Design and Test of a Real-Time Voice Communication System for Power Line Communication Channels

by

Rick G. Dorbolo
B. Sc. EE., The University of Manitoba, 1991

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
January 1994
© Rick G. Dorbolo, 1994
In presenting this thesis in partial fulfilment of the requirements for an advanced
degree at the University of British Columbia, I agree that the Library shall make it
freely available for reference and study. I further agree that permission for extensive
copying of this thesis for scholarly purposes may be granted by the head of my
department or by his or her representatives. It is understood that copying or
publication of this thesis for financial gain shall not be allowed without my written
permission.

Department of Electrical Engineering
The University of British Columbia
Vancouver, Canada

Date Feb 7, 1994
Abstract

Intrabuilding electric power distribution lines provide easy access and universal coverage for digital speech communications. However, power line communication channels exhibit unpredictable and variable levels of signal attenuation and noise. To enable effective communication requires appropriate forward error correction of the encoded speech.

This thesis describes the design, implementation and testing of synchronization, voice compression and forward error correction algorithms required to achieve a robust real-time communication link across power line channels. Communication was connection oriented and synchronous with a selectable baud rate of 9600 bps or 19200 bps. Synchronization (delineation) was accomplished using a seven bit synchronization marker placed between frames of encoded speech. The encoded speech data was generated using the U.S. digital cellular standard VSELP 7950 bps voice encoder. Error detection and correction bits were added to the speech data to produce aggregate data rates of approximately 9600 and 19200 bps. Error detection of the perceptually significant bits was provided through the use of a seven bit CRC in the 9600 bps encoded speech. The 19200 bps scheme employed an error detection/correction scheme where the amount of protection was based on the perceptual significance of each parameter.

The synchronization algorithm was implemented in hardware and tested. Synchronization was maintained at bit-error rates (BERs) greater than 0.1. Subjective testing of the synchronization algorithms under random noise indicated that the 9600 and 19200 bps coding schemes were usable with channel BERs up to approximately 7x10^{-3} and 2x10^{-2}, respectively. Tests performed on actual power line communication channels indicated approximately equal performance using either the 9600 or 19200 bps coding scheme, due to the comparatively higher BER of the 19200 bps channel.
# Table of Contents

Abstract ................................................................................................................................. ii  

List of Figures ....................................................................................................................... vii  

Acknowledgments .................................................................................................................. x  

Chapter 1 Introduction .......................................................................................................... 1  
Section 1 Power Lines and Power Line Modems .................................................................. 1  
  Topic 1 Power Line Topology ............................................................................................... 1  
  Topic 2 Power Line Modems ............................................................................................... 2  
  Topic 3 Problems Encountered with PLMs ....................................................................... 4  
Section 2 Voice Compression Technology ........................................................................... 5  
  Topic 1 Requirements of Compressed Speech .................................................................... 5  
  Topic 2 Synchronous Communication Environment .......................................................... 6  
Section 3 Outline of Thesis .................................................................................................. 7  

Chapter 2 Voice Processing ................................................................................................ 9  
Section 1 System Overview ................................................................................................ 9  
Section 2 VSELP Compression Algorithm .......................................................................... 9  
Section 3 Forward Error Correction with VSELP ............................................................... 11  
Section 4 Channel-Induced Degradations .......................................................................... 14  
  Topic 1 Bit Errors ................................................................................................................ 14  
  Topic 2 Processing Delay ................................................................................................... 15
# List of Figures

| Figure 1 | Typical domestic power supply (A). Typical commercial/industrial power supply (B) | 3 |
| Figure 2 | One Way Voice Processing System | 9 |
| Figure 3 | Block diagram of VSELP speech decoder. | 11 |
| Figure 4 | The R0 and LPC1 parameter versus time for the utterance “SPEECH TEST” | 13 |
| Figure 5 | The LAG parameter and frame energy (R0) parameter versus time for the utterance “SPEECH” | 14 |
| Figure 6 | End to end processing delay components of VSELP based speech encoder | 16 |
| Figure 7 | Power line channel contains periodic error bursts spaced 1/120 second apart in addition to random errors. | 19 |
| Figure 8 | Placement of synchronization markers in the speech data stream. | 22 |
| Figure 9 | Synchronization algorithm | 24 |
| Figure 10 | Simulated synchronization time in bits for frame lengths of 168, 336, 672, and 1344 bits. | 26 |
| Figure 11 | Autocorrelation of the chosen marker. | 28 |
| Figure 12 | Number of times frame synchronization was lost versus channel BER and maximum walk distance during a 90 second test interval for a channel “bad” cross-phase (X-phase) and of the same phase. | 30 |
Figure 13  Number of times frame synchronization was lost versus channel BER and maximum walk distance, during a 90 second test interval for a channel “good” cross phase (X-phase). .................................. 31

Figure 14  Maximum number of frames lost versus channel BER and maximum walk distance, during a 90 second test interval. ................................. 31

Figure 15  Differential encoder (a), differential decoder (b). .......................... 33

Figure 16  Number of times carrier synchronization was lost over a 90-second interval. .............................................................. 34

Figure 17  Original bit sync circuit (a), bit sync circuit with random walk filter (b). 35

Figure 18  State transition diagram for a length N random walk filter with inputs Up/Down and outputs UP'/DOWN'. ....................................... 36

Figure 19  Construction of test pattern .................................................. 39

Figure 20  Processing steps required for subjective testing. ......................... 45

Figure 21  MOS score versus symbol error rate for the GSP0 and LAG parameters with and without error detection. ......................... 48

Figure 22  MOS score versus symbol error rate for the Frame energy (R0) and 1st LPC coefficient with and without error detection. .............. 48

Figure 23  MOS score versus symbol error rate for the 2nd and 4 LPC coefficient with and without error detection. .................................. 49

Figure 24  MOS score versus symbol error rate for the residual vectors CODE1 and CODE2, 6th LPC coefficient, and with error detection the for residual vectors and the 8th LPC coefficient. .......................... 49
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Symbol and CRC placement of class 1 symbols.</td>
</tr>
<tr>
<td>26</td>
<td>Mean degradation curves before and after error correcting bits are assigned.</td>
</tr>
<tr>
<td>27</td>
<td>Periodic burst error spacing for (a) same phase transmission (b) cross phase transmission.</td>
</tr>
<tr>
<td>28</td>
<td>Double errors as a function of average bit error rate and symbol spacing.</td>
</tr>
<tr>
<td>29</td>
<td>A comparison of the number of symbols with errors with and without interleaving as a function of bit error rate.</td>
</tr>
<tr>
<td>30</td>
<td>A comparison of the half rate and full rate encoding schemes for a random binary symmetric channel.</td>
</tr>
<tr>
<td>31</td>
<td>A comparison of the full rate encoding schemes with a random binary symmetric channel and various power line channels.</td>
</tr>
<tr>
<td>32</td>
<td>A comparison of the half rate (9600 bps) and full rate (19200 bps) encoding schemes over four power line communication channels.</td>
</tr>
</tbody>
</table>
Acknowledgments

I would like to acknowledge Dr. Robert Donaldson for his supervision during this thesis and for recommending this exciting and challenging thesis topic. His guidance during the development of this thesis has proved invaluable.

I also wish to thank Barry Buternowsky, Haleh Hobooti, and Raymond Woo for their assistance. Barry’s technical assistance with the power line modems was a great help. Haleh generously donated her time to proof reading this thesis. Raymond Woo assistance and cooperation allowed me to complete my subjective tests without difficulties. In addition I would like to thank the many students who provided me with valuable advice and suggestion.

Finally, I wish to thank all the people who volunteered to participate in my subjective tests. Their assistance was essential for this thesis.
Chapter 1 – Introduction

In-building electrical power distribution circuits are ubiquitous in coverage and easily accessed using a standard 120 volts wall plug. Distribution circuits provide a potential communications medium [1, 2, 3]. Power lines were not designed for communication, however, and are subject to numerous impairments such as noise, frequency and time fading, phase inversion, bursty errors, and intersymbol interference. Accurate models for power line communication channels are unavailable. Modems designed to use power line channels for data transmission can be complex [2, 4, 5]. Transmission of speech over a power line is potentially more difficult than data transmission. However there are applications where speech transmission is desirable. One compelling application is an intercom or a building public address system. Two enabling technologies that are becoming available are the single chip digital power line modems and low rate digital speech encoders. A combination of these technologies may allow for cost effective implementations of digital speech over power lines.

This thesis describes the transmission and real-time transmission of compressed speech using a 9600/19200 bps power line modem. This introductory chapter provides an overview of the problems faced, background, and an outline of the thesis.

1.1 Power Lines and Power line Modems

Power line Topology

Figure 1a illustrates a typical residential power service. Each house is supplied from a distribution transformer with two 120 volt lines (with a 180° phase difference) and a neutral. This power supply is delivered, via branch circuits, from a circuit panel to all locations within the residence [1, 6]. The
branch circuits can be configured to supply a line-to-neutral source of 120 volts or else a line-to-line source of 240 volts.

A typical industrial/commercial supply (figure 1b) is three phase. As in the residential case, the loads are supplied by branch circuits and are placed between the supply line and neutral. Although both residential and commercial/industrial power supplies have been studied in the past, this thesis will only be concerned with the problems encountered for an industrial/commercial power distribution; commercial/industrial power line channels are much more hostile than residential power channels.

Two issues concerning power line topology are relevant to this thesis. First, power line modems are connected onto the power line (plugged in) in the same manner as loads. Second, the physical distance between modems is not the only determinant of communication quality of connection. This fact results from the building wiring method. Transmission along the same branch circuit will provide the best connectivity. Transmission across different branch circuits on the same phase often requires the signals to travel across the central circuit panel. Thus, transmissions between modems located far from the circuit panel and across different branch circuits are normally subject to much greater attenuation than transmissions on the same branch circuit. Finally, transmissions across power phases are subject to even greater attenuation. The modem signal, in this case, must propagate across phases through capacitive coupling between the power lines, by propagation across windings in the distribution transformer, or through loads placed between power phases. Thus, signals between two modems placed in adjacent rooms may be plugged into different phases and as a result, may experience unanticipated levels of noise or attenuation.

**Power line Modems**

Power line modems (PLMs) operate principally in the same way as telecommunication and radio
modems [7]. The salient issues with regard to power line modems design include high signal attenuation, high noise, and a usable bandwidth between 50kHz and 300kHz [1, 8].

Considerable research involving power line communication has been conducted at the University of British Columbia. The culmination these efforts has been two modem designs. The first is a single chip implementation of an FSK modem [9] with a baud rate of 1.2 — 19.6 kbps. The second is versatile LSI implementation of a PSK modem with an adjustable baud rate of 1.2 — 19.6 kbps [4]. The PSK modem was used in this thesis to perform the speech—over—powerlines implementation. In addition to these two modems, further work is presently underway to produce a single chip, all-digital PSK modem.

Figure 1 Typical domestic power supply (A). Typical commercial/industrial power supply (B).
Recently, two commercial products have been released. Echelon Incorporated has developed a power line modem module as part of an in-building control system. Adaptive Networks [10] has also developed a line of single chip power line modems. These modems claim speeds of 100kbs and 1000kbs. Further, they boast throughputs of 10kbps and 100kbps with a BER of $10^{-9}$ using proprietary spread spectrum and error correction techniques. Complete details on these modems are unfortunately unavailable due to their proprietary nature.

**Problems Encountered with PLMs**

Power line communication channel response varies with the distance between the transmitter and receiver as well as with the number and type of power line loads. As a result, power line channels exhibit highly variable frequency versus attenuation characteristics and noise power density profiles. Further, these characteristics may change over time due to load changes.

The channels resulting from the above conditions have highly variable noise and attenuation and have no reliable model. Without a model, accurate analysis of different transmission schemes becomes difficult or impossible. In the past [4, 11], analysis was performed assuming binary symmetric transmission over a gaussian noise channel. All such analysis can at best result in a gross approximation of the actual performance. Later in this thesis, a power line channel model is described which includes noise produced by a periodic noise source and a binary symmetric noise source.

Data transmissions across power lines are subject to noise caused by the variety of electrical loads. The worst offenders are those which regularly and rapidly switch on and off line. Examples include universal motors, dimmer switches, and computer power supplies [1, 3, 12]. In addition to producing noise, the loads may also greatly attenuate signal transmissions. This combination of high attenuation
along with high noise will produce a channel with low signal to noise ratio (SNR) or a channel which can change from high SNR to low SNR as loads change.

In addition to the possibly high bit error rates (BER), power line channel errors are bursty and exhibit periodicity [1, 3, 5, 13]. These bursty, periodic errors must be considered in designing any forward error correction (FEC) scheme which, unfortunately, adds complexity to the communication system.

1.2 Voice Compression Technology

The power line modem chosen had a selectable bit rate of 1.2kbps–19.6kbps. Standard telephone grade digitalized speech requires a data rate of 64kbps or 32kbps. Thus, some form of low rate speech encoding was required to perform real-time transmission of speech across the power lines. This section outlines some currently available speech compression technologies and the requirements for use with the power line modem selected.

Requirements of Compressed Speech

The first requirement of digitized speech is a data rate below 19200bps. Many methods exist which provide voice compression below 32 kbps. Speech coding between 14 kbps and 32 kbps provides good to excellent quality speech suitable for toll-grade telephony. Coding rates between 6kbps and 13kbps provide fair to good speech quality suitable for mobile radio or voice mail applications. Rates below 6kbps provide poor to fair quality coding suitable for secure voice applications. The ratings are given in [14] where the specific algorithms are explained in detail. Improvements in the speech coding algorithms, along with an increase in digital signal processor (DSP) performance, have allowed for a continual improvement in speech qualities at various bit rates.
The second requirement of the speech compression algorithm is tolerance to channel errors. As stated earlier, power line communication is subject to noise, primarily from electrical loads. Voice coding techniques generally are able to tolerate BERs on the order of $10^{-4}$. Higher error rates can cause rapid degradation of sound quality. Two schemes are available to maintain speech quality in the presence of channel errors. The first approach uses traditional error control coding techniques such as block codes or convolutional codes to reduce the effective BER [15]. Of course, adding error control bits reduces the net channel rate which implies that a lower bit rate voice code must be used and hence a lower error free sound quality. A second approach to reduce the significance of channel errors is to construct voice coding algorithms to reduce the perceptual significance of errors [16]. Later in this thesis, a voice coding technique is described which provides graceful degradation of voice quality with a BER from $10^{-3}$ to $10^{-1}$.

The third requirement of the coding technique is a real-time implementation requiring two conditions. First, the coding algorithm must be simple enough to be executed in real-time. Such is the case if the voice coder can process a section of speech in the same or less time than the transmission time of the speech sample. Second, the end-to-end processing delay must be sufficiently small to allow for normal conversation. The end-to-end processing delay consists of the algorithmic processing delays, coding delays and any communication delays. For a real-time implementation the end-to-end processing delays should be less than 200ms.

**Synchronous Communication Environment**

In the design of a power line speech communication system, it is necessary to choose between a packet oriented or synchronous connection oriented data transmission scheme. Prior to this thesis, much research was directed towards packetized, power line computer data transmission [3, 4, 11]. Computer data requires a very small BER ($<10^{-7}$) and no tolerance to lost bits. An ARQ scheme which would
achieve such low BERs over power lines has been implemented [17]. Unfortunately, the scheme requires large overheads, which results in throughputs lower than that of a synchronous system during the periods of low channel noise and even lower during high noise conditions. Compressed voice data does not require the very low BERs provided by a packetized ARQ scheme. The voice coding technique employed in this thesis worked satisfactorily with a BER to $10^{-2}$ and with some lost bits. Given the low channel rate of 19.2 kbps, efficiency of the connection scheme was of great concern. It was clear that a synchronous system would be required.

Given the requirement for synchronous communication, much of this thesis involved the design and implementation of a synchronous system. The synchronization schemes constructed will be shown to function effectively at BERs greater than $10^{-1}$.

### 1.3 Outline of Thesis

The body of the thesis is broken into four sections: voice processing, frame synchronization, quantitative and qualitative measurements, and forward error correction. Chapter 2 describes the VSELP voice processing algorithm and introduces the concepts of speech frames and speech symbol parameters. Chapter 2 also describes the impairing effects of channel disturbances and the need for forward error correction.

Chapter 3 describes modem synchronization. First the requirements of this communication system are presented. This is followed by a detailed description of the algorithm implemented as well as the measured performance. The chapter ends with hardware modifications required to enhance the synchronization performance for use with voice communication.
The bulk of the experimental procedures and results are provided in chapter 4. The chapter begins with a description of the quantitative channel parameters. The scheme used to generate pseudo-random test patterns is presented. This is followed by the channel BER measurements, as well as the measurements used to quantify the performance enhancements of the hardware modifications. Mean opinion score (MOS) testing methodology is presented next. MOS tests used to assess speech parameter sensitivity are presented followed by a summary of the test results. Finally, the MOS test used to measure the effectiveness of the FEC scheme is presented.

