALCI
LACK PCODI
ADD PATH
TBLR CALCI I(n) SAVED IN PREVIOUS CODED SUBBAND SAMPLES TABLE
LACK DMI MINIMIZE STEPSIZE; d(n):=IO
ADD PATH
TBLR IO d(n) SAVED IN TABLE
ZAC ZERO "DECODED" RESULT; QUANTIZED SUBBAND SAMPLE:=0
SACL OUTPCM QUANTIZED SUBBAND SAMPLE SAVED IN OUTPCM
RET
* DETERMINE STEPSIZE i.e. DELTA(n) BY BACKWARD STEPSIZE ADAPTATION

* DETERMINE d(n) = LOG(DELTA(n))

* READ IN THE PREVIOUS CODED RESULT RESULT I(n-1) FROM TABLE

READIN LACK PCOD1
ADD PATH
TBLR CALC1 CALC1 CONTAINS THE PREVIOUS CODED RESULT I(n-1)

* DETERMINE m([I(n-1)]) = LOG(M([I(n-1)])) FROM TABLES

LAC NBIT
SUB ONE,1
SACL CALC2
LACK LOG22
ADD CALC2,4
ADD CALC1
TBLR RES1 RES1 CONTAINS m([I(n-1)]) = LOG(M([I(n-1)]))

* DETERMINE (1 - GAMMA)* DC FROM TABLE

LACK ODD1
ADD PATH
TBLR CALC1 CALC1 CONTAINS (1 - GAMMA)* DC

* DETERMINE GAMMA*d(n-1) FROM TABLE

LAC DNI
ADD PATH
TBLR RESO RESO CONTAINS d(n-1)
LT GAMMA
MPY RESO
PAC
SACh CALC2,1 CALC2 CONTAINS GAMMA*d(n-1)

* DETERMINE GAMMA*d(n-1)* m([I(n-1)]) + (1 - GAMMA)* DC

ZALH CALC2
ADD RES1,12
ADD CALC1,8
SACh CALC1 CALC1 CONTAINS GAMMA*d(n-1)* m([I(n-1)]) + (1 - GAMMA)* DC, i.e. d(n)

* CONTROL THE LIMITATION OF d(n) TO ITS RANGE AND SAVE d(n) IN TABLE

LAC CALC1
SUB I127
BGZ SATDNH d(n) GREATER THAN MAXIMUM POSSIBLE VALUE OF d(n)?
LAC X0
SUB CALCI
BGZ SATDNL d(n) LESS THAN MINIMUM POSSIBLE VALUE OF d(n)?
B SAVEDN NO SATURATION NEEDED
SATDNH LAC I127 SATURATE d(n) TO MAXIMUM POSSIBLE VALUE
SACh CALC1 B SAVEDN
SATDNL LAC X0 SATURATE d(n) TO MINIMUM POSSIBLE VALUE
SACh CALC1
SAVEDN LACK DN1
ADD PATH
TBLW CALC1
#
# DETERMINE DELTA(n)=EXP(d(n))
#
# ROUND d(n) TO ONE OF THE ENTRY POINTS OF THE EXPONENT TABLE
ZAHN CALC1
SUCH I0
SACH CALC1 CALC1 CONTAINS d(n)-I0
LT CALC1
MPY STEP
PAC
SACH CALC1 CALC1 CONTAINS (d(n)-I0)*STEP
LAC CALC1,II
SACH CALC1 CALC1 CONTAINS INTEGRAL(d(n)-I0)*STEP=ENTRY POINT
#
# DETERMINE THE OUTPUT OF THE EXPONENT TABLE
LACK CEIPO
ADD CALC1
TBLR CALC1 CALC1 CONTAINS OUTPUT OF EXPONENT TABLE=DELTA(n)
#
# DECODE PCMAB DECODER INPUT (I(n)) INTO A QUANTIZED SUBBAND SAMPLE
#
# READ IN I(n) FROM TABLE OF CODED SUBBAND SAMPLES AND
# SAVE I(n) IN TABLE OF PREVIOUS CODED SUBBAND SAMPLES
LACK COD1
ADD PATH
TBLR CALC2 CALC2 CONTAINS CODED RESULT I(n)
LACK PCOD1
ADD PATH
TBLW CALC2 I(n) SAVED IN TABLE
#
# DETERMINE (I(n)+0.5)*DELTA(n)
LAC CALC2,1
ADD ONE
SACL CALC2 CALC2 CONTAINS I(n)+0.5 (FORMAT=Q1)
LT CALC2
MPY CALC1
PAC
ACCU CONTAINS (I(n)+0.5)*DELTA(n), i.e. THE
QUANTIZED SUBBAND SAMPLE (FORMAT=Q16,Q17 or Q20)
#
# CONTROL THE LIMITATION OF THE QUANTIZED SUBBAND SAMPLE TO ITS RANGE AND
# SAVE THE QUANTIZED SUBBAND SAMPLE (IN Q14,Q15 or Q18 FORMAT) IN OUTPCM
SACL RES0 RES0=DODD DODD DODD DD-- ; D=DATA BIT --DON'T CARE
SACH RES1 RES1=SSSS SSSS SSSS SSSD ; S=SIGNBIT
LAC RES1
SUB ONE
B6Z SATOPH QUANTIZED SUBBAND SAMPLE GREATER THAN MAX. VALUE?
ZAC
SUB ONE,1
SUB RES1
B6Z SATOPL QUANTIZED SUBBAND SAMPLE LESS THAN MIN. VALUE?
LAC RES1,14 SAVE QUANTIZED SUBBAND SAMPLE WITHOUT SATURATION
SACL RESI
LAC RESO,14
SACH RESO
LAC RESO
AND V3FFF
ADD RESI
SACL OUTPCM
RET

