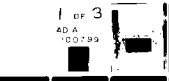
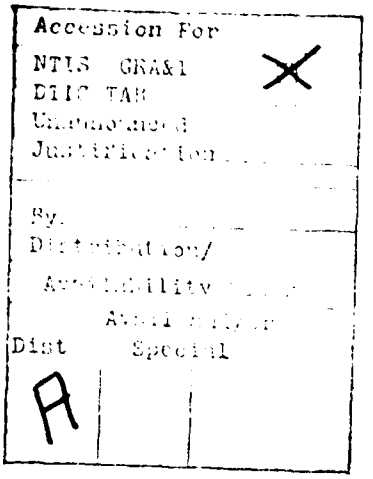
AD-A100 799 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 6/5
ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA C--ETC(U)
DEC 80 M D TOWNSEND
UNCLASSIFIED AFIT/GE/EE/80D-46 NL



AFIT/GE/EE/80D-46



ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA COMPRESSION TECHNIQUES

THESIS

AFIT/GE/EE/80D-46 Melvin D. Townsend
Captain USAF

Approved for public release; distribution unlimited

AFIT/GE/EE/80D-46

ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA COMPRESSION TECHNIQUES

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Melvin D. Townsend, B.S.E.E.

Captain USAF

Graduate Electrical Engineering

December 1980

Acknowledgements

This thesis is the result of a great deal of effort and could not have been accomplished without the assistance and understanding of many people. I first want to thank my fellow students, especially Capt. ''Chip'' Lutz and Capt. Lee Baker, for their assistance in microcomputer inferface problems. Next I would like to thank my thesis committee, and in particular Dr. Rustan my thesis advisor, for their assistance and counsel during this thesis. Finally, I wish to extend my sincerest thanks to my wife, Gail, without whose love and support this thesis would never have come to completion.

Contents

List of Figures

Abstract

EKG data compression techniques were investigated for potential real time implementation on an 8 bit Motorola 6800 microprocessor. Research indicated entropy reduction transform techniques such as the Fast Fourier Transform and the discrete Karhunen-Loeve Transform were not feasible for implementation on the 6800. Two redundancy reduction (RR) techniques (TOLAN and DOWER) utilizing 2nd order difference operations in conjunction with variable length encoding were studied in detail. One such RR technique (TOLAN) was fully implemented and tested with ''in vivo'' EKG data. Analysis revealed compression ratios ranging from 1.25:1 to 2.26:1. Investigation of the poor performance of the compression algorithm showed significant degradation of the 2nd order difference ''decorrelator'' due to a noisy collection environment. It was concluded that real time EKG data compression is feasible on the 6800 but that time compression techniques which store a zero value sequence counter versus the value of zero are not efficient in a high noise environment.

ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA COMPRESSION TECHNIQUES

I. Introduction

Background

Currently the USAF School of Aerospace Medicine (USAFSAM) is receiving more than thirty thousand electrocardiogram (EKG) data records a year from Air Force flight personnel worldwide. This data is presently recorded on paper strip charts for immediate medical analysis and long term storage (via microfilm).

Advances in computerized biomedical analysis has generated a need for digitized storage of the EKG waveform to allow computerized interpretation and comparison of present and past cardiographic data. As computerized diagnosis becomes more accurate, a long term digital record of historical EKG data will enable the cardiology staff at USAFSAM to identify developing heart disease before it becomes a danger to a flight crew member or his fellow crewman.

Data compression of the sampled EKG has been an area of active research since the late nineteen sixties. References (1), (7), (12), (26), (28), (29), (32), (33), and (35) are representative of the research efforts performed in the last ten years. The reasons for compressing EKG data are twofold: 1) digital storage costs are rapidly approaching

analog storage costs and ; 2) increased use of computer aided diagnosis requires large digital data bases.

In the past, EKG data compression has generally been performed at a central computer facility. The data is normally collected in a physician's office or clinic and transmitted in an analog format over a standard telephone link to the central computer. The data is then digitized and input to a computer for diagnosis and storage (in compressed form).

With the current advances in microprocessor technology, sampling and compression at the collection site is now a viable alternative. State of the art digital communication systems can operate at 9600 bits per second over the standard 3 kilohertz (KHz) bandwidth telephone channel. With error detection and correction protocols (Ref 27), high fidelity digital transmission of the compressed EKG to the central processing center appears to be the wave of the future (Ref 21:253-254).

The Department of Defense (DOD) is currently installing a computerized EKG interpretation system at centralized US military medical centers worldwide. To collect and transmit this EKG data, remote medical clinics will use a commercial EKG ''cart'' containing a Motorola 6800 microprocessor. This EKG cart performs internal data compression, record formatting, and error protective ''channel'' encoding for digital transmission to the central medical center.

As an independent study, USAFSAM has sponsored this

thesis to investigate the general field of microprocessor based EKG data compression. The results of this study are to be used for comparison against the data compression achieved in the DOD system and to create a measurement baseline to evaluate future microcomputer based EKG data compression systems.

Problem

The problem addressed by this thesis was the analysis of currently available EKG data compression algorithms for implementation in a microprocessor based computational enviroment. Additionally, a performance measure was to be developed by which differing EKG data compression techniques could be compared.

Scope

In this thesis, an EKG Data Acquistion and Analysis System (EKG-DAAS) was constructed. The EKG-DAAS collects 3 leads of an amplified EKG (i.e., Analog/Digital (A/D) dynamic range $=\pm$ 5 volts), and samples the data at a operator controlled rate between 300-700 hertz. Data is digitized at 12 bit precision and subsequently rounded to 8 bits for uniform truncation error performance (Ref 24:413-418).

The EKG-DAAS hardware was developed around a Motorola Exorciser microcomputer (6800 microprocessor) with associated A/D converter, disk memory,computer terminal, and

hard copy printer. The EKG-DAAS hardware is controlled via a software program called EKG-EXEC. EKG-EXEC is an assembly language program which performs terminal, printer, and disk input/output (I/O) operations as well as providing a supporting structure for the data compression and analysis software. Because of the lack of a high order language (e.g. FORTRAN, PASCAL, etc.) and the desire for maximum program execution speed, all programming done in this thesis is in assembly language. This has proven to be a major limitation.

Only one EKG data compression algorithm was completed and implemented on the EKG-DAAS. ''In vivo'' EKG data was taken, however, and EKG data compression performance analyzed.

General Approach

To accomplish the objectives in the problem statement, the literature was first searched for EKG compression algorithms whose implementation and execution on a 6800 microprocessor was considered feasible. As a result of this literature search and private correspondence (Ref 11 and 31), two compression routines were found. These two EKG data compression algorithms will hereafter be referred to as the Tolan and Dower methods (Ref 31,12).

With the Tolan and Dower algorithms identified, construction of the EKG-DAAS was begun by assembling the data acquistion subsystem. This data acquistion system

consists of an Analog/Digital converter and a interrupt sampling clock. The A/D was calibrated and coding of the EKG-EXEC software started. Several months of effort resulted in the EKG-EXEC software and programming of the Tolan compression algorithm initiated. Upon completion of the Tolan compression routines, the Dower algorithm was analyzed but not implemented.

Based on the results of the Tolan compression algorithm, a performance measure was formulated which compares achieved compression against an approximate maximum compression computed from the data statistics. The Dower compression algorithm was analyzed for similarity with the Tolan technique and an estimated performance figure calculated with respect to the compression measure or ''metric''.

Sequence of Presentation

Chapter 2 begins the thesis development with a general survey of the field of data compression. Data compression is shown to be divided into two subclasses (Entropy Reduction and Redundancy Reduction) and each subclass is defined. Several EKG compression techniques found in the literature were included in the ER category. These ER techniques were described and their performance advantages and limitations analyzed. Chapter 2 proceeds by describing redundancy reduction and several RR electrocardiogram compression algorithms are also analyzed. Chapter 2

concludes with the determination that implementation of the identified ER compression routines would require programming efforts beyond the scope of this thesis. Hence this thesis is limited to redundancy reduction compression.

Chapter 3 describes, in detail, the Tolan and Dower redundancy reduction algorithms whose implementation on the Exorciser was considered feasible. The Tolan algorithm is described first followed by a discussion of the Dower method. Two other EKG compression techniques, discovered during the research of this thesis, are also summarized.

Chapter 4 reviews the hardware and software configuration of the EKG-DAAS. The hardware system is described first and the specifications of the Exorciser microcomputer, A/D converter, and I/O peripherals are presented. Following the hardware description, the EKG-EXEC program is documented and the software design philosphy examined. Finally the Tolan compression module is discussed.

Chapter 5 outlines the results obtained using the EKG-DAAS with the Tolan algorithm. A detailed description of the experimental setup is presented along with a discussion of the performance parameters measured by EKG-EXEC. Finally a performance metric is described and the Tolan compression results compared against this metric. Chapter five ends with an analysis of how well the Dower compression technique would have performed against the performance metric. Chapter 6 concludes the thesis and

presents recommendiations for future study.

This thesis contains five appendices. Appendix A surveys basic electrocardiology and is recommended to readers unfamiliar with this subject. Appendix B is a tutorial on Information and Coding Theory and is likewise recommended to those readers unfamiliar with this field. Appendix C contains a listing of the EKG-EXEC assembly language software as well as one BASIC prcgram used on the Exorciser for data analysis. Appendix D contains data printouts of the data taken in the data collection experiment. Appendix E presents photocopies of the specification sheets for the equipment used in this thesis.

With the sequence of presention outlined, attention now turns to the theoretical section of this thesis. The first subject is data compression theory.

II. Data Compression and the EKG

Data compression is an operation in which data from an information source is ''simplified'' or ''filtered'' in a manner that produces an approximation of the original with at most some predefined amount of distortion. Some form of data compression is usually necessary when storage limitations, bandwidth requirements, or transmission channel capacity prohibit operation on the original data.

In general, data compression can be divided into two types of operations (Ref 10:4). The first operation, called entropy reduction (ER), is an irreversible transformation which reduces or compresses the data by mapping a source into an approximation of itself with a lower entropy rate. Sampling is an example of such a transformation. The second operation, known as redundancy reduction (RR) compresses the sampled data train by reducing, or eliminating, the redundancy existing in digital sequence. Since the redundant components of the data train contain no information about the source, RR is an ''exact'' data compression operation.

This chapter discusses data compression as it has been applied to the electrocardiogram (EKG). Two types of data compression strategies prevail. The first strategy involves two (or more) ER operations on the EKG data and attempts to compress the data by filtering selected components of discrete transforms. Because ER operations are irreversible, this type of data compression is sometimes

referred to as ''inexact'' compression. The second stategy performs data compression by reducing the redundancy present in the sampled EKG data train. This redundancy arises from two causes : 1) neighboring signal samples are not statistically independent and ; 2) the quantized signals amplitudes do not occur with equal probability.

Because of the speed limitations of the Motorola Exorciser microcomputer used in this thesis, only redundancy reduction algorithms were tested. This chapter, therefore, is intended as a review of the work that has been done in EKG data compression to allow comparison with the results obtained by this author.

Entropy Reduction

As is known from information theory (Ref 19), entropy {H(X)} is defined as a measure of the ''randomness'' or ''uncertainty'' of an information source. If the symbols emitted by the source are statistically independent, then the source is said to be ''memoryless'' and entropy is given by the equation:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i \qquad \text{(bits)} \qquad (1)$$

where p_i represents the probability of occurrence of the ith symbol and N is the number of distinct symbols output by the source. For those ''symbols'' in a data set which occur

with zero probability, the term $p_i \log_2 p_i$ is defined equal to zero.

Immediately obvious from Eq.(1) is the observation that H(X) is defined on a discrete probability distribution. In fact, Shannon (Ref 30) defined the entropy of a continuous source as equal to positive infinity. Thus an ER transformation must, in some manner, ''discretize'' a continuous waveform into a countable set of components and attach some probability to the elements of the set.

Sampling. The clearest example of this ''discretization'' is sampling, obviously the most important operation necessary for digital signal processing. In sampling, an electrical circuit (such as an Analog-to-Digital (A/D) converter) periodically measures the value of a signal x(t) and records the data as a numeric (usually binary) number. If the signal is sampled at least twice the highest frequency component of x(t), and the duration of the sampling operation is long enough that ''aliasing'' effects (Ref 36:68-72) of ''windowing'' are negligible, then all of the ''frequency domain'' components of interest in x(t) will be preserved. Physical constraints, however, limit the accuracy of the amplitude measurement to some finite precision. Amplitude information residing below the sensitivity of the A/D converter is irretrievably lost.

As an example, let the A/D converter digitize to 8 bits. With 8 levels there are 256 possible outputs, each with a certain probability of occurrence. If the continuous

waveform fed to the A/D is the output of a stationary stochastic process with uniform statistics, then each numeric value will occur with probability of 1/256 (assuming the source max/min deviation $\geq$ the A/D dynamic range). Assuming sample-to-sample independence, the source entropy is then calculated as

$$H(X) = \sum_{256} (1/256) \log_2 256 = 8. \qquad (2)$$

Entropy has been ''reduced'' from + ∞ to 8. If the stochastic source has ''less random'' statistics (like gaussian), entropy would be reduced even futher.

All of the EKG compression routines discussed in this thesis are implemented on digital computers, hence the ER operation of sampling is always performed. In ''inexact'' EKG data compression, the next operation is another ER mapping in which the sample sequence is transformed into an alternate domain and filtered.

Orthogonal Expansions and Filtering. A bandlimited waveform x(t) can be expanded (over a given interval T) as a linear combination of orthonormal basis functions $\phi_n(t)$ (Ref 36:20-21). That is

$$x(t) = \sum_{n=0}^{\infty} a_n \phi_n \qquad (3)$$

The functions $\phi_n(t)$, as $n=0,1,\ldots,\infty$, are defined as orthonormal if

$$\int_T \phi_i(t)\phi_k(t)\,dt = 1 \quad , \quad i=k$$
$$= 0 \qquad \text{otherwise} \qquad (4)$$

The orthonormal set $\{\phi_n(t)\}$ is called <u>complete</u> (C.O.N.) if, for any given $\varepsilon > 0$, there exists an N and a <u>finite</u> expansion

$$\tilde{x}(t) = \sum_{n=0}^{N-1} a_n \phi_n(t) \qquad (5)$$

such that

$$\int_T |x(t)-\tilde{x}(t)|^2\,dt < \varepsilon \qquad (6)$$

Given a complete, orthonormal set $\{\phi_n(t)\}$, then representation of any physical, noise-limited signal (over a prescribed interval T) is possible with a <u>finite</u> set of weighting coefficients $\{a_0, a_1, \ldots a_{N-1}\}$.

If k (k<N) terms in Eq.(5) are selectively suppressed (i.e., $a_0, a_1, \ldots a_k = 0$), then the signal x(t) is said to be filtered of those components and a compressed representation of x(t) is produced.

EKG Filtering Compression. A great deal of work (1),(2),(26),(33),(35),(21) has been done in attempting to ''compress'' EKG data by storing, or transmitting, selective components of an orthonormal expansion. This type of compression strategy is complicated by the complex characteristics of the EKG waveform. Foremost are the limitations imposed due to the ''nonstationarity'' or variability of the EKG source, both within any given waveform and the general population as a whole.

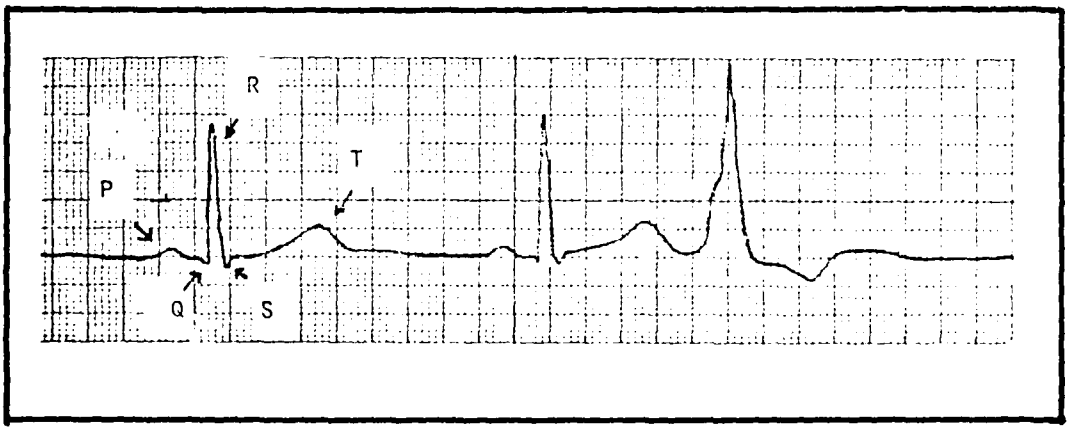


Fig 1. Typical EKG waveform with arrythmia.

During an EKG collection, the heart rate of the person undergoing testing can vary. This variation, or arrythmia, requires that a suitable reference point be established on

the P-QRS-T complex (Fig 1) around which the orthogonal expansion can be performed. Without this reference, the data falling within an expansion interval T (based purely on a constant Δt) will appear to be ''almost'' random. If the ''filtering'' used to effect compression is based on an expected waveform within the transform window, improper registration will cause severe degradation in compression efficiency. The above situation is known as the ''epoch'' problem and is discussed by Womble (Ref 35:703).

Assuming that the expansion interval is aligned properly on some feature of the EKG (e.g. R wave), then the problem of variable P,QRS, and T complexes must be considered.

If the individual under test has a heart disease, then in many cases the P,QRS,and T segments of that person's EKG may vary significantly from the population ''norm''. Again, if the compression filtering is based on an assumed, or ''normal'' waveform, then significant reduction in compression efficiency can be expected.

These problems, among others, require that the EKG ''filtering'' compressor be robust enough to handle the variations possible within the waveform. With these limitations in mind, discussion will now proceed to compression transforms.

Transforms. Two major types of discrete filtering compression strategies, as applied to EKGs, will now be discussed. These are the Fast Fourier Transform (FFT) and

the discrete Karhunen-Loeve Transform (DKLT). These two techniques are representative of current research into EEG filtering compression and will serve as good basepoints against which to compare the redundancy reduction compression techniques outlined in chapter 3.

Fast Fourier Transform. The fast Fourier transform is a computationally efficient algorithm for calculating the discrete Fourier transform (DFT). The following discussion involves the DFT, but calculation by the FFT is implied.

Data compression using the DFT is achieved by zonal filtering in which ''zones'' of the transform sequence are selectively discarded. Usually these zones are defined by a ''cutoff frequency'' f_c and only those components less than or equal to f_c are saved. Data reconstruction is performed by computing the inverse DFT (FFT) on the filtered sequence and replacing the filtered components with zeroes. Using the symmetry relationships described in Oppenheim and Shaffer (Ref 24:103-105), then the transform sequence for N=8 is:

$$X(k) = \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} R(0) + jIm(0) \\ R(1) + jIm(1) \\ R(2) + jIm(2) \\ R(3) + jIm(3) \\ R(4) + jIm(4) \\ R(3) - jIm(3) \\ R(2) - jIm(2) \\ R(1) - jIm(1) \end{bmatrix} \quad (7)$$

where R is the real part, Im the imaginary part of the complex number X, and $j=\sqrt{-1}$.

Zonal filtering takes advantage of the symmetry in Eq.

(7) by saving those positive frequency components less than or equal to a certain cutoff frequency f_c. IF f_c were chosen as equal to X(2), then only X(0),X(1), and X(2) would be saved. On reconstruction, X(3),X(4),and X(5) would be set equal to zero (i.e. they have been filtered out) while X(6) and X(7) would be recreated from X(1) and X(2) using symmetry. The inverse DFT (FFT) now produces a ''filtered'' approximation of x(n).

EKG data compression via filtered FFT spectra has been studied by Womble (Ref 35) and the TRW corporation (Ref 33). In both studies the distortion criterion used in establishing f_c was visible reproducibility with no detectable distortion.

In the TRW study, compression ratios (defined as bits in : bits out) from 2:1 to as high as 17:1 were obtained using zonal filtering. The high compression ratios were measured using highly rhythmic EKGs taken from an individual (an astronaut) in a low noise environment. This extraordinary compression ratio is far from ''normal'', however. Womble (Ref 35) has shown that, on the average, 40-80 terms of a 512 sample FFT (sampled at 500 Hz) are necessary to reproduce the EKG with acceptable visual distortion. Since the FFT requires storage of 2 numbers per term (real and imaginary components) then, in general, FFT zonal filtering compression ratios of 5:1 to 3:1 are more common.

The FFT is often used in digital signal processing

because of its speed and representation in the frequency domain. As will be seen from the data in chapter 5, compression ratios of 5 : 1 are usually higher than those obtainable by redundancy reduction techniques. Even so, the FFT is not the ''optimal'' transform for representing a sample sequence.

Discrete Karhunen-Loeve Transform. Because all of the filtering compression techniques are ER transformations, distortion upon reconstruction is inevitable. One would like a performance measure against which this distortion could be evaluated.

One common measure of performance is the mean square error defined by the equation:

$$e(M)=E\{(\underline{X}-\tilde{\underline{X}})^2\} \tag{8}$$

where E is the statistical expectation operator, $\underline{X}$ is a discrete signal composed of N sample values, and $\tilde{\underline{X}}$ is the estimate of $\underline{X}$ via some orthogonal coordinate system. Given that a representation of $\underline{X}$ is desired with less than N components, Ahmed and Rao (Ref 2:200-203) have shown that the DKLT is the optimum transform to minimize mean square error.

The DKLT can be described as follows: Let the orthonormal transform matrix [T] be defined as

$$[T]' = [\,\phi_1,\phi_2,\ldots\phi_N] \quad \{[T]'=[T] \text{ transpose}\} \tag{9}$$

and ϕ_i are N dimensional, real valued basis vectors. Hence a transform vector $\underline{Y}$ can be created by

$$\underline{Y} = [T]\,\underline{X} \tag{10}$$

where $\underline{X} = [x_1, x_2, \ldots x_N]$; $\underline{Y} = [y_1, y_2, \ldots y_N]$. Since the ϕ_i are orthonormal, then

$$\underline{X} = y_1\phi_1 + y_2\phi_2 + \ldots + y_N\phi_N = \sum_{i=1}^{N} y_n\phi_n \tag{11}$$

The goal is to optimally represent $\underline{X}$ by a subset $\{y_1, y_2, \ldots, y_M\}$ where $M < N$. If the remaining N-M terms are represented by the constants b_i, then an estimate $\tilde{\underline{X}}$ is defined such that

$$\tilde{\underline{X}} = \sum_{i=1}^{M} y_i\phi_i + \sum_{i=M+1}^{N} b_i\phi_i \tag{12}$$

An error vector, $\Delta\underline{X}$ is now created where

$$\Delta\underline{X} = (\underline{X} - \tilde{\underline{X}}) = \sum_{i=M+1}^{N} (y_i - b_i)\phi_i \tag{13}$$

Now the mean square error defined in Eq.(8) can be redefined

as

$$e(M) = E\{(\Delta\underline{X})'(\Delta\underline{X})\}$$

$$= E\{\sum_{i=M+1}^{N}\sum_{j=M+1}^{N}(y_i - b_i)(y_j - b_j)(\emptyset'_i\emptyset_j)\}$$

$$= \sum_{i=M+1}^{N} E\{(y_i - b_i)^2\} \tag{14}$$

Ahmed and Rao (Ref 2:202) show that by minimizing Eq.(14)

$$e(M) = \sum_{i=M+1}^{N}[\emptyset'_i E\{(\underline{X} - \bar{\underline{X}})(\underline{X} - \bar{\underline{X}})\}'\emptyset_i] \tag{15}$$

where $\bar{\underline{X}}$ is the mean of $\underline{X}$.

The expectation in Eq. (15) is recognized as the covariance of $\underline{X}$, hence Eq. (15) becomes

$$e(M) = \sum_{i=M+1}^{N}(\emptyset'_i[K_x]\emptyset_i) \tag{16}$$

where $[K_x]$ is the covariance matrix of $\underline{X}$.

By minimizing Eq.(16), Ahmed and Rao (Ref 2:200-205)

show that the othornormal functions ϕ_i become the eigenfunctions of the covariance matrix $[K_x]$ and the mean square error becomes

$$e(M) = \sum_{i=M+1}^{N} \lambda_i \qquad (17)$$

where the λ_i are the eigenvalues of $[K_x]$. By expanding $\underline{X}$ with the eigenfunctions corresponding to the M largest eigenvalues, then $\underline{X}$ is filtered of N-M components with minimum mean square error. This is the discrete Karhunen-Loeve transform.

The DKLT has been studied by Ahmed (Ref 1) and Womble (Ref 35). Both studies compared the performance of EKG data compression using the DKLT against other orthogonal transforms.

Ahmed's tests utilized canine EKGs and clearly showed the optimality (in the ''mean square sense'') of the DKLT. Ahmed, however, deemed the DKLT to be too complex for practical implementation, and proceeded with the development of EKG compression strategies using suboptimal transforms. The details of these suboptimal expansions will not be given here, but the reader is referrenced to Ahmed's paper (Ref 1).

Womble, on the otherhand, demonstrated that the DKLT is not to difficult to implement. In his experiments, Womble

took 3 lead vectorcardiogram (VCG) data (see appendix A), hereafter referred to as X,Y,Z, and first transformed the data into a different, orthogonal coordinate system (U,V,W). The transformation was determined by solving for the eigenvalues of the 3 X 3 matrix:

$$S = 1/N \sum_{i=1}^{N} \begin{bmatrix} X(i) \\ Y(i) \\ Z(i) \end{bmatrix} \begin{bmatrix} X(i), Y(i), Z(i) \end{bmatrix} \tag{18}$$

where N was chosen equal to 200 and X(i),Y(i),Z(i) are the X,Y,Z components of the VCG in the Frank (Ref 14) coordinate system.

Next an ''average'', or mean, heartbeat was calculated using 900 patients and the data (in the eigenvector coordinate system) was subtracted from the mean forming a ''patient'' vector ρ. The vector ρ is defined as

$$\rho = \begin{bmatrix} u(1) - u_m(1) \\ u(2) - u_m(2) \\ \vdots \\ u(N) - u_m(N) \\ v(1) - v_m(1) \\ v(2) - v_m(2) \\ \vdots \\ v(N) - v_m(N) \\ z(1) - z_m(1) \\ z(2) - z_m(2) \\ \vdots \\ z(N) - z_m(N) \end{bmatrix} \tag{19}$$

This patient vector ρ is expanded using the eigenfunctions

of the matrix

$$L = 1/N \sum_{i=1}^{N} (\rho_i)(\rho_i)' \qquad (20)$$

where N is now a large number of patients being ''averaged'' (up to 300 reported in Ref 33). The eigenvectors of this tremendous matrix are calculated (once for all time), and then M (M << 200) are used to expand the data vector ρ.

The above approach has allowed accurate representation of EKG sample sequences, properly aligned on the heartbeat, with an M=20 DKLT coefficients per lead per second (Ref 35). At Womble's sample rate of 250 Hz, this represents a compression ratio of over 12 : 1 . With the most significant eigenfunctions stored from the solution of Eq. (20), then the compression of digitized EKG data by filtering the DKLT expansion on ρ is approaching feasibility on a microcomputer.

The preceeding discussion was intended as a basic tutorial on some of the entropy reducing (ER) transformations currently under test for EKG data compression. As stated earlier, ER operations are ''inexact'' because some ''information'' is always discarded. By discarding data efficiently, zonal filtering can achieve compression ratios larger than those obtained using the redundancy reduction techniques implemented in this thesis.

Redundancy Reduction

Redundancy reduction is an operation which performs data compression by indentifying and removing the redundant components of a digital sequence. This redundancy exists for two reasons : 1) the sample points are not statistically independent and ; 2) the quantized amplitude values of the sample train do not occur with equal probability. The redundant components of the data sequence carry no ''information'' (see appendix B), hence their removal does not affect the ''message'' content of the data.

This section of chapter two discusses redundancy reduction (RR) operations which have been applied to the EKG. This is done because the compression algorithms examined in this thesis are of the RR type, and the background given here will aid in the descriptions given in chapter three.

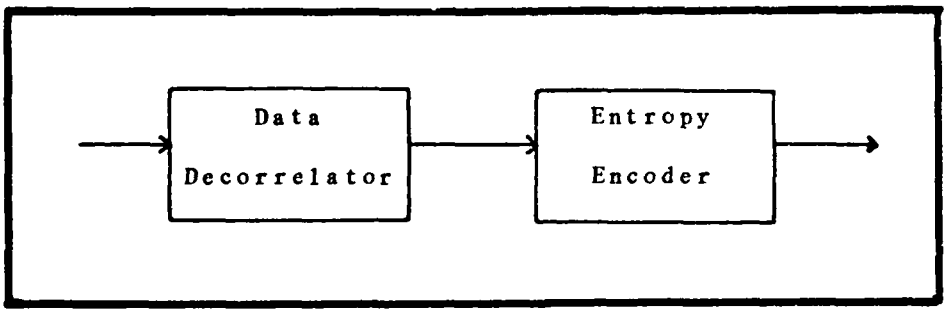


Fig 2. Typical EKG Redundancy Reduction Data Compressor.

The sampled EKG contains redundancy resulting from both

sample-to-sample correlation and unequal amplitude probability. As is illustrated by Figure 2, each of these redundancy components can be reduced by a different operator.

The decorrelator, as inferred by its title, transforms the data in a manner which decorrelates the data stream. The decorrelated residual sequence is then efficiently compressed by means of the entropy encoder.

To reduce the rendundancy arising from intersample correlation, two basic approaches are used. The first approach utilizes linear predictor/interpolators (Ref 28,33) to produce an information bearing ''residual '' sequence. The second method generates the residual sequence by taking successive differences on the data stream. As will be shown, these residual sequences carry all of the source information with minimum intersample redundancy. The predictor/interpolator method is discussed first.

Predictors and Interpolators. A predictor estimates the next sample value of a sequence (i.e. x_n) based on a linear combination of k past samples. That is

$$\tilde{x}_n = \sum_{i=1}^{k} a_i x_{n-i} \tag{21}$$

where a_i are coefficients chosen to minimize the mean square error $\sigma_e^2 = E\{(X-\tilde{X})^2\}$ between the sample sequence X_n and the

predicted sequence $\tilde{X}_n$. An error , or residual, sequence e_n can then be formed where

$$e_n = X_n - \tilde{X}_n \tag{22}$$

thus

$$X_n = \tilde{X}_n + e_n \tag{23}$$

Conceptually, this technique decomposes a sample value into a part which is correlated with k past sample values (i.e. the redundant component) and a part which is uncorrelated with them. Linear mean square estimation theory (Ref 24:385-430) shows that the uncorrelated part (e_n) may be retained alone with no loss of information.

In their research in EKG data compression, Ruttimann and Pipberger (Ref 28:616) prove that since $E\{e_n\}=0$, then the mean square error σ_e^2 is equal to the variance of e_n . With a second order predictor (k=2), Ruttimann and Pipberger have demonstrated a variance reduction (σ_x^2/σ_e^2) of over 25 : 1. This reduction in variance implies that the EKG residual sequence e_n is much more tightly clustered around a given mean than is the original sequence X_n . This clustering, as will be discussed later, enhances data compression by means of entropy ,or source encoding.

In an analogous way, interpolators estimate a value of x_n . In the case of the interpolator, however, the estimate

of x_n consists of a linear combination of past and future samples. That is

$$\tilde{x}_n = \sum_{i=1}^{k} a_i x_{n-i} + \sum_{i=1}^{m} b_i x_{n+i} \qquad (24)$$

where k past and m future values are used. As with the predictor, the coefficients a_i and b_i are again chosen to minimize the mean square error. As was the case with the second order predictor, Ruttiman and Pipberger (Ref 28:617) have found that a second order interpolator (k=1,m=1) yields EKG residual sequences e_n with the most significant variance reductions (greater than 31 : 1).

EKG data compression utilizing predictors/interpolators have been studied by the TRW Corporation (Ref 33), as well as Ruttimann and Pipberger (Ref 28). In both cases, predictors/interpolators of order 2 seem to prevail. As is shown by the two groups above, second order systems have the smallest residual sequence variance hence are most ameanable to entropy encoding. An alternate approach used to reduce intersample redundancy is by means of difference operations.

Difference Reduction. In difference reduction, the residual sequence is formed by taking successive differences of the sample data train. Because there is no multiplication (as in Eq.(24)), the difference operation is inherently a

simpler procedure than is prediction or interpolation. Difference sequences, however, are not optimized with respect to mean square error and less ''efficient'' redundancy reduction (decorrelation) occurs.

An example of an EKG difference reduction compressor is illustrated in Figure 3. This technique was implemented by Cox and Ripley (Ref 7) and tested against a data base of 45 patients. Figure 4 shows that Cox and Ripley's second difference decorrelator produced a sharply peaked relative frequency distribution. This ''peakedness'' or clustering of the residual sequence permits efficeicnt entropy encoding. A variant of the difference reduction above is time compression.

Time Compression. EKG time compression, as implemented by Dower,Berghofer, and Stewart (Ref 12,29), uses a second order difference reduction but instead of keeping the value of zero (the most common), a run length counter (Δt) is kept. This run length counter measures the number of repetitive $\Delta^2 X$ sequence terms equal to zero. By saving (then encoding) only those $\Delta^2 X \neq 0$, and the Δt between the nonzero second differences, data compression can be realized.

Time compression is the method that has been implemented and studied extensively by this author. Chapter 3 discusses two different RR compression algorithms which both use second order time compression for the decorrelator. For now, dicussion is focused on the second

operation identified in Figure 2, the entropy encoder.

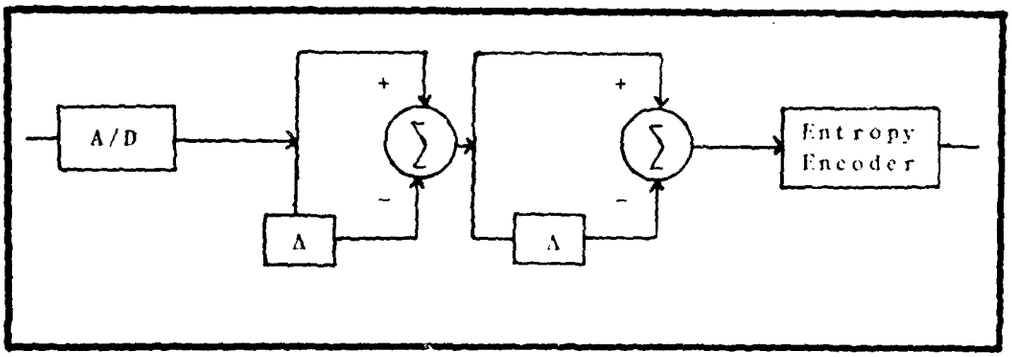


Fig 3. Second Order Difference Operator (from Ref 7:336).

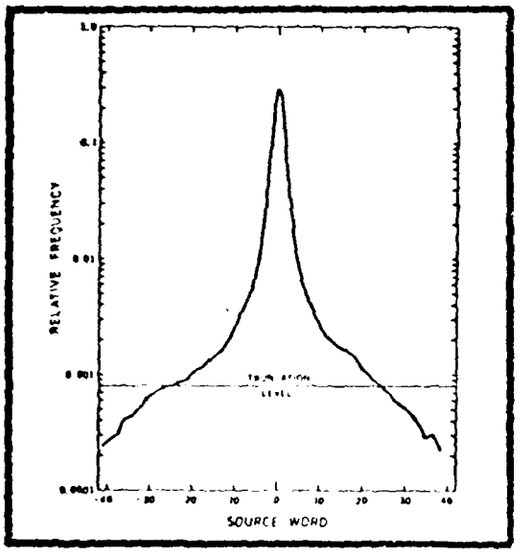


Fig 4. Relative Frequency of 2nd Order Difference Operator (Ref 7:336)

Entropy Encoding. Entropy, or source, encoding operates on the remaining redundancy in the residual sequence e_n resulting from unequal source symbol probability. How well e_n can be compressed is given by Shannon's noisless source coding theorem (Ref 30) which states that the average number of binary symbols per source output can be made to approach the entropy of the source and no less. Unfortunately, the EKG ''source'' has memory, hence the entropy of this source will be less than that which would be calculated from Eq.(1). In general, the true value of a nonstationary, memory source is difficult (or impossible) to calculate.

To circumvent this ''memory'' problem, the assumption is made in the literature (Ref 29,12, 7,35) that the ''uncorrelated'' nature of the residual sequence e_n approaches independence, hence the entropy of e_n as calculated by Eq.(1) is a good bound on the possible compression. Since the variance of e_n is sharply peaked around zero, entropy encoding via variable length coding (VLC) appears attractive.

In variable length coding, those values of e_n which occur most often are assigned the shortest code words (i.e. fewest code symbols per source symbol). As example, assume that both the source symbol alphabet and the code alphabet are binary. The values in the sequence e_n are the result of algebraic operations on fixed length numeric sample values and are hence fixed length binary numbers (e.g. L=8). The variable length coder maps these fixed length numeric values

(binary) into uniquely decodable (UD) binary code words whose bit length is a function of the frequency of occurrence of the values in e_n .

Coding theory (Ref 19:237-248) shows that, in principle, it is possible to have variable length codes with an average bit length ($\underline{l}$) equal to the entropy of source. Hence with the right VLC, a compression ratio of $L/\underline{l}$ is achievable.

A common VLC used in EKG data compression research (Ref 7,28) is the Huffman code. Huffman, in 1952, developed an algorithm (Ref 16) for generating the optimal UD code (assuming stationary, memoryless source). This code is in a class of UD codes called prefix codes in which no code word is the prefix of any other. Unfortunately, a codeword must be assigned to every possible symbol which occurs, regardless of how infrequently. This means that although the average code word bit length approaches the entropy of the source, the longest code word can be substantially larger. In a straight Huffman code, these ''long'' code words can induce severe problems due to ''buffer overflow'' (Ref 17).

