A PATTERN RECOGNITION SYSTEM FOR AMBULATORY EEG

by

Gilbert J.P. ARBEZ

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
MASTERS OF SCIENCE
in
ELECTRICAL ENGINEERING

Winnipeg, Manitoba

A PATTERN RECOGNITION SYSTEM FOR AMBULATORY EEG

BY

GILBERT J.P. ARBEZ

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Gilbert J.P. ARBEZ

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Gilbert J.P. ARBEZ

ABSTRACT

This thesis describes a real time pattern recognition system for prolonged ambulatory EEG recordings. These recordings are played back at 20 times the recording speed. In the long term, this research should lead to a diagnostic aid for the analysis of EEG ambulatory recordings. In the short term, the system will aid in the development of EEG analysis techniques.

Three subsystems, a feature extractor, a terminal classifier, and a syntactic analyzer comprise the pattern recognition system. The terminal classifier assigns a terminal class to each one second EEG epoch using the epoch features generated by the feature extractor. Then, the terminal string produced is merged by the syntactic analyzer to produce a compact string of nonterminals describing the analyzed signal.

The system is composed of two MC68000 educational boards, one completed with an A/D circuit and a hardware multiplier. To provide a flexible system, software was written in a modular form and labels are used to define system parameters. The system analyzes a single EEG channel and can also be used as a development system.

ACKNOWLEDGEMENTS

The author would like to express his appreciation to all who have given the support and assistance in his research. Foremost is Dr. E. Shwedyk who not only provided guidance but much encouragement and support. The author also appreciated working with all the individuals in the University of Manitoba Engineering Biomedical Laboratory. In particular, Eric Brusse was helpful in extracting data from the EEG database. Much help was obtained from the personnel of the Winnipeg Health Science's Children's EEG lab. Dr. P. Jayakar was instrumental in analyzing the recognition system's output. A much appreciated grant from the Sigma Xi, The Scientific Research Society provided funds for hardware purchases. Finally the author would like to thank Monique Dion for her moral support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

I

INTRODUCTION

The electroencephalogram (EEG), a recording of low level electrical signals on the scalp, has proven to be a useful non-invasive tool for the study of neurological problems including sleep staging and maturation of newborn (Fig 1). By visually scanning an EEG printout, signal patterns can be associated with certain phenomena. For example, alpha waves (Fig 1 a) occur when a subject closes his eyes while in a restful state. Rhythms such as sleep spindles (Fig 1 c), are used extensively in sleep research. Epilepsy also can be associated with certain wave shapes. Spike and wave complexes (Fig 1 d) are typical for the petit mal variant of epilepsy.

These EEG signals are monitored with a set of scalp electrodes. A conventional EEG recording confines the patient to a laboratory or hospital bed and records the signal for a brief period of time, lasting 30-45 minutes, approximately 3% of a twenty four hour day. Abnormalities, extremely useful diagnostically and therapeutically, may not be picked up

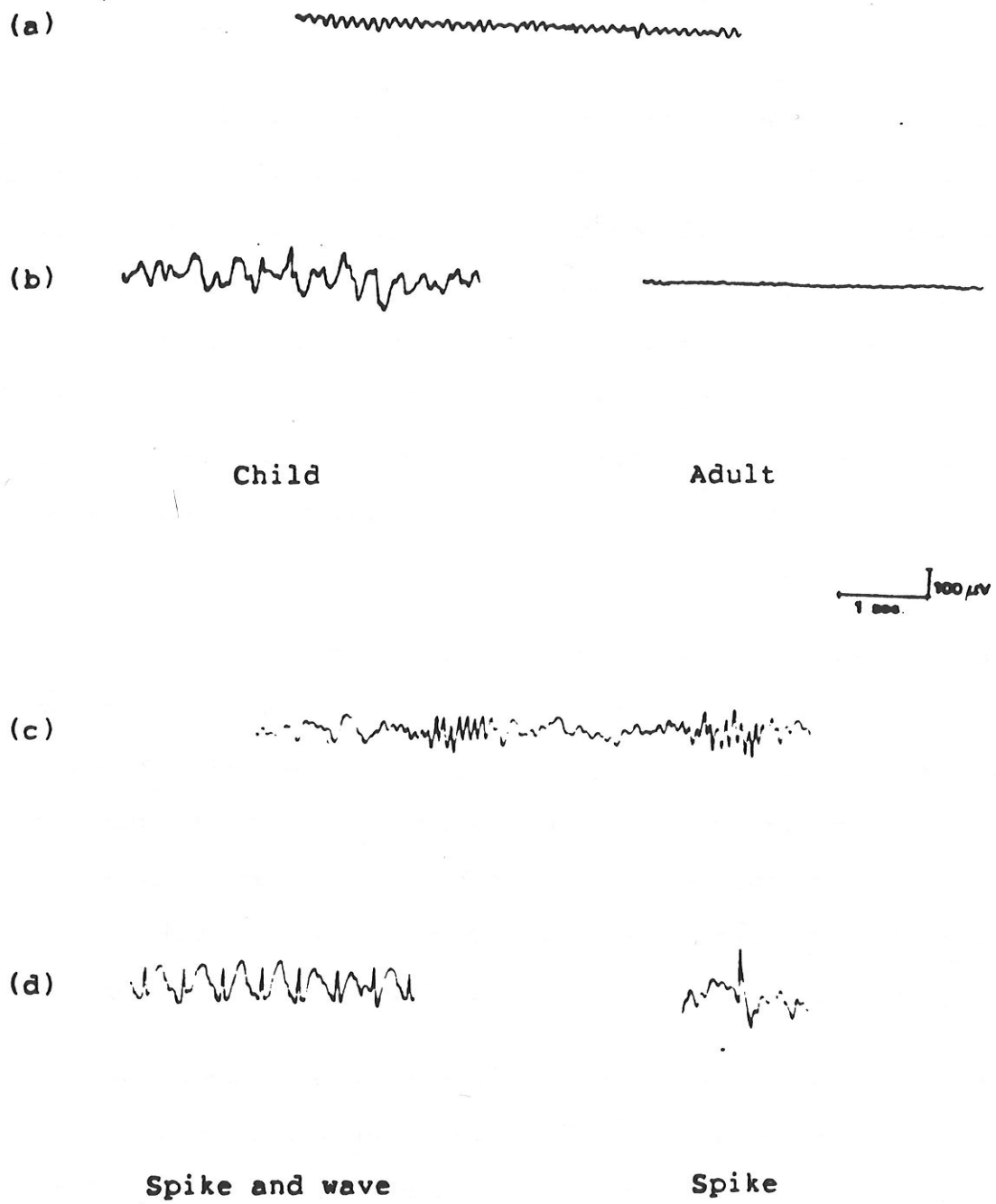

Figure 1: EEG Wave Forms a) alpha wave b) wave of a child and adult both a awake state c) sleep spindles d) epileptic related signals

in the course of such a brief record. In order to free patients from the bed and to obtain longer recordings, much interest and work has been generated for prolonged recording techniques.

Ives and colleagues at the Montreal Neurological Institute have pioneered the use of a 24-hour 4-channel on-patient ambulatory EEG (AEEG) recording system used mainly with adults. A recorder strapped to the patient, fed by a preamplifier (amplifying the EEG signal), records a 24-hour signal on a normal 120 minute cassette (Fig 2) permitting the patient to go about normal activities. The Children's Hospital of Winnipeg are using the system with epileptic children.

Although providing prolonged recording and freedom of movement, the burden of analyzing the 24-hour EEG signal lies on the electroencephalographer (EEGer). The actual playback time (at 20 times the recording speed) is only 1 1/2 hours, however its detailed visual analysis can require from 2 1/2 to 4 hours. Of the entire record, approximately 5% to 10% of the signal is useful for diagnostic purposes.

The EEGer's task is divided in two steps: signal pattern recognition and diagnosis. Signal pattern recognition consist of identifying and classifying the different patterns that comprise the recording. For example, frequency and amplitude of rhythms such as alpha waves are measured. Other

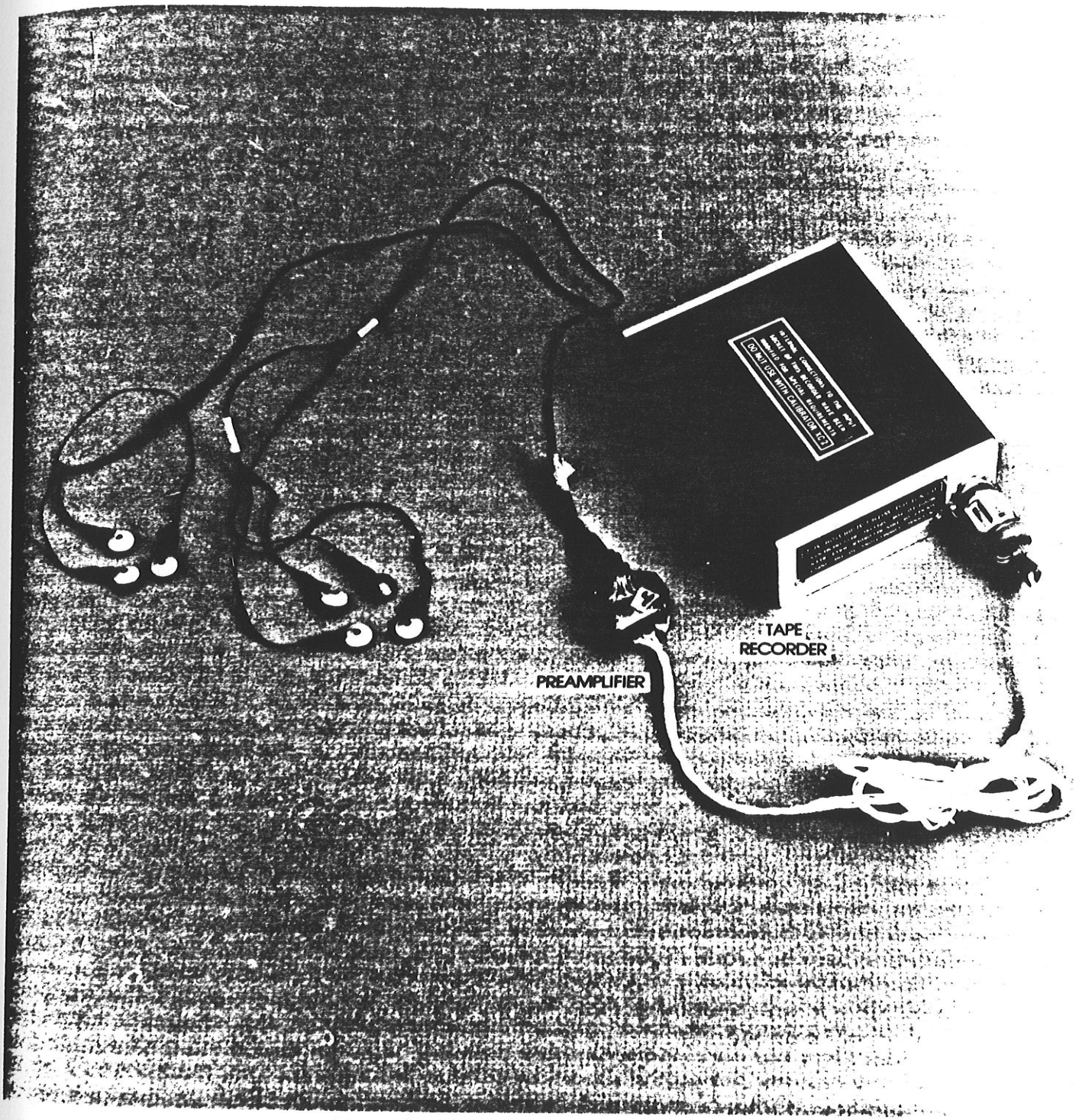

Figure 2: Ambulatory Recording System

signals such as spikes, slow waves, eye blinks are identified. The second task consists of interpreting these patterns to learn about the patient's health state. For example spikes can be interpreted as noise or an indicator of epilepsy. Different stages of sleep are identified using the different signal patterns corresponding to each stage.

The first process, especially in the case of the AEEG, is very time consuming and hence computer analysis for this purpose becomes attractive. Automation of signal pattern recognition will free the EEGer to perform his real job - that of diagnostician. As a first step to computer analysis of the AEEG, Thorne (1) in a preliminary study developed a pattern recognition technique which he divided into two steps: a decision-theoretic analysis and a syntactic pattern recognition analysis.

Decision-theoretic analysis consists of dividing the EEG signal into one second segments (epochs). This epoch size tracks the data well and is short enough to detect any significant transients. A set of discriminating values are first computed to form a feature set. From this feature set, linear discriminant analysis classifies the epoch into one of seven different classes. Once the decision-theoretic technique has classified the EEG epochs into terminals, or primitives, the syntactic pattern recognition analysis merges classifications using a context free grammar. Thirty second records, containing 30 symbols describing each sec-

ond, were collapsed into a few nonterminal symbols representing that record.

The work done by Thorne and the experience gained with the Children's Hospital of Winnipeg served as the ground work for the research presented in this thesis. Two objectives were adopted: 1) To further the research in pattern recognition techniques and 2) To develop a system for real-time implementation of these techniques.

As alluded above, three subsystems comprise the pattern recognition system: feature extraction, terminal classification, and syntactic analysis. Feature extraction is the process of computing from the digitised data a set of discriminating values which are then used by the terminal classifier to classify the one second epoch. Syntactic analysis will reduce the terminal string produced by the terminal classifier to a more concise string.

The feature extractor first computes the power density spectrum (PDS) from 128 samples of EEG data (a one second epoch) using either an autoregressive (AR) model or a known zero autoregressive (KZAR) model. From the PDS and the model coefficients, a discriminant feature set is determined for epoch classification.

The next subsystem, the terminal classifier, uses linear discriminant analysis, to classify the epochs. Classification is performed by a set of decision functions which use

the feature set provided by the feature extractor. The EEG, much like any other human characteristic, is unique for each individual. As an example, in younger patients the signal will be usually larger in amplitude (Figure 1d). Intra-subject variability of the EEG indicates that the ideal set of decision functions is unique for each patient. To account for this variability, an adaptive procedure was implemented. It uses the on-going classifications of the subject record to adjust the decision functions.

The terminal classifier provides a string of EEG classifications. A 24 hour recording provides over 86,000 such classifications. The syntactic analyzer, the third subsystem, identifies patterns within this string to give a more concise description of the recording by collapsing sections of the terminal classifier string output.

The three subsystems must perform their operations within the time window used to digitize the EEG epoch signal. At the increased playback speed of 20 times, each epoch occupies 50 ms (playback time). Playback-time implementation was accomplished using the 68000 processor. It provides 16-bit computational accuracy (32 bit additions) and easily manipulates the 12-bit digitized signal. Two Motorola educational boards, the first one augmented with a wire-wrap board containing a 16-bit hardware multiplier and a single channel analog to digital converter, provide a system for both developing analysis routines and the processing of a

single EEG channel. The additions to the first board provide the speed necessary for the computational load required by the feature extractor. The feature set is dumped from the first board via a parallel port to the second one where terminal classification and syntactic analysis process the feature set.

Modular programming was used to provide a flexible software environment where parameters such as model order, model used (AR,KZAR), algorithms used (BURG,DURBIN), decision functions, or grammars, can easily be changed. The three subsystems are linked only by the data passed from one operation to another. Changes within the routines of each subsystem affects the other subsystems only if the data exchanged is affected. As long as the feature set definition remains the same, changes to the feature extractor remain transparent to the terminal classifier.

Objectives for each of the three subsystems can be summarized as follows:

1. Feature extractor: Provide a real-time feature extractor; verify the effect on pattern recognition for the AR model versus the KZAR model, the BURG algorithm versus the DURBIN algorithm.
2. Terminal classifier: Provide a flexible real-time terminal classifier with an adaptation scheme.

3. Syntactic analysis: Develop a system for flexible grammar definition and for the real-time analysis of a 24-hour EEG record.

Chapter II describes in more detail each of the three subsystems and the algorithms used in performing these tasks. Chapter III takes a look at hardware implementation. Chapter IV presents the system performance, that is comparing the differences between models and algorithms; and verifying the accuracy of the 16-bit system using an Amdahl mainframe as a reference. Recommendations and conclusions complete the thesis in the fifth and last chapter.

II

THE PATTERN RECOGNITION SYSTEM

Chapter I introduced the pattern recognition system which consists of three subsystems: feature extraction, terminal classification, and syntactic analysis. This chapter presents an overview of the subsystems' pertinent theory and their functional descriptions. The three subsystems in the EEG pattern recognition system are implemented in a real-time 68000 based system using modular programming. Chapter III gives an overview of the recognition system hardware and software. A general software and detailed hardware description can be found in the appendices. The 'EEG Pattern Recognition System Software' manual, printed under separate cover, contains software flowcharts and listings.

2.1 THE FEATURE EXTRACTOR

Feature extraction consists of taking a one second epoch of sampled EEG data, and computing a set of features to be used for classification of that epoch. Such a definition is a heuristic approach at determining a suitable discriminat-

ing set for epoch classification. The electroencephalographer in his pattern recognition measures frequency and relative magnitude of the signal for rhythmic signals while signals such as spikes and eye movements are distinguished by their shapes (Fig 3).

The power density spectrum was found to give discriminating values (1). To determine the PDS, the EEG signal is treated as a slow varying non-stationary signal. EEG generation is modelled as an AR process (see Fig 4), with the AR model parameters varying from one epoch to the next. An epoch is chosen to be one second in length. The one second interval, over which stationarity is assumed, is short enough to track the signal and long enough to reliably compute the PDS using autoregressive modelling.

The autoregressive model filter is all pole with the following transfer function

$$H(z) = G \,/\, \left(1 + \sum_{k=1}^{p} a(k)\, z^{-k}\right) \qquad 2.1$$

where G is the filter gain, p is the model order and a(k) are the autoregressive model coefficients. It provides good spectral results from the short record lengths (2). Efficient algorithms exist (DURBIN and BURG) to provide the AR parameters. BURG's algorithm (3) guarantees a stable filter. DURBIN's algorithm (2) is more efficient, but can generate an unstable filter. This instability can be detected during computation.

(a)

(b)

Figure 3: EEG Signal Classifications (a) EEG rhythms (b) Eye blink and spike

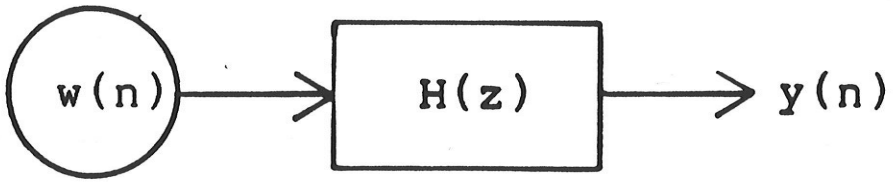

Figure 4: Modelling the EEG Signal: w(n) is a white noise sequence, H(z) typically is an all pole model.

The AR model does not accurately represent the the low frequency region of the EEG power spectrum. The recording equipment introduces a zero which filters out components less than approximately 1 Hz. To account for this zero, the AR model is extended to the known zero autoregressive (KZAR) model by simply adding a zero at dc (i.e.z=1).

$$H_{KZAR}(z) = G(1 - z^{-1}) \; / \; (1 + \sum_{k=1}^{p} a(k)z^{-k}) \qquad 2.2$$

The same methods of determining the parameters of the AR filter are used for the KZAR filter except that the data is first passed through the inverse of the known zero (Fig 5). The signal s(n) is computed from the EEG signal y(n) leaving an AR model.

The feature set is comprised of spectral features determined from the PDS and the AR model parameters. A stepwise discriminant analysis is used to determine which features are discriminatory for epoch classification. Decision functions are also produced from this analysis.

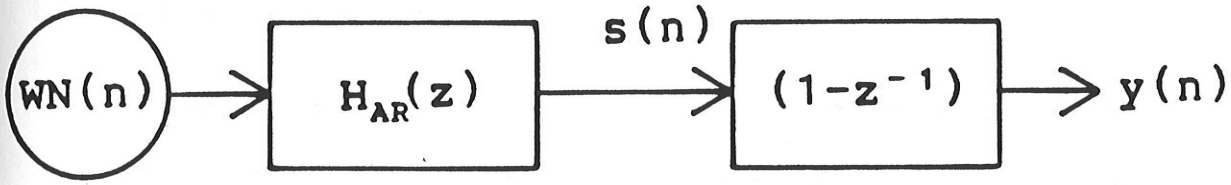

Figure 5: The KZAR Model

2.1.1 Implementation of the Feature Extractor

Fig 6 illustrates the feature extractor system. The A/D converter digitizes the EEG signal at playback time by interrupting the 68000 processor at a rate of 2.56 kHz. A swinging buffer configuration of two 128 word (16-bits) buffers, provides a configuration where one buffer is filled while the other one is being processed.

To eliminate any dc value that may have been inadvertently introduced in the data the mean value of each 128 data array is subtracted out. The mean value is computed by summing the 128 data points using 32-bit arithmetic and then shifting the 32-bit word by 7 positions to the right (corresponds to division by 128).

Next a Hamming window is applied to the data to reduce the effect of the truncation when computing the AR and autocorrelation coefficients. Windowing the data consists of multiplying the data array with stored hamming window coefficients to yield a new data array

$$s_1(n) = s(n)\, w(n) \qquad 2.3$$

where $w(n) = 0.54 - 0.46\cos(2\pi n/127) \quad n=0,1,\ldots,127$ 2.4

are the window coefficients.