SATOPH LAC V3FFF
ADD ONE,14
SACL OUTPCM
RET

SATOPL LAC ONE,15
SACL OUTPCM
RET

PCMQR DECODING DONE

===============================================================================================================

RES1=5DD0 0000 0000 0000
RESO=-0D DDDD DDDD DDDD
ACCU LOW=0DDD DDDD DDDD
ACCU LOW=SDDD DDDD DDDD DDDD (G=-,-015,016 FORMAT)

Saturate quantized subband sample to max. and save
Saturate quantized subband sample to min. and save
APPENDIX H

Subroutine INVQMF
INVQMF FILTERING

THIS SUBROUTINE INCORPORATES:
  - INVERSE QMF FILTER TREE
  - 8 kHz SPEECH SIGNAL RECONSTRUCTION
  - TAKE OVER OF NEW VOICING STRATEGY FOR
  - PCM DECODING

INPUT: 1 kHz SAMPLES (8 STREAMS)
OUTPUT: 8 kHz SAMPLES (1 STREAM)

SUBBAND PROCESSING: 1, 8, 5, 2, 7, 3, 6

BIT ALLOCATION:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICED</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UNVOICED</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FAN's:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>INTERMEDIATE</th>
<th>96</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICED</td>
<td>&gt;76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>&gt;96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNVOICED</td>
<td>&gt;CC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AUTHOR: ROLAND KLEUTERS
DATE: 29-5-1987

MACRO'S

MACRO TO REALIZE FILTERING IN LOW BRANCH OF FIRST INVQMF STAGE

FLTFLD #MACRO
ZAC
LT XF16
MPY CF1
LTD XF15
MPY CF3
LTD XF14
MPY CF5
LTD XF13
MPY CF7
LTD XF12
MPY CF9
LTD XF11
MPY CF11
LTD XF10
MPY CF13
LTD XF9
MPY CF15
LTD XF8
MPY CF14
LTD XF7
MPY CF12
LTD XF6
MPY CF10
LTD XF5
MPY CF8
LTD XF4
FILTER RESULT HAS TO BE AMPLIFIED WITH 2

8 kHz OUTPUT OF TMS32010

* MACRO TO REALIZE FILTERING IN HIGH BRANCH OF FIRST INVQMF STAGE

```
FILTER RESULT HAS TO BE AMPLIFIED WITH 2
8 kHz OUTPUT OF TMS32010
```

```
**FILTER RESULT HAS TO BE AMPLIFIED WITH 2
8 kHz OUTPUT OF TMS32010**
```
MACRO TO REALIZE FILTERING IN LOW BRANCH OF SECOND INVQMF STAGE
FLTSLO #MACRO
ZAC LT X58 MPY CS1 LTD X57 MPY CS3 LTD X56 MPY CS5 LTD X55 MPY CS7 LTD X54 MPY CS6 LTD X53 MPY CS4 LTD X52 MPY CS2 LTD IMP5 MPY CS0 APAC FILTER RESULT HAS TO BE AMPLIFIED WITH 2
SACL RES0 SACH RES1 ADD RES0 ADDH RES1 SACH RES0 4 kHz INPUT TO FIRST INVQMF STAGE
$END

MACRO TO REALIZE FILTERING IN HIGH BRANCH OF SECOND INVQMF STAGE
FLTSHI #MACRO
ZAC LT X58 MPY CS0 LTD X57 MPY CS2 LTD X56 MPY CS4 LTD X55 MPY CS6 LTD X54 MPY CS7 LTD X53 MPY CS5 LTD X52 MPY CS3 LTD IMP5 MPY CS1 APAC FILTER RESULT HAS TO BE AMPLIFIED WITH 2
SACL RES0 SACH RES1 ADD RES0 ADDH RES1 SACH RES0 4 kHz INPUT TO FIRST INVQMF STAGE
$END
* MACRO TO REALIZE FILTERING IN LOW BRANCH OF THIRD INVQMF STAGE