For practical implementation in EKG data compression, Ruttimann and Pipberger (Ref 28) and Cox and Ripley (Ref 7) constructed a modified Huffman code. In this modified code, the residual sequence source words are partitioned into a frequent set and an infrequent set (known as ''else''). A Huffman code was then formed with all of the residual words

in the first set plus a special code word used as a prefix for any source word from the infrequent set. The prefix, when it occurs, is followed by a fixed length suffix which contains the value of the infrequent source word. The probability that ''else'' will occur must be kept small enough to mantain the efficiency of the truncated Huffman code.

Using the modified Huffman code in conjunction with a second order interpolator, Ruttimann and Pipberger (Ref 28) have attained compression ratios as high as 9 : 1. To get this ratio, however, significant digital signal processing was performed on the ''raw'' 8 bit data. This preprocessing involved digital filtering for noise reduction and Lagrange interpolation/decimation to produce an effective 200 Hz sample rate. Nonetheless, this represents a significant approach to the compression ratios attainable by the discrete Karhunen-Loeve transform mentioned previously.

Although optimal, Huffman codes are not the only VLC used in EKG data compression. Two different codes will be described in chapter three, one of which (Dower code) may possibly be ''more optimal'' than Huffman in the compression of quantized EKGs.

Summary

This chapter has reviewed the theory of data compression and how this theory has been applied to the electrocardiogram. It was shown that data compression could

be partitioned into two types of operations: 1) entropy reducing transformations which map a data source into an approximation of itself with a lower entropy rate and ; 2) redundancy reduction operations which compress by removing redundancy resulting from sample-to-sample dependence and unequal source symbol probabilities.

The first operation was applied to the EKG via ''filtering'' of orthogonal expansions of the digitized EKG. Filtering transforms, especially the discrete Karhunen-Loeve transform, were shown to be very efficient ''compressors'' if the loss of the filtered components were tolerable. If this loss was not acceptable, then redundancy reduction operations are used.

In redundancy reduction techniques, the EKG is first processed by passing the digitized data through a decorrelator which reduced the redundant component caused by source symbol dependence. This decorrelator could be implemented with predictor/interpolators or difference operations. Next the data is encoded via entropy encoding operations, usually utilizing variable length codes. It was finally shown that a second order interpolator, followed by an ''optimal'' Huffman encoder could achieve compression ratios which approach those obtained by the discret Karhunen-Loeve transform. Both of these last two techniques, however, require substantial computational overhead.

III. Tolan and Dower EKG Compression Techniques

Two EKG data compression algorithms are studied in detail in this chapter. The first compression technique, hereafter referred to as the Tolan method, was conceived by Dr. Gil Tolan,MC from the USAF School of Aerospace Medicine (USAFSAM), Brooks AFB, Texas. The second compression approach, referred to as the Dower method, was developed by Mr. Roger Dower and Mr. Dave Berghofer from Shaughnessy Hospital, Vancouver, B.C., Canada. Both of the EKG compression procedures are redundancy reduction operations using time compression for decorrelation and variable length codes for entropy encoding.

This author had originally intended to implement and test both the Tolan algorithm and the Dower algorithm on the Motorola microcomputer (See Chapter 4). Unfortunately, difficulties with hardware failures and insufficient software tools (i.e., a high order language) prevented implementation of the Dower algorithm. Nonetheless, this chapter compares the design of both the Tolan and the Dower algorithms and illustrates their differences as well as their similarities. Based on the results of the Tolan compression approach (chapter 5), this comparison will illustrate the potential performance of the Dower compression stategy in a real time microcomputer enviroment. Before continuing with the Dower and Tolan algorithm descriptions, a short digression will be made.

In both the Tolan and The Dower compressors, a second

difference operation is used to produce an information bearing sequence with reduced component-to-component dependence. Why the second difference operation results in the lowest correlation is a question for closer scrunity.

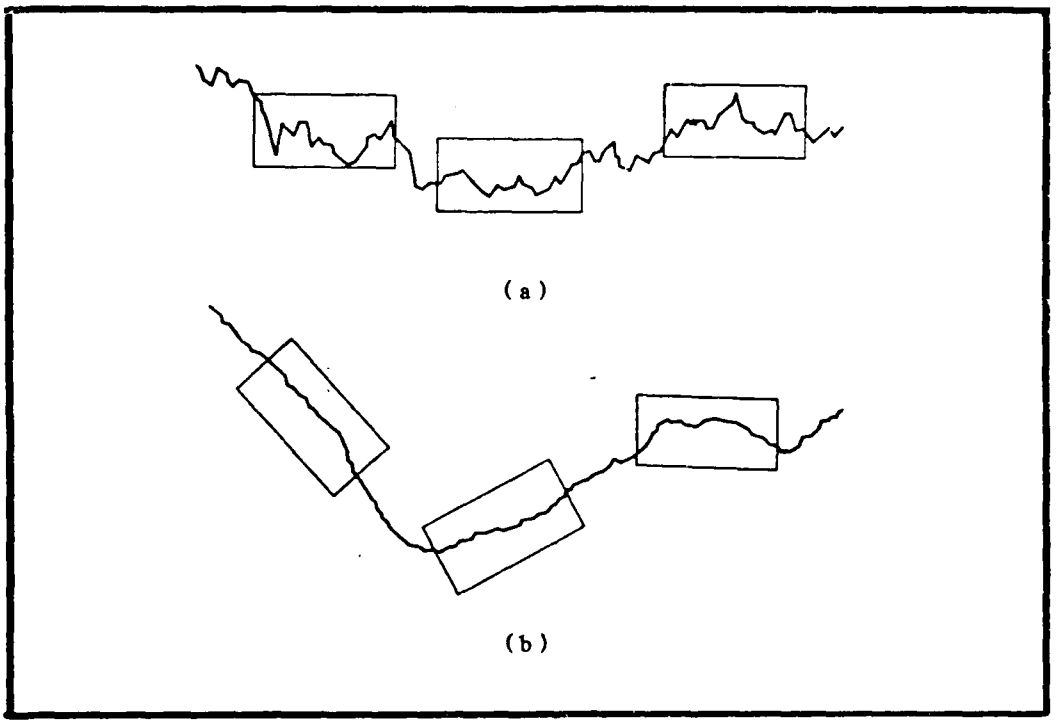


Fig 5. Nonstationary waveforms with random level and slope. (From ref 6:91)

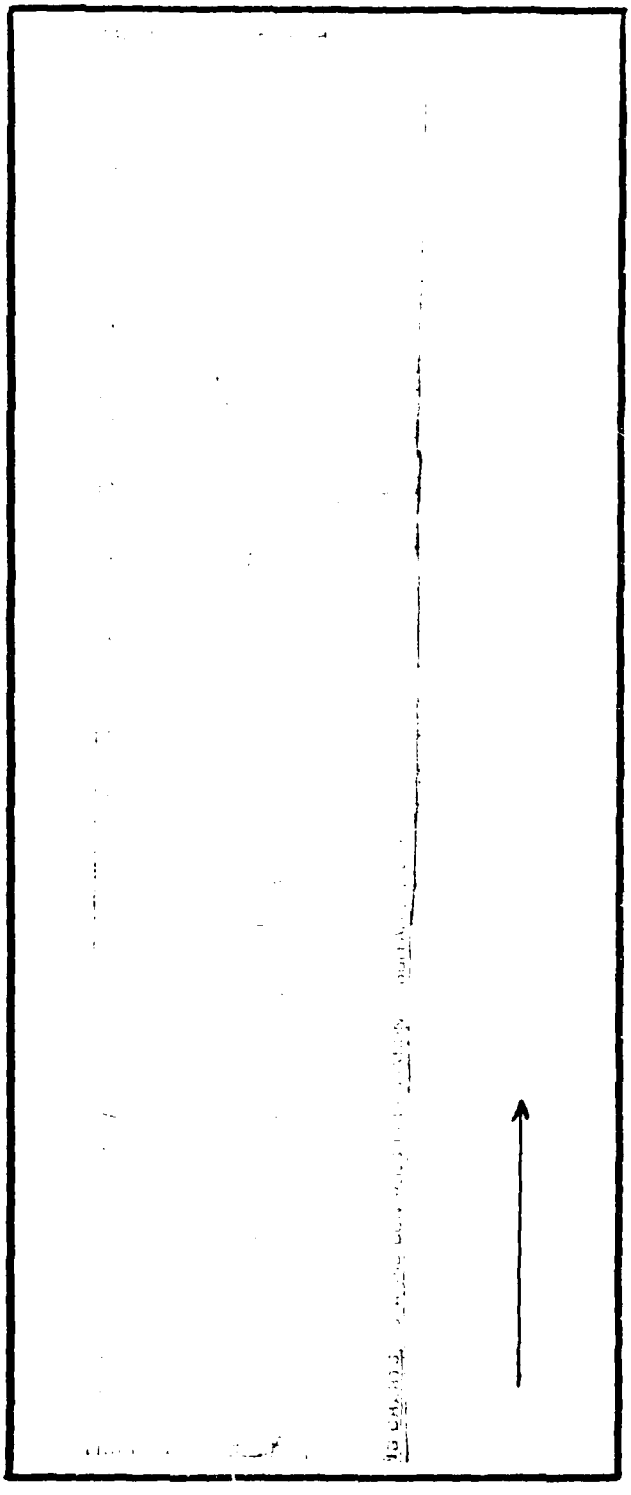


Fig 6. EKG waveform with moving average slope.

A nonstationary process (like the EKG) exhibits a wide amplitude distribution due to the random ''wander'' of the waveform as shown in Figure 6. This wander, or ''moving average'', can be induced by patient variation, EKG apparatus drift, or both. Time series analysis (Ref 6) shows that a nonstationary process which has sample functions which are ''locally stationary'' or homogenous (Figure 5) can be represented by a process model which calls for the d'th difference of the process to be stationary. The proof of this assertion is given by Box and Jenkins (Ref 6:85-125).

If the nonstationary process sample sequence (time series) exhibits a random level as illustrated in Figure 5a, then a first difference operation will remove this ''moving average'' and force the resulting difference sequence to be centered around zero. If the waveform exhibits a random slope, as shown in Figure 5b, then a second difference operation will remove this quadratic ''bias'' with a corresponding reduction in amplitude distribution variance (Figure 7.) Comparison of Figures 5 and 6 shows that an EKG trace can ''look'' similar to the example in Figure 5b.

From the theory of stochastic processes (Ref 9:330-331), it is known that the expected value of a sample mean obtained by sampling a wide-sense stationary random process along a sample function in time is equal to the constant mean value of that random process.

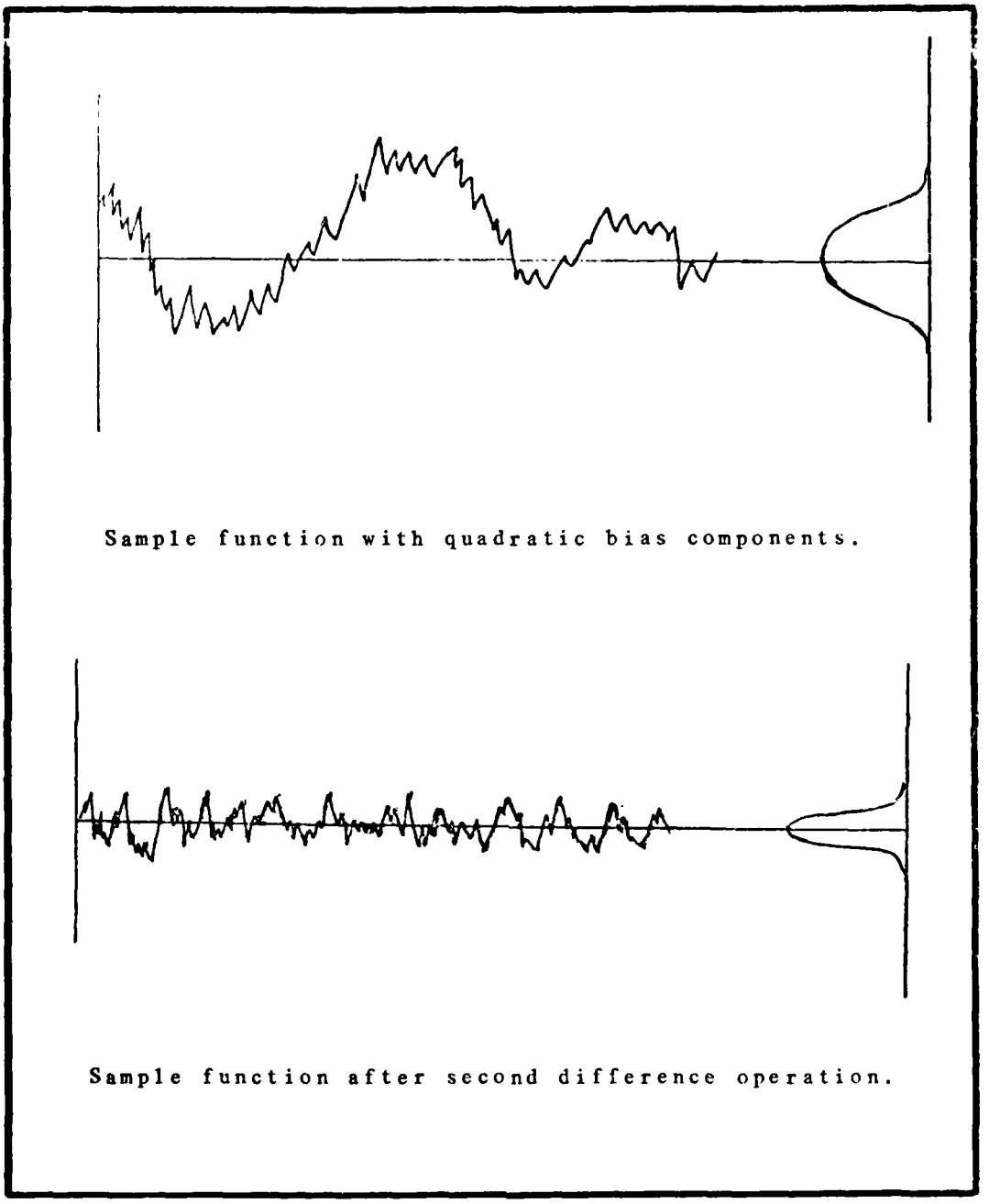


Fig 7. Simulated variance reduction via second differencing.

In addition, the variance of the sample mean is inversely proportional to the number of samples taken when the samples are pairwise uncorrelated. Another fact about widesense stationary processes is that their autocorrelation functions are dependent only on the time difference between observation of the sample sequence and that $R(t_1 - t_2) \leq R(0)$ for $t_1 - t_2 \neq 0$. Since the second difference is the first ''difference'' which can be modelled as coming from a ''stationary'' process, and the maximimum correlation occurs for zero time difference, then the correlation must be reduced for adjacent sequence values in the second difference operation.

This hueristic argument is far from complete and does not explain why the third difference exhibits worse behavior than the second difference. The EKG waveform is a complex function from a complex source and higher order effects could begin to dominate with the third difference operation. Further analysis of this anomaly is left for future study.

The remainder of this chapter is organized in the following manner. First the Tolan algorithm is described, followed by a description of the Dower procedure. Next a short synopsis will be made of three other EKG data compression algorithms uncovered during the research of this thesis.

Tolan EKG Data Compressor

The Tolan EKG data compression algorithm is a redundancy reduction procedure which processes a three lead EKG (VCG) and produces a compressed, digital output. This digital output sequence could subsequently be channel encoded for ''errorless'' transmisson (Ref 27 and Appendix B) or stored for later retrieval and reconstruction.

As was the case for the RR techniques described in chapter 2, the Tolan compressor is subdivided into a data decorrelator and an entropy encoder. The decorrelator is discussed first.

Tolan Decorrelator. The Tolan decorrelator is a second order difference reduction operation which utilizes a time compression approach to form a decorrelated output sequence. A second order system was chosen based on the experimental results of Dower and Berghofer (Ref 12) and Cox and Ripley (Ref 7) which showed maximum compression gain with a second order difference operation. The Tolan decorrelator works on a three lead EKG (VCG) signal set, assumed to be sampled at a constant rate. The data compression is achieved in real time between successive samples.

The Tolan decorrelator algorithm is defined in Figure 8. Close examination of the algorithm in Figure 8 reveals that the second difference data is stored only if : 1) any of the three Δ^2 values are nonzero or ; 2) if the Δt counter

records more than 127 repetitive cases where $\Delta^2x=\Delta^2y=\Delta^2z=0$. Rule 1 was chosen due to the high degree of correlation between the leads of an EKG (i.e. when one lead is changing, so are the others). The capacity of the time counter in rule 2 was arbitrarily chosen to be sufficient to record the long quiescent periods which occur in the EKG (see Figure 1) and short enough to be efficiently stored.

Step 12 of the Tolan algorithm in Figure 8 calls the variable length encoding subroutine (Figure 10) which encodes the Δ^2 terms calculated by the second difference decorrelator. This encoding procedure is the next subject to be discussed.

Tolan Entropy Encoder. The Tolan code is an uniquely decodable variable length code which stores the Δ^2 values as a contiguous sequence of binary 1's. The length of this ''run'' of binary 1's is equal to the magnitude of the Δ^2 term. To delineate between the ''runs'', binary 0's are used as codeword delimiters. Since the second difference has both negative and positive values, a sign bit (0 for positive, 1 for negative) immediately follows the 0 bit delimiter. The three values $\Delta^2x,\Delta^2y,\Delta^2z$ are encoded and stored sequentially followed by a delimited, 7 bit, uncoded Δt value. A Δ^2 value of zero is indicated by 3 successive 0 bits.

```
 1: Δx=0, Δy=0, Δz=0, Δt=1
 2: x(i)=(a/d ch 0),y(i)=(a/d ch 1),z(i)=(a/d ch 2)
 3: mem(0)=x(i),mem(1)=y(i),mem(2)=z(i)
 4: x̃(i+1)=x(i)+Δx, ỹ(i+1)=y(i)+Δy, z̃(i+1)=z(i)+Δz
 5: if ready for next sample then GOTO 6 else GOTO 5
 6: i=i+1,x(i)=(a/d ch 0),y(i)=(a/d ch 1),z(i)=(a/d ch 2)
 7: Δ²x=x(i)-x̃(i), Δ²y=y(i)-ỹ(i), Δ²z=z(i)-z̃(i)
 8: if Δ²x≠0 or Δ²y≠0 or Δ²z≠0  then GOTO 11 else GOTO 9
 9: Δt=Δt + 1
10: if Δt ≤ 127 then GOTO 4 else GOTO 12
11: Δx=Δx + Δ²x, Δy=Δy + Δ²y, Δz=Δz + Δ²z
12: go subroutine coder { Δ²x,Δ²y,Δ²z ,Δt}
13: if memory is full then STOP else GOTO 14
14: Δt=1
15: GOTO 4
where

x(i),y(i),z(i)= sampled, 8 bit precision, EKG data
    Δx,Δy,Δz = first difference {Δx(n)=x(n)-x(n-1)}
  Δ²x,Δ²y,Δ²z = second difference {Δ²x(n)=Δx(n)-Δx(n-1)}
           Δt = time difference between nonzero Δ² values
x̃(i+1),ỹ(i+1),z̃(i+1)= next predicted data points
```

Fig 8. Tolan Collection and Decorrelation Algorithm.

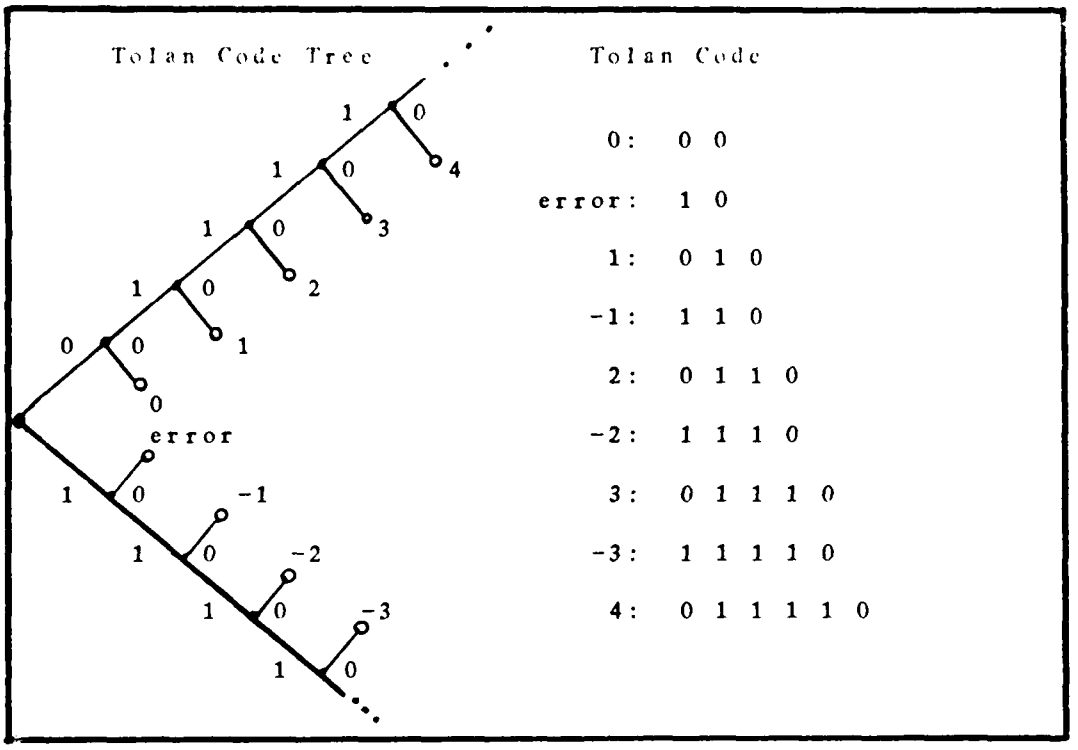


Fig 9. Tolan Code and Code Tree (8 smallest codewords).

As an example of Tolan encoding, let $\Delta^2 x=3$, $\Delta^2 y=-1$, $\Delta^2 z=0$, $\Delta t=13$. With these values, then the code string generated is:

D S V V V D S V D S D T T T T T T T D

0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 1 0 1 0

where

D = Delimiter bit

S = Sign bit

V = Value bit

T = Time bit

Examination of the above sequence shows that the Δ^2

codewords are prefix codes as shown in the tree diagram of Fig. 9. To decode the sequence correctly, however, it is imperative that codeword synchronization be maintained in order to determine when the coded Δ^2 values end and the uncoded, 7 bit Δt variable starts. This loss of synchronization is detected by the ''error'' state shown in the code tree of Figure 8. The decoder which implements the code tree in Figure 9 will not be discussed here but is listed in Appendix C.

In contrast with the Huffman code (see chapter 2 and Ref 16) and the Dower code to be discussed, the Tolan code is <u>not</u> constructed using the apriori knowledge of the source word (i.e. $\Delta^2x, \Delta^2y, \Delta^2z$) relative frequency of occurrence. This means that the Tolan code will only produce bit compression (bits out/bits in/$<$ 1) if the Δ^2 values are <u>sharply peaked</u> around a mean of zero such that few code words exceed the 8 bit, fixed length value of the Δ^2 terms. As will be shown in chapter 5, the second order difference decorrelator does produce such a sharply peaked relative distribution.

The algorithm which implements the Tolan variable length coder is shown in Figure 10. This algorithm works in conjunction with the Tolan decorrelator in Figure 8.

```
 1: entry (Δ²x,Δ²y,Δ²x,Δt)
 2: cnt ←- 0
 3: reset current memory bit, increment bit pointers
 4: if end of memory,set eom flag and RETURN else GOTO 5
 5: cnt ←- cnt + 1
 6: if cnt=1 then tvar ←- Δ²x
 7: if cnt=2 then tvar ←- Δ²y
 8: if cnt=3 then tvar ←- Δ²z
 9: if cnt ≥ 4 then GOTO 18 else GOTO 10
10: if tvar ≥ 0 then GOTO 11 else GOTO 12
11: reset memory bit, increment bit counters and GOTO 13
12: set memory bit, increment bit counters
13: if end of memory,set eom flag and RETURN else GOTO 14
14: if tvar=0 then GOTO 5 else GOTO 15
15: set memory bit, increment bit counters
16: if end of memory,set eom flag and RETURN else GOTO 17
17: tvar ←-  tvar - 1 and GOTO 14
17: store 7 bit Δt counter to memory, update bit pointers
18: if end of memory,set eom flag and RETURN else RETURN
```

Fig 10. Tolan Variable Length Encoder Algorithm.

```
 0: START OF DOWER DATA COLLECTION AND DECORRELATION
 1: FLAG=TDT=DT=DX1=DY1=DZ1=DX2=DY2=DZ2=DDX=DDY=DDZ=CNT=0
 2: If sample interrupt detected then GOTO 3 else GOTO 2
 3: X(n) ← A/D CH 0, Y(n) ← A/D CH 1, Y(n) ← A/D CH 2
 4: CNT=CNT + 1
 5: If CNT =1 then GOTO 6 else GOTO 7
 6: X(n-1) ← X(n), Y(n-1) ← Y(n), Z(n-1) ← Z(n) and GOTO 2
 7: If CNT =2 then GOTO 8 else GOTO 13
 8: DX1 ← X(n)-X(n-1),DY1 ← Y(n)-Y(n-1),DZ1 ← Z(n)-Z(n-1),DT ← DT+1
 9: If DX1≠0 OR DY1≠0 OR DZ1≠0 OR DT > 127 then GOTO 11 else GOTO 10
10: CNT ← 1  and GOTO 2
11: TDT ← DT-63, X(n-2) ← X(n-1), Y(n-2) ← Y(n-1), Z(n-2) ← Z(n-1)
12: X(n-1) ← X(n), Y(n-1) ← Y(n), Z(n-1) ← Z(n) DT ←0 and GOTO 2
13: DX2 ← X(n)-X(n-1),DY2 ← Y(n)-Y(n-1),DZ2 ← Z(n)-Z(n-1),DT ← DT+1
14: If DX2≠0 OR DY2≠0 OR DZ2≠0 OR DT > 127 then GOTO 16 else GOTO 15
15: CNT ← 2  and GOTO 2
16: DDX ← DX2-DX1, DDY ← DY2-DY1, DDZ ← DZ2-DZ1
17: If |DDX| > 63 then GOTO 18 else GOTO 21
18: DUMX(n)=.25X(n-1)+.75X(n)
    DUMX(n-1)=.25X(n-2)+.5X(n-1)+.25X(n)
    DUMX(n-2)=.75X(n-2)+.25X(n-1)
19: X(n) ← DUMX(n), X(n-1) ← DUMX(n-1), X(n-2) ← DUMX(n-2)
20: DX1=X(n-1)-X(n-2), DX2=X(n)-X(n-1)  and GOTO 16
```

Fig 11-a. Dower Collection and Decorrelation Algorithm.

```
21: If |DDY| > 63 then GOTO 22 else GOTO 25

22: DUMY(n)=.25Y(n-1)+.75Y(n)

    DUMY(n-1)=.25Y(n-2)+.5Y(n-1)+.25Y(n)

    DUMY(n-2)=.75Y(n-2)+.25Y(n-1)

23: Y(n) ← DUMY(n), Y(n-1) ← DUMY(n-1), Y(n-2) ← DUMY(n-2)

24: DY1=Y(n-1)-Y(n-2), DY2=Y(n)-Y(n-1)  and GOTO 16

25: If |DDZ| > 63 then GOTO 26 else GOTO 29

26: DUMZ(n)=.25Z(n-1)+.75Z(n)

    DUMZ(n-1)=.25Z(n-2)+.5Z(n-1)+.25Z(n)

    DUMZ(n-2)=.75Z(n-2)+.25Z(n-1)

27: Z(n) ← DUMZ(n), Z(n-1) ← DUMZ(n-1), Z(n-2) ← DUMZ(n-2)

28: DZ1=Z(n-1)-Z(n-2), DZ2=Z(n)-Z(n-1)  and GOTO 16

29: If FLAG=0 then GOTO 30 else GOTO 31

30: VLC ← (0:DX1,DY1,DZ1)

31: VLC ← (TDT:DDX,DDY,DDZ)

32: If MEMORY FULL then STOP else GOTO 33

33: X(n-2) ← X(n-1), Y(n-2) ← Y(n-1), Z(n-2) ← Z(n-1)

34: X(n-1) ← X(n), Y(n-1) ← Y(n), Z(n-1) ← Z(n)

35: TDT ← DT-63, FLAG=1 and GOTO 2
```

Fig 11-b. Power Collection and Decorrelation Algorithm.

Dower EKG Data Compressor

The Dower EKG compression technique, like the Tolan method, is a redundancy reduction procedure. The Dower compressor combines a zero order time compression operation with a second order difference reduction transformation to produce a decorrelated residual frame sequence. This residual frame sequence is then compressed by a specially tailored variable length code whose average codeword bit length approaches the entropy bound without the buffer overflow problem encountered with the ''optimal'' Huffman code.

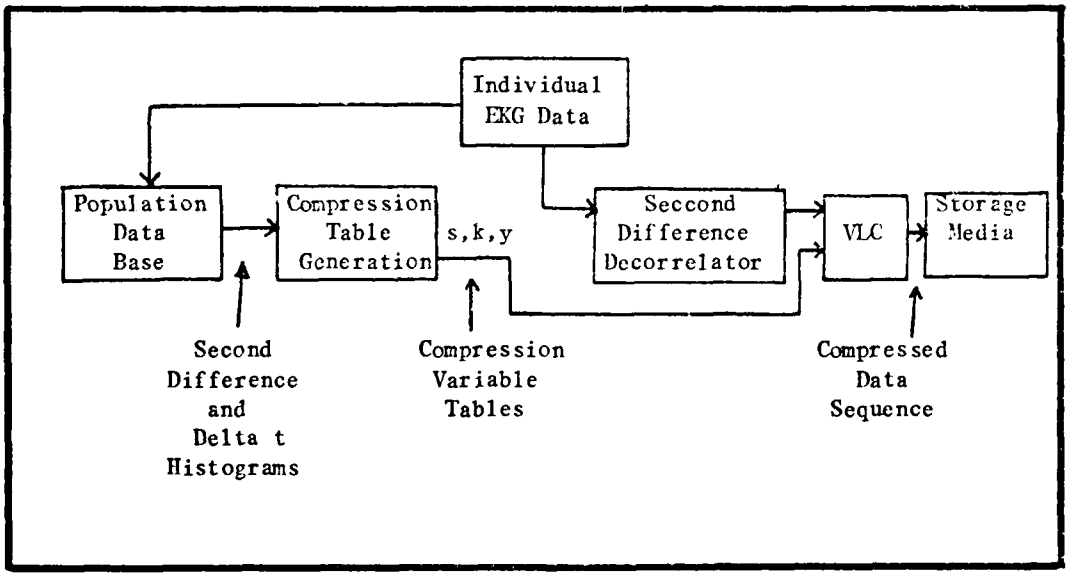


Fig 12. Dower EKG Data Compression System.

Dower Decorrelator. The Dower data collection and decorrelation operation (Figure 11) begins with a zero order time compression process where only those EKG sample values (x,y,z) which differ from the previous sample (Δx or Δy or $\Delta z \neq 0$) are saved along with a run length conter Δt. This operation produces a sequence of data frames (Δt: x,y,z). Following the creation of the data frames, a second difference with respect to frame number is performed generating a sequence of second order difference frames (Δt: $\Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$).

The Dower algorithm operates on 8 bit data, hence there are 256 potential source symbols for each lead. When second differences are taken, the range increases to 1024 potential Δ^2 symbols. By experimental evidence, Dower and Berghofer (Ref 12) have found that 0 $\pm 63\Delta^2$ values are sufficient to reproduce all but the fastest EKG artifacts (e.g.,pacemaker spikes). The variable length encoder, therefore, is designed to expect 127 source symbols (values) and no more. To insure the Δ^2 dynamic range of 0 ± 63 is not exceeded, the Δ^2 values are limited by a preprocessor shown in steps 18-26 of Figure 11. This preprocessor is iterative, and irreversably modifies the three sample points which produced the $|\Delta^2| > 63$ until the Δ^2 value falls within the encoder range. With care taken to record the necessary initial conditions (first x,y,z and first $\Delta x, \Delta y, \Delta z$),this second difference frame sequence now contains all the significant information in the original sample sequence with

reduced interframe correlation.

The mechanics of the Tower decorrelator are best illustrated by an example. Figure 13 shows three hypothetical sample sequences. By applying the rules in the preceeding paragraph, a set of zero order data frames (Δt:x,y,z) is formed. That is:

(0:0,-1,2),(0:1,1,1),(0:3,2,3),(0:4,3,1),(3:3,1,-3),(0:1,0-1),
(0:-1,-1,-2),(0:-2,-2,-1),(4:-3,1,-1)

To complete the correlation reduction process, the second difference with respect to frame number must now b performed. This results in the set of second differenc frames(Δt: $\Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$) shown below.

(0:0,-1,2),(0:1,2,-1)|(0:1,-1,3),(0:-1,0,0),(3:-2,-3,-2),
(0:-1,1,6),(0:0,0,-3),(0:1,0,2),(4:0,4,-1)

The data sets preceeding the verticle bar are the inital conditions necessary for reconstruction. The first data set is the first zero order time compression sample frame. The second frame is the first difference between the first two zero order data frames.

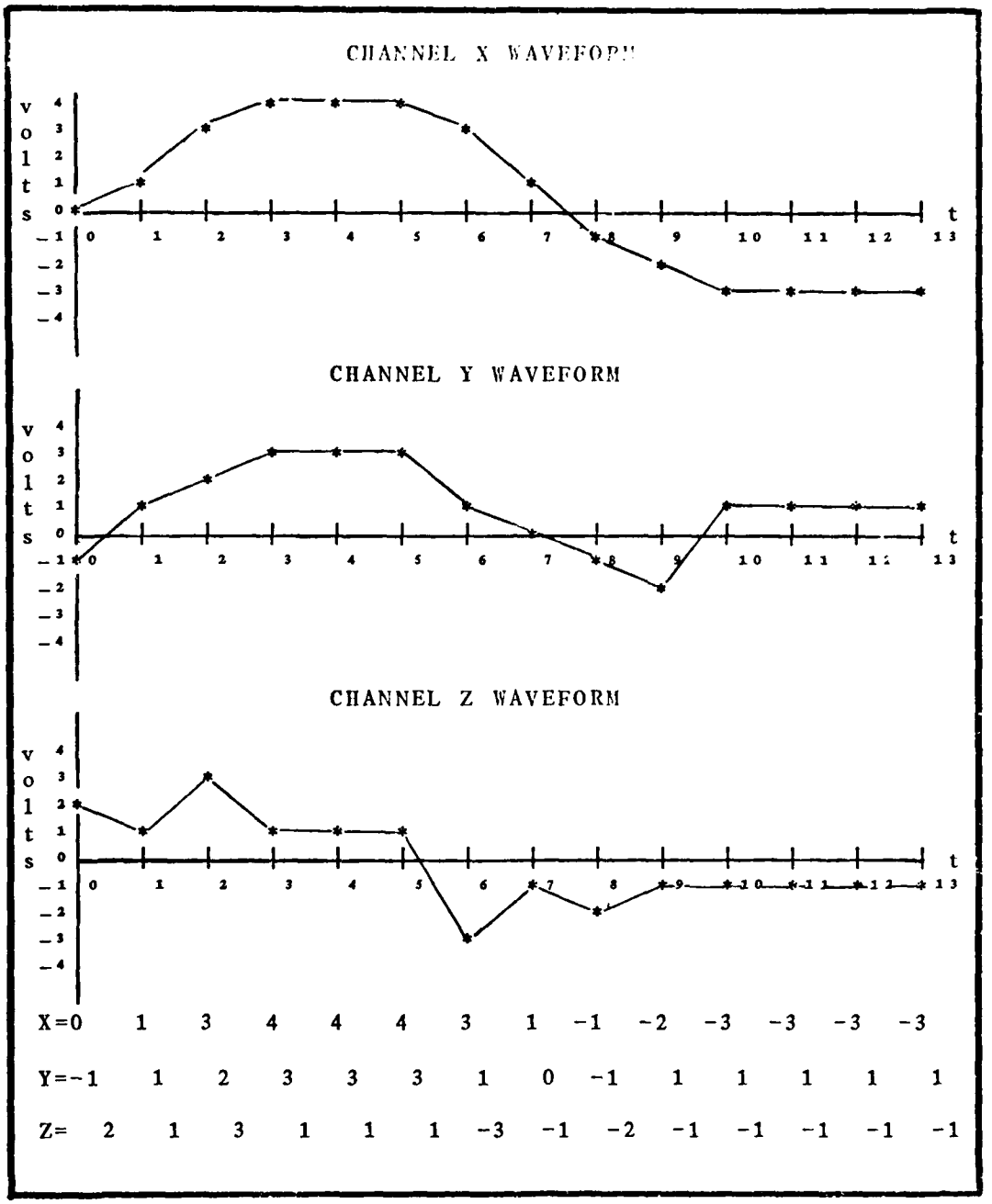


Fig 13. Three hypothetical sample sequences.

Dower Entropy Encoder. The Dower entropy encoder also uses variable length codes for the second stage of the EKG data compressor. The Dower code, however, is significantly different than the prefix code used in the Tolan compression method.

As was mentioned previously, the Dower VLC is configured for 127 source symbols (8 bit Δ^2 values). Unlike the Tolan code, however, the Dower VLC does not perform a 1:1 mapping between a single source symbol and a single code symbol.