Chapter 5 begins with some FEC background followed by a description of the two FEC schemes used in this thesis. The reasoning behind the parameter interleaving and the choice of parameter distance is also presented. The chapter ends with a subjective sound quality comparison of the 9600 bps and the 19200 bps encoding schemes.

Chapter 6 provides a brief summary and the conclusions of the thesis along with recommendations for future work.
Chapter 2 – Voice Processing

2.1 System Overview

This chapter describes the VSELP compression algorithm and explains the motivation for the forward error correction algorithm used.

The test setup, shown in figure 2, was used to test one way (half duplex) transmission of speech. A voice signal was sampled at 8000 samples/second to 12 bits using an IBM pc and A/D card. The 12 bit sampled speech was next padded to 16 bits, source encoded using the VSELP algorithm and channel encoded using forward error correction (FEC) on a SUN SPARC workstation. The voice data was then transmitted over the power line channel. Finally, the received speech data was similarly decoded, decompressed and stored or played back.

![Figure 2 One Way Voice Processing System](image)

2.2 VSELP Compression Algorithm

The VSELP algorithm [18, 19] performs the 8:1 voice compression to produce output at 7950 bps, as required by the power line channel. This algorithm was chosen as the target algorithm in this thesis for several reasons:

1. Algorithm complexity is moderate.
2. The VSELP algorithm is standardized and open, and has been chosen as the North American cellular phone standard [20, 21].

3. The algorithm is tolerant to channel errors.

4. VSELP CODEC chips are commercially available[22].

Other CELP algorithms [23, 24] were considered, but they were deemed less desirable because of their proprietary status and restricted availability.

The VSELP compression scheme is a modification of the CELP [25, 26] voice compression algorithms. CELP (Code Excited Linear Predictive coding) algorithms begin by processing pulse code modulated (PCM) [27, p. 22] speech samples using a linear predictive (LPC) filter [28]. The LPC filter removes most of the harmonic redundancy from the speech. Next, the residual from the LPC filter is vector quantized [27, 29, pp. 273–7]. The quantized residual vector and the LPC filter parameters form the compressed speech code to be transmitted (or stored). The received speech is decoded by first expanding the received quantized residual vector using a codebook, by creating an inverse LPC filter from the received LPC filter parameters, and finally by passing the expanded vector through the inverse filter. In the absence of channel errors a close approximation of the original speech results.

Figure 3 is a block diagram of the VSELP speech decoder. The VSELP algorithm differs from the generic CELP algorithm in that it uses two fixed codebooks, one adaptive codebook, and three gain parameters to quantize the residual. This codebook structure speeds the vector quantization process by using three small codebooks rather than one large codebook. Further, the codebooks are constructed to reduce the degradation caused by bit errors.

The VSELP algorithm processes speech on a frame-by-frame basis. The frame consists of fifty 8kHz samples, and represents 20ms of speech. For each frame, the frame parameters, including LPC parameters
and the frame energy are calculated. The frames are further divided into four subframes. Residual vectors are calculated for each subframe. The residual vectors (CODE_1 and CODE_2), adaptive codebook vector (LAG), and gain vectors (GSP0) are known as subframe parameters. Table 1 lists the number of bits to encode each parameter. In total, the frame parameters require 43 bits and the subframe parameters account for 116 bits of the data stream. A detailed description of the parameters in table 1 is given in the digital cellular phone standard [18, 30]

The performance loss due to bit errors varies according to the parameter corrupted. For example, the residual codewords were designed to have redundancy. An error in a residual encoder parameter will have a smaller impact than an error in an LPC parameter. When designing FEC schemes, the sensitivity of each parameter to bit errors must be considered. Measurements to determine the sensitivity of each parameter are discussed in chapter 4.

![Figure 3 Block diagram of VSELP speech decoder.](image)

### 2.3 Forward Error Correction with VSELP

VSELP encoded speech is noticeably degraded by bit errors. A bit error rate (BER) of $10^{-4}$ produces a noticeable degradation in sound quality. BERs higher than $10^{-3}$ produce frequent unpleasant clicks,
<table>
<thead>
<tr>
<th>Frame Parameters</th>
<th>Subframe Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Codeword bits</td>
</tr>
<tr>
<td>RO</td>
<td>5</td>
</tr>
<tr>
<td>LPC1</td>
<td>6</td>
</tr>
<tr>
<td>LPC2</td>
<td>5</td>
</tr>
<tr>
<td>LPC3</td>
<td>5</td>
</tr>
<tr>
<td>LPC4</td>
<td>4</td>
</tr>
<tr>
<td>LPC5</td>
<td>4</td>
</tr>
<tr>
<td>LPC6</td>
<td>3</td>
</tr>
<tr>
<td>LPC7</td>
<td>3</td>
</tr>
<tr>
<td>LPC8</td>
<td>3</td>
</tr>
<tr>
<td>LPC9</td>
<td>3</td>
</tr>
<tr>
<td>LPC10</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Frame and Subframe parameters for VSELP CODEC.

pops and unnatural sounding tones. FEC is therefore essential to maintain high quality communication in a high BER power line environment.

Three categories of techniques are available to improve the sound quality of VSELP coded speech. The first is to construct the VSELP code to be tolerant to bit errors. This can be accomplished, for example, by quantizing vectors with grey codes or by requantizing other parameters to a log scale from a linear scale. The second method uses traditional forward error correction (FEC) methods such as trellis encoding, and Viterbi decoding and interleaving to decrease the decoded BER. A third method makes use of parity or CRC bits to detect errors in the frame parameters. When a code-word error is detected,
the code-word from the previous frame or subframe can be substituted for the damaged code-word. This is possible because codewords often remain stationary from frame to frame.

Figures 4 and 5 show how the speech parameters vary over time. Figure 4 gives the frame energy (R0) and the first LPC coefficient (LPC1) for the utterance “SPEECH TEST”. The other reflection coefficients behave similarly to R0. A significant interframe autocorrelation of R0 and the LPC1 parameter are observed. Figure 5 shows how the LAG parameter varies between subframes. The lag parameter value appears more random between subframes than does R0, but does show correlation between subframes, especially at higher frame energies.

The specific FEC encoding methods examined will be discussed later in this thesis. The goal is to find a combination of FEC and error detection techniques to minimize the effect of bit errors.

![Figure 4](image)

Figure 4 The R0 and LPC1 parameter versus time for the utterance "SPEECH TEST"
2.4 Channel-Induced Degradations

Transmission over power line channels causes bit errors which degrade the reconstructed speech sound quality. Random and periodic burst errors occur and some speech frames are never received. This section briefly discusses the effect of these errors on the quality of the decoded speech.

Bit Errors

Two categories of bit errors are observed in power line communications. Random (impulse) errors are caused from electrical loads such as electric heaters or lamps switching on or off. Such switching occurs infrequently, and associated errors tend to be infrequent as well. Periodic errors are caused by devices which switch on and off rapidly. Drill motors, computer power supplies, and dimmer switches are examples of such devices, with the latter two producing noise at 120Hz. Both types of noise cause errors with burst length ranging from several bits to several dozen bits (approximately 0.5ms to 2ms).
The effects of channels errors as well as methods to minimize their impact on transmitted speech will be discussed in later chapters.

**Processing Delay**

Time is required to perform speech compression, channel coding, transmission, reception, decoding and decompression. End-to-end delay is defined as the time from the first sample of speech input at the microphone to when that sample is reconstructed at the output speaker (this definition differs somewhat from one-way coding delay which does not include the transmission time [31]). Delay in excess of 1 ms. is perceptible, but end-to-end delay greater than 200ms is considered unacceptable [32, 33].

As shown in figure 6, the three components of end-to-end delay include frame-formation (algorithmic delay), channel delay, and coding delay.

1. **Frame-formation delay** is a fixed 20ms delay equal to the time to accumulate enough samples to constitute a speech frame. The VSELP coding algorithm requires all 160 speech samples to be present before the LPC parameters can be computed. Thus, a minimum 20ms delay is introduced.

2. **Channel delay** is the time required to transmit (physically) a speech frame from transmitter to receiver. The channel delay can vary between zero and 20ms, depending on the channel rate. Because the entire channel capacity is used to transmit the voice data, the channel delay is 20ms.

3. **Encoding and decoding delays** represent the amount of time for the VSELP and channel coding algorithms to process speech. These delays depend on the processing hardware speed and the software code execution time. In a single-processor real-time implementation, frame encoding delay and decoding each must be less than 20ms (the duration of a frame).

End-to-end delay which is the sum of the above delays, will vary between 40 and 80ms depending on the particular implementation.
Lost Frames

In addition to the bit errors described above, massive errors may occur on power line channels. A massive error results in a received speech frame which contains mostly incorrect data (BER ≈ 0.5). Massive errors have two causes. The first is loss of or incorrect initial frame synchronization. A failure in bit timing recovery causes bit slip, which causes speech frame data to be shifted by at least one bit. In such a case, every speech parameter at the receiver will be decoded incorrectly. At those times when the modem synchronization algorithm attempts initially to synchronize or re-synchronize the modems, mis-synchronization may also occur if the frame synchronization algorithm accidentally locks onto a marker sequence in the data stream\(^1\). A mis-synchronized frame, when decoded, will have every parameter incorrect. A frame with all the coded speech parameters incorrect produces an unpleasant sounding click or chirp.

The second cause of massive errors is a long error burst or a channel fade. Power line communication on marginal channels is marked with many such random fades. These fades result from equipment being

---

\(^1\) Bit stuffing is used in data communication to prevent occurrences of the marker sequences from being repeated in the data sequence. This technique could not be used here because bit stuffing changes the length between markers. Errors in the stuff bits or in the bits preceding the stuffed bits will result in mis-decoded frames and frame lengths. Mis-decoded frame lengths usually cause loss of frame synchronization. In addition, because stuff bits are added after FEC they cannot be protected.
switched on line or from high load events. The result is a large number of incorrectly received bits. As before, such a long error burst will produce unpleasant clicks and pops in the received frame.

The FEC coding scheme, described in chapter 5, includes provisions to detect and mask frames which contain massive errors. When a frame with massive errors is detected, the parameters from the previous frame are used. If a succession of massive error frames are detected the output is gradually muted as silence is deemed preferable to noise.
Chapter 3 – Frame Synchronization

For data to be transmitted synchronously, the receiver and transmitter must be synchronized (de-linedated). Frame synchronization is required by the receiver to determine the start of each transmitted speech frame, to enable successful decoding of the compressed speech data.

This chapter describes the frame synchronization algorithm developed for this speech-over-powerlines communication system. The requirements of the synchronization routine are presented, the algorithm is explained, and the measured performance of the algorithm is presented. The chapter ends with a section describing the enhancements performed on the hardware of an existing data modem.

3.1 Synchronous DLL Design Requirements

This section details the synchronization requirements of this voice-over-powerlines communication system.

Low Overhead

The selected voice coding algorithm required a transmission rate of 7950 bps. The modems had selectable bit rates of 19200 and 9600 bps. Transmission of voice data at the 9600 bps rate leaves 9600 – 7950 = 1650 bps either for synchronization, or FEC or both. This amounts to 1650/50 = 33 bits/frame. A minimum amount of FEC requires an overhead of 8 bits/frame. There remains 25 bits/frame for synchronization (a maximum of 13% of the bandwidth.) We are interested in a synchronization scheme with minimum overhead to maximize the remaining bits available for FEC.
Channel Random and Burst Errors

Channel errors impair modem synchronization by increasing the time required to synchronize the modems, and by increasing the likelihood that modems will lose synchronization. It was desired to have the modems able to synchronize with a BER of up to $10^{-1}$. This value was chosen because a BER of $10^{-1}$ was felt to be an upper limit on the coded speech tolerance to channel errors.

The power line channel was found to have bursty periodic errors along with random errors [5, 13]. The periodic errors occur every 1/120th of a second [1]. At 19200 baud, these burst errors may affect several to dozens of bits per burst depending on the power line conditions. Periodic errors are a result of the 60 Hz power line cycle. Transmission over the same phase is characterized by a burst error occurring at 120 Hz, while transmission across phases results in two bursts (one from each phase) occurring every 1/120 of a second. This difference in error distribution will be discussed further in terms of parameter interleaving.

![Figure 7](image)

Figure 7 Power line channel contains periodic error bursts spaced 1/120 second apart in addition to random errors.

Burst errors are easier to overcome than random errors. The synchronization algorithm is optimized to function best with synchronization words (markers) which are likely to be either error free or completely destroyed (i.e. hit by burst errors). Conversely, if the periodic nature of errors is not taken into account, synchronization will not be possible.
Bit Slip Errors

Bit slip is the most difficult impairment to deal with in terms of bit synchronization. A failure in bit timing recovery results in a bit slip. When a bit slip occurs the net effect is an apparent change in the size of the received frame, and frame synchronization usually will be lost. The probability of bit slips depends on the ability of the bit timing recovery circuitry. The original power line modems were designed to transmit packeted data on channels with relatively low bit error rates (BERs <10⁻³). This allowed the modems to operate with the most basic of bit synchronization circuits. When attempts were made to implement synchronous communication bit slips was identified as a major problem. Accordingly, the modem synchronization routine was modified to correct for single bit slips. Later in this chapter, circuitry which reduces the number of bit synchronization losses is presented. Unfortunately, the number of bit slips for our modem/channel combination has not been characterized; although we know bit slips will occur, we do not know the probability.

Synchronization Time

When a modem begins transmitting, the receiving modem requires time to establish frame synchronization. The interval of time from when the first bit is transmitted to the time when the receiving modem has established a lock will be defined as the synchronization time. Obviously, we would like this interval to be as small as possible. In applications such as intercom or building public address, synchronization times less than 400ms are suitable. Other applications such as infant monitors, which are connected for long durations, would function acceptably with longer synchronization times.

Two factors affect the synchronization time. The first is the number of bits used for synchronization. The larger the number of bits dedicated to synchronization, the shorter will be the synchronization time.
The second factor is the channel bit error rate. As stated previously, the synchronization time increases as the channel BER increases.

3.2 Synchronization Algorithm Developed

This section provides a motivation and a detailed description of the synchronization algorithms used in this thesis.

Motivation

Frame synchronization through the use of a fixed binary pattern or "sync word" embedded between successive fixed length frames is the most commonly used method of synchronization and is the method used here. Once the position of the sync word is detected, the receiver may obtain the position of each bit relative to the sync word.

Frame synchronization technique has been very well studied. Optimal solutions have been developed[34, 35] for finding the location of markers in a block of data subject to gaussian noise. While these solutions are optimal for gaussian channels, they require a considerable amount of storage space and real-time processing. The requirements of this thesis dictate the use of a simpler scheme. Further, the power line channel is not a gaussian channel and the optimality of these schemes is not assured for other channels.

The scheme chosen to implement the synchronization routine is a simplification of those described elsewhere [36, 37]. These algorithms were developed for use in radio communications and were each specifically adapted for use with burst errors due to jamming interference, which is similar to the burst noise of the power line channel. Computationally, the algorithm presented requires only 2 bytes of storage when implemented as part of the modem microcontroller.
Description

The synchronization scheme requires the periodic placement of markers into the data stream, as shown figure 8. The markers are placed at the start of every nth speech frame. Typically n was varied between 1 and 3.

![Placement of synchronization markers in the speech data stream.](image)

The synchronization algorithm used by the modems is described in figure 9. The following pseudocode describes the synchronization algorithm for an M bit long marker.

1. Set the Confidence Counter (CC) to zero.
2. Shift a bit received from the power line into the input register.
3. Compare content of input register to the marker. If contents are not equal go to step 2.
4. Increment CC.
5. Get N frames of data from the power line.
6. Shift M bits into the input shift register.
7. Compare the shift register to the marker; if the marker has 0 or 1 errors increment CC and go to step 10.
8. Decrement CC.
9. If CC = 0 go to step 1.
10. If the value of CC is greater than the walk_distance parameter decrement CC.
The algorithm begins in the Start/Reset state (step 1.) Here, the receiving modem waits for the receipt of an error free marker (steps 2 and 3.) When a marker is found, the confidence counter is incremented; the modem transmits N frames to the host, and then loads in the next marker (steps 4–6.) The modem enters sync state 1 which corresponds to the value of the confidence counter. If this marker has more than one error the modem returns to the start/reset state. If the marker has one or zero errors sync state 2 is entered. As in state 1, the number of errors in the received marker determines the next state (steps 5–11.) The sync state may increase to an arbitrary number, defined as the “walk distance,” given that correct markers are received. As the walk distance is increased, the likelihood that the modem will mistakenly enter a lost sync state (due to random errors) decreases, but the time till the modem recognizes synchronization loss from a bit slip and attempts to resynchronize is increased.