FILTLO #MACRO
ZAC
LT X16
MPY CT1
LTD X15
MPY CT3
LTD X14
MPY CT5
LTD X13
MPY CT4
LTD X12
MPY CT2
LTD INPT
MPY CT0
APAC
SAACL RES0
SACH RES1
ADD RES0
ADDH RES1
SACH RES0
FILTER RESULT HAS TO BE AMPLIFIED WITH 2
2 kHz INPUT TO SECOND INVQMF STAGE
$END

* MACRO TO REALIZE FILTERING IN HIGH BRANCH OF THIRD INVQMF STAGE

FILTTHI #MACRO
ZAC
LT X16
MPY CT0
LTD X15
MPY CT2
LTD X14
MPY CT4
LTD X13
MPY CT5
LTD X12
MPY CT3
LTD INPT
MPY CT1
APAC
SAACL RES0
SACH RES1
ADD RES0
ADDH RES1
SACH RES0
FILTER RESULT HAS TO BE AMPLIFIED WITH 2
2 kHz INPUT TO SECOND INVQMF STAGE
$END

* MACRO TO LOAD DELAY VARIABLES OF SECOND INVQMF STAGE

LOADDS #MACRO I
LACK :X.5;
LARK 0,I52
TLBR ++
ADD ONE
TLBR ++
ADD ONE
TLBR ++
ADD ONE
TLBR ++
ADD ONE
MACRO TO SAVE DELAY VARIABLES OF SECOND INVQMF STAGE

SAVED S
MACRO I
LACK :I.S:
LARK 0,152:
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR *
$END

MACRO TO LOAD DELAY VARIABLES OF THIRD INVQMF STAGE

LOADT S
MACRO I
LACK :I.S:
LARK 0,IT2
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR *
$END

MACRO TO SAVE DELAY VARIABLES OF THIRD INVQMF STAGE

SAVEDT S
MACRO I
LACK :I.S:
LARK 0,IT2
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR ++
ADD ONE
TBLR *
$END
ACRO DEFINITION DONE; MAIN PART OF SUBROUTINE BEGINS

INVQMF NOP ENTRY POINT

SELECT PATH

LACK 4
SUB PATH
BGZ PATHGZ
BZ HIGH4 SUBBAND 2
ADD ONE
BZ HIGH6 SUBBAND 7
ADD ONE
BZ HIGH5 SUBBAND 3
B HIGH7 SUBBAND 6
PATHGZ SUB ONE
BZ LOW7 SUBBAND 5
SUB ONE
BZ LOW5 SUBBAND 4
SUB ONE
BZ LOW6 SUBBAND 8

THIRD INVQMF STAGE:

LOW4 0-500 Hz
HIGH4 500-1000 Hz
LOW5 1500-2000 Hz
HIGH5 1000-1500 Hz
LOW6 2500-4000 Hz; NOT CODED
HIGH6 3000-3500 Hz; NOT CODED
LOW7 2000-2500 Hz
HIGH7 2500-3000 Hz
TAKE OVER NEW VOICING STRATEGY

LOW4 DMV A4 DOUBLE BUFFER

INPUT CALCULATION
ZALH B4
SUBH C4
SACH INPT 1 kHz INPUT TO FILTER
LAC OUTPCM
SACL A4 BUFFERED 1 kHz INPUT TO THIRD INVQMF STAGE

LOAD DELAY VARIABLES OF LOW4
LOAD OT XIT2

FILTER 0-500 Hz
FLTTO

SAVE DELAY VARIABLES OF LOW4
SAVEDT XIT2

BRANCH TO SECOND INVQMF STAGE
B LOW2
HIGH4 NOP NO DOUBLE BUFFER

* INPUT CALCULATION
  ZALH C4
  ADDH B4
  SACH INPT 1 kHz INPUT TO FILTER
  LAC OUTPCM
  SACL C4 BUFFERED 1 kHz INPUT TO THIRD INVQMF STAGE

* LOAD DELAY VARIABLES OF HIGH4
  LOADDT X118

* FILTER 500-1000 Hz
  FLTHI

* SAVE DELAY VARIABLES OF HIGH4
  SAVEDT X118

* BRANCH TO SECOND INVQMF STAGE
  B LOWS

LOW5 DMV AS DOUBLE BUFFER

* INPUT CALCULATION
  LAC B5,15
  SUB C5,15
  SACH INPT 1 kHz INPUT TO FILTER
  LAC OUTPCM
  SACL AS BUFFERED 1 kHz INPUT TO THIRD INVQMF STAGE

