RESEARCH ON SPOKEN LANGUAGE PROCESSING
Progress Report No. 23 (1999)
Indiana University

A Real Time PC Based Cochlear Implant Speech Processor
with an Interface to the Nucleus 22 Electrode Cochlear Implant
and a Filtered Noiseband Simulation[1]

Adam R. Kaiser and Mario A. Svirsky[2]

DeVault Otologic Research Laboratory
Department of Otolaryngology-Head & Neck Surgery
Indiana University School of Medicine
Indianapolis, IN 46202

[2] The authors make no claim about the suitability or safety of the device described herein for any purpose, nor do they assume any liability for the use of the information about this interface. Individual researchers must evaluate and develop protocols to ensure subject safety according to all applicable laws and guidelines.

A Real Time PC Based Cochlear Implant Speech Processor with an Interface to the Nucleus 22 Electrode Cochlear Implant and a Filtered Noiseband Simulation

Abstract: Cochlear implants are electronic devices that have enabled individuals with severe to profound hearing losses to regain some hearing. Nearly all of those who receive cochlear implants (CIs) regain the sensation of sound. There is, however, substantial variability in speech perception performance among users of cochlear implants (Staller et al., 1997). One factor that may contribute to the individual differences in performance is the speech processing strategy used. Traditionally, researchers implemented experimental real time strategies on specialized digital signal processors (DSPs). This approach requires specialized programming knowledge and consequently can be expensive. In this respect, we have simplified the development of real time processing strategies by using an IBM compatible laptop computer to perform all signal processing, and have implemented only a few interface functions on a DSP. Both the continuous interleaved sampling (CIS) and n-of-m speech processing schemes were implemented on a PC in C++. In addition to encoding and transferring speech cues to a cochlear implant, the processor described can also implement an acoustic noiseband based model of a cochlear implant for presentation to listeners with normal hearing.

Introduction

Cochlear implants are electronic auditory prostheses that have enabled individuals with severe to profound hearing losses to regain the sensation of sound. They perform this function through a process beginning with the reception of an acoustic sound wave and ending in the electronic stimulation of the cochlea. First, a microphone is used to receive sound and to convert it to an electrical representation. Second, in multi-channel speech processors, this analog representation is converted to a more easily manipulated digital form. Thirdly, the speech processor encodes and transforms the digitized speech according to the strategy(s) implemented on a particular device. In general, these strategies use this digitally filtered signal to assign a stimulation level to electrode(s) implanted in the cochlea. The stimulation level and the electrode on which it is to be used are encoded and sent via a radio frequency link to a receiver implanted beneath the subject's skin. The implanted device referred to as a receiver/stimulator decodes the information and finally directs the delivery of an electrical stimulus. The research platform described here performs the functions that are normally performed by the external speech processor and enables a researcher to have complete control over this process. The interface described is specific to the Nucleus 22 electrode cochlear implant system, but it can be readily adapted to other platforms as well.

Improvements in speech processing strategies have resulted in increased speech perception performance for users of cochlear implants (Eddington, 1980; Loeb & Kessler, 1995; Kieffer et al., 1997; Wilson et al., 1991). Inherent in the development of such strategies is the need to evaluate subject performance using the new signal processing strategies and stimulation techniques. The complexity of digitizing sound, processing it, tailoring stimulation for a specific patient, and finally encoding the information in a format compatible with a subject's particular CI device is a formidable task. For some purposes researchers have chosen to evaluate new strategies by processing speech off line, generating files containing information that will be used to stimulate patients at a later date (for one example see Fu & Shannon, 1999). This methodology enables complete algorithm flexibility and relieves the experimenter from implementing a real time system; however, it does not allow the subject to practice using the new processing strategy in conversation, nor does it allow the subject to adjust parameters (such

as volume) in real time. For other purposes, researchers have chosen to implement a real time system on a dedicated DSP platform. A summary of such interfaces can be found in (Eddington et al., 1998). This method also enables complete flexibility and real time parameter adjustments but requires a detailed knowledge of device programming. The platform described here allows both complete algorithmic flexibility and real time parameter adjustment. In addition, since the speech processing algorithms are implemented in C++ on a PC, they are more easily adapted for specific signal processing research applications by less specialized programmers and engineers than required for DSP-based development.