Not included in the description above is a provision added which attempts to correct for single bit slips. The modem implementation of step 7 from the above listing also compares the marker to bit shifted markers. If a bit shifted marker is received, then a bit will be either added or removed from the next frame in an attempt to resynchronize the modems. Unfortunately, characterization of the frequency of bit slip occurrences is difficult. Consequently, the present criteria as to which marker patterns should be interpreted as a bit shift was chosen using a trial-and-error method as opposed to more formal analytical methods. Presently, the criteria used is the receipt of a single marker which is exactly the same as a shifted version of the marker. The probability of this condition being satisfied by a properly synchronized marker corrupted by random errors is given by (1) below where P_{fbsc} is the probability of a faulty bit slip correction, P_e is the probability of a bit error, and the binary sequence “01100101” is used as the marker.

\[ P_{fbsc} = 2P_e^5(1 - P_e)^2 \]  

(1)
This value is important because if $P_{\text{false}}$ is too high, synchronization will be lost more often from falsely detected bit slips than from actual bit slips.

**Theoretical Performance Under AWGN** Once the first marker has been found, the action of the
algorithm is identical to that of a random walk process[38, pp 212–3]. The state is increased (a step to the right) when a marker with one or zero errors is received, and the state is decreased (a step to the left) when a marker with two or more errors is received. This “walk” can increase to a maximum distance N, and remains at state N when one or zero errors are found in the received marker.

Given the above situation, the probability \( P_r \) that a step to the right will be taken is the probability of one or zero errors. For a binary symmetric channel,

\[
P_r = (1 - P_e)^8 + 8P_e(1 - P_e)^7
\]

where \( P_e \) is the probability of a random bit error. Provided \( P_r > 0.5 \), the value of N is made sufficiently large, and no bit slips occur, frame synchronization using the above algorithm will be maintained. As N is decreased, the likelihood that random errors will cause the process to “walk” leftward to the start/reset state (therefore causing loss of synchronization) increases.

One, therefore, is motivated to choose a large value of N to reduce the probability that a false loss of synchronization (described above) will occur. However, when a bit slip occurs the number of frames lost until synchronization reoccurs is approximately proportional to N, which suggests that N be a small value.

The synchronization algorithm described above was simulated with additive white gaussian noise (binary symmetric channel) to determine a suitable value for N and to determine the synchronization time. These simulations were performed assuming that bit slip errors would not occur and that the synchronization algorithm would not attempt to correct for bit slip errors. To determine a suitable value of N, 30000 frames of 168 bits each were transmitted with a given \( P_e \). The value of N was increased until no loss of synchronization was observed. Thirty thousand frames equals 600 seconds of speech, which was felt to be a likely time between bit slip synchronization loss. The results are as follows: for \( P_e = 0.12 \) a suitable N was 9, for \( P_e = 0.1 \) a suitable N was 6, for \( P_e = 0.07 \) a suitable N was 4. These values
of $P_e$ are presented because, as stated earlier, the synchronization algorithm was designed to work with a BER up to $10^{-1}$. Thus, a suitable value of $N$, for use with a channel with a BER up to $10^{-1}$, is at least 6.

Simulations were also used to determine the synchronization time, which was measured versus channel error rate and frame length. Figure 10 gives the synchronization time required for a channel with a BER from 0 to 0.15, and frame lengths from 168 to 1344 bits. Each point on figure 10 was obtained from the average synchronization time of 400 simulated synchronization attempts. The average synchronization time in seconds is determined from figure 10 by dividing the average synchronization delay in bits by the channel bit rate. Thus, given that a synchronization time less than 250ms is desired for a channel BER up to 0.1 and a data rate of 19200 bps, a frame length of 366 bits or less should be chosen. A coded speech frame with 8 bits of parity protection is 168 bits long, therefore two frames of speech can be transmitted between synchronization markers with an acceptable synchronization time.

![Figure 10: Simulated synchronization time in bits for frame lengths of 168, 336, 672, and 1344 bits.](image)

**Marker Selection** The marker is chosen to provide the best possible synchronization performance with the least overhead. Marker overhead is proportional to the product of the marker length and the
frequency at which the marker is placed into the data stream (the marker-to-marker distance.) Up to a point, the longer the marker, the longer the distance between successive markers can be. Thus, there is a trade-off between marker length and marker-to-marker distance. The requirement of framed speech transmission prescribes that a minimum marker distance be at least 364 bits, which is the length of a half-rate encoded frame. The other requirements, mentioned earlier in this chapter, dictate that the marker be around 8 bits long. Therefore, an 8 bit long marker was chosen on the basis of the 8 bit microcontroller word length.

The standard marker sequence is usually chosen to have the lowest possible autocorrelation sidelobes. Barker sequences[39, pp 351–2] have ideal autocorrelation functions and are often used as markers. For an 8–bit marker, only a small amount of sidelobe information is available. For the algorithm described in section 3.2, the autocorrelation properties are not important at all. Only when the addition of bit slip correction is included into the algorithm are low sidelobes required. It seemed possible and practical to have the synchronization algorithm attempt to correct for bit slips. It was therefore required that:

1. A one-bit–shifted version of the marker have as low an autocorrelation function as possible.

2. The bit-shifted versions (a right shift and left shift) be distinct.

The second requirement provides a distinction between a bit shift resulting from a lost bit or an inserted bit. Based on these criteria, the marker sequence 01100101 was chosen by an exhaustive computer search. This sequence, when shifted by one bit to the right or left, will have 2 bits in agreement and 5 bits different. These facts were used to obtain eqn. (1). The autocorrelation function of the marker, when bracketed by a random bit stream, is shown in figure 11.
3.3 Finding an Optimal Walk Distance

Earlier, a suitable value of N was developed by simulating the synchronization algorithm. This section presents the synchronization results obtained from transmission of pseudo-random frames over power line communication channels.

To find the most suitable value of N, test frames were transmitted between two modems over a variety of channels. No measured false-unlocks were detected on any of the channels with a BER < $10^{-2}$ and values of N>2. Channels with a BER > $10^{-2}$, however, required a larger value of N to reduce the probability of frame synchronization loss. Detailed measurements were performed for 3 representative channels: a same-phase channel, a low random noise cross-phase channel, and a cross-phase channel with a high noise level.

The tests were performed between 8pm and 10pm which is a time when channel conditions are found to remain constant. Once a suitable channel was found, 90 seconds (9820 frames) of 21 byte frames at
19200 bps were transmitted. The BER and the number of times the frame synchronization was lost were measured. Then, the output power from the modem was reduced, thereby increasing the BER, and the same test was run again. A 90-second test period was chosen to reduce the likelihood that the power line channel would vary during a test. The results from these tests are displayed in figures 12 and 13 where the BER reported was the average over the 90-second period.

The number of times frame synchronization was lost is shown in figure 12 for the case of cross-phase and same-phase transmission. Curves A – C were measured for the same X-phase channel, which is an example of a channel with both burst and random noises. Curve D is a same-phase transmission. There are several items to note from this figure:

1. For N below some threshold, synchronization loss probability is prohibitively high.
2. For N above some threshold, synchronization loss probability is low.
3. A value of N=8 appears optimum.
4. In a high random noise environment a large value of N is required.
5. The same-phase transmission has a higher BER, yet works well with a smaller value of N.
6. Higher values of N are required in higher noise environments.
7. Even with high N values frame synchronization may be lost.

Figure 13 gives the number of times frames synchronization is lost versus the maximum walk distance N for a channel with lower random noise than in figure 12. For this channel two points can be made, in addition to points 1 and 2 above

3. A value of N = 6 appears optimum.
4. As the BER increases the number of bit-slips increases also.
Combining this list with the one based on figure 12 leads to other observations. As the BER increases so does the optimum N. For high noise conditions an optimum value of N would be 8, 9 or 10.

The product of walk distance and number of times frames were lost versus the maximum walk distance is of interest. This product gives the maximum number of frames lost until the synchronization algorithm enters the START/RESET state. This product is shown in figure 14 for the three representative channels of figures 12 and 13. We observe that the product reaches the lowest point and remains constant after a walk distance of N = 9 or 10, and remains approximately constant for curves A, D, E, F, and G, but increases for line B, C, and H via bit slips. Thus, a value of N = 9 or 10 appears to be best for minimizing the number of lost frames.

![Figure 12](image_url)

Figure 12 Number of times frame synchronization was lost versus channel BER and maximum walk distance during a 90 second test interval for a channel "bad" cross-phase (X-phase) and of the same phase.
Figure 13 Number of times frame synchronization was lost versus channel BER and maximum walk distance, during a 90 second test interval for a channel “good” cross phase (X-phase).

Figure 14 Maximum number of frames lost versus channel BER and maximum walk distance, during a 90 second test interval.

3.4 Modem Hardware Modifications

This section describes hardware modifications performed to enable existing data modems to work
over poor quality channels. The modified power line modem was designed originally to transmit computer data [4]. Convolutional encoding and majority logic decoding enhancements to the modem are described in [11]. The designs were intended for general data transmission and were optimized to provide low bit error rates on good channels. At channel BERs above $10^{-3}$ frequent bits slips and carrier phase synchronization loss were observed. These problems are unacceptable for speech transmission as bit slip will result in lost frame synchronization and carrier synchronization loss results in bit inversion.

Two simple enhancements were performed which alleviated these problems. The problem of carrier synchronization loss was solved by differentially encoding the bits (DPSK). The problem of bit slips was corrected by adding a random walk filter in the bit synchronization circuits.

Differential encoding handles carrier synchronization loss by encoding the bits such that the absolute phase of the received signal does not carry information. Data is encoded by difference in the carrier phase between bits. Figure 15 illustrates how differential encoding and decoding is performed before the modulator and after the demodulator, respectively, using the equations[40]

$$M'_n = M'_{n-1} \oplus M_n$$

$$R'_n = R_n \oplus R_{n-1}$$

where $M_i$ is the i-th bit to the modulator and $M'_i$ is the i-th differentially encoded bit. Likewise, $R_i$ is i-th received bit from the demodulator and $R'_i$ is the i-th differentially decoded bit from the demodulator. Because the differential encoded signal does not depend on the absolute phase of the received signal, a carrier phase slip will result in the inversion of only one or two bits.

The main problem with DPSK is an increased BER because of error propagation. Differential decoding algorithms produce two bit errors for a single received bit error. Thus, at high SNR a
differentially coded communication system will have approximately twice the BER of a coherent system. Figure 16 approximates\(^2\) the number of frames corrupted by carrier synchronization loss as a function of channel BER (which was calculated only from frames which did not experience carrier synchronization loss) for a 90-second period, over the same phase, with a framelenlength of 48 bytes at 19200 bps. Given the large number of frames corrupted due to carrier loss, differential encoding was considered necessary.

![Diagram](image)

**Figure 15** Differential encoder (a), differential decoder (b).

The second modification to the hardware was the inclusion of a random walk filter (also known as sequential filter) [41, 42] in the bit-synchronization circuitry. As stated, the original modem was designed to transmit computer data with a low error rate and high SNR. With high SNR a modest bit synchronization circuit was effective. Voice communication does not require such high SNR, but under lower SNR excessive bit jitter as well as bit-slip were observed.

\(^2\) A frame with more then 25% bits in error is considered here to have lost carrier synchronization. If carrier synchronization loss occurred near enough to the end of the frame then it would not have been recorded. Similarly, frames damaged from other means are recorded here as damaged by carrier loss, although this event was considered unlikely.
Figure 16 Number of times carrier synchronization was lost over a 90-second interval.

Figure 17a gives the original bit synchronization circuit used in the power line modem. The circuit generated the bit synchronization (bit sync) signal by dividing down a 24X-clock signal. When the phase detector detects bit sync, the clock is divided by 24. If the phase of the bit sync signal is detected leading, then the counter counts to 26 instead of 24, effectively lengthening the bit. Similarly, if the bit phase detector detects the bit sync signal lagging, then the counter is set only to 22.

This simple method effectively performs bit synchronization. At lower SNRs, the phase detector will produce incorrect phase differences and thus spurious UP/DOWN signals. The net effect, under random noise, is to increase the phase jitter. However, during a deep fade such as the periodic fades which may last dozens of bits on power line channels, loss of sync-lock may be lost result in bit slip and frame synchronization loss.

Adding a random walk filter after the phase detector reduces spurious UP/DOWN phase detector signals. Figure 18 illustrates the action of an N-step random walk filter. The filter generates an UP’ signal
upon receipt of sufficient UP signals for it to “walk” \( N \)-steps in the up direction. Given that spurious UP/DOWN signals from phase detectors are random, the likelihood that a false UP’ or DOWN’ signal will be generated decreases approximately exponentially with \( N \) [38, pp 212–3]. Thus, by increasing \( N \) the number of spurious UP’/DOWN’ signals can be reduced to an arbitrarily small value. The trade-off for this increased performance is a reduced locking-range and an increased lock-time.

The circuit chosen was constructed using an \( N=16 \) step filter. The hardware implementation is given in Appendix D. A 16 step filter was chosen on the basis of simple construction and performance, up to BERs of 0.1.

![Diagram](image)

Figure 17 Original bit sync circuit (a), bit sync circuit with random walk filter (b).
Figure 18  State transition diagram for a length N random walk filter with inputs Up/Down and outputs UP'/DOWN'
Chapter 4 – Quantitative and Qualitative Measurements

This chapter describes the quantitative and subjective measurements performed to evaluate power line voice communication quality. The procedure used to measure the real-time channel parameters, BER, and number of lost frames is described. This description is followed by a summary of the quantitative results obtained. A description of the subjective test procedure is given, followed by the test results used to estimate the sensitivity of the encoded speech parameters to noise.

4.1 Real-Time Quantitative Measurements

The quantitative measurements recorded during the development of the synchronization algorithm and FEC schemes included BER, number of lost frames, number of synchronization losses, and number of mis-synchronizations. This section describes the method used to measure these performance indicators as well as the measured results.

Test Pattern Generation and Decoding

Given that the synchronization algorithm was intended to synchronize speech frames, ideally the essential parameters would be measured using compressed speech data. Unfortunately, measuring these parameters directly from received speech data in real-time would be difficult. For example, determining the BER in real-time requires the receiver to have a priori knowledge of the correct speech data. The frame-to-frame randomness of speech data makes this task computationally difficult. One approach is to use test patterns which are related mathematically from byte-to-byte and frame-to-frame.
The test pattern was constructed to fulfill the several requirements. It must possess a frame id number in order to enable determination of the number of frames lost and the time between synchronization losses. The number and location of errors should be measurable to enable determination of the BER and correlation (burstiness) of errors. The test method must be sufficiently robust to work in a high noise environment with up to 20 percent BER. The test pattern length should be arbitrary to allow testing of any frame length. The test pattern generation should be simple and rapid enough for real-time analysis. Finally, the test pattern should be representative of the speech data (compressed speech data appears more or less random from bit-to-bit and frame-to-frame).

Test pattern randomness is very important with regard to bit and frame synchronization. Bit synchronization requires delineation of bit edges[43], while frame synchronization requires detection of frame markers. In the case of frame synchronization, frame-to-frame randomness is essential; a frame containing the synchronization marker sequence repeated in consecutive frames could result in a false synchronization lock. Fixed (non-random) test frames require the exclusion of the sync marker. Using frames which exclude the marker sequence is inappropriate because speech data will contain some markers, unless bit stuffing is used.

The encoding technique uses a 16-bit frame id along with the byte position within the frame to produce pseudo-random frames. The pseudo-random frame is derived using the frame id. The id is first rotated by the number of one bits the integer contains, to obtain $\tilde{id}$:

$$\tilde{id} = \text{rot}(id, \text{ones}(id))$$  \hspace{1cm} (5)

where the function $\text{rot}(a, b)$ rotates the binary $a$ $b$-times to the left, and the function $\text{ones}(a)$ gives the number of ones in the binary representation of $a$. Next, $\tilde{id}$ it is split into two bytes: $\text{high}$ and $\text{low}$.

$$\text{high} = \tilde{id}/255, \quad \text{low} = \tilde{id} \% 255$$  \hspace{1cm} (6)
where % denotes the remainder function. Finally, the frame values of \( a_1 \) through \( a_{n/2} \) and \( b_1 \) through \( b_{n/2} \) are calculated using:

\[
a_i = \text{lfsr}(2i) \oplus \text{high} \quad b_i = \text{lfsr}(2i + 1) \oplus \text{low}
\]

(7)

The symbol \( \oplus \) denotes modulo-2 exclusive OR. The function \( \text{lfsr}(y) \) is the output of a “maximal length” linear feedback shift register\([44, \text{p 175}]\) at the \( y \)th time step.

The \( a_n \)'s and \( b_n \)'s are arranged to form a frame as shown in figure 19.