In the Dower coder, the source symbols (Δ^2_i) are mapped onto the state space of a 14 bit accumulator (A). This state space is partitioned into symbol regions R_i which are assigned according to the probability of occurrence of the source symbols. The size ($||R_i||$) of the symbol region R_i is given by the relation

$$||R_i|| = ds_i = s_{i+1} - s_i = Pr(\Delta_i^2)16384 \qquad (24)$$

where s_i is the initial state of R_i and s_{i+1} is the initial state of R_{i+1}. The initial state, s_i, is also determined by symbol probability. The symbol regions R_i corresponding to thos symbols which occur least often are assigned to the low end of the state space range such that if:

$$Pr(\Delta_k^2) > Pr(\Delta_n^2) > Pr(\Delta_m^2) > Pr(\Delta_j^2)$$

then

$$||R_k|| > ||R_n|| > ||R_m|| > ||R_j||$$

and

$$s_k > s_n > s_m > s_j$$

This relationship is illustrated in Figure 14. For those potential symbols Δ_i^2 which do no occur, 1 state is always assigned.

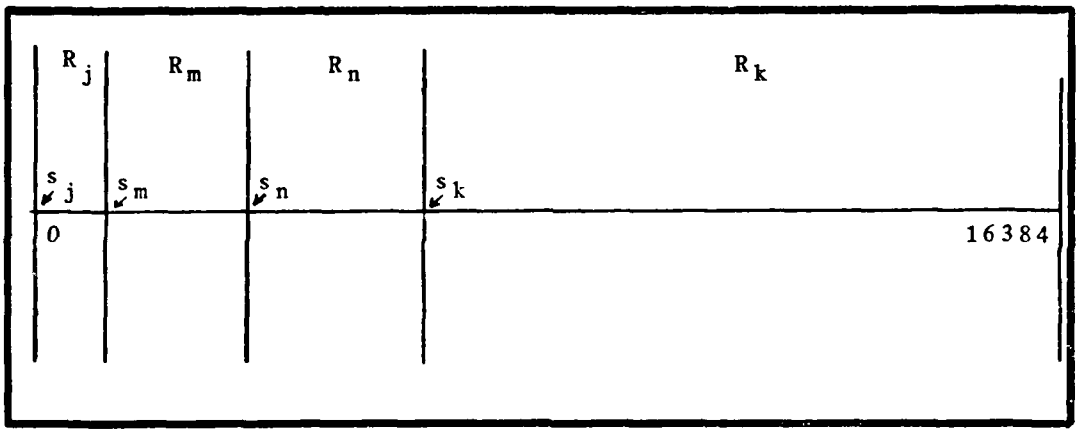


Fig 14. Example of State Space Partition for 4 Symbols.

The physical configuration of the Dower encoder is shown in Figure 15. As can be seen from Figure 15, the memory storage buffer for the encoded data is contiguous with accumulator and all accumulator bit shifts (right or left) also shift the entire memory buffer. Right shifting operations always shift a binary 0 into the most significant bit (MSB) of A.

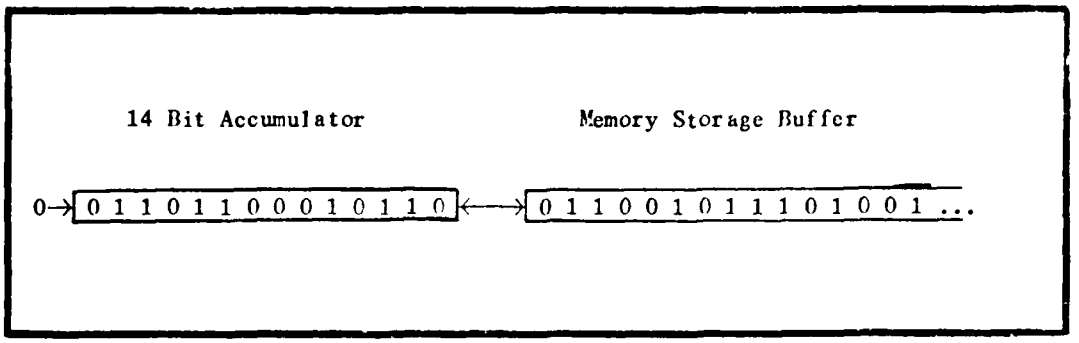


Fig 15. Dower Encoder Accumulator-Memory Buffer Interface.

A source symbol Δ_i^2 is encoded by adding the value s_i to the current contents of the accumulator. If the value in the accumulator is such that $s_i + A \geq 2^{14}$, then an arithmetic overflow condition would occur with a corresponding loss of data. To insure this overflow does not occur, the accumulator is first right shifted k_i times into memory. The value k_i is the maximum number of right shifts which would ever be necessary and can be found from the inequality:

$$2^{14} \leq 2^{k_i}(s_{i+1} - s_i) < 2^{15} \quad . \tag{25}$$

The k_i values are tabulated in Table I for various values of $ds = s_{i+1} - s_i$. This right shifting diminishes the value in the accumulator (i.e., divides A by 2 per right shift) until the value in A can be safely mapped into a region R_i. The initial state, s_i can now be added to A without overflow occuring.

In some instances, the maximum shift k_i would not be

necessary (i.e., $k_i - 1$ right shifts would still prevent overflow). To determine when this situation occurs, a third variable y_i is defined where

$$y_i = ds - 2^{14-k_i} \tag{26}$$

reperesents the *excess states* by which ds exceeds the next lowest integral power of 2.

Table I

Dower Code Variables Per Accumulator Partition Size

Partition Size	k	y
ds=1	14	0
1<ds<4	13	ds-2
4<ds<8	12	ds-4
8<ds<12	11	ds-8
16<ds<32	10	ds-16
32<ds<64	9	ds-32
64<ds<128	8	ds-64
128<ds<256	7	ds-128
256<ds<512	6	ds-256
512<ds<1024	5	ds-512
1024<ds<2048	4	ds-1024
2048<ds<4096	3	ds-2048
4096<ds<8192	2	ds-4096
8192<ds<16384	1	ds-8192

If the value in A $< y_i$ after the initial k_i shifts, then one too many right shifts occured. A is then *left shifted* once and s_i added to A without fear of overflow. If A $\geq y_i$ then a full k_i right shifts were required. To insure maximum efficiency in the next encoding, however, the position of the accumulator in state space map should be

right justified within the current symbol region. This right justification is obtained by adding $y_i + s_i$ to A whenever $A \geq y_i$.

By use of the variable y_i maximum efficiency in the number of right shifts (i.e. information bits stored to memory) can be obtained. Dower and Berghofer (Ref 12) assert that, on the average, the number of right shifts per source symbol (Δ_i^2) will approach quite closely to the value $-\log_2 Pr(\Delta_i^2)$ which is the self information ''content'' of the Δ_i^2 value (see appendix B).

The algorithm which implements the Dower encoder is shown in Figure 15. This algorithm assumes that the variable tables for $\Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$ and Δt have been calculated from the second difference histograms.

The Dower decoding operation is just the inverse of the encoding process. For decoding, the current state of the accumulator is mapped into the state space table of the current frame variable($\Delta t, \Delta_f^2 x, \Delta_f^2 y, \Delta_f^2 z$). Once the correct region R_i, is determined, then s_i is subtracted from A. If $A < 2y_i$ then one right shift is performed followed by k_i left shifts. If $A \geq 2y_i$, then y_i is subtracted from A followed by k_i left shifts. For a more detailed example of the Dower encoding/decoding operation, the reader is referred to Dower and Berghofer (Ref 12).

Although conceptually more difficult than the Tolan VLC, the algorithmic structure of the Dower entropy encoder is only slightly more complex. As was shown above, the

Dower VLC can be implemented solely with shifting and table look up operations. The Dower method does, however, require that code tables (s_i, k_i, y_i) be constructed prior to the encoding process.

1: Entry $(\Delta t, \Delta_f^2 X, \Delta_f^2 Y, \Delta_f^2 Z)$

2: IF this is the first frame stored then Acc$\leftarrow$0 else GOTO 3

3: DDVAR $\leftarrow \Delta t$, k$\leftarrow$ k(Δt), s$\leftarrow$ s(Δt), y$\leftarrow$ (Δt)

4: RIGHT SHIFT Acc k times

5: IF Acc $<$ y then GOTO 8 else GOTO 6

6: LEFT SHIFT Acc once

7: Acc $\leftarrow$ Acc + s and GOTO 9

8: Acc $\leftarrow$ Acc + s + y

9: IF memory is full, then SET eom flag and RETURN else GOTO 10

10: IF DDVAR=Δt then GOTO 11 else GOTO 12

11: DDVAR $\leftarrow \Delta_f^2 X$, k $\leftarrow$ k(Δ_f^2X), s $\leftarrow$ s(Δ_f^2X), y $\leftarrow$ y(Δ_f^2X) and GOTO 4

12: IF DDVAR=$\Delta_f^2 X$ then GOTO 13 else GOTO 14

13: DDVAR $\leftarrow \Delta_f^2 y$, k $\leftarrow$ k(Δ_f^2Y), s $\leftarrow$ s(Δ_f^2Y), y $\leftarrow$ y(Δ_f^2Y) and GOTO 4

14: IF DDVAR=$\Delta_f^2 Y$ then GOTO 15 else RETURN

15: DDVAR $\leftarrow \Delta_f^2 Z$, k $\leftarrow$ k(Δ_f^2Z), s $\leftarrow$ s(Δ_f^2Z), y $\leftarrow$ y(Δ_f^2Z) and GOTO 4

Fig 16. Dower VLC algorithm.

Tolan versus Dower

The similarites and differences between the Tolan and Dower EKG compression techniques are now discussed. The decorrelators are compared first.

Both the Dower and the Tolan decorrelators use a sceond difference technique where time compression is used to eliminate the storing of the most common Δ^2 value of zero. The Tolan algorithm, however, continuously calculates the second difference and applies these differences to the Tolan VLC. If a long ''run'' of $\Delta^2 = 0$ values occur, the Tolan encoder overflows at $\Delta t = 128$. This run counter overflow forces a storage ''dump'' to the VLC with a resultant loss in efficiency.

The Dower decorrelator approaches time compression slightly differently. In the Dower algorithm, only those sample points where the EKG data was changing (i.e., $\Delta x .or. \Delta y .or. \Delta z \neq 0$) are saved forming data frames. A second difference with respect to frame number is performed and the Δ_f^2's and Δt are fed to the VLC. Although this frame methodology appears more efficient than the Tolan technique, the Dower decorrelator is still constrained by a maximum Δt of 127 (i.e., maximum Dower VLC code range). On the basis of this analysis, is appears that both the Tolan and the Dower decorrelators have similar performance.

Since the Tolan and Dower decorrelators appear about equally efficient, the real compression payoff is in the variable length encoders. In the Tolan VLC, code words as

long as 1024+2 bits are feasible. These long codewords would only occur if an extremely large and fast ''spike'' (e.g. a pacemaker pulse) appeared in the data. In the population as a whole, spikes of this magnitude occur very infrequently. Nevertheless, the Tolan compression efficiency will degrade seriously in an enviroment where ''impulsive'' artifacts appear.

The Dower VLC is designed to minimize sensitivity to impulses. In the Dower system, the VLC is configured such that the longest codeword (i.e., the number of right shifts to memory) is 14 bits. Extremely large signal spikes are numerically filtered to reduce their second difference within the 0 ± 63 range. Such drastic limiting action would occur infrequently, however, and not seriously affect the reproduction of the EKG.

As an example of their performance, let a second difference of 45 be encountered by both the Tolan and Dower encoders. The Tolan VLC would require 47 bits to encode this data. The exact codeword size of the Dower routine is dependent on the $\Delta^2 = 45$ probability of occurrence. Nonetheless, the ''codeword'' is always $\leq$ 14 bits long ; an obvious increase in efficiency over 47 bits.

A calculation of the performance differential between the Tolan and Dower routines will be estimated in chapter 5. For now it is sufficient to say that the Dower EKG compressor outperforms the Tolan method. A short synopsis will now be made of two other EKG compression techniques.

C1 - Microcomputer EKG Compressors.

Two other EKG compression techniques were discovered by this author. The first is referred to as the Turning Point Method and is discussed in reference 32. The second method is currently in use by Marquette Electronics for data compression in a commercial EKG ''cart''. The Turning Point technique is presented first.

Turning Point Algorithm. The turning point algorithm is, by definition, a 2:1 data compressor where one of two consecqutive sample points is discarded. The algorithm which determines which sample point is discarded is as follows. The first sample point is stored and assigned as the reference point (X_0). The next two consecqutive points become X_1 and X_2. With 3 sample points there are 8 ''paterns'' or combinations which reflect the ''trends'' in data (see Figure 17). The Turning Point algorithm stores the circled point (Figure 17) which becomes the new reference point X_0. The point not circled (X_1 or X_2) is discarded. The next two points are sampled, their values are assigned to X_1 and X_2, and the process repeated.

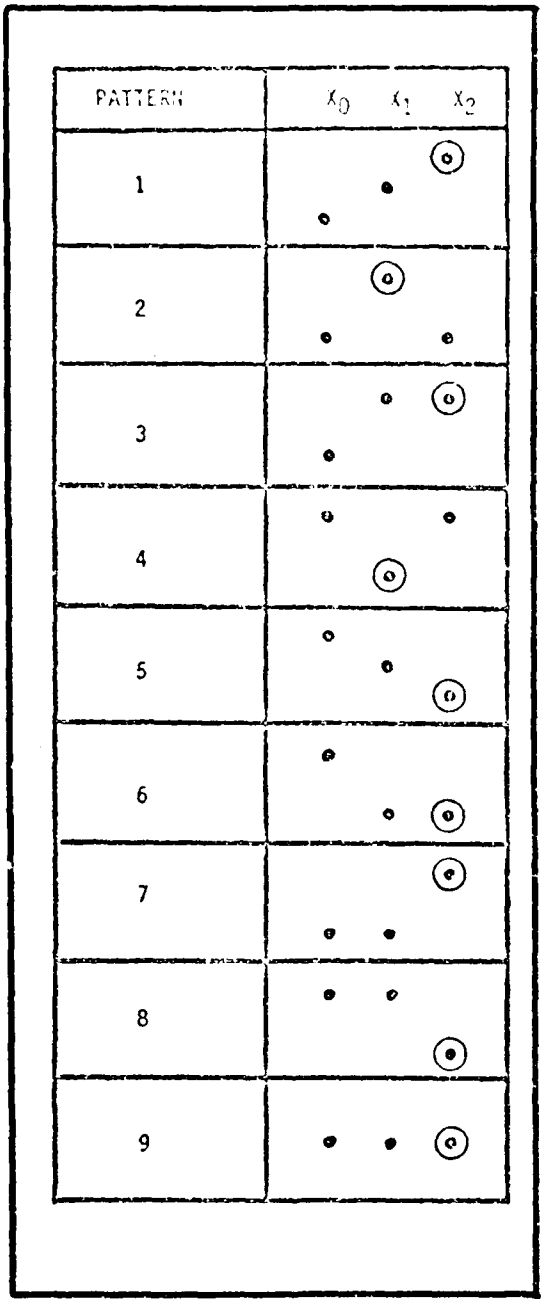


Fig 17. Turning Point Patterns (from Ref 32:6.61).

It can be shown (Ref 32:6.59-6.65) that :

$$\text{if } (X_2 - X_1)*(X_1 - X_0) < 0 \quad X_0 = X_1$$

$$\text{if } (X_2 - X_1)*(X_1 - X_0) \geq 0 \quad X_0 = X_2$$

hence the sign of the product of consecqutive first differences determines the ''signigicant'' point to be saved.

The advantage of the Turning Point compressor is speed of execution. Execution of this algorithm on even the slowest microprocessor would pose no problem at all. The disadvantages of the turning point routine are numerous, however. First the routine discards data so that reconstruction of the original sample sequence is not possible. In this regard, the Turning Point algorithm falls in the class of Entropy Reducing (ER) techniques as was discussed in chapter 2. Second no attempt is made to use the probabilistic distribution of the EKG to enhance data compression as is done with the Tolan and Dower routines. This obviously leads to inefficiency. Finally, this routine only produces a 2:1 compression ratio; incredibly poor in relation to the other techniques already discussed in this thesis. Only where the simplest technique is necessary, would the Turning Point technique be beneficial.

Marquette Algorithm. The Marquette algorithm was developed by Marquette Electronics, Milwaukee, Wisconsin. This similar to the Dower and Tolan algorithms in that variable length encoding is used to compress the output of a

''difference'' decorrelator.

In contrast to the Dower and Tolan techniques, the Marquette compressor only calculates the first difference for input into the VLC. The Marquette variable length encoder stores the first difference data as 1,3,5,or 7 nibbles where a nibble is defined as 4 bits. These nibble codes are arranged as follows:

Range of Difference	Code Length in Nibbles
(-7,7)	1
(-127,127)	3
(-2047,2047)	5
(-32767,32767)	7

To encode the first differences, and delineate between code words, the following rules apply:

1) Differences must be coded on the smallest possible range, and attempts to encode a difference of +5, for example, using more than 1 nibble will result in a decoding error.

2) Single nibble codes are difference plus 8. A nibble value of zero does not occur.

3) Three nibble codes start with a single zero nibble. The remaining two nibbles are obtained as follows:
   (1) Positive Differences +8
   (2) Negative Differences +7

4) Five nibble codes start with two zero nibbles. The remaining three nibbles are obtained as follows:
   (1) Positive Differences +128
   (2) Negative Differences +127

5) Seven nibble codes start with three zero nibbles. The remaining four nibbles are obtained as follows:

(1) Positive Differences +2048
(2) Negative Differences +2047

6) The sequence of nibbles in a code starts with the zero flag nibbles (if any) followed by the most significant through least significant nibble.

The above set of encoding rules were obtained from Mr. Tom Divers, Marquette Electronics project engineer (Ref 11).

From their own analysis, Marquette has shown (Ref 11) that at a 250 Hz sampling rate, 89.1 percent of the first differences fall within the ± 7 range with 99.8 percent falling within a ± 127 range. Marquette reports that at an A/D precision of 10 bits, an average of 4.89 bits/sample (across the total EKG population) is obtained with their compression routine.

The Marquette EKG compression appears to work well, even with a first order difference correlation reducer. The Marquette variable length encoder , however, is tailored to the 8 bit ASCII data communications enviroment and is not ''optimum'' in any sense. The Marquett VLC does perform ''exact'' redundancy reduction entropy compression with sufficient efficiency to make this encoding scheme commercially viable.

Chapter III Summary.

This chapter has looked in detail at two EKG data compression techniques which perform ''exact '' redundancy reduction. The Tolan routine, which was implemented by this author (see chapters 4 and 5) decorrelated the sample

sequence data by performing a second order time compression operation. The residual sequence resulting from the decorrelator was ''compressed'' by a uniquely decodable variable length code which was shown to be ''suboptimal'' with respect to the ''optimal'' Huffman code.

The Dower compression technique also performed a second difference operation, but preceeded the second order ''differencer'' by a zero order time compressor. The Dower zero order time compressor produced a sequence of ''data frames'' which in turn were converted to a sequence of second order difference frames with respect to frame number. Since the Dower VLC limited the Δt time compression counter to a maximum of 127 (as did the Tolan VLC), input symbols, the Dower and Tolan decorrelators were considered to perform equally well. The Dower entropy encoder was also shown to be a variable length coding operation but not a ''prefix'' code as was the Tolan VLC. The Dower VLC maps decorrelator ''symbols'' into the ''state space'' of 14 bit accumulator which encodes data by adding the initial address of a symbol's state space region to the accumulator. To prevent accumulator overflow, the accumulator data is shifted out to a memory storage buffer. The ''number of shifts'' necessary to prevent accumulator overflow represent the codeword size and it was shown that the Dower VLC approached the ''entropy'' bound of the decorrelated input sequence.

The chapter was concluded by a synopsis of two other

EKG compression techniques. The first of these two, the Turning Point algorithm, was shown to be of marginal use because it is not an ''exact'' technique and produces compression of only 2:1. The second technique, however, was the Marquette compression system and it was shown to produce acceptable compression worthly of commercial application.

From the available algorithms, the Dower compression technique has the capacity to produce the best ''exact'' compression of any of the techniques studied in this chapter. An interesting experiment would be the combining of the second order interpolator (discussed in chapter 2) used by Ruttiman and Pipberger (Ref 28) and the variable length encoder used by Dower (Ref 12). This combination should prove to be very powerful and effective and is left for futher study.

The next chapter in this thesis discusses the configuration of the EKG Data Acquistion and Analysis System assembled by this author to test the Tolan EKG compression algorithm.

IV. EKG-Data Acquisition and Analysis System

The EKG-Data Acquisition and Analysis System (EKG-DAAS) was assembled for this thesis as the testbed on which experimental EKG data could be acquired, compressed, analyzed, stored, and reconstructed. The EKG-DAAS design can be separated into the categories of hardware and software. The hardware is discussed first.

EKG-DAAS Hardware.

The EKG-DAAS was constructed around the Motorola Exorciser microcomputer (appendix E) which uses a 6800 microprocessor for its central processing unit (CPU). In the EKG-DAAS, the Exorciser is configured with 32 kilobytes (K) of read/write (RAM) memory and 16 K of read only memory (ROM). In addition, the Exorciser was equiped with the EXBUG debuging module which allowed interactive program debugging with preselectable software breakpoints, execution tracing modes, and CPU register display.

To provide extended memory, a Midwest Scientific Instruments (MSI) FD-8 Disk Memory unit was interfaced to the Exorciser and provides approximately 290 K of online user memory. The FD-8 is accessed by a MSI Disk Operating System (DOS) and communicates with the CPU via a MEX6820 Input/Output Module installed in the Exorciser chassis.

Terminal input/output (I/O) is accomplished by means of a standard RS-232 serial interface which has switch selectable baud rate from 110 to 9600 baud. In the EKG-DAAS configuration, the Exorciser serial ''port'' was connected in parallel with a Heathkit H-14 dot matrix line printer for program listings and data printouts.

Data I/O was accomplished by means of a Sinetrac ST-6800 Analog/Digital-Digital/Analog (A/D-D-A) converter module which samples and digitizes analog data to 12 bit precision. The ST-6800 has the capability of sampling 32 distinct analog channels (A/D) as well as output 2 channels (D/A) with simple memory addressed LDA (load) and STA (store) instructions. For the EKG-DAAS, the ST-6800 was setup for $\pm$ 5 volt, 2's compliment data and was addressed (A/D ch 0) at E400 Hexadecimal (Hex). The internal configuration of the Exorciser as used in the EKG-DAAS is illustrated in Figure 18.

To uniformly sample the EKG input data via the ST-6800 required the use of an external interrupt timer as is shown in Figure 19. This timer allowed data sampling rates between 300 and 700 Hertz but for the duration of this research was set, and calibrated, at 500 Hertz. The interrupt was interfaced to the Exorciser via an interrupt line on the ST-6800.

Considerable problems arose in this thesis due to hardware problems associated with the FD-8 Disk Memory. The original configuration of the EKG-DAAS used two FD-8 systems

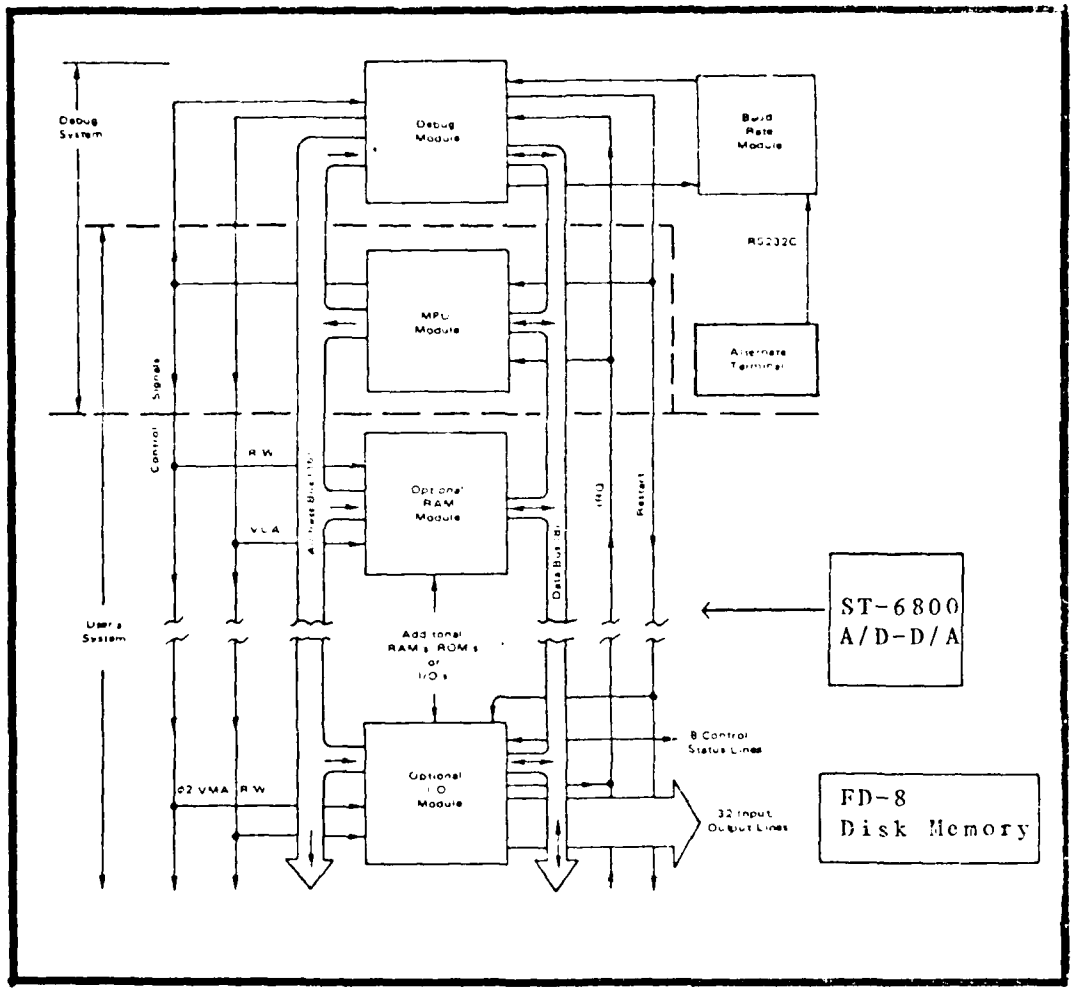


Fig 18. EXORCISER Component Module Layout (From Ref 22).

but one failed about midway through the software development. This failure caused a major rewrite of the thesis software and destroyed several weeks of work. Although the EKG-DAAS can now operate in a one disk environment, considerable ''manhandling'' of the data diskettes is necessary.

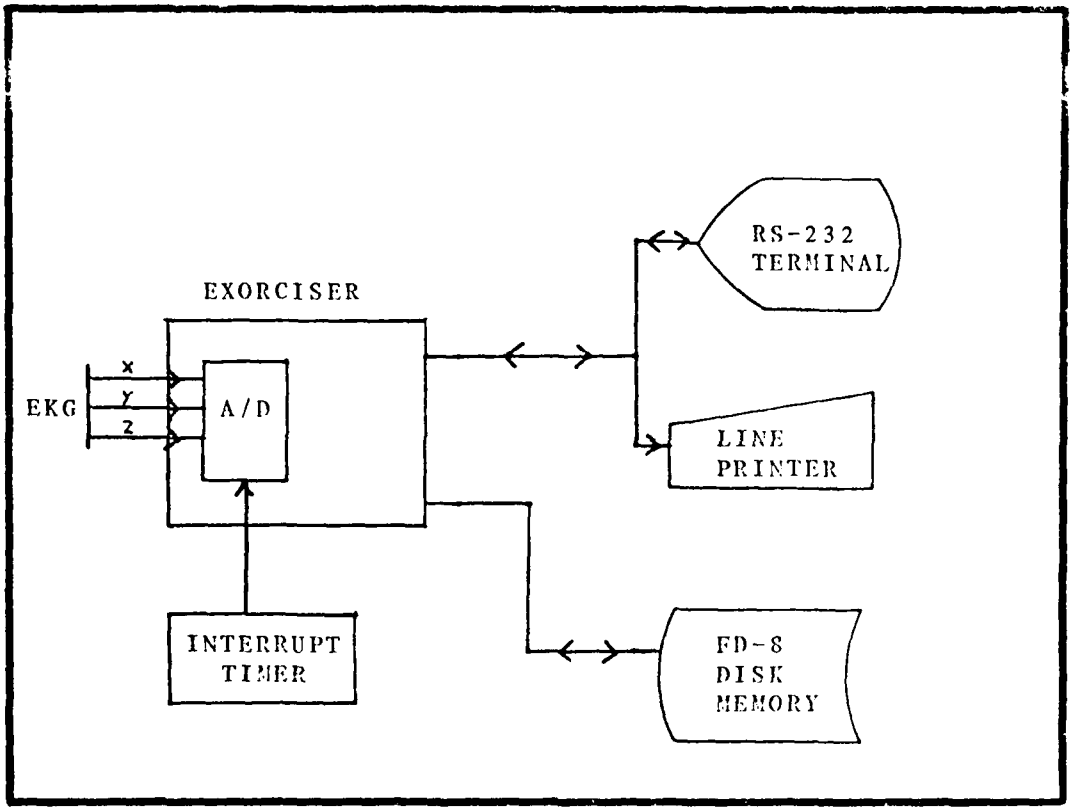


Fig 19. EKG-DAAS Hardware Configuration

Further information concerning the Exorciser's hardware capability can be obtained in appendix E and reference 22. Attention is now turned to the EKG-DAAS software.

EKG-DAAS Software.

The EKG-DAAS software was written in 6800 assembly language and controls all aspects of terminal, disk memory, and A/D-D/A operation. The EKG-DAAS programs consist of approximately 4300 lines of assembly language and are listed in appendix C.

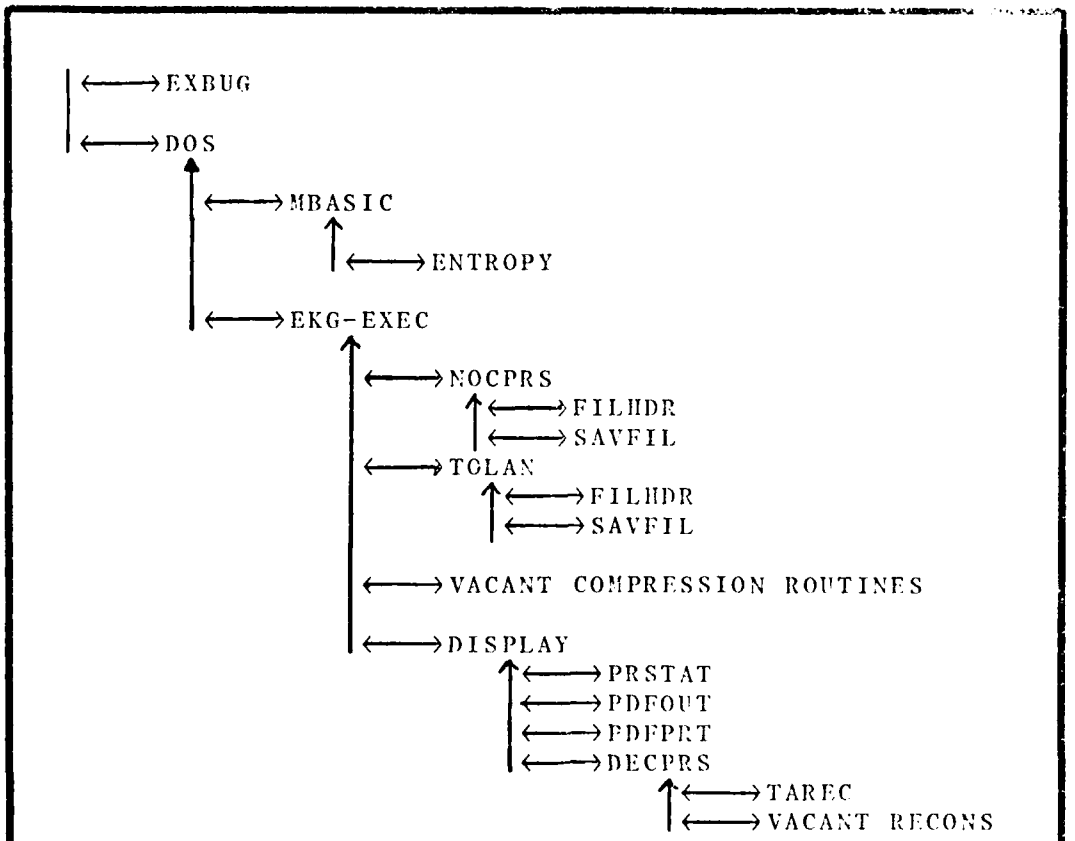
The software used in the EKG-DAAS was written in 6800

assembly language for two reasons: 1) no high order language (e.g. PASCAL, FORTRAN) was available on the Exerciser and 2) speed limitations imposed by the thesis requirement for online, real time EKG data compression made it imperative that the compression programs run as fast as possible. The laborious task of writing and testing assembly language slowed software development to the point where only one EKG data compression/reconstruction routine (Tolan) was completed.

The EKG-DAAS software is integrally tied to the MSI DOS (Ref 23). All disk I/O operations initiated by the EKG-DAAS routines flow through the MSI-DOS and hence the DOS must be ''live'' somewhere in memory. To insure that the DOS routines are always available, the DOS was disassembled and relocated in high memory ROM (C400 Hex).

The basic flow of EKG-DAAS program control is illustrated in Figure 20. The EKG-DAAS software is broken into overlayed modules which are called into memory and executed by EKG-EXEC and DISPLAY. The basic memory map and overlay structure is illustrated in Figure 21.

As can be seen in Figure 21, extensive memory management was required in order to allow a sufficiently large memory buffer for the EKG data. As configured in Figure 21, the EKG-DAAS could collect 11.6 seconds of uncompressed (3 leads,8 bits/lead,500 samples/sec) EKG data. For compression with the Tolan algorithm, a maximum of 26.2 seconds of data (TA1359PA, appendix D) was



DOS.....Disk Operating System
EXBUG.....Debug Monitor Program
MBASIC....Modified MSJ BASIC Interpreter
EKG-EXEC..EKG-DAAS Executive Program
NOCP S....Data Collection Without Compression
TOLAN.....Data Collection With Compression (Tolan algor)
DISPLAY...Data Output Module
PRSTAT....Statistic Printout Driver
PDFPRT....Amplitude Histogram Printout Driver
PDFOUT....Amplitude Histogram Oscilliscope Display Driver
DECPRS....Reconstruction and Display Module
TAREC.....TOLAN Reconstruction and Display (Oscilliscope)

Fig 20 EKG-DAAS Software Control Flowgraph.

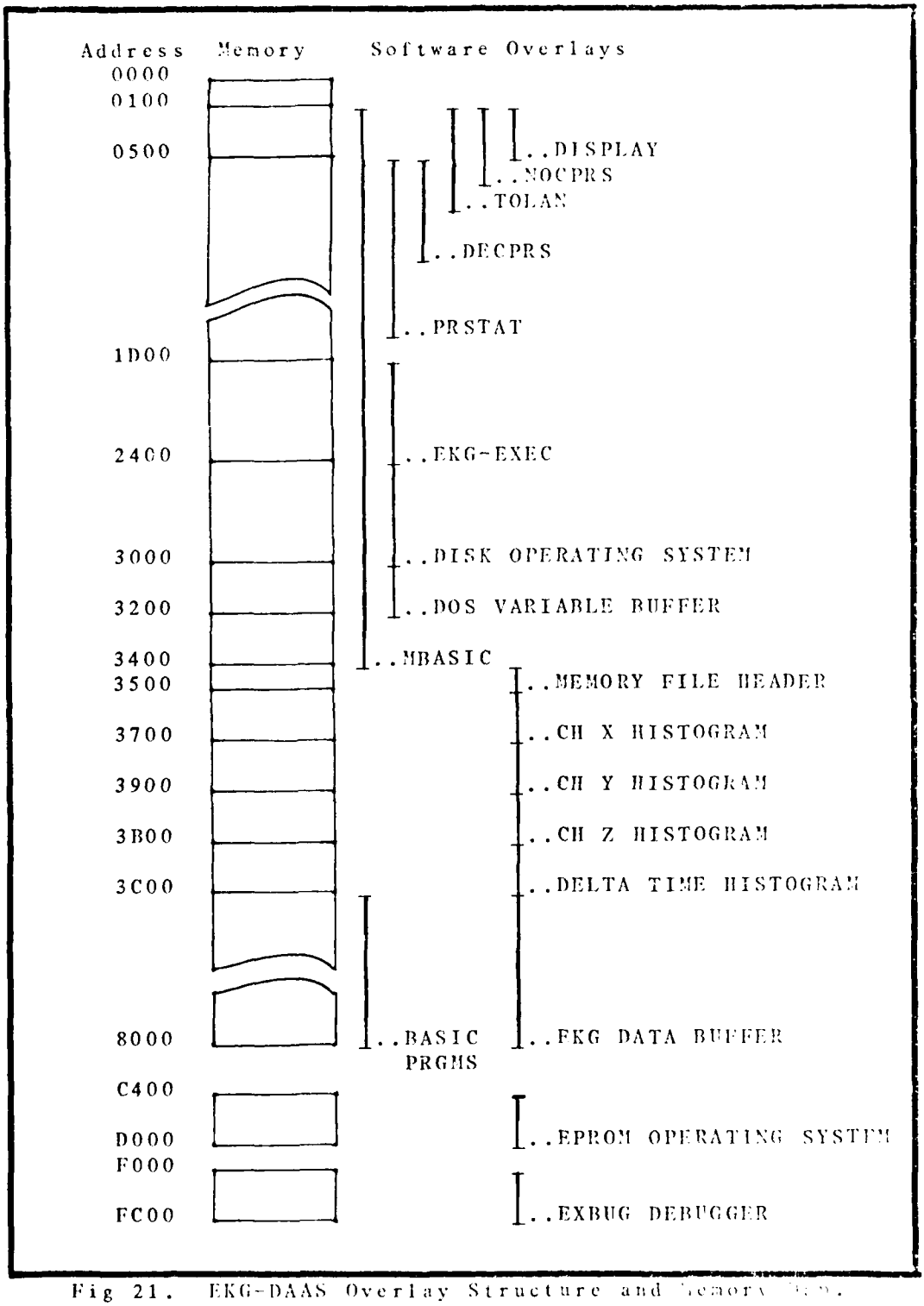


Fig 21. EKG-DAAS Overlay Structure and Memory Map.

collected.