While the bulk of speech processing and patient-specific tailoring can be performed on a personal computer, one cannot avoid implementing functions to communicate with the subjects implanted receiver/stimulator in specialized hardware. For the Nucleus 22 device, only four communication functions are needed to implement a wide variety of stimulation strategies. These functions set the stimulation rate and inter pulse interval, direct a stimulation pulse at a given stimulation level to be sent to a specific electrode, and lastly relay interface status back to the PC. Buy using off the shelf components to the greatest extent possible, we have attempted to simplify the construction of this interface. With the exception of a radio frequency pulse generator, the described interface is implemented in software from off the shelf components.

Materials

Figure 1 depicts a block diagram of the hardware used to implement the processor. The core of the Windows 98 PC based platform is based on a DELL Inspiron 3500 laptop equipped with an Intel 350 MHz PII processor. It has 64 Megabytes of RAM, a 4.5 Gigabyte hard drive, and a sound card capable of implementing DirectX 7.1 sound digitization and sound presentation functionality. In preliminary work, the SoundBlaster Live Value (Model SB4670) has been used for this purpose on a desktop PC. Additionally, a Motorola DSP56309EVM evaluation module was used to drive a radio frequency pulse generator according to the communication protocol required of the Nucleus 22 receiver/stimulator. The DSP and PC communicate via a serial cable between a PC com port and the JTAG/OnCE port on the DSP. This connection is needed during program upload and initiation. Real time communication, however, is maintained through a Domain Technologies, Inc. Host Interface Adapter (DSP563xxEVM Host Port Interface Type B). This adapter allows fast bi-directional communication via the host PC's printer port, which must be configured in EPP mode at address 0x378, and the DSP's host port.
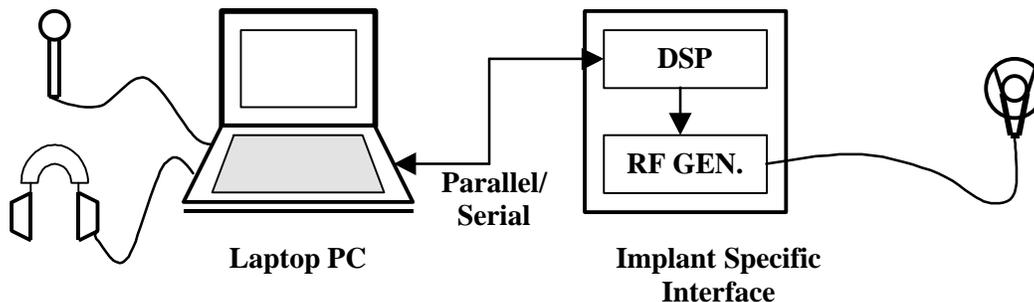

Figure 1: A microphone or alternatively any analog sound source is connected to the PC via the sound card. The PC performs all A/D conversions and DSP routines, leaving the implant-specific interface to generate the RF pulses transferred to the receiver stimulator. The implant-specific interface is initialized via the PC's serial port. After initialization, commands that set the stimulation rate, inter pulse interval, and stimulation parameters are sent via the PC's parallel port. The PC may also request the status of the DSP's internal FIFO buffer to ensure proper stimulation timing.

Several software packages were used to implement the interface. The Microsoft Visual C++ compiler was used in conjunction with the runtime version of the Microsoft DirectX 7.1 software development kit. This software was used to implement all of the speech processing algorithms. Matlab version 5.2.1 was used to calculate filter parameters for the digital filters. Two programs were used to develop, debug and load software on the Motorola DSP evaluation board. Domain Technologies, Inc. EVM563xx version 3.0 software was used to load and monitor the program that implements the implant specific interface on the DSP. The implant specific interface program code was written in assembly language and compiled using the Motorola DSP56300 Assembler Version 6.2.0. PC side communications to the host port adapter were implemented using the interface functions printed in the DSP563xxEVM Host Interface Adapter Reference Manual. No additional drivers were used for this purpose.