![Figure 19 Construction of test pattern](image)

The received frames are decoded in a reverse manner. First, the received bytes are processed using an operation which is the inverse of eqn. (7):

\[
\text{high}_n = \text{lfsr}(2n) \oplus a_n, \quad \text{low}_n = \text{lfsr}(2n + 1) \oplus b_n
\]

(8)

Once decoded, the \( \text{high}_i \)'s, and \( \text{low}_i \)'s are compared. If the frame has not been corrupted then

\( \text{high}_1 = \text{high}_2 = \ldots = \text{high}_n = \text{high}' \) as well \( \text{low}_1 = \text{low}_2 = \ldots = \text{low}_n = \text{low}' \). Under such noiseless conditions \( \text{high}' \) and \( \text{low}' \) are processed with a reverse of equations (7) and (5), respectively:

\[
\tilde{i}' = \text{high}' \times 255 + \text{low}'
\]

(9)

\[
\tilde{i}' = \text{rotr} \left( \tilde{i}', \text{ones} \left( \tilde{i}' \right) \right)
\]

(10)

where \( \text{rotr}(a,b) \) performs a \( b \)-fold right rotation of symbol \( a \), and \( i' \) is the decoded frame number. Finally, by using the value of \( i' \), the transmitted frame can be reconstructed using (5), (6), and (7). A comparison of the original and the reconstructed frames provides the number and location of bit errors within the frame.
In the case of a noise corrupted transmission, if a sufficient number of high values are the same then that value is chosen as high'. Likewise, sufficient “agreement” among the low leads to a low’ value. More specifically, the algorithm used for the qualitative measurements in this thesis employed the agreement condition:

\[ \text{high}_1 = \text{high}_2 = \text{high}_3 \quad \& \quad \text{low}_1 = \text{low}_2 = \text{low}_3 \quad (11) \]

The probability of a random frame producing a false agreement is \( P_{fa} = 2^{-32} \approx 10^{-10} \). The probability that random errors will cause a failed agreement is given by

\[ P_{na} = 1 - (1 - BER)^{48} \quad (12) \]

where \( P_{na} \) is the probability random channel errors will cause a failed agreement (i.e., 11 is not satisfied) and BER is the channel bit error rate.

With noisy transmission, the agreement condition (11) is met with probability \( P_{na} \). When the condition is met, \( i' \) is used in equations (5), (6), and (7) to generate an error free frame. The error free frame is then used to determine the number of bit errors in the received frame. If noise has corrupted any of the first 6 bytes then the agreement condition will not be met (with approximately \( P_{fa} \)). In this case, the frame id from the previous frame is incremented and used as the basis to generate the error free frame. Receipt of a frame with the agreement condition met, therefore, provides a “decoder synchronization” which is used to decode the test frames in which the agreement condition is not met.

As long as \( P_{na} \) is sufficiently large and \( P_{fa} \) sufficiently small, this scheme of measuring the number and location of errors is effective. For example, where the BER = 0.01, 38% of frames (\( P_{na} = 0.38 \)) will satisfy the agreement condition. With a higher BER = 0.1, \( P_{na} = 0.006 \). Such small values of \( P_{na} \) may be unacceptable because this implies that a large number of frames are received between decoder
synchronizations. The bursty nature of power line errors, however, meant that $P_{na}$ is a conservative value and therefore we did not observe long periods of time between frame synchronizations.

This method of encoding therefore turns the frame id into pseudo random data. Through the use of more complex agreement conditions it is possible to adapt this method for testing of channels with BERs approaching 0.5.

**Bit Error Rate**

BER measurements over power line channels through the electrical engineering building at the University of British Columbia were performed. By varying the location of the transmitting and receiving modems, BERs ranging from $10^{-6}$ to values in excess of 0.2 were observed. These results are consistent with those described previously for this modem [4, 11].

The BER was found to change by as much as one order of magnitude during the course of a day. In addition, the BER across some channels would change by a factor of 2 or 3 over the time-frame of a few seconds. Thus, it is important to consider the measurement period of a recorded BER value. In most cases in this thesis, the measurement period was 60 seconds. Further, it was discovered that channel quality was less variable in the evening, between 8–10 pm, when most building equipment would not be switched on and off randomly. Thus, the channel measurements reported were taken during this time.

**Performance of Sequential Filter**

The design of a random walk filter was discussed in chapter 3. Its effect on the bit error rate and the synchronization loss rate parameters is presented below. The objective of these measurements was to determine how much (if any) improvement is obtained using a random walk filter. The best way to observe the performance improvements is by measuring the above parameters over the same channel with
and without the sequential filter in place. Unfortunately, the power line channel conditions vary with time. To help reduce the time varying effects, the following measurement scheme was chosen.

1. The channel BER was measured with the sequential filter in place.
2. BER and frame and synchronization loss over a 60 second interval were measured using the sequential filter.
3. BER and frame and synchronization loss over a 60 second interval were measured without using the sequential filter.
4. BER and frame and synchronization loss over a 60 second interval were measured again using the sequential filter.
5. If initial BER (step 2) and re-measured BER (step 4) differ by more than 4 percent, then steps 1 through 5 were repeated.

A 60 second measurement interval was chosen to minimize the effect of the time varying channel. Test data were transmitted across 3 different power line channels to perform these measurements. By varying the output power of the transmitting modem, the parameters could be measured over a range of bit error rates.

The results obtained from measurements across three typical channels appear in table 2. Part A of table 2 shows the results obtained within the UBC Electrical Engineering (MCLD 458) communication lab across power phases. At full modem transmit power, a BER of $5.9 \times 10^{-4}$ was measured for the modem with the sequential filter employed. No bit-slips were measured with or without the sequential filter. As the output power of the modem was reduced, the BER increased. With a BER of about $0.93 \times 10^{-2}$, the first bit slips without the sequential filter were recorded. As the output power was further reduced, the number of bit slips occurring increased both with and without the sequential filter. However, table 2
shows that the number of bit slips occurring was very much greater without the filter. We can further compare the BER with and without the sequential filter. We observe that at low BERs (high SNR) the BER with and without the sequential filter is approximately the same. At lower SNR, however, the BER without the sequential filter can be up to 20% higher.

Similar results were obtained with the other channels (parts B and C of table 2). Use of the sequential filter considerably reduced the number of times bit synchronization was lost. The reduction is large at high BER values. The BER was not greatly affected by the sequential filter, but was always lower with it than without it.

<table>
<thead>
<tr>
<th>BER (x10^-2)</th>
<th>bit slips</th>
<th>BER (x10^-2)</th>
<th>bit slips</th>
<th>BER (x10^-2)</th>
<th>bit slips</th>
<th>BER (x10^-2)</th>
<th>bit slips</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A) X-phase &quot;good&quot; channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with seq. filter</td>
<td>without filter</td>
<td>with seq. filter</td>
<td>without filter</td>
<td>with seq. filter</td>
<td>without filter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.059</td>
<td>0</td>
<td>0.067</td>
<td>0</td>
<td>1.46</td>
<td>0</td>
<td>1.45</td>
<td>0</td>
</tr>
<tr>
<td>0.15</td>
<td>0</td>
<td>0.16</td>
<td>0</td>
<td>1.92</td>
<td>0</td>
<td>2.00</td>
<td>3</td>
</tr>
<tr>
<td>0.93</td>
<td>0</td>
<td>1.00</td>
<td>2</td>
<td>2.97</td>
<td>0</td>
<td>3.94</td>
<td>121</td>
</tr>
<tr>
<td>3.24</td>
<td>0</td>
<td>3.75</td>
<td>15</td>
<td>4.71</td>
<td>0</td>
<td>6.38</td>
<td>82</td>
</tr>
<tr>
<td>3.77</td>
<td>5</td>
<td>4.15</td>
<td>9</td>
<td>4.93</td>
<td>1.5</td>
<td>6.61</td>
<td>131</td>
</tr>
<tr>
<td>4.19</td>
<td>6.5</td>
<td>4.87</td>
<td>15</td>
<td>10.3</td>
<td>4.5</td>
<td>11.6</td>
<td>216</td>
</tr>
<tr>
<td>6.73</td>
<td>4</td>
<td>8.28</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5</td>
<td>7</td>
<td>14.6</td>
<td>470</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Number of times bit synchronization was lost and BER for three typical channels with and without the addition of a random walk receiver filter

4.2 Subjective Measurements

Two series of subjective measurements were performed. The first series was used to measure the perceptual significance of errors in selected speech code parameters. The second series of tests
determined the performance of the communication system as a function of channel quality (BER) and data transmission rate (9600 bps and 19200 bps). This section describes the experimental method and the results obtained in the perceptual sensitivity experiments.

**Subjective Test Procedure**

In both series of tests speech samples were recorded in a quiet room. The recording apparatus consisted of a Sony F-VX30 dynamic microphone, KROHN-HITE model 3202 dual variable filter for input and output low-pass anti-aliasing filtering (with a cutoff at 3300 Hz), 12-bit A/D converter on the Spectrum 56001 DSP board running on an IBM-AT compatible, and a Sony TC-330 cassette/reel tape recorder used as a microphone preamplifier. Sampling was performed at 8000 samples/second. This hi-fidelity system provided a digital audio signal of higher quality than would be available in a typical application of this communication system, thus allowing an assessment of the voice coding algorithms.

The speech data was then transferred to sun workstations for processing. The processed speech data was then transferred back to the computer in the quiet room to be played to listeners. The speech test using the modem required the additional step of transferring the data to the IBM PC's interfaced to the modems. This arrangement is shown in figure 20. Blocks 3 and 4 were required when measuring the performance across the power line.

Subjects in the quiet room listened to the speech samples reconstructed and played from the D/A converter of the Spectrum 56001 DSP board. The Sony TC-330 cassette/reel tape recorder amplified the signal to the level appropriate for the JVC HA-D500 headphone. As with the recording setup, the quality of the system exceeded the performance of any application of a communication system employing the VSELP algorithm.
The listening tests for perceptual significance measurements were conducted with subjects chosen from students and staff in the electrical engineering department. They were seated in the quiet room at a desk in front of the test computer and given the test instructions (Appendix A and B). The computer recorded the score for each of the samples.

Testing Perceptual Significance of Parameter Errors

**Background** As stated in chapter 2, the output of the speech encoder is a frame of speech parameters. Errors in some parameters are more noticeable than errors in others. This section presents the method used to determine the perceptual significance of each parameter as well as the results obtained.

Three common subjective measurement procedures include: isopreference method, relative preference method, and the category-judgement method[45]. The category judgement method has the listener assign a category to a sample of speech relative to some subjective standards. This method was chosen for our use. The commonly used categories are based on a five point scale: unsatisfactory = 1, poor = 2, fair =
3, good = 4 and excellent = 5. When the listening tests are performed with a set of subjects the mean opinion score (MOS) of the sample is recorded. MOS ratings are commonly used as a subjective standard for comparison of communication system and speech compression algorithms [46, 47, 48]. For example, toll grade communication is 4.0–4.5. The VSELP compression algorithm has a MOS of approximately 3.5 [20, 49].

**Test procedure** The category judgement test method described above was used, but with a modified set of MOS categories. These categories were based on speech degradation with respect to an error-free compressed sample as opposed to speech absolute quality. This was done because the amount of speech degradation symbol errors produced was to be measured, rather than the quality of the signal, and because with VSELP compressed speech does not qualify for the “excellent” category. A similar approach was chosen by Coleman [49] where a five level “Listening Effort Scale” was used to estimate speech intelligibility.

Subjects were given two reference samples, a “good sample” consisted of the compressed speech without errors; a “poor sample” included many errors and was arbitrarily chosen as a lower limit of speech quality. With respect to these two references, subjects were asked to select the appropriate category from the list below. By judiciously choosing a poor reference sample, the test could be better focused to discern minors degradation in quality. Further, the choice of categories 4 and 5 attempted to force the subjects to answer the question “At what level are symbol errors detectable?”

- 5 – Equal to or better than good sample
- 4 – Slightly worse than the good sample
- 3 – Significantly worse than good sample
2 – Slightly better than poor sample

1 – Worse than or equal the poor sample

The test consisted of 120 voice samples. Each sample contained two phonetically balanced sentences read by a male and female speaker and chosen from the list: “Lift the square stone over the fence.”, “Mesh wire keeps chicks inside.”, “The two met while playing on the sand.”, “Eight miles of woodland burnt to waste.”, and “The crooked maze failed to fool the mouse.” Three female and two male readers were used. All readers spoke with a Canadian accent.

Test Results  Fourteen subjects participated in these tests. The MOS scores for the R0, LPC1, LPC2, LPC4, GSP0, LAG, CODE1, and CODE2 parameters are graphed as a function of symbol error rate in figures 21 to 24. These graphs enable direct comparison of the perceptual sensitivity of the parameter errors. As well, they show the utility of adding error detection. The error detected scores give the sound quality improvement when damaged speech parameters are replaced with the most recent correct parameter.

Table 3 restates the data given in figures 21 to 24 which highlight the relevance of forward error correction. This table answers the question “At what symbol error rate does a perceptual decrease in quality occur?” For example a MOS score of 4.0 has category of “slightly worse than good sample.” Thus, compressed speech with a symbol error rate of 0.019 in the R0 parameter will have a MOS of 4.0 when decoded. We can compare this to the 0.15 symbol error rate the protected R0 parameter can tolerate before decoded speech has a MOS of 4.0.
Figure 21  MOS score versus symbol error rate for the GSP0 and LAG parameters with and without error detection.

Figure 22  MOS score versus symbol error rate for the Frame energy (R0) and 1st LPC coefficient with and without error detection.
Figure 23: MOS score versus symbol error rate for the 2nd and 4 LPC coefficient with and without error detection.

Figure 24: MOS score versus symbol error rate for the residual vectors CODE1 and CODE2, 6th LPC coefficient, and with error detection the for residual vectors and the 8th LPC coefficient.
<table>
<thead>
<tr>
<th>MOS</th>
<th>R0</th>
<th>LPC1</th>
<th>LPC2</th>
<th>LPC4</th>
<th>LPC6</th>
<th>LPC8</th>
<th>GSPO</th>
<th>LAG</th>
<th>CODE 1&amp;2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>1.9/15</td>
<td>1.5/7</td>
<td>3.7/12</td>
<td>7.5/35</td>
<td>27/NM</td>
<td>NM/79</td>
<td>1.5/6.3</td>
<td>0.7/10</td>
<td>5/12</td>
</tr>
<tr>
<td>3.5</td>
<td>3.0/27</td>
<td>4.0/15</td>
<td>7.2/30</td>
<td>15/50</td>
<td>42/NM</td>
<td>NM/NA</td>
<td>4.5/13</td>
<td>3/20</td>
<td>21/24</td>
</tr>
<tr>
<td>3.0</td>
<td>5.0/40</td>
<td>7.0/25</td>
<td>15/40</td>
<td>24/69</td>
<td>65/NM</td>
<td>NM/NA</td>
<td>9.0/27</td>
<td>9.0/35</td>
<td>33/50</td>
</tr>
</tbody>
</table>

NA = not applicable, NM = not measured.

Table 3 Summary of MOS degradation used to determine error sensitivity of compressed speech parameters.

**Lossless scores**

Listening tests place the quality of the VSELP encoder with a MOS score between 3.5 and 4.0[20, 49]. Voice coders scoring above 4 are considered high-quality, producing no noticeable quality degradation compared to a standard telephone line [14]. Voice coders with a MOS score of 3.5 produce “communication quality” speech that is a level of quality “characterized by a level of speech degradation that is easily detectable but not bad enough to impede natural telephone communication” [14]

**4.3 Subjective Evaluation of FEC schemes**

Subjective testing was used to compare the quality of speech encoded at 9600 bps with a minimal amount of FEC versus 19200 bps encoded speech with considerably more FEC. The implementation of these two methods is described in the following chapter. These subjective tests had two goals. The first was to compare the quality of the 19200 bps encoded speech versus the quality of 9600 bps encoded speech over a random noise channel to determine the noise tolerance increase offered by the FEC schemes.
The second goal was to determine the net sound quality difference of the 9600 bps implementation and the 19200 bps implementation. This comparison was required to determine if the increase in channel errors caused by the higher baud rate transmission would negate the benefits of the additional FEC.

Similar subjective tests of voice compression algorithms have been conducted previously to determine the quality of compressed speech for use over mobile channels using DPCM speech coding at 16-32 kbps [50], and to evaluate the RPE-LTP 13kbit/s GSM [49] algorithm subject to random noise. These tests used the MOS rating technique to assess the overall quality of the communication system subject to noise. MOS scoring is used in our subjective tests as well.