For the KZAR model, the data array is passed through the inverse of the known zero before windowing. The array s(n)

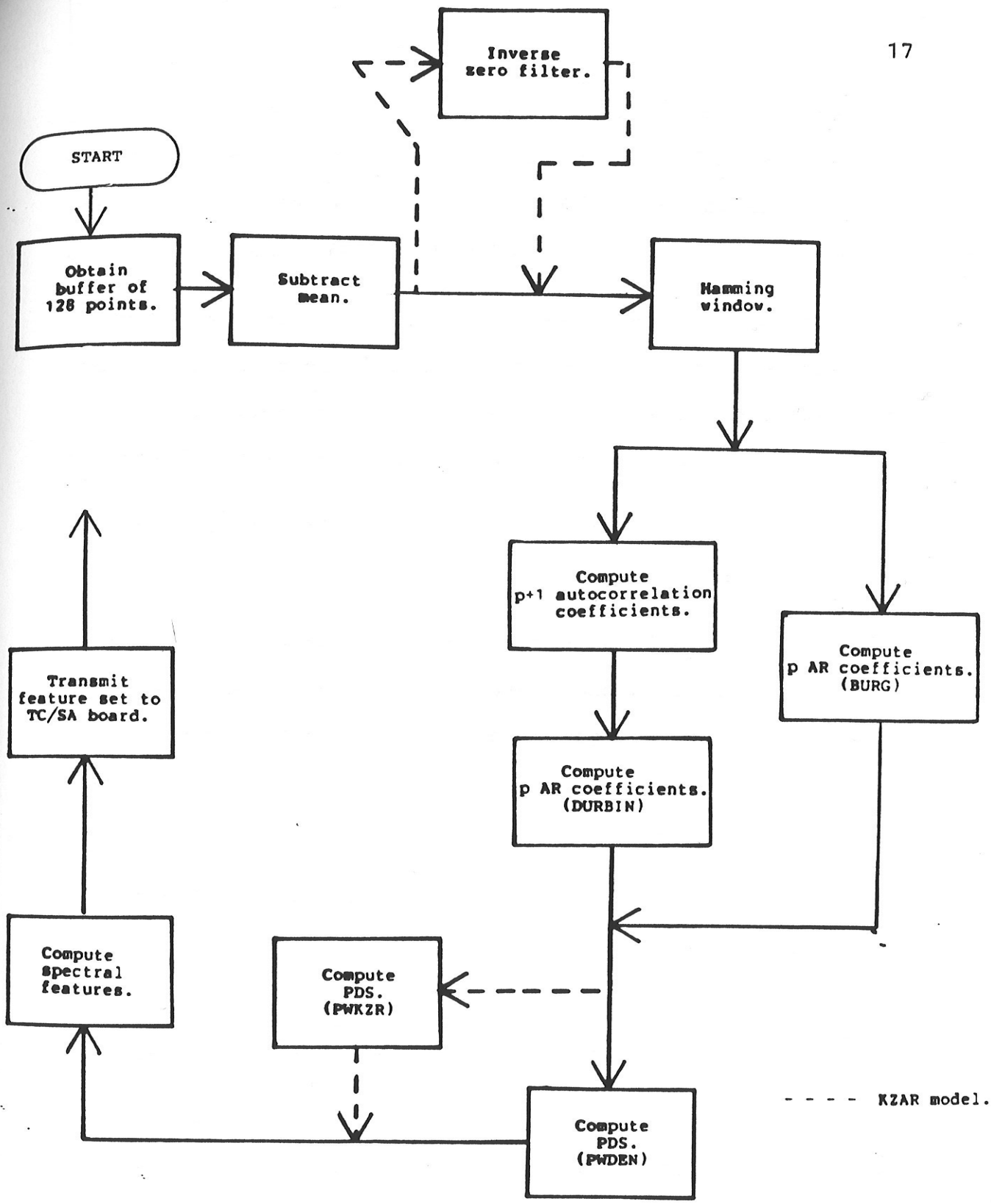

Figure 6: Block Diagram of the Feature Extractor

is computed from the EEG data array, y(n), with its mean removed as shown in Fig. 7. Once the data array has been preprocessed, AR coefficients and G^2 are computed using one of two routines: BURG or DURBIN.

Durbin's algorithm computes p (model order) coefficients from p+1 autocorrelation values. Using a hardware multiplier which provides multiplication and accumulation, the autocorrelation coefficients are estimated using

$$r(v) = \sum_{i=0}^{N-1} s(i)s(i+v) \qquad 0 < v < p \ ; \ N=128 \qquad 2.5$$

These estimated autocorrelation coefficients are then used in a recursive procedure developed by Durbin (2). This recursive procedure is represented by the following set of equations.

$$k = - r(1)/r(0) \qquad 2.6a$$

$$E = (1 - k^2)r(0) \qquad 2.6b$$

$$a(1,1) = k \qquad 2.6c$$

$$k = -\left[r(i) + \sum_{j=1}^{i-1} a(j,i-1)r(i-j) \right]/E \qquad 2.7a$$

$$a(i,i) = k$$

$$a(j,i) = a(j,i-1) + k\, a(i-j,i-1) \qquad 1<j<i-1 \qquad 2.7b$$

$$E = (1-k^2)E \qquad 2.7c$$

$$G^2 = E/\text{scale} \qquad 2.8$$

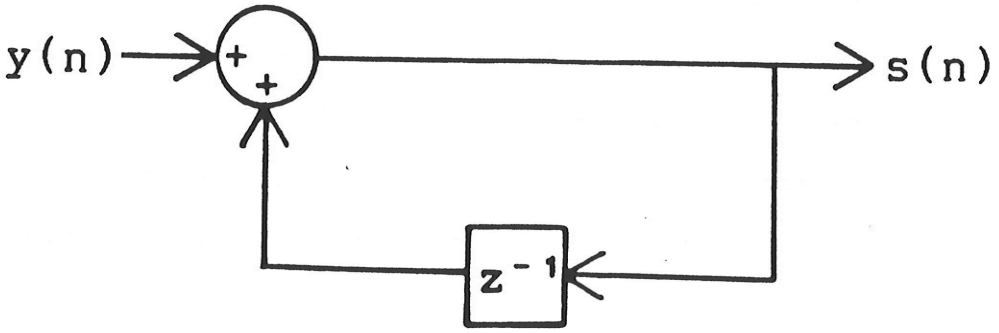

Figure 7: Passing the data through the zero inverse.

Equations 2.6 indicate initialization of variables before recursion is initiated. The partial coefficient k is used to verify stability. During recursion, if k ever exceeds 1 (2.7a), then the filter produced is unstable. The epoch energy, E, is recursively updated by equation 2.7c. The gain G^2 is simply the scaled value of E at the end of the recursive procedure (2.8). The scaling factor allows proper scaling of the PDS to provide significant values. The shape of the PDS and relative scale is of more importance than the absolute value of the PDS.

A matrix a(j,i), (2.7b), is used to compute the AR coefficients (Fig. 8). At each recursion, the coefficients of the model order i are computed from the coefficients of the previous model order i-1 with the initial conditions specified by a(1,1) (2.6c). Thus the iterations are started at i=2 and repeated until i=p. The coefficients are then transferred from the matrix to an array for further processing (2.9).

$$a(j) = a(j,p) \qquad 1 < j < p \qquad 2.9$$

The subroutine DURBIN first calls the subroutine AUTCR1 which computes the autocorrelation coefficients r(k). DURBIN then proceeds to calculate the AR coefficients and G^2 from the results of AUTCR1.

$$a(j,i) = \begin{bmatrix} a(1,1) & a(1,2) & a(1,3) & \cdots & a(1,p) \\ & a(2,2) & a(2,3) & \cdots & a(2,p) \\ & & a(3,3) & \cdots & a(3,p) \\ & & & & a(4,p) \\ & & & & \vdots \\ & & & & \vdots \\ & & & & \vdots \\ & & & & a(p,p) \end{bmatrix}$$

The index i refers to the model order.

Figure 8: Matrix a(j,i) used to compute the AR coefficients

Burg's algorithm (3) computes the AR coefficients directly from the data array and does not require computation of the autocorrelation coefficients as does DURBIN. His criteria was to minimize the average output power of the m+1 prediction error filter with respect to the single parameter a(m,m).

The algorithm starts by initializing two working arrays (b1, b2), the element a(1,1) of matrix a(m,k), and the power P as shown in the following equations.

$$b1 = s(n-1) \qquad 1 < n < 127 \qquad 2.10a$$

$$b2 = s(n) \qquad 1 < n < 127 \qquad 2.10b$$

$$a(1,1) = 2\sum_{n=0}^{N-1} s(n)s(n+1) \Big/ \sum_{n=0}^{N-1} [s^2(n) + s^2(n+1)] \qquad 2.11$$

$$P = \sum_{n=0}^{N-1} s^2(n) \qquad N=128 \qquad 2.12a$$

$$P = [1 - a^2(1,1)]\, P \qquad 2.12b$$

Recursion starts with index m=2 and ends when m=p. Coefficients of model order m are computed using coefficients of the lower model order m-1. Each recursion consists of updating b1 and b2 (2.13), updating the elements of the matrix a(m,k) from b1 and b2 (2.14), and finally evaluating P (2.18).

$$b1 = b1(n) - a(m-1,m-1)b2(n) \qquad 2.13a$$

$$b2 = b2(n) - a(m-1,m-1)b1(n+1) \qquad 2.13b$$

$$0 < n < N-m-1$$

$$a(m,m) = 2\sum_{i=0}^{N-m-1} b1(i)b2(i) \Big/ \sum_{i=0}^{N-m-1} [b1^2(i) + b2^2(i)] \qquad 2.14a$$

$$a(m,k) = a(m-1,k) - a(m,m)a(m-1,m-k) \qquad 2.14b$$

$$P_{i-1} = [1 - a^2(m,m)]\, P_i \qquad 2.15$$

Burg's algorithm produces coefficients whose values differ in sign from that of Durbin's. The results of Burg's algorithm are negated (2.16) to allow the use of the same routine for PDS computation regardless of the routine used to compute the AR coefficients. The filter gain squared is computed using $G^2 = P/\text{scale}$.

$$a(j) = -a(p,k) \qquad 1 < k < p \qquad 2.16$$

$$\gamma(i) = a(i) + \sum_{k=1}^{p-i} a(k)a(k+i) \qquad 2.17$$

$$0 \leq i \leq p \qquad a(0) = 1.0$$

$$P(\omega) = G^2/\{\gamma(0) + 2\sum_{i=1}^{p} (\gamma(i)\cos(i\omega T)\} \qquad 2.18$$

$$\omega = 2\pi f \qquad T = 1/N$$

Using the AR coefficients and G^2 produced by one of the above algorithms, the power density spectrum is evaluated. An efficient algorithm for PDS computation is presented by Makhoul (2). The autocorrelation of the impulse response of the inverse filter $A(z) = 1 + \sum_{k=1}^{p} a(k)z^{-k}$, is evaluated as shown in 2.17 and used to compute the PDS (2.18). The values of $\cos(i\omega T)$ are precomputed and saved for the the bandwidth 0 - 30 Hz and a frequency resolution of 0.5 Hz.

This procedure produces the PDS directly when using an AR model. With the KZAR model another step is required due to the known zero introduced into our modelling. Equation 2.19 shows the evaluation of the PDS for the KZAR model where H_{AR} is the AR component of the KZAR transfer function and H_{KZAR} is the zero $(1 - z^{-1})$.

$$PDS = |H_{KZAR}(z)|^2 = |H_{AR}(z)H_{ZERO}(z)|^2$$
$$= |H_{AR}(z)|^2 |H_{ZERO}(z)|^2 \qquad 2.19$$

The second term can be evaluated as shown in 2.20. The values for $|H_{AR}(z)|^2$ are provided by PWDEN. Equation 2.21 shows the procedure used for evaluating the power density spectrum for the KZAR model.

$$|H_{ZERO}(z)|^2 = |(1 - e^{-j\omega T})|^2$$
$$= |1 - \cos(\omega T) + j \sin(\omega T)|^2$$
$$= \{1 - \cos(\omega T)\}^2 + \sin^2(\omega T)$$
$$= 2(1 - \cos(\omega T)) \qquad 2.20$$

$$P_{KZAR}(\omega) = P_{AR}(\omega)\, 2(1 - \cos\omega T) \qquad T=1/N \qquad 2.21$$

The subroutine PWDEN computes the PDS from the AR coefficients and G^2. PWKZR calls PWDEN before multiplying the results with the factor of 2.21. The PDS is then used by the subroutine FEAT to compute spectral features.

The resultant feature set is composed of :

1) Value G^2

2) AR coefficients

3) Spectral Features

where G^2 and the AR coefficients are provided by DURBIN or BURG. Sixteen spectral features are produced by FEATRS. The EEG spectrum is divided into the six frequency bands 0-1.5 Hz, 2-4 Hz, 4.5-7 Hz, 7.5-12 Hz, 12.5-16 Hz, 16.5-30 Hz. The power in each band along with their ratios to the total power and 4 features related to the spectral maximum (maximum power value, the frequency of this maximum, a value reflecting the slope around the maximum, and the number of spectral peaks) make up the spectral features.

The power in the six bands are computed by summing the power in each of the 0.5 Hz intervals of the PDS (2.22) and their ratio to the total power is then taken to form 6 other features (2.23).

$$P(\omega_i - \omega_j) = \sum_{k=i}^{j} P(k) \qquad 2.22$$

$$P(\omega_i - \omega_j) = \sum_{k=i}^{j} P(k) \Big/ \sum_{k=0}^{60} P(k) \qquad 2.23$$

Four spectral maximum features computed are the maximum power value, the frequency at which this maximum is found, a value proportional to the slope around this maximum and finally the number of spectral peaks found in the PDS.

The slope around each point P(i) is computed using 2.24. If s1 is positive and s2 is negative, then P(i) is a peak value. The number of peaks is incremented by 1 and P(i) is compared with the current maximum peak value. The slope value is computed by summing s1 and s2. Initially, for P(0), only s2 is calculated and tested to see if it is negative. If negative, then P(0) becomes the current maximum, and the slope around the maximum is set to 2*s2.

$$s1(i) = P(i) - P(i-1) \qquad 2.24a$$

$$s2(i) = P(i+1) - P(i) \qquad 2.24b$$

Stepwise discriminant analysis determines which elements of the feature set have discrimination potential for terminal classification. The results of this analysis are used by the terminal classifier to determine the set of discriminant features to be extracted from the feature set and used in epoch classification.

2.2 TERMINAL CLASSIFICATION

Terminal classification is performed generally by dividing an N-dimensional feature space into M decision regions (M classes). The discriminant feature set defines a point which falls into one of the decision regions. The class associated with that region becomes the class to which the EEG epoch being processed belongs.

Given M classes and N features, the linear decision functions $g_i(\underline{x})$ are given by

$$g_i(\underline{x}) = \underline{w}_i^T \bullet \underline{x} \quad 1 < i < M \qquad 2.25$$

where $\underline{x}_i = \{x_1, x_2, x_3, x_4, \ldots x_N, 1\}$ and the weight vectors $\underline{w}_i = \{w_{i1}, w_{i2}, w_{i3}, w_{i4}, w_{i5}, \ldots w_{iN}, c_i\}$. The decision rule for classification consists of choosing i such that $g_i(\underline{x})$ is maximized. Multiplying the weight vectors by -1 results in minimizing $g_i(\underline{x})$ instead of looking for a maximum value.

A decision weight matrix is stored in memory as shown in Fig 9. The routine CLASS uses this matrix along with an array containing discriminant features to classify the EEG epoch used to compute the feature set. The decision is returned in register D0 which contains the number corresponding to the class to which the epoch belongs. Changing the weight matrix changes the terminal classifier without any other modifications.

One or more decision spaces can be used. In parallel classification, a single space (and thus a single set of decision functions) classifies the epochs. Sequential classification (5) consists of dividing the M classes into 2 groups, performing a decision determining in which group the epoch belongs using 2 decision functions, and then repeating the process with the group chosen until only two classes are left. A final decision is then made to determine the epoch class (Fig 10).

M - the number of weight vectors
(i.e. number of classes)

N+1 - the number of elements in each vector
(i.e. number of features + 1)

$\underline{w}_1$
$\underline{w}_2$
$\underline{w}_3$
$\underline{w}_4$
.
.
.
$\underline{w}_M$

vectors each containing N+1 values
$\underline{w}_i = \{w_{i1}, w_{i2}, w_{i3}, \ldots, w_{iN}, c_i\}$

Figure 9: Decision weight matrix stored in memory

Parallel classification requires N•M multiplications and (N+1)•M additions. Sequential classification requires at most $2N\{1+INT(\log_2 M)\}$ multiplications and $2(N+1)\{1+INT(\log_2 M)\}$ additions if M is not a power of 2, in which case the term $1+INT(\log_2 M)$ becomes simply $\log_2 M$. Sequential classification proves to be more efficient than parallel classification but more difficult to implement. A combination of both methods allows grouping of like classes together (See Fig 11).

The classification routine was written to allow easy implementation of parallel classification or a combination of parallel and sequential classification. All that needs to be done is to define different weight matrices for each parallel classification scheme. Within 50 milliseconds, 2500 multiplications will take 40 milliseconds leaving 10 milli-

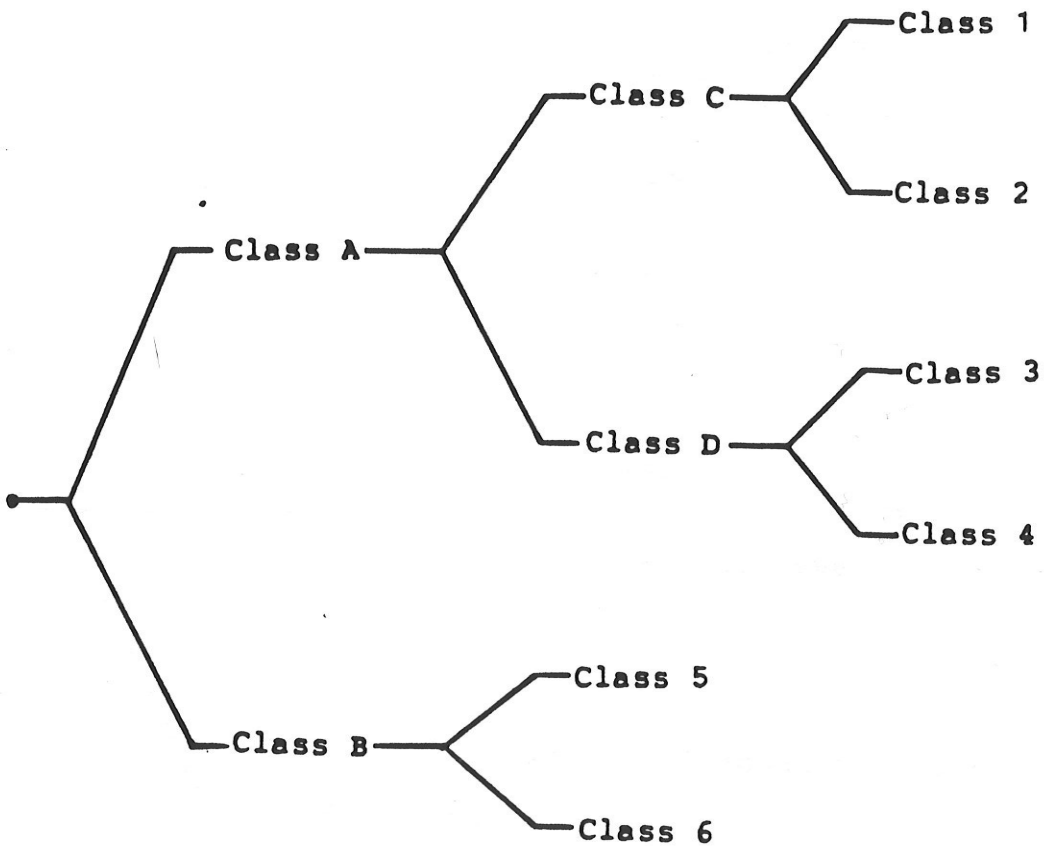

Figure 10: Sequential Classification

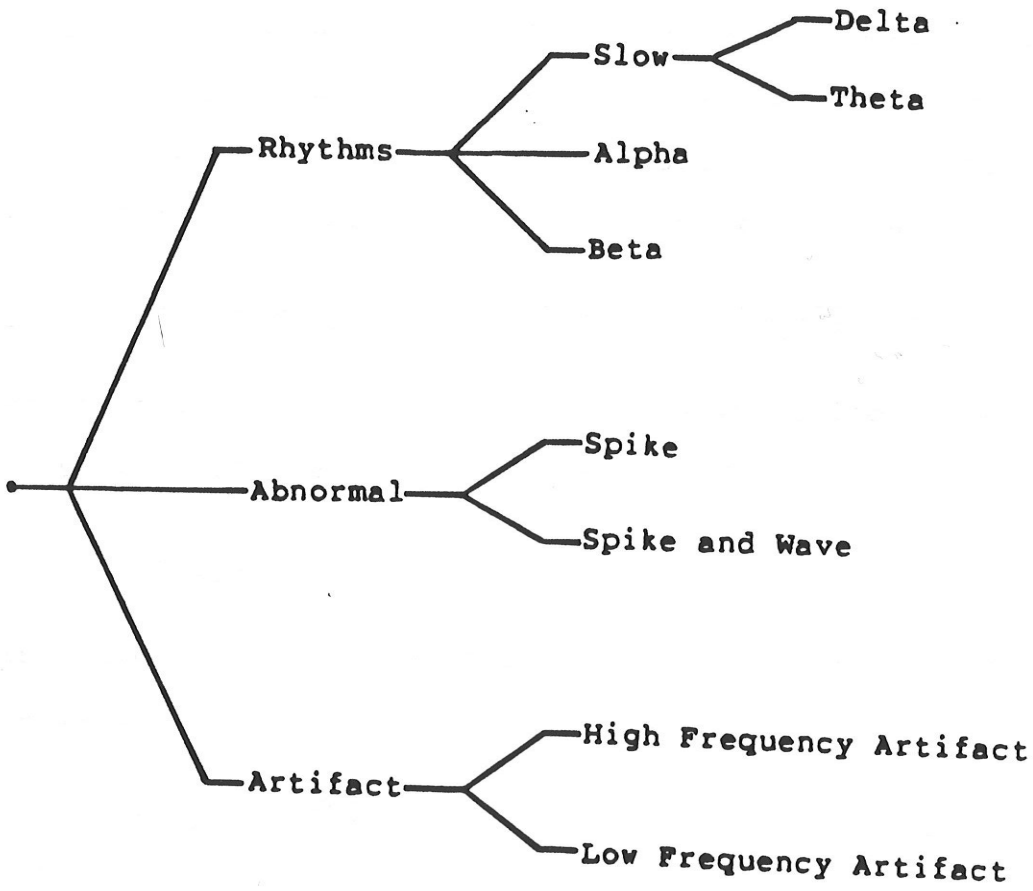

Figure 11: Combining Serial and Parallel Classification

seconds for overhead and syntactic processing. For parallel classifications, N•M must not exceed 2500. For example, using 50 discriminant features, the maximum number of classes allowed would be 50. In the case of sequential classification, the number of classes could be increased.