Methods

In an effort to keep the PC platform general and to maintain flexibility, several general guidelines were followed during development. First, the interface was designed to allow the easy implantation of the CIS and n-of-m signal processing strategies in addition to developing new ones. Second, researchers without DSP programming experience should be able to modify the programs to suit their own purposes. Third, all speech processing should be implemented on the PC leaving only FIFO (first –in, first- out buffer) maintenance and communication protocol implementation to the implant specific interface. Fourth, the device itself should be the limiting factor in performance, not the interface. Fifth, subject specific information should be loaded from a patient-specific file at runtime.

The resulting platform consists of two main functional sections. The first section to be described is the personal computer on which we implemented of the speech processing algorithms. Each step in the signal processing pathway will be discussed in order from input to output. There are several parameters that may be configured in a configuration file similar to the example in appendix A that is automatically loaded at runtime. These parameters will be referred to in ALL CAPS. When they are specified in the parameter file they must be followed by the appropriate parameter(s) in the following line. Following the description of the PC side algorithms, the functionality of the implant specific interface will then be outlined.

The steps in signal processing performed with this platform are summarized in Figure 2. Signal processing begins by using the sound card to sample the data at 22,050 Hz at 16 bits of resolution. Care should be taken to utilize as much of this dynamic range as possible. The data is then converted to single precision floating point. Pre-emphasis is performed using an N = 1 Butterworth filter with Fc = 1200 Hz. This filter tends to decrease the prominence of the low frequency components of the incoming signal and has been implemented in a variety of research interfaces at this and other institutions (Eddington, 1998; Shannon, 1999). The Fc is set in software and is not yet configurable in the parameter file.

The next step is to perform automatic gain control / dynamic range compression (AGC/DRC). The algorithm used here is based heavily on techniques developed by researchers at MIT (Eddington et al., 1993). It is beyond the scope of this paper to fully describe the operation of this section. Basically, this step normalizes the overall volume of the incoming signal. First, an estimate of the envelope of the incoming signal is made based on a full wave rectified version of the incoming signal. The envelope of the signal is then adjusted based on the previous estimate of the envelope and the current rectified signal. When the rectified signal is greater than the previous envelope estimate, the envelope estimate is increased to the level of the rectified signal at a rate specified by the single time constant AGCATTACK. When the rectified signal is less than the previous envelope estimate, the envelope estimate is decreased to the level of the rectified signal at a rate specified by the single time constant AGCRELEASE. Both constants are specified in seconds in the parameter file. Once the current envelope estimate has been calculated, a gain is selected such that the current envelope estimate multiplied by the gain follows the

piecewise linear function described by the points in DRCTABLE. The values specified in this table are listed as input/output pairs in dB (20*log(amplitude)). They are referenced to a full scale input of zero dB. Full scale is determined by the input to the analog to digital converter. At a minimum, the parameter file must specify an input/output pair for and input of 0 dB and for –100 dB. Pairs should be listed in by decreasing input volume. The output sample from this stage is equal to the input sample multiplied by the gain determined using the relationship above.
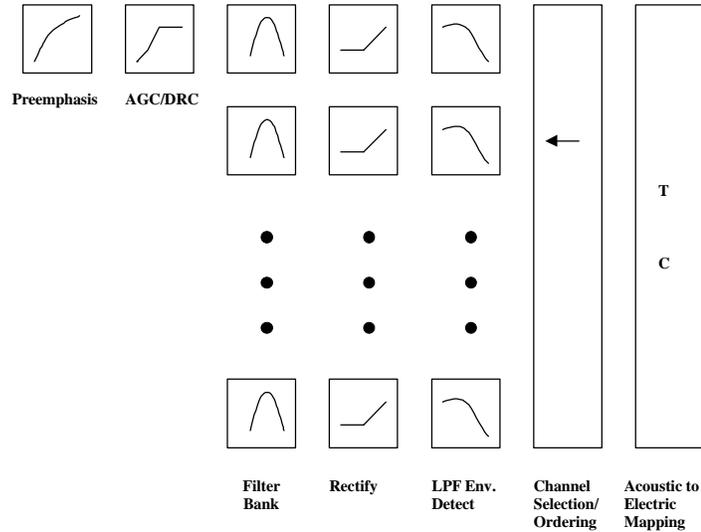