With the basic structure of the EKG-DAAS software defined, this chapter will now examine the components of the EKG-DAAS software in more detail. In the next section of this chapter, a software module will be described along with a simplified flowchart of that module's operation.

EKG-EXEC. EKG-EXEC is the executive command module which controls the execution flow of the EKG-LAB system. Upon input of a command number, EKG-EXEC loads the appropriate overlay routine into the program work buffer (0100-1D00 Hex) and then passes control to that overlay. The above command and control operation is illustrated in Figure 22.

In addition to the command ''handler'', EKG-EXEC contains the utility subroutines FILHDR, SAVFIL, HXASC, OVRLAY, and PDFPRT. These subroutines are described as follows:

FILHDR. FILHDR clears the memory data buffer, initializes the statistics buffer variables, and queries the console for data such as FILENAME, SUBJECT, DATE, etc.

SAVFIL. SAVFIL reads the filename in the memory buffer header and then writes the memory file to disk. Disk I/O is passed through EOS subroutines.

HXASC. This subroutine converts hexadecimal data to ASCII for display on the terminal and printer devices.

OVRLAY. OVRLAY is the routine which actually performs the overlay function. After an overlay is loaded into memory, OVRLAY jumps program control to the overlay program.

PDFPRT. PDFPRT prints the amplitude distribution to the terminal device (printer). Although resident in EKG-EXEC, PDFPRT is called only by the DISPLAY module.

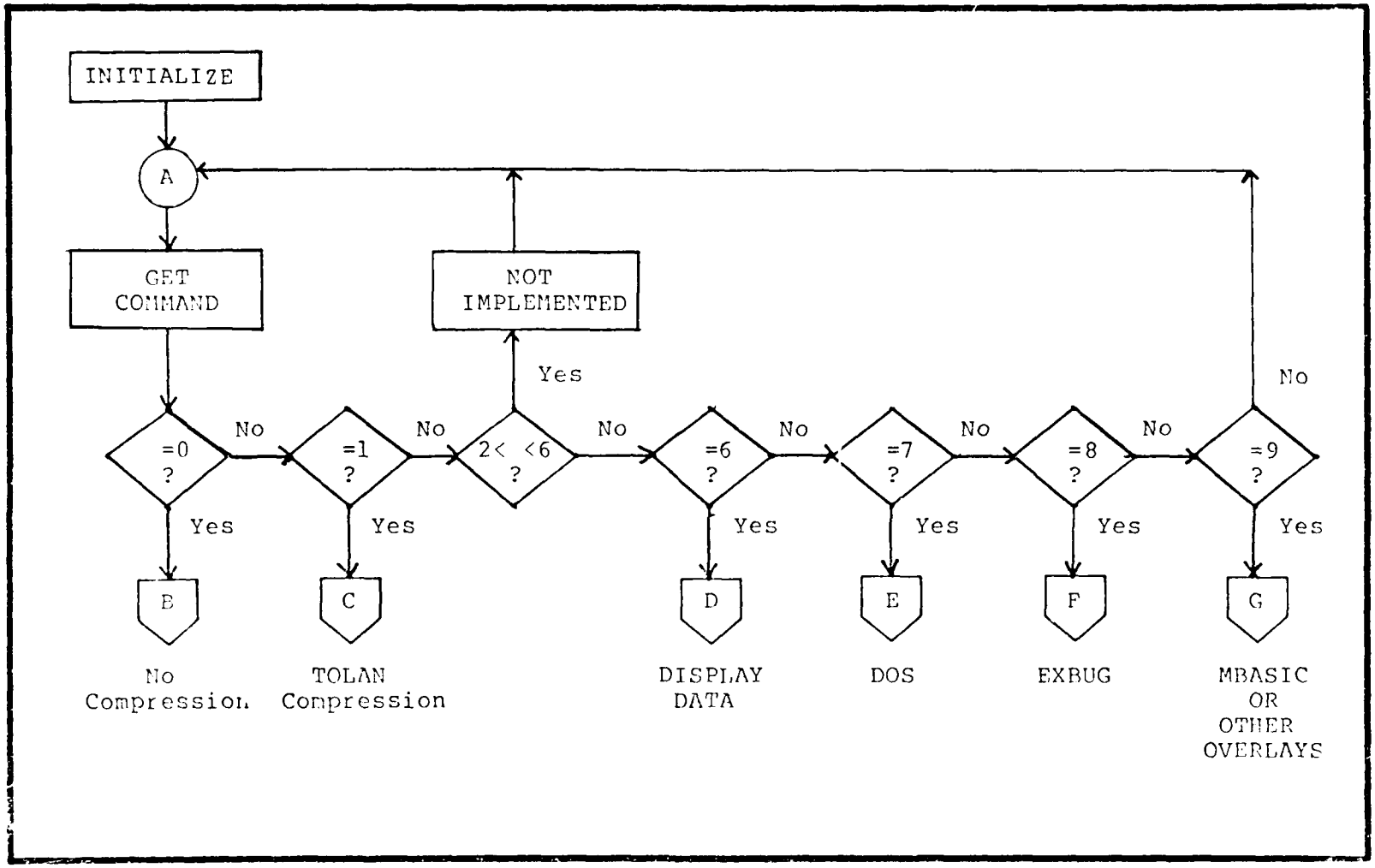


Fig 22. EKG-EXEC Functional Flowchart

NOCPRS. NOCPRS is a data acquisition module in which the EKG waveform is sampled and stored without data compression. The data is rounded to 8 bits from 12 bits. The reason for this rounding is explained in chapter 5.

This module was constructed for two reasons : 1) uncompressed data was considered useful for doing experimental studies on potential data compression techniques implemented after the original data collection session and ; 2)this module was the structure around which the Tolan (and potentially other) compression routines were built. The basic operation of NOCPRS is illustrated in Figure 23.

As is seen in Figure 23, NOCPRS does more than just sample the EKG. Statistical parameters are collected and updated during an EKG data collection. These parameters are used to measure the real time performance of the compression (no compression) software. A detailed description of these measurement parameters is described in chapter 5.

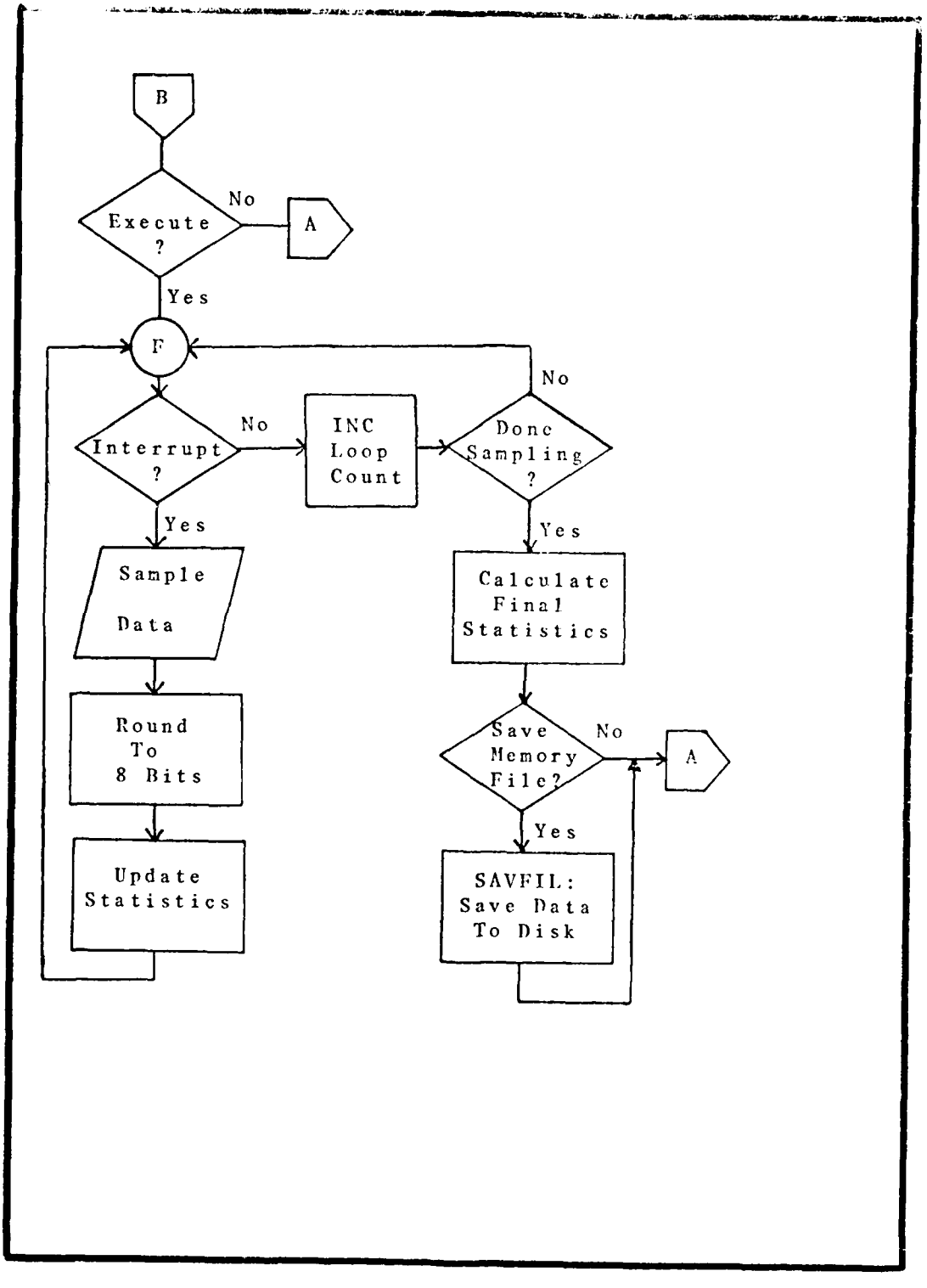


Fig 23. NOCPRS Functional Flowchart.

DISPLAY. DISPLAY is a command module similar in construction to EKG-EXEC. DISPLAY controls the data output modules which display the sampled EKG data and data statistics to two output devices. The first device is the terminal (printer) and the second device is an oscilloscope. The basic command structure of DISPLAY is shown in Figure 24. DISPLAY is broken into 5 working submodules. These modules are described as follows:

PRSTAT. This submodule reads the memory file header (3C00-3D00 Hex), formats the statistical data found there, and prints this data to the terminal (printer). The statiscal data in appendix D was generated by PRSTAT.

PDFPRT. This submodule prints the memory file histogram tables to the terminal (printer). PDFPRT output is also listed in appendix D.

PDFOUT. PDFOUT scans a user selected lead histogram (X,Y,Z) and formats the data for display to an oscilloscope. The data is output via D/A ch 0. An example of PDFOUT output is found in Figure 31 in chapter 5.

DECPRS. The DECPRS module scans the memory file header and identifies the compression technique which was used to encode the data in memory. The appropriate decompression algorithm is then selected and the data decoded and output on D/A channel 0.

LOAD. LOAD initiates a data file load from disk memory to RAM. This load is performed by EOS routines called by load.

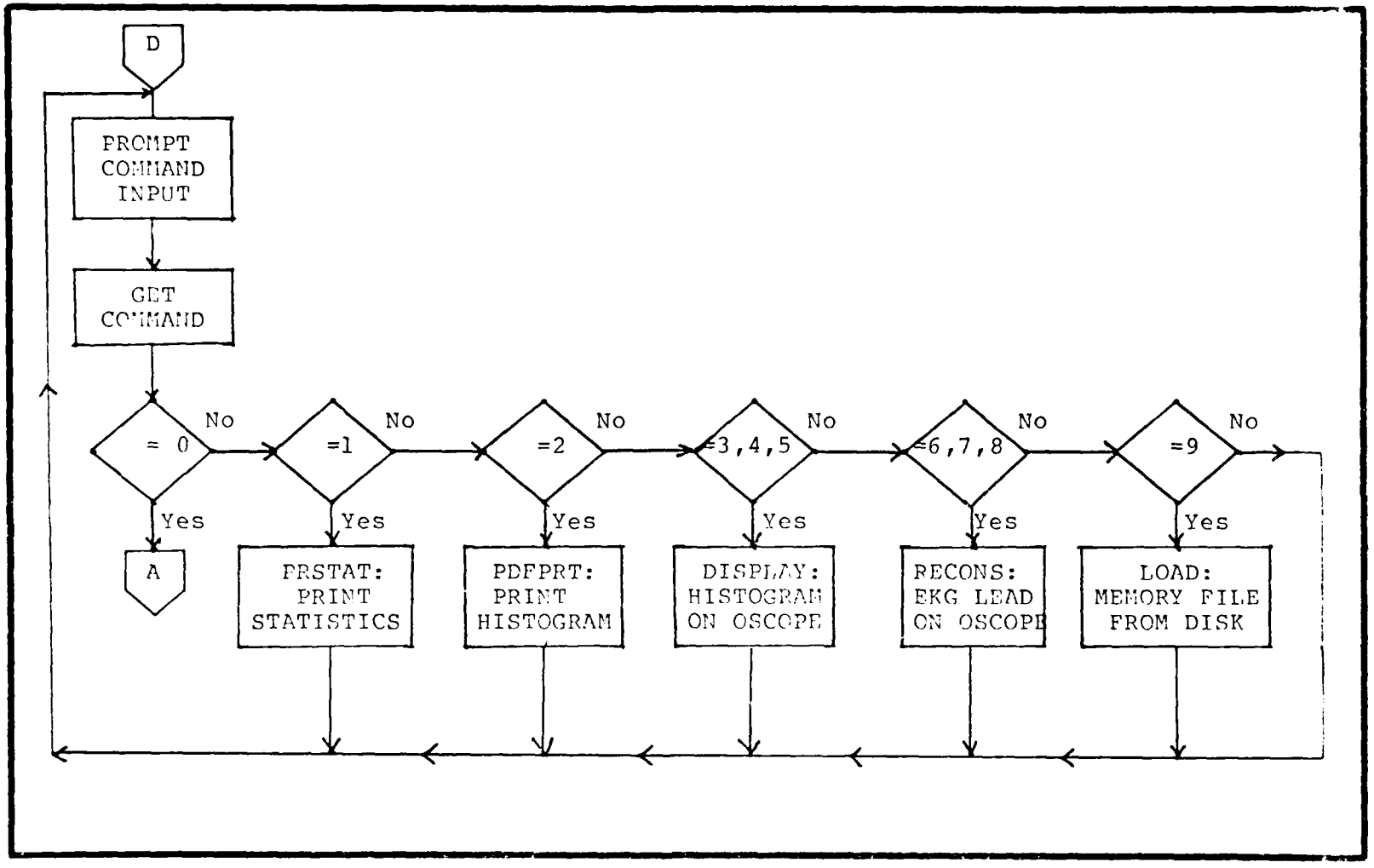


Fig 24. DISPLAY Functional Flowchart.

TOLAN. The TOLAN module is the ''heart'' of the EKG-DAAS and performs the actual data compression on the sampled EKG. As was described in chapter 3, the TOLAN algorithm first performs a second difference time compression operation followed by a variable length encoder. The operation of the TOLAN compression module is shown in Figure 25.

To detect sample clock (or CPU clock) drift, time calibration operations are performed prior to and after the data collection run. Other statistical data parameters are also collected allowing post collection measurement of compression performance.

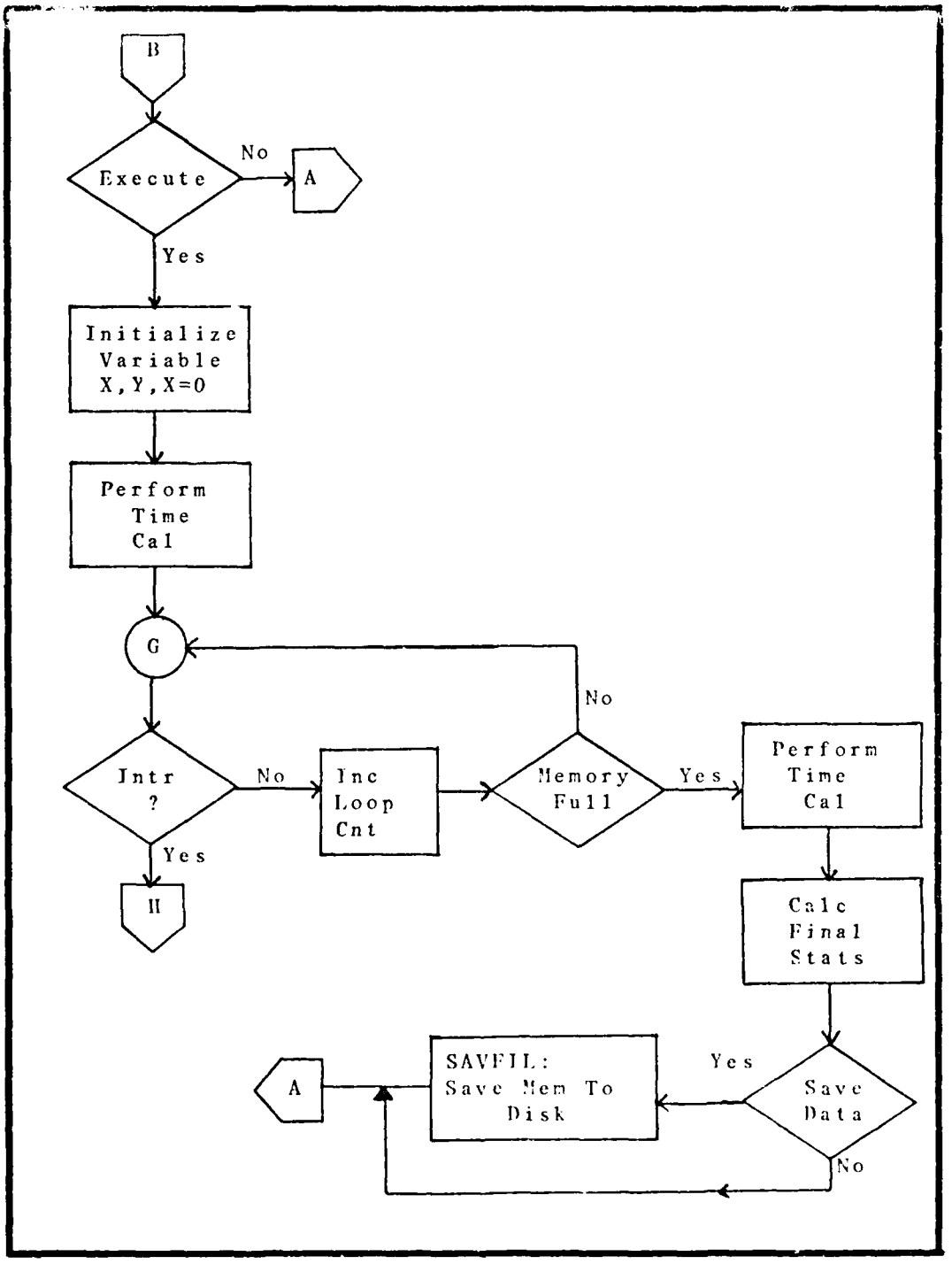


Fig 25a. TOLAN Functional Flowchart.

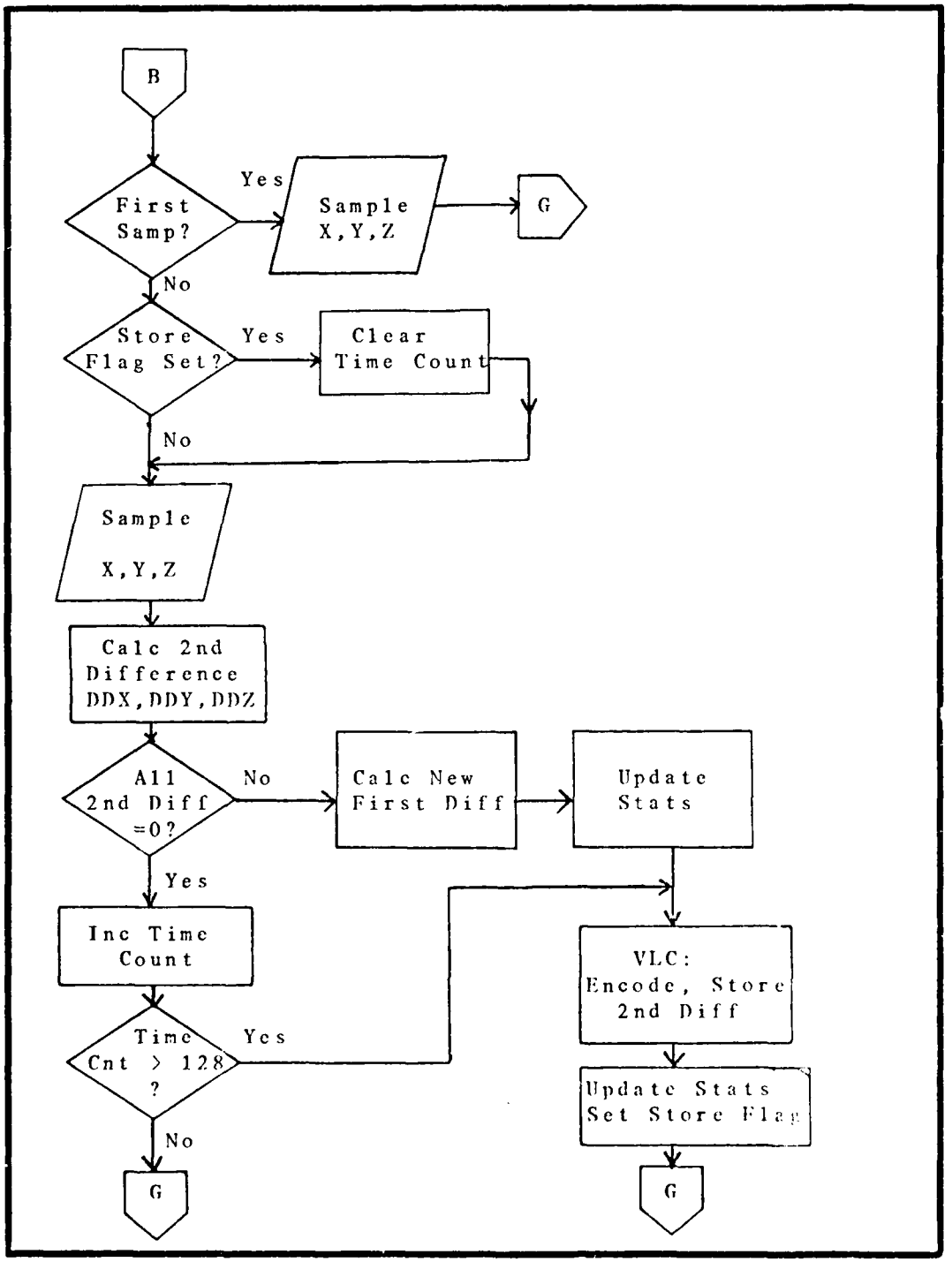


Fig 25b. TOLAN Functional Flowchart

Summary

This chapter has presented an overview of the complex software which makes up the EKG-DAAS. The reader is referred to the program listings in appendix C for a more thorough description of the program operation.

The software structure in the EKG-DAAS was written in a ''Top Down'' manner and all attempts have been made to document the operation of each routine. Since assembly language is difficult to read, this chapter was written to assist the reader in understanding the basic structure of the EKG-DAAS. The next chapter presents the results of the EKG experiment where data was collected, analyzed, and compressed by the EKG-DAAS.

V. Experimental Procedure, Data Analysis, and Results

This chapter presents the results of an EKG collection experiment where ''in vivo'' EKG data was taken from test subjects in real time. Data was taken and stored in both compressed and uncompressed formats for later analysis and reconstruction.

Chapter 5 is organized as follows. First the ''experimental'' setup is described along with a description of the EKG equipment, collection enviroment, and subject personnel. Next the parameters used to determine compression performance are defined followed by the analysis and results of the experimental data. Finally, the chapter concludes with a comparison between the results obtained using the Tolan compression algorithm and the estimated performance of the Dower technique. Discussion now turns to the experimental procedure.

Experimental Procedure

The data was taken from a set of nine fellow students during a laboratory course on electrocardiograms. The equipment used to produce the EKG was the model DR-12 research recorder built by Electronics for Medicine, Inc.(see appendix E). The DR-12 is a vintage medical recording system built in the late 1950's and is constructed with vacuum tube amplification circuitry.

The personnel used for test subject were all Air Force

officers in good physical health. No test subjects with obvious heart disease were used although significant EKG variations between subjects was noted (Fig 27-29). Output of the DR-12 was limited to one EKG signal which could be switched to any of the 6 ''limb leads'' (Ref 13:29-34) by controls on the DR-12. The electrodes of the EKG were applied to the wrists of the test subjects and in some cases, not all, an electrode jelly was applied to reduce skin-electrode resistance.

Since the EKG-Data Acquisition and Analysis System (EKG-DAAS) was configured for a 3 lead system, the X,Y,Z inputs were connected in common and the single signal available from the DR-11 applied to this connection. A Brush Instruments Mark II recorder and a Tektronics Model 465M Oscilloscope were used as analog output devices (Appendix E). The display instruments were connected in common with the A/D inputs as is shown in Figure 26.

Prior to the data recording session, the A/D was calibrated in accordance with the operating manual (Ref 8). The A/D was configured for a dynamic range of of -5.000 volts to +4.9976 volts with 2's complement binary representation.

The actual data collection proceeded as follows. First the subject was connected to the DR-12 and the amplitude of the resulting EKG signal adjusted to fall within the A/D dynamic range. A test run of all six limb leads (I,II,III,AVL,AVR,AVF) was then taken (without storage by

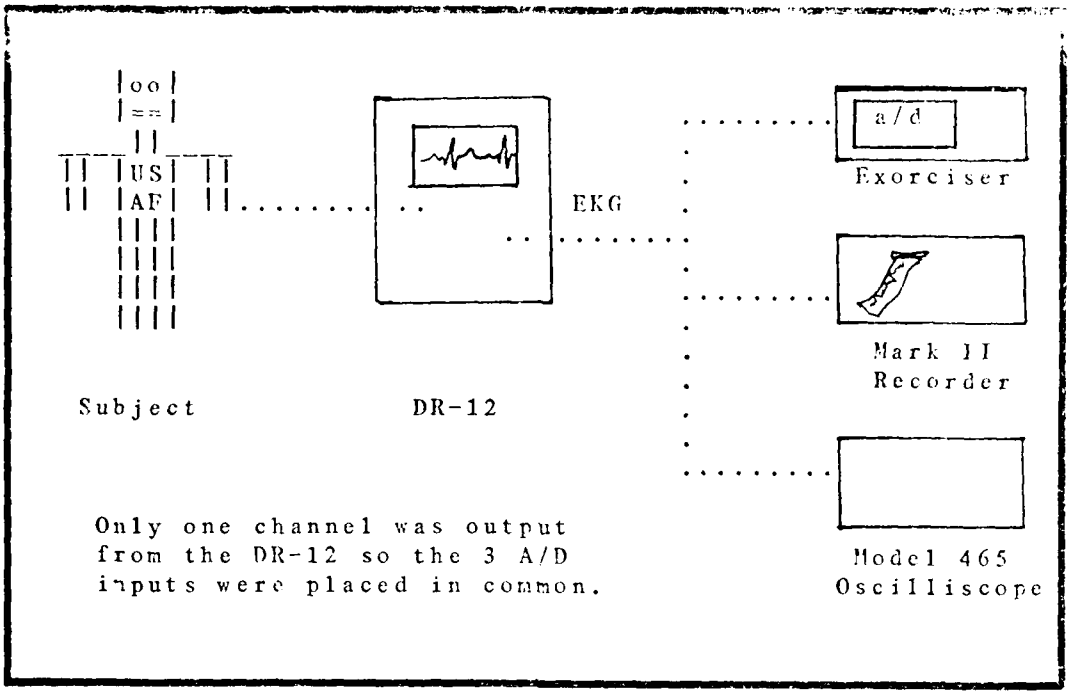


Fig 26. Experimental Data Collection Setup.

EKG-DAAS) and the lead with the least ''noisy'' signal was selected. EKG-DAAS was then executed and a data collection made <u>without</u> compression. This uncompressed data collection was followed as soon as possible with another ''run'' in which the Tolan compression algorithm was enabled. Following both data collections (uncompressed and compressed), the raw data traces from the Brush recorder were annotated with the time and subject and filed for later data comparison with the reconstructed waveform.

The data was stored on ''floppy'' diskettes and processed post collection for the entropy and maximum compression statistics. Before the results of these data

collections are presented, the compression measurement parameters will be defined.

EKG Compression Measurement Parameters

To permit determination of compression performance, a set of statistical parameters was calculated and saved during each data collection. These statistical parameters are described as follows:

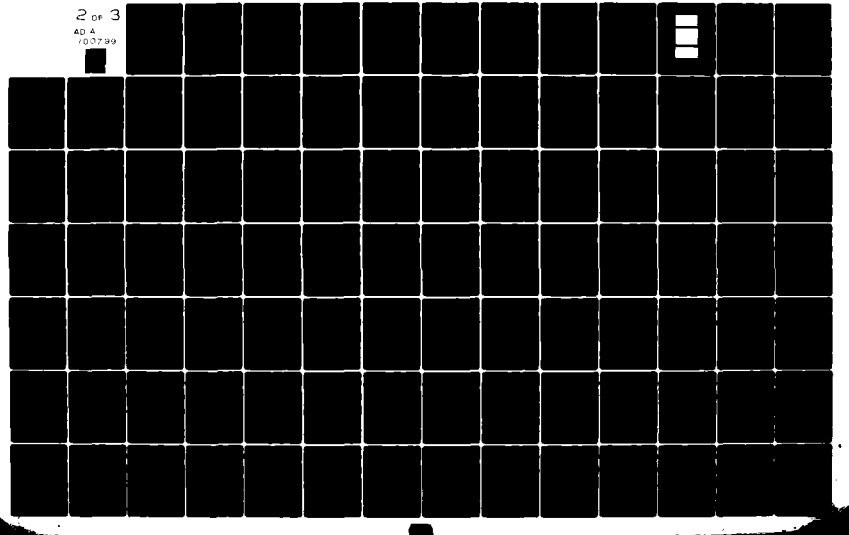
Number of Samples. This statistic was saved to determine the total number of bits that were input to the compression stages. The total number of bits were calculated by (8 bits/sample)* 3 leads)*(num of samples/lead).

2nd Difference Frequency of Occurence. Four frequencies of occurrence tables were kept with double precision binary counters. Following the data collections, these $\Delta^2x, \Delta^2y, \Delta^2z, \Delta t$ histogram tables were input to a BASIC program (*ENTROPY*) where the entropy of second difference ''source'' was calculated.

Total Waiting Loop Counts During Collection. A counting loop was established in the TOLAN compression module which allowed determination of the percent of the sampling period used for the compression and statistics calculations. One circuit of this counting loop takes 46 machine cycles of the 6800 microprocessor. A count of the total number of loop cycles completed following the sampling/compression interrupts is kept in the collection statistics buffer.

Maximum Loop Count Per Interrupt. To offset the inaccuracy which would develop if the interrupt clock period changed between runs (or if the master clock in the Exorciser drifted), a loop count calibration was performed immediately before and after each collection run. This calibration was accomplished by performing 256 sequential interrupts with no interrupt processing except return-from-interrupt. The before and after calibration counts were then averaged and the maximum loop counts per interrupt calculated. Time Efficiency of the compression operation was

AD-A100 799 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 6/5
ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA C--ETC(U)
DEC 80 M D TOWNSEND
UNCLASSIFIED AFIT/GE/EE/80D-46 NL



then calculated from the equation:

T.E.=(1-(Total Loop Count/((Num of Samples)*(Max Cnt))))*100

Channel Maximums and Minimums. The channel maximums and minimums were retained to allow determination if the analog inputs exceeded the A/D dynamic range.

Number of Memory Bits Available. This number was constant and was determined by the amount of Read/Write (RAM) memory available for data storage. For the current configuration of the Exorciser and the EKG-EXEC program this was 139248 bits (17406 bytes).

Number of Bits Available to Variable Length Coder. This counter measured the number of bits out of the data decorrelator and allowed calculation of the decorrelator's compression ratio (bits out/bits in).

Number of Bits Used to Store Channel X,Y,Z. These counters measured the number of bits used to store the data from the three input leads. This count is the number of code bits out of the variable length encoder.

Number of Bits Used to Store Time. This counter was identical to the channel counters above but measured the number of code bits used to store the Δt run counts.

Total Compression Ratio Achieved. This figure was calculated post collection by dividing the total code bits stored by the total R/W memory bits available.

The statistical data defined above was compiled by the EKG-EXEC program and is printed by the DISPLAY software module as illustrated in appendix D.

Fig 27. Original and Reconstructed EKG with Best Compression Ratio (2.260 : 1)

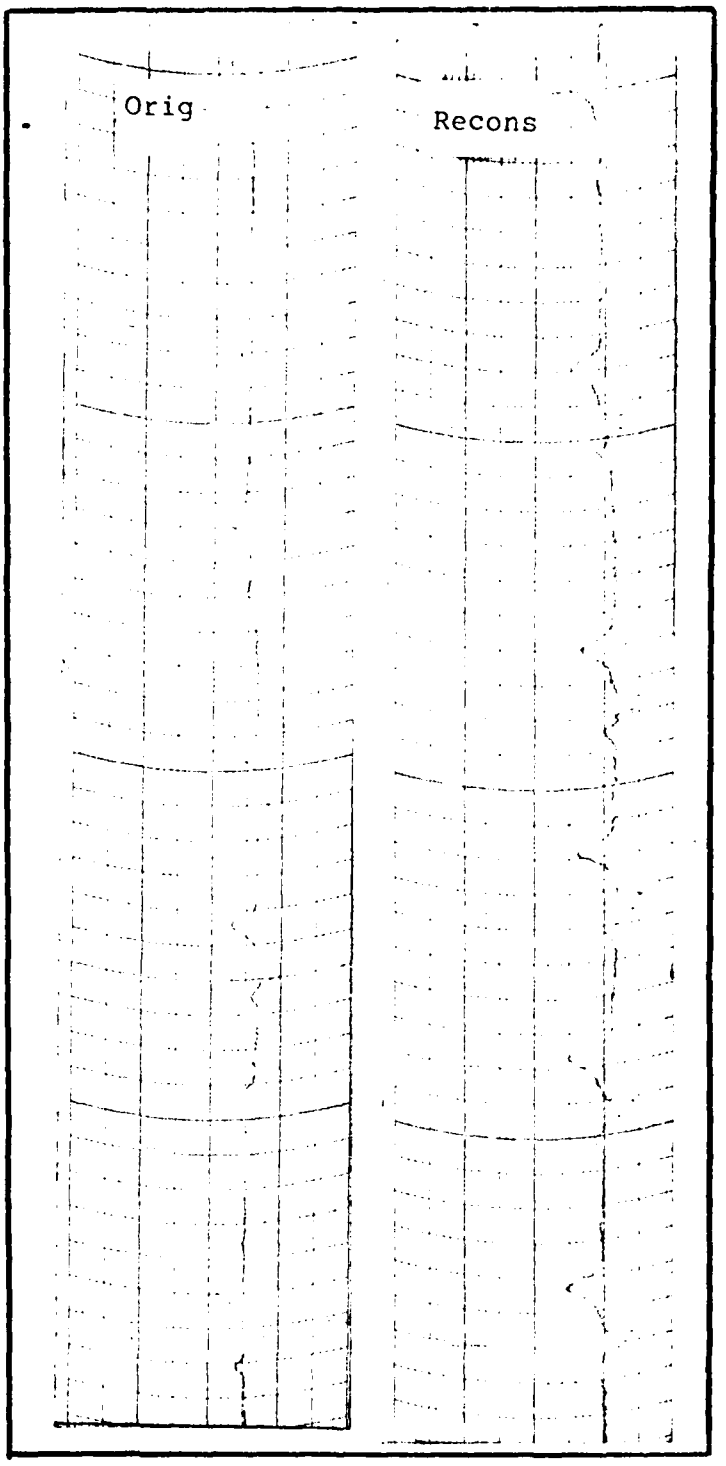


Fig 28. Original and Renconstructed EKG with Average Compression Ratio (1.60 : 1)

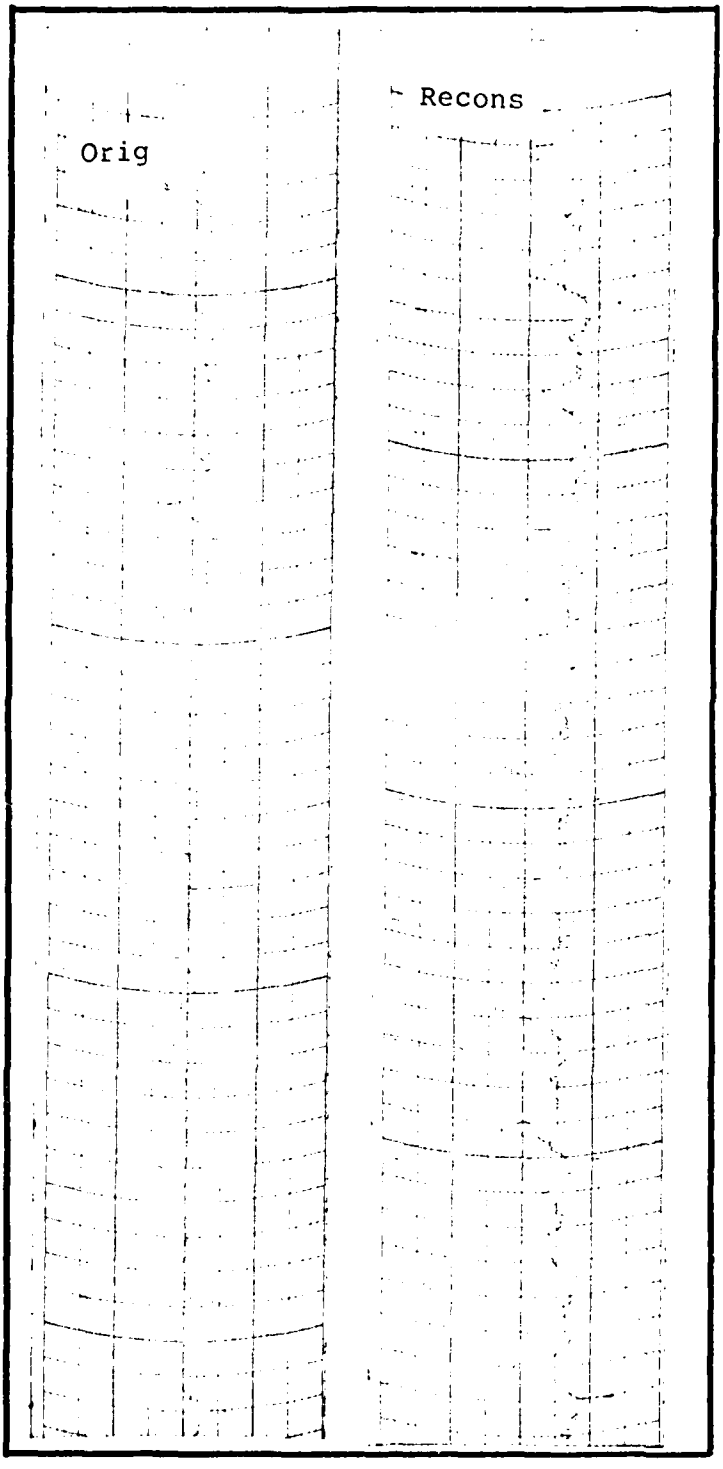


Fig 29 . Original and Reconstructed EKG with Worst Compression Ratio (1.252 : 1)

Data Analysis and Results

To determine the compression efficiency of a redundancy reduction EKG compression algorithm is, in general, a difficult job to accomplish. The reasons for this difficulty are: 1) the sampled EKG data sequence contains significant correlation (memory) between sample values making calculation of the absolute bound of the entropy extremely difficult (Ref 34:479-489) and ; 2) the techniques used for both the data decorrelator and entropy encoder vary significantly from algorithm to algorithm.