On-going training consists of using the current classifications to update the decision functions. Consider the distance calculation in 2.26 where $\underline{x}$ is the coordinate of the feature set of the epoch to be classified, and $\underline{w}_{mi}$ is the mean of the training set of features of the class i.

$$||\underline{x} - \underline{w}_{mi}|| = ||\underline{x}||-2\underline{x}^{T}\bullet\underline{w}_{mi} + ||\underline{w}_{mi}|| \qquad 2.26$$

The term $||\underline{x}||$ can be ignored since it is independent of i. Thus the distance will minimized if

$$-2\underline{x}^{T}\bullet\underline{w}_{mi} + ||\underline{w}_{mi}|| \qquad 2.27$$

is minimized. Equivalently, minimizing 2.27 is the same as maximizing

$$\underline{x}^{T}\bullet\underline{w}_{mi} - 1/2||\underline{w}_{mi}|| \qquad 2.28$$

Comparing the results in 2.28 to the decision function of 2.25, the weight vector $\underline{w}_i$ of the decision function can be considered the training set mean $\underline{w}_{mi}$ with the constant c defined by 2.29.

$$c_i = -1/2||\underline{w}_{mi}|| \qquad 2.29$$

The weight vectors can be adjusted as current classifications are averaged. After a sufficient number of epochs of a particular class have been processed, the weight vectors are updated to better conform to the recorder and subject being processed.

To accomplish this, a correction matrix, separate from the decision weight matrix, is updated every time a classification is done. Fig 12 shows how the correction matrix is saved in memory. MXCNT indicates the number of classifications of a particular class needed before updating the weight matrix. This value is compared to the last value of the correction array (CNT(i)) each time it is updated. When CNT(i) has reached MXCNT, each element of the array is divided by MXCNT by shifting the number of times indicated by SHFCNT. MXCNT is limited to a power of 2 permitting efficient division using the shift operation. Updating the correction arrays consists of summing the elements of the feature set to the corresponding element in the correction array and then dividing by 2 (averaging). The value of c in the weight arrays is computed using

$$c_i = -1/2 \sum_{k=1}^{N} w_{ik}^2 \qquad 2.30$$

The routine CORMAT implements the adaptation procedure. A user can simply call the subroutine after classification is performed by CLASS.

MXCNT - # of samples to be taken before updating the weight matrix (16 bits)

SHFCNT - # of shifts needed to divide the elements of the correction matrix by MXCNT

$\underline{w}_1$
$\underline{w}_2$
$\underline{w}_3$
.
.
.
$\underline{w}_M$

Correction arrays
$\underline{w}_i = \{w_{i1}, w_{i2}, w_{i3}, \ldots . w_{iN}, CNT(i)\}$

Figure 12: Correction Matrix

2.3 SYNTACTIC PATTERN RECOGNITION SUBSYSTEM

The 24-hour EEG record transforms into a string of over 86,000 terminals necessitating an on-going syntactic analysis. Thorne's grammar, made up of five steps, is utilized to provide ongoing analysis. These five steps are

1. Merge adjacent terminals,
2. Merge patterns of mixed strings formed by terminals (T) and nonterminals (NT),
3. Assign irregular nonterminal code to remaining isolated unclassified terminals,
4. Merge adjacent identical nonterminals,
5. Merge patterns of nonterminals.

The production merging rules making up his grammar, are placed in a priorized order to omit any conflict as to which production is to be applied at any time.

Four buffers are used for syntactic analysis. A record buffer contains the final result when all analysis is done. A working buffer is divided into two sections (Fig 13): a nonterminal section and a mixed string section. The other two buffers, time buffers, contain the number of epochs associated with each symbol in their corresponding buffer.

The first three grammar steps transform the mixed string into nonterminals while the last two steps merge the nonterminal string. Reduction of the mixed string is repeated until the working buffer is full and no further reduction can be done. The nonterminal string is then merged making more room for additional terminals. Merging of both strings until no merging of either string can be done indicates that a certain number of leftmost symbols can be considered in the their final form. Moving them to a record buffer makes more

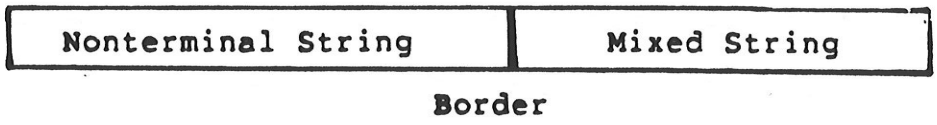

Figure 13: Working Buffer

room for further processing.

The symbols are represented by single byte numbers giving 255 possible symbols. Defining the first NT symbol partitions the 255 values where the first partition contains num-

bers that are terminals, the other partition nonterminals. For example, if hex 20 is the first nonterminal symbol, then 0 - 1F make up the set of terminal values while 20 - FE make up the nonterminal values. FF is reserved for the IRREGULAR code.

In the first step, the mixed string section of the working buffer is scanned from the begining to the end checking for adjacent identical symbols by the routine MRGTER. When this arises, they are counted and then replaced by a nonterminal symbol. If the identical symbol is a terminal, the nonterminal code is obtained from a table using the terminal code as the offset from the start of the table to address. If identical nonterminals are adjacent, they are merged without changing their code. After the merge the end of the string is shifted over and checked for more identical symbols.

At the end of the working buffer, terminals may be merged together before all of them are available (see Fig 15). T1 will not be merged with NT1 unless the following terminals are also T1, then two adjacent NT1 will be merged together. If the following terminal is different, then T1 will be isolated and should be merged with NT1. This condition is looked for after all merging of identical symbols has been performed.

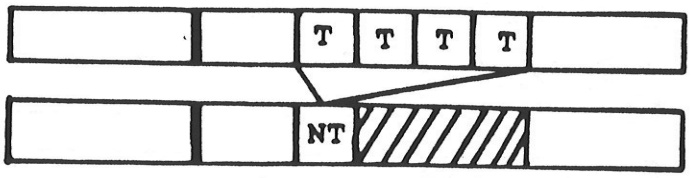

Figure 14: Merging Identical Symbols in the Mixed String

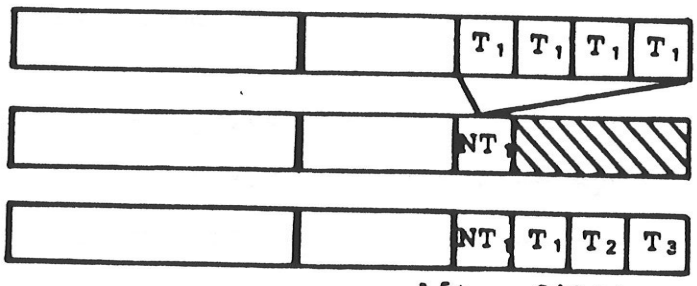

Figure 15: Merging identical terminals at the end of the buffer

In the second step, once the identical terminals have been merged, the string is verified for patterns of mixed strings by the subroutine MXSTR. The production rules for merging mixed strings of terminals and nonterminals are coded using two words (see Fig 16). If the first upper byte of the 32 bit rule is zero, then two symbols are merged XX <- YYZZ, that is the code XX replaces the pattern YYZZ where XX is a nonterminal code and YY,ZZ can be either terminal or

```
00XX YYZZ     Case 1 - Two at a time
WWXX YYZZ     Case 2 - Three at a time
01XX 00YY     Case 3 - Single conversion
```

Figure 16: Production rule coding for merging mixed strings

nonterminal codes.

If the upper byte is nonzero, then three symbols are merged, WW <- XXYYZZ. The three symbol pattern XXYYZZ is replaced by WW. If the upper byte is equal to 01 then the a single symbol is replaced by a nonterminal code, that is YY is replaced by XX. If the first byte is 02, then the end of the production rule table has been reached. These values of 00, 01, and 02 being terminal codes will never occur in the first byte of a production rule. This scheme means that the nonterminal codes must start at a value greater than 02.

Mixed string productions limited to merging 3 symbols, will merge combinations of terminals and nonterminals in the mixed string section of the working buffer. The productions are applied from the start of the terminal section to the symbol occupying the fourth last position, guaranteeing that the last 4 symbols merge only by production rules which merge 3 symbols and leaving the last symbol intact. This guarantees that any terminals which are added to fill the buffer and are to merge with the last symbol will do so.

For example, consider the case where a production rule merging 3 symbols occurs before one merging 2 symbols and that the two symbols merged in the second rule are the same as the first two symbols of the first rule (merging 3 symbols). If these two symbols are the last ones the work buffer and rules are applied from beginning to end. The lower priority production rule will then be applied even though there is a possibility that the next terminal produced should be merged using the first rule. The merging of identical terminals always remains the higher priority. As well lower priority rules will not be applied before higher priority rules.

In the third step, the subroutine ASSIRR scans the mixed string for terminals sandwiched between nonterminals. They are assigned an IRREGULAR code ($FF). When this is completed, the boundary is moved to the right until the first terminal is encountered (Fig. 17). Thus symbols merged into a

string containing nonterminals now become part of the nonterminal section of the work buffer.

Identical juxtaposed nonterminals are now merged in the fourth step. This usually occurs when a long string of similar terminals have been truncated by the end of the working buffer. It will also merge any IRREGULAR codes that may be juxtaposed.

When the buffer is full of nonterminals or no further merging can be done in the mixed string section of the buffer, the nonterminal section of the working buffer is collapsed in the fifth step: merging of nonterminal strings.

The subroutine MRGNT uses production rules to merge patterns of nonterminals. There is no limit to the number of nonterminals that can be merged. The productions are coded as shown in Fig 18. The first word contains the number of words which contain the production rule. If the number of terminals to be merged is even, then the second word of the rule is of the form 00XX, where XX is the nonterminal replacing the string. If this number is odd then the second word is of the form XXYY where XX is the nonterminal replacing the merged string and YY is the first nonterminal in the merge string. The end of the production list has a count of 0.

As in the case of merging mixed strings, a production rule is applied one at a time to the nonterminal section of

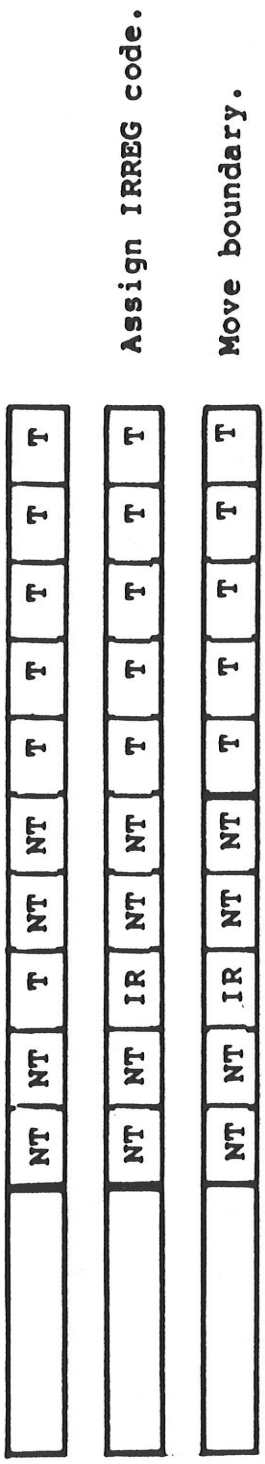

Figure 17: Moving the buffer boundary

```
COUNT   00XX YYZZ ....AABB    Case 1 - Even number of
                                       symbols
COUNT   XXYY ZZAA ....BBCC    Case 2 - Odd number of
                                       symbols
```

Figure 18: Rules for merging nonterminal strings

the working buffer. Again the last few symbols in the string must be protected to make sure a production rule of lower priority is applied before one of higher priority. Defining a maximum number of symbols merged, say X, as part of the production list, the rules are applied making sure that the last X symbols are not processed by rules merging less than X symbols.

After merging of patterns, the nonterminal string is checked for adjacent identical nonterminal symbols which are merged. Work on the terminal string can now be resumed since the merging of the nonterminal string has left room for additional terminals.

When no merging of either section of the buffer can be done and that the buffer is full, a predetermined number of symbols are removed from the start of the buffer and stored in the record buffer (times included). More room is available and the process is started again.

Whenever any merging occurs, the time buffer is always updated. The end of the string is then moved over to the right freeing the bytes at the end of the work buffer.

III

IMPLEMENTATION OF THE PATTERN RECOGNITION SYSTEM

Chapter II describes how the system recognizes EEG signal patterns. This chapter introduces the hardware and software which makes up the system. Two Motorola 68000 educational boards (ECB) compose the system. The first one, the feature extraction (FE) board, is interfaced to a wire wrap board containing a single channel 12-bit A/D converter, a 16-bit multiplier, and 32K of static RAM replacing the on-board dynamic RAM. These additions increase the board speed. The second board, the terminal classification/syntactic analysis (TC/SA) board, receives the feature set via a parallel port and provides terminal classification and syntactic analysis. More information regarding hardware and software is found in the appendices. A detailed description of the system software is found in the manual 'EEG Pattern Recognition System Software' (13). The MC68000 EDUCATIONAL COMPUTER BOARD USER'S MANUAL (8) contains a complete description of the educational board.

Software was written in a modular style to allow customizing of the recognition system. For example, one may use either the KZAR model or the AR model using either BURG's or DURBIN's algorithms. Parameters such as the number of features in each of the subsystems are easily changed via a variable list. For instance buffers can be moved in the memory by changing labels. Labels are also used to define nonterminal and terminal symbols in the syntactic analysis. This facilitated easy coding of the production rules.

3.1 SYSTEM HARDWARE

The ECB provides a suitable system coupled with an assembler, not only for processing the EEG signal, but also for system development. The board provides

1. a 4 MHz MC 68000 processor,
2. 32 kilobytes dynamic RAM,
3. a 16 kbyte ROM monitor (TUTOR),
4. two serial RS-232 communications ports,
5. a PI/T MC68230 timer/parallel interface device,
6. and a wire wrap area.

The 16-bit processor, with a 16-bit data bus, a 23-bit address bus, eight 32-bit data registers, seven 32-bit address registers, two 32-bit stack pointers, a 32-bit program counter and a 16-bit status register, offers programming capabilities such as fourteen addressing modes and 56 powerful

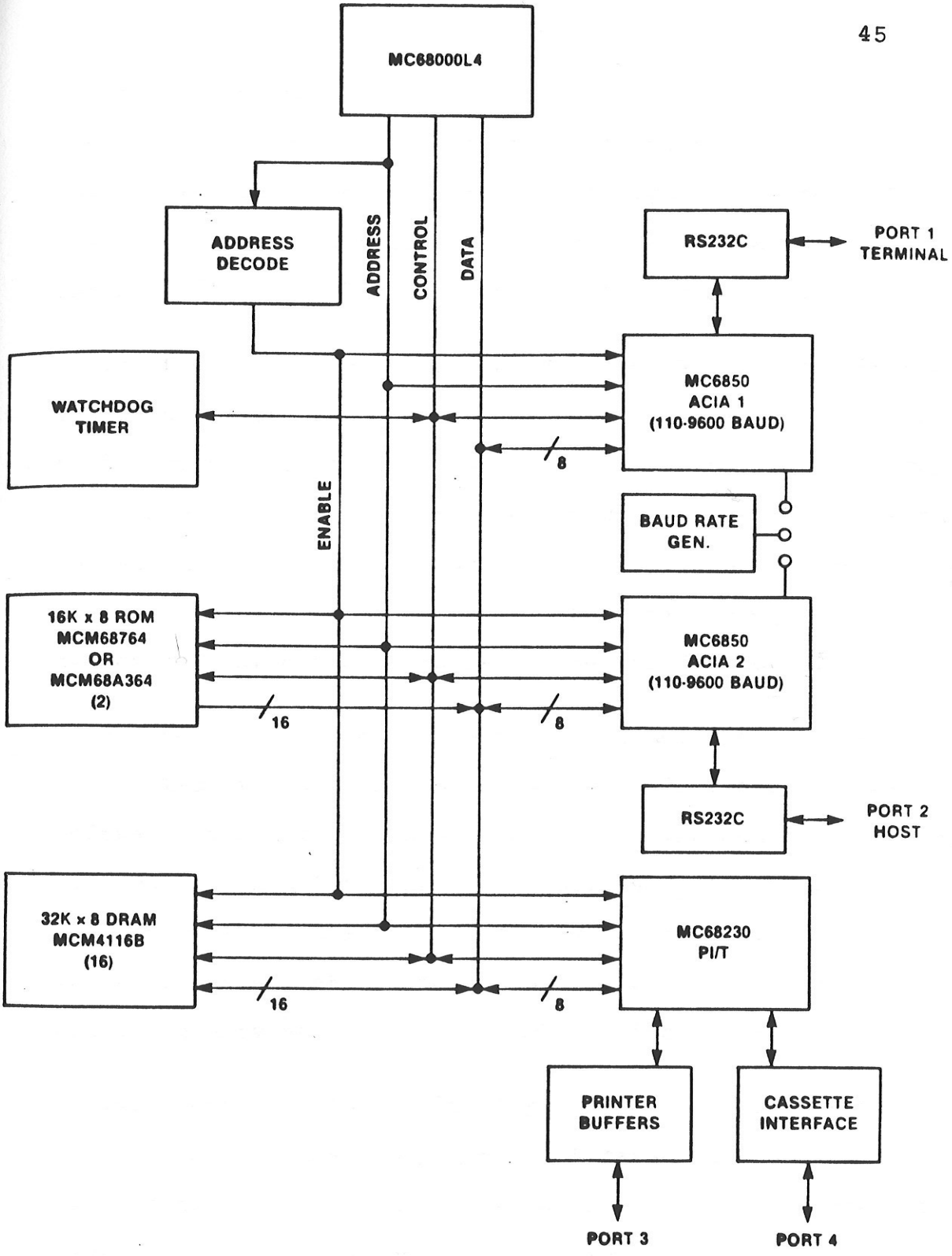

Figure 19: Functional block diagram of the ECB

instruction types including 16 bit signed and unsigned multiply instructions. It is thus well suited for the heavy computation load required for the analysis of the EEG signal. A detailed description of the device can be found in the Motorola data sheet (7) and user's manual (6).

The first 2304 bytes of the 32 kbyte RAM (address 0-$8FF) are used for the 68000 exception vector table and the monitor's scratchpad RAM leaving over 28 kilobytes for the user. The dynamic RAM slows down the 68000 by requesting the bus roughly every 2 ms and inducing wait states when memory is accessed. The FE board's dynamic RAM was therefore replaced by faster static RAM to increase the board's speed.

A total of 26 commands comprise the TUTOR monitor command set. Two of these are used for uploading and downloading software to and from a host computer. The monitor also has various commands for debugging routines. For instance, up to eight breakpoints can be set and a trace command executes instructions one at a time. Software can also be examined and changed by an on board assembler/disassembler.

Two MC6850 ACIA's provide asynchronous communication to a terminal and a host computer. A pass through capability linking the terminal directly to the host with a 68000 assembler computer provides for convenient assembly and editing of source code. Software can then be downloaded directly into the educational board from the host. The MC68230

timer/parallel interface contributes a 24 bit programmable timer used for timing routines and parallel ports. One of the ports is used for communication between the two ECB's for feature set transfer. The wire wrap area on each board was used to add two 2732A EPROM providing 8K of non volatile memory for the user's programs.

3.1.1 Recognition System Hardware

Figure 20 shows a block diagram of the hardware used for development of the EEG analysis routines. Modifications were made on the FE board to 1) provide the bus to the wire wrap board, 2) allow parallel communication with TC/SA board, and 3) disable the dynamic RAM (see the appendix A).

The system bus is brought to the wire wrap board using a 50 pin ribbon cable. The 50 signals consist of the 16-bit data bus (D0-D15), the 23 data bus lines A1-A23, 4 function code bits (FC0-FC1), the upper and lower data strobe lines ($\overline{UDS}$,$\overline{LDS}$), 6 control lines ($\overline{RESET}$,$\overline{AS}$,R/$\overline{W}$,$\overline{DTACK}$, 6800 IRQ, $\overline{VMA}$), 3 clocks (4 MHz, 8 MHz, 1 MHz), and 4 user lines. Two of the user lines are used for signal RAMEN which enables the static RAM and for signal E1 used in addressing the A/D and the hardware multiplier (address segment $20000 - $2FFFF). The other two are available for future use.