Figure 2: Signal Processing Steps. Data is sampled at 22,050 Hz and 16 bits of resolution prior to conversion to single precision floating point. Preemphasis and AGC/DRC is then performed, followed by frequency-specific filtering and strategy-specific channel selection. The resulting data is scaled logarithmically and mapped to the patient's threshold and comfort levels.

A filter bank with NUMCHAN channels is next used to divide the incoming signal into its corresponding frequency components. These filters are N=4 infinite impulse response (IIR) filters implemented in a single direct form II stage. The characteristics of the filter are specified by the filter coefficients following the words IIRFILTERTAPS in the specification file. An example Matlab program to generate and save the filter coefficients in the correct order appears in Appendix B. The frequency characteristics of the resulting filter are specified by the coefficients listed in the following order for each consecutive channel: $a_0, a_1, a_2, a_3, a_4, b_0, b_1, b_2, b_3, b_4$. These coefficients correspond to the transfer function in equation 1. Note that a_0 must be unity.

$$H(z) = \frac{\sum_{k=0}^{4} b_k z^{-k}}{\sum_{k=0}^{4} a_k z^{-k}}.$$

Equation 1: Filter description for a single channel of an N channel implementation.

Once filtered, the resulting signal is half wave rectified and subsequently low pass filtered using an N = 2 IIR filter with coefficients listed immediately after the IIROUTPUTLPF flag in the following order a_0, a_1, a_2, a_4, b_0, b_1, b_2. This filter is also implemented in a single Direct Form II stage. These coefficients correspond to the transfer function in equation 1. Note that a_0 must be unity in this case as well.

The processing steps up to this point are identical for both the CIS and the n-of-m processing schemes. The following step of channel selection, however, is dependent upon which PROCESSORTYPE is specified in the configuration file. When this parameter equals 1, CIS is selected. When this parameter is equal to 2, n-of-m is selected. CIS simply sequences through each of the channels one at a time and, as will be described further, selects the appropriate electrode and patient-specific stimulation level at which to stimulate. When n-of-m is selected, the channel with the most energy in its corresponding input frequency band is selected from those that have not been stimulated in the last NFORNOFM-1 stimulation cycles. The NFORNOFM parameter defines the number of maxima to be used in the n-of-m stimulation scheme. Since signal processing is performed at 22,050 Hz and the total STIMULATIONRATE is typically 1250 pulses per second, on average, only one in STIMULATIONRATE/22,050 samples streaming from the LPF filters is used in the channel selection process.

The platform we have developed also has the functionality to measure the minimum detectable level (T), and the maximum comfortable level (C). This option can be selected by specifying PROCESSORTYPE = 0 to execute this function. Stimulation levels can be sequentially tried by selecting the appropriate channel and either the T or C level using the left and right arrow keys. The level can be adjusted by using the up and down arrow keys. A one half second stimulation at the rate specified by the STIMULATIONRATE will be delivered to the corresponding electrode with each press of the F12 key. Care must be taken when setting the stimulation rate. For example, creating a map for use with a 20 channel n-of-m processor with 6 maxima requires a stimulation rate of 250Hz during T and C adjustment but a rate of 6*250=1250Hz when actually using the n-of-m processing scheme. One must correctly adjust STIMULATIONRATE to reflect the maximum rate that can be expected on any one channel for each speech processing scheme.