In this thesis, as is done in the literature (Ref 7,12,28), the assumption is made that the output of the decorrelator is ''almost decorrelated''. Decorrelated is not ''independent'' (unless the source was statistically gaussian), but it is assumed that true entropy of the 2nd difference sequence approaches the value which would be calculated by Eq.(1) (reproduced below).

$$\Delta^2 \text{ entropy} \cong - \sum -p_i \log_2 p_i \qquad (27)$$

(p_i is the probability of the i'th second difference).
This second difference entropy can then be used as an upper bound on the potential entropy encoding compression of the Tolan redundancy reduction technique.

The Δ^2 entropy values tabulated in Table II were calculated by Eq.(27). To calculate the approximate limit on the entropy encoder's compression ratio, the uncompressed data word length of 8 bits was divided by the ''lowest''

average code word (i.e., the 2nd difference entropy). A look at Table II shows that the entropy encoder compression ratio varied between 58% and 71% of this entropy bound. Since the Tolan variable length code is suboptimal, a lower efficiency is expected. Nevertheless, the Tolan entropy encoder performed more-or-less consistently across the data set. This last observation would imply that the data decorrelator influences the overall compression ratio more

Table II
Experimental Data Summary

Subject Id	2'nd Difference Entropy	Maximum Compression Possible (Approx)	Achieved 2nd Difference Compression	Percent of Max Encoder Comprs	Percent of Sample Interval	Achieved Total Comprs
TA1545T	3.2801	2.42 : 1	1.55 : 1	64.0%	93.9%	1.36 : 1
TA1548T	3.0171	2.63 : 1	1.61 : 1	61.2%	91.4%	1.50 : 1
TA1559B	3.016	2.64 : 1	1.72 : 1	65.1%	93.0%	1.53 : 1
TA1511S	3.323	2.39 : 1	1.65 : 1	69.0%	94.0%	1.43 : 1
TA1520B	2.601	3.06 : 1	1.81 : 1	59.1%	90.9%	1.72 : 1
TA1448L	2.930	2.72 : 1	1.73 : 1	63.6%	90.6%	1.60 : 1
TA1439S	3.267	2.43 : 1	1.63 : 1	67.1%	93.4%	1.43 : 1
TA1359P	2.487	3.22 : 1	1.88 : 1	58.3%	77.8%	2.26 : 1
TA1413L	3.783	2.10 : 1	1.50 : 1	71.4%	95.4%	1.25 : 1

than the entropy encoder.

As was described in chapter 3, the Tolan data decorrelator uses time compression in conjunction with a 2nd

difference operation. The Tolan time compression technique makes the assumption that the second difference value of zero occurs so frequently that the encoding of 0 would be less efficient than the storing of a zero value run counter. In the experimental situation in this thesis, signal noise was quite evident in the EKG traces (Fig 29). The sharp ''spikes'' induced by noise are accentuated by the second difference operation, hence a $\Delta^2 \neq 0$ was a common occurrence. This forced the storage of a lot of 7 bit time counters.

In 8 out of the 9 compression runs made, the frequent storage of time counts actually caused the Tolan data decorrelator to produce negative compression (i.e. more bits out than went in). Since the entropy encoder was producing a larger positive compression ratio, the overall compression figure remained positive. The effect of this operation is graphically illustrated in Figure 30.

A look at the original and reconstructed EKG traces (Fig 27-29) in conjunction with the data in Table II, shows that as the ''noise'' level increased on the signal the compression became progressively worse. Since it was concluded that the entropy encoder performed approximately the same across the data set, the degradation in total compression must be due, in large degree, to the degradation in the Tolan time compression data decorrelator performance.

A clear effect of the noise is illustrated in Figure

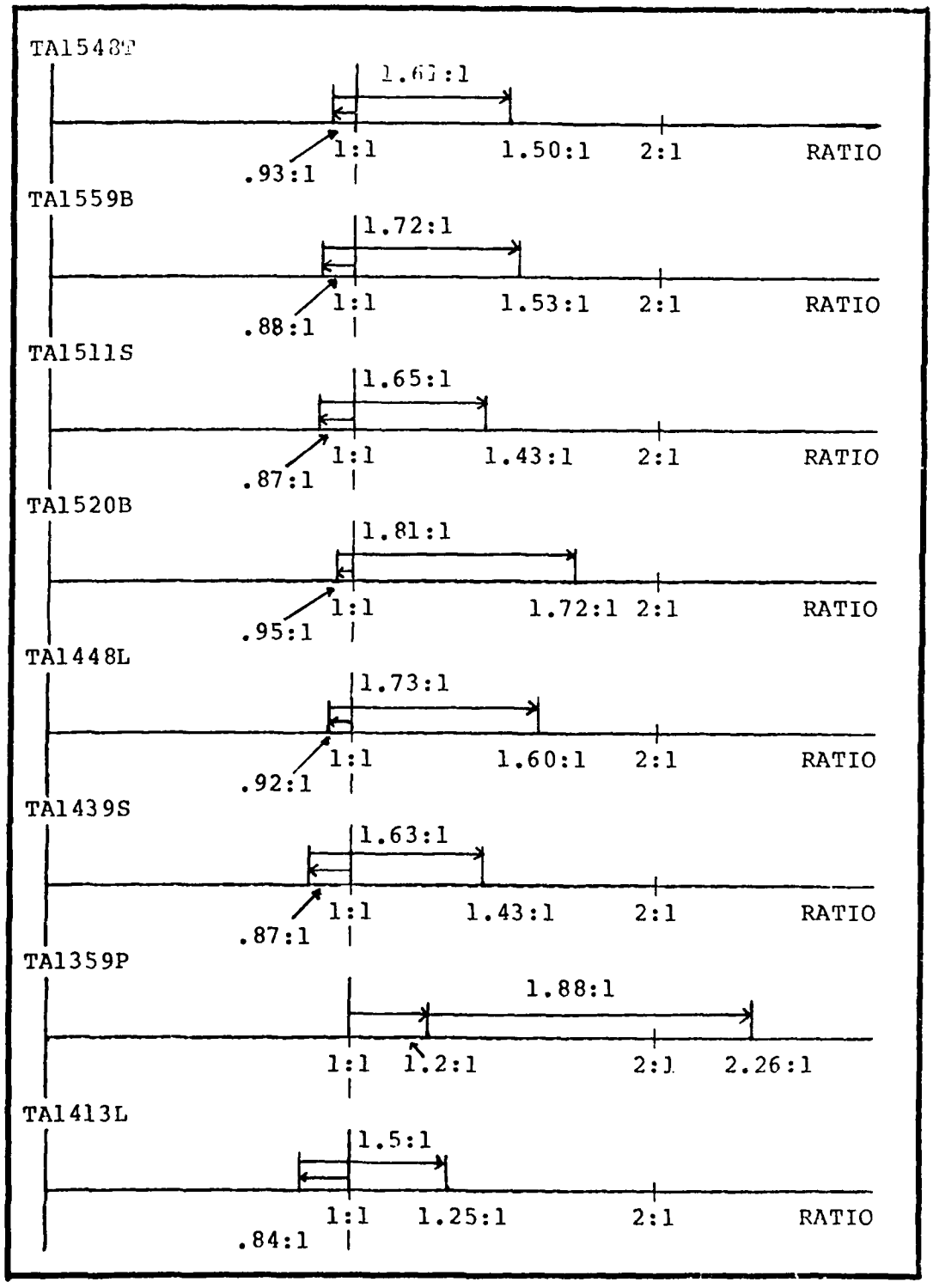
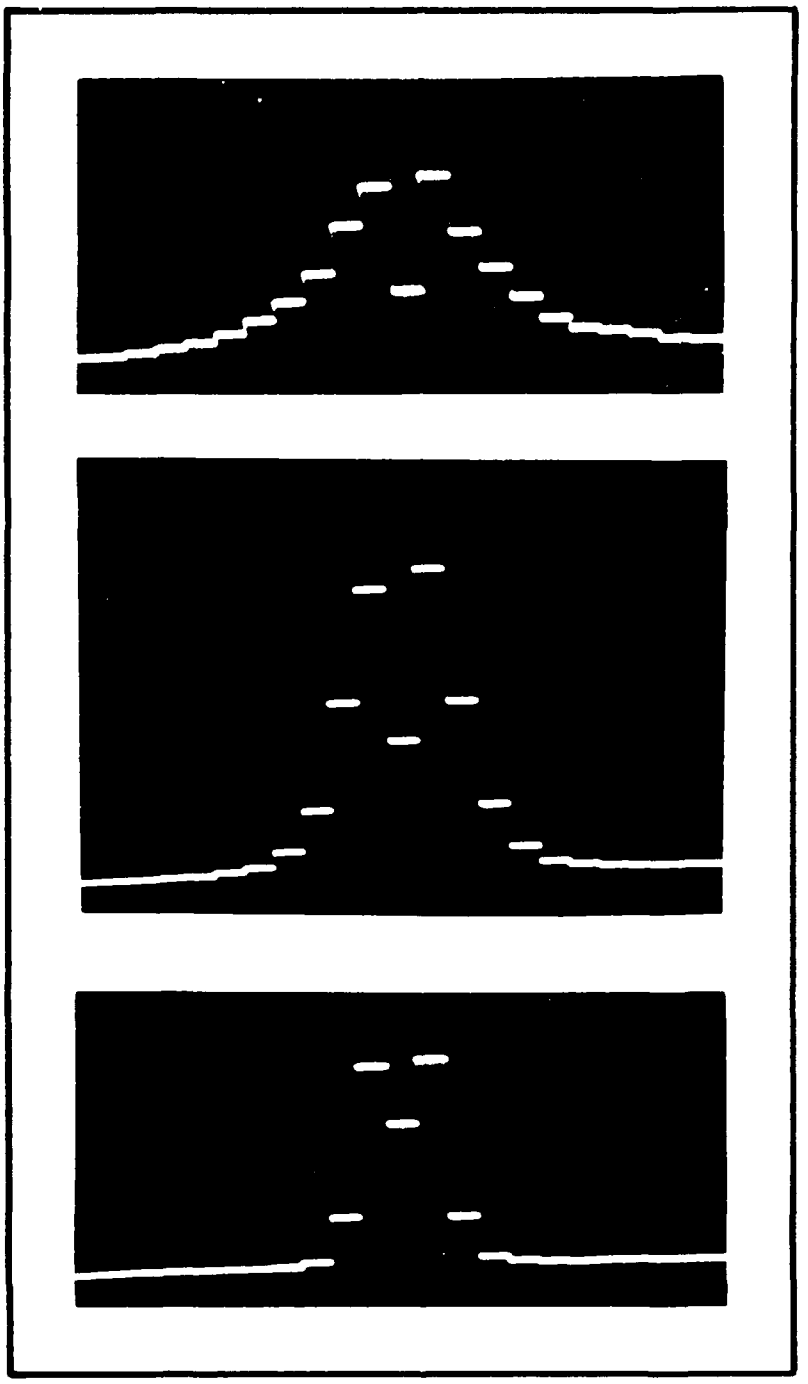


Fig 30 . Tolan Compression Ratio Breakdown.

31. From Figure 31, it can be seen that as the ''noise'' level increased the variance of the 2nd difference distribution became increasingly larger. As the distribution became less peaked, the efficiency of the variable length encoder decreased with a subsequent loss in achieved compression ratio.

The nonzero component at $\Delta^2 = 0$ in Figure 31 can be explained as follows. The decision was made early in the design of the EKG-DAAS that 8 bit data would be used versus 9 or 10 as recommended by the American Heart Association (Ref 3). This decision was made to simplfy the software (i.e. single precision could be used). Since the A/D converter has 12 bit resolution, the sample was rounded to 8 bits for uniform error distribution (Ref 24:424-432). Unfortunately because of this rounding action, small differences in the least significat bits of the A/D converter may have affected the rounding operation. Since the A/D cannot sample all three channel simultaneously, the probability that a ''noisy'' signal will change the least significant bit (or bits) is high. With a rounding operation, these changes may ripple to affect the least significant bit of the 8 bit data.

At most this effect would only cause a change in the least significant bit of the 8 bit rounded values. This would cause, however, the second differences between of channel X,Y,Z to be different even though they were connected in common. As was described in chapter 3, any of



the three leads with a non zero second difference forced the storage of all three data points. Hence many zeros data points were stored because of 2nd difference asymmetry which would not have occurred had the 12 bit A/D data been truncated instead of rounded.

This fact undoubtably affected the overall compression efficiency of the Tolan algorithm. Nonetheless, the distribution in Figure 31 would not have changed (except the zero value count) significantly and noise would still have broadened the 2nd difference distribution.

Tolan and Dower Performance Comparison

As has been reiterated several times in the text of this thesis, the author's original intention was to implement both the Dower and the Tolan algorithms for experimental test and comparison. Since time did not permit the Dower implementation, an experimental comparison was not possible.

As an attempt to compare the results of the Dower and Tolan compression routines, the results quoted from the papers by Dower, Berghofer, and Stewart (Ref 12,29) will be used. Dower states that his state space variable length encoder approaches the entropy bound of the second difference source ''with about 1.65% wastage'' (Ref 12:3). This value is significaltly higher than the approximately 30% ''wastage'' observed with the Tolan variable length encoder.

Assuming similar operations of the second difference time compression (in this author's ''noisy'' enviroment), the Dower decorrelator is expected to have neqative compression. With the Tolan and Dower decorrelators assumed ''equal'', then the real gain of the Dower technique over the Tolan approach is in the VLC.

Assuming the Dower entropy encoder approached closely to the entropy bound, it can be extrapolated that the Dower algorithm would have achieved a maximum compression of approximately (1.2)*(3.22)=3.86:1 for the ''best'' EKG in Table II and Figure 30. The worst compression ratio would have been (.84)*(2.1)=1.76*1 for the worst (noisest) EKG.

Chapter 5 Summary

This chapter began with a description of the EKG collection experiment. Although the EKG apparatus was limited to one channel, sucessful collection and compression of EKG data was performed. Analysis of the data revealed that time compression is inefficient in a ''noisy'' enviroment and that ''rounding'' of the 12 bit A/D samples in conjunction with placing all three sample lead in common accentuated the degradation caused by the time compression data decorrelator. Nonetheless, the Tolan algorithm did achieve an overall positive data compression figure but significantly lower than the 9:1 ratio achieved by Ruttiman and Pipberger (Ref 28) or the average value of 7.3:1 reported by Stewart,Berghofer, and Dower (Ref 29). Finally

it was heuristically shown that if the Dower compression algorithm lived up to the statements by Dower, then a compression ratio gain of 3.86:1 to 2.26:1 could have been achieved with the Dower EKG data compression technique. This thesis will now proceed to provide conclusions and recommendations

VI. Summary, Conclusions, and Recommendations

Summary and Conclusions

This thesis has investigated the field of electrocardiogram data compression with the objective of evaluating compression algorithms on a 6800 microprocessor based computer system. Accomplishment of this goal required the construction of the EKG-Data Acquisition and Analysis System utilizing the Motorola Exorciser microcomputer.

To determine those EKG compression algorithms which had potential for Exorciser implementation, a literature search was made to locate EKG data compression techniques. In addition to the literature search, personal correspondence (Ref 11,31) yielded several EKG compression algorithms. The results of this research is presented in chapter 2.

Since thesis requirements dictated the need for an online, real time data compression algorithm, only the fastest EKG compressors could be considered. Two routines were selected for detailed analysis and inplementation. These two compression algorithms (Tolan and Dower) were discussed at length in chapter 3.

It was deduced from the methodology of the two data compression techniques that the Dower algorithm would perform better than the Tolan procedure. To test this

hypothesis, the EKG-DAAS was constructed and the Tolan algorithm implemented. The implementation of the EKG-DAAS is documented in chapter 4.

Time constraints prohibited completion of the Dower compression algorithm, but data was sucessfully compressed, analyzed, and decompressed with the Tolan algorithm. The results of this analysis are presented in chapter 5.

The conclusions of this research effort are as follows. First, EKG data compression can be accomplished in real time by a microprocessor based computer system. The Exorciser is a slow microcomputer (1 MHz cycle time) yet it was still possible to implement the Tolan algorithm with a 500 Hz sample rate. The second conclusion is that signal noise can dramatically affect the efficiency of the EKG routines in the same class as the Tolan algorithm. The expense and implementation difficulties of prefiltering, low electromagnetic noise environment, and proper EKG lead attachment are well worth the gain in compression achieved. Finally, a software project of this magnitude should not be attempted totally in assembly language. Although assembly language offers the greatest flexibility and speed, algorithm implementation and debugging efforts are enormous. A high order language would have allowed this author to complete his original thesis objectives.

Recommendations

Several recommendations are offered for further study in microcomputer based EKG data compression. First, implementation of the EKG compression algorithms using a 16 bit microprocessor (e.g. 6 MHz Intel 8086) would permit an order of magnitude improvement in speed of execution. Hardware multiply and divide along with 16 bit arithmetic registers would permit easy implementation of the Dower algorithm and would even make use of the Transform compressors (i.e. FFT) feasible. Second, futher study is needed on determination of decorrelator inefficiency on the overall data compression. A large study of different decorrelators such as 1, 2, 3 difference operations with and without time compression is needed. Next, an EKG compression algorithm implemented using the Ruttiman and Pipberger 2nd order interpolator (Ref 28) for the decorrelator along with the Dower variable length encoder (Ref 12) should be built. This combination should prove to be very efficient. Finally, programming and experimental testing should be done on a full scale microcomputer development system, complete with a high order language, A/D-D/A capability, and flexible disk file manipulation software. Such a system is the Zilog MCZ1/25 microcomputer resident here at A.F.I.T.

Bibliography

1. Ahmed, Nasir et al. ''Electrocardiographic Data Compression Via Orthogonal Transforms,'' IEEE Transactions on Biomedical Engineering, BME-22, (6) : 484-487 (November 1975).

2. Ahmed,N and Rao,K.R. Orthogonal Transforms for Digital Signal Processing. New York: Springer-Verlag, 1975.

3. American Heart Association, Rep. Comm. ECG. ''Recommendations for Standardization of Leads and Specifications for Instruments in Electrocardiography and Vectorcardiography,'' Circulation, 54: 11-31 (1975).

4. Berger, Toby. Rate Distortion Theory: A Mathematical Basis For Data Compression. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1971.

5. Blasbalg, H. and Van Blerkom, R. ''Message Compression,'' IRE Transaction on Space Electronics and Telemetry, 8 : 228-238 (September 1962).

6. Box, E and Jenkins, G.W. Time Series Analysis: Forecasting and Control. San Francisco, California : Holden-Day Inc, 1976.

7. Cox, J.R. and Ripley, K.L. ''Compact Digital Coding of Electrocardiographic Data,'' Proceedings Sixth Hawaii International Conference on System Sciences. 333-336. Honolulu, Hawaii : University of Hawaii, January 9-11, 1973.

8. DATEL SYSTEMS,Inc. Sinetrac Series Model ST-6800 A/D-D/A Peripheral Systems Instruction Manual, Part No. 58-12140-25, Mansfield, Mass. January, 1979.

9. Davenport, W.B. Probability and Random Processes: An Introduction For Applied Scientists and Engineers. New York : McGraw-Hill Book Company, 1970.

10. Davisson, Lee D. and Gray, Robert M., editors. Benchmark Papers In Electrical Engineering and Computer Science, Volume 14 : Data Compression. Stroudsburg, Pennsylvania : Dowden, Hutchinson,and Rose, Inc., 1976.

11. Diver, Tom. Project Engineer. Personal Correspondence. Marquette Electronics, Inc.,Milwaukee, Wisconsin, 15 May 1980.

12. Dower,RG. with appendix by Berghofer,D. ''Optimal ECG Data Compression,'' working paper submitted for publication in Proceedings of the Engineering Foundation Conferences: Computerized Interpretation of the ECG V. Asilomar, California. April 27-March 2, 1980.

13. Dubin,Dale, M.D. Rapid Interpretation of EKG's (Second Edition). Tampa, Florida: COVER Publishing Company, 1970.

14. Frank, Ernest. ''An Accurate, Clinical Practical System For Spatial Vectorcardiography,'' Circulation, 13 : 737-749 (May 1956).

15. Gray, R.M. and Davisson, L.D. ''A Mathematical Theory of Data Compression?'', Benchmark Papers In Electrical Engineering and Computer Science, Volume 14 : Data Compression, edited by L.D. Davisson and R.M. Gray. 21-25. Stroudsburg, Pennsylvania : Dowden, Hutchinson, and Rose, Inc., 1976.

16. Huffman, David A. ''A Method For The Construction of Minimum Redundancy Codes,'' Proceedings of the IRE, 40 : 1098-1101 (September 1952).

17. Jelinek,Frederick. ''Buffer Overflow in Variable Length Coding of Fixed Rate Sources,'' IEEE Transactions on Information Theory, IT-14 (3) : 490-501 (May 1968).

18. Makhoul, John. ''Linear Prediction: A Tutorial Review,'' Proceedings of the IEEE, 63 (4) : 561-580 (April 1975).

19. McEliece, Robert J. The Theory of Information and Coding : Encyclopedia of Mathematics and its Applications, Volume 3. Gian-Carlo Rota, editor. Reading, Massachusetts : Addison-Wesley Publishing Company, 1977.

20. McFee,R. and Baule,G.M. ''Reasearch in Electrocardiography and Magnetocardiography,'' Proceedings of the IEEE , 60 (3) :290-303 (March 1972).

21. McGibbon, C.I. ''Data Communication Networks of Interest to Medical Information Processing,'' Proceedings of the IFIP TC 4 Working Conference on Optimization of Computer ECG Processing, edited by H.K Wolf and P.W. MacFarlane. 253-254. Halifax, Nova Scotia, Canada. June 5-7, 1979.

22. Motorola Inc., M6800 EXORciser Users Guide, Phoenix,Arizona: Second Edition, 1975.

23. Midwest Scientific Instruments, Operating and Assembly Manual, The FD-8 Floppy Disk Memory System, Olathe, Kansas.

24. Oppenheim, Alan V. and Shafer, Ronald W. Digital Signal Processing. Englewood Cliffs, N.J. : Prentice-Hall,Inc. 1975.

25. Papoulis, Athanasios. Probability, Random Variables, and Stochastic Processes. New York: McGraw Hill Book Company, 1965.

26. Rautaharju,P.M., Warren, J., and Wolf, H. ''Waveform Vector Analysis of Orthogonal Electrocardiograms: Quantification and Data Reduction, Journal of Electrocardiography, 6: 103-111 (March 1973).

27. Ristenbatt, Marlin P. ''Alternatives in Digital Communications,'' Proceedings of the IEEE, 61 (6):703-721 (June 1973).

28. Ruttimann, Urs E. and Pipberger, Hubert V. ''Compression of the ECG by Prediction or Interpolation and Entropy Encoding,'' IEEE Transactions on Biomedical Engineering, BME-26, 11 : 613-622 (November 1979).

29. Stewart,D.,Berghofer,D. with appendix by Dower,RG. ''Data Compression of ECG Signals,'' Proceedings of the Engineering Foundation Conferences, Computerized Interpretation of ECG IV. Asilomar, California. January, 1979.

30. Shannon, Claude E. ''A Mathematical Theory of Communication,'' Bell Systems Technical Journal, 27 :379-423 (July 1948).

31. Tolan, Gil. (MD),Lt.Col.USAFMC,Personal Correspondence, USAF School of Aeorspace Medicine, Brooks AFB, Texas. April,1980.

32. Tompkins, W. and Webster, J.G., editors. Design of Microcomputer-Based Medical Instrumentation. Englewood Cliffs, N.J.: Prentice-Hall, Inc.,1980.

33. TRW Systems Group, Houston, Texas. ''Investigation of Data Compression Techniques,'' Final Report, NASA-CR-115177. September, 1971, N71-35323.

34. Viterbi, Andrew and Omura, Jim. Principles of Digital Communication and Coding. New York : McGraw Hill Book Company, 1979.

35. Womble, Edward M. et al. ''Data Compression For Storing and Transmitting ECG/VCG's,'' Proceedings of the IEEE, 65 (5) : 702-706 (May 1977).

36. Ziemer, R.E. and Tranter,W.H. Principles of Communcations: Systems, Modulation, and Noise. Boston, Massachusetts : Houghton Mifflin Company, 1976.

Appendix A

The Electrocardiogram

Introduction

The electrocardiogram (or EKG) is a record of the electrical activity of the heart as measured from the body surface. The magnitude, shape, and timing of the electrical potentials generated by the heart reveal a great deal of information concerning the health of the cardiac system. This appendix will describe how the electrical signals from the heart are generated, how the cardiac cycle is coordinated and controlled, and finally how the EKG can be used as a diagnostic tool.

The Physiology and Electricial Characteristics of the Heart

The heart (Fig. A1) is an organ about the size of a fist with four main pumping chambers and a specialized electrical conduction system. At the top are two thin walled pumps called the atrium which prime the main pumps of the heart, the ventricles. The ventricles are separated by a thick wall of muscle tissue called the septum. Blood from the right ventricle goes to the lungs and blood from the left ventricle goes to the rest of the body.

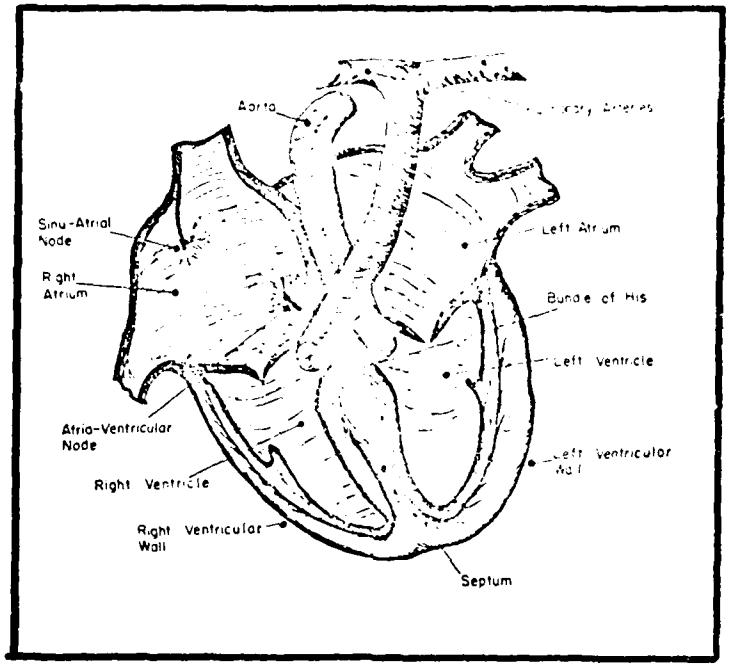


Fig A1. The Heart Cross-Section (From Ref 20:291).

During the resting state between contractions, the cells of the heart are ''polarized'' with an electrical potential existing between the inside (-) and the outside (+) of the cell. This potential is generated by an ionic gradient across the cellular membrane and is normally maintained for approximately .2 to .4 of a second before spontaneous ''depolarization'' occurs. Depolarization (caused by an inrush of sodiun ions into the cell) induces the cell to contract for approximately 1/4 of a second. Because all heart cells contain specialized conducting fibers, the depolarization of one cell initiates the depolarization of neighboring cells and a ''wave of excitation'' sweeps across the myocardium (heart) at a rate

of about 1 meter per second.

The voltages measured at the body surface are the superposition of thousands of heart cells depolarizing (or polarizing) as the wave of excitement flows through the myocardium. Early work by Wilson (Ref 20:292) showed that the heart could be represented by an equivalent electrical dipole whose vector orientation sweeps through a closed loop during one cardiac cycle. In simple terms, as the wave of excitement flows toward a positive skin electrode, a positive slope is generated on the EKG record representing the projection of the heart vector onto the axis of the EKG lead.

To observe this sweeping dipole vector, Wilson developed the 12 lead EKG system in almost univerisal use today. This system (Fig. A2) attempts to measure the heart vector from a variety of vantage points in hope of determing the actual direction of propagation of the wave of excitement. Unfortunately, the leads of the Wilson system are not orthogoal and reconstruction of the actual heart vector orientation and amplitude is difficult. To overcome this problem, Frank (Ref 14:737-749) developed the vector cardiogram which combines 7 leads in a summing network to produce three orthogonal components of the heart vector. Frank VCG systems are popular in heart diagnosis and research because all of the information is contained in three leads of data versus 12.

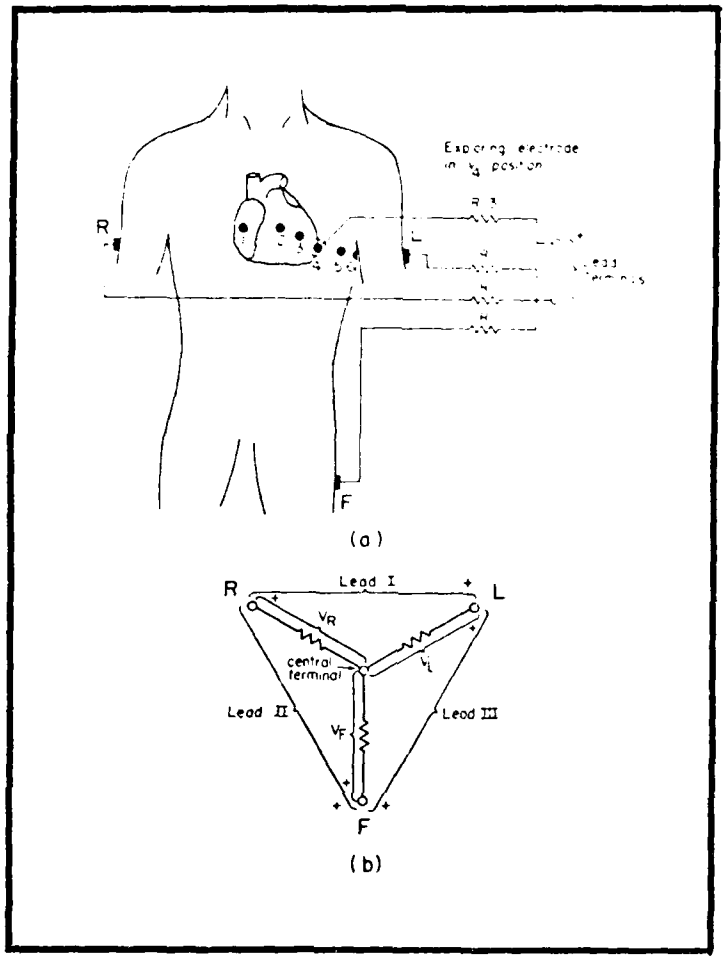


Fig A2. Wilson EKG Electrode System (From Ref 20)

The Cardiac Cycle

For effective pumping action, the heart muscle must contract in a controlled, coordinated way. This means that the stimulating wave of excitation must follow a well defined conduction path to allow the heart to contract in the most efficient manner. In ''normal'' hearts, the contraction sequence begins in the right atrium where

special cells in an area known as the Sino-Atrium (SA) node spontaneously depolarize faster than the rest of the heart tissue. The SA node hence initiates a wave of excitation which covers the atrium in about 80 milliseconds. This atrial contraction produces the electrical signal called the P wave (Fig. A3) on a typical EKG record.

The wave of excitation started by the SA node then reaches another specialized receptor known as the Atrial-Ventricular (AV) node. Here connecting fibers delay the excitation impulse for about 50 milliseconds to allow the ventricles to fill with blood. After the 50 millisecond delay, the excitation signal is relayed to special conducting fibers in the septum known as the Bundle of His. These conducting fibers rapidily (30 ms) transmit the excitation wave to the interior (endocardium) wall of the ventricles where the wavefront propagates radially to the outer wall (epicardium) in another 30 milliseconds. The ventricular wave of excitation produces the QRS waveform complex seen on the EKG.

Following the ventricular contraction, the muscle cells of the ventricles repolarize over a period of 100 milliseconds. No muscular action is occuring but the EKG responds to this electrical activity and the T wave is noted on the EKG record. Atrial repolarization occurs during the QRS hence it is generally invisible on the EKG.

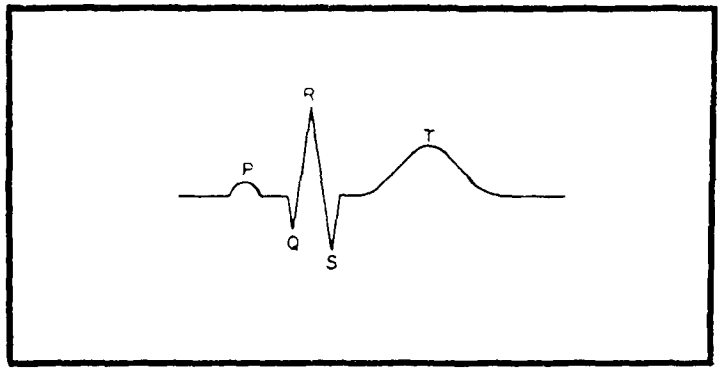


Fig A3. Typical EKG Waveform (From Ref 20)

Finally the heart rests for approximately .2 of a second and the cycle starts again. The above cardiac cycle is typical of a healthy heart. Disease, however, can effect this sequence dramatically.

Heart Disease and the EKG

The variety of ailments which plauge the human heart are numerous and no attempt will made to describe the spectrum of diseases possible. There are, however, several common heart defects which routine EKG analysis usually detects. These include premature contractions (Atrial and Ventricle), bundle branch blocks, hypertrophy, and infarction.

Premature Contractions. Premature contractions are caused by the spontaneous depolarization of heart tissue outside of the SA node. This depolarization initiates a wave of excitement causing the atrium, or more notably, the

ventricles to contract out of rhythm with their normal cycle. Points where this ''unscheduled'' depolarization occurs are called ectopic foci and can arise because of infarct damage (to be disscussed later),coronary heart disease producing oxygen starvation and a number of other causes. Premature Ventricular Contractions (PVC's) are highly visible on the EKG record. Normally the ventricles contract simultaneously and the voltage vectors generated tend to cancel keeping the QRS amplitude relatively small. A PVC, however, causes depolarization of one ventricle before the other generating an unbalanced, hence larger, voltage output.

Bundle Branch Blocks. A bundle branch block is caused by a block of the impulse of the right or left Bundle Branch. This causes a delay in the transmission of the stimulation impulse to ventricle blocked and forces the two ventricles to contract at slightly different times. This difference in ventrical contraction time shows up on the EKG as a double humped QRS.

Hypertrophy. Hypertrophy is an enlargment of one section of the heart muscle tissue. This enlargment affects the duration and strength of the wave of excitement and is visible on the EKG record as a diphasic trace if atrial hypertrophy is present. If ventricular hypertrophy exists, the QRS amplitudes are much larger than normal due to the fact that more tissue is depolarizing.

Infarction. Myocardial infarction is an injury to the heart tissue caused by an occlusion of a coronary artery. An area of the heart is then without a blood supply and often permanent damage occurs. This heart disease is often the one most commonly called a ''heart attack'' and, as is well known, is many times fatal. If a person survives the orginal ''attack'', then this permanent damage shows up in the EKG as a change in the QRS and T waves. This change occurs because the infarcted tissue no longer responds to the excitation wave and the wave moves around, not through, the damaged tissue.