The 68230 port B of each board is used for communications between them. Two lines on the FE board are exchanged to

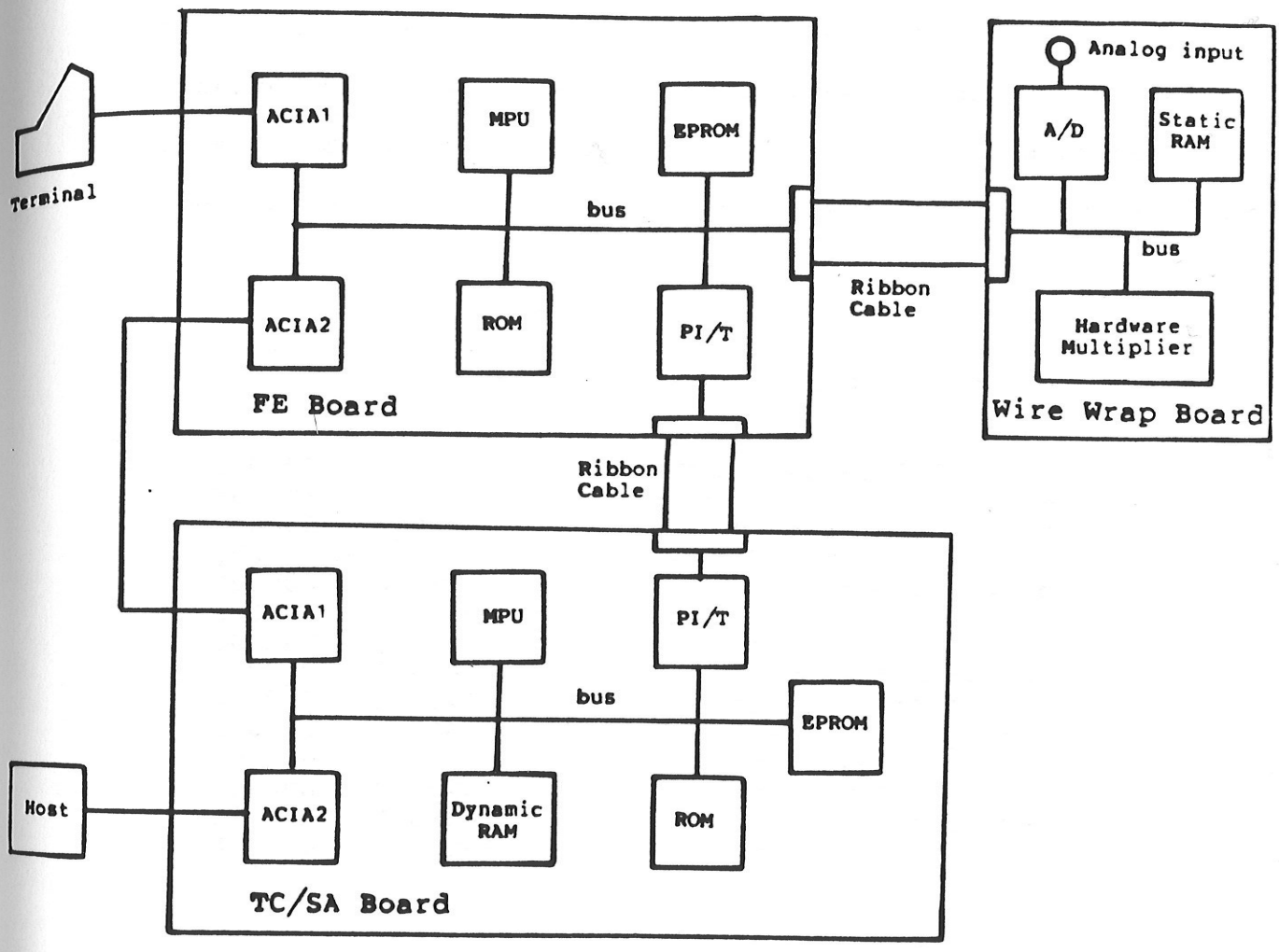

Figure 20: Block Diagram of the EEG Analysis System Hardware

provide proper handshaking with the control lines of the TC/SA board. The parallel communications provides transfer rates of 300 ns per byte.

A terminal is connected to port 1 of the FE board while port2 is connected to port 1 of the TC/SA board. The host is accessed using port 2 of the TC/SA board. Using the transparent mode of the ECB boards, the terminal can access either board or the host. As well, software can be downloaded

TABLE 1

Memory Map of the FE Board

$00000 - $00007	RESET vector (ROM)
$00008 - $003FF	Exception Vector Table (RAM)
$00400 - $008FF	TUTOR Scratchpad Memory
$00900 - $07FFF	User Memory
$08000 - $0BFFF	TUTOR Monitor Firmware
$0C000 - $0DFFF	User EPROM
$0EFFF - $0FFFF	Not Used
$10000 - $1003F	PI/T Device (lower byte)
$10040 - $10043	Address of both ACIA's
$10044 - $1FFFF	Redundant mapping of ACIA's
$20000 - $20007	A/D Converter
$2000A - $2000E	Hardware Multiplier
$2000F - $2FFFF	Reserved for other user devices
$30000 - $FFFFFF	Not used

to either board.

Table 1 shows the memory map for the FE board which includes the hardware multiplier and A/D system located in the page $20000 - $2FFFF. The TC/SA board has an identical memory map except that $20000 - $2FFFF is not used.

A 12 bit A/D system was designed which gives an accuracy of 0.0244%. For a signal range of ±2.5 volts, this gives a resolution of 1.22 millivolts. The playback time signal which is bandlimited to 1000 Hz (50 Hz EEG recording time), is sampled at 2.56 kHz. The aperture time of a sample and hold (S/H) is given by $\beta/\pi f$ where β refers to the A/D resolution $1/2^n$ and f the maximum frequency of the bandlimited signal. For a 1000 Hz bandlimited signal and a 12 bit resolution, the aperture time is therefore restricted to 77 nanoseconds or less. The AD583 sample and hold (9) satisfies this restriction with an aperture of 50 ns.

Total conversion time including the S/H settling time is 35 microseconds, 10 microseconds settling time and 25 microseconds (maximum) for the A/D conversion. Thus if desired, the sampling rate can be increased to 28.5 kHz or other channels can be processed by the same converter.

Two control pins (CONVERT START and STATUS) start the digitizing process and interrupt the processor when converted data is available. A clock of 2.56 kHz generates a 10 microseconds CONVERT START signal to the A/D which asserts the STATUS signal. Conversion starts when the CONVERT START signal goes low (Fig 21). The 10 microseconds pulse provides the time for the S/H to settle. The STATUS signal goes low when the conversion is completed allowing the S/H to resume tracking and induce a system interrupt (6800 IRQ) via the edge sensing circuit. A data read clears the interrupt signal.

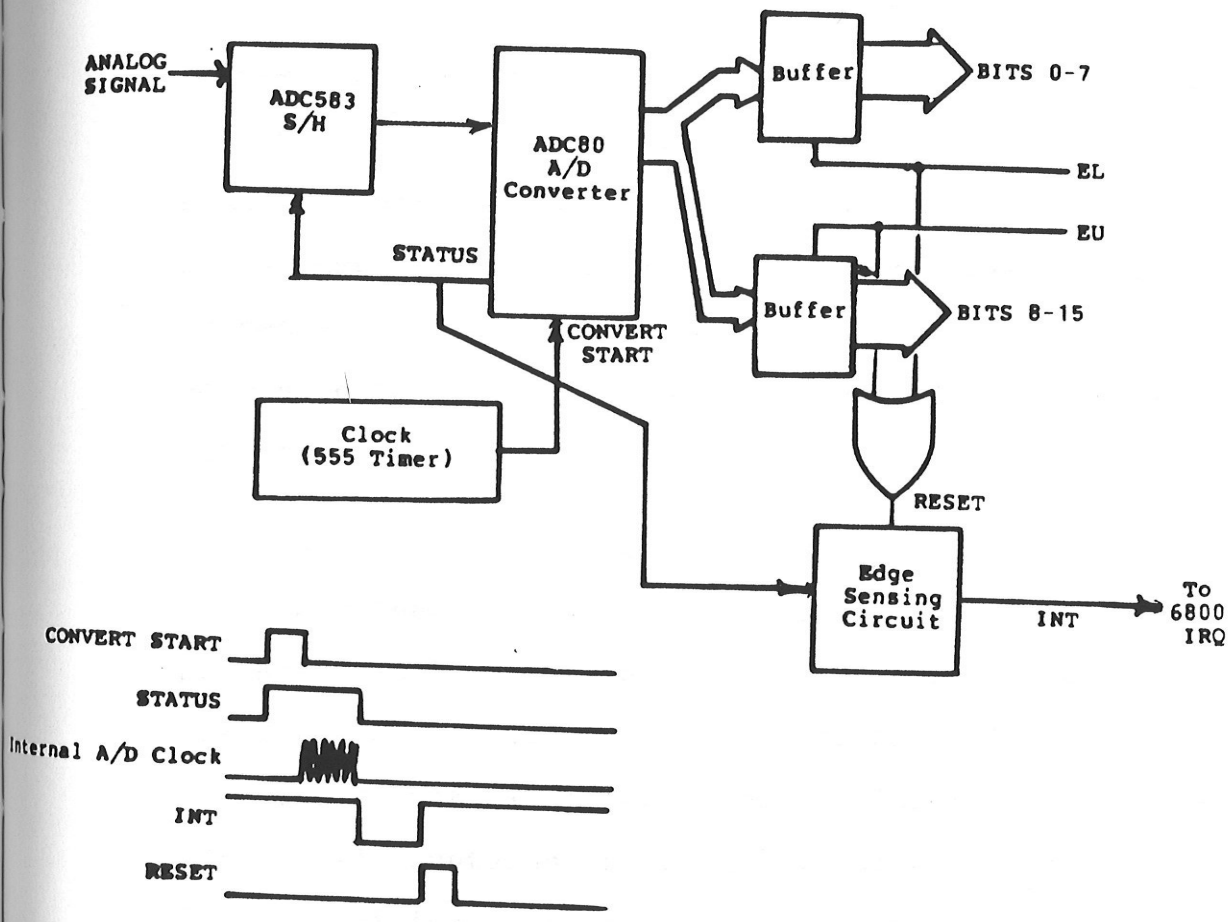

Figure 21: A/D Conversion

This configuration imposes little overhead for the processor which needs only to read the data as it becomes available. Upon power up, the 68000 initializes its buffers and interrupt vector before responding to the 6800 IRQ interrupt. This interrupt is autovectored using the address located in the exception vector table at address $70.

The feature extraction algorithms require many multiplications. A 16-bit hardware multiplier/accumulator was thus added to increase the FE board's speed. The 115 ns multiply/accumulate time of the multiplier make computation of processes such as autocorrelation a simple transfer of array values to the device. Data is multiplied and summed automatically.

Data is first loaded into the X register (clocked by CKX), and then into the Y register (clocked by CKY). Multiplication is initiated automatically upon loading data into the Y register (Fig 22). Asserting CKP will load or accumulate the results into the output register. The XTP register (Fig. 22) contains three additional bits (sign extended to 16 bits) giving more accumulation capability and can be used for overflow detection. Register addresses are shown in Table 2. Four control signals, monitored as the data is loaded, configure the multiplier. A register (MLSTAT) is loaded by the CPU to determine the values of these signals before any processing is undertaken. The control signals are:

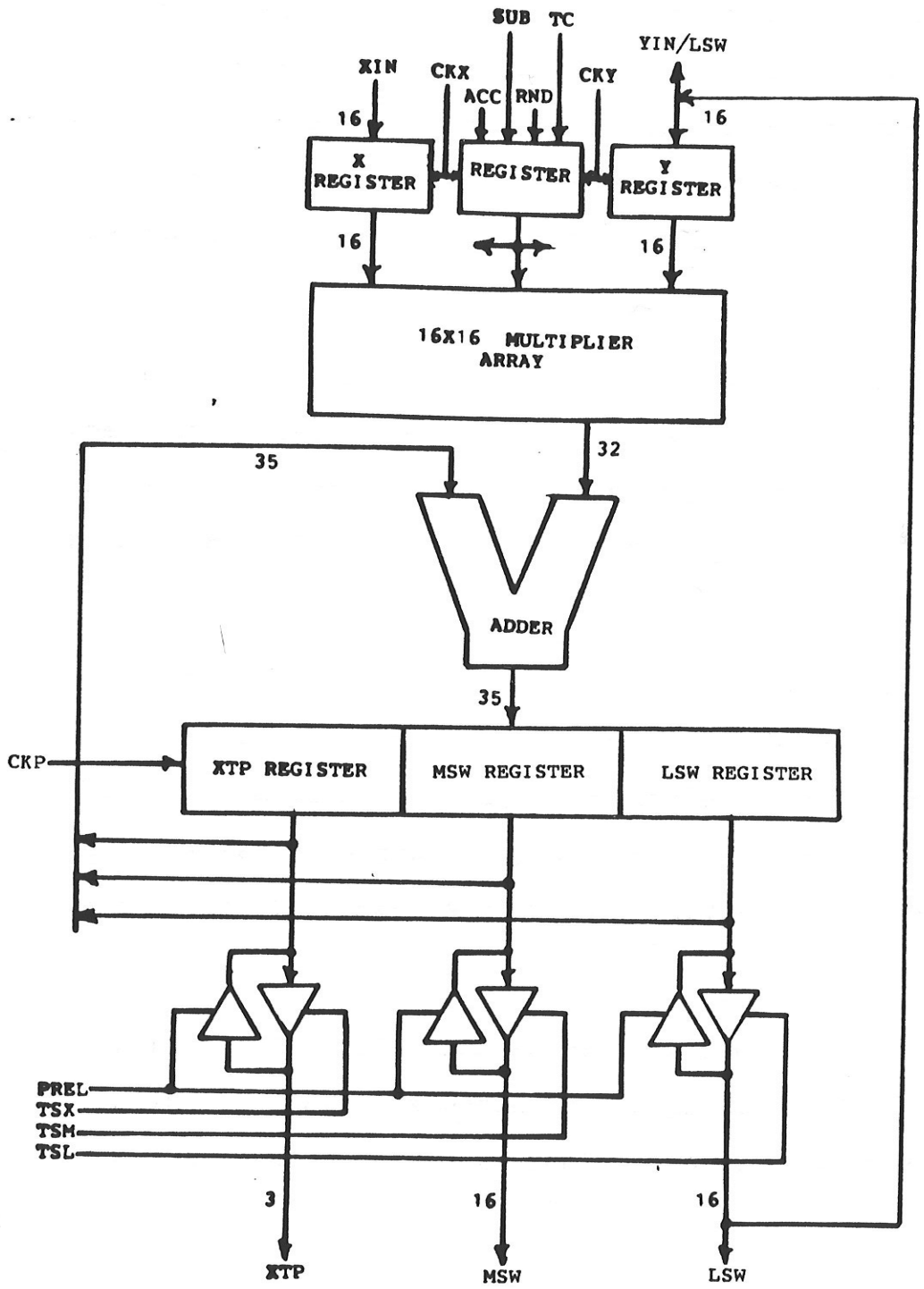

Figure 22: Block Diagram of the TDC1010J Multiplier

bit 0 - ACC (Accumulate): When this bit is low, the multiplication result is loaded directly into the output register. A 1 places the multiplier in the accumulation mode. The results of the multiplication are added or subtracted to the contents of the output register (see bit 1 - SUB).

bit 1 - SUB (Subtract): When ACC is high, the results of the multiplication are added to the output register if SUB is low and subtracted from the output register if SUB is high. When ACC is low, SUB is ignored.

bit 2 - RND (Round): A one at this position results in the rounding of the output register. The least significant bit of the most significant word (upper 16 bits) of the output register is adjusted by adding a 1 to the most significant bit of the least significant word (lower 16 bits).

bit 3 - TC (Two's complement): A low at bit 3 results in treating the data as unsigned integers, while a high will have the data processed as two's complement numbers.

TABLE 2

Addressing the Multiplier Registers

	READ	DATA SIZE	WRITE	DATA SIZE
$2000A	XTP	16		
	XTP + MSW	32		
$2000B			STATUS	8
$2000C	MSW	16	YIN	16
	MSW + LSW	32		
$2000E	LSW	15	XIN	16

XIN - X register.
YIN - Y register.
LSW - Least significant word of the output register.
MSW - Most significant word of the output register.
XTP - XTP register (16 bits).

3.2 SOFTWARE

The 68000 software routines were developed using a Motorola cross-assembler available on the University of Manitoba Amdahl mainframe. The object code, stored in S-format code, is decoded by the ECB's when downloaded from the host. The cross assembler features include macros, conditional assembly (11) and labels which are used to define system variables.

Once a routine is assembled and downloaded into the educational board, the TUTOR firmware provides commands for displaying and modifying memory and internal registers, for program control and execution; an editor/assembler for modifying code; and access to TUTOR routines via the TRAP #14 instruction (Chapter 5 of ECB user Manual 8). The original

source on the host need only be changed for major changes or once the routine has been debugged using the on-board assembler/disassembler.

A fixed point arithmetic, where the lower 12 bits form the fractional portion and the upper 4 bits the integer, was chosen for both boards. Thus the EEG signal is digitized to values between $\pm(1.0 - 1/2^{12})$. When multiplying two 16 bit values, the 32 bit result has its decimal point located between bits 23 and 24. The leftmost 8 bits form the integer while the lower 24 bits form the mantissa. When a 32 bit result is truncated to 16 bits, the total result should be shifted left by 4 bits. The variable FXPT has been defined to allow the user to change the position of the decimal point. The label indicates the number of leftmost bits which define the integer portion of the 16 bit values. Presently it is set to 4. The variable ONE and MINONE should be changed along with FXPT to give the values 1.0 and -1.0 relative to the decimal point.

3.2.1 Feature Extraction Software

The memory map for the FE board RAM is shown in Table 3. Two data buffers (256 bytes) are used as swinging buffers for processing the digitized data. Eight kilobytes of RAM ($1000-$2FFF) is reserved for program development. The routines and tables containing hamming window coefficients and coefficients used in computing the power density spectrum

also occupy this space. Once the routines have been debugged, they can be saved on EPROM resulting in a system operating independently from the host.

TABLE 3

Memory Map of the FE Board RAM

$0000 - $03FF	Exception vector table
$0400 - $08FF	TUTOR scratchpad RAM
$0900 - $0FFF	Not used
$1000 - $2FFF	Feature extraction routines
$3000 - $30FF	First data buffer
$3100 - $31FF	Second data buffer
$3200 - $39FF	Scratchpad RAM for routines
$3A00 - $3FFF	Routine communication memory
$4000 - $40FF	Feature set memory
$4100 - $6FFF	Not used
$7000 - $7FFF	Reserved for system stacks

The scratchpad RAM is used for processing by the routines while the communications memory is reserved to pass data from one routine to another. For example, the routine PWDEN saves the power density spectrum in the communication memory for the routine FEATRS which computes the spectral features. Table 4 shows the addressing scheme of the feature set for an AR model order of 20. The feature set occupies less space for smaller model orders.

A set of system variables are defined to allow changes to be made easily to routines. As research continues, parameters can be easily modified by changing labels. For example the data buffers can be doubled by changing the values of

TABLE 4

Feature Set Memory Allocation

ADDRESS	Value Stored	Number of Bytes Occupied
$4000 - $4001	Model order	2
$4002 - $4003	G^2	2
$4004 - $402B	AR coefficients	40 (max)
$402C - $402D	Terminator (0000)	2
$402E - $4039	Power in Bands	12
$403A - $4045	Power Ratios	12
$4046 - $404D	Max Spectral Features	8
$404E - $404F	Terminator (0000)	2

DTBUF1 and DTBUF2 for proper addressing of buffers and the values BFLNGT and N so that the routines will access 256 data points instead of 128. Certain values will not change such as the labels used in addressing the multiplier registers. Appendix B contains the list of all variables used by the feature extractor software.

A feature extractor system was implemented with the program FETEXT (see Figure 23). It initializes the FE subsystem, prompts the user for the model order (2 characters) and controls feature extraction using the AR model with Durbin's algorithm. After the model order has been entered, processing begins by striking the 'B' on the keyboard. Striking the character 'E' will end the processing. After each successful extraction, the feature set is transmitted to the TC/SA board, the character '0' is sent to the terminal, the register A0 is cleared, and the next epoch is processed.

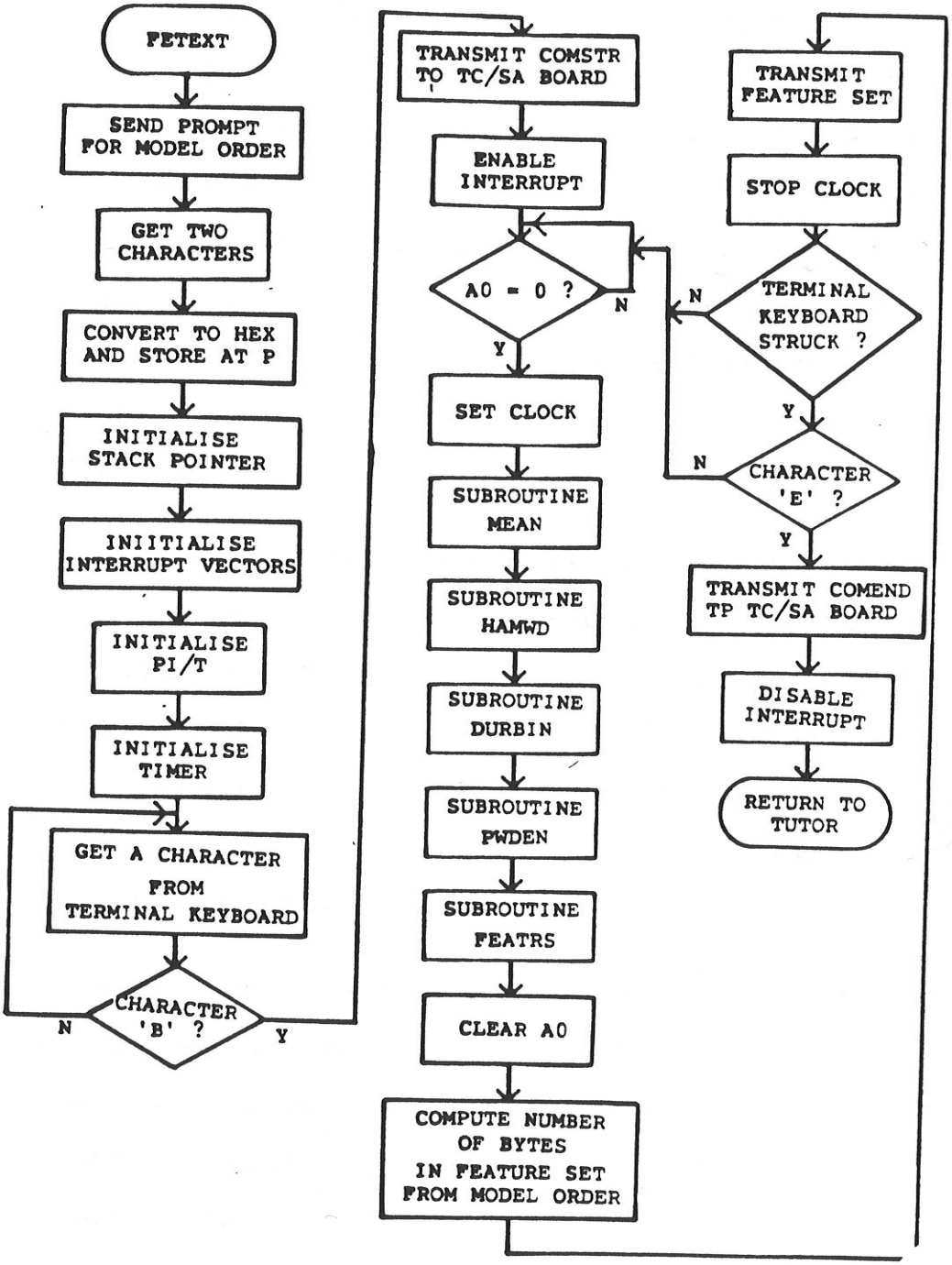

Figure 23: Flowchart of FETEXT

The registers A0, A6, D7 have special purposes for EEG epoch feature extraction. All other registers are available for use by each of the subroutines. A0 is used to preserve the address of the data buffer being processed while the other data buffer is being filled by the interrupt data acquisition routine ACQUIS. Once buffer processing is completed A0 is cleared. The data acquisition routine verifies that A0 is cleared (processing ended) before loading into A0 the address of the buffer just filled. The main routine also uses A0 as a flag to indicate when analysis can begin.