The last step in the PC portion of signal processing maps the amplitude of the channel selected during the channel selection stage to an appropriate simulation level. The numbers immediately following the ELECTRODETANDC flag in the specification file define the mapping parameters for each channel in a single row. The parameters specify the electrode, mode, T level, C level, acoustic minimum, and finally an acoustic maximum in that order. The electrode simply refers to the actual electrode on the implanted receiver/stimulator. The mode refers to the mode of stimulation. For common ground stimulation, a mode of 0 should be selected. For bipolar stimulation a mode of 1 should be selected. For bipolar + 1 stimulation the mode should be 2. The remaining modes follow in this pattern. Both the T and C levels specify the amplitude and phase duration of the stimulation pulses according to the 6.40b stimulation level specification.[3] The acoustic minimum specifies the acoustic level of the low pass filter (LPF) output in dB (20*log(filteroutput)) that will cause stimulation at the T level. The acoustic maximum specifies the acoustic level which will cause stimulation at the C level. Stimulation above the C level is prevented, but stimulation at levels as low as T/2 are simply extrapolated from T and C in conjunction with the acoustic maximum and minimum. If a stimulation pulse is specified for a simulation level below T/2 a level of T/2 is presented to maintain power to the implant.

There are several controls other than T and C adjustments that the experimenter may make in real time. First, F1 and F2 respectively decrease or increase the volume of the signal in increments of 5% of

---

[3] For a complete specification contact Cochlear Corporation, 61 Inverness Drive East, Suite 200, Englewood, CO 80112 U.S.A.

each electrodes dynamic range. This is accomplished by adjusting the C level by the fraction of the dynamic range between the T and the C level specified by the volume. For example, a volume of .75, a T level of 100, and a C level of 200 would result in an effective C level of 175. Second the acoustic value that is to be mapped to the T level can be increased with F4 and decreased with F3. This control has the effect of increasing the dynamic range of acoustic signals mapped to electrical stimulation, and therefore volume, with each press of F4. The converse is true for F3. The last control is a sensitivity adjustment that maintains the range of acoustic levels mapped to electrical stimulation but changes both the acoustic maxima and minima simultaneously. F7 effectively decreases the volume by increasing the acoustic maxima and minima while F8 does the opposite. All of these controls operate across all of the channels simultaneously. Only the T and C level adjustments are made on a channel by channel basis.

While this program is primarily intended to implement speech processing strategies for users of cochlear implants, it can also be used to generate a filtered noiseband simulation of these strategies for listeners and researchers with normal hearing. Rosen (1999) and others have used similar models in their research. The first five steps in the CI speech processor and the filtered noiseband simulation are equivalent. The processes differ beyond the low pass filters as diagramed in Figure 3. The remaining steps begin by using the channel selected during the scheme specific channel selection process. If the algorithm has chosen to stimulate a particular channel, the amplitude of the LPF output is sampled and used to modulate a stream of white noise. This same amplitude sample is used to modulate the amplitude of the white noise for a period of time equal to 1/STIMULATIONRATE in the case of CIS, or NFORNOFM/STIMULATIONRATE in the case of the n-of-m processing strategy. If the channel is not selected after this time, then its amplitude is set to 0. The next stage filters the streaming periods of noise and/or silence using the identical filters specified in the input filterbank stage. The resulting output is then summed with the output from the remaining channels and streamed to the sound card as output.
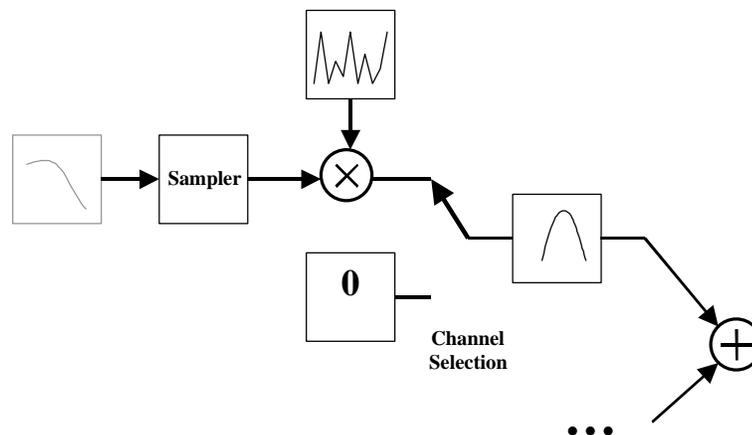
Figure 3: Filtered noiseband simulation. The input to the filter bank for each channel is based on white noise modulated with a sampled version of the LPF output. This sample is updated at a rate based on the simulation strategy and simulation rate.