Summary.

This appendix has briefly examined the physiology of the heart and discussed how the myocardial tissue generates the electrical fields measured by the electrocardiogram. Though EKG analysis has been practiced for fifty years, intense research continues in improving EKG diagnosis. Dramatic improvement in computer aided EKG analysis and better understanding of the electro-physiology of the heart is leading to improved cardiac health care worldwide.

The heart diseases discussed above are only a small subset of the problems which can afflict the human heart. Should the reader desire a more thourgh background on heart physiology and cardiac disease, Dubin's book (Ref 13) is highly recommended. This programmed text carefully leads the reader through EKG analysis and is easily read. For a

more firm background on the electro-physics of the heart, the article by McFee and Baule (Ref. 20) provides a good tutorial review on EKG history and research.

Appendix B

Fundamentals of Information Theory

Introduction

In 1948, Claude Shannon published a classic paper (Ref 30:379-423) titled ''A Mathematical Theory of Communication'' in which he laid the foundation of modern information theory. Shannon used his ''information theory'' to describe, mathematically, the interrelationships between the components of a ''typical'' communication system as illustrated in Figure B-1.

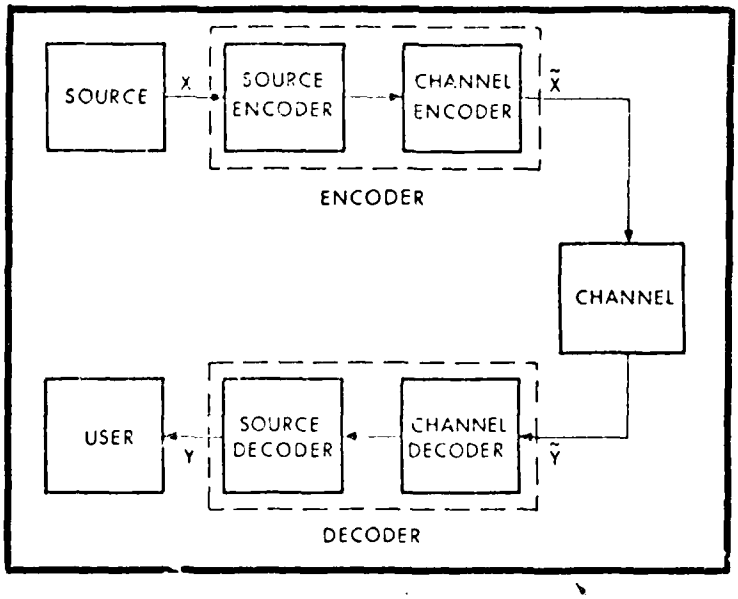


Fig B-1. The communication system model (From Ref 4).

This appendix is written as a basic tutorial on information theory, and is intended to acquaint the reader

with the terminology used in describing the data compression techniques in chapters 2 and 3. The appendix begins with a review of the information source, proceeds to a discussion of the transmission channel and Shannon's rate distortion theory, and concludes with a description of the system encoder/decoder.

Information Source

For his initial analysis, Shannon proposed modelling the information source as a discrete stochastic process whose output is governed by known statistics. As pointed out by Davisson and Gray (Ref 10: 2-4), the discrete-time model was commonly used for any or all of the following reasons: (1) digital communication links have become common place; (2) a continuous time process can be modelled as discrete by sampling, orthogonal function expansion, or waveform segmentation; (3) greater simplicity.

The source is characterized by a finite set of possible outcomes known as its alphabet A. The occurrence of a particular alphabet symbol is governed by probabilistic descriptors (i.e. probability density functions) and it is assumed that the source produces only one symbol from the alphabet every T_s seconds. Hence the information source has a symbol rate of $R_s=1/T_s$ symbols per second.

The next question of interest is how much information is conveyed by the occurrence of a given source symbol? If X is a discrete random variable occurring at time t, and x is

an element of A, then the random variable self information can be defined. That is

$$I(x) = -\log p\{X=x\} \qquad (B.1)$$

where $p\{X=x\}$ is the probability that $X=x$.

According to this description, the less probable an event is, the more information is conveyed when it occurs. The base of the logarithm is unspecified, but in this thesis it is assumed to be base 2. Hence the occurrence of symbol x reveals I(x) bits of information.

The amount of information received per observation is of interest, but one would like a measure of the ''uncertainty'' or ''randomness'' of the source. If the stochastic process defining the source is considered stationary (a pretentious assumption but one generally made) then the output of the source is a sequence of random variables with identical probabilistic descriptors. The probabilistic descriptor will be defined as {u} and could represent the moments of the random variable or its probability density function (PDF). If, in addition, the source is considered ergodic, then statistical averages equal time averages and calculation of the set {u} is greatly simplified.

Given that the discrete process is ergodic, or at least stationary, then the source output at any time is described by the random variable X with range $A=\{x(1),x(2),\ldots,x(n)\}$. The measure of the ''uncertainty'' or ''randomness'' is

defined as the entropy of the source and is given by the equation

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \qquad (B.2)$$

where p_i is the probability of occurence of the discrete value x_i. If $p_i = 0$, then the term $\log_2 1/p_i$ is defined equal to 0 (i.e. no contribution to the entropy). Should the range A be of infinite extent (i.e. a continuous source), then the above series may not converge nor would p(x) necessarily be defined. In this case, H(X) is defined as positive infinity.

As example (Ref 19:15), let X represent the outcome of a single roll of a fair die. Then A={1,2,3,4,5,6} and $p_i = 1/6$ for each i. Here $H(X) = \sum_{6} 1/6 \log_2 6 = 2.58$ bits.

In the above example, the statistics governing the occurrence of a given outcome were uniform. This represents the ''most random'' case with a resultant maximum of the entropy function. Should the die be ''loaded'', then the predicted outcome is ''less random'' and hence the value of H(X) would be reduced.

The next important component in Shannon's communication model is the transmission channel.

Transmission Channel

The transmission channel is also assumed to be a discrete time, finite alphabet device which accepts and transmitts to the receiver one symbol in a finite alphabet B each T_c seconds. The alphabet B is often binary, hence the dimension of the symbol space (defined as ||B||) is 2 and B={0,1}. The transmission rate of the channel is defined as $R_c=1/T_c$ channel symbols (bits if binary) per second. If the output of the source process is defined as {X}, and the channel is ''noisy'', then the received symblol {Y} might not equal the transmitted symbol {X}. A noiseless channel transmitts symbols with no error in which case {X}={Y}.

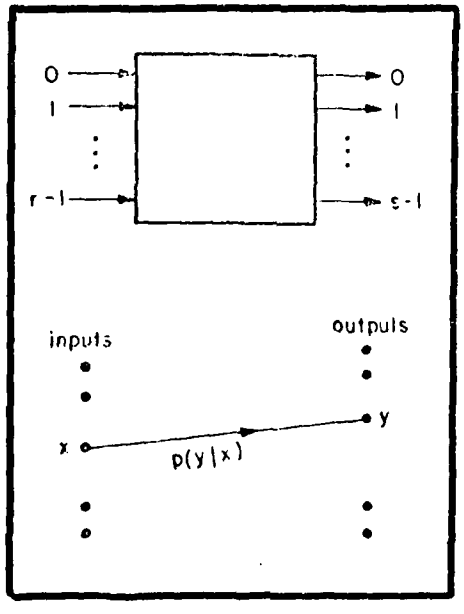


Fig B-2. A discrete memoryless channel (From Ref 19:19)

For simplicity, let the channel be modelled as a

discrete memoryless channel (DMC) as depicted in Figure B-2. The DMC is described mathematically by a conditional probability which relates the chance that a given output y was the result of a given input x. If the input to the channel is a random variable X, and the output is a random variable Y, then a quantity called conditional entropy can be defined. The equation defining conditional entropy is:

$$H(X|Y) = \sum_{x=1}^{n} \sum_{y=1}^{n} p(x,y) \log_2 \frac{1}{p(x|y)} \qquad (B.3)$$

For a given pair X,Y of random variables, H(X|Y) represents the amount of uncertainty remaning about X after Y has been observed.

Now since H(X) represents the uncertainty about X before X is known and H(X|Y) represents the uncertainty after, the difference H(X)-H(X|Y) must represent the amount of information provided about X by Y. This quantity is called the mutual information between X and Y, and is denoted by :

$$I(X;Y) = H(X) - H(X|Y) \qquad (B.4)$$

With the above definitions in hand, the most important quantity of a communications channel can be described; that quantity is the channel capacity. Channel capacity is

defined as the maximum amount of information, per unit of time, which can be ''reliably'' transmitted over the channel. That is :

$$C=\max\{I(X;Y)\} \tag{B.6}$$

Channel capacity is closely related to another important parameter of a communication system known as the rate-distortion function.

Rate-Distortion. Shannon postulated the existence of a mathematical distortion measure, d(X,Y), to measure the distortion or loss resulting if a source symbol X is reproduced as Y. Unfortunately this abstract distortion measure is difficult to quantify. As stated by Berger (Ref 4:6), ''the unavailablity of a distortion measure that is both physically meaningful and analytically tractable constitutes one of the major obstacles to progress in (communication) system design.''

Assuming one has such a distortion measure, then associated with most source-user pairs is a function R(D) called the rate distortion function. The rate distortion is important in that it gives the designer a mathematical tool to measure the amount of distortion that can be expected for a given transmission rate. A communication system can achieve a given fidelity D if and only if the capacity C exceeds R(D). Hence R(D) is the effective rate at which the

source produces information subject to the constraint that the user can tolerate an average distortion of D.

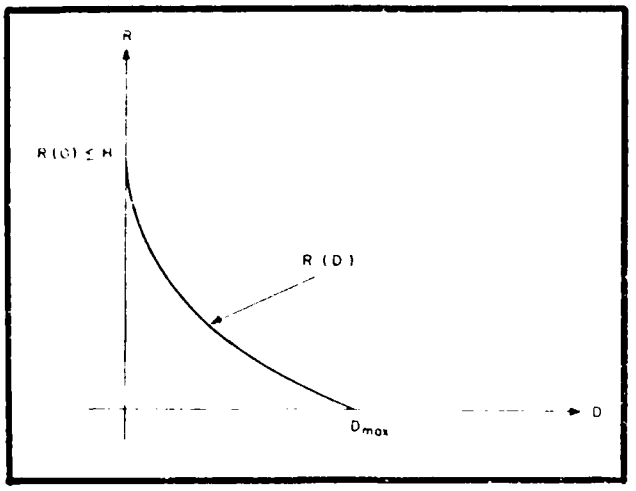


Fig B-3. A typical rate distortion function (From Ref 4:7).

The simplest source-user pair is a discrete memoryless source (DMS) and a single letter fidelity criteria. The DMS produces statistically independent, identically distributed, discrete random variables. Assuming a single letter fidelity criterion, every time the system presents the letter y to the user when the source output was actually x, a nonnegative penalty p(x,y) is determined. The rate-distortion function for the above case is plotted in Figure B-3. As can be seen from the Figure, R(0) is equal to H(X). This last result leads to Shannon's famous channel coding theorem which states that if the symbol rate is less than the channel capacity, it is possible to transmit with perfect fidelity. That is, if the entropy of the source,

H(X), is less than or equal to the channel capacity C, then transmission with zero error can be obtained. This amazing result is not without cost, however. To reduce the source entropy (as seen by the channel) requires coding. *Encoder*

In most circumstances, the output of the information source is not suitable for direct input into the channel. To match the source to the channel, much like matching impedances in circuit theory, is the job of the encoder.

The encoder incoporates all of the functions which process the source data for transmission over the communication channel. This includes coding, analog-to-digital conversion, and modulation. In order to transmit without loss, integers K and L must exist such that KTs=LTc. This guarantees that the received sequence {Y} has the same symbol rate as the transmitted sequence {X}.

As illustrated in Figure B-1, the endoder is divided into two functional subunits. The first of these subunits is the source encoder.

Source Encoder. Source encoding is the operation by which the source output is mapped into an alterate symbol set with the goal of reducing source sequence dependence (i.e. reduce redundancy). Source coding is the transformation in which data compression occurs.

The first source codes, as conceived by Shannon, were block codes where blocks of source symbols were mapped into a single representative channel symbol. This type of

encoding is simple, but efficient only if the source repeats a given, constant length, sequence of symbols on a frequent basis. Another type of source coding is variable length coding.

In variable length codes, those source symbols which occur most frequently are assigned the shortest channel codewords.

One of the most popular variable length codes in use was originally proposed by Huffman (Ref 16:31-34) in the early fifties. This code is in a class known as uniquely decodable (UD) codes. UD codes imply that the codewords, regardless of length, are unique sequences. Hence decoding is instantaneous upon codeword reception(i.e. the decoding does not depend upon reception of the next codeword). McEliece has shown (Ref 19:244-245) that the Huffman code is optimal in the class of UD source codes.

Many techniques for source coding are available. Chapter 2 of this thesis discusses some approaches used in ''compressing'' electrocardiogram waveforms.

The otuput of the source coder is generally a sequence of discrete symbols (i.e. binary 1|0) suitable for futher processing by the channel coder.

Channel Encoder. Shannon proved that if the rate of the source, as seen by the channel, is less than the channel capacity C, then ''noiseless'' or error free transmission is theoretically possible. The goal of channel encoder is to combat channel noise to achieve a probability of error (Pe)

which approaches zero.

Channel encoders allow a reduction of Pe by selectively reinserting redundancy which has been removed by the source coder. This redundancy allows error detection, and with the proper codes, error correction by the decoder. How close the Pe approaches zero is purely a function of the effort (and money!) spent on channel coding.

A common channel coding technique is the use of ''parity'' bits in digital communication. With parity checking, a bit (or bits) is added to the source word which represents the number (odd or even) of ''ones'' in the word. For example, if odd parity is defined and the source word is 1100101 then a single parity bit of 0 implies an even number of ones (1100101|0).

Parity checking in the example above will detect a single bit error in transmission. A more powerful technique is Hamming codes.

The Hamming code is in a large class of codes known as linear codes. Hamming not only detects errors in transmission, but will correct errors to a certain level. This correction is accomplished by multiplying the received codeword by a matrix known as the syndrome. In the case of Hamming codes, the output from this transformation is the bit position in error.

Decoder

As would be expected, the decoder is the inverse

operation which outputs an estimate of the data input to the encoder. The decoder is composed of the channel decoder and the source decoder. The channel decoder uses the redundancy added by the channel encoder to perform error checking and/or correctio . The source decoder takes the ''correct'' data output from the channel decoder and ''decompresses'' the data to produce an estimate of the information source. Information theory has shown that if enough time, complexity, and money is spent on channel encoding, and the information rate is below the channel capacity, then ''error free'' transmission is possible.

Conclusion

This appendix has been a very brief summary of a very large field of study. The key words underlined throughout this text are terminology which appear in the theory chapters (2 and 3) of the thesis. For a more thorough, mathematical treatment of the fields of Information Theory and Coding, the reader is encouraged to refer to the textbooks by McEliece (Ref 19) and Berger (Ref 4). The IEEE Press and Benchmark book (Ref 10) is an execellent source for a survey of the key papers in the field of Data Compression.