Calling the following sequence of subroutines will extract features from the epoch data addressed by A0 using the AR model and Durbin's algorithm: MEAN, HAMWD, DURBIN, PWDEN, FEATRS. To change to another model or algorithm, this sequence of subroutines must be changed. Appendix B contains the description of all subroutines developed for the feature extractor.

A6 and D7 are reserved for the data acquisition routine. A6 is used for addressing the data buffer while D7 contains the last terminating address to which A6 is compared. This provides an indication that the data buffer is full and ready to be processed.

The implemented feature extraction system also recovers from errors using exception processing. A divide by zero induces an exception automatically. TRAP instructions ini-

tiate exception processing for other errors. The TRAPV instruction is used for detecting overflow errors. TRAP #0 is invoked if Durbin's filter becomes unstable. TRAP #1 is invoked when the data acquisition routine ACQUIS has filled a data buffer and feature extraction of the other buffer is not yet completed (register A0 not cleared). Each exception transmits to the terminal a character as shown in Table 5 and an error command to the TC/SA board. Epoch feature extraction is then resumed.

TABLE 5

Error Detection by FETEXT

Character Transmitted to Terminal	Type of Error
1	Filter Unstable (TRAP #0)
2	Data Buffer Overflow (TRAP 1)
3	Overflow error (TRAPV)
4	Divide by Zero

3.2.2 TC/SA Board Software

Table 6 shows the memory map for the TC/SA system RAM. As in the case of the FE board, 8K of RAM has been set aside for development of software. Production rule tables and the decision weight matrix also occupy this space. Buffers used by terminal classification and syntactic analysis routines are also defined as shown in Table 6.

Terminal classification on the TC/SA board is initiated upon the reception of an epoch's feature set. It is performed as a background task (processor interrupted when a feature set is received) while syntactic analysis operates in the foreground. Terminal classification must not occupy the full 50 ms window to allow processing time for the syntactic analysis.

TABLE 6

Memory Map for TC/SA Board RAM

$0000 - $03FF	Exception Vector Table
$0400 - $08FF	TUTOR scratchpad
$0900 - $0EFF	Not used
$0FF0 - $0FFF	Status bytes
$1000 - $2FFF	RAM for development of routines
$3000 - $3FFF	Not used
$4000 - $40FF	Buffer for feature set
$4100 - $41FF	Working buffer for syntactic analysis
$4200 - $43FF	Work time buffer
$4400 - $44FF	Terminal circular buffer
$4500 - $45FF	Array for discriminant features
$4600 - $49FF	Correction weight weight matrix
$4A00 - $4FFF	Weight matrix
$5000 - $5FFF	Record buffer
$6000 - $6FFF	Record time buffer
$7000 - $7FFF	System stacks

As in the case of the FE board software, parameters used by TC/SA routines are easily changed using labels. Appendix B also contains the list of labels used by both subsystems. The program TCSA (Figure 24) initializes the TC/SA board for processing and invokes syntactic analysis by calling the

subroutine SYNANL. An interrupt from the board's PI/T interrupts the syntactic analysis and passes control to the program TRCLS which does terminal classification. TRCLS also updates the correction matrix and a dummy weight matrix to allow the study of adaptation. This dummy weight matrix is initialized with the values of the decision weight matrix used for classification. The actual decision weight matrix is thus not affected by the on going classifications. After an actual EEG signal is analyzed, the dummy weight matrix will contain decision functions modified by the analysis classifications.

The classification code is stored in two terminal buffers. One of them is used by the syntactic analysis subsystem while the other stores the last hour of epoch classifications. If an error code is received from the FE board, then the code TRIRR is stored in the terminal buffers and classification is bypassed. As the first terminal buffer is filled, the syntactic analysis routines will load its working buffer and corresponding time buffer where merging is done. Results are then transfered to the record buffer.

Processing is not started until the control command COMSTR is received from the FE board. Upon the reception of this command, bit 0 of the STATUS byte is set by TRCLS to start the syntactic analysis and terminal classification.

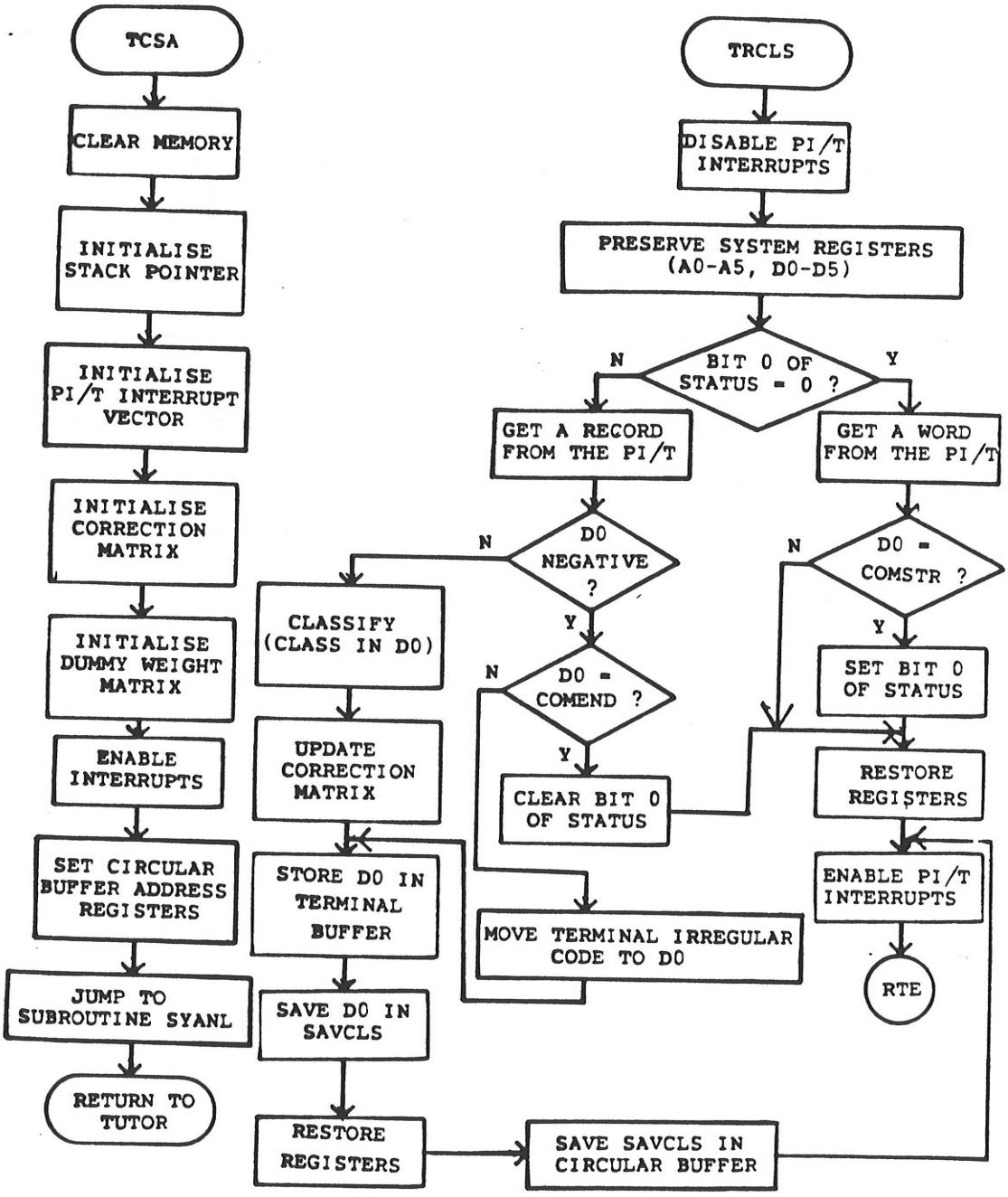

Figure 24: Flowcharts of TCSA and TRCLS

Registers D0-D4, A0, and A1 have the following functions for the syntactic analysis subsystem:

D0 - Working offset register.

D1 - Working offset register.

D2 - Offset to the end of the symbol string.

D3 - Offset of the start of the mixed string.

A0 - Address of the work buffer.

A1 - Address of the record buffer.

D0 - D4 are offsets in indexed addressing of A0, and used to access symbols in the work buffer. Register A1 is used to store results of the syntactic analysis in the record buffer. Time values, that is the number of epochs per symbol, are stored using constant offsets from A0 and A1.

IV

SYSTEM TESTING AND EVALUATION

The EEG pattern recognition system is a complex system involving the implementation of heuristic methods. It was evaluated and tested to verify its performance and its limitations. This evaluation consisted of three stages: simulated data stage, database stage, and real signal stage.

In the first stage, programs on the Amdahl mainframe were developed to create a simulated EEG epoch which was downloaded into the 68000 system. The 68000 feature extractor then extracted features from this simulated epoch. In parallel, the Amdahl mainframe was used to extract features from the same epoch. Using the Amdahl as a reference, the feature extraction routines were tested to assure their proper operation.

Similarly, the terminal classifier masking and classification routines were tested by creating simulated feature sets and decision routines. The Amdahl programs always served as a reference to which the 68000 routines results were compared.

On the other hand, the syntactic analysis subsystem was tested without any parallel processing on the Amdahl. An invented grammar and terminal string provided the necessary data for testing the proper operation of the syntactic analysis routines.

As part of this first stage of evaluation, the FE routines' processing times were measured using the PI/T timer with the model order 10. Table 7 shows these results. BURG with an 89.3 ms processing time can not be used for processing the EEG signal at playback speed. As will be seen further, Burg's algorithm offers no classification advantages over Durbin's algorithm. Thus, it was not critical that Burg's algorithm is not available. However the time required for extracting features from an epoch using the AR model with Durbin's algorithm including data acquisition and transmission of the feature set to the TC/SA board was measured at 47.14 milliseconds. Since 50 milliseconds is alloted for processing one epoch, the maximum model order that can be used with the system is 10.

In the second stage, EEG epochs from a database being developed at the University of Manitoba in conjunction with the Health Sciences Centre were used to evaluate the feature extraction methods and the 68000 system accuracy. A first set of 368 epochs were used to test recognition capabilities of four feature extraction methods: AR model using Durbin's algorithm, AR model using Burg's algorithm, KZAR model using

TABLE 7

Timing of FE Subroutines (Model order 10)

SUBROUTINE	TIME (milliseconds)
MEAN	2.24
HAMWD	3.49
DURBIN	20.80
PWDEN	14.20
FEATRS	3.30
KZDAT	1.70
BURG	89.30
PWKZR	15.80
Data acquisition (estimated)	3.30
Transmission of the feature set (estimated)	0.82

Durbin's algorithm, and the KZAR model using BURG's algorithm. These epochs were divided into the following 6 classes: alpha (80 epochs), beta (40 epochs), theta (65), spike and slow (80 epochs), polyspike (23 epochs), and artifact (80 epochs).

Stepwise discriminant analysis produced the classification results presented in Table 8 for three of the methods. Durbin's algorithm was unstable for approximately 65% of the epochs when used with the KZAR model and therefore it was not possible to obtain classification capability for this method. The figures in the table indicate the percentage of correct classifications.

TABLE 8

Results of SWDA Comparing Feature Extraction Methods

CLASS	AR/DURBIN	AR/BURG	KZAR/BURG
ALPHA	84.1%	82.5%	84.1%
BETA	93.1%	93.1%	93.1%
THETA	92.6%	87.0%	74.1%
SPSLW	49.2%	52.5%	67.8%
POLYSP	71.4%	71.4%	35.7%
ARTIF	67.2%	72.1%	63.9%

The KZAR model did provide better classification of the spike and slow wave (SPSLW) but performed poorly when classifying the polyspikes (POLYSP). Burg's algorithm offers no increase in recognition capacity over the use of Durbin's algorithm. The choice for the present recognition system configuration which uses AR modelling and Durbin's algorithm was motivated by these results.

A second set of epochs were extracted from the database to conform to the classes used in the system implementation (Thorne's classes). The set was not only used in testing the accuracy of the system but in the third stage of testing to determine the decision functions used in terminal classification. The 504 epochs were divided the following groups: normal (119 epochs), slow (79 epochs), low amplitude (54 epochs), fast (53 epochs), low frequency artifact (82 epochs), spike and slow wave (86 epochs), and spike (31 epochs). The SWDA classification matrix for this set is shown in Table 9.

TABLE 9

Classification Matrix

GROUP	PERCENT CORRECT	NUMBER OF CASES CLASSIFIED INTO GROUP - NORM	SLOW	LOAMP	FAST	LOFART	SPSHW	SPIKE
NORM	83.2	99	9	8	0	0	0	3
SLOW	81.0	0	64	1	0	8	5	1
LOAMP	55.6	7	6	30	1	10	0	0
FAST	67.9	0	1	6	36	9	1	0
LOFART	34.1	0	12	22	0	28	13	7
SPSHW	60.5	0	8	1	1	10	52	14
SPIKE	38.7	0	10	2	1	1	5	12

System accuracy was tested by comparing the feature extraction results of the 68000 system to the ones generated by the Amdahl system. Appendix C shows the error calculations for 5 epochs one from each of the following classes: normal, fast, low frequency artifact, spike and wave, and spike. Errors are generally within 15 to 20 percent except for a few sensitive features.

The AR coefficients ranged in value from 0.0022 to 1.87 utilizing the whole range of the fixed point arithmetic (±8.0 at a resolution of 0.0002441). When computing the small value coefficients considerable error can be introduced. For example, the fourth AR coefficients of the normal epoch was computed as 0.14795 by the 68000 system while the Amdahl produced a result of 0.05054. Since errors in the autocorrelation coefficients were restricted to the lower 1 or 2 bits, errors produced in the AR coefficients were due to Durbin's algorithm.

Inaccuracies of the AR coefficients can cause further errors in the computation of the PDS. But the algorithm used for calculating the PDS caused larger errors. The denominator of equation 2.18 is usually quite small in value. The 68000 system routine often produces zero as its value. When this happens the value $7FFF is stored in the PDS array. The Amdahl results confirm that these values should be much smaller. The low frequency artifact, spike and wave, and spike EEG epochs seem to be the most sensitive to these errors.

Errors in the first power band features and the slope around the maximum feature are affected by these errors. In the case of the spike and wave epoch features, the 68000 system produced the values 23.44092 (power in first band), 18.26465 (power in the second band), and -6.34619 (slope around the maximum) while the Amdahl gave the corresponding values 2.85986, 3.47363, and 0.13623.

While these errors can be quite large, classification of these epochs were correct indicating a robust system. The results of the third stage presented below confirms this fact.

The third and final stage of testing consisted of processing an EEG recording independent of the database epochs. The signal processed contained various background activity and paroxysmal signals. The second set of EEG epochs were

used by SWDA to define the decision functions used by the system terminal classifier.

Analysis was performed for approximately 50 minutes of EEG signal, that is roughly 2.5 minutes at playback speed. The terminals and the syntactic analysis results were saved. The results were then compared to the actual signal by Dr. P. Jayakar MD.

The background activity which consisted mostly of low level signals and artifact was classified as either artifact or low level signals. Some slow wave artifact was classified as slow. The syntactic analysis regrouped this background activity into sections of artifact, normal activity, low amplitude activity, and irregular sections.

There was considerable misclassification of artifact into isolated spike and wave complexes. There were also two occurrences of spike classification of artifact signals. These misclassified artifact epochs did contain some spike and slow wave characteristics. Runs of large amplitude low frequency artifact were often classified as spike and slow wave. The syntactic analysis identified these regions as paroxysmal signals (PAROX nonterminal). Missed signals consisted of low amplitude spikes and pseudo spikes which were not detected by the system. This reflects the poor classification of spike generated by the SWDA.

Abnormal signals of the recording consisted of three spike and wave complexes. The syntactic analysis properly identified these sections as paroxysmal signals (PAROX non-terminal). Considering that the spike and wave performance according to the SWDA was only 60.5%, the system performed very well in identifying these abnormal signals.

This testing indicates that abnormal signals can be detected by the implemented pattern recognition system. However, there are considerable misclassifications of artifact which contain similar characteristics to other signals (slow, spike, spike and wave).

During another test, the syntactic analysis result buffers (8K) were filled within 3 hours. The amount of signal analyzed by the system can be increased by using the additional 4K of RAM ($3000-$3FFF) not used. The terminal buffer used to save one hour of results could also be used to save syntactic analysis results. Further, since the amount of RAM used by the syntactic analysis output is dependent on the implemented grammar, modifying the grammar may merge terminals more efficiently. For instance, many irregular codes were not merged using the present system. They could be merged with adjacent symbols.

V

CONCLUSIONS AND RECOMMENDATIONS

This thesis described a 68000 EEG pattern recognition system composed of three subsystems: a feature extractor, a terminal classifier, and a syntactic analyzer. Besides the first use as an analysis system for a single channel of EEG, the recognition system can be used as a development system for pattern recognition software. This real time analysis is a first attempt at detecting not only certain EEG patterns such as spikes, but in classifying the whole signal.

The on board debug tools and interfacing capability to a host provides the tools needed for developing and testing EEG analysis routines not only for the analysis of one channel but can be used in developing a 4 channel system.

The present implemented analysis system properly identified paroxysmal regions of an EEG signal. However, it has misclassified artifact and missed some spike and pseudo spike signals. Furthermore, the feature extraction subsystem's speed is stretched to the limit. The maximum AR model order is limited to 10 and only Durbin's algorithm can be used for feature extraction.

Burg's algorithm does not offer better classification but does provide a stable algorithm and the option of the KZAR model. With the KZAR model, Durbin's algorithm is very unstable. The KZAR model did increase spike and wave classification but performed poorly with spike epochs. A more complete database is needed before a proper conclusion can be made with respect to the KZAR model's use.

Recommendations can be divided into two groups: classification technique research and 68000 system hardware improvements. With the completion of the EEG database at the University of Manitoba, terminal classification and feature extraction should be improved. First the KZAR model should be properly tested and verified. Other known zeros can be added to improve classification and even decrease the number of poles required in the filter transfer function as Bohlin (12) has suggested. Efficient algorithms for the moving average autoregressive model should also investigated to decrease classification errors and model order.

Since feature set definition is a heuristic process, other features should be developed and investigated. For example, error features had been developed by Thorne (1) using an inverse filtering technique. It was found to improve the detection of spikes. The adaptation scheme implemented in the system needs to be investigated to see if on-going classifications can improve terminal classification. The present feature set could also be reviewed. For example, the

slope around the spectral maximum is computed by summing the values of s1 and s2. A more meaningful feature would be to sum the magnitudes, |s1| + |s2|. This value would better represent the sharpness of the spectral peak.

As a second step, the syntactic analysis should be studied to reduce the need for strict terminal classification criteria. For instance when misclassifying a spike and slow wave as artifact, it can be corrected when merged with surrounding spike and wave epochs which were properly classified. The recognition system can be used to help develop grammars used in syntactic analysis. Conditions and context in which classification errors occur can be determined by examining the output of the terminal classifier. A grammar can then be developed to absorb classification errors from the adjacent terminals.

Two hardware improvements are required by the recognition system. The first one consists of increasing the TC/SA RAM so that more data can be saved during the analysis of EEG signals. Adding 128K provides enough memory to save all terminals of a 24 hour tape (86K) and the results of the syntactic analysis.

The second hardware recommendation involves increasing the feature extractor board speed. Changing the processor to an 8 MHz 68000 will at most increase the system speed by 2. For better improvement, a DMA (MC68440) will increase

the subroutines speed considerably. For example to compute eleven autocorrelation coefficients requires 2706 transfers from memory to the hardware multiplier. The 68000 processor uses 64,944 cycles to perform these transfers (16 cycles per transfer plus overhead). Thus at a clock rate of 4 MHz, 16.24 milliseconds is needed for these transfers. The MC68440 only needs 2 cycles per transfer or 5412 cycles for the complete autocorrelation computation. Using the same system clock, the time required for the transfers would be reduced to 1.35 milliseconds.