The second functional component of the platform is implemented in the implant specific interface. After the PC has processed the acoustic speech signal and has determined the appropriate electrode and stimulation level to use, the signal is then sent via the parallel port to the implant specific interface. The following four function prototypes describe the functions that have been developed to facilitate the transfer of information and timing.

1.void Nuc22SendStimPulse(unsigned int elect, unsigned int mod, unsigned int amplitude, unsigned int phase)

As implied by its name, this function sends stimulation information to the interface. The electrode is simply the electrode to be stimulated, and the mode is specified as described above. The amplitude specifies the current level, and must be between 1 and 240. The length of each phase of the signal is specified as phase*.4μs and should be between 12 and 1024.

2.void Nuc22SetStimRate(unsigned int rate)

This function sets the pulse rate in Hz. The interface produces pulses at this rate. The end of the second of the biphasic pulse is used as the timing mark. For pairs of pulses that take longer than 1/rate to transmit, the interface will keep track of how far it has fallen behind, and will catch up during the transmission of the subsequent stimulation pulses. Clearly, the average length of time for a pulse to be sent must be less than 1/rate, or the interface cannot keep up, and the stimulation rate will not be met. This is a limitation of the receiver/stimulator and not of the interface.

3.void Nuc22SetIPI(unsigned int ipi)

Nuc22SetIPI sets the length of the inter phase interval as ipi*.4μs. In general this value should be approximately 10μs.

4.unsigned int Nuc22FIFORemaining(unsigned int slots)

This function returns the number of locations left to be filled in the internal FIFO buffers. When streaming from a data file, this function can be used to determine how many pulses can be sent to the FIFO before it overflows. Samples sent to the FIFO when it is full will be discarded.

These basic functions allow the implementation of a variety of stimulation strategies, whether they are implemented from preprocessed stimuli or used by a real time program such as such as the one described here.

Future Directions

The PC based speech processing algorithms described in this report are completely independent of the receiver/stimulator to which they will ultimately interface. Work is underway in our laboratory to interface the PC based signal processing platform described here to other cochlear implant systems.

References

Eddington, D.K. (1980). Speech discrimination in deaf subjects with cochlear implants. *Journal of the Acoustical Society of America*, 68 (3), 885-891.

Eddington, D.K., Rabinowitz, W.M., Svirsky, M.A. & Tierney, J. (1993). Speech Processors for Auditory Prostheses (Fourth quarterly progress report, NIH Contract N01-DC-2-2402) National Institutes of Health, Bethesda, MD: Neural prostheses program.

Eddington, D.K., Garcia, N., Noel, V., Tierney, J., & Whearty, M. (1998). Speech Processors for Auditory Prostheses (Final report, NIH project N01-DC-6-2100). National Institutes of Health, Bethesda, MD: Neural prostheses program.

Fu, Q.J., & Shannon, R.V. (1999). Effects of electrode configuration and frequency allocation on vowel recognition with the Nucleus-22 cochlear implant. *Ear & Hearing*. 20(4), 332-44.

Kiefer, J., Muller, J., Pfenningdorff, T., Schon, F., Helms, J., von Ilberg, C., Baumgartner, W., Gstottner, W., Ehrenberger, K., Arnold, W., Stephan, K., Thumfart, W., & Baur, S. (1997). Speech understanding in quiet and in noise with the CIS speech-coding strategy (MED EL Combi-40) compared to the MPEAK and SPEAK strategies (Nucleus). *Advances in Oto-Rhino-Laryngology, 52*, 286-290.

Lawson, D.T., Wilson, B.S., Zerbi, M., & Finley, C.C. (1996). Speech processors for auditory prostheses (Third quarterly progress report, NIH project N01-DC-5-2103). National Institutes of Health, Bethesda, MD: Neural prostheses program.