Appendix C

This appendix contains the 6800 assembly language source programs of the EKG-DAAS.

```
00030                *:
00040                *
00050                * PROGRAM NAME: EKG-EXEC
00060                * AUTHOR : CAPT. NEIL TOLAN(AFIT)
00070                * VERSION : 1.8
00080                * VERSION DATE 2 OCT 80
00090                *
00100                *
00110                * PROGRAM DESCRIPTION
00120                *
00130                * THIS PROGRAM IS THE EXECUTIVE ROUTINE WHICH
00140                * CONTROLS THE EXORCISER EKG DATA ACQUISITION
00150                * SYSTEM. THIS ROUTINE CALLS OVERLAYED PROGRAMS
00160                * WHICH PERFORM DATA COLLECTION,COMPRESSION,STORAGE
00170                * AND RECONSTRUCTION.
00180                * THIS SOFTWARE IS IN SUPPORT OF THESIS RESEARCH
00190                * TO IDENTIFY THE MOST EFFICIENT EKG DATA COMPRESS-
00200                * ION ALGORITHM IN THE TEST SET.
00210                *
00220                * COMMAND OPTIONS
00230                *
00240                * 0=STORAGE WITHOUT COMPRESSION (10 BIT)
00250                *
00260                * 1=COMPRESS WITH ALGOR TOLAN-A
00270                *
00280                * 2=COMPRESS WITH ALGOR TOLAN-B
00290                *
00300                * 3=COMPRESS WITH ALGOR DOWER
00310                *
00320                * 4=COMPRESS WITH ALGOR 2ND ORDER INTERPOL
00330                *
00340                * 5=COMPRESS WITH ALGOR TURNPT
00350                *
00360                * 6=DISPLAY COLLECTED DATA & STATS
00370                *
00380                * 7=JUMP PROGRAM CONTROL TO DOS
00390                *
00400                * 8=JUMP PROGRAM CONTROL TO EXBUG
00410                *
00420                * 9=LOAD & EXECUTE OTHER OVERLAYS
00430                *
00440                * S=SAVE CUR MEM FILE TO DISK
00450                *
00460                * START OF PROGRAM
00470                *
00480                *
00490 1D00                 ORG     $1D00     PROGRAM START LOCATION
00500                *
00510                      OPT     O         ASSB OPT TO CREATE OBJ FILE
00520                      OPT     NOG       ASSB OPT TO SUP PCC LIST
00530                *
00540                *
00550                * LABLE DECLARATIONS
00560                *
```

```
00570                * SUPPORT SUBROUTINE ADDRESSES
00580                *
                     [illegible]
                     [illegible]
00610  CA36  KEYBD0 EQU  $CA36  EOS. KEYBD INPUT ROUTINE
00620  F000  EXBUG  EQU  $F000  EXBUG. EXBUG ENTRY PT
00630  2800  DOS    EQU  $2800  DOS. DOS ENTRY PT
00640  C75B  WRITE0 EQU  $C75B  EOS. CLR PASSWRD BUFFER
00650  CC87  CLRPAS EQU  $CC87  EOS. CLR PASSWRD BUFFER
00660  C7C8  DRIVE  EQU  $C7C8  EOSIO. DSK DRIVE SELECT
00670  C803  RLIB   EQU  $C803  EOSIO.
00680  C807  WLIB   EQU  $C807  EOSIO.
00690  CE52  LOAD2  EQU  $CE52  EOS. LOAD PRGM ROUTINE
00700  CBDA  DOSTR2 EQU  $CBDA  EOS. ALT EOS ENTRY LOC
00710  E055  BYTE   EQU  $E055  MIKBUG. GET TO HEX DIG FRM TE
00720                *
00730                * DATA BUFFERS
00740                *
00750  3066  NAME   EQU  $3066  NAME BUFFER FOR FILE I/O
00760  3060  TEMPX1 EQU  $3060  TEMP 2 BYTE STOR BUFFER
00770  3062  TEMPX2 EQU  $3062  TEMP 2 BYTE STOR BUFFER
00780  3058  STARTX EQU  $3058  LOWEST ADDR USED IN PRGM
00790  305A  ENDX   EQU  $305A  HIGHEST ADDR USED IN PRGM
00800  305C  GOX    EQU  $305C  START EXECUTE ADDR
00810  001B  PROGX  EQU  $001B  DISK I/O ERR VEC ADDR
00820  0008  BMEMH  EQU  $0008
00830  3002  ENDBUF EQU  $3002  BUF WITH ADDR OF LAST CHAR IN
00840  3400  HDRSTR EQU  $3400  START OF MEM FILE HDR SECTOR
00850  3004  BUFFER EQU  $3004  KEYBD INPUT BUFFER START
00860  3057  TYPE   EQU  $3057  TYPE OF FILE FOR I/O
00870  1C96  STKSAV EQU  $1C96  TEMP STACK SAVE BUFFER
00880  1C98  CPRTYP EQU  $1C98  COMPRESSION TYPE BUFFER
00890  1C9A  COUNT  EQU  $1C9A  GENERAL 8 BIT COUNTER
00900  1C9B  NAMPTR EQU  $1C9B  NAME POINTER FOR CONSOLE I/O
00910  1C9D  VECSAV EQU  $1C9D  IRQ VEC SAVE BUFFER
00920  1C9F  OLAYGO EQU  $1C9F  OVERLAY EXECUTE FLAG
00930  1CA0  LGOFLG EQU  $1CA0  PRSTAT LOAD VS EXECUTE FLG
00940  1CA1  FILHLC EQU  $1CA1  FILHDR SUB ADDR PASS BUF
00950  1CA3  SAVELC EQU  $1CA3  SAVEFIL SUB ADDR PASS BUF
00960  1CA5  HXASLC EQU  $1CA5  HEXASC  SUB ADDR PASS BUF
00970  1CA7  HXBUF  EQU  $1CA7  HEXASC PARAMETER BUFFER
00980  1CA9  PDFPLC EQU  $1CA9  PDFPRT SUB ADDR PASS BUF
00990  1CAB  OVRLLC EQU  $1CAB  OVRLAY SUB ADDR PASS BUF
01000  1CAD  OVRBUF EQU  $1CAD  OVRLAY PARAMETER PASS BUFFER
01010  3490  LOCPCT EQU  $3490  TOTL CAL LOOPS DONE IN 1/2 IT
01020  3494  SAMPNO EQU  $3494  NUM OF SAMPLES TAKEN
01030  3496  LPCAL  EQU  $3496  NUM OF CAL LOOPS PER 1 INTR
01040  3497  MAXZ   EQU  $3497  MAX VLU IN CH Z
01050  3498  MAXZLO EQU  $3498  LOC OF MAX VLU IN CH Z
01060  349A  MINZ   EQU  $349A  MIN VLU IN CH Z
01070  349B  MINZLO EQU  $349B  LOC OF MIN VLU IN CH Z
01080  349D  MAXY   EQU  $349D  MAX VLU IN CH Y
01090  349E  MAXYLO EQU  $349E  LOC OF MAX VLU IN CH Y
01100  34A0  MINY   EQU  $34A0  MIN VLU IN CH Y
```

```
01110          34A1     MINYLO  EQU     $34A1    LOC OF MIN VLU IN CH Y
01120          34A3     MAXX    EQU     $34A3    MAX VLU IN CH X
01130          34A4     MAXXLO  EQU     $34A4    LOC OF MAX VLU IN CH X
01140          34A6     MINX    EQU     $34A6    MIN VLU IN CH X
01150          34A7     MINXLO  EQU     $34A7    LOC OF MIN VLU IN CH X
01160          34A9     MEMBIT  EQU     $34A9    NUM OF BITS AVAIL FOR STO
01170          34AC     DTABIT  EQU     $34AC    NUM OF BITS USED TO STO DTA
01180          34B0     XBITS   EQU     $34B0    NUM OF BITS USED TO STO X
01190          34B3     YBITS   EQU     $34B3    NUM OF BITS USED TO STO Y
01200          34B6     ZBITS   EQU     $34B6    NUM OF BITS USED TO STO Z
01210          34B9     TBITS   EQU     $34B9    NUM OF BITS USED TO STO TIME
01220          34BC     ACELCT  EQU     $34BC    # BITS FED TO VAR LEN CODER
01230          34C2     BASSAV  EQU     $34C2    SAVE LOC FOR VLU   $3620 PRM
01240          3460     MAXMIN  EQU     $3460    START OF MAX,MIN ASCII BUFFER
01250          34C3     ENTRPY  EQU     $34C3    START OF ENTROPY TABLE IN ASC
01260          3C00     SECZRO  EQU     $3C00    SEC 0 OF MEM FILE
01270          0019     SAVEX   EQU     $0019    TEMP LOC TO SAVE INDEX REG
01280          3500     XPDF    EQU     $3500    LOC OF X PDF BUFFER
01290          3700     YPDF    EQU     $3700    LOC OF Y PDF BUFFER
01300          3900     ZPDF    EQU     $3900    LOC OF Z PDF BUFFER
01310          3B00     TPDF    EQU     $3B00    LOC OF TIME PDF BUFFER
01320          8000     BUFEND  EQU     $8000    END OF MEM BUF
01330                   *
01340                   * HARDWARE ADDRESSES
01350                   *
01360          E500     DACZRO  EQU     $E500    DAC 0 ADDRESS
01370          E502     DACONE  EQU     $E502    D :THIS ROUTINE IS THE EXECUTIVE
01460                   * CONTROLLER OF THE EKG DATA ACQ SYS
01470                   *
01480                   *
01490 1D00 0F           START   SEI              STOP POSSIBLE INTR ON RESET
01500 1D01 CE 4000              LDX     #$4000
01510 1D04 FF E500              STX     DACZRO   CLR DACS & SET SEL 1 HIGH
01520 1D07 FF E502              STX     DACONE
01530 1D0A CE 214A              LDX     #FILHDR
01540 1D0D FF 1CA1              STX     FILHLC   PUT FILHDR ADDR IN PASS BUFF
01550 1D10 B6 3620              LDA A   $3620
01560 1D13 B7 34C2              STA A   BASSAV   SAVE CURRENT #$3620 IN BUF FO
01570 1D16 CE 208E              LDX     #SAVEPL
01580 1D19 FF 1CA3              STX     SAVELC   PUT SAVEPL ADDR IN PASS BUF
01590 1D1C CE 23DB              LDX     #HXASC
01600 1D1F FF 1CA5              STX     HXASLC   PUT HXASC ADDR IN PASS BUF
01610 1D22 CE 22C2              LDX     #PDFPRT
01620 1D25 FF 1CA9              STX     PDFPLC   PUT PDFPRT ADDR IN PASS BUF
01630 1D28 CE 2067              LDX     #OVRLAY
01640 1D2B FF 1CAB              STX     OVTLLC   PUT OVRLAY ADDR IN PASS BUF
01650 1D2E CE 1D00              LDX     #START
01660 1D31 FF 2801              STX     DOS+1
01670 1D34 7F 1C9F              CLR     OLAYGO   CLR OVERLAY PGM FLAG
01680 1D37 7F 1CA0              CLR     LGOFLG   CLR PRSTAT LOAD FLG
01690 1D3A 4F                   CLR A
01700 1D3B BD C7C8              JSR     DRIVE    ENSURE EXEC TSK TO DRIVE 0
01710 1D3E BD 1E6F              JSR     CMDIN    PROMPT & GET IN CMD
01720 1D41 C1 30                CMP B   #'0      IS CMD ASCII 0?
```

```
01736 1D43 26 09          BNE    EXCM1     NO. CHECK OTHER CMDS
01740 1D45 CE 1DC7        LDX    #NOCPR    LOAD OVERLAY NAME PTR
[illegible]               STX    OVRBUF
01760 [illegible]         JMP    OVRLAY    LOAD & EXECUTE OVRLAY
01770 1D4E C1 31    EXCM1 CMP B  #'1
01780 1D50 26 09          BNE    EXCM2
01790 1D52 CE 1DCF        LDX    #TOLAN1
01800 1D55 FF 1CAD        STX    OVRBUF
01810 1D58 7E 2067        JMP    OVRLAY
01820 1D5B C1 32    EXCM2 CMP B  #'2
01830 1D5D 26 09          BNE    EXCM3
01840 1D5F CE 1DD7        LDX    #TOLAN2
01850 1D62 FF 1CAD        STX    OVRBUF
01860 1D65 7E 2067        JMP    OVRLAY
01870 1D68 C1 33    EXCM3 CMP B  #'3
01880 1D6A 26 09          BNE    EXCM4
01890 1D6C CE 1DDF        LDX    #DOWER
01900 1D6F FF 1CAD        STX    OVRBUF
01910 1D72 7E 2067        JMP    OVRLAY
01920 1D75 C1 34    EXCM4 CMP B  #'4
01930 1D77 26 09          BNE    EXCM5
01940 1D79 CE 1DE6        LDX    #INTER
01950 1D7C FF 1CAD        STX    OVRBUF
01960 1D7F 7E 2067        JMP    OVRLAY
01970 1D82 C1 35    EXCM5 CMP B  #'5
01980 1D84 26 09          BNE    EXCM6
01990 1D86 CE 1DEE        LDX    #TURNPT
02000 1D89 FF 1CAD        STX    OVRBUF
02010 1D8C 7E 2067        JMP    OVRLAY
02020 1D8F C1 36    EXCM6 CMP B  #'6
02030 1D91 26 09          BNE    EXCM7
02040 1D93 CE 1DF6        LDX    #DISPI
02050 1D96 FF 1CAD        STX    OVRBUF
02060 1D99 7E 2067        JMP    OVRLAY
02070 1D9C C1 37    EXCM7 CMP B  #'7
02080 1D9E 26 09          BNE    EXCM8
02090 1DA0 CE 2BCD        LDX    #$2BCD
02100 1DA3 FF 2801        STX    DOS+1
02110 1DA6 7E 2800        JMP    DOS
02120 1DA9 C1 38    EXCM8 CMP B  #'8
02130 1DAB 26 03          BNE    EXCM9
02140 1DAD 7E F000        JMP    EXBUG
02150 1DB0 C1 39    EXCM9 CMP B  #'9
02160 1DB2 26 09          BNE    EXCMS
02170 1DB4 CE 1DFE        LDX    #OVLMSG
02180 1DB7 BD CA87        JSR    OUTPUT
02190 1DBA 7E CBDA        JMP    DOSTR2
02200 1DBD C1 53    EXCMS CMP B  #'S
02210 1DBF 26 03          BNE    EXCEND
02220 1DC1 BD 208E        JSR    SAVEPL
02230 1DC4 7E 1D00  EXCEND JMP   START
02240                     *
02250 1DC7 4E       NOCPR FCC    /NOCPRS* /
02260 1DCF 54       TOLAN1 FCC   /TOLAN-A*/
```

```
02[illegible] 1DD7 54      TOLAN2 FCC    /TOLAN-B*/
02[illegible] 1DDF 44      DOWER  FCC    /DOWER* /
0[illegible]  1E49         INTER  FCC    /INTERPL*/
0[illegible]  1E54         TURNPT FCC    /TURNPT* /
02310 1DF6 44              DISPL  FCC    /DISPLAY*/
02320 1DFE 1A              OVLMSG FCB    $1A
02330 1DFF 45                     FCC    /ENTER "RUN FILNAME" TO LOAD & EXECUT
02340 1E25 4F                     FCC    /OVERLAY "FILENAME"./
02350 1E38 0D0A                   FDB    $0D0A
02360 1E3A 28                     FCC    /(WARNING: OVERLAY MUST FIT BETWEEN /
02370 1E5D 30                     FCC    /0100-1400)/
02380 1E67 0D0A                   FDB    $0D0A,$0D0A,$003F,$2004
02390                      *
02400                      * END EKG-EXEC
02410                      *
02420                      *FUNCTION :CMDIN
02430                      *INPUTS (REG) :NONE
02440                      *OUTPUTS (REG):B
02450                      *CALLS :OUTPUT,KEYBD0
02460                      *DESTROYS :ALL REGS
02470                      *PURPOSE :TO PRINT PROMPT MESSAGE AND READ COMMAND
02480                      * FROM CONSOLE.
02490                      *
02500 1E6F CE 1E7F CMDIN   LDX    #CMDMSG  LOAD OUT PTR
02510 1E72 BD CA8F         JSR    OUTCR
02520 1E75 BD CA36         JSR    KEYBD0
02530 1E78 FE 3002         LDX    ENDBUF   SET PRT TO LAST CHAR ENTERED
02540 1E7B 09              DEX
02550 1E7C E6 00           LDA B  0,X
02560 1E7E 39              RTS
02570                      *
02580 1E7F 1A07    CMDMSG  FDB    $1A07,$0D0A,$0A0A,$0A0A,$0A0A
02590 1E89 45              FCC    /EKG DATA ACQUISITION SYSTEM/
02600 1EA4 0D0A            FDB    $0D0A,$0D0A
02610 1EA8 43              FCC    /COMMAND OPTIONS:/
02620 1EB8 0D0A            FDB    $0D0A,$0D0A
02630 1EBC 44              FCC    /DATA ACQUISITION/
02640 1ECC 0D0A            FDB    $0D0A,$0D0A
02650 1ED0 20              FCC    / 0=STORAGE WITH NO COMPRESSION/
02660 1EEE 0D0A            FDB    $0D0A
02670 1EF0 20              FCC    / 1=COMPRESSION VIA TOLAN ALGORITHM A
02680 1F14 0D0A            FDB    $0D0A
02690 1F16 20              FCC    / 2=COMPRESSION VIA TOLAN ALGORITHM B
02700 1F3A 0D0A            FDB    $0D0A
02710 1F3C 20              FCC    / 3=COMPRESSION VIA DOWER ALGORITHM/
02720 1F5E 0D0A            FDB    $0D0A
02730 1F60 20              FCC    / 4=COMPRESSION VIA 2ND ORDER INTERPL
02740 1F89 0D0A            FDB    $0D0A
02750 1F8B 20              FCC    / 5=COMPRESSION VIA TURNING POINT ALG
02760 1FB1 0D0A            FDB    $0D0A,$0D0A
02770 1FB5 44              FCC    /DATA DISPLAY/
02780 1FC1 0D0A            FDB    $0D0A,$0D0A
02790 1FC5 20              FCC    / 6=PRINT DATA STATISTICS OR /
02800 1FE1 44              FCC    /DISPLAY DATA/
```

```
02810 1FFD 0D0A           FDB    $0D0A,$0D0A
02820 1FF1 50             FCC    /PROGRAM CONTROL/
02830 2000 0D0A           FDB    $0D0A,$0D0A
02840 2004 20             FCC    / 7=RETURN TO DOS/
02850 2014 0D0A           FDB    $0D0A
02860 2016 20             FCC    / 8=RETURN TO EXBUG/
02870 2028 0D0A           FDB    $0D0A
02880 202A 20             FCC    / 9=LOAD & EXECUTE USER ENTERED OVERL
02890 2050 0D0A           FDB    $0D0A,$0D0A
02900 2054 45             FCC    /ENTER COMMAND NOW=/
02910 2066 04             FCB    4
02920                *
02930                * END CMDIN
02940                *
02950                *
02960                *FUNCTION :OVRLAY
02970                *INPUTS (REG) :X
02980                *OUTPUTS (REG) :NONE
02990                *CALLS :CLRNAM,PUTNAM,OVERLAY AT 0,X
03000                *DESTROYS :ALL REGISTERS
03010                *PURPOSE :TO LOAD AND THEN EXECUTE OVERLAYS
03020                * ASSOCIATED WITH EKG-EXEC &
03030                * SPECIFIED BY THE INPUT COMMAND.
03040                *
03050                *
03060                *
03070 2067 BD 20E1 OVRLAY JSR    CLRNAM   CLR NAME BUFFER
03080 206A BD CC87        JSR    CLRPAS   CLEAR PASSWORD BUFFER
03090 206D CE 3066        LDX    #NAME    PUT DOS FILENAME BUF IN NAMPT
03100 2070 FF 1C9B        STX    NAMPTR
03110 2073 FE 1CAD        LDX    OVRBUF   POINT X TO OVRLAY NAME
03120 2076 BD 20EF        JSR    PUTNAM   PUT OVRLAY NAME IN NAME
03130 2079 86 22          LDA A  #$22     DEFINE FILE TYPE
03140 207B B7 3057        STA A  TYPE
03150 207E BD CE52        JSR    LOAD2    LOAD FILE POINTED TO BY STK
03160 2081 86 AA          LDA A  #$AA     SET UP COMPARE FOR OVRLAY FCN
03170 2083 B1 1C9F        CMP A  OLAYGO   IS FLAG TRUE?
03180 2086 27 05          BEQ    OVRRTS   YES. TREAT OVRLAY AS SUBR
03190 2088 8E A049        LDS    #$A049   NO. TREAT OVRLAY AS ABS JUMP
03200 208B 6E 00          JMP    0,X      JUMP TO OVERLAY
03210 208D 39        OVRRTS RTS
03220                *
03230                * END OVRLAY
03240                *
03250                *
03260                *FUNCTION :SAVEFL
03270                *INPUTS (REG) :NONE
03280                *OUTPUTS (REG) :NONE
03290                *CALLS :CLRPAS,DRIVE,WLIB,WRITE0,OUTPUT
03300                *DESTROYS :ALL REGISTERS
03310                *PURPOSE :TO SAVE EKG DATA FILES RESIDING IN
03320                *IN MEMORY LOCATIONS 3A00-7FFF ON DISK. FILE NAME IS
03330                * PICKED OFF OF FILE HEADER AT LOC 3A0C-3A0F.
03340                *
```

```
                      *
2088 8D 51    SAVEFL BSR    CLRNAM    CLR NAME BUFFER
2090 CE 2141         LDX    [illegible] [illegible]
2093 8D 4F           BSR    CLRNA0    [illegible]
2095 B6 3620         LDA A  $3620     GET 1ST LOC FOR SAVE FROM PAS
2098 B7 34C2         STA A  BASSAV    SAVE IN BUFFER
209B CE 2141         LDX    #NAMMSG   SET UP PTR TO PRT NAME
209E FF 1C9B         STX    NAMPTR
20A1 CE 3402         LDX    #HDRSTR+2  POINT X TO FILENM IN DATA
20A4 8D 49           BSR    PUTNAM    PUT FILENAM IN "NAME" BUF
20A6 86 22           LDA A  #$22      SET TYPE OF FILE
20A8 B7 3057         STA A  TYPE
20AB CE 3400         LDX    #HDRSTR   LOAD BUFFER START
20AE FF 3058         STX    STARTX
20B1 CE 7FFF         LDX    #BUFEND-1  LOAD BUFFER END
20B4 FF 305A         STX    ENDX
20B7 CE 2800         LDX    #$2800    LOAD LGO ADDRESS
20BA FF 305C         STX    GOX
20BD DF 1B           STX    PROGX
20BF BD CC87         JSR    CLRPAS    CLR PASSWRD BUFFER
20C2 CE 2119         LDX    #DRVMSG
20C5 BD CA8F         JSR    OUTNCR
20C8 BD E055         JSR    BYTE
20CB BD C7C8         JSR    DRIVE     SELECT DRIVE 01 FOR STORE
20CE BD C807         JSR    WLIB
20D1 DE 08           LDX    ENEMH
20D3 BD C75B         JSR    WRITE0    WRITE BUFFER TO DISK
20D6 4F              CLR A
20D7 BD C7C8         JSR    DRIVE     RESET DRIVE BACK TO 0
20DA CE 2123         LDX    #CATMSG   LOAD OUTPUT POINTER
20DD BD CA87         JSR    OUTPUT
20E0 39              RTS
                      *
20E1 CE 3066  CLRNAM LDX    #NAME     POINT TO NAME BUFFER
20E4 C6 08    CLRNA0 LDA B  #$8       SET UP CHAR COUNT
20E6 86 20           LDA A  #$20      ASCII SPACE
20E8 A7 00    CLRNA1 STA A  0,X       FILL NAME BUF WITH SPACES
20EA 08              INX
20EB 5A              DEC B
20EC 26 FA           BNE    CLRNA1
20EE 39              RTS
                      *
20EF BF 1C96  PUTNAM STS    STKSAV    SAVE STACK POINTER
20F2 7F 1C9A         CLR    COUNT     CLEAR COUNTER
20F5 35              TXS              TRANSFER X TO STACK
20F6 CE 3066         LDX    #NAME     PICK UP DOS [illegible]
20F9 C6 08           LDA B  #8        SET UP NAME [illegible]
20FB 32       PUTNA1 PUL A            PULL CHAR FROM STACK
20FC 81 04           CMP A  #$04      IS IT 4?
20FE 27 15           BEQ    PUTNA2    YES. NAME ENDED
2100 A7 00           STA A  0,X       NO. STORE CHAR IN "[illegible]"
2102 08              INX
2103 FF 3060         STX    TEMPX1    SAVE INDEX REG
2106 FE 1C9B         LDX    NAMPTR    LOAD CURRENT [illegible]
```

```
03890 2109 A7 00          STA A  0,X       STORE IN NAMEMS
03900 210B 08             INX              INCR POINTER
03910 210C FF 1C93        STX    NAMPTR
03920 210F FE 3060        LDX    TEMPX1    RESTORE INDEX REG
03930 2112 5A             DEC B
03940 2113 26 F6          BNE    PUTNA1    8 CHARS YET?
03950 2115 FE 1C96 PUTNA2 LDS    STKSAV    YES. RESTORE STACK
03960 2118 39             RTS
03970                *
03980 2119 0D0A    DRVMSG FDB    $0D0A
03990 211B 44              FCC    /DRIVE? /
04000 2122 04              FCB    4
04010 2123 1A07    CATMSG FDB    $1A07,$0D0A,$0D0A
04020 2129 44              FCC    /DATA SAVED ON DISK FILE /
04030 2141 0008    NAMMSG RMB    8
04040 2149 04              FCB    4
04050                *
04060                * END SAVEFL
04070                *
04080                *
04090                *FUNCTION :FILHDR
04100                *INPUTS (REG) :NONE
04110                *OUTPUTS (REG) :NONE
04120                *CALLS :OUTPUT,KEYBD0
04130                *DESTROYS :ALL REGISTERS
04140                *PURPOSE :TO INIT DATA BUFFER WITH ACCOUNT
04150                * DATA (NAME,DATE,TIME,ETC) AND CLEAR FOR NEW
04160                * DATA STORAGE.
04170                *
04180 214A BD 21D7 FILHDR JSR    CLRBUF    FILL DATA BUF WITH NULS
04190 214D BD 21C9        JSR    CLRSEC    FILL MEM FILE SEC 3400 WITH A
04200 2150 BD 2288        JSR    PDFCLR    INITIALIZE COLL PARAM
04210 2153 FE 1C98        LDX    CPRTYP    PICK UP COMPR TYPE
04220 2156 FF 3400        STX    HDRSTR    STR IN HEADER SECTOR
04230 2159 CE 21E5        LDX    #NAMEMS   OUT PTR
04240 215C BD CA8F        JSR    OUTNCR    "NAME FILE PLEASE?"
04250 215F BD CA36        JSR    KEYBD0
04260 2162 C6 02          LDA B  #$02      HEADER STOR OFFSET
04270 2164 8D 42          BSR    TXTSTR    STORE FILENAME IN HEADER SEC
04280 2166 37             PSH B            SAVE NEXT AVAIL HEAD LOC
04290 2167 CE 2201        LDX    #SUBJMS
04300 216A BD CA8F        JSR    OUTNCR    "TEST SUBJECT ID?"
04310 216D BD CA36        JSR    KEYBD0
04320 2170 33             PUL B            RETRIEVE STOR OFFSET
04330 2171 8D 35          BSR    TXTSTR
04340 2173 37             PSH B
04350 2174 CE 221F        LDX    #RATEMS
04360 2177 BD CA8F        JSR    OUTNCR    "COLLECTION SAMPLING RATE?"
04370 217A BD CA36        JSR    KEYBD0
04380 217D 33             PUL B
04390 217E 8D 28          BSR    TXTSTR
04400 2180 37             PSH B
04410 2181 CE 2247        LDX    #DATEMS
04420 2184 BD CA8F        JSR    OUTNCR    "DATE (05 JUL 80)?"
```

```
04430 2187 BD CA36          JSR    KEYBD0
04440 218A 33               PUL B
04450 218B 8D 1B            BSR    TXTSTR
04460 218D 37               PSH B
04470 218E CE 225E          LDX    #TIMEMS
04480 2191 BD CA8F          JSR    OUTNCR   "TIME (1420)?"
04490 2194 BD CA36          JSR    KEYBD0
04500 2197 33               PUL B
04510 2198 8D 0E            BSR    TXTSTR
04520 219A 37               PSH B
04530 219B CE 2270          LDX    #CONTMS
04540 219E BD CA8F          JSR    OUTNCR   "CONTENTS (MAX 80)?"
04550 21A1 BD CA36          JSR    KEYBD0
04560 21A4 33               PUL B
04570 21A5 8D 01            BSR    TXTSTR
04580 21A7 39               RTS
04590                *
04600 21A8 BF 1C96 TXTSTR   STS    STKSAV   SAVE STACK
04610 21AB 8E 3003          LDS    #BUFFER-1  POINT TO INPUT BUFFER
04620 21AE CE 3400          LDX    #HDRSTR  POINT TO HEADER
04630 21B1 4F               CLR A           SET A TO ZERO
04640 21B2 4C       TXTST1  INC A
04650 21B3 08               INX
04660 21B4 11               CBA             IS A = OFFSET?
04670 21B5 2D FB            BLT    TXTST1   NO. KEEP INCREMENTING
04680 21B7 32       TXTST2  PUL A           YES. GET CHAR FROM STACK
04690 21B8 81 04            CMP A  #$04     IS IT 4
04700 21BA 27 06            BEQ    TXTRTN   YES. END TEXT ,RETURN
04710 21BC A7 00            STA A  0,X      NO. STORE CHAR IN HEADER
04720 21BE 5C               INC B
04730 21BF 08               INX
04740 21C0 20 F5            BRA    TXTST2
04750 21C2 A7 00    TXTRTN  STA A  0,X
04760 21C4 5C               INC B
04770 21C5 BE 1C96          LDS    STKSAV   RESTORE STACK
04780 21C8 39               RTS
04790 21C9 CE 3400 CLRSEC   LDX    #HDRSTR  POINT TO DATA BUF START
04800 21CC 86 20            LDA A  #$20     ASCII SPACE
04810 21CE A7 00    CLRSE1  STA A  0,X      STORE SPACE
04820 21D0 08               INX
04830 21D1 8C 3500          CPX    #XPDF    IS SECTOR SPACED?
04840 21D4 26 F8            BNE    CLRSE1   COUNT > ZRO?
04850 21D6 39               RTS             NO. RTN
04860                *
04870 21D7 CE 3400 CLRBUF   LDX    #HDRSTR  POINT TO DATA BUF
04880 21DA 86 00            LDA A  #00      ASCII NULL
04890 21DC A7 00    CLRBU1  STA A  0,X      STORE NULL
04900 21DE 08               INX
04910 21DF 8C 8000          CPX    #BUFEND  IS INDEX REG AT END?
04920 21E2 26 F8            BNE    CLRBU1   NO. KEEP GOING
04930 21E4 39               RTS             YES. RETURN
04940                *
04950 21E5 0D0A     NAMEMS  FDB    $0D0A,$0D0A
04960 21E9 4E               FCC    /NAME DATA FILE PLEASE? /
```

```
04970 2200 04            FCB    4
04980 2201 0D0A    SURVMS FDB    $0D0A,$0D0A
04990 2203 [illegible]
05000 221E 04            FCB    4
05010 221F 0D0A    RATEMS FDB    $0D0A,$0D0A
05020 2223 43            FCC    /COLLECTION SAMPLING RATE (500 HZ)? /
05030 2246 04            FCB    4
05040 2247 0D0A    DATEMS FDB    $0D0A,$0D0A
05050 224B 44            FCC    /DATE (05 JUL 80)? /
05060 225D 04            FCB    4
05070 225E 0D0A    TIMEMS FDB    $0D0A,$0D0A
05080 2262 54            FCC    /TIME (1423)? /
05090 226F 04            FCB    4
05100 2270 0D0A    COMTMS FDB    $0D0A,$0D0A
05110 2274 43            FCC    /COMMENTS (MAX=80)? /
05120 2287 04            FCB    4
05130                    *
05140                    * END FILHDR
05150                    *
05160                    *FUNCTION :PDFCLR
05170                    *INPUTS (REG) :NONE
05180                    *OUTPUTS (REG):NONE
05190                    *CALLS :NOTHING
05200                    *DESTROYS :A,B,X,CC
05210                    *PURPOSE :TO INITIALIZE STATISTIC VAR & CLEAR
05220                    * PDF BUFFERS
05230                    *
05240 2288 4F      PDFCLR CLR A            FILL PARAM BUFF WITH 0
05250 2289 CE 3490       LDX    #LOOPCT
05260 228C A7 00   PDFCL0 STA A  0,X
05270 228E 08            INX
05280 228F 8C 34C3       CPX    #ENTRPY  FILL WITH 0 UP TO ENTROPY BUF
05290 2292 26 F8         BNE    PDFCL0
05300 2294 CE 021F       LDX    #$021F   STORE MAX POSSIBLE BITS IN ME
05310 2297 86 F0         LDA A  #$F0
05320 2299 FF 34A9       STX    MEMBIT
05330 229C B7 34AB       STA A  MEMBIT+2
05340 229F 4F            CLR A
05350 22A0 CE 3500       LDX    #XPDF    NOW CLR PDF BUF AREA
05360 22A3 A7 00   PDFCL1 STA A  0,X
05370 22A5 08            INX
05380 22A6 8C 3C00       CPX    #SECZRO
05390 22A9 26 F8         BNE    PDFCL1
05400 22AB 86 80         LDA A  #$80     NOW SET UP INITIAL MAX & MINS
05410 22AD C6 7F         LDA B  #$7F
05420 22AF B7 3497       STA A  MAXZ
05430 22B2 B7 349D       STA A  MAXY
05440 22B5 B7 34A3       STA A  MAXX
05450 22B8 F7 349A       STA B  MINZ
05460 22BB F7 34A0       STA B  MINY
05470 22BE F7 34A6       STA B  MINX
05480 22C1 39            RTS
05490                  *
05500                  * END PDFCLR
```

```
06050 2327 DE 19          LDX    SAVEX     GET CUR PDF ADDR
06060 2329 A6 00          LDA A  0,X       LDA MSB BYTE OF PDF VLU
06070 232B A6 01          LDA B  1,X       LD LSB
06080 232D CE 23CD        LDX    #PDFMS5
06090 2330 FF 1CA7        STX    HXBUF
06100 2333 BD 23DB        JSR    HXASC     CONV & STO MSB IN ASC STR
06110 2336 17             TBA
06120 2337 CE 23CF        LDX    #PDFMS5+2
06130 233A FF 1CA7        STX    HXBUF
06140 233D BD 23DB        JSR    HXASC     CONV & STORE LSB IN ASC STR
06150 2340 CE 23BB        LDX    #PDFMS3
06160 2343 BD CA87        JSR    OUTPUT    PRINT STRING
06170 2346 32             PUL A            RETRIEV VLU INDEX
06180 2347 DE 19          LDX    SAVEX     RETRIEC CUR PDF ADDR
06190 2349 39             RTS
06200              *
06210 234A 1A0C  PDFMS1   FDB    $1A0C,$0D0A,$0A0A,$0A0A,$0A0A
06220 2354 3A             FCC    /:FILE /
06230 235A 0008  PDFNAM   RMB    8
06240 2362 0D0A           FDB    $0D0A,$0D0A
06250 2366 43             FCC    /CHANNEL /
06260 236E 0001  PDFMS2   RMB    1
06270 236F 20             FCC    / AMPLITUDE DISTRIBUTION/
06280 2386 0D0A           FDB    $0D0A,$0D0A
06290 238A 44             FCC    /DATA VALUE/
06300 2394 20             FCC    /                NUMBER OF OCCURENCES/
06310 23B6 0D0A           FDB    $0D0A,$0D0A
06320 23BA 04             FCB    4
06330 23BB 20    PDFMS3   FCC    /    /
06340 23BF 0002  PDFMS4   RMB    2
06350 23C1 2E             FCC    /............./
06360 23CD 0004  PDFMS5   RMB    4
06370 23D1 20             FCC    / (HEX)/
06380 23D7 04             FCB    4
06390 23D8 0D23  PDFMS6   FDB    $0D23
06400 23DA 04             FCB    4
06410              *
06420              *
06430              * END PDFPRT
06440              *
06450              *FUNCTION :HXASC
06460              *INPUTS (REG) :A,X,DTA FOR CONV,LOC OF ASC STOR
06470              *OUTPUTS(REG) :NONE
06480              *CALLS :NOTHING
06490              *DESTROYS :A,CC
06500              *PURPOSE :TO CONVERT 1 BYTE OF HEX DATA
06510              * INTO 2 BYTES OF ASCII
06520              *
06530              *
06540 23DB FE 1CA7 HXASC  LDX    HXBUF
06550 23DE 36             PSH A            SAVE DTA ON STK
06560 23DF 44             LSR A            SHIFT TOP 4 BITS TO LOW 4
06570 23E0 44             LSR A
06580 23E1 44             LSR A
```

```
05510                *
05520                *
05530                *FUNCTION :PRINT
05540                *INPUTS (REG):NONE
05550                *OUTPUTS (REG):NONE
05560                *CALLS :OUTPUT
05570                *DESTROYS(REG):A,B,X,CC
05580                *PURPOSE :TO PRINT AMPLITUDE DISTRIBUTION
05590                * DATA COMPILED DURING DATA COLLECTION.
05600                *
05610 22C2 CE 235A PDFPRT LDX    #PDFNAM  PUT CUR MEM FILE NAME IN MSG
05620 22C5 FF 1C9B        STX    NAMPTR
05630 22C8 CE 3402        LDX    #HDRSTR+2
05640 22CB BD 20EF        JSR    PUTNAM
05650 22CE CE 234A        LDX    #PDFMS1  LOAD MSG POINTER
05660 22D1 86 58          LDA A  #'X
05670 22D3 B7 236E        STA A  PDFMS2   STOR ASCII X TO MSG
05680 22D6 BD CA87        JSR    OUTPUT   "CHANNEL X AMPLITUDE .."
05690 22D9 CE 3500        LDX    #XPDF    POINT TO X PDF DTA
05700 22DC 86 7F          LDA A  #$7F     INIT COUNTER
05710 22DE 8D 25          BSR    PDFPR1   PRINT X PDF DATA
05720 22E0 CE 234A        LDX    #PDFMS1  STOR ASCII Y TO MSG
05730 22E3 86 59          LDA A  #'Y
05740 22E5 B7 236E        STA A  PDFMS2
05750 22E8 BD CA87        JSR    OUTPUT
05760 22EB CE 3700        LDX    #YPDF    POINT TO Y PDF DTA
05770 22EE 86 7F          LDA A  #$7F
05780 22F0 8D 13          BSR    PDFPR1
05790 22F2 CE 234A        LDX    #PDFMS1
05800 22F5 86 5A          LDA A  #'Z
05810 22F7 B7 236E        STA A  PDFMS2
05820 22FA BD CA87        JSR    OUTPUT
05830 22FD CE 3900        LDX    #ZPDF
05840 2300 86 7F          LDA A  #$7F
05850 2302 8D 01          BSR    PDFPR1
05860 2304 39             RTS
05870                *
05880 2305 4C      PDFPR1 INC A
05890 2306 81 7F          CMP A  #$7F     256 VLUES PRNT'D YET?
05900 2308 27 07          BEQ    PDFPR2   YES. EXIT
05910 230A BD 231B        JSR    PRTVLU   NO. PRINT VLUE TO CONSOLE
05920 230D 08             INX
05930 230E 08             INX
05940 230F 20 F4          BRA    PDFPR1   PRINT NEXT VLU
05950 2311 BD 231B PDFPR2 JSR    PRTVLU
05960 2314 CE 23D8        LDX    #PDFMS6
05970 2317 BD CA8F        JSR    OUTNCR
05980 231A 39             RTS
05990                *
06000 231B 36      PRTVLU PSH A            SAVE VALUE INDEX IN A
06010 231C DF 19          STX    SAVEX    SAVE LOC POINTED TO BY X
06020 231E CE 23BF        LDX    #PDFMS4  PICK UP STRING PTR
06030 2321 FF 1CA7        STX    HXBUF
06040 2324 BD 23DB        JSR    HXASC    CONV VLU INDEX TO ASCII & STR
```

```
06590 23E2 44            LSR A
06600 23E3 8D 08         BSR    HXTOAS   CONV SHIFT'D BITS TO ASC
06610 23E5 A7 00         STA A  0,X      STR IN ALPH STRING
06620 23E7 32            PUL A
06630 23E8 8D 03         BSR    HXTOAS   CONV LOW 4 TO ASC
06640 23EA A7 01         STA A  1,X      STR IN ALPH STRING
06650 23EC 39            RTS
06660                *
06670 23ED 84 0F   HXTOAS AND A  #$0F     CLR TOP 4 BITS
06680 23EF 81 0A         CMP A  #$0A     IS NUM GE HEX A?
06690 23F1 2C 03         BGE    HXTOA1   YES. BRANCH & ADD BIAS OF 37
06700 23F3 8B 30         ADD A  #$30     NO. ADD BIAS OF30 & RET
06710 23F5 39            RTS
06720 23F6 8B 37   HXTOA1 ADD A  #$37
06730 23F8 39            RTS
06740                *
06750                * END HXASC
06760                *
06770                * END OF EKG-EXEC ROUTINES
06780                *
06790                    END
```

```
00030 ********************************************************
00040 *
      *    OVERLAY NAME     :DISPLY
      *    AUTHOR           :CAPT M. SCHMID
00070 *    VERSION           :1.5
00080 *    VERSION DATE      :4 NOV 80
00090 *
00100 ********************************************************
00110 *
00120 *            PROGRAM DESCRIPTION
00130 *
00140 *    THIS PROGRAM DISPLAYS THE STATISTICAL DATA
00150 * CALCULATED DURING THE COLLECTION OF AN EKG
00160 * RECORDING.  THE USER HAS SEVERAL MODES OF DATA
00170 * OUTPUT AND THEY ARE LISTED IN THE COMMAND STRING
00180 * BELOW.  THIS PROGRAM ALSO PRODUCES PROBABILITY
00190 * DENSITY FUNCTION (PDF) PLOTS OF THE DATA COLLECTED
00200 * ON CHANNEL X,Y,OR Z VIA D/A CONVERTERS. THE DATA
00210 * (PDF) IS DISPLAYED ON AND OSCILLISCOPE.
00220 * IN ADDITION, THE COMMAND STRING ALLOWS RECONSTRUCT
00230 * AND DISPLAY OF PRIVIOUSLY COLLECTED DATA ON THE
00240 * OSCILLISCOPE.  THE RECONSTRUCTION ROUTINES ARE
00250 * OVERLAYED OVER THE PRINT STATISTICS (PRSTAT)
00260 * ROUTINES.
00270 *
00280 *   COMMAND OPTIONS
00290 *
00300 *      0= RETURN TO EKG-EXEC
00310 *      1= PRINT CURRENT MEMORY FILE STATISTICS
00320 *      2= PRINT PDF AMPLITUDE TABLES
00330 *      3= DISPLAY XPDF ON THE OSCILLISCOPE
00340 *      4= DISPLAY YPDF ON THE OSCILLISCOPE
00350 *      5= DISPLAY ZPDF ON THE OSCILLISCOPE
00360 *      6= RECONSTRUCT & DISPLAY CH X ON
00370 *         THE OSCILLISCOPE
00380 *      7= RECONSTRUCT & DISPLAY CH Y ON
00390 *         THE OSCILLISCOPE
00400 *      8= RECONSTRUCT & DISPLAY CH Z ON
00410 *         THE OSCILLISCOPE
00420 *      9= LOAD MEMORY FILE FROM DISK
00430 *
00440 ********************************************************
00450 ********************************************************
00460 *
00470 *           START OF DISPL
00480 *
00490 ********************************************************
00500 *
00510 0100          ORG    $0100     PROGRAM ORIGIN
00520 *
00530               OPT    O         ASSB OPT. LIST ASSEMBLY
00540               OPT    NOG       ASSB OPT. SUPPRESS FCC LIST
00550 *
00560 ********************************************************
```

```
00570                *
00580                *          LABEL DECSRIPTIONS
00590                *
00600                * SYSTEM EQU'S
00610                *
00620       CA8F     OUTNCR EQU    $CA8F    EPROMDOS. OUTPUT STRING WITH
00630       CA36     KEYPD0 EQU    $CA36    EPROMDOS. INPUT ALPH STRING
00640       1D00     START  EQU    $1D00    EXG-IXC.  START ADDRESS
00650       CCA0     MAKNAA EQU    $CCA0    EPROMDOS. GET FILE NAME FROM
00660       C803     RLIB   EQU    $C803    EPROMDOS. SET UP READ I/O
00670       2800     DOS    EQU    $2800    DOS. RESTART LOC
00680       C43C     READ0  EQU    $C43C    EPROMDOS. READ DISK ROUTINE
00690       E055     BYTE   EQU    $E055    MIKBUG. INPUT 2 HEX DIG PRT T
00700       C7C8     DRIVE  EQU    $C7C8    EPROMDOSIO. READ DISK ROUTINE
00710       0500     PRSTAT EQU    $0500    PRINT COLLEC STAT ROUTINE
00720                *
00730                * DATA BUFFERS
00740                *
00741       0020     CHNLBF EQU    $0020    CHANNEL DESIGNATOR FOR DECFRS
00750       3400     HDRSTR EQU    $3400    MEM FILE HDR ADDR
00760       3500     XPDF   EQU    $3500    LOC OF CH X PDF BUFFER
00770       3700     YPDF   EQU    $3700    LOC OF CH Y PDF BUFFER
00780       3900     ZPDF   EQU    $3900    LOC OF CH Z PDF BUFFER
00790       FFF8     IRQVEC EQU    $FFF8    LOC OF INT VECTOR ADDRESS
00800       1C9D     VECSAV EQU    $1C9D    TEMP IRQVEC SAVE LOC
00810       3002     ENDBUF EQU    $3002    BUF WITH ADDR OF LAST CHAR IN
00820       1C96     STKSAV EQU    $1C96    TEMP STACK STORAGE BUFFER
00830       0019     SAVEX  EQU    $0019    TEMP INDEX REG STORAGE BUFFER
00840       000A     EMEMH  EQU    $000A    END MEM ADDR FOR DISK I/O
00850       001B     PROGX  EQU    $001B    ERR VEC ADDR
00860       1C9F     OLAYCO EQU    $1C9F    OVRLAY MODE FLAG
00870       3057     TYPE   EQU    $3057    FILE TYPE BUFFER
00880       1CA9     PDFPLC EQU    $1CA9    PDFPRT ADDR PASS BUFFER
00890       1CAB     OVRLLC EQU    $1CAB    OVRLAY ADDR PASS BUFFER
00900       1CAD     OVRBUF EQU    $1CAD    OVRLAY PARAMETER BUFFER
00910                *
00920                * HARDWARE ADDRESSES
00930                *
00940       E500     DACZRO EQU    $E500    D/A CONVERTER ZERO
00950       E400     ADCZRO EQU    $E400    A/D CONVERTER ZERO
00960                *
00970                *
00980                *FUNCTION      :DISPL
00990                *INPUTS (REG)  :NONE
01000                *OUTPUTS(REG)  :NONE
01010                *CALLS         :OUTNCR,KEYPD0
01020                *DESTROYS      :A,B,CC,X
01030                *PURPOSE       :DISPL IS THE COMMAND EXEC [illegible]
01040                *               COMMAND INPUT FROM THE [illegible] AND T
01050                *               DIRECTING EXECUTION OF THE [illegible] O
01060                *               IONS.
01070                *
01080                *
01090 0100 CE 0100 DISPL  LDX    #DISPL
```

```
01100 0103 FF 2801          STX    DOS+1
01110 0106 CE 025B          LDX    #DISPMS
0111? 0109 ?? C?5F          JSR    [illegible]   "[illegible] ...."
01130 010C BD CA36          JSR    [illegible]   [illegible]
01140 010F FE 3002          LDX    ENDBUF   SET UP IN DATA BUF POINTER
01150 0112 09               DEX
01160 0113 E6 00            LDA B  0,X      GET LAST CHAR INPUT FROM TERM
01170 0115 C1 30            CMP B  #'0      WAS IT 0?
01180 0117 26 03            BNE    DISPL1   NO. CHEK IF 1
01190 0119 7E 1D00          JMP    START    YES. RET TO EKG-EXEC
01200 011C C1 31   DISPL1   CMP B  #'1      WAS COMMAND 1?
01210 011E 26 0C            BNE    DISPL2   NO. CHEKC IF 2
01220 0120 CE 01E4          LDX    #PRSNAM  GET "PRSTAT*" TO SEND TO OVRL