* LOAD DELAY VARIABLES OF LOWS
  LOADDT X114

* FILTER 1500-2000 Hz
  FLTTL0

* SAVE DELAY VARIABLES OF LOWS
  SAVEDT X114

* BRANCH TO SECOND INVQMF STAGE
  B HIGH5

HIGH5 NOP NO DOUBLE BUFFER

* INPUT CALCULATION
  LAC C5,15
  ADD B5,15
  SACH INPT 1 kHz INPUT TO FILTER
  LAC OUTPCM
  SACL C5 BUFFERED 1 kHz INPUT TO THIRD INVQMF STAGE

* LOAD DELAY VARIABLES OF HIGH5
  LOADDT X120

* FILTER 1000-1500 Hz
  FLTHI
SAVE DELAY VARIABLES OF HIGH5
SAVEDT IIT20

BRANCH TO SECOND INVQMF STAGE
B HIGH2

LOW6 NOP

THE 3500-4000 Hz BAND IS NOT CODED

ZERO 2 kHz INPUT TO SECOND INVQMF STAGE
ZAC
SACL RES0 2 kHz INPUT TO SECOND INVQMF STAGE

BRANCH TO SECOND INVQMF STAGE
B LOW3

HIGH6 NOP

THE 3000-3500 Hz BAND IS NOT CODED

ZERO 2 kHz INPUT TO SECOND INVQMF STAGE
ZAC
SACL RES0

BRANCH TO SECOND INVQMF STAGE
B LOW3

LOW7 DMov A7 DOUBLE BUFFER

INPUT CALCULATION
LAC B7,15
SUB C7,15
SACH INPT 1 kHz INPUT TO FILTER
LAC OUTPCM
SACL A7 BUFFERED 1 kHz INPUT TO THIRD INVQMF STAGE

LOAD DELAY VARIABLES OF LOW7
LOADDT IIT26

FILTER 2000-2500 Hz
FLTTLO

SAVE DELAY VARIABLES OF LOW7
SAVEDT IIT26

BRANCH TO SECOND INVQMF STAGE
B HIGH3

HIGH7 NOP NO DOUBLE BUFFER

INPUT CALCULATION
LAC C7,15
ADD B7,15
SACH INPT 1 kHz INPUT TO FILTER
LAC OUTPCM
SACL C7 BUFFERED 1 kHz INPUT TO THIRD INVQMF STAGE
LOAD DELAY VARIABLES OF HIGH7
LOAD DT IXT32

FILTER 2500-3000 Hz
FLITHI

SAVE DELAY VARIABLES OF HIGH7
SAVEDT IXT32

UPDATE FRAME AND TAKE OVER NEW VOICING STRATEGY IF END OF FRAME
LAR 0, FRAME
BANZ SFRAI
SACL FRAME
DMOV STRAT
B BRSEC
SFRAI SAR 0, FRAME
MOT END OF FRAME; PREPARE NEXT FRAME

BRANCH TO SECOND INVQMF STAGE
BRSEC B HIGH3

SECOND INVQMF STAGE:

LOW2 0kHz-1kHz
HIGH2 1kHz-2kHz
LOW3 3kHz-4kHz
HIGH3 2kHz-3kHz

LOW2 DMOV A2 DOUBLE BUFFER

INPUT CALCULATION
ZALH B2
SUBH C2
SACH IMP5
LAC RES0
SACL A2 BUFFERED 2 kHz INPUT TO SECOND INVQMF STAGE

LOAD DELAY VARIABLES OF LOW2
LOADDS IXS2

FILTER 0-1 kHz
FLTSLD

SAVE DELAY VARIABLES OF LOW2
SAVEDS IXS2

BRANCH TO FIRST INVQMF STAGE
B LOW1

HIGH2 NOP NO DOUBLE BUFFER

INPUT CALCULATION
ZALH C2
ADDH B2
SACH IMP5
LAC RES0
SACL C2 BUFFERED 2 kHz INPUT TO SECOND INVQMF STAGE
LOAD DELAY VARIABLES OF HIGH2
LOADDS IXS10
FILTER 1-2 kHz
FLTSHI
SAVE DELAY VARIABLES OF HIGH2
SAVEDS IXS10
BRANCH TO FIRST INVQMF STAGE
B LOW1
LOW3 DMOV A3 DOUBLE BUFFER
INPUT CALCULATION
LAC B3,15
SUB C3,15
SACH IMPS 2 kHz INPUT TO FILTER
LAC RESO
SACL A3 BUFFERED 2 kHz INPUT TO SECOND INVQMF STAGE
LOAD DELAY VARIABLES OF LOW3
LOADDS IXS18
FILTER 3-4 kHz
FLTSL0
SAVE DELAY VARIABLES OF LOW3
SAVEDS IXS18
BRANCH TO FIRST INVQMF STAGE
B HIGH1
HIGH3 NOP NO DOUBLE BUFFER
INPUT CALCULATION
LAC C3,15
ADD B3,15
SACH IMPS 2 kHz INPUT TO FILTER
LAC RESO
SACL C3 BUFFERED 2 kHz INPUT TO SECOND INVQMF STAGE
LOAD DELAY VARIABLES OF HIGH3
LOADDS IXS26
FILTER 2-3 kHz
FLTSHI
SAVE DELAY VARIABLES OF HIGH3
SAVEDS IXS26
BRANCH TO FIRST INVQMF STAGE
B HIGH1
FIRST INVQMF STAGE:

LOW1: 0kHz-2kHz

HIGH1: 2kHz-4kHz

LOW1 DMV A1 DOUBLE BUFFER

INPUT CALCULATION
ZALH B1
SUB C1,14
SACH INF,1
LAC RES0
SACL A1

BUFFERED 4 kHz INPUT TO FIRST INVQMF STAGE

FILTER 0-2 kHz
FLTLO

INVQMF DONE
RET

HIGH1 NOP

NO DOUBLE BUFFER

INPUT CALCULATION
LAC C1,14
ADDH B1
SACH INF,1
LAC RES0
SACL C1

BUFFERED 4 kHz INPUT TO FIRST INVQMF STAGE

FILTER 2-4 kHz
FLTFO

INVQMF DONE
RET

INVQMF FILTERING DONE
APPENDIX I

Subroutine DEMULTIPLEXING
DEMULTIPLEXING

THIS SUBROUTINE INCORPORATES:
- RECEIVING OF PCM CODED DATA AND
- DEMULTIPLEXING

INPUT: 16 KBPS BITSTREAM
OUTPUT: FAW + 1 kHz SAMPLES (8 STREAMS)

SUBBAND PROCESSING: 1, 8, 4, 5, 2, 7, 3, 6

BITALLOCATION:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOICED</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>INTERMEDIATE</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>UNVOICED</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

FAW's:
- VOICED >76
- INTERMEDIATE >96
- UNVOICED >CC

AUTHOR: ROLAND KLEUTERS
DATA: 27-5-1987

MACRO'S

MACRO TO RECEIVE SIGNBIT AND EXTEND SIGN IN RECEIVE BUFFER
RECSB $MACRO
IN SIGIN,3 SIGNBIT RECEIVED VIA PIN 12 OF EXPANSION PORT
LAC ONE AND SIGIN SELECT SIGNBIT; SIGNBIT IS LSB OF ACCU
SACL DATA DATA=>OOOO or >0001
ZAC EXTEND SIGN IN RECEIVE BUFFER
SUB DATA
SACL DATA DATA=>0000 or >FFFF
$END

MACRO TO RECEIVE ONE BIT
RECEIVE $MACRO
LAC DATA,1
SACL DATA PLACE RESERVED FOR NEW RECEIVE BIT
IN SIGIN,3 RECEIVE BIT RECEIVED VIA PIN 12 OF EXPANSION PORT
LAC ONE
AND SIGIN SELECT RECEIVE BIT; RECEIVE BIT IS LSB OF ACCU
XOR DATA
SACL DATA RECEIVE BIT INSERTED INTO LSB OF DATA
$END

MACRO DEFINITION DONE; MAIN PART OF SUBROUTINE BEGINS

DEMUX NOP ENTRY POINT
CHECK IF DEMULTIPLEXER IS IN ALIGNMENT (SWITCH NEO OFF)
LACK >FF
XOR SWITCH
BNZ RMODE

DEMULTIPLEXER IS NOT IN ALIGNMENT

FAW SEARCH OR 128 BIT (=FRAME) SKIP?
LAC POINT
BZ SKPBIT

FAW SEARCH; INSERT ONE BIT IN SEARCH FOR A FAW RECEIVE
LACK >FF
AND DATA SELECT 8 LSB's OF DATA
SACL DATA DATA CONTAINS 0000 0000 XXXX XXXX

CHECK FAW
LACK >76
XOR DATA VOICED?
BZ VOIFND
LACK >96
XOR DATA INTERMEDIATE?
BZ INTFND
LACK »CC
XOR DATA UNVOICED?
BZ UNVFND
REI AT NEXT ENTRY AGAIN FAW SEARCH

FAW FOUND, SET STRATEGY
VOIFND ZAC STRATEGY=0 (VOICED)
SACL STRAT B INSBIT
INTFND LACK 1 STRATEGY=1 (INTERMEDIATE)
SACL STRAT B INSBIT
UNVFND LACK 2 STRATEGY=2 (UNVOICED)
SACL STRAT