Speech sample creation is described below. First, four representative power line channels (modem locations) were chosen. In each case, the transmitting modem was located in MCLD 458. The receiving modem was placed either in the hallway near the MCLD 458 or in MCLD 203. The phase into which the transmitting modem was plugged could be changed to allow for same-phase or cross-phase transmission. Transmissions to the nearby hallway were considered to be over a “good channel” while longer distance transmissions to room 203 were considered to be over a “poor channel.” The four channels created can be described as 1 – good same-phase, 2 – good cross-phase, 3 – poor same-phase, 4 – poor cross-phase. For each of the four channels, measurements were performed with the transmitter power varying from 7.1v to .071v RMS (77dBmV to 37dBmV). Thereby, the performance with many types of channel conditions could be measured. Next, the speech samples were transmitted. These tests were performed between 8pm and 10pm which is a time when power line conditions were found to be consistent. For each of the four test channels, the following steps were used create the speech samples:

1. The transmitter power was set to maximum (77dBmV).
2. The channel BER was measured at 19200 bps for 30 seconds. If the BER was less than $10^{-1}$, the
19200 bps encoded speech sample was transmitted.

3. The 9600 bps encoded sample was transmitted.

4. The rate was then increased to 19200 bps and the channel BER was remeasured for 30 seconds.

5. If the channel BERs of steps 2 and 4 were within 10% of each other, then the two speech samples were stored, the transmitter power was reduced, and steps 2 to 4 were repeated to create a new set of speech samples. If the channel BER of steps 2 and 4 were not within 10% of each other, then steps 2 to 4 were repeated without changing the transmitter power.

With the speech samples stored, the final step was to decompress them and perform subjective tests in which 19 people were asked to rate the randomly ordered samples according to the following five-category MOS scale: 1 = unsatisfactory, 2 = poor, 3 = fair, 4 = good, 5 = excellent. The instruction sheet supplied to the candidates appears in appendix B. As with the previous test, the subjective tests were conducted in the quiet room with the aforementioned speech processing apparatus. The sentences chosen were “Fairy tales should be fun to write.” and “The juice of lemons makes a fine punch”, read by a female and male reader, respectively.
Chapter 5 – Error Control in Speech Coding

This chapter describes the two schemes implemented which enhance the quality of speech transmitted over a noisy power line channel. The motivations and general options available for error control and forward error correction (FEC) are presented, followed by a detailed description of the speech coding algorithms. Finally, a subjective sound quality comparison of the two speech coding and transmission methods is presented for both a random noise channel and power line channel.

5.1 Error Control Options and Background

Various approaches are available to improve the sound quality of compressed speech. The first scheme makes use of a voice compression algorithm designed to minimize the effect of channel errors [51]. The VSELP scheme [30] chosen reduces the significance of bit errors through the use of vector residual quantization, log scale LPC scalar quantization with grey coding, and post-filtering. In the second scheme, redundant bits are added to the encoded bit stream to provide symbol error detection. As stated in chapter 2, the codewords tend to remain stationary from frame to frame. We can often reduce the perceptual significance of an error in a parameter by substituting the previous parameter. The third available method is to employ conventional FEC code to reduce the BER following decoding. Typically, this may involve the use of convolution encoding and decoding, or BCH block coding and decoding. The scheme created to deal with the power line burst errors for the thesis uses a code repetition scheme. Later, it will be shown how this provides both error detection and correction.

When protecting speech parameters on a symbol-by-symbol basis one would ideally allocate error control bits on the basis of the perceptual significance of errors in a given parameter. This is the basis
of the subjective testing performed in chapter 4. Allocation of protection bits on the basis of perceptual significance is used for mobile communication [52, 53, pp 81–95] where symbols are protected using BCH block codes [15, pp. 141–77]. In addition, LeBlanc describes channel coding [54] of CELP encoded speech for a mobile channel using Reed-Solomon or rate punctured convolutional codes along with convolutional interleaving.

In addition to allocation based on the perceptual significance of errors in encoded parameters, it is appropriate to consider allocating redundant bits to allow for graceful degradation or a minimal level of service in severe noise conditions. Codewords which are essential for a minimal level of communication would be allocated additional error correction bits, possibly at the expense of sound quality on better channels.

Ideally, the perceptual significance errors in each bit of each symbol would be considered when formulating an FEC scheme; however, three problems arise. First, measuring the significance of each encoded bit by subjective testing would be very time intensive. Second, finding the perceptual significance of errors in a single bit does not necessarily provide information on the effect of multiple bit errors. Third, there exists no theoretical scheme for encoding of variable importance data which also allows for symbol error detection.

9600 bps Coding Scheme

The 9600 bps coding scheme was designed to work with the power line modem in 9600 bps mode. This scheme is a simplification of the method chosen for the USDC digital cellular phone standard [18]. The FEC method described in the standard begins by using a seven bit cyclic redundant code (CRC) [15] to confirm the accuracy of the 12 most perceptually significant bits of each frame. To provide further
redundancy, the 84 most significant bits of each frame along with the aforementioned CRC are 1/2 rate convolutionally encoded and interleaved. The complete bit stream rate is 13500bps.

When the speech frame is received, it is first de-interleaved and then trellis decoded. Next, the CRC is checked. If the CRC check fails, the 12 most perceptually significant bits from the most recent packet (which passed the CRC check) would be utilized. This method has not been tested using the PLM because the aggregate rate of 13.5 kbps would under-utilize the 19.2 kbs channel capacity.

The 9600 bps FEC method implemented for this thesis work was a simplified version of the USDC method. Here, only the seven bit CRC is calculated using the 12 most significant bits. These “most significant bits” are specified in the USDC standard but no justification is given for their validity. The aggregate bit stream including the eight bit synchronization marker, seven-bit CRC, and two bits of modem overhead result in a 8800 bps bit stream; (7+2+8+159)*50 = 8800 bps. This allows transmission at the lower bit rate of 9600 bps and leaves available some bandwidth for additional signaling.

Decoding the received bit stream requires recalculating and comparing the CRCs. If the CRCs are equal (CRC agreement) the received frame is sent to the voice decoder. If the CRC are not equal, then depending on how many frames received since the last CRC agreement the following will occur: If the last CRC agreement was one or two frames previous then the parameters from the last correct frame are used. If last CRC agreement was three to five frames previous, then the parameters from the last correct frame are used, but the R0 (frame energy) parameter is decremented by two (4 dB). If the last CRC agreement was six or more frames previous then the R0 parameter is set to zero, muting the output. Thus, if a succession of incorrect CRCs are received, the parameters from the last correct frame are

---

3 A “correct frame” is understood as a frame where the received and recalculated CRC are equal. This condition will exist, with a small probability, even if the CRC protected bits are in error.
recycled for two frames and then the output is muted over the next three frames. This would occur typically if frame synchronization was lost.

The advantages of using this method include simple implementation, ability to transmit in full duplex mode at a baud rate of 19.2 kbps, and a lower bit rate (9600 bps) for more robust communication when 19200 bps is not viable. Conventional forward error correction is not provided. The CRC detects only errors of the most significant speech parameter bits. Therefore this method provides speech protection only through the use of error detection and parameter repeat. A disadvantage of using this method is its comparative intolerance to noise with speech quality becoming unacceptable when BER \( \approx 10^{-2} \).

### 5.2 19200 bps Coding Algorithm

A full-rate (19200 bps) voice coding scheme was desired to utilize the full 19200 bps bandwidth. Requirements for this scheme include a simple implementation executable in real-time and the ability to provide symbol error detection and error correction. Also, the error correction scheme must be suited to function in a burst noise environment. and, the voice coding scheme must provide a graceful degradation in speech quality during noisy communication times.

Given these criteria and the symbol perceptual sensitivity data in chapter 4, a simple scheme was constructed where the speech parameters are divided into three classes. Class 3 symbols are the least important. No error correction or error protection is provided. Class 2 symbols were considered significant enough to require a four bit error protection CRC. Class 1 symbols required the most protection. Class 1 symbols each had a four bit CRC (similar to the class 2 symbols), but in addition, each symbol and corresponding CRC was duplicated. This simple method provided both error detection and error correction of class 1 symbols.
Table 4 Redundant bit allocation for symbols.

Table 4 provides a listing of symbol class and the number of bits required by each parameter, as well as the total number of bits required. Class 3 symbols comprise most of the 344 bits per frame. The aggregate bandwidth required for this implementation including the encoded data, synchronization marker and modem overhead is (344+8+2) = 354 bits/frame or 17700 bps.

Parameter decoding was somewhat more complicated than the encoding. Received parameters are first deinterleaved. Next, the CRC value of each class 2 symbol is computed (CRC_c) and compared to the received CRC value (CRC_r). If the CRCs were equal (CRC_c = CRC_r), then the parameter received is passed to the voice decoder. If the recomputed and received CRCs are not equal, then the parameter from the last frame with a CRC agreement is used. Decoding class 1 symbols requires recomputing and comparing CRCs for both copies of the received parameter. Call P1 and P2 the first and second copies of the received parameter, respectively. Likewise, call CRC_r1, CRC_r2, CRC_c1 and CRC_c2 the received CRC and computed CRC values of P1 and P2, respectively. The following method is used to determine which parameter is accepted:

1. If P1 = P2, CRC_r1 = CRC_c1, and CRC_r2 = CRC_c2 then select P1
2. If $P_{r1} \neq P_{r2}$, $CRC_{r1} = CRC_{c1}$, and $CRC_{r2} \neq CRC_{c2}$ then select $P_{r1}$

3. If $P_{r1} \neq P_{r2}$, $CRC_{r1} \neq CRC_{c1}$, and $CRC_{r2} = CRC_{c2}$ then select $P_{r2}$

4. If $P_{r1} = P_{r2}$, $CRC_{r1} \neq CRC_{c1}$, and $CRC_{r2} \neq CRC_{c2}$ then select $P_{r1}$

5. If $P_{r1} \neq P_{r2}$, $CRC_{r1} \neq CRC_{c1}$, and $CRC_{r2} \neq CRC_{c2}$ then compare $P_{r1}$ or $P_{r2}$ and choose the parameter with the smaller hamming distance compared with the parameter used in the last frame.

Rule 1 is chosen if no errors are detected. Rules 2 & 3 apply if the parameter copies do not agree. In this case, the parameter which does not have a CRC error is chosen. Rule 4 assumes the received parameters are correct but both CRCs are erroneous. Rule 5 assumes that one of the parameter copies is correct but both CRCs are incorrect. In this case, the parameter closest to the parameter used in the last frame is chosen. These rules were optimized for a burst noise environment. The CRC values are placed immediately after the symbol as shown in figure 25. This way a burst will hit the symbol and its CRC, as opposed to hitting the symbol and CRC of another symbol.

![Symbol and CRC placement of class 1 symbols.](image)

In addition to the above, the decoding algorithm detects synchronization loss. The frame is considered to be mis-synchronized if it contains 15 or more CRC errors. In such a case, all the parameters from the last frame are used again. This recycling continues for three frames, after which the output is muted.
This coding scheme was chosen primarily because of its simplicity. It is very likely that a more efficient coding scheme could be constructed. Unfortunately, selection of better schemes is made difficult without an appropriate general model for power line communication channels.

**Selection of Parameter Significance**

Allocation of error control bits to compressed speech is a difficult task. Ideally, each bit produced by the compression algorithm would be equally significant. Such speech data would then be best protected from errors using an FEC scheme which reduces the channel BER. Unfortunately, the data stream produced by the voice encoder is not uniform with respect to error significance. In addition, there is a need for symbol error detection which recognizes the periodic bursty nature of channel errors, as well as the need for a simple coding algorithm. Making this task more difficult is the fact that data regarding the perceptual significance of each coded speech symbol was measured in isolation from other parameters effectively assuming that errors occurred only in one parameter at a time (chapter 4.) We have no information on how the combination of several symbol errors would be perceived.

Without direct information on the perceptual effects of errors in several parameters, the following allocation rule emerges: For any bit error rate, the sound quality degradation (caused by errors) will be no less than that produced by errors only in the most perceptually sensitive parameter. This rule is roughly analogous to the rule “A chain is no stronger than its weakest link.” Based on this rule, error control bits were allocated as follows:

1. The MOS degradation curves for all the tested parameters (given in chapter 4) were plotted on the same axis. A representative illustration appears in figure 26a. The probability of symbol errors was adjusted for relative parameter length (number of bits each encoded parameter requires).
2. A target MOS degradation curve is constructed. A representative illustration of this is shown in figure 26a. In our case, the residual parameter degradation curve was selected as the target degradation curve because the residual parameter is less sensitive to noise than most other parameters, and the residual parameter is too large to be error protected.

3. By assigning error control bits, the MOS degradation curves of the parameter will be shifted to the right. The object is to assign sufficient bits to each parameter such that parameter’s degradation curve will lie on the target degradation curve.

4. If the degradation curves of each parameter lie at, or to the right of, the target curve and unused error control bits remain available, then the target curve is shifted to the right and step 3 is repeated.

5. When all the error control bits have been assigned, the composite MOS degradation curve will look approximately like figure 26b.

Step 3 was accomplished by adding either a four-bit CRC, or by duplicating the parameter and the CRC. Adding the CRC provides error protection (a class 2 symbol). Therefore, the degradation curve of a class 2 symbol is replotted using the symbol with error detected scores provided in section 4.2. With the symbol and the CRC repeated (a class 1 symbol), the likelihood that both copies received will be damaged symbols is approximately the square of the probability that one symbol copy and its CRC will be damaged. Thus, a class 1 symbol has a degradation score of a class 2 symbol with the probability of symbol error re-scaled to the square root of its probability of error.

5.3 Parameter Interleaving

Given that the errors on the power line are often bursty and periodic, an appropriate interleaving of the repeated speech parameters used by class 1 symbols greatly reduces the likelihood that both the original
Figure 26 Mean degradation curves before and after error correcting bits are assigned.
and redundant parameters will be hit by a burst error. This section describes parameter interleaving which will minimize the effect of errors on a periodic burst error power line channel.

**Optimum Interleave Distance**

As discussed previously in chapter 3, bit errors are correlated and periodic with a period of $1/120$ seconds. Error correction is facilitated by transmitting repeated parameters in a way which reduces the probability of having an error in both the original and repeated parameter.

Two general categories of periodic channel errors are considered. The simpler case involves transmission on the same phase, as shown in figure 1. Typical error patterns [5] for the same phase case are shown in figure 27a. The more complex case occurs with cross-phase transmission. Typical error patterns for cross-phase transmission situations are shown in figure 27b.

Periodic same-phase errors occur with a frequency of 120 Hz. To minimize the effect of these errors, the repeated parameters are placed $\frac{1}{2} \times \frac{1}{120}$ seconds after the initially transmitted set. This corresponds to exactly 80 full rate bits. Such placements will minimize the number of double parameter errors.

Cross-phase errors are much harder to combat. As shown in figure 27b, two error bursts may be encountered. These burst are centered 60 degrees ($1/360$ seconds) apart. This results in a much wider effective burst period. As with the same-phase transmission, the redundant parameter should be placed $\frac{1}{2} \times \frac{1}{120}$ seconds after the first.

Figure 28 is an example of a cross-phase transmission within the electrical engineering (Macleod) building. The duplicate eight-bit symbol was spaced a distance ranging from zero to 100 bits from the original eight bit symbol. For the lower BERs, the percentage of frames with both parameters damaged (a double error) reaches a minimum with an 80 bit ($1/240$ second) separation. At higher BERs, there is no clear optimum separation which reduces the probability of double errors over cross-phase transmissions.
Compatibility of Symbol Data with Burst Error Channel

In addition to having good properties with respect to leaving a copy of a transmitted parameter value undamaged, the periodic burst errors on the power line also result in fewer symbol errors than a random error channel. Burst errors will create errors in only a few symbols, as opposed to random
errors which may damage many symbols. We can show the difference between a random channel and a burst channel as follows.

Consider a burst error of length \( L \) bits with period \( P \) bits and a symbol of length \( M \) bits. The probability of a symbol error \( P_{se} \) is

\[
P_{se|\text{burst}} = \frac{M}{P} + \frac{L - 1}{P} = \frac{M + L - 1}{P}
\]

Assuming a periodic error burst occurs at 60Hz, and a 19200 baud rate, error bursts will occur every 320 bits. We may calculate an effective error rate as \( \text{BER} = \frac{1}{2P} \), given a single periodic burst of length \( l \) bits with a burst period of \( p = 320 \) bits. Using equation 13, we may calculate the probability of a symbol error. We can now compare the symbol error rate with random errors to the symbol error rate with burst errors. The probability of a symbol error given random errors is

\[
P_{se|\text{random}} = (1 - P_e^M)
\]

A plot of the number of symbol errors as a function of channel BER for both the periodic burst error case and the random error channel indicates a substantially lower symbol error rate with a burst channel.

### 5.4 Comparison of 9600 and 19200 bps Performance

Figures 30, 31 and 32 summarize the subjective quality tests by comparing the 19200 bps and the 9600 bps encoding schemes. These tests measure the subjective quality, as described in chapter 4, of the encoded speech subject to errors caused by the power line errors and random errors.
Figure 29 A comparison of the number of symbols with errors with and without interleaving as a function of bit error rate.