Loeb, G.E., & Kessler, D.K. (1995). Speech recognition performance over time with the Clarion cochlear prosthesis. *Annals of Otology, Rhinology, & Laryngology, 166*, (Suppl.), 290-292.

Rosen, S., Faulkner, A., & Wildinson, L. (1999). Adaptation by normal listeners to upward spectral shifts of speech: Implications for cochlear implants. *Journal of the Acoustical Society of America,* 106, 3629-3636.

Staller, S., Menapace, C., Domico, E., Mills, D., Dowell, R.C., Geers, A., Pijl, S., Hasenstab, S., Justus, M., Bruelli, T., Borton, A.A., & Lemay, M. (1997). Speech perception abilities of adult and pediatric Nucleus implant recipients using spectral peak (SPEAK) coding strategy. *Otolaryngology-Head & Neck Surgery, 117,* 236-242.

Wilson, B.S., Finley, C.C., Lawson, D.T., Wolford, R.D., Eddington, D.K., & Rabinowitz, W.M. (1991). Better speech recognition with cochlear implants. *Nature, 352*(6332), 236-238.

Appendix A

```
PROCESSORTYPE                           IIROUTPUTLPF
2                                       1.0000
                                        -1.9496
0 = Ts and Cs                           0.9509
1 = CIS                                  0.3094e-003
2 = n of m                               0.6187e-003
                                        0.3094e-003
NFORNOFM
2                                       IIRFILTERTAPS
                                        1.000000e+000
STIMULATIONRATE                         -3.824137e+000
1250                                     5.498750e+000
                                        -3.523986e+000
NUMCHAN                                  8.494152e-001
6                                        3.075495e-003
                                        0.000000e+000
DOAGCDRC                                -6.150989e-003
0                                        0.000000e+000
                                        3.075495e-003
1 = do both agc and drc                       .
0 = do not do agc or drc                      .
                                              .
AGCATTACK
0                                        1.000000e+000
                                        -3.794601e+000
AGCRELEASE                               5.474094e+000
.250                                     -3.556699e+000
                                        8.786601e-001
DRCTABLE                                 1.963396e-003
  0   0                                  0.000000e+000
  0  -5                                  -3.926793e-003
 -30  -5                                 0.000000e+000
 -50 -50                                 1.963396e-003
-100 -100


ELECTRODETANDC
20 1 115 146 -80 -50
18 1 115 146 -80 -50
16 1 110 135 -80 -50
12 1 105 140 -80 -50
9 1 110 141 -80 -50
6 1 85 112 -80 -50
```

Appendix B

```
%The Following Code generates the IIR bandpass filters
% using Butterworth filters and plots the resulting frequency
% response

clear;
SamplingRate = 22050;
NyquistRate = SamplingRate/2.0;

%Wn specifies the upper and lower frequency bounds in
%the order of each channel
Wn=[[150 555];
  [555 876];
  [876 1387];
  [1387 2190];
  [2190 3446];
  [3446 5500]];

  ylim('manual');
  xlim([.01 11000]);
  ylim([-90 1]);
  hold on

  Wn = Wn./NyquistRate;

for i = 1:6,
  A =1;
  [B, A] = butter(2,Wn(i,:));
  [H F] = freqz(B,A,1000,SamplingRate);
  H = 20*log10(abs(H));
  semilogx(F,H);
  taps = [taps;A(:)];
  taps = [taps;B(:)];
end

%Now to generate the coefficients for the preemphasis filter
[B,A] = butter(1,1200./NyquistRate,'high');
[H F] = freqz(B,A,1000,SamplingRate);
H = 20*log10(abs(H));
semilogx(F,H);

%Now to generate the coefficients for the low pass output filters
[B,A] = butter(2,125./NyquistRate);
[H F] = freqz(B,A,1000,SamplingRate);
H = 20*log10(abs(H));
semilogx(F,H);
hold off;

%Save the coefficients for cutting and pasting into
% the parameter file
fid = fopen('c:\IIRTAPS.txt','w');
fprintf(fid,'%e\n',taps);
fclose(fid);
```