INITIALIZATION FOR 128 BIT (=FRAME) SKIP
INSBIT LACK 127
SACL COUNT COUNT 128 Bits
ZAC
SACL POINT NEXT ENTRY AT SKPBIT RET

SKIP 128 BITS BEFORE SECOND FAW CHECK
SKPBIT RECEIVE
LAR 0,COUNT
BANZ CONTSK
128 BITS SKIPPED
2AC
SACL ERROR NO ERROR ALLOWANCES
LACK >FF
AND DATA SELECT B LSB's OF DATA
SACL DATA DATA CONTAINS 0000 0000 xxxx xxx:
B FAWCHK

128 BITS NOT YET SKIPPED
CONT SK SAR 0,COUNT RET NEXT ENTRY AGAIN AT SKP BIT

Frame Alignment Word CONFIRMATION

FAWCHK NOP
LACK >76
XOR DATA VOICED?
BZ CONF0
LACK >96
XOR DATA INTERMEDIATE?
BZ CONF1
LACK >CC
XOR DATA UNVOICED?
BZ CONF2
LAR 0,ERROR FAW NOT CONFIRMED; ERROR ALLOWED?
BANZ ERR ERROR ALLOWED?

ERROR NOT ALLOWED; SO SWITCH TO MISALIGNMENT
LACK >FF
SACL SWITCH TURN OFF DECODER
SACL POINT RET NEXT ENTRY AT FRAME SEARCH

ERROR ALLOWED; DECREMENT NO. OF ERROR ALLOWANCES; STRATEGY UNCHANGED
ERR SAR 0,ERROR B INISYN

FAW CONFIRMED; SET STRATEGY AND NO. OF ERROR ALLOWANCES
CONF0 ZAC
SACL STRAT SET VOICED STRATEGY (=0)
LACK 4
SACL ERROR 4 FRAMES IN ERROR ALLOWANCE
B INISYN
CONF1 LACK 1
SACL STRAT SET INTERMEDIATE STRATEGY (=1)
LACK 4
SACL ERROR 4 FRAMES IN ERROR ALLOWANCE
B INISYN
CONF2 LACK 2
SACL STRAT SET UNVOICED STRATEGY (=2)
LACK 4
SACL ERROR 4 FRAMES IN ERROR ALLOWANCE
INITIALIZE FOR A NEW FRAME

INISYN LACK OFF
SAACL DEMO NEW ACCESS MARK
LACK 1
SAACL BAND NEXT SUBBAND TO RECEIVE (0-500 Hz)

SYNCHRONIZE DECODER IF NEEDED
LAC SWITCH
BZ INIOUT DECODER ALREADY ON, i.e. SYNCHRONIZATION NEEDED?
LACK 4
SAACL SWITCH DELAY FOR DECODER TO SWITCH ON
LACK 13
SAACL STATE SET STATE TO 13 STATE=0 TO 15
LACK 7
SAACL FRAME INITIALIZE DECODER STATUS

INIOUT RET NEXT ENTRY AT RECEIVE MODE

MULTIPLEXER IS IN ALINEMENT; RECEIVE MODE

RCMODE NOP

CHECK IF DECODER IS ON (SWITCH=0); IF NOT DECREMENT SWITCH
ZAC SWITCH
BZ FAWSUB
SUB ONE
SAACL SWITCH

SELECT FAW or ONE OF THE SUBBANDS FROM WHICH DATA HAS TO BE SENT
FAWSUB LACK 4
SUB BAND
BZ FWSB62
SUB ONE
BZ NEXT3 SUBBAND 3 (1000-1500 Hz)
SUB ONE
BZ NEXT2 SUBBAND 2 (500-1000 Hz)
SUB ONE
BZ NEXT1 SUBBAND 1 (0-500 Hz)

FAW

RECEIVE FAW BITS

NEW FAW ZALS DEMO NEW ACCESS CHECK
BZ FSIG

INITIALIZATION FOR A NEW FAW ACCESS
ZAC
SAACL DEMO NO NEW ACCESS ANYMORE
SAACL DATA CLEAR RECEIVE BUFFER
LACK 7
SAACL COUNT 8 FAW BITS TO RECEIVE
RECEIVE A FAW BIT AND INSERT IT INTO LSB OF DATA
FSI6 RECV
  LAR 0,COUNT
  BANZ CONT

ALL FAW BITS RECEIVED; CHECK FAW
  B FAWCHK

NOT ALL FAW BITS RECEIVED, PREPARE NEXT BIT
CONT SAR 0,COUNT
  RET

------------------------------
RECEIVE SUBBAND BITS
------------------------------

A BIT OF SUBBAND 1 (0-500 Hz) HAS TO BE RECEIVED
------------------------------
NEXT1 ZALS DEMO NEW ACCESS CHECK
  BZ LOOP1