BIBLIOGRAPHY

1) THORNE R.E., Syntactic Recognition of Abnormalities in the Electroencephalogram, University of Manitoba, Winnipeg, Manitoba R3T 2N2, B.Sc.E.E. Thesis, 1981

2) MAKHOUL, J., Linear Prediction: A Tutorial Review, Modern Spectrum Analysis, IEEE PRESS, 345 East 47 Street, New York, NY 10017, 1978

3) ANDERSON N., On the Calculation of Filter Coefficients for Maximum Entropy Spectral Analysis, Modern Spectrum Analysis, IEEE PRESS, 345 East 47 Street, New York, NY 10017, 1978

4) BURG J.P., Maximum Entropy Spectral Analysis, Modern Spectrum Analysis, IEEE PRESS, 345 East 47 Street, New York, NY 10017, 1978

5) NADIM G.E.E., Data Compression Techniques as Applied to the Electroencephalograph Signal, University of Manitoba, Winnipeg, Manitoba R3T 2N2, M.Sc. Thesis, 1982

6) Motorola, MC68000 16-Bit Microprocessor User's Manual, Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1982

7) Motorola Semiconductors, MC68000 16-Bit Microporcessing Unit, Motorola, 3501 Ed Bluestein Blvd., Austin, Texas 78721, 1979

8) Motorola, MC68000 Educational Computer Board User's Manual, Motorola Inc., 1711 W. 17th Street, Tempe, Arizona 85281, 1982

9) Analog Devices, AD583 IC Sample and Hold Gated Op Amp, Analog Devices, Data Acquisition Databook 1982, One Technology Way, P.O. Box 280, Norwood, MA 02062

10) Analog Devices, ADC80 12-Bit Successive Approximation Integrated Circuit A/D Converter, Analog Devices, Data Acquisition Databook 1982, One Technology Way, P.O. Box 280, Norwood, MA 02062

11) Motorola, M68000 Cross Macro Assembler Reference Manual, Motorola Inc., 1711 W. 17th Street, Tempe, Arizona 85281, 1979

12) BOHLIN T., Analysis of EEG Signals with Changing Spectra Using a Short-Word Kalman Estimator, Mathematical Biosciences 35, 221-259,1977

13) ARBEZ G.J.P., EEG Pattern Recognition System Software, University of Manitoba (Elec. Eng. Dept.), Winnipeg, Manitoba R3T 2N2, 1984

Appendix A

HARDWARE ADDITIONS AND MODIFICATIONS

This appendix describes hardware additions and modifications made to the two ECB Motorola boards. The ECB User's manual (8) describes the boards and their functions in detail.

A.1 ADDRESS DECODING FOR THE WIRE WRAP BOARD

Figure 25 shows the address decoding circuit for the wire wrap board. A decode signal E1 from the ECB is low when addressing the address segment $20000 - $2FFFF. When address lines A7 to A15 and E1 are low the 74LS138 decoder is enabled accessing the region $20000-$2007F. A3 to A5 decoded by the decoder divides this region into 8 byte segments as

TABLE 10

Decoding for the Wire Wrap Board

Decoder Output	Address Segment
Y0	$20000-$20007,$20040-$20047
Y1	$20008-$2000F,$20048-$2004F
Y2	$20010-$20017,$20050-$20057
Y3	$20018-$2001F,$20058-$2005F
Y4	$20020-$20027,$20060-$20067
Y5	$20028-$2002F,$20068-$2006F
Y6	$20030-$20037,$20070-$20077
Y7	$20038-$2003F,$20078-$2007F

shown in Table 10.

The 74LS138 provides active low signals. Address lines A1 and A0 with the control signals R/$\bar{W}$, $\overline{UDS}$, and $\overline{LDS}$ are used for accessing device registers.

The 3-input OR gates provide seven data acknowledge signals. These signals are active high. The acknowledge signal is clocked using a 74LS74 D flip flop by the system clock to meet processor specs (6).

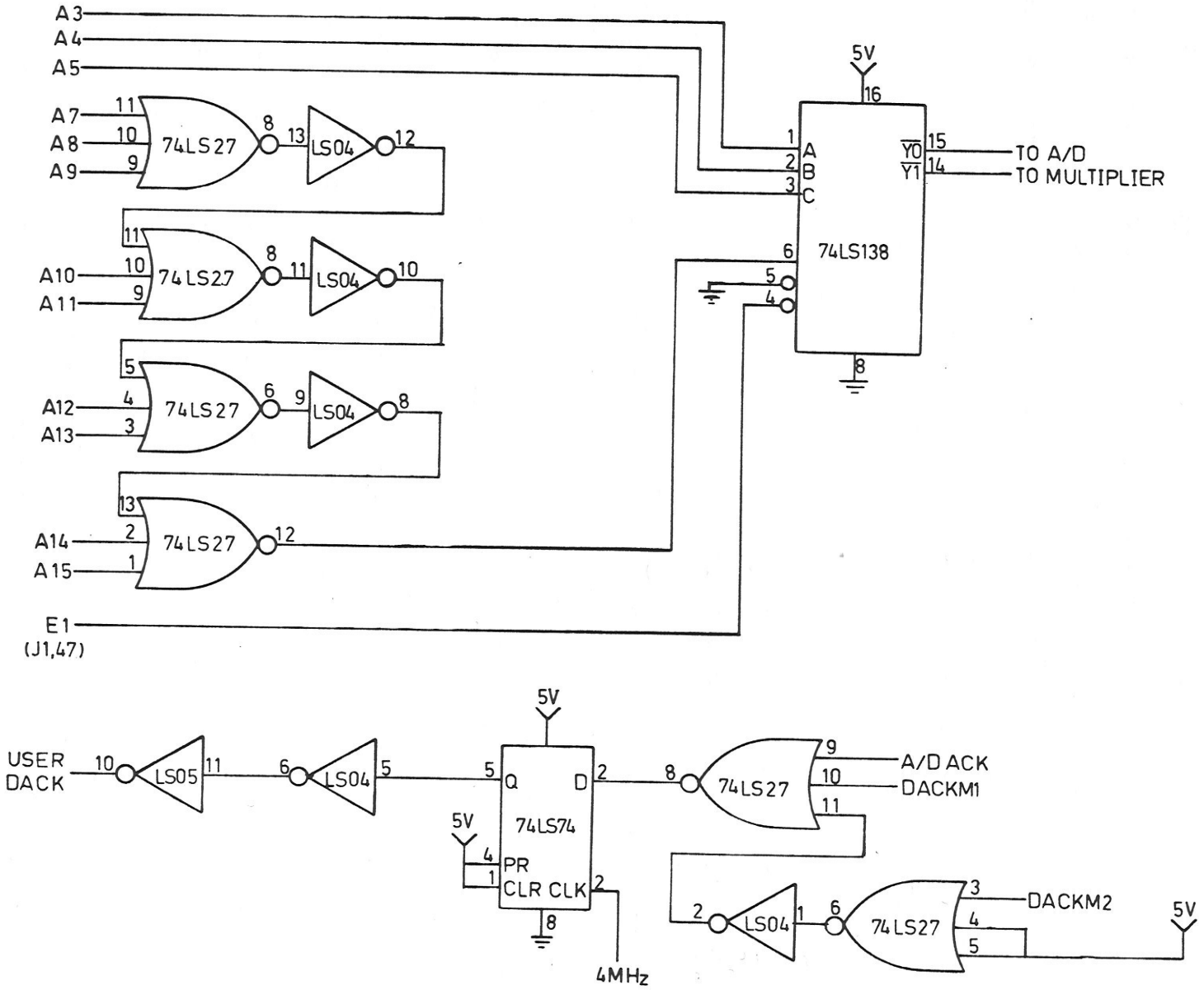

Figure 25: Decoding Circuit for the Wirewrap Board

A.2 A/D CONVERTER

The EEG signal is assumed to be bandlimited to 50 Hz. The playback signal (20 times faster) is then limited to 1000 Hz. A sampling rate of 2.56 Kz provides 128 samples equivalent to 1 second of the real EEG signal.

Digitizing the signal to 12 bits requires a maximum sample and hold (S/H) aperture time of 77ns.

$$t = \beta / \pi f$$

β - resolution of the A/D = $1/2^n$

f - maximum frequency of the signal

The ADC583 S/H (9) meets the specifications with an aperture time of 50 ns. It will allow an increase in the band of the signal to 1550 Hz, that is an EEG bandlimited signal to 78Hz. A successive approximation A/D converter was chosen for speed. The ADC80 (10) digitizes within 25 microseconds and offers a two's complement notation. This desired option free's the processor from scaling the data.

Four sections compose the A/D system (Figure 26): A/D converter, timing circuit, interface to the 68000, and the interrupt edge sensing circuit.

Digitization is initiated by a 10 microsecond clock pulse generated by a 555 timer. When the clock signal asserts the A/D CONVERT START signal, the STATUS signal also is asserted but the digitization does not start until the CONVERT START

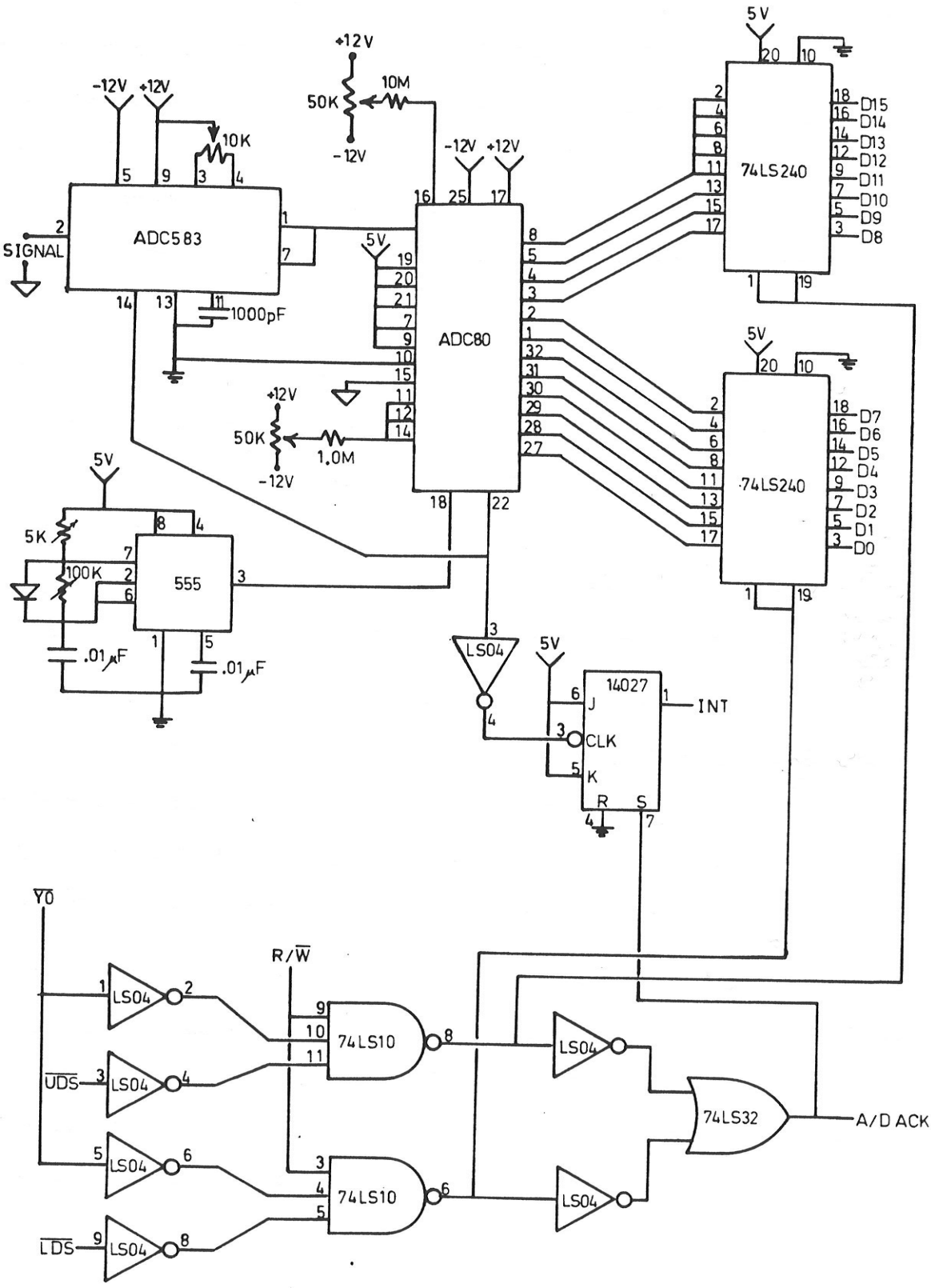

Figure 26: A/D Converter Circuit

signal goes low (Figure 26). STATUS remains high until the digitization is complete. Connecting the A/D STATUS signal to the S/H CONTROL signal allows the S/H to track the signal until a sample is needed. The 10 microsecond pulse provides the required settling time for the S/H before successive approximation is initiated.

R_1 of the S/H is used to eliminate any dc offset. Zero adjust of the A/D converter is provided by R_2 while R_3 adjusts the converter gain. Connecting pins 14 and 12 to pin 11 of the A/D configures the device for an input range of $\pm$2.5 volts. The two's complement notation is obtained by using the signal BIT 1 MSB as the most significant bit.

The output lines provide the digitized data after conversion until the next sample is taken. The buffers (74LS240) enable the 68000 processor to read the results. The enable signals EL and EU are generated using signals $\overline{Y0}$, R/$\overline{W}$, $\overline{UDS}$ and $\overline{LDS}$ (Figure 26). Data can be read at $20002.

A 14027 J/K flip flop monitors the STATUS signal for a falling edge asserting the 68000 INT signal line with a low. The flip flop is reset by ADACK when the data is read by the processor. The 68000 is interrupted only when data is available freeing it from any A/D tasks.

A.3 THE HARDWARE MULTIPLIER

The TRW TDC1010J 16 bit multiplier contains two 16 bit input registers and a 35 bit output register used for accumulation. Data is clocked into the input registers by CKX and CKY (Figure 22). Signals TSX, TSM, and TSL will ouptut the upper 3 bits, bits 16 to 31, and the lower 16 bits (0-15) of the output register respectively when asserted (set low). The results of the multiplication are clocked or accumulated into the output register by the CKP signal. Control signals ACC, SUB, RND, and TC (see chapter 3) configure the multiplier and are clocked in by CKX and CKY.

Y1 , A2, and A1 are used to produce EN1, EN2, EN3. These signals with R/$\bar{W}$, $\overline{UDS}$, and $\overline{LDS}$ will produce each of the control signals described in the above paragraph (Figure 27) at the addresses shown in Table 11. Delaying the signal CKY using two CMOS buffers (14050) produces the CKP clock. The delay allows multiplication to be done before the results are transfered to the output register. When addressing the XTP register, two octal buffers (74LS244) are used to sign extend the 3 bits to 16 bits.

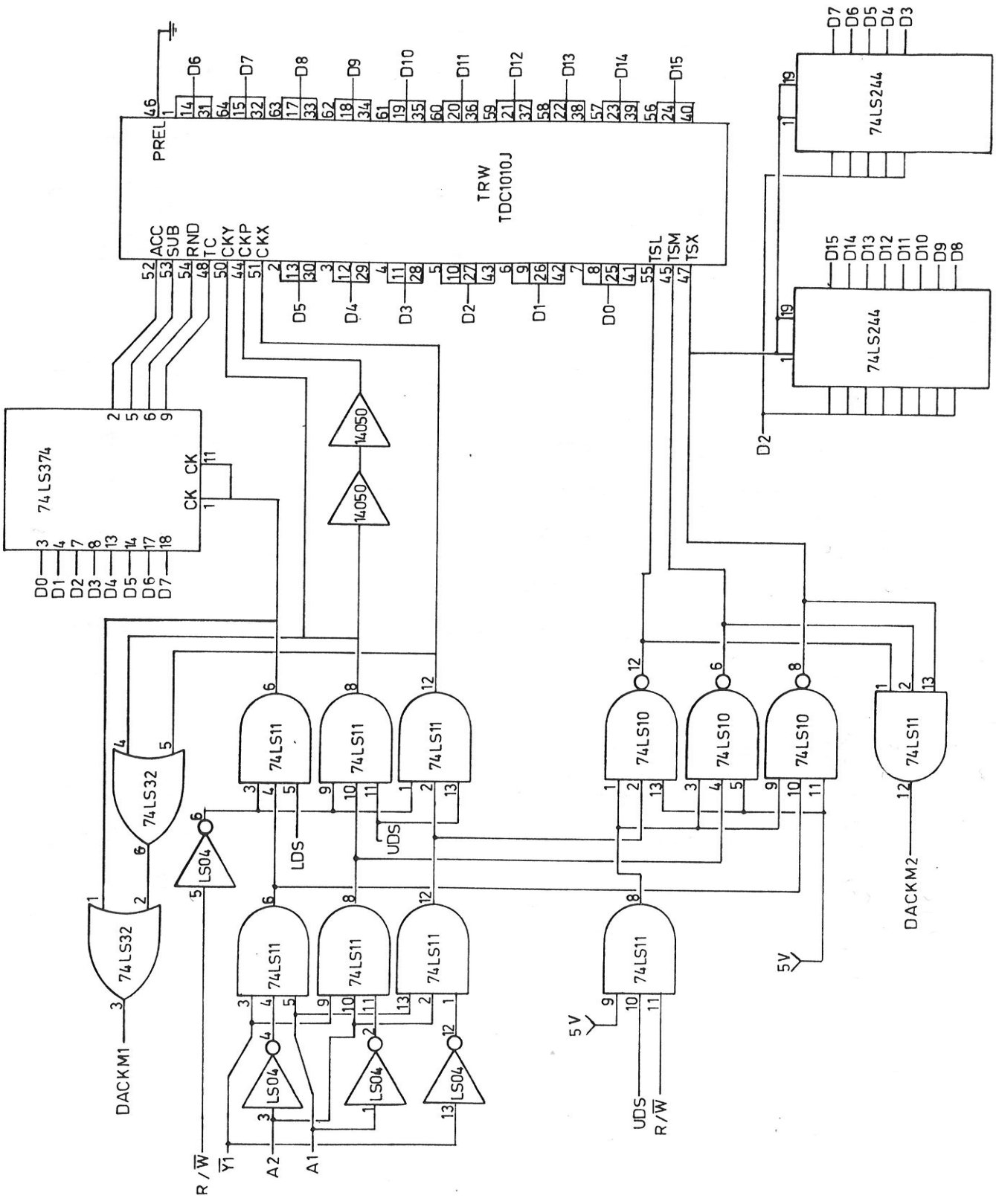

Figure 27: Interfacing the TDC1010J Hardware Multiplier to the 68000 system

TABLE 11

Address Decoding for the Hardware Multiplier

Address	Read	Write
$2000E	TSL	CKX
$2000C	TSW	CKY,CKP
$2000B	-	CKC
$2000A	TSX	-

A.4 STATIC RAM OF THE FE BOARD

The FE board dynamic RAM is replaced by sixteen 2048 (2kX8) static RAM chips. The RAMEN signal from the ECB board is used along with the $\overline{UDS}$ and $\overline{LDS}$ to enable the 74LS138 eight bit decoders which decode A12, A13, A14. The addressed RAM device is enabled using the outputs of the decoders.

The FE board's refresh circuit is disabled by isolating it from the processor bus request signals. Cutting trace to pin 8 of U46 disconnects the BR signal which is then tied to high. The 1 MHz clock clocking the BGACK signal is disabled by cutting the trace leading to pin 3 of U46.