Figure 30 provides a comparison of the MOS of the 19200 bps and 9600 bps encoding schemes subject to random bit errors. This figure provides a comparison of the encoding schemes on the sole basis of channel error rate. Several salient result can be noted from figure 30:

1. All MOS scores are higher for the 19200 bps encoding method, provided BER < 0.10.
2. The 19200 bps encoded voice data was found to be tolerant to at least twice as many errors as the 9600 bps encoded voice data, for the BER range of figure 30.
3. The 19200 bps and 9600 encoded scores fell below 2 (poor) for BER of 3.8x10^{-2} and 1.7x10^{-2}, respectively.

In addition, we observe that the quality (MOS) of the 19200 bps encoded sample reaches its maximum approximately at a BER of 2x10^{-3} where an error free sample scores 4.5. Likewise, the quality of the 9600 bps encoded sample reaches its maximum at approximately a BER of 5x10^{-4}.
Figure 30 A comparison of the half rate and full rate encoding schemes for a random binary symmetric channel

Figure 31 compares the MOS of the 19200 bps coding scheme, when subject to binary symmetric errors as well as to power line errors. These results demonstrate the difference in sound quality between a random noise channel and the periodic bursty power line channel. The main results obtained from figure 31 include:

1. The quality score at BERs below 0.01 (approximately) appear similar for the power line and random error channels. This result probably arises because at low BER the error bursts are short and appear randomly.

2. At higher BERs (above 0.01), the quality scores tend to be higher for power line channels than for
the random error channel. Burst errors tend to corrupt many bits in the same speech parameter, rather than few bits in many parameters.

Figure 31 A comparison of the full rate encoding schemes with a random binary symmetric channel and various power line channels.

Figure 32 presents the results of the quality measurements for the 9600 bps and 19200 encoding schemes with transmission over power line communication links. These results include the effects on channel BER of bit slips, mis-synchronizations and other degrading effects caused by transmission over power lines.
The measurements in figure 32 give no convincing indication as to which method provides better performance. The decrease in channel quality brought on by transmission at 19200 bps (as compared to 9600 bps) has negated the beneficial effects of a more accurate coding scheme. Since both schemes appear to work equally well, we may conclude that the lower rate 9600 bps scheme is more practical because of its simpler implementation. Figure 32 does indicate that for BER $\lesssim 0.03$, MOS for the 19200 bps scheme exceed 2.5. This statement cannot be made for the 9600 bps scheme which shows MOS $\lesssim 1.5$ for BER $\simeq 0.02$.

![Figure 32](image.png)

**Figure 32** A comparison of the half rate (9600 bps) and full rate (19200 bps) encoding schemes over four power line communication channels.
Chapter 6 – Conclusions

6.1 Summary

This thesis presented a system for real time voice transmission of speech over power lines. This system employed synchronous digital transmission, voice compression, and forward error correction. Data transmission across the power line is subject to many disturbances. Subjective testing was performed to determine how to minimize the effect of these disturbances.

The thesis work was divided into two parts. The first involved substantial and essential modifications to the existing power line modems, which were designed to transmit packetized data. The modifications necessary to transmit compressed speech required the insertion of periodic synchronization markers into the compressed data, as well as the design and employment of a real-time algorithm to perform the synchronization and to detect its loss. Converting the modems to transmit synchronous data required determination of a suitable marker sequence, distance between markers, and other parameters required by the synchronization routine. In addition, the modem hardware had to be enhanced to prevent bit slip, which was a major cause of lost synchronization. The synchronization algorithm was designed and modified to perform on channels with a bit error rate as high as 0.1.

The second area of study involved determination of an effective method to encode and protect speech data. This work required information on which components of the compressed speech frame were sensitive to errors. Subjective listening tests were used to obtain the required information. In these tests, errors were added to individual speech parameters, and human subjects were asked to rate the degradation in speech quality. Using this information, two schemes were created to protect speech data. The data rate of the VSELP encoder is 7950 bps. Given the selectable modem data rates of 9600 and 19200, two
coding schemes were created. For the 9600 bps case, only a small amount of bandwidth was available to implement forward error correction. A CRC was used to determine whether one of the 12 most significant bits were corrupted; if corruption occurred the previous frame was repeated. For most parameters, the degradation caused by errors was reduced when error detection was employed and the parameter from the previous frame was substituted for the corrupted parameter. This requirement for parameter protection provides motivation for the use of a symbol-by-symbol protection scheme. In the case of 19200 bps transmission, a complex encoding strategy was devised which placed each of the symbols into one of three groups. The first group required no error correction. The second group was provided with only error detection. The third group received both error detection and correction. The categorization for each symbol was chosen on the basis that the combined degradation of speech quality caused by errors in many speech parameters would be dominated by parameters which were most sensitive to errors.

Subjective testing was used to determine the best error control scheme. Under random noise tests with equal bit error rates, the 19200 bps encoding scheme tolerated 2 to 3 times more bit errors at a given MOS score than the 9600 bps scheme, depending on the channel bit error rate, as expected. The question was “Would the greater number of channel errors encountered with a 19200 bps modulation scheme (as opposed to 9600 bps) offset the coding gains?” The answer was yes. Tests on power line channels yielded similar speech quality for both data rates. The simplicity of the lower rate technique suggests that it is the preferred choice. However, for BER \( \approx 0.02 \) it appears that 19200 bps is the preferred choice.

6.2 Future Work

Future work towards voice over power line can be grouped into the areas of constructing a complete real-time system, advances in speech coding, and advances in error control coding.
The release of a single-chip VSELP voice codec announced in April 1993 for use in commercial
digital cellular phones enables the construction of a complete real-time communication system. This
chip contains the digital and processing components required for two-way voice communication, and
was designed as part of a chip set. Adapting the present modem design to work with this chip would
be straightforward. The most involved step of the adaptation would be de-interleaving the output from
the chip.

By replacing the microcontroller with a more powerful digital signal processor, the VSELP coding
and modem control could be performed with a single chip. With the use of a single chip modem, a
complete speech transceiver could be constructed with only a few chips. This hardware implementation
would likely be time intensive because it involves re-implementing the VSELP algorithm.

One other hardware option is to redesign the modulator/demodulators of the power line modems.
If cellular phone standard, $\pi/4$ shifted differentially encoded quadrature phase modulation at 13200 bps
was used, then the cellular phone VSELP coding chip could be used directly. This would include the
FEC coding used for cellular phones. This would not be an optimal implementation, but it may be
cost effective.

Two schemes which may enable full duplex communication across power lines include using time
division duplex or frequency division duplex to divide the power line channel into two synchronous
channels. Experiments towards time division duplex were conducted by the author and this scheme was
found workable with minor modifications to the modem transmitter circuitry. The frequency division
alternative assigns different carrier transmission frequencies to each of the modems.

Reed Solomon or BCH codes could replace the simple code repetition scheme used. Any coding
gains would directly improve sound quality over noisy channels. An optimal scheme which could take
into account the bursty periodic errors, as well as graceful degradation of voice quality in the presence of errors would have to be determined.

Voice coding algorithms are continually developing. Presently, voice coding chips with half the data rate of the VSELP chip have been proposed. These chips are still under development, but similar parameter sensitivity experiments could be conducted with these chips. Lower bit rate codes imply less redundancy and therefore reduce the likelihood that symbol repetition would work well. These reduced rate codes may require additional error protection. Use of a different voice coding scheme would, of course, require a suitable and probably new data protection scheme.
Bibliography


pp 561–79, April 1975.


Appendix A  Subjective Test I Instruction Sheet

Subjective Test Instructions

Thank you for taking part in this speech listening test. In this test you will be asked to subjectively rank the quality of various speech samples by assigning a score from 1 (poor quality) to 5 (good quality) for each sample.

Procedure

1. Review the good quality and poor quality references by using the computer menu or by pressing the 'g' or 'p' keys. Listen to the references as many times as you require to familiarize yourself.
2. Listen to the speech sample by using the menu or by pressing the 's' key.
3. Rate the speech sample from 1 to 5 using the menu or the keys '1' to '5'. Rate samples equal to or better than the good reference as 5. Rate samples equal to or worse than the poor reference as 1. Rate the other samples between 2 and 4 as you see fit. You may listen to the speech samples and the references as many times as required to make your rating.
4. Repeat steps 1 & 2 until all the speech samples have been rated.

   It may be difficult to rank some samples, but please try your best. Thanks.

Rating Scale

<table>
<thead>
<tr>
<th>Score</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Equal or Better than &quot;good&quot; sample</td>
</tr>
<tr>
<td>4</td>
<td>Slightly worse than &quot;good&quot; sample</td>
</tr>
<tr>
<td>3</td>
<td>Significantly worse than &quot;good&quot; sample</td>
</tr>
<tr>
<td>2</td>
<td>Slightly better than &quot;poor&quot; sample</td>
</tr>
<tr>
<td>1</td>
<td>Equal or Worse than &quot;poor&quot; sample</td>
</tr>
</tbody>
</table>
Appendix B  Subjective Test II Instruction Sheet

Subjective Test Instructions

Thank you for taking part in this speech listening test. In this test you will be asked to subjectively rank the quality of various speech samples by assigning a score from 1 (unacceptable quality) to 5 (excellent quality) for each sample.

Procedure

1. Sit in front of computer with headphones on.
2. Listen to the speech sample by pressing the 's' key or by using the menu.
3. Determine the sample's sound quality value as per the "quality scale" below. You may listen to the speech sample as many times as required to make your rating.
4. Record the sound quality value of the speech sample from 1 to 5 by pressing the keys '1' to '5' or by using the menu.
5. Repeat steps 2 – 4 until all the speech samples have been rated. Sample selection and record of your rating will be performed automatically by the computer.

   It may be difficult to rank some samples, but please try your best. Thanks.

Quality Scale

<table>
<thead>
<tr>
<th>Value</th>
<th>Sound Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Excellent</td>
</tr>
<tr>
<td>4</td>
<td>Good</td>
</tr>
<tr>
<td>3</td>
<td>Fair</td>
</tr>
<tr>
<td>2</td>
<td>Poor</td>
</tr>
<tr>
<td>1</td>
<td>Unacceptable</td>
</tr>
</tbody>
</table>
Appendix C  Firmware Listings

; Synchronous Communication Demo Program
; Written by Rick Dorbolo
; Aug. 1993
;
; File: PLV009.ASM
; Written in Intel 8051 assembly code
;
; hardware addresses
.equ PORT, 0
;
; ascii characters
.equ CR, h'0D
.equ LF, h'0A
.equ ESC, h'1B
.equ SYNCH, h'CC
.equ SYNL, h'AA
.equ SOPKTH, h'9D
.equ SOFKTL, h'61
.equ H_S0P1, h'37
.equ H_S0P2, h'38
.equ H_E0P1, h'a3
.equ H_E0P2, h'a4
.equ EVEN, h'02
.equ ODD, h'01

.opdef TX_BITS, P1.0 ; to modulator or encoder
.opdef PM_CLK, P1.1 ; clock to set parameters
.opdef MOD_ENC, P1.2 ; modulator=1, encoder=0
.opdef RELAY_CONTROL, P1.3 ; 1 = rx, 0 = tx
.opdef DEC_OUT, P1.4 ; from decoder
.opdef CTS, P1.5 ; to DTE
.opdef CLK_SET, P1.6 ; set tx&rx clocks
.opdef RTS, P1.6 ; request to send*
.opdef PARAM, P1.7 ; param(2C1B1B0) shift reg

.opdef TxD_DTE, P3.0 ; from DTE
.opdef RxD_DTE, P3.1 ; to DTE
.opdef RX_BITS, P3.2 ; rx bits from demod
.opdef BIT_SYNC, P3.3 ; bit synchronization
.opdef BAUD_24, P3.4 ; baud x 24
.opdef TX_DATA_CLK, P3.5 ; to DTE
.opdef SYNCHASYNC, P3.6 ; sync or async
.opdef DCD, P3.7 ; data carry detect

; DATA MEMORY assignments
.segment .data
.data
.org 0x20

FLAG:  rs 1 ; storage for flags
.equ BAUD0, (FLAG - 0x20) + 0
.equ BAUD1, (FLAG - 0x20) + 1
.equ C1, (FLAG - 0x20) + 2
.equ C2, (FLAG - 0x20) + 3
.equ CODEC, (FLAG - 0x20) + 4 ; 0=no; 1=yes
.equ ESCAPE, (FLAG - 0x20) + 5
.equ XEDOFF, (FLAG - 0x20) + 6
.equ TEST, (FLAG - 0x20) + 7 ; 0=no; 1=yes (2*dclk)

FLAG2: .rs 1 ; additional storage for flags
.equ MODE, (FLAG2 - 0x20) + 0 ; 0=async/1=sync
.equ DSTATE, (FLAG2 - 0x20) + 1 ; temp state flag

; transmit related constants and data
.equ BUF_SIZ, 40
.equ XON_LMT, (BUF_SIZ * 1 / 4)
.equ XOFF_LMT, (BUF_SIZ * 3 / 4)

.state: .rs 1
zeros: .rs 1
buffer: .rs BUF_SIZ
dmemory: .rs 1
in_sr: .rs 1
invert: .rs 1
decoded: .rs 1
r_state: .rs 1
temp: .rs 1
sfrm_lth: .rs 1 ; Subframe Length
sf_cmtr: .rs 1 ; Current Subframe
frm_reps: .rs 1 ; Number of subframes
distance: .rs 1 ; Random walk distance

.code
.org 0

START:
  ajmp INIT

.org 0x40

INIT:
  ; set up stack pointer
  ; note: stack is only 32 bytes deep (0x60 to 0x7f) !!!
  mov SP, #0x60

  ; reset the bits that control the modem hardware
  mov A, 0xFF
  mov P0, A
  mov P1, A
  mov P2, A
  mov P3, A

  ; set up the 8051's internal counters
  ; t0 : integrate and dump counter for rx bits
  ; t1 : used as baud rate generator for serial port in async operation
  mov TMOD, #b'00101101
  mov TL0, #0
  mov TH0, #0
  setb TNO ; turn on timer/counter 0
  clr RI
  setb IT1 ; bit sync in INT1
; This routine test bi-directional synchronization

; simple test routine for retrieving a byte from p-line
; and sending up to host (no codec)
Synchronization Routine

Register bank utilization

- \( R_0 \) == input shift register
- \( R_1 \) == synchronization state 0 == searching,
  \( 1 \leftrightarrow 4 \) How many markers have been found
- \( R_2 \) == loop counter variable
- \( R_3 \) == result of correlation lookup
- \( R_4 \) == last Marker
- \( R_6.7 \) == non-differential decoded input bit
- \( R_5 \) == syndrom of last Marker
- \( R_7 \) == loop counter register

.equ MARKER, b'01100101'; This is the interframe marker
.equ DIFMKR, b'00100110'; This is the differential marker

; ************************************
; * Routine 1stpkt *
; ************************************

pktlst:

    mov R1, #0 ; Initialize R1 to zero ie. searching
    clr TX_BITS ; Send Zeros to codec
    setb CLK_SET ; Synchronize CODEC clock (somehow ?)
    clr CLK_SET

; This initial part looks for the first synchronization flag
SYl:

    jnb R1, syl_r1 ; Has the host send up a character ?
    acall getch ; get character
    cjne A, #'I', syl_r2 ; Is it an initialize
    djnz r_state, syl_r1 ; Is r_state now 0
    ajmp init ; Yes it is, now reset

SYl_R2:

    mov r_state, #4

SYl_R1:

    acall getbit ; Get the next bit from the powerline shift into A and R0
    mov A, R0
    mov R4, A ; Store last marker in R4
    mov DPTR, #covar ; Search the correlation table for contents of A
    movc A, R9A + D PTR
    mov R3, A ; Store correlation result in R3
    anl A, #b'00111111' ; Check to see if byte is marker?
    cjne a, #b'00100000, SYl ; Have we found the Marker with no errors
    mov A, R6
    xrl A, #DIFMKR ; Determine correlation with differently encoded marker
    mov D PTR, #onebits
    movc A, R9A + D PTR

84
anl A,#b'00001100 ;if correlation is 4 of getter assume not inverted
cjne A,#0,inverts ;is correlation result greater then 4
mov invert,#0 ;Yes set invert to non invert
sjmp SYA
inverts:
mov invert,#255 ;Set invert to invert

;Now the initial marker has been found we can start to tx the first frame
SYA:
mov R1,#1 ;Start R1 off at 1
ajmp SYC ;Now go get frame

*******************
* get_frm Routine *
*******************
s_get_fr:
acall getbit ;Throw these away
acall getbit
acall getbit
acall getbit
acall getbit
acall getbit

s_g_f_c:
clr TX_BITS ;Send Zeros to codec
clr CLK_SET ;

;Now we must try to resync, the next byte should be a Marker
mov B,#b'00000000 ;Initial the DPSK state

acall getbit ;Get a bit for unknown reasons

acall bytesimp
mov R0,A
mov R4,A ;store last marker in R4
mov A,R6
xrl A,#DIFMKR ;determine correlation with differently encoded marker
mov DPTR,#onebits
movc A,0A + DPTR
anl A,#b'00001100 ;if correlation is 4 of getter assume not inverted
cjne A,#0,invt2 ;is correlation result less then 4
mov invert,#0 ;Yes set invert to no invert
sjmp cont
invt2:
mov invert,#255 ;Set invert to invert

cont:
mov A,R4

85
mov DPTR,#covar  ;search the correlation table for contents of A
movc A,\$A + DPTR
mov R3,A  ;store correlation result in R3
anl A,\$b'00111111' ;check to see if byte is marker?
cjne a,\$b'00100000',SY4
inc R1  ;it is! now inc R1 upto 8 and go get next frame
mov a,R1
cjne a,distance,SY3
dec R1

SY3:
clr CTS
mov SBUF,\#'M'  ;DEBUG SEND CONSTANT TO HOST
ajmp SYC

SY4:
mov A,R3  ;The Marker has been corrupted! perform action specified by table
anl A,\$b'00000011' ;Store the syndrom in R5
cjne R5,\$b'00000000',SY4A  ;Has a single error occurred?