INITIALIZATION FOR A NEW SUBBAND 1 ACCESS
  ZAC
  SA CL DEM O NO NEW ACCESS ANYMORE
  LACK 2
  XOR STRAT UNVOICED?
  BANZ VOINI

FURTHER INITIALIZATION FOR UNVOICED STRATEGY
  ZAC
  SA CL COUNT 1 DATABIT TO RECEIVE
  B SIGN1

FURTHER INITIALIZATION FOR VOICED AND INTERMEDIATE STRATEGY
  VOINI
  LACK 2
  SA CL COUNT 3 DATABITS TO RECEIVE

RECEIVE THE SIGNBIT OF SUBBAND 1 AND EXTEND THE SIGN IN RECEIVE BUFFER
  SIGN1 RECSB
  RET

RECEIVE A SUBBAND 1 DATABIT AND INSERT IT INTO LSB OF DATA
LOOPI RECV
  LAR 0,COUNT
  BANZ CONT1

ALL SUBBAND 1 BITS RECEIVED; UPDATE COD1 AND PREPARE NEXT SUBBAND
  LACK COD1
  TBLW DATA
  LACK \FF
  SA CL DEM O NEW ACCESS MARK
  LACK 4
  SA CL BAND NEXT SUBBAND TO RECEIVE (1500-2000 Hz)
  RET
NOT ALL SUBBAND 1 BITS RECEIVED, PREPARE NEXT BIT
CONT1 SAR 0, COUNT
RET

A BIT OF SUBBAND 2 (500-1000 Hz) HAS TO BE RECEIVED

NEXT2 ZALS DEMO
NEW ACCESS CHECK
BZ LOOP2

INITIALIZATION FOR A NEW SUBBAND 2 ACCESS
ZAC
SACL DEMO
NO NEW ACCESS ANYMORE
LAC STRAT
VOICED?
BNZ INUN2

FURTHER INITIALIZATION FOR VOICED STRATEGY
LACK 2
SACL COUNT
3 DATA BITS TO RECEIVE
B SIGN2

FURTHER INITIALIZATION FOR INTERMEDIATE AND UNVOICED STRATEGY
INUN2 LACK 1
SACL COUNT
2 DATA BITS TO RECEIVE

RECEIVE THE SIGNBIT OF SUBBAND 2 AND EXTEND THE SIGN IN RECEIVE BUFFER
SIGN2 REC SB
RET

RECEIVE A SUBBAND 2 DATA BIT AND INSERT IT INTO LSB OF DATA
LOOP2 RECEIVE
LAR 0, COUNT
BNZ CONT2

ALL SUBBAND 2 BITS RECEIVED; UPDATE C002 AND PREPARE NEXT SUBBAND
LACK C002
TBLW DATA
LACK >FF
SACL DEMO
NEW ACCESS MARK
LACK 3
SACL BAND
NEXT SUBBAND TO RECEIVE (1000-1500 Hz)
RET

NOT ALL SUBBAND 2 BITS RECEIVED, PREPARE NEXT BIT
CONT2 SAR 0, COUNT
RET

A BIT OF SUBBAND 3 (1000-1500 Hz) HAS TO BE RECEIVED

NEXT3 ZALS DEMO
NEW ACCESS CHECK
BZ LOOP3
**INITIALIZE FOR A NEW SUBBAND 3 ACCESS**

[ZAC]

**SACL DEMO** NO NEW ACCESS ANYMORE

**LACK** 1

**IOR STRAT** INTERMEDIATE?

**BNZ** VOIN3

**FURTHER INITIALIZE FOR INTERMEDIATE STRATEGY**

[ZAC]