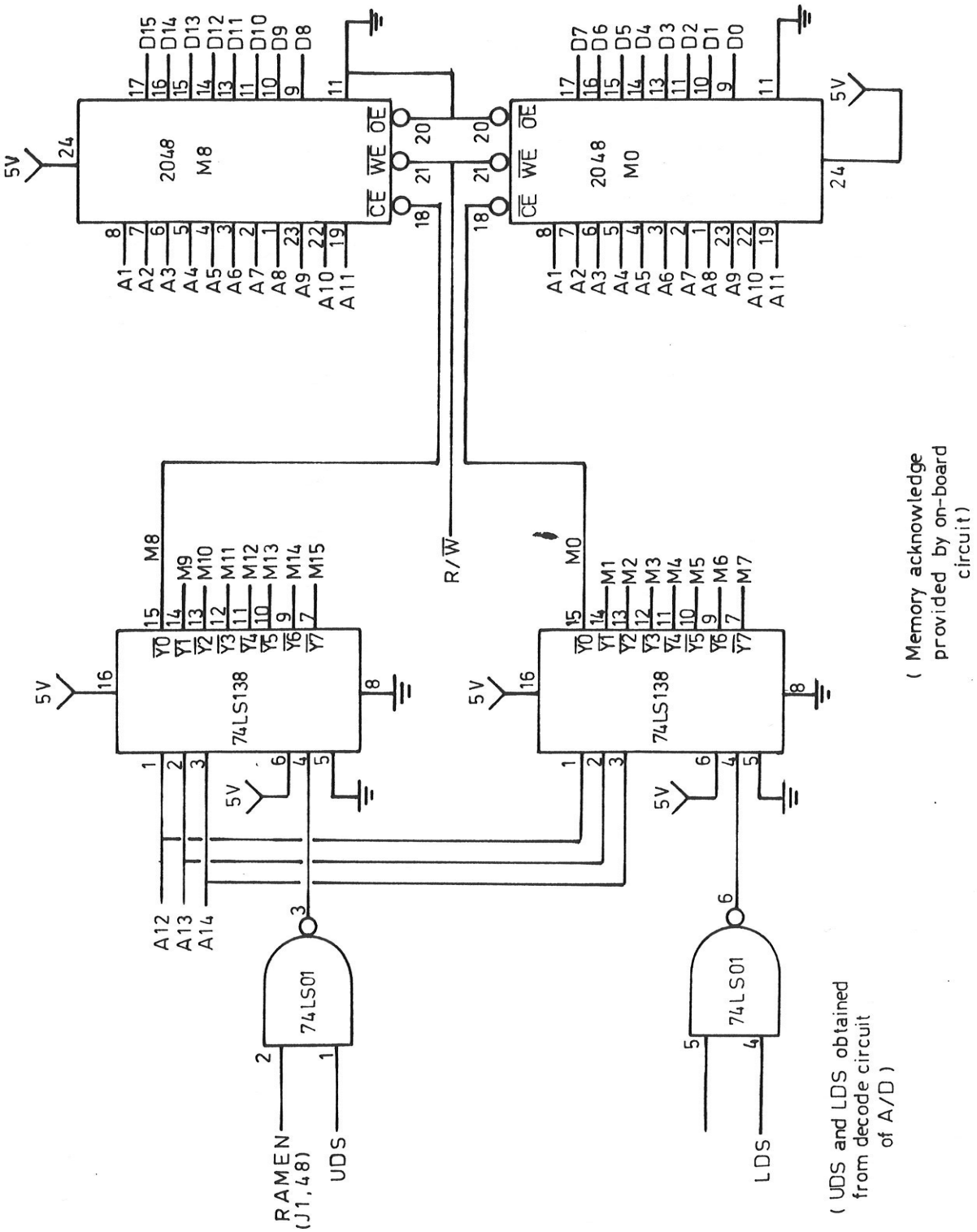

Figure 28: FE Board Static RAM

A.5 ADDING EPROMS

On both ECB boards, 2732 EPROMs are interfaced to the system using the wirewrap areas. Unused gates available on the ECBs are used for address decoding. The signals $\overline{Y0}$, A13, A14, A15, and R/$\bar{W}$, places the EPROMs in the address segment $C000 - $DFFF (Figure 30). These signals were obtained on the board as follows:

Y0 - pin 10 of U32

A13 - pin 6 of U33

A14 - pin 6 of U35

A15 - pin 5 of U32

R/W - pin 5 of U33

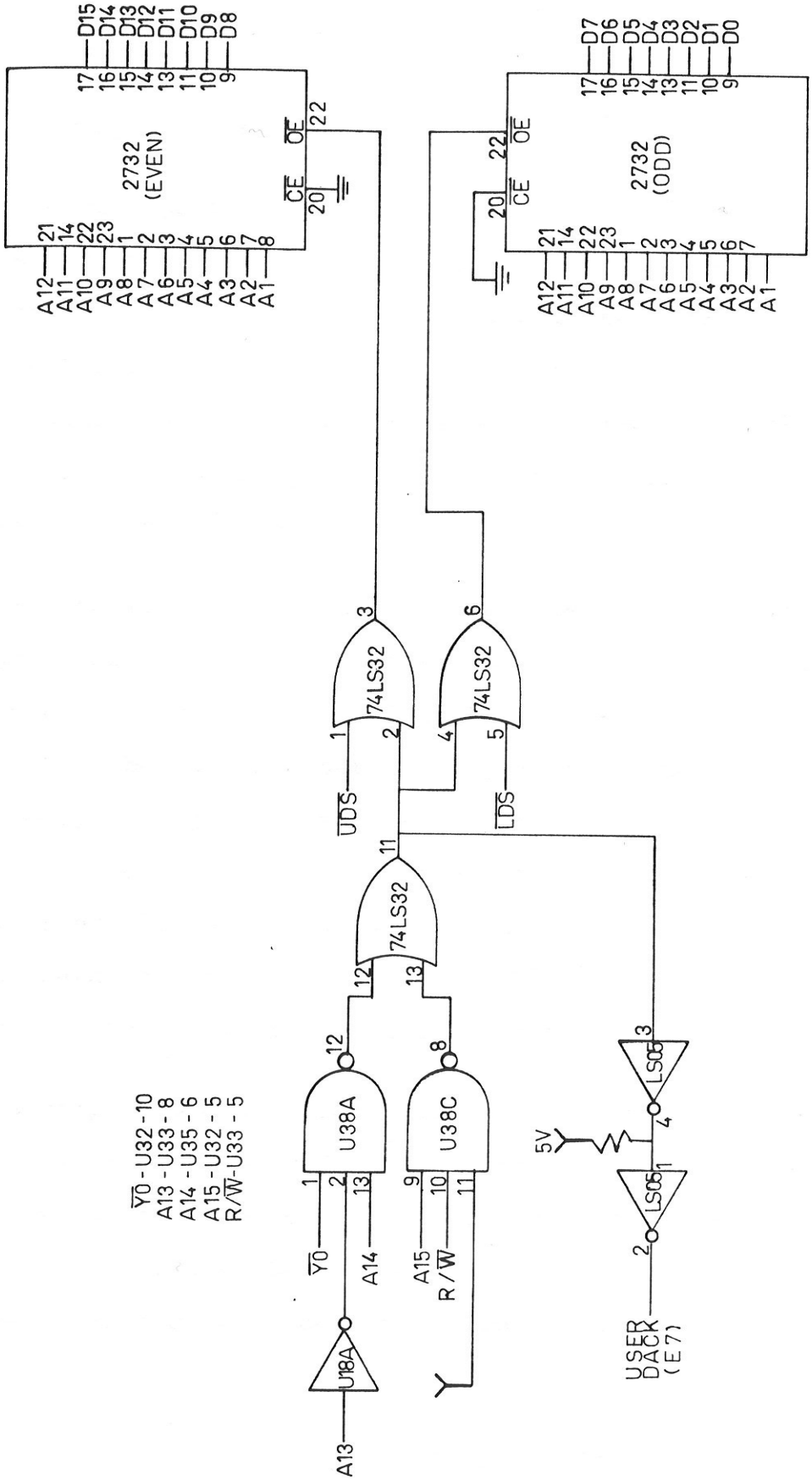

Figure 30: Interfacing EPROM to the ECB

A.6 PARALLEL COMMUNICATIONS

The PI/T port B parallel ports of each board are used to transmit data from the FE board to the TC/SA board. The FE board port is configured as a double buffered output port while the TC/SA port as a double buffer input port. The two handshake lines connected as shown in Figure 31 using an interlocked handshake protocol to transfer data (see the data sheet for the MC68230).

To provide proper connection of the control signals, the pins 1 and pin 5 of the edge connector of the FE board are exchanged. The signal H4 of the FE board (and thus H3 of the TC/SA board) is asserted (high) when data is loaded into the output register and thus present at the output pins. The TC/SA device will load the data into the data register and assert its H4 signal which then indicates to the FE port that data has been transfered. The transfer takes 300 ns giving a byte transfer rate of 3.33 MHz.

A simple communications protocol is used since error free transmission is assumed. The first word transmitted if positive is treated as a count indication of the number of bytes in a record being transmitted. If negative, it is treated as a control command. Three commands have been defined:

COMSTR $FF00 (start processing),
COMEND $FF01 (end processing),
COMERR $FF02 (error detected).

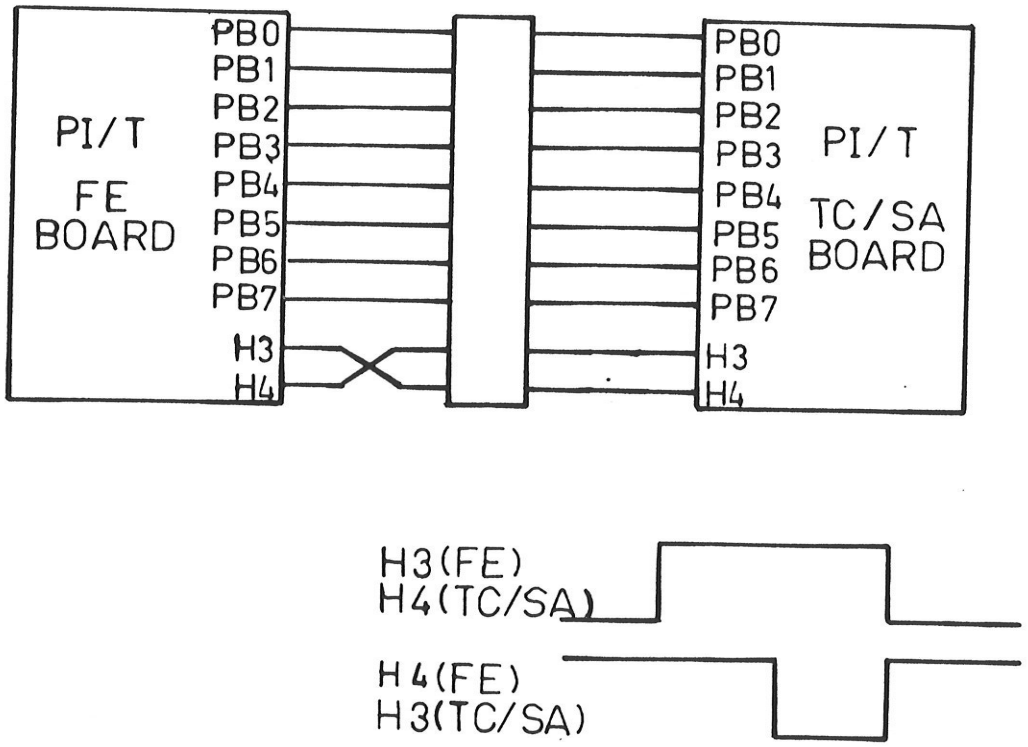

Figure 31: Parallel Communications

COMSTR indicates to the TC/SA board that feature extraction has started, while COMEND indicates that processing has ended. When an error in the feature extraction is detected, then COMERR is transmitted to the TC/SA board.

Appendix B

SYSTEM SOFTWARE

This appendix contains four sections describing the labels used and the software subroutines developed for the pattern recognition system. The first two sections give the list of labels and subroutine descriptions for the FE subsystem. The other two sections give the list of labels and subroutine descriptions for the other two subsystems.

The subroutine descriptions include initialization needed before the subroutine is called, the registers and variables that are used by the routine and any other subroutine it may call. For more detailed information of the system software consult the 'EEG Recognition System Software' manual (13).

B.1 FE SOFTWARE LABEL LIST

N (128)	Number of data points (data buffer size).
N1 (127)	N - 1.
DTBUF1 ($3000)	Address of the first data buffer.
DTBUF2 ($3100)	Address of the second data buffer.
HMWD	Address of the hamming window coefficients.
ACOEFF ($3A00)	Address for storing autocorrelation coefficients in the communications memory.
FEAT ($4000)	Address of the buffer containing the feature set.
P ($4000)	Address of the system model order.
AI ($3300)	The address of the array a(j,i) for DURBIN.
AR ($4004)	The address of the AR coefficients.
GSQR ($4002)	Address for storing of G^2.
B1ADD ($3500)	Address of the array b1.
B2ADD ($3600)	Address of the array b2.
FXPT (4)	Value used to determine where the decimal point is located in the 16 bit fixed point numbers.
AMMADD ($3700)	Address of the array a(m,k) used by BURG.
COSIW	Address of the values $1/2\cos(i\omega T)$ used in computing the PDS.
PWCOS	Address of the values $1/2(1-\cos\omega T)$ for PDS computation when using the KZAR model.
PDS ($3A00)	Address of the power density spectrum.
ROW ($3300)	Address of the array $\gamma(i)$.

```
BFLNGT (N*2)    Length of data buffers in bytes.
TUTOR (228)     Trap #14 function number - return to monitor.
GETNUMD (225)   TRAP #14 function number - convert decimal
                ASCII number to hex.
OVERR (0)       Trap number if buffers overflow.
VECINT ($70)    Interrupt vector location for (INT) (data
                acquisition).
VECOVF ($1C)    Interrupt vector location for the TRAPV
                instruction.
VECERR ($80)    Interrupt vector location for the TRAP #0
                instruction.
VECDIV ($14)    Interrupt vector location for the divide by
                zero trap.
VECBOV ($84)    Interrupt vector location for the TRAP #1
                instruction.
COMSTR ($FF00) Communication control word - start processing.
COMEND ($FF01) Communication control word - stop processing.
COMERR ($FF02) Communication control word - error encountered.
BUFF ($7000)    Buffer used in communications with terminal.
ONE ($1000)     Value representing 1.0.
MINONE ($F000) Value representing -1.0.
STKPNT ($8000) Value for initializing the system stack pointer.
ADDATA ($20002) Address of output of A/D system.
PORT1 ($10040)  Address of the terminal acia.
ATSCL ($FF0)    Location for saving autocorrelation scale value.

**************Address of multiplier registers**************

Input registers:  XIN (4) , YIN(2) , MLSTAT (1).
```

Output registers: XTP (0), MSW (2), LSW (4).

Multiplier base address: MULT ($2000A).

**********Address of PI/T registers**********

PITADD ($10001)	Base address of device.
PGCR (0)	Port control register.
PSQR (PGCR+2)	Port service request register.
PADDR (PSQR+2)	Port A data direction register.
PBDDR (PADDR+2)	Port B data direction register.
PCDDR (PBDDR+2)	Port C data direction register.
PIVR (PCDDR+2)	Port interrupt vector register.
PACR (PIVR+2)	Port A control register.
PBCR (PACR+2)	Port B control register.
PADR (PBCR+2)	Port A data register.
PBDR (PADR+2)	Port B data register.
PCDR (PBDR+2)	Port C data register.
PSR (PCDR+2)	Port status register.
TC ($10020)	Address of timer.

Note that the multiplier and PI/T registers are addressed by using the variables as offsets from the base addresses (except for TC).

B.2 FE SUBROUTINES

GETCHR: Obtains a character from the terminal. The character is returned in the lower byte of register D0.

Register used: D0.

Variables used: PORT1.

DMPCHR: Transmits the character found in the lower byte of the register D0 to the terminal.

Initialization: D0 - character to be transmitted.

Registers used: D0.

Variable used: PORT1.

SNDMES: Transmits a message to the terminal. The message in memory should be terminated with the byte $FF.

Initialization: A5 - address of the message.

Registers used: D0, A5.

Subroutines used: DMPCHR.

MEAN: This subroutine computes the mean of an N word array and subtracts that mean from the elements of that array.

Initialization: A0 - address of the N-word array (preserved).

Registers used: A1, D0, D1, D2.

Variables used: N

Note that division by 128 is performed by a shift operation. If N is changed, so must the shift operation.

HAMWD: The data array of N elements is passed through a hamming window.

Initialization: A0 - address of the N-word array (preserved).

Registers used: A1, A2, A3, D0, D1.

Variables used: HMWD, MULT ,MLSTAT, XIN, YIN, MSW, FXPT.

KZDAT: This subroutine takes the data array and passes it through the inverse of the zero at dc.

Initialization: A0 - address of the N-word array (preserved).

Registers used: A1, D0, D1.

Variables used: N.

AUTCR1: The p+1 autocorrelation coefficients are computed and stored in the communications memory at address ACOEFF. The results are scaled down such that $r(0) < 1.0$. The scale value (number of shifts) is stored in ATSCL.

Initialization: A0 - address of the N-word data buffer (preserved).

Registers used: A1 - A5, D0, D1, D2, D3.

Variables used: MULT, MLSTAT, N, YIN, XIN, ACOEFF, P, ATSCL.

DURBIN: Using the autocorrelation computed by AUTCR1, DURBIN calculates the AR coefficients and the value G^2. The number of coefficients is determined by the value stored at

location P. The results are stored in the feature set array.

Initialization: A0 - address of data buffer (preserved).

Registers used: A1 - A5, D0 - D6.

Variables used: AI, ACOEF, P, AR, GSQR, MLSTAT, MSW, XIN, YIN, MLSTAT, ATSCL, FXPT, ONE, MINONE.

Subroutines: AUTCR1.

BURG: Using BURG's algorithm, the AR coefficients (number determined by model order found at address P) and the value G^2 are calculated and stored in the feature set.

Initialization: A0 - address of data buffer (preserved).

Registers: A1 - A5, D0 - D6.

Variables: MULT, XIN, YIN, MLSTAT, MSW, B1ADD, B2ADD, AMMADD, N1, AR, GSQR, N, FXPT.

PWDEN: From the AR coefficients the power density spectrum is calculated and stored in the communications memory for further computation of spectral features by the routine FEATRS.

Initialization: AR coefficients and G^2 should be computed and stored in the feature set memory at locations AR and GSQR respectively.

Registers: A1-A5, D0-D3.

Variables: MULT, MSW, XIN, YIN, P, AR, ROW, GSQR, PDS, COSIW, MLSTAT, ONE.

PWKZR: This routine calculates the power density spectrum when the KZAR model is used. It calls PWDEN and then multiplies the results by the factors $1/2(1-\cos\omega T)$.

Initialization: AR coefficients and G^2 as in the case of PWDEN.

Registers: A1-A5, D0-D3.

Variables: MULT, MSW, XIN, YIN, PDS, PWCOS.

Subroutines: PWDEN.

FEATRS: This routine will calculate from the power density spectrum the spectral features and store them in the feature array.

Initialization: The power density spectrum addressed by PDS.

Registers: A1-A4, D0-D5.

Variables: FEAT, PDS, P.

Subroutines: SUM.

SUM: The subroutine SUM computes power in frequency bands for the subroutine FEATRS.

Initialization: A1 - the address of the lowest frequency value in the band.

D1 - the number of elements in the band to be summed.

Registers: A1, A2, D0, D1, D2.

Note: The 32 bit result is returned in register D0.

ACQUIS: This routine is invoked when the A/D interrupts the processor. Data from the convertor is stored in the data buffer addressed by A6. ACQUIS will automatically switch buffers when one has been filled.

Initialization: Interrupt vector at address $70.
A6 must contain address of first buffer to be filled.
Interrupt mask cleared to accept A/D interrupts.

Registers: A6, D7.

Variables: ADDATA, DTBUF1, DTBUF2, BFLGTH.

INTOUT: Initializing the PI/T 68230 device allows parallel communication with the TC/SA board. Port B is set as an output port.

Registers: A0, D0.

Variables: PITADD, PGCR, PBDDR, PBCR.

TRNREC: A record, such as the feature set, is transmitted to the TC/SA board using a simple protocol (see appendix A).

Initialization: D0 - number of bytes in the record.
A2 - Address of the record to be transmitted.

Registers: A1, A2, D0, D1.

Variables: PITADD.

Subroutines: TRNBYT, TRNWRD.

TRNWRD: The 16 bit word located in register D0 is transmitted to the TC/SA board. The register D0 is unaltered. The high order byte is first transmitted before control is passed to the routine TRNBYT to transmit the low order byte.

Initialization; D0 - Word to be transmitted.
A1 - Address of PI/T device (preserved).

Variables: PSR, PBDR.

TRNBYT: A single byte, in low order position of register D0, is transmitted to the TC/SA board.

Initialization: D0 - The byte to be transmitted.
A1 - Address of the PI/T device (PITADD).

Variables: PSR,PBDR.

INTCLK: The timer in the PI/T is initialized by loading the count register with $FFFFFF. The clock is disabled first. The clock can be started by loading $11 into the time control register.

Register: D0, A1.

Variables: TC.

STPCLK: This routine disables the clock and loads the timer value into register D0.

Registers: A1, D0.

Variable: TC.

B.3 TC/SA SOFTWARE LABEL LIST

**********Syntactic Analysis Variables**********

WORBUF	($4100)	Address of working buffer.
TIMBUF	($100)	Offset of time buffer from working buffer.
RECBUF	($5000)	Address of record buffer.
TMRBUF	($1000)	Offset of record time buffer from record buffer.
WBLNG	($100)	Length of working buffer.
TSTCD	($08)	Value of first nonterminal code.
IRREG	($FF)	Value of irregular nonterminal code.
TRCNT	($20)	Number of symbols to be transfered to record buffer.
MXRUL	(4)	Maximum nonterminal production rule merge.

**********Codes for terminals and nonterminals**********

N ($0)	Normal terminal.
S ($1)	Slow activity terminal.
L($2)	Low amplitude activity terminal.
F($3)	Fast activity terminal.
A ($4)	Artifact terminal.
SSW ($5)	Sharp and slow wave terminal.
SPK ($6)	Spike terminal.
TRIRR ($7)	Irregular terminal.
NORM ($8)	Normal nonterminal.

SLOW ($9)	Slow nonterminal.
LOW ($A)	Low activity nonterminal.
FAST ($B)	Fast activity nonterminal.
ARTIF ($C)	Artifact nonterminal.
ISOSSW ($D)	Isolated sharp/slow wave activity nonterminal.
ISOSP ($E)	Isolated spike nonterminal.
PAROX ($F)	Paroxysmal nonterminal.
POLYSP ($10)	Polyspike nonterminal.

**********Terminal Classification Variables**********

WRKARY ($4500)	Address of discriminant feature array.
FEAT ($4000)	Address of feature set.
TERM1 (26)	Offset used to address power features in the feature set.
TERM2 (38)	Offset used to address power ratios in the feature set.
TERM3 (60)	Offset indicating end of feature set.
CRMTAD ($4600)	Correction matrix address.
NEWWGT ($4A00)	Weight matrix address.
MXCNT ($80)	Maximum count before updating weight matrix.
SAVCLS ($FF2)	Saving class number while restoring registers.
MSKSCL ($FF6)	Location to save masking scale value.

**********Other variables**********

TBSTR ($4400)	Start of circular buffer.
TBEND ($4500)	End of circular buffer.
STATUS ($FF0)	Address of status byte.

STKPNT ($8000) Value for initializing stack pointer.

COMSTR ($FF00) Communication control word - start processing.

COMEND ($FF01) Communication control word - stop processing.

COMERR ($FF02) Communication control word - error encountered.

TUTOR ($228) Trap #14 function number - return to monitor.

BGTRBF ($7000) Start address of second terminal buffer.

ENTRBF ($7E10) End address of second terminal buffer.

BFAD ($FF4) Location for storing second terminal buffer address.