;Yes a single bit error has occure, let pretend it didn't happen
inc R1  ;it is! now inc R1 upto 8 and go get next frame
mov a,R1
cjne a,distance,SY4AA
dec R1

SY4AA:
clr CTS
mov SBUF,\#'m'  ;DEBUG SEND CONSTANT TO HOST
ajmp SYC

SY4A:
    dec R1  ;Reduce the synchronization score
cjne A,\$b'00000001',SY5; See if Marker has been shift one to the left
mov A,R4
mov R0,A
clr CTS
mov SBUF,\#'L'  ;DEBUG SEND CONSTANT TO HOST

cjne R1,\#0,SYC_LEFT
    ajmp syn_rst  ;I guess sync has been lost go do bitwise search.

SY5: cjne R5,\$b'00000010',SY6 ;Are we one bit ahead?
sccall getbit  ;We are one bit ahead so get a bit to discard
clr CTS
mov SBUF,\#'R'  ;DEBUG SEND CONSTANT TO HOST

cjne R1,\#0,SYC
    ajmp syn_rst  ;sync has been lost go do bitwise search.

SY6:  ;Assume then that the marker by corrupted but not shifted
clr CTS
mov SBUF,\#'S'  ;DEBUG SEND CONSTANT TO HOST

cjne R1,\#0,SYC
    ajmp syn_rst  ;sync has been lost go do bitwise search.

;***********************
;* ROUTINE GET FRAME *
;This repeats code from SYC SYB
;NOTE changes in that code will have to be reflected here

SYC_LEFT:
ACALL GETBIT ;DEBUG TRY TO GET ANOTHER BIT
jnb CODEC,SYC1ALEF ;Is Codec enabled
acall delay11 ;Yes, Wait for 19200 phase correction
acall delay11
acall delay11
acall delay11
acall delay11

SYC1ALEF:
jnb BIT_SYNC,SYC1ALEF ;Wait for rising edge (for CODEC synchronization)
clr CLK_SET ;Synchronize CODECs clock
jnb CODEC,SY2_LEFT ;Is Codec enabled
mov r7,#8 ;Yes, Get initializing first 9 half rate bits do not send to host.
acall getbit ;Decreasing r7 from 9 to 8 will force bitcodc to miss 2 bits
acall delayll ;and calling getbit bit will get one of them

SYCILEFT:
acall bitcodc ;Get bits
djnz r7,SYCILEFT
acall getlbyte ;get the next byte
sjmp SY2ALEFT ;Bypass non-codec medician

SY2_LEFT:
acall get7bit ;Get 7 bits
SY2ALEFT:
mov r2,sfrm_lth ;Common Codec & non codec initialization stuff
mov sf_cntr, frm_reps ;initialize the loop center
sjmp SYD ;Initialize subframe counter
;Rejoin Loop

SYC:
ACALL GETBIT ; DEBUG TRY TO GET ANOTHER BIT
jnb CODEC,SYCIA ;Is Codec enabled
acall delay11 ;Yes, Wait for 19200 phase correction
acall delay11
acall delay11
acall delay11
acall delay11
SYCIA:
jnb BIT_SYNC,SYCIA ;Wait for rising edge (for CODEC synchronization)
setb CLK_SET ;Synchronize CODECs clock
clr CLK_SET
jnb CODEC,SY2 ;Is Codec enabled :
mov r7,#9 ;Yes, Get initializing first 9 half rate bits do not send to host.
mov r7, zeroes ;Yes, Get initializing first zeroes half rate
;bits do not send to host.

SYC1:
acall bitcodc ;Get bits
djnz r7,SYC1

SY2:
mov sf_cntr, frm_reps ;Initialize subframe counter
SY2A:
    mov R2,sfrm_lth
    ;initialize the loop counter
SY2B:
    acall getbyte
    ;get the next byte
    mov SBUF, A
    ;send the byte to host
SYD:
    mov SBUF, A
SYE:
    cjne R2,#16,SYF
    mov a,state
    cjne a,#'r',SYF
    setb CTS
    ;set the CTS line.
SYF:
    djnz R2,SY2B
    djnz s_f_cntr,SY2A
    jnb CODEC,SYG
    acall bitcodc
    clr TX_BITS
SYG:
    ret
    ;loop to get next subframe
    ;loop to get next subframe
    ;Is Codec Enabled ?
    ;Yes, Eat the following bits
    ;We have just received a frame and sent it to host

;******************************
;Receive Subroutines *
;******************************

rx_mode:
    ;Setup to modem for rx
    setb psw.3
    ;setup mc to use bank 01 for R0 to R7
    clr psw.4
    setb RELAY_CONTROL
    ;set relay to rx position
    ret

getbit:
    ;Get a bit from the powerline and place into R0 and A
smpl_sln:
    jnb IE1, smpl_sln
    ;wait for falling edge
    mov A, TLO
    ;read length of pulse
    mov TL0, #0
    ;clear counter
    clr IE1
    ;clear edge indicator
    mov DPTR, #bit_tab
    movc A, @A + DPTR
    ;map sum into proper output
    rrc A
    ;bit in C, confidence in A
    mov a,R0
    ;place the next bit frm C into the shift register and A
    rrc a
    mov dmemory, A
    xrl a,b
    ;decode
    mov R0.a
    ;store result in shift register
    mov b, dmemory
    ;Update state
    anl b, #b'10000000
    mov a, dmemory
    ;These lines store the non-encoded data in in_sr
    rlc a
    ;a shift register R6
    mov a,R6
    rrc a
    mov R6,a

88
ret

getbyte:
    mov R7,#8
byl1:
    call getbit
    jnb CODEC,byla
    xrl A,invert
    rlc A
    call delayll
    mov TX_BITS,C
    mov c,DEC_OUT
    mov A,decoded
    rrc A
    mov decoded,A
    call getbit
    xrl A,invert
    rlc A
    call delayll
    mov TX_BITS,C
    mov c,DEC_OUT
    mov A,decoded
    jmp byl3
byla:
    mov a,RO
byl3:
    djnz r7,byl

ret

bitcodc:
    call getbit
    xrl A,invert
    rlc A
    call delayll
    mov TX_BITS,C
    mov c,DEC_OUT
    mov A,decoded
    rrc A
    mov decoded,A
    call getbit
    xrl A,invert
    rlc A
    call delayll
    mov TX_BITS,C
    mov c,DEC_OUT
    mov A,decoded
    ret

;This PROCEEDURE gets 2 bits from the powerline and places them into the codec.

;Get a byte from the power line and place it in A, and R0.

;Is the codec enabled

;Invert bit if required

;Send the last bit to codec

;Send another bit to codec

;Invert bit if required

;Timing adjustment

;Send the last bit to codec

;Place the decoded output into variable decoded

;Send another bit to codec

;End of bitcodc
bytesimp: ;Get a byte from the power line and place it in A, and R0. 
    mov R7,#8
    simp:
    acall getbit
    mov a,R0
    djnz r7,simp
    ret

get7bit: ;Get next seven bits and place it in A and R0.
    mov R7,#7
    byl2: acall getbit
    djnz r7,byl2
    ret

;Calling this PROCEEDURE produces an 11 machine cycle delay

delay11:
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    ret

; end of Procedure delay11

; Using the above scheme the prob of a false detect = 1/256
; the prob of a second false detect is 2/156 + 16/256

; ******************************************************
; * BI-DIRECTIONAL TIMING DIAGRAM *
; ******************************************************

;The CTS line tells the host which state the modem is in
;CTS LINE 1 -----------------------------
;  0 ------
;:modem TTTTTTTTTTTTTTTTTTTTTTTTTTT
;:action s123 22s1234 22s1
;
;:Host TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
;:action 123 2 12345 1121
;
; R
; T == Transmit
; R == Receive
; W == Wait
;
; ***************
; RECEIVE SECTION
; ***************
;
; Each frame is taken asynchronously from the host and buffered
;
; CTS LINE 1 -----------------------------
; 0 ------- -----------------------------
; SENT BY HOST 1 2 3 4 5 20 21
; RX BY MODEM FR HOST 1 2 3 4 19 20 21
; SEND BY MODEM SSSSSS11111122222233333444444... 19992000211111
;
; RI line 1 XXXXX - - - - - - - - - - -
; 0 XXXXX---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- 91
s_rst2:
cjne a,\#'r',s_rst3          ; Are we in Continuous Receive Synchronous mode
ajmp syn_r
s_rst3:
ajmp syn_s                   ; We must be in Continuous Send Synchronous Tx mode

syn_a:
mov D PTR,\#slook_a
acall puts
acall tx_mode
acall syncro
acall send_frm
mov R1,\#1
loop_a:
acall rx_mode
acall s_get_fr
acall tx_mode
acall send_frm
ajmp loop_a

syn_b:
mov D PTR,\#slook_b
acall puts
acall rx_mode
acall delay
acall delay
acall rx_mode
acall pktlst
loop_b:
acall tx_mode
acall send_frm
acall rx_mode
acall s_get_fr
ajmp loop_b

syn_r:
mov D PTR,\#slook_r          ; Tell host searching for Marker
acall puts
acall rx_mode
acall delay
acall delay
acall rx_mode
acall pktlst
loop_r:
acall s_g_f_c
jnb RI, syn_r1        ; Has the host send up a character?
acall getch           ; get character
cjne A, \#'I', syn_r2  ; Is it an initialize
;;; dec r_state
djnz r_state, syn_r2  ; Yes, now increase r_state
djnz r_state, syn_r1  ; Is r_state now 0
ajmp init            ; Yes it is, now reset

syn_r2:
mov r_state,\#4
syn_r1:
ajmp loop_r

Are we in Continuous Receive Synchronous mode
We must be in Continuous Send Synchronous Tx mode

Tell host searching for Marker
Send a syncro and get a next frame
Start R1 off at 2 for debugging

Tell host searching for Marker
This should get rid of any bit texed for host

Tell host searching for Marker

6=8 cycles

Tell host searching for Marker

Has the host send up a character?
Get character
Is it an initialize
Yes, now increase r_state
Is r_state now 0
Yes it is, now reset
syn_s:
  mov DPTR,#slook_s
  acall puts
  jnb CODEC,syn_s_b
  mov DPTR,#c_enbld
  acall puts
  sjmp syn_s_c

syn_s_b:
  mov DPTR,#c_dnbld
  acall puts
  syn_s_c:
  acall tx_mode
loop_s:
  acall s_s_f_c
  clr cts
  ajmp loop_s

s_s_f_c:
  setb CTS
  ajmp frm_srt
  mov a,#0
  add a,sfrm_lth
  mov RO,a
  mov R1,a
  mov sf_cntr,frm_reps
  clr RI
  setb MOD ENC
  clr B
  acall fill_buffer
  mov a,#0
  zerol:
  jnb TX_DATA_CLK, zerol
  xrl A,B

send_frm:
  setb CTS
  wb1a:
    jnb TX_DATA_CLK, wb1a
    setb TX_BITS
    wb1b:
      jb TX_DATA_CLR,wb1b
      ;send a 1 bit over the line ;wait for high
      ;wait for low
wb2a:
  jnb TX_DATA_CLK, wb2a
  clr TX_BITS
  wb2b:
    jb TX_DATA_CLR,wb2b
    ;send a 0 bit over the line ;wait for high
    ;wait for low
frm_srt:
  mov a,#buffer
  add a,sfrm_lth
  mov R0,a
  mov R1,a
  mov sf_cntr,frm_reps
  clr RI
  setb MOD ENC
  clr B
  acall fill_buffer
  mov a,#0
  zerol:
  jnb TX_DATA_CLK, zerol
  xrl A,B

;Tell host in Continus send mode
;Turn on flow, you now have 8/34k seconds to read next Character
;Send a frame
;Send a frame
;Send a frame
;Turn on flow, you now have 8/34k seconds to read next Character
;Send a frame
rrc A ;move next bit to carry
mov TX_BITS, C ;output the bit
mov B.0,C ;Store the output B.0

zero2:
    jb TX_DATA_CLK, zero2 ;wait for low

sendsync:
    mov A, #101 ;1st character to tx is FLAG
    mov R7, #8 ;initialize R7 for an 8 bit byte
    acall fill_buffer

stxsubl:
    jnb TX_DATA_CLK, stxsubl ;wait for high
    xrl A,B ;Differential Encode Bit
    rrc A ;move next bit to carry
    mov TX_BITS, C ;output the bit
    mov B.0,C ;Store the output B.0

stxsub3:
    jb TX_DATA_CLK, stxsub3 ;wait for low
    acall fill_buffer ;try to fill buffer after each bit
    djnz R7, stxsubl ;go do the next bit

;Flag sent
;Send a 0 after flag
    acall fill_buffer
    mov a, #0
zero3:
    jnb TX_DATA_CLK, zero3 ;wait for high
    xrl A,B ;Differential Encode Bit
    rrc A ;move next bit to carry
    mov TX_BITS, C ;output the bit
    mov B.0,C ;Store the output B.0

zero4:
    jb TX_DATA_CLK, zero4 ;wait for low

jnb CODEC,txbpas1 ;Is Codec Enabled
clr TX_BITS ;Yes, Wait one full rate bit

txwait1:
    jnb TX_DATA_CLK, txwait1 ;wait for high

txwait2:
    jb TX_DATA_CLR, txwait2 ;wait for low
; Activate Codec if required
txbpsa1:
        mov A, @R1
        dec Rl
        jnb CODEC, loopbyte
        setb CLK_SET
        clr MOD_ENC
        sjmp loopbyte

loop_s_f:
        mov a, #buffer
        add a, frm_lth
        mov Rl, a
        mov A, @R1
        dec Rl
        ; Get 1st byte
        ; decrease the output pointer
        ; Are we using the encoder ?
        ; Yes, Enable CODEC clock and sync it
        ; enable CODEC
        ; Set output pointer to beginning of data
        ; Get 1st byte
        ; decrease the output pointer

loopbyte:
        mov R7, #8
        acall fill_buffer
        ; initialize R7 for an 8 bit byte

loopbit:
        jnb TX_DATA_CLK, loopbit
        ; wait for high
        jb CODEC, txsub2
        ; Is the CODEC disabled ?
txsub2:
        xrl A, B
        ; Yes, Differential Encode Bit
        txsub2:
        rrc A
        mov TX_BITS, C
        mov B.0, C
        ; move next bit to carry
        ; output the bit
        ; Store the output B.0

        txsub3:
        jnb TX_DATA_CLK, txsub3
        ; wait for low
        acall fill_buffer
        ; try to fill buffer after each bit
        jnb CODEC, txsub4
        ; Are we using the CODEC

        txsub3a:
        jnb TX_DATA_CLK, txsub3a
        ; Yes CODEC in place now wait
        ; for another high
        acall fill_buffer
        txsub3b:
        jb TX_DATA_CLK, txsub3b
        ; Wait for a low

        txsub4:
        djnz R7, loopbit
        ; go do the next bit

; At this point check to see whether or not to turn on flow from host this
; should occur if 1 - We have just txed 2nd from last byte of subframe
; 2 - We are not transmitting the last subframe of a frame

        cjne Rl, #buffer+2, txsub4a2
        ; 1 - Have we just Txed 2nd last byte
        djnz sf_cntr, txsub4a1
        ; 2 - Are we on the last subframe
        ajmp txsub4a2
        ; We have not enabled flow. Continue

95
txsub4a1: ; We have just txed 2nd last sub-frame now turn on flow and reset
; buffer input pointer

setb CTS
mov a,#buffer
add a,sfrm_1th
mov R0,a

; Set input pointer to beginning data

txsub4a2:
mov A,R1
dec R1
cjne R1,#buffer-1,loopbyte
inc sf_cntr
djnz sf_cntr,loop_s_f

; We have reached the end of the frame buffer ?
; Yes we have, now get
; ready to send next subframe or frame

inc sf_cntr
djnz sf_cntr,loop_s_f

; Are we done frame ?