**SACL COUNT** 1 DATABIT TO RECEIVE

**B** SIGN3

**FURTHER INITIALIZE FOR VOICED AND UNVOICED STRATEGY**

**VOUN3** LACK 1

**SACL COUNT** 2 DATABITS TO RECEIVE

* RECEIVE THE SIGNBIT OF SUBBAND 3 AND EXTEND THE SIGN IN RECEIVE BUFFER

**SIGN3** REC3B

**RET**

* RECEIVE A SUBBAND 3 DATABIT AND INSERT IT INTO LSB OF DATA

**LOOP3** RECEIVE

**LAR** 0,COUNT

**BNZ** CONT3

* ALL SUBBAND 3 BITS RECEIVED AND PERHAPS ALSO END OF 15 BIT DATABLOCK

* UPDATE CODE3 AND PREPARE NEXT THING TO RECEIVE

**LACK** CODE3

**TBLW** DATA

**LACK** 1FF

**SAACL DEMO** NEW ACCESS MARK

**LAC STRAT** VOICED?

**BNZ** NEXT6F END OF 15 BIT DATABLOCK

**LACK** 6

**SAACL BAND** NEXT SUBBAND TO RECEIVE (2500-3000 Hz)

**RET**

* NOT ALL SUBBAND 3 BITS RECEIVED, PREPARE NEXT BIT

**CONT3** SAR 0,COUNT

**RET**

*---------------------------------------------------------------*

* A BIT OF SUBBAND 4 (1500-2000 Hz) HAS TO BE RECEIVED

*---------------------------------------------------------------*

**NEXT4** ZALS DEMO NEW ACCESS CHECK

**BNZ** LOOP4

* INITIALIZE FOR A NEW SUBBAND 4 ACCESS

[ZAC]

**SAACL DEMO** NO NEW ACCESS ANYMORE

**LACK** 2

**IOR STRAT** UNVOICED?

**BNZ** VOIN4
* FURTHER INITIALIZATION FOR UNVOICED STRATEGY
  LACK 1
  SA CL COUNT 2 DATABITS TO RECEIVE
  B  SIGN 4
*
* FURTHER INITIALIZATION FOR VOICED AND INTERMEDIATE STRATEGY
VOID4 ZAC
  SA CL COUNT 1 DATABIT TO RECEIVE
*
* RECEIVE THE SIGNBIT OF SUBBAND 4 AND EXTEND THE SIGN IN RECEIVE BUFFER
SIGN4 RECSB
  RET
*
* RECEIVE A SUBBAND 4 DATABIT AND INSERT IT INTO LSB OF DATA
LOOP4 RECEIVE
  LAR 0,COUNT
  BNZ CONT4
*
* ALL SUBBAND 4 BITS RECEIVED; UPDATE COD4 AND PREPARE NEXT SUBBAND
LACK COD4
  TBLW DATA
  LACK >FF
  SA CL DEMO NEW ACCESS MARK
  LACK 5
  SA CL BAND NEXT SUBBAND TO RECEIVE (2000-2500 Hz)
  RET
*
* NOT ALL SUBBAND 4 BITS RECEIVED, PREPARE NEXT BIT
CONT4 SAR 0,COUNT
  RET
*
* ........................................................................
* A BIT OF SUBBAND 5 (2000-2500 Hz) HAS TO BE RECEIVED
* ........................................................................
* NEXT5 ZALS DEMO NEW ACCESS CHECK
  BZ LOOPS
*
* INITIALIZATION FOR A NEW SUBBAND 5 ACCESS
ZAC
  SA CL DEMO NO NEW ACCESS ANYMORE
*
* RECEIVE THE SIGNBIT OF SUBBAND 5 AND EXTEND THE SIGN IN RECEIVE BUFFER
SIGN5 RECSB
  RET
*
* RECEIVE THE SUBBAND 5 DATABIT AND INSERT IT INTO LSB OF DATA
LOOP5 RECEIVE
*
* ALL SUBBAND 5 BITS RECEIVED; UPDATE COD5 AND PREPARE NEXT SUBBAND
LACK COD5
  TBLW DATA
  LACK >FF
  SA CL DEMO NEW ACCESS MARK
  LACK 2
  SA CL BAND NEXT SUBBAND TO RECEIVE (500-1000 Hz)
  RET
A 6:1 of subband 6 (2500-3000 Hz) has to be received only accessed by unvoiced and intermediate strategy

NEXT6 ZALS DEMO NEW ACCESS CHECK
  B2 LOOP6
  INITIALIZATION FOR A NEW SUBBAND & ACCESS
  ZAC
  SACL DEMO NO NEW ACCESS ANYMORE
  RECEIVE THE SIGNBIT OF SUBBAND 6 AND EXTEND THE SIGN IN RECEIVE BUFFER
  SIGN6 RECEB8 RET
  RECEIVE THE SUBBAND 6 DATABIT AND INSERT IT INTO LSB OF DATA
  LOOP6 RECEIVE
  ALL SUBBAND 6 BITS RECEIVED AND ALSO END OF 15 BIT DATABLOCK
  UPDATE CODE6 AND CHECK FOR END OF FRAME
  LACK CODE6
  TBLW DATA
  LACK >FF
  SACL DEMO NEW ACCESS MARK
  NEXT6F LAC FRAME
  BN2 MENDFM
  END OF FRAME, PREPARE NEXT FRAME
  ZAC
  SACL BAND NEXT TO RECEIVE=FAM
  RET
  NOT END OF FRAME, PREPARE NEXT DATABLOCK
  MENDFM LACK I
  SACL BAND NEXT SUBBAND TO RECEIVE (0-500 Hz)
  RET
  DEMULTIPLEXING DONE
==============================================================================