FXPT (4) Value used to determine where the decimal point is located in the 16 bit fixed point numbers.

PITADD, PGCR, PSQR, ... Variables for addressing PI/T registers (see the FE variable list).

B.4 TC/SA SUBROUTINES

MASK: This subroutine will mask the feature set using a mask array. The power band values being 32 bits long are first examined to see if any of the discriminant ones are larger than 16 bits. If so, a shift count is evaluated such that all values shifted will occupy 16 bits including the power band frequencies. The number of shifts is saved in MSKSCL. Scaling will not affect classification results as all features are scaled.

Initialization: Feature set stored at address FEAT.

Registers: D0-D3, A0-A3.

Variables: FEAT, MSKARY, TERM1, TERM2, TERM3, WRKARY, MSKSCL.

CLASS: Using the discriminant features and the weight matrix, epoch classification is done by CLASS which returns the class value in register D0.

Initialization: A0 - Address of discriminant feature array.

A1 - Address of weight matrix

Registers: D0-D5, A0, A1, A3-A4.

Variable: MSKSCL.

INTCOR: The correction matrix in initialized by setting the values of MAXCNT and SHFCNT and zeroing its elements.

Initializing: A1 - address of weight matrix

A2 - address of correction matrix

D0 - value MXCNT

Registers: D0, D1, D2, A1-A4.

CORMAT: After a classification is done using CLASS, invoking CORMAT will update the correction matrix and the weight matrix when enough classifications have occured.

Initialization: D0 - class number

A0 - address of discriminating set

A1 - address of weight matrix

A2 - address of correction matrix

Registers: D0-D3, A0-A5.

INTINP: Port B of the PI/T 68230 is initialized as an input port to accept data from the FE board. To enable interrupts, bit 2 of the port B control register must be set as well as clearing the 68000 interrupt mask. Disabling the interrupt consist of clearing the interrupt bit of the port B control register and/or setting the system interrupt mask such that interrupts from the device is ignored. The device defaults to interrupt vector number 15 (vector located at address $3C). Before allowing the first interrupt, this vector must be initialized.

Initialization: The interrupt vector at $3C.

Registers: A0, D0.

Variables: PITADD, PBDDR, PBCR, PGCR, PBDR.

GETREC: The FE board will transmit a record where the first word (2 bytes) indicate the number of bytes there are to follow. This routine will take the first word and then store the record received in the feature set buffer.

Initialization: A1 - Address of feature set buffer.

Registers: D0, D1, D2, A0, A1.

Variables: PITADD

Subroutines: GETWRD, GETBYT.

GETWRD: This routine loads a word from Port B of the PI/T. The first byte is loaded into the high order byte of the low order word of register D0. Control is then passed to GETBYT to load the second byte in the low order byte of D0.

Initialization: A0 - Base address of PI/T (preserved).

Registers: A0, D0.

Variables: PSR, PBDR.

GETBYT: A single byte from the 68230 port B is loaded into the low order byte of register D0.

Initialization: A0 - Base address of PI/T (preserved).

Registers: A0, D0.

Variables: PSR, PBDR.

SYNANL: This subroutine controls the syntactic analysis process. It monitors the status bit 0 of STATUS. Processing starts when the bit is set and stopped after it is cleared. Results of the analysis are stored in the record buffer and corresponding time buffer.

Initialization: A2, A3 - Addresses of circular terminal buffer.

Registers: D0-D7, A0-A6.

Variables: STATUS, WBLNG, TIMBUF, TRCNT.

Subroutines: INITBUF, MRGTER, MXSTR, ASSIRR, MRGINT, MRGNT, FILLBF.

Macros: GETIM, APPEND.

INITBUF/FILLBF: Registers are initialized before processing is started. D0 (BGPT1) and D1 (BGPT2) are are both initialized to the start of the buffer (0). D2 (ENDPT) is initialized to the value 0, the start of the work buffer. The registers A0 and A1 are loaded with addresses of the work buffer and record buffer respectively. The work buffer is

also filled with terminals as they become available. When calling the subroutine FILLBF, the initialization process is bypassed and the work buffer is filled with terminals.

Registers: A0 - A3, D0 - D4

Variables: WORBUF, RECBUF, WBLNG, TIMBF.

Macros: GETER, SAVTIM.

MRGTER: Identical symbols are merged in the mixed string. A table is used in determining the nonterminal code when the merged symbol is a terminal. The time buffer is updated as merging occurs.

Initialization: A0 - address of work buffer (WORBUF).
D2 - offset pointing to the end of the symbol string.
D3 - offset pointing to the first symbol in the mixed string.

Registers: A0, A4, D0-D6.

Variables: TBMTER, TIMBUF, TSTCD.

Macros: GETIM, SAVTIM, ADRTIM, APPEND.

MXSTR: Production rules are used to merge the mixed string.

Initialization: A0 - address of work buffer (WORBUF).
D2 - offset pointing to the end of the symbol string.
D3 - offset pointing to the first symbol in the mixed string.

Registers: A0, A4, D0-D6.

Variables: MXPROD, TIMBUF.

Macros: LDWRD, GETIM, SAVTIM, ADRTIM, APPEND.

ASSIRR: Isolated terminal codes not merged by the previous two subroutines are assigned the IRREG code ($FF). Then the TERF (D3) pointer is moved to the first terminal of the mixed string.

Initialization: A0 - address of work buffer (WORBUF).
D2 - offset pointing to the end of the symbol string.
D3 - offset pointing to the first symbol in the mixed string.

Registers: A0, A4, D2-D6.

Variables: TSTCD, IRREG.

MRGINT: Identical nonterminals are merged in the nonterminal string. As in all merges, the time buffer is updated.

Initialization: A0 - address of work buffer (WORBUF).
D2 - offset pointing to the end of the symbol string.
D3 - offset pointing to the first symbol in the mixed string.

Registers: A0, D0 - D6.

Variables: TIMBUF.

Macros: GETIM, SAVTIM, APPEND.

MRGNT: Using production rules from the list at address TBMNT, the nonterminal string in the work buffer is reduced always updating the time buffer.

Initialization: A0 - address of work buffer (WORBUF).
D2 - offset pointing to the end of the

symbol string.

D3 - offset pointing to the first symbol in the mixed string.

Registers: A0, A4, A5, D0-D7.

Variables: TBMNT, TIMBUF, MXRUL.

Macros: ADRTIM, SAVTIM, LDWRD, APPEND.

****Syntactic Analysis Macros****

GETER \1

This macro obtains from the circular buffer the next terminal incrementing A3 taking into account the end of the buffer. The byte is saved in the lower byte of the data register "\1".

SAVTIM \1,\2,\3,\4

A symbol epoch time is stored in location \2 + \3 + \4 where \2 is an offset data register, \3 as a base address register, and \4 is a constant offset. \1 is the data register containing the time value to be saved.

GETIM \1,\2,\3,\4

The time value addressed by \2 + \3 + \4 (same definition as for SAVTIM) is loaded into the data register \1.

ADDTIM \1,\2,\3,\4

Here the time count in \1 is added to the time count addressed by \2 + \3 + \4 (same definition as for SAVTIM).

ADRTIM \1,\2,\3,\4

The time value addressed by \2 + \3 + \4 (same definition as for SAVTIM) is added to the value found in register \1.

LDWRD \1,\2,\3,\4

Two bytes are loaded into \1 instead of a single one as in GETER. The address of this word is \2 + \3 + \4 where \2, \3, \4 have the same definition as in SAVTIM.

APPEND

When merging, the symbol string is split into two sections. The APPEND macro will append the end string to the first one. D0 points to the location of the end of the first section of the work buffer to which the end string is to be moved. D1 points to the start of the end string. D2 points to the end of the end string.

Appendix C

ERROR RESULTS

This appendix contains the error calculation made for five different epochs one from each of the following classes: normal, fast, low frequency artifact, spike and slow and spike. Each of the 27 feature values are shown both in hexadecimal and decimal form. The error value is taken as the absolute value of the 68000 result subtracted from the Amdahl result. The percent error shown is calculated from the ratio of the error to the feature value produced by the Amdahl system.

Four epoch error values are also determined: the maximum error value, the maximum percent error, an average error, and an average percent error. The maximum error and maximum percent error are the maximum values found in the last two columns of the tables. The average error is computed by averaging the 27 feature error values. The average percent value is computed by taking the ratio of the sum of the error values to the sum of the feature values generated by the Amdahl system.

NORMAL EPOCH

	AMDAHL HEX	68000 HEX	AMDAHL DECIMAL	68000 DECIMAL	ERROR	PERCENT ERROR
GSQR :	004BF	00484	0.29663	0.28223	0.01440	4.86%
AR1 :	FE842	FE82B	-1.48389	-1.48950	0.00562	0.38%
AR2 :	00CEA	00D51	0.80713	0.83228	0.02515	3.12%
AR3 :	FF9FC	FF8EB	-0.37598	-0.44263	0.06665	17.73%
AR4 :	000CF	0025E	0.05054	0.14795	0.09741	192.75%
AR5 :	00305	0015B	0.18872	0.08472	0.10400	55.11%
AR6 :	00123	00265	0.07104	0.14966	0.07861	110.65%
AR7 :	FFE18	FFD65	-0.11914	-0.16284	0.04370	36.68%
AR8 :	0056F	005D7	0.33960	0.36499	0.02539	7.48%
AR9 :	FF7ED	FF7A0	-0.50464	-0.52344	0.01880	3.73%
AR10 :	002EE	00305	0.18311	0.18872	0.00562	3.07%
POWER BND1:	002E8	0038C	0.18164	0.22168	0.04004	22.04%
POWER BND2:	00399	0041A	0.22485	0.25635	0.03149	14.01%
POWER BND3:	00743	00794	0.45386	0.47363	0.01978	4.36%
POWER BND4:	011C8	01146	1.11133	1.07959	0.03174	2.86%
POWER BND5:	00073	0006C	0.02808	0.02637	0.00171	6.09%
POWER BND6:	000DB	000C6	0.05347	0.04834	0.00513	9.59%
RATIO BND1:	000B5	000D7	0.04419	0.05249	0.00830	18.78%
RATIO BND2:	000E0	000F9	0.05469	0.06079	0.00610	11.16%
RATIO BND3:	001C5	001CC	0.11060	0.11230	0.00171	1.55%
RATIO BND4:	00455	00419	0.27075	0.25610	0.01465	5.41%
RATIO BND5:	0001C	00019	0.00684	0.00610	0.00073	10.71%
RATIO BND6:	00035	0002F	0.01294	0.01147	0.00146	11.32%
MAX POWER :	0045C	00403	0.27246	0.25073	0.02173	7.97%
FRQ OF MAX:	02200	02200	2.12500	2.12500	0.0	0.0%
SLP OF MAX:	FFEBA	FFEC4	-0.07959	-0.07715	0.00244	3.07%
# OF PEAKS:	00003	00007	0.00073	0.00171	0.00098	133.33%

MAXIMUM ERROR VALUE: 0.104004
MAXIMUM % ERROR: 192.753623%
AVERAGE ERROR: 0.024939
AVERAGE % ERROR: 15.568727%

FAST EPOCH

	AMDAHL 68000 HEX		AMDAHL DECIMAL	68000 DECIMAL	ERROR	PERCENT ERROR
GSQR :	003CC	003E8	0.23730	0.24414	0.00684	2.88%
AR1 :	FFB22	FFB3A	-0.30420	-0.29834	0.00586	1.93%
AR2 :	00009	FFFE0	0.00220	-0.00781	0.01001	455.56%
AR3 :	000A1	000BA	0.03931	0.04541	0.00610	15.53%
AR4 :	00180	0018D	0.09375	0.09692	0.00317	3.39%
AR5 :	FFDC5	FFDC4	-0.13940	-0.13965	0.00024	0.18%
AR6 :	0031B	00306	0.19409	0.18896	0.00513	2.64%
AR7 :	FFE6B	FFE71	-0.09888	-0.09741	0.00146	1.48%
AR8 :	FFF27	FFF35	-0.05298	-0.04956	0.00342	6.45%
AR9 :	000DC	000C2	0.05371	0.04736	0.00635	11.82%
AR10 :	FF76E	FF793	-0.53564	-0.52661	0.00903	1.69%
POWER BND1:	0008B	0008D	0.03394	0.03442	0.00049	1.44%
POWER BND2:	0001B	00019	0.00659	0.00610	0.00049	7.41%
POWER BND3:	0000F	0000C	0.00366	0.00293	0.00073	20.00%
POWER BND4:	00080	0007B	0.03125	0.03003	0.00122	3.91%
POWER BND5:	0010D	0010B	0.06567	0.06519	0.00049	0.74%
POWER BND6:	000A3	00093	0.03979	0.03589	0.00391	9.82%
RATIO BND1:	00184	00193	0.09473	0.09839	0.00366	3.87%
RATIO BND2:	0004A	00047	0.01807	0.01733	0.00073	4.05%
RATIO BND3:	00029	00022	0.01001	0.00830	0.00171	17.07%
RATIO BND4:	00164	00160	0.08691	0.08594	0.00098	1.12%
RATIO BND5:	002EC	002FC	0.18262	0.18652	0.00391	2.14%
RATIO BND6:	001C4	001A5	0.11035	0.10278	0.00757	6.86%
MAX POWER :	00075	00070	0.02856	0.02734	0.00122	4.27%
FRQ OF MAX:	03200	03200	3.12500	3.12500	0.0	0.0%
SLP OF MAX:	00014	00016	0.00488	0.00537	0.00049	10.00%
# OF PEAKS:	00003	00005	0.00073	0.00122	0.00049	66.67%

MAXIMUM ERROR VALUE: 0.010010
MAXIMUM % ERROR: 455.555556%
AVERAGE ERROR: 0.003174
AVERAGE % ERROR: 2.571805%

LOW FREQUENCY ARTIFACT EPOCH

	AMDAHL HEX	68000 HEX	AMDAHL DECIMAL	68000 DECIMAL	ERROR	PERCENT ERROR
GSQR :	0094F	009EC	0.58179	0.62012	0.03833	6.59%
AR1 :	FEC5E	FEC72	-1.22705	-1.22217	0.00488	0.40%
AR2 :	00325	00323	0.19653	0.19604	0.00049	0.25%
AR3 :	00222	001E7	0.13330	0.11890	0.01440	10.81%
AR4 :	00253	00292	0.14526	0.16064	0.01538	10.59%
AR5 :	FF50F	FF4DE	-0.68384	-0.69580	0.01196	1.75%
AR6 :	00A02	009DB	0.62549	0.61597	0.00952	1.52%
AR7 :	FFE1C	FFE47	-0.11816	-0.10767	0.01050	8.88%
AR8 :	FFF4D	FFF62	-0.04370	-0.03857	0.00513	11.73%
AR9 :	FFD4D	FFCFF	-0.16870	-0.18774	0.01904	11.29%
AR10 :	0034E	0038F	0.20654	0.22241	0.01587	7.68%
POWER BND1:	02DC2	1770E	2.85986	23.44092	20.58105	719.65%
POWER BND2:	03794	1243C	3.47363	18.26465	14.79102	425.81%
POWER BND3:	00545	004F6	0.32935	0.31006	0.01929	5.86%
POWER BND4:	0021F	001F8	0.13257	0.12305	0.00952	7.18%
POWER BND5:	00141	0013C	0.07837	0.07715	0.00122	1.56%
POWER BND6:	00213	001EA	0.12964	0.11963	0.01001	7.72%
RATIO BND1:	00345	0046E	0.20435	0.27686	0.07251	35.48%
RATIO BND2:	003F9	00373	0.24829	0.21558	0.03271	13.18%
RATIO BND3:	00060	0000E	0.02344	0.00342	0.02002	85.42%
RATIO BND4:	00027	00005	0.00952	0.00122	0.00830	87.18%
RATIO BND5:	00017	00003	0.00562	0.00073	0.00488	86.96%
RATIO BND6:	00026	00005	0.00928	0.00122	0.00806	86.84%
MAX POWER :	01374	07FFF	1.21582	7.99976	6.78394	557.97%
FRQ OF MAX:	00800	00A00	0.50000	0.62500	0.12500	25.00%
SLP OF MAX:	0022E	F9A76	0.13623	-6.34619	6.48242	4758.42%
# OF PEAKS:	00003	00004	0.00073	0.00098	0.00024	33.33%

MAXIMUM ERROR VALUE: 20.581055
MAXIMUM % ERROR: 4758.422939%
AVERAGE ERROR: 1.818359
AVERAGE % ERROR: 545.256365%

SPIKE AND WAVE EPOCH

	AMDAHL HEX	68000 HEX	AMDAHL DECIMAL	68000 DECIMAL	ERROR	PERCENT ERROR
GSQR :	00A25	00B42	0.63403	0.70361	0.06958	10.97%
AR1 :	FE20E	FE40E	-1.87158	-1.74658	0.12500	6.68%
AR2 :	0128E	00E27	1.15967	0.88452	0.27515	23.73%
AR3 :	FFA3F	FFE82	-0.35962	-0.09326	0.26636	74.07%
AR4 :	00482	00118	0.28174	0.06836	0.21338	75.74%
AR5 :	FF7A3	FFB13	-0.52271	-0.30786	0.21484	41.10%
AR6 :	0082F	00386	0.51147	0.22021	0.29126	56.95%
AR7 :	FFD4E	00211	-0.16846	0.12915	0.29761	176.67%
AR8 :	FFECB	FFC75	-0.07544	-0.22144	0.14600	193.53%
AR9 :	000DD	0001F	0.05396	0.00757	0.04639	85.97%
AR10 :	00066	00181	0.02490	0.09399	0.06909	277.45%
POWER BND1:	0A62D	1FFFC	10.38599	31.99902	21.61304	208.10%
POWER BND2:	0E659	1E90F	14.39673	30.56616	16.16943	112.31%
POWER BND3:	01B46	012E3	1.70459	1.18042	0.52417	30.75%
POWER BND4:	0090F	007CA	0.56616	0.48682	0.07935	14.01%
POWER BND5:	00308	00332	0.18945	0.19971	0.01025	5.41%
POWER BND6:	002A6	00252	0.16553	0.14502	0.02051	12.39%
RATIO BND1:	00308	003F6	0.18945	0.24756	0.05811	30.67%
RATIO BND2:	00434	003C9	0.26270	0.23657	0.02612	9.94%
RATIO BND3:	0007F	00025	0.03101	0.00903	0.02197	70.87%
RATIO BND4:	0002A	0000F	0.01025	0.00366	0.00659	64.29%
RATIO BND5:	0000E	00006	0.00342	0.00146	0.00195	57.14%
RATIO BND6:	0000C	00004	0.00293	0.00098	0.00195	66.67%
MAX POWER :	045CF	07FFF	4.36304	7.99976	3.63672	83.35%
FRQ OF MAX:	00A00	00C00	0.62500	0.75000	0.12500	20.00%
SLP OF MAX:	FED3E	FDA11	-1.17236	-2.37085	1.19849	102.23%
# OF PEAKS:	00001	00003	0.00024	0.00073	0.00049	200.00%

MAXIMUM ERROR VALUE: 21.613037
MAXIMUM % ERROR: 277.450980%
AVERAGE ERROR: 1.685511
AVERAGE % ERROR: 144.968969%

SPIKE EPOCH

	AMDAHL HEX	68000 HEX	AMDAHL DECIMAL	68000 DECIMAL	ERROR	PERCENT ERROR
GSQR :	029A3	02DBC	2.60229	2.85840	0.25610	9.84%
AR1 :	FE507	FE607	-1.68579	-1.62329	0.06250	3.71%
AR2 :	0112D	00EF5	1.07349	0.93481	0.13867	12.92%
AR3 :	FF95B	FFB5D	-0.41528	-0.28979	0.12549	30.22%
AR4 :	0029B	0014F	0.16284	0.08179	0.08105	49.78%
AR5 :	FFC24	FFD16	-0.24121	-0.18213	0.05908	24.49%
AR6 :	0052F	00423	0.32397	0.25854	0.06543	20.20%
AR7 :	FFF29	00036	-0.05249	0.01318	0.06567	125.12%
AR8 :	FFB7B	FFAF3	-0.28247	-0.31567	0.03320	11.75%
AR9 :	00554	0053F	0.33301	0.32788	0.00513	1.54%
AR10 :	FFD3B	FFD65	-0.17310	-0.16284	0.01025	5.92%
POWER BND1:	0E8FA	1B6ED	14.56104	27.43286	12.87183	88.40%
POWER BND2:	041DD	0560F	4.11646	5.37866	1.26221	30.66%
POWER BND3:	0312A	03266	3.07275	3.14990	0.07715	2.51%
POWER BND4:	035CE	03380	3.36279	3.21875	0.14404	4.28%
POWER BND5:	0086C	00890	0.52637	0.53516	0.00879	1.67%
POWER BND6:	00C17	00B4D	0.75562	0.70630	0.04932	6.53%
RATIO BND1:	0046C	0056D	0.27637	0.33911	0.06274	22.70%
RATIO BND2:	0013F	00110	0.07788	0.06641	0.01147	14.73%
RATIO BND3:	000EF	0009F	0.05835	0.03882	0.01953	33.47%
RATIO BND4:	00105	000A3	0.06372	0.03979	0.02393	37.55%
RATIO BND5:	00029	0001B	0.01001	0.00659	0.00342	34.15%
RATIO BND6:	0003B	00023	0.01440	0.00854	0.00586	40.68%
MAX POWER :	05783	07FFF	5.46948	7.99976	2.53027	46.26%
FRQ OF MAX:	00000	00200	0.0	0.12500	0.12500	0.0%
SLP OF MAX:	FDE7F	FF9F6	-2.09399	-0.37744	1.71655	81.98%
# OF PEAKS:	00002	00003	0.00049	0.00073	0.00024	50.00%

MAXIMUM ERROR VALUE: 12.871826
MAXIMUM % ERROR: 125.116279%
AVERAGE ERROR: 0.733887
AVERAGE % ERROR: 62.082734%