setb CTS
mov R0,a

; Yes, finish up frame

txsub4a4:
jnb CODEC,txsub5
mov R7,0

; Is Codec Enabled ?
; Yes, Send half rate zero bits

mov R7,.zeros

; Frame sent

fill_buffer:

mov temp,a
jb R1,step4
ret

; store a for deration of subroutine
; Is there a character waiting
; No, then return .. this better
; not happen getting 1st character

step4:
mov A,SBUF
clr R1
mov R0,a

; put character into buffer and A
; reset character waiting flag

clr R1
mov R0,a
cjne R1,0,step4a

; Is it an initialize

dec r_state
djnz r_state,step5
ajmp init

; Yes it is, now reset

step4a:
mov r_state,#8

; Last Character was not
; an I so reset r_state

step5:
cjne R0,#buffer,#step6
clr CTS

; Is the buffer full to end of subframe/frame ?
; Yes it is. So stop flow

mov a,temp
ret

; Yes it is. So return

step6: dec R0
mov a,temp
ret

; reduce input buffer pointer
; return to pre-subroutine state

96
; ***********************
; * Subroutine tx_mode *
; ***********************

tx_mode:
clr RELAY_CONTROL ; 1 cycle set relay to Tx position
clr psw.3 ; 1 cycle Select register bank 00
clr psw.4 ; 1 cycle
setb MOD_ENC ; 1 cycle bypass encoder
ret ; 2 cycles
; 6 = total cycles

; ***********************
; * Subroutine syncro - send a synchronization burst *
; ***********************

syncro:

mov R6, #10 ; initialize loop to send 10 '10101010' bytes
syncrol:
mov R7, #8 ; initialize R7 for an 8 bit byte
mov A, #h'AA ; put 10101010 symbol into A
syncro2:
jnb TX_DATA_CLK, syncro2 ; wait for high
rrc A ; move next bit to carry
mov TX_BITS, C ; output the bit
syncro3:
jb TX_DATA_CLK, syncro3 ; wait for low
djnz R7, syncro2 ; go do the next bit
djnz R6, syncrol ; have we send the 10 bytes
ret

; ***********************
; * Procedure s_0_cd - send a zero to codec for a half rate bit *
; ***********************

s_0_cd:
acall fill_buffer

s_0_a:
jnb TX_DATA_CLK, s_0_a ; wait for high
mov b, #0 ; Clr the DPSK memory
clr TX_BITS ; Send a zero to codec
s_0_b:
jb TX_DATA_CLK, s_0_b ; wait for low
acall fill_buffer ; try to fill buffer after each bit
s_0_c:
jnb TX_DATA_CLK, s_0_c ; wait for another high

97
acall fill_buffer
s_0_d:
  jb TX_DATA_CLK, s_0_d ; Wait for a low
ret ; End of s_0_cd

; **********************
; * Barry's Subroutines *
; **********************

; wait for commands:
; ESC to initialize the modem,
; anything else to run the modem
; '\x00' is ignored

cmdwait:  mov  A, #CR ; send out a <CR><LF>
          acall echo
          acall getch ; wait for a character
          acall toupper ; convert to upper case
          ljmp chksync ; TEST line jump right away to sync
chkquest: cjne A, #'?', chksync
          mov DPTR, #helpinfo
          acall puts
          sjmp cmdwait
chksync:  cjne A, #'S', chkq ;
          acall getch
          cjne A,#'A',chksyn1 ; Are we in mode A
          mov state,#'a'
          mov DPTR,#init_a_m
          acall puts
          ljmp syn_rst
chksyn1:  cjne A,#'B',chksyn2 ; Are we in mode B
          mov state,#'b'
          mov DPTR,#init_b_m
          acall puts
          ljmp syn_rst
chksyn2:  cjne A,#'R',chksyn3 ; Are we in mode R
          mov state,#'r'
          mov DPTR,#init_r_m
          acall puts
          ljmp syn_rst
chksyn3:  ; We must be in mode S
          mov state,#'s'
          mov DPTR,#init_s_m
          acall puts
          ljmp syn_rst
chkq:
cjne A, #'Q', chkk
acall getch
mov sfrm_lth,a
acall getch
mov frm_reps,a
acall getch
mov distance,a
ljmp cmd_ret

chkk:
cjne A, #'K', chkb
mov DPTR, #kprompt
acall puts
acall getword
sjmp cmdwait

chkb:
cjne A, #'B', chkp
acall getch
mov FLAG, A
acall getch
mov zeroes, A
acall ch_baud
sjmp cmdwait

chkp:
cjne A, #'P', chkv
mov DPTR, #bdequ
acall puts
mov A, FLAG
acall putch
ljmp cmdwait

chkv:
cjne A, #'V', chkz
mov DPTR, #version
acall puts
ajmp cmdwait

chkz:
cjne A, #'Z', cmd_ret
ljmp INIT

cmd_ret: ret

ch_baud: mov C, C2 ; send C2 to codec
mov PARAM, C
clr PM_CLK
setb PM_CLK
mov C, Cl ; send Cl to codec
mov PARAM, C
clr PM_CLK
setb PM_CLK
mov C, BAUD1 ; send BAUD1 to mux
mov PARAM, C
clr PM_CLK
setb PM_CLK
mov C, BAUD0 ; send BAUD0 to mux
mov PARAM, C
clr PM_CLK
setb PM_CLK ; CODEC bypass set by host
ret

; ------- puts -----------------------------------------------
; Prints the zero terminated string pointed to by DPTR to the DTE.
; -------
puts:  clr  A
movc A, @A + DPTR ; get next character
inc DPTR ; advance pointer
jz ps2 ; jump out if end-of-string
acall echo ; send out characters
sjmp puts ; do it again
cpl CTS ; Toggle CTS to see if it works
ps2:  ret

; ---------- putch -----------------------------------------------
;
; Prints the character in A to the DTE.
;
-----------------------------------------------
putch:  jnb TI, putch ; wait for transmit buffer empty
clr TI ; clear interrupt flag
mov SBUF, A ; send character
ret

; ---------- echo -----------------------------------------------
;
; Prints the character in A to the DTE expanding <cr> to <cr><lf>.
;
-----------------------------------------------
echo:  acall putch ; send the character
cjne A, #CR, echol ; check for cr
mov A, #LF ; if so, send a lf
acall putch
mov A, #CR ; restore A (A was CR)
echol: ret

; ---------- getch -----------------------------------------------
;
; Waits for a character from the DTE then gets it and returns it in A.
;
-----------------------------------------------
getch:  jnb RI, getch ; loop until receive buffer is full
clr RI ; clear receive interrupt flag
mov A, SBUF ; get received character
cpl CTS ;
ret

; ---------- printword ---------------------------------------------
;
; Prints the word pointed to by R0. Note that MSByte is stored in the byte
; pointed to by R0 and the LSByte is stored in the byte pointed to by R0+1.
;
---------------------------------------------
printword: mov A, 0R0 ; get MSByte
acall printbyte ; print it
inc R0 ; point to LSByte
mov A, 0R0 ; get it
acall printbyte ; print it
ret

; ---------- toupper ---------------------------------------------
;
; Converts the letter in A to upper case.
;
---------------------------------------------
toupper:   cjne A, #'a'+3, ; compare with 'a'
          jc   tul  ; quit if lower
          cjne A, #z'+1, *+3 ; compare with 'z'
          jnc  tul  ; jump if higher
          add  A, #A' - 'a' ; convert
          ret

tul:   ; --------------- getdigit -----------------------------
        ; Reads a single hex digit from the serial port. Returns the value in
        ; A. C will be set if CR was pressed.
        ; -----------------------------------------------
        getdigit:  acall  getch  ; read a character
                  acall  toupper  ; convert to upper case
                  cjne A, #CR, notcr  ; check for CR
                  setb  C  ; return with C set
                  ret
        notcr:   cjne   A, #0', *+3  ; try again if < '0'
                  cjne   A, #9'+1, *+3  ; try for letter if > '9'
                  jnc   gd1  ; display
                  acall  echo  ; subtract offset
                  add   A, #-0'  ; subtract offset
                  simp  gd2
        gd1:    cjne   A, #A', *+3  ; try for a letter
                  cjne   A, #F'+1, *+3
                  jnc   getdigit
                  acall  echo
                  add   A, #-A'+10
                  simp  gd2
        gd2:    clr   C  ; return with C clear
                  ret
        ; --------------- getbyte -----------------------------
        ; Reads a 2 digit hex number from the serial port. Returns the value in
        ; A. C will be set if CR was pressed.
        ; -----------------------------------------------
        getbyte:  mov   B, #0  ; assume 0
                   acall  getch  ; try to get a digit
                   jc   gb1  ; jump out if CR pressed
                   mov   B, A  ; store in B
                   acall  getch  ; try to get a digit
                   jc   gb1  ; jump out if CR pressed
                   xch   A, B  ; combine two bytes
                   swap  A
                   orl  A, B
                   clr  C  ; return with C clear
                   ret
        gb1:   mov   A, B  ; retrieve partial result
                   ret  ; return (C set)
        ; --------------- getword -----------------------------
        ; Reads a 4 digit hex number from the serial port. Stores the value at
        ; the location pointed to by R0. Note that the MSByte is stored at the
        ; low address.
        ;
; --- getword: clr A ; clear out the value
mov @R0, A
inc R0
mov @R0, A
mov B, #4 ; initialize digit counter

; now we must shift the 2 byte value left by 4 bits and combine
xch A, @R0
swap A
xch A, @R0
xchd A, @R0
dec @R0
xch A, @R0
swap A
xch A, @R0
xchd A, @R0
inc @R0
djnz B, gw1 ; go do the next digit

; ----------- delay ---------------
; Delays execution by 512 clock cycles. Destroys R0.

; ----------- delay -------------
delay: mov R0, #0
d1: djnz R0, d1
ret

; ***************
; * CONSTANT STRINGS *
; ***************

hello: .db "19200 BPS Intra-Building Power Line Modem (PLV009DEMO, OCTOBER, 1993)\r\n"
version: .db "Intra-Building Power Line Modem (PLV009DEMO, OCTOBER 1993)\n"
kprompt: .db "Enter start of packet pattern> \0"
bdequ: .db \"\rBAUD = \0\n"

helpinfo: .db \"\r19200 BPS Intra-Building Power Line Modem. Command Summary\"
   .db \"\rB - Enter baud rate\n   .db \"\rP - Print baud rate\n   .db \"\rS - Enter sync word\n   .db \"\rQ - Print user settable parameters\n   .db \"\r? - Print help info\n   .db \"\rZ - Initialize\n   .db \"\rESC I - User mode\n   .db \"\r\n"
slook_a: .db \"Sync-Lost h=a \0\n}slook_b: .db \"Sync-Lost h=b \0\n"
This is a table of the values to be output to port 1 according to the number of bits in the early/late values. Bit 4 gives the decoded bit value and bits 3..0 give the confidence value.

Note that we go to 26 because the bit sync can move to 25 samples.

```
; This table is used for interleaving, i.e. when the preamble is sent

; bit 6 is set to one so that p1.6 can be used as an input port

; number of ones = 0
.db (12 << 1) + 0

; number of ones = 1
.db (11 << 1) + 0

; number of ones = 2
.db (10 << 1) + 0

; number of ones = 3
.db (9 << 1) + 0

; number of ones = 4
.db (8 << 1) + 0

; number of ones = 5
.db (7 << 1) + 0

; number of ones = 6
.db (6 << 1) + 0

; number of ones = 7
.db (5 << 1) + 0

; number of ones = 8
.db (4 << 1) + 0

; number of ones = 9
.db (3 << 1) + 0

; number of ones = 10
.db (2 << 1) + 0

; number of ones = 11
.db (1 << 1) + 0

; number of ones = 12
.db (0 << 1) + 0

; number of ones = 13
.db (1 << 1) + 1

; number of ones = 14
.db (2 << 1) + 1

; number of ones = 15
.db (3 << 1) + 1

; number of ones = 16
.db (4 << 1) + 1

; number of ones = 17
.db (5 << 1) + 1

; number of ones = 18
.db (6 << 1) + 1

; number of ones = 19
.db (7 << 1) + 1

; number of ones = 20
.db (8 << 1) + 1

; number of ones = 21
.db (9 << 1) + 1

; number of ones = 22
.db (10 << 1) + 1

; number of ones = 23
.db (11 << 1) + 1

; number of ones = 24
.db (12 << 1) + 1

; number of ones = 25
.db (12 << 1) + 1

; number of ones = 26
.db (12 << 1) + 1

; number of ones = 27
```

The number of bits in the early/late values. Bit 4 gives the decoded bit value and bits 3..0 give the confidence value.

This table is used for interleaving, i.e. when the preamble is sent.
at half of the bit rate. It is similar to the one above.
Note that we go up to 52 because the bit sync can move to 25 samples

bit_tab_2:
.db (24 << 1) + 0; number of ones = 0
.db (23 << 1) + 0; number of ones = 1
.db (22 << 1) + 0; number of ones = 2
.db (21 << 1) + 0; number of ones = 3
.db (20 << 1) + 0; number of ones = 4
.db (19 << 1) + 0; number of ones = 5
.db (18 << 1) + 0; number of ones = 6
.db (17 << 1) + 0; number of ones = 7
.db (16 << 1) + 0; number of ones = 8
.db (15 << 1) + 0; number of ones = 9
.db (14 << 1) + 0; number of ones = 10
.db (13 << 1) + 0; number of ones = 11
.db (12 << 1) + 0; number of ones = 12
.db (11 << 1) + 0; number of ones = 13
.db (10 << 1) + 0; number of ones = 14
.db ( 9 << 1) + 0; number of ones = 15
.db ( 8 << 1) + 0; number of ones = 16
.db ( 7 << 1) + 0; number of ones = 17
.db ( 6 << 1) + 0; number of ones = 18
.db ( 5 << 1) + 0; number of ones = 19
.db ( 4 << 1) + 0; number of ones = 20
.db ( 3 << 1) + 0; number of ones = 21
.db ( 2 << 1) + 0; number of ones = 22
.db ( 1 << 1) + 0; number of ones = 23
.db ( 0 << 1) + 0; number of ones = 24
.db ( 1 << 1) + 1; number of ones = 25
.db ( 2 << 1) + 1; number of ones = 26
.db ( 3 << 1) + 1; number of ones = 27
.db ( 4 << 1) + 1; number of ones = 28
.db ( 5 << 1) + 1; number of ones = 29
.db ( 6 << 1) + 1; number of ones = 30
.db ( 7 << 1) + 1; number of ones = 31
.db ( 8 << 1) + 1; number of ones = 32
.db ( 9 << 1) + 1; number of ones = 33
.db (10 << 1) + 1; number of ones = 34
.db (11 << 1) + 1; number of ones = 35
.db (12 << 1) + 1; number of ones = 36
.db (13 << 1) + 1; number of ones = 37
.db (14 << 1) + 1; number of ones = 38
.db (15 << 1) + 1; number of ones = 39
.db (16 << 1) + 1; number of ones = 40
.db (17 << 1) + 1; number of ones = 41
.db (18 << 1) + 1; number of ones = 42
.db (19 << 1) + 1; number of ones = 43
.db (20 << 1) + 1; number of ones = 44
.db (21 << 1) + 1; number of ones = 45
.db (22 << 1) + 1; number of ones = 46
.db (23 << 1) + 1; number of ones = 47
.db (24 << 1) + 1; number of ones = 48
.db (24 << 1) + 1; number of ones = 49
.db (24 << 1) + 1; number of ones = 50
.db (24 << 1) + 1; number of ones = 51
.db (24 << 1) + 1; number of ones = 52
.db (24 << 1) + 1; number of ones = 53
.db (24 << 1) + 1; number of ones = 54

: sync is detected if 0, 1, 2, 3, 13, 14, 15, or 16 bits match sync word
### sync_ok:
```
.db 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1
; 0: too many errors, 1: 0 or 1 error, 2: 0 or 1 error but data inverted
```

### data_ok:
```
.db 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2
; 0: too many errors, 1: 0 or 1 error, 2: 0 or 1 error but data inverted
```

---

This table gives the correlation of the input to the number `b0000 1101`

- The table format is SNVVVVOO
- Where S = 0/1 = Noninvert/Invert
- N = Not used
- VVVV = value of correlation 0 <-> 8
- 00 = 00 = No Rotation Suggested
- 01 = Left Rotation Suggested
- 10 = Right Rotation Suggested
- 11 = More than one error, Rotation unadvised

Nov 16, 1992

With the introduction of DPSK the S field is no longer used

New DPSK flag is the 101 decimal

### covar:
```
.db b'00010011
.db b'00010111
.db b'00010011
.db b'00010111
.db b'00010011
.db b'00010111
.db b'00010011
.db b'00010111
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
.db b'00001011
.db b'00001111
```

---

106
Appendix D  Schematic Diagrams