01230 0123 FF 1CAD          STX    OVRBUF   STORE "PRSTAT" TO OVRLAY BUFF
01240 0126 7F 1C9F          CLR    OLAYGO   RESTE OLAYGO FLAG
01250 0129 7E 04D6          JMP    OVRLAY
01260 012C C1 32   DISPL2   CMP B  #'2      WAS COMMAND 2?
01270 012E 26 0F            BNE    DISPL3   NO. CHEK IF 3
01280 0130 BD 04DB          JSR    PDFPRT   PRINT PDF AMPLITUDE TABLES
01290 0133 CE 0463          LDX    #DISMS1
01300 0136 BD CA8F          JSR    OUTNCR   "PRESS RETURN"
01310 0139 BD CA36          JSR    KEYBD0
01320 013C 7E 0100          JMP    DISPL
01330 013F C1 33   DISPL3   CMP B  #'3      WAS COMMAND 3?
01340 0141 26 0F            BNE    DISPL4   NO. CHECK 4
01350 0143 CE 3500          LDX    #XPDF    GET ADDRESS OF X PDF DATA
01360 0146 BD 0470          JSR    MAXSCN   SCAN PDF & FIND SHIFT NECESSA
01370               *                       TO ALLOW OUTPUT OF PDF VIA 12
01380 0149 BD 04A0          JSR    PDFTRN   TRANSFER X PDF TO WORK BUFFER
01390 014C BD 04BA          JSR    SCALE    SCALE WORK BUFFER BY SHIFT FR
01400 014F 7E 01F4          JMP    PDFOUT   NOW DISPLAY X PDF TO OSCOPE.
01410 0152 C1 34   DISPL4   CMP B  #'4      WAS COMMAND 4?
01420 0154 26 0F            BNE    DISPL5   NO. CHECK 5
01430 0156 CE 3700          LDX    #YPDF    GET ADDRESS OF Y PDF DATA
01440 0159 BD 0470          JSR    MAXSCN
01450 015C BD 04A0          JSR    PDFTRN
01460 015F BD 04BA          JSR    SCALE
01470 0162 7E 01F4          JMP    PDFOUT
01480 0165 C1 35   DISPL5   CMP B  #'5      WAS COMMAND 5?
01490 0167 26 0F            BNE    DISPL6   NO. RET & PRINT COMMAND PROMP
01500 0169 CE 3900          LDX    #ZPDF
01510 016C BD 0470          JSR    MAXSCN
01520 016F BD 04A0          JSR    PDFTRN
01530 0172 BD 04BA          JSR    SCALE
01540 0175 7E 01F4          JMP    PDFOUT
01550 0178 C1 36   DISPL6   CMP B  #'6
01560 017A 26 0F            BNE    DISPL7
01561 017C 7F 0020          CLR    CHNLBF   SET CHNL DESIGNATOR FOR CH X
01570 017F 7F 1C9F          CLR    OLAYGO
01580 0182 CE 01EC          LDX    #DECPRS  LOAD & RUN DECPRS ROUTINE
01590 0185 FF 1CAD          STX    OVRBUF   STORE "DECPRS" TO OVRLAY BUFF
01600 0188 7E 04D6          JMP    OVRLAY
01610 018B C1 37   DISPL7   CMP B  #'7
01620 018D 26 10            BNE    DISPL8
```

```
01621 018F 86 01          LDA A  #1       SET CHNL DESIGNATOR FOR CH Y
01622 0191 97 20          STA A  CHNLBF
[illegible]
01650 0199 FF 1CAD        STX    OVRBUF   STORE "DECPRS" TO OVRLAY BUFF
01660 019C 7E 04D6        JMP    OVRLAY
01670 019F C1 38   DISPL8 CMP B  #'8
01680 01A1 26 10          BNE    DISPL9
01681 01A3 86 02          LDA A  #2       SET CHNL DESIGNATOR FOR CH Z
01682 01A5 97 20          STA A  CHNLBF
01690 01A7 7F 1C9F        CLR    OLAYCO
01700 01AA CE 01EC        LDX    #DECPRS  LOAD & RUN DECPRS ROUTINE
01710 01AD FF 1CAD        STX    OVRBUF   STORE "DECPRS" TO OVRLAY BUFF
01720 01B0 7E 04D6        JMP    OVRLAY
01730 01B3 C1 39   DISPL9 CMP B  #'9
01740 01B5 26 2A          BNE    CMDERR
01750 01B7 86 22          LDA A  #$22     SET UP FILE TYPE FOR LOAD
01760 01B9 B7 3057        STA A  TYPE
01770 01BC CE 024C        LDX    #NAMSG   "ENTER FILENAME?"
01780 01BF BD CA8F        JSR    OUTNCR
01790 01C2 BD CCA0        JSR    MAKNAA
01800 01C5 BD C803        JSR    RLIB
01810 01C8 CE FFFF        LDX    #$FFFF
01820 01CB DF 0A          STX    EMEMH
01830 01CD CE 2800        LDX    #DOS
01840 01D0 DF 1B          STX    PROGX
01850 01D2 CE 3400        LDX    #HDRSTR  PUT ADDR TO LOAD IN INDEX
01860 01D5 BD C43C        JSR    READ0
01870 01D8 CE 0456        LDX    #FILEMS
01880 01DB BD CA8F        JSR    OUTNCR
01890 01DE BD CA36        JSR    KEYBD0
01900 01E1 7E 0100 CMDERR JMP    DISPL
01910                     *
01920 01E4 50      PRSNAM FCC    /PRSTAT* /
01930 01EC 44      DECPRS FCC    /DECPRS* /
01960                     *
01970 01F4 86 AA   PDFOUT LDA A  #$AA     SET INTERRUPT DONE TEST FLAG
01980 01F6 B7 0248        STA A  DONTST
01990 01F9 0E      PDFOU1 CLI             CLEAR INTERRUPT MASK FOR STOP
02000 01FA 7F 024B        CLR    KOUNT    CLR BUFFER PTR COUNTER
02010 01FD FE FFF8        LDX    IRQVEC   GET CUR IRQ VECTOR
02020 0200 FF 1C9D        STX    VECSAV   SAVE IN TEMP BUFFER
02030 0203 CE 0240        LDX    #STOP    LOAD ADDRESS OF STOPING INTR
02040 0206 FF FFF8        STX    IRQVEC
02050 0209 CE 07FF        LDX    #$07FF   SEND MAX POS PUSLE FOR SCOPE
02060 020C FF E500        STX    DACZRO
02070 020F CE 3200        LDX    #$3200   GET START ADDR OF [illegible]
02080 0212 7C 024B PDFOU2 INC    KOUNT    BUMP UP PTR COUNTER
02090 0215 27 0C          BEQ    PDFOU3   HAS WHOLE BUFFER [illegible]
02100 0217 A6 00          LDA A  0,X      NO. GET MSB BYTE OF [illegible]
02110 0219 E6 01          LDA B  1,X      GET LSB
02120 021B B7 E500        STA A  DACZRO   SEND TO D/A CONV
02130 021E F7 E501        STA B  DACZRO+1 SEND TO D/A CONV
02140 0221 08             INX             INC INDEXED ADDRES
```

```
02150 0222 08              INX
02160 0223 20 FD           BRA     PDFOU2    LOOP AGAIN
[illegible]
02180                      *
02190 0228 27 02           BEQ     PDFOU4    YES. EXIT DISPLAY LOOP
02200 022A 20 CD           BRA     PDFOU1    NO. KEEP SENDING DISPLAY
02210 022C 0F        PDFOU4 SEI
02220 022D CE 4000         LDX     #$4000    RESET INTR GATE.
02230 0230 FF E500         STX     DACZRO
02240 0233 B7 E400         STA A   ADCZRO    RESET ST6800 INT FLIP FLOP
02250 0236 01              NOP
02260 0237 FE 1C9D         LDX     VECSAV    GET ORG INT VEC ADDRESS
02270 023A FF FFF8         STX     IRQVEC    PUT BACK IN INT VEC BUFFER
02280 023D 7E 0100         JMP     DISPL     RET & PRINT COMMAND PROMPT
02290                      *
02300 0240 7F 0248 STOP    CLR     DONTST    CLEAR DONE TEST FLAG
02310 0243 B7 E400         STA A   ADCZRO    CLREA ST6800 IN FLIP FLOP
02320 0246 01              NOP
02330 0247 3B              RTI               RETURN FROM INTERRUPT AND STO
02340 0248 0001    DONTST  RMB     1         INTR DONE TEST FLAG
02350 0249 0001    SHFCNT  RMB     1         BUF WITH NUM OF SHF FOR 12 BI
02360 024A 0001    MAXCNT  RMB     1         TEMP BUF TO COUNT 256 THRU 0
02370 024B 0001    KOUNT   RMB     1         TEMP COUNTER
02380                      *
02390 024C 45      NAMMSG  FCC     /ENTER /
02400 0252 04              FCB     4
02410 0253 44      DRVMSG  FCC     /DRIVE? /
02420 025A 04              FCB     4
02430                      *
02440 025B 1A07    DISPMS  FDB     $1A07,$0C0D,$0A0A,$0A0A,$0A0A,$0A0A
02450 0267 0D0A            FDB     $0D0A
02460 0269 44              FCC     /DATA DISPLAY/
02470 0275 0D0A            FDB     $0D0A,$0D0A
02480 0279 43              FCC     /COMMAND OPTIONS:/
02490 0289 0D0A            FDB     $0D0A,$0D0A
02500 028D 20              FCC     /     0=RETURN TO EKG-EXEC/
02510 02A6 0D0A            FDB     $0D0A,$0D0A
02520 02AA 20              FCC     /     1=PRINT CURRENT MEMORY FILE STA
02530 02D5 0D0A            FDB     $0D0A,$0D0A
02540 02D9 20              FCC     /     2=PRINT PDF AMPLITUDE TABLES/
02550 02FA 0D0A            FDB     $0D0A,$0D0A
02560 02FE 20              FCC     /     3=DISPLAY CH X PDF ON OSCILLISC
02570 0325 0D0A            FDB     $0D0A,$0D0A
02580 0329 20              FCC     /     4=DISPLAY CH Y PDF ON OSCILLISC
02590 0350 0D0A            FDB     $0D0A,$0D0A
02600 0354 20              FCC     /     5=DISPLAY CH Z PDF ON OSCILLISC
02610 037B 0D0A            FDB     $0D0A,$0D0A
02620 037F 20              FCC     /     6=RECONSTRUCT & DISPLAY CH X ON
02630 03A4 4F              FCC     /OSCILLISCOPE/
02640 03B0 0D0A            FDB     $0D0A,$0D0A
02650 03B4 20              FCC     /     7=RECONSTRUCT & DISPLAY CH Y ON
02660 03D9 4F              FCC     /OSCILLISCOPE/
02670 03E5 0D0A            FDB     $0D0A,$0D0A
02680 03E9 20              FCC     /     8=RECONSTRUCT & DISPLAY CH Z ON
```

```
02690 040E 4F                FCC    /OSCILLISCOPE/
02700 041A 0D0A              FDB    $0D0A,$0D0A
02710 041C 20                FCC    /     9=LOAD MEMORY FILE FROM DISK/
02720 0442 0D0A              FDB    $0D,$0A
02730 0443 45                FCC    /ENTER COMMAND NOW=/
02740 0455 04                FCB    4
02750 0456 46         FILEMS FCC    /FILE LOADED. /
02760 0463 50         DISMS1 FCC    /PRESS RETURN/
02770 046F 04                FCB    4
02780                  *
02790                  *
02800                  *FUNCTION       :MAXSCN
02810                  *INPUTS (REG)   :X
02820                  *OUTPUTS        :SHIFT REQUIRED IN SHFCNT
02830                  *CALLS          :NOTHING
02840                  *DESTROYS       :A,B,CC
02850                  *PRUPOSE        :THIS ROUTINE SCANS THE PDF BUFFER PO
02860                  *                TO BY X AND RETURNS THE MINIMUM
02870                  *                SHIFT RIGHTS NECESSARY TO BRING THE
02880                  *                AMPLITUDE COUNTS DOWN TO 11 BITS MAG
02890                  *                FOR OUTPUT FROM THE ST6800 D/A CONVE
02900                  *
02910                  *
02920 0470 DF 19   MAXSCN STX    SAVEX     SAVE PDF ADDR IN X
02930 0472 09             DEX              ADJUST X FOR PROPER INDEXED L
02940 0473 09             DEX
02950 0474 7F 0249        CLR    SHFCNT
02960 0477 7F 024A        CLR    MAXCNT
02970 047A 7C 024A MAXSC1 INC    MAXCNT    256 BYTES SCANNED YET?
02980 047D 27 1E          BEQ    MAXSC4    YES. EXIT WITH CORRECT SHIFT
02990 047F 08             INX              NO. INC X & LOOK AT NEXT 2 BY
03000 0480 08             INX
03010 0481 A6 00          LDA A  0,X       GET MSB BYTE OF VALUE
03020 0483 81 08          CMP A  #08       IS IT LESS THAN 8?
03030 0485 2D F3          BLT    MAXSC1    YES. NO SHIFT NECESS, GO CHEC
03040 0487 5F             CLR B            NO. SET UP COUNTER
03050 0488 0C             CLC              INSURE CARRY CLEAR
03060 0489 5C         MAXSC2 INC B          NOW SHIFT UNTIL MSB 1 BIT HIT
03070 048A 48             ASL A
03080 048B 25 02          BCS    MAXSC3    CARRY SET?
03090 048D 20 FA          BRA    MAXSC2    NO. KEEP SHIFTING LEFT
03100 048F 17         MAXSC3 TBA           YES. PUT SHIFT COUNT IN A
03110 0490 86 06          LDA A  #6        GET MAX SHIFT
03120 0492 10             SBA              AND GET DIFFERENCE
03130 0493 B1 0249        CMP A  SHFCNT    IS SHIFT LE CURRENT SHIFT COU
03140 0496 2F E2          BLE    MAXSC1    YES. IGNORE THIS VALUE & CHEC
03150 0498 B7 0249        STA A  SHFCNT    NO. SAVE THIS SHIFT COUNT
03160 049B 20 DD          BRA    MAXSC1    NOW GO CHECK NEXT BYTE VALUE
03170 049D DE 19   MAXSC4 LDX    SAVEX     RETRIEVE INPUT X
03180 049F 39             RTS
03190                  *
03200                  *
03210                  *FUNCTION       :PDFTRN
03220                  *INPUTS         :X
```

```
03230                *OUTPUTS       :NONE
03240                *CALLS         :NOTHING
03250                *DESTROYS      :A,CC
03260                *PURPOSE       :THIS ROUTINE TRANSFERS THE PDF DATA
03270                *               POINTED TO BY X INTO A WORK BUFFER
03280                *               AREA AT 3200-33FF.
03290                *
03300                *
03310 04A0 DF 19   PDFTRN STX    SAVEX    SAVE INDEX IN TEMP BUFFER
03320 04A2 BF 1C96        STS    STKSAV   SAVE STACK POINTER
03330 04A5 35             TXS             SET UP STACK FOR BUFF TRANSF
03340 04A6 CE 3200        LDX    #$3200   POINT TO WORK BUFF START
03350 04A9 8C 3400 PDFTR1 CPX    #$3400   DONE TRANSFERING YET?
03360 04AC 27 06          BEQ    PDFTR2   YES. RET
03370 04AE 32             PUL A           NO. GET NEXT BYTE
03380 04AF A7 00          STA A  0,X      STORE IN WORK BUFFER
03390 04B1 08             INX
03400 04B2 20 F5          BRA    PDFTR1   GO GET NEXT BYTE TO TRANSFER
03410 04B4 DE 19   PDFTR2 LDX    SAVEX    RETRIEVE ENTRY X
03420 04B6 BE 1C96        LDS    STKSAV   RETRIEVE STACK PTR
03430 04B9 39             RTS
03440                *
03450                *
03460                *FUNCTION      :SCALE
03470                *INPUTS        :DATA IN WORK BUFFER
03480                *OUTPUTS       :DATA IN WORK BUFFER SCALED BY SHFCNT
03490                *CALLS         :NOTHING
03500                *DESTROYS      :A,B,CC
03510                *PURPOSE       :THIS ROUTINE SCALES THE PDF DATA IN
03520                *               THE WORK BUFFER (3200-33FF) BY THE
03530                *               AMOUNT CALCULATED BY MANSON TO ENABL
03540                *               THE ST6800 D/A CONVERTER TO OUTPUT
03550                *               A POSITIVE VOLTAGE PROPORTIONAL TO
03560                *               THE FREQUENCY OF OCCURENCE OF EACH
03570                *               OF THE POSSIBLE 256 AMPLITUDE VALUES
03580                *
03590                *
03600 04BA DF 19   SCALE  STX    SAVEX    SAVE ENTRY X
03610 04BC CE 3200        LDX    #$3200   LOAD WORK BUFFER START ADDRES
03620 04BF 5F             CLR B
03630 04C0 B6 0249 SCALE1 LDA A  SHFCNT   PICK UP SCALE SHIFT COUNT
03640 04C3 27 0E          BEQ    SCALE4   IF SHFCNT = 0, NO SCALING NEC
03650 04C5 27 07   SCALE2 BEQ    SCALE3   IS SHFCNT = 0?
03660 04C7 67 00          ASR    0,X      NO. SCALE DATA POINTED TO BY
03670 04C9 66 01          ROR    1,X
03680 04CB 4A             DEC A
03690 04CC 20 F7          BRA    SCALE2   GO SHIFT DOWN AGAIN
03700 04CE 08      SCALE3 INX              INCREMENT WORK BUFFER DATA PO
03710 04CF 08             INX
03720 04D0 5C             INC B            256 2 BYTE WORDS SHIFTED YET?
03730 04D1 26 ED          BNE    SCALE1   NO. KEEP LOOPING & SHIFTING
03740 04D3 DE 19   SCALE4 LDX    SAVEX    RETRIEVE ENTRY X
03750 04D5 39             RTS
03760                *
```

```
                    *FUNCTION     :OVRLAY
                    *INPUTS       :X
                    *OUTPUTS      :NONE
                    *CALLS        :ROUTINE LOCATED TO BY X
                    *DESTROYS     :ALL REG
                    *PURPOSE      :THIS IS A VECTOR ROUTINE TO CALL OVRLAY
                    * EKG-EXEC TO LOAD ROUTINES FOR EXECUTION.
                    *
04D6 FE 1CAB OVRLAY LDX     OVRLLC
04D9 6E 00          JMP     0,X
                    *
04DB FE 1CA9 PDFPRT LDX     PDFPLC
04DE 6E 00          JMP     0,X
                    *
                    ***************************************************
                    *
                    *       END OF OVERLAY DISPLAY ROUTINES
                    *
                    ***************************************************
                    *
                          END
```

```
                    *
00040               *
                    * OVERLAY NAME :PRSTAT
                    * [illegible]
00070               * VERSION : 1:9
00080               * VERSION DATE : 2 OCT 80
00090               *
00100               *
00110               *
00120               * PROGRAM DESCRIPTION
00130               *
00140               * THIS ROUTINE, WHEN LOADED AND EXECUTED, TAKES
00150               * THE STATUS DATA CURRENTLY PRESENT IN THE
00160               * MEMORY FILE HEADER SECTOR AND PRINTS THIS DATA
00170               * TO THE CONSOLE. THIS ROUTINE CAN BE CALLED
00180               * IMMEDIATELY FOLLOWING A DATA COLLECTION OR
00190               * AFTER A PREVIOUSLY COLLECTED DATA FILE HAS BEEN
00200               * LOADED INTO MEMORY FROM DISK.
00210               *
00220               *
00230               *
00240               *
00250               * START OF PRSTAT
00260               *
00270               *
00280               *
00290 0500                  ORG     $0500     OVERLAY START ADDRESS
00300               *
00310                       OPT     O         ASSB OPT-GEN OBJ FILE
00320                       OPT     NOG       ASSB OPT-SUPPRESS FCC LIST
00330               *
00340               *
00350               *
00360               * LABLE DESCRIPTIONS
00370               *
00380               * SUPPORT SUBROUTINE ADDRESSES
00390               *
00400       CA8F    OUTHCR  EQU     $CA8F     EPROMDOS. ALPH STRING TO CONS
00410       CA36    KEYED0  EQU     $CA36     EPROMDOS. ALPH STRING IN FRM
00420       1D00    START   EQU     $1D00     EKG-EXEC. SOFT EGK-EXEC START
00430       2800    DOS     EQU     $2800     DOS ERROR VECTOR, USED FOR JM
00440               *
00450               * DATA BUFFERS
00460               *
00470       1CA0    LGOFLG  EQU     $1CA0     LOAD VERSUS EXECUTE FLAG
00480       1CA5    HXASLC  EQU     $1CA5     HXASC ADDR PASS BUFFER
00490       1CA7    HXBUF   EQU     $1CA7     HXASC PARAMETER BUFFT
00500       1C96    STKSAV  EQU     $1C96     TEMP STACK SAVE BUFFER
00510       3400    HDRSTR  EQU     $3400     MEM FILE HEADER START
00520       3490    LOOPCT  EQU     $3490     TOTL CAL LOOPS EXEC IN WAIT
00530       3494    SAMPNO  EQU     $3494     NUM OF SAMPLES TAKEN
00540       3496    LPCAL   EQU     $3496     NUM OF CAL LOOPS PER 1 INTR
00550       3497    MAXZ    EQU     $3497     MAX VLU IN CH Z
00560       3498    MAXZLO  EQU     $3498     LOCATION OF MAX VLU IN CH Z
```

```
00570        349A    MINZ    EQU     $349A    MIN VLU IN CH Z
00580        349B    MINZLO  EQU     $349B    LOCATION OF MIN VLU IN CH Z
[illegible]
00610        34A0    MINY    EQU     $34A0    MIN VLU IN CH Y
00620        34A1    MINYLO  EQU     $34A1    LOCATION OF MIN VLU IN CH Y
00630        34A3    MAXX    EQU     $34A3    MAX VLU IN CH X
00640        34A4    MAXXLO  EQU     $34A4    LOCATION OF MAX VLU IN CH X
00650        34A6    MINX    EQU     $34A6    MIN VLU IN CH X
00660        34A7    MINXLO  EQU     $34A7    LOCATION OF MIN VLU IN CH X
00670        34A9    MEMBIT  EQU     $34A9    NUM OF BITS AVAILABLE FOR STO
00680        34AC    DTABIT  EQU     $34AC    NUM OF BITS USED TO STOR DTA
00690        34B0    XBITS   EQU     $34B0    NUM OF BITS USED TO STOR X DT
00700        34B3    YBITS   EQU     $34B3    NUM OF BITS USED TO STOR Y DT
00710        34B6    ZBITS   EQU     $34B6    NUM OF BITS USED TO STOR Z DT
00720        34B9    TBITS   EQU     $34B9    NUM OF BITS USED TO STOR TIME
00730        34BC    ACELCT  EQU     $34BC    # BITS FED TO VAR LEN CODER
00740        34C2    BASSAV  EQU     $34C2    SAVE MEM LOC $3620 FROM BASIC
00750        34C3    ENTRPY  EQU     $34C3    START OF ENTROPY BUFFER
00760        3460    MAXMIN  EQU     $3460    START OF MAX,MIN ASCII DATA
00770                *
00780                *
00790                *FUNCTION :PRSTAT
00800                *INPUTS (REG) :NONE
00810                *OUTPUTS (REG):NONE
00820                *CALLS :OUTPUT,HXASC
00830                *DESTROYS(REG):A,B,CC,X
00840                *PURPOSE : TO LIST STATUS INFORMATION FROM DATA
00850                * COLLECTION FILE TO TERMINAL.
00860                *
00870                *
00880 0500 86 08    PRSTAT LDA A  #8       CLR MSG FIELDS IN STAT OUTPUT
00890 0502 CE 0A66         LDX    #NAME
00900 0505 BD 0930         JSR    MSGCLR
00910 0508 86 35           LDA A  #53
00920 050A CE 0A8C         LDX    #SUBJ
00930 050D BD 0930         JSR    MSGCLR
00940 0510 86 0A           LDA A  #10
00950 0512 CE 0ADF         LDX    #RATE
00960 0515 BD 0930         JSR    MSGCLR
00970 0518 86 14           LDA A  #20
00980 051A CE 0B07         LDX    #DATE
00990 051D BD 0930         JSR    MSGCLR
01000 0520 86 08           LDA A  #8
01010 0522 CE 0B39         LDX    #TIME
01020 0525 BD 0930         JSR    MSGCLR
01030 0528 86 19           LDA A  #25
01040 052A CE 0B5F         LDX    #TYPE
01050 052D BD 0930         JSR    MSGCLR
01060 0530 86 35           LDA A  #53
01070 0532 CE 0F29         LDX    #COMENT
01080 0535 BD 0930         JSR    MSGCLR
01090 0538 86 08           LDA A  #8
01100 053A CE 0B96         LDX    #XENT
```

```
01110 053D BD 0930        JSR     MSGCLR
[illegible]               LDA A   #8
[illegible]               [illegible]
01140 0545 BD 0930        JSR     MSGCLR
01150 0548 86 08          LDA A   #8
01160 054A CE 0FFA        LDX     #ZPNT
01170 054D BD 0930        JSR     MSGCLR
01180 0550 86 08          LDA A   #8
01190 0552 CE 0C2C        LDX     #TPNT
01200 0555 BD 0930        JSR     MSGCLR
01210 0558 86 08          LDA A   #8
01220 055A CE 0CCE        LDX     #MAXCPR
01230 055D BD 0930        JSR     MSGCLR
01240 0560 86 08          LDA A   #8
01250 0562 CE 0D11        LDX     #CTPACH
01260 0565 BD 0930        JSR     MSGCLR
01270 0568 86 05          LDA A   #5
01280 056A CE 0D57        LDX     #CTPHDP
01290 056D BD 0930        JSR     MSGCLR
01300 0570 86 05          LDA A   #5
01310 0572 CE 0D09        LDX     #TPHDP
01320 0575 BD 0930        JSR     MSGCLR
01330 0578 86 05          LDA A   #5
01340 057A CE 0D2B        LDX     #COLMSR
01350 057D BD 0930        JSR     MSGCLR
01360 0580 86 08          LDA A   #8
01370 0582 CE 0E2F        LDX     #AMINX
01380 0585 BD 0930        JSR     MSGCLR
01390 0588 86 08          LDA A   #8
01400 058A CE 0DFD        LDX     #AMAXX
01410 058D BD 0930        JSR     MSGCLR
01420 0590 86 08          LDA A   #8
01430 0592 CE 0E93        LDX     #AMINY
01440 0595 BD 0930        JSR     MSGCLR
01450 0598 86 08          LDA A   #8
01460 059A CE 0F61        LDX     #AMANY
01470 059D BD 0930        JSR     MSGCLR
01480 05A0 86 08          LDA A   #8
01490 05A2 CE 0E93        LDX     #AMINY
01500 05A5 BD 0930        JSR     MSGCLR
01510 05A8 86 08          LDA A   #8
01520 05AA CE 0EC5        LDX     #AMANZ
01530 05AD BD 0930        JSR     MSGCLR
01540 05B0 86 08          LDA A   #8
01550 05B2 CE 0FF7        LDX     #AMINZ
01560 05B5 BD 0930        JSR     MSGCLR
01570 05B8 CE 3402        LDX     #[illegible]2   NOW PUT DATA IN FILE HDR IN M
01580 05BB FF 093D        STX     FNTLOC
01590 05BE CE 0266        LDX     #[illegible]
01600 05C1 BD 093C        JSR     MSGOUT
01610 05C4 CE 0AA[illegible]  LDX     #[illegible]
01620 05C7 BD 093C        JSR     MSGOUT
01630 05CA CE 0A[illegible]   LDX     #[illegible]
01640 05CD BD 093[illegible]  JSR     MSGOUT
```

```
01660 05D3 BD 093C          JSR    MSGPUT
                            LDX    #TIME
                            JSR    MSGPUT
01690 05DC CE 0F29          LDX    #COMENT
01700 05DF BD 093C          JSR    MSGPUT
01710 05E2 FE 34C3          LDX    ENTRPY    GET LOC OF ENTROPY BUFFER
01720 05E5 8C 2020          CPX    #$2020    CHK IF ENTROPY CALC MADE?
01730 05E8 27 03            BEQ    PRST00    NO. STORE NOT CALC MSG
01740 05EA 7E 06A4          JMP    PRSTA0    YES. GET ASCII STRING FROM HD
01750 05ED CE 09FD PRST00   LDX    #ENTCLR
01760 05F0 FF 095D          STX    FRMLOC
01770 05F3 CE 0B96          LDX    #XENT
01780 05F6 BD 093C          JSR    MSGPUT
01790 05F9 CE 09FD          LDX    #ENTCLR
01800 05FC FF 095D          STX    FRMLOC
01810 05FF CE 0FC8          LDX    #YENT
01820 0602 BD 093C          JSR    MSGPUT
01830 0605 CE 09FD          LDX    #ENTCLR
01840 0608 FF 095D          STX    FRMLOC
01850 060B CE 0BFA          LDX    #ZENT
01860 060E BD 093C          JSR    MSGPUT
01870 0611 CE 09FD          LDX    #ENTCLR
01880 0614 FF 095D          STX    FRMLOC
01890 0617 CE 0C2C          LDX    #TENT
01900 061A BD 093C          JSR    MSGPUT
01910 061D CE 09FD          LDX    #ENTCLR
01920 0620 FF 095D          STX    FRMLOC
01930 0623 CE 0CCB          LDX    #MAXCPR
01940 0626 BD 093C          JSR    MSGPUT
01950 0629 CE 09FD          LDX    #ENTCLR
01960 062C FF 095D          STX    FRMLOC
01970 062F CE 0D11          LDX    #CPRACH
01980 0632 BD 093C          JSR    MSGPUT
01990 0635 CE 09FD          LDX    #ENTCLR
02000 0638 FF 095D          STX    FRMLOC
02010 063B CE 0D57          LDX    #CPREFF
02020 063E BD 093C          JSR    MSGPUT
02030 0641 CE 09FD          LDX    #ENTCLR
02040 0644 FF 095D          STX    FRMLOC
02050 0647 CE 0D99          LDX    #TIMEFF
02060 064A BD 093C          JSR    MSGPUT
02070 064D CE 09FD          LDX    #ENTCLR
02080 0650 FF 095D          STX    FRMLOC
02090 0653 CE 0DCB          LDX    #COUNTR
02100 0656 BD 093C          JSR    MSGPUT
02110 0659 CE 09FD          LDX    #ENTCLR
02120 065C FF 095D          STX    FRMLOC
02130 065F CE 0DFD          LDX    #AMAXX
02140 0662 BD 093C          JSR    MSGPUT
02150 0665 CE 09FD          LDX    #ENTCLR
02160 0668 FF 095D          STX    FRMLOC
02170 066B CE 0E2F          LDX    #AMINX
02180 066E BD 093C          JSR    MSGPUT
```

```
02190 0671 CE 09FD          LDX    #ENTCLR
02200 0674 FF 095D          STX    FRMLOC
02210 0677 20 0A            BRA    PRSTA0
02220 0679 BD 093C          JSR    MSGPUT
02230 067C CE 09FD          LDX    #ENTCLR
02240 067F FF 095D          STX    FRMLOC
02250 0682 CE 0E93          LDX    #AMINY
02260 0685 BD 093C          JSR    MSGPUT
02270 0688 CE 09FD          LDX    #ENTCLR
02280 068B FF 095D          STX    FRMLOC
02290 068E CE 0EC5          LDX    #AMAXZ
02300 0691 BD 093C          JSR    MSGPUT
02310 0694 CE 09FD          LDX    #ENTCLR
02320 0697 FF 095D          STX    FRMLOC
02330 069A CE 0EF7          LDX    #AMINZ
02340 069D BD 093C          JSR    MSGPUT
02350 06A0 7E 070A          JMP    PRSTA1
02360 06A3 CE 34C3 PRSTA0   LDX    #ENTRPY  GET ADDRESS OF ENTRPY BUFFER
02370 06A6 FF 095D          STX    FRMLOC
02380 06A9 CE 0B96          LDX    #XENT    NOW PRINT APPROPRIATE DAT TO
02390 06AC BD 093C          JSR    MSGPUT
02400 06AF CE 0BC8          LDX    #YENT
02410 06B2 BD 093C          JSR    MSGPUT
02420 06B5 CE 0BFA          LDX    #ZENT
02430 06B8 BD 093C          JSR    MSGPUT
02440 06BB CE 0C2C          LDX    #TENT
02450 06BE BD 093C          JSR    MSGPUT
02460 06C1 CE 0CCE          LDX    #MAXCPR
02470 06C4 BD 093C          JSR    MSGPUT
02480 06C7 CE 0D11          LDX    #CPRACH
02490 06CA BD 093C          JSR    MSGPUT
02500 06CD CE 0D57          LDX    #CPRERF
02510 06D0 BD 093C          JSR    MSGPUT
02520 06D3 CE 0D99          LDX    #TIMERF
02530 06D6 BD 093C          JSR    MSGPUT
02540 06D9 CE 0DCB          LDX    #COLDUR
02550 06DC BD 093C          JSR    MSGPUT
02560 06DF CE 3460          LDX    #MAXMIN  NOW GET ADDR WHERE ASCII MAX,
02570 06E2 FF 095D          STX    FRMLOC
02580 06E5 CE 0DFD          LDX    #AMAXX
02590 06E8 BD 093C          JSR    MSGPUT
02600 06EB CE 0F2F          LDX    #AMINX
02610 06EE BD 093C          JSR    MSGPUT
02620 06F1 CE 0E61          LDX    #AMAXY
02630 06F4 BD 093C          JSR    MSGPUT
02640 06F7 CE 0E93          LDX    #AMINY
02650 06FA BD 093C          JSR    MSGPUT
02660 06FD CE 0EC5          LDX    #AMAXZ
02670 0701 BD 093C          JSR    MSGPUT
02680 0704 CE 0EF7          LDX    #AMINZ
02690 0707 BD 093C          JSR    MSGPUT
02700 070A FE 3400 PRSTA1   LDX    HDRSTR   NOW DECODE CONTRS TYPE & PUT
02710 070D 8C 4E43          CPX    #$4E43   (NC) NOCPRS?
02720 0710 26 08            BNE    PRSTA2   NO
```

```
[illegible]  [illegible] CE 09F9        LDX   #MOCOMP  YES. POINT TO MOCOMP MSG
02740 0715 FF 095D        STX   FRMLOC
[illegible]  [illegible] 20 42        BRA   PRSTA7
[illegible]  [illegible] [illegible]  [illegible]  [illegible]  [illegible]
02770 071D 26 08          BNE   PRSTA3  NO.
02780 071F CE 09C2        LDX   #TOLA    YES. POINT TO TOLAN-A MSG
02790 0722 FF 095D        STX   FRMLOC
02800 0725 20 35          BRA   PRSTA7
02810 0727 8C 5442 PRSTA3 CPX   #$5442   (TB) TOLAN-B?
02820 072A 26 08          BNE   PRSTA4  NO
02830 072C CE 09CA        LDX   #TOLB    YES. POINT TO TOLAN-B MSG
02840 072F FF 095D        STX   FRMLOC
02850 0732 20 28          BRA   PRSTA7
02860 0734 8C 444F PRSTA4 CPX   #$444F   (DO) DOVER?
02870 0737 26 08          BNE   PRSTA5   NO
02880 0739 CE 09D2        LDX   #DOVR    YES. POINT TO DOVER MSG
02890 073C FF 095D        STX   FRMLOC
02900 073F 20 1B          BRA   PRSTA7
02910 0741 8C 5450 PRSTA5 CPX   #$5450   (TP) TURNING POINT?
02920 0744 26 08          BNE   PRSTA6   NO.
02930 0746 CE 09D8        LDX   #TURNPT  YES. POINT TO TURNING POINT M
02940 0749 FF 095D        STX   FRMLOC
02950 074C 20 0E          BRA   PRSTA7
02960 074E 8C 494E PRSTA6 CPX   #$494E   (IN) 2ND ORDER INTERPOLATOR?
02970 0751 27 03          BEQ   PRST61   YES. TYPE IS RECOGNIZED. CONT
02980 0753 7E 095F        JMP   ERROR    NO. TYPE NOT RECOGNIZED. ERR
02990 0756 CE 09E6 PRST61 LDX   #INTER
03000 0759 FF 095D        STX   FRMLOC
03010 075C CE 0B5F PRSTA7 LDX   #TYPE    PUT CONFRS TYPE IN MSG
03020 075F BD 093C        JSR   MSGPUT
03030 0762 CE 0A12        LDX   #HDRMSG  NOW PT TO STATMSG PAGE 1
03040 0765 BD CA8F        JSR   OUTNCR   AND PRINT PAGE 1
03050 0768 BD CA36        JSR   KEYBD0
03060 076B CE 0CCB        LDX   #HDRMS2  NOW PT TO STATMSG PAGE 2
03070 076E BD CA8F        JSR   OUTNCR   AND PRINT PAGE 2
03080 0771 BD CA36        JSR   KEYBD0
03090 0774 CE 0FD4        LDX   #NUMSMP+2  PICK UP NUM OF SMPLS,
03100 0777 B6 3495        LDA A SAMPNO+1 CONVRT TO ASCII, & STORE
03110 077A BD 0955        JSR   HXASC    TO OUTPUT STRING.
03120 077D CE 0FD2        LDX   #NUMSMP
03130 0780 B6 3494        LDA A SAMPNO
03140 0783 BD 0955        JSR   HXASC
03150 0786 CE 1019        LDX   #NUMLPS  PICK UP VLU OF CAL TST
03160 0789 B6 3496        LDA A LPCAL    & STORE TO OUTPUT STRING
03170 078C BD 0955        JSR   HXASC
03180 078F CE 1060        LDX   #TOTLUP+6  PICK UP TOTL LOOP COUNT
03190 0792 B6 3493        LDA A LOOPCT+3 ,CONVERT TO ASCII & STORE
03200 0795 BD 0955        JSR   HXASC    TO OUTPUT STRING
03210 0798 CE 105E        LDX   #TOTLUP+4
03220 079B B6 3492        LDA A LOOPCT+2
03230 079E BD 0955        JSR   HXASC
03240 07A1 CE 105C        LDX   #TOTLUP+2
03250 07A4 B6 3491        LDA A LOOPCT+1
03260 07A7 BD 0955        JSR   HXASC
```

```
03270 07AA CE 105A      LDX    #TOTLUP
[illegible]             LDA A  LOOPCT
[illegible]             JSR    HXASC
03300 07B3 CE 10C5      LDX    #XMAXNM  PICK UP MAX VLU ON CH X
03310 07B6 B6 34A3      LDA A  MAXX     CON TO ASCII & STR IN
03320 07B9 BD 0955      JSR    HXASC    TO OUTPUT STRING
03330 07BC CE 10E1      LDX    #XMAXLO+2
03340 07BF B6 34A5      LDA A  MAXXLO+1
03350 07C2 BD 0955      JSR    HXASC
03360 07C5 CE 10DF      LDX    #XMAXLO
03370 07C8 B6 34A4      LDA A  MAXXLO
03380 07CB BD 0955      JSR    HXASC
03390 07CE CE 10F2      LDX    #XMINNM  PICK UP MIN VLU ON CH X
03400 07D1 B6 34A6      LDA A  MINX     CONV TO ASCII & STORE IN
03410 07D4 BD 0955      JSR    HXASC    OUTPUT STRING
03420 07D7 CE 110E      LDX    #XMINLO+2
03430 07DA B6 34A8      LDA A  MINXLO+1
03440 07DD BD 0955      JSR    HXASC
03450 07E0 CE 110C      LDX    #XMINLO
03460 07E3 B6 34A7      LDA A  MINXLO
03470 07E6 BD 0955      JSR    HXASC
03480 07E9 CE 111F      LDX    #YMAXNM  PICK UP MAX VLU ON CH Y
03490 07EC B6 349D      LDA A  MAXY     CONV TO ASCII & STORE IN
03500 07EF BD 0955      JSR    HXASC    OUTPUT STRING
03510 07F2 CE 113B      LDX    #YMAXLO+2
03520 07F5 B6 349F      LDA A  MAXYLO+1
03530 07F8 BD 0955      JSR    HXASC
03540 07FB CE 1139      LDX    #YMAXLO
03550 07FE B6 349E      LDA A  MAXYLO
03560 0801 BD 0955      JSR    HXASC
03570 0804 CE 114C      LDX    #YMINNM  PICK UP MIN VLU ON CH Y
03580 0807 B6 34A0      LDA A  MINY     CONV TO ASCII & STORE IN
03590 080A BD 0955      JSR    HXASC    OUTPUT STRING
03600 080D CE 1168      LDX    #YMINLO+2
03610 0810 B6 34A2      LDA A  MINYLO+1
03620 0813 BD 0955      JSR    HXASC
03630 0816 CE 1166      LDX    #YMINLO
03640 0819 B6 34A1      LDA A  MINYLO
03650 081C BD 0955      JSR    HXASC
03660 081F CE 1179      LDX    #ZMAXNM  PICK UP MAX VLU ON CH Z
03670 0822 B6 3497      LDA A  MAXZ     CONV TO ASCII & STORE IN
03680 0825 BD 0955      JSR    HXASC    OUTPUT STRING
03690 0828 CE 1195      LDX    #ZMAXLO+2
03700 082B B6 3499      LDA A  MAXZLO+1
03710 082E BD 0955      JSR    HXASC
03720 0831 CE 1193      LDX    #ZMAXLO
03730 0834 B6 3498      LDA A  MAXZLO
03740 0837 BD 0955      JSR    HXASC
03750 083A CE 11A6      LDX    #ZMINNM  PICK UP MIN VLU ON CH Z
03760 083D B6 349A      LDA A  MINZ     CONV TO ASCII $ STORE IN
03770 0840 BD 0955      JSR    HXASC    OUPUT STRING
03780 0843 CE 11C2      LDX    #ZMINLO+2
03790 0846 B6 349C      LDA A  MINZLO+1
03800 0849 BD 0955      JSR    HXASC
```

```
03810 084C CE 11C0      LDX    #MINZLO
03820 084F B6 349B      LDA A  MINZLO
[illegible]             JSR    [illegible]
[illegible]             LDX    [illegible]
03850 0858 B6 34A3      LDA A  MEMBIT+2 AVAILABLE FOR DATA STOR
03860 085B BD 0955      JSR    HXASC    , CONV TO ASCII & STOR
03870 085E CE 120C      LDX    #AVABIT+2  IN OUTPUT STRING
03880 0861 B6 34AA      LDA A  MEMBIT+1
03890 0864 BD 0955      JSR    HXASC
03900 0867 CE 120A      LDX    #AVABIT
03910 086A B6 34A9      LDA A  MEMBIT
03920 086D BD 0955      JSR    HXASC
03930 0870 CE 124B      LDX    #TCPBIT+6
03940 0873 B6 34BF      LDA A  ACELCT+3
03950 0876 BD 0955      JSR    HXASC
03960 0879 CE 1249      LDX    #TCPBIT+4
03970 087C B6 34BE      LDA A  ACELCT+2
03980 087F BD 0955      JSR    HXASC
03990 0882 CE 1247      LDX    #TCPBIT+2
04000 0885 B6 34BD      LDA A  ACELCT+1
04010 0888 BD 0955      JSR    HXASC
04020 088B CE 1245      LDX    #TCPBIT
04030 088E B6 34BC      LDA A  ACELCT
04040 0891 BD 0955      JSR    HXASC
04050 0894 CE 127F      LDX    #TOTBIT+6  PICK UP NUM OF BITS
04060 0897 B6 34AF      LDA A  DTABIT+3 ACTUALL STORED
04070 089A BD 0955      JSR    HXASC    BECAUSE OF DATA
04080 089D CE 127D      LDX    #TOTBIT+4  COMPRESSION, CONV TO
04090 08A0 B6 34AE      LDA A  DTABIT+2 ASCII & STORE IN
04100 08A3 BD 0955      JSR    HXASC    OUTPUT STRING
04110 08A6 CE 127B      LDX    #TOTBIT+2
04120 08A9 B6 34AD      LDA A  DTABIT+1
04130 08AC BD 0955      JSR    HXASC
04140 08AF CE 1279      LDX    #TOTBIT
04150 08B2 B6 34AC      LDA A  DTABIT
04160 08B5 BD 0955      JSR    HXASC
04170 08B8 CE 12B7      LDX    #XBITNM+4  PICK UP NUM OF BITS
04180 08BB B6 34B2      LDA A  XBITS+2  ACTUALLY STORED FOR
04190 08BE BD 0955      JSR    HXASC    DATA IN CHANNEL X,
04200 08C1 CE 12B5      LDX    #XBITNM+2  CONV TO ASCII &
04210 08C4 B6 34B1      LDA A  XBITS+1  STORE TO OUTPUT STRING
04220 08C7 BD 0955      JSR    HXASC
04230 08CA CE 12B3      LDX    #XBITNM
04240 08CD B6 34B0      LDA A  XBITS
04250 08D0 BD 0955      JSR    HXASC
04260 08D3 CE 12EF      LDX    #YBITNM+4  PICK UP NUM OF BITS
04270 08D6 B6 34B5      LDA A  YBITS+2  ACTUALLY STORED FOR
04280 08D9 BD 0955      JSR    HXASC    DATA IN CHANNEL Y,
04290 08DC CE 12ED      LDX    #YBITNM+2  CONV TO ASCII &
04300 08DF B6 34B4      LDA A  YBITS+1  STORE IN OUTPUT STRING
04310 08E2 BD 0955      JSR    HXASC
04320 08E5 CE 12EB      LDX    #YBITNM
04330 08E8 B6 34B3      LDA A  YBITS
04340 08EB BD 0955      JSR    HXASC
```

```
04350 08EE CE 1327        LDX    #ZBITNM+4  PICK UP NUM OF BITS
04360 08F1 B6 34B8        LDA A  ZBITS+2  ACTUALLY STORED FOR
04370 08F4 BD 0955        JSR    HXASC    DATA IN CHANNEL 0,
04380 08F7 CE 1323        LDX    #ZBITNM+2  CONV TO ASCII &
04390 08FA B6 34B7        LDA A  ZBITS+1  STORE IN OUTPUT STRING
04400 08FD BD 0955        JSR    HXASC
04410 0900 CE 1323        LDX    #ZBITNM
04420 0903 B6 34B6        LDA A  ZBITS
04430 0906 BD 0955        JSR    HXASC
04440 0909 CE 136D        LDX    #TBITNM+4  PICK UP NUM OF BITS
04450 090C B6 34BB        LDA A  TBITS+2  ACTUALLY STORED FOR
04460 090F BD 0955        JSR    HXASC    TIME (OR OTHER PARAMETER)
04470 0912 CE 136B        LDX    #TBITNM+2  CONV TO ASCII &
04480 0915 B6 34BA        LDA A  TBITS+1  STORE IN OUTPUT STRING
04490 0918 BD 0955        JSR    HXASC
04500 091B CE 1369        LDX    #TBITNM
04510 091E B6 34B9        LDA A  TBITS
04520 0921 BD 0955        JSR    HXASC
04530 0924 CE 0F89        LDX    #COSTAT  GET OUTPUT STRING ADDR
04540 0927 BD CA8F        JSR    OUTNCR   & SEND TO CONSOLE
04550 092A BD CA36        JSR    KEYBD0   WAIT FOR TERM INPUT
04560 092D 7E 2800        JMP    DOS      JUMP BACK TO CALLING ROUTINE
04570                *
04580                *FUNC: MSGCLR
04590                *INPUTS: A (# OF SPACES TO CLR),X (LOC TO PUT SPACES
04600                *OUTPUTS: ASCII $20 TO MEM AT X
04610                *CALLS: NOTHING
04620                *DESTROYS: A,B,CC
04630                *PURPOSE: THIS ROUTINE CLRS THE MSG BUFFER EACH TIME
04640                * PRSTAT IS CALLED FOR NEW INFO OFF OF THE MEM DATA
04650                *
04660 0930 4C      MSGCLR INC A           INC COUNTER
04670 0931 C6 20          LDA B  #$20     ASCII SPACE
04680 0933 4A      MSGCL1 DEC A
04690 0934 27 05          BEQ    MSGCL2   IS BUFFER CLRED?
04700 0936 E7 00          STA B  0,X      NO. KEEP LOOPING
04710 0938 08             INX
04720 0939 20 F8          BRA    MSGCL1
04730 093B 39      MSGCL2 RTS
04740                *
04750                * END OF MSGCLR
04760                *
04770                ********************
04780                *
04790                *FUNC: MSGPUT
04800                *INPUTS: X(ADDR WHERE DATA IN FRMLOC GOING)
04810                *OUTPUTS: ASCII DATA TO ADDR IN X
04820                *CALLS: NOTHING
04830                *DESTROYS: A,X,CC
04840                *PURPOSE: THIS ROUTINES TRANSFER ASCII TEXT
04850                * FROM MEM FILE HDR TO STAT MSG BUFFER
04860                *
04870 093C BF 1C96 MSGPUT STS    STKSAV   SAVE CURRENT STACK PTR
04880 093F BE 095D        LDS    FRMLOC   GET LOC OF DATA FOR TRANSFER
```

```
048[illegible] 0942 34            DES
04900 0943 32      MSGPU1 PUL A
0[illegible]  [illegible] 81 04         CMP A  #4       END OF MSG?
04[illegible]  [illegible] 27 05        BEQ    MSGPU2   YES. EXIT
04930 0948 A7 00          STA A  0,X      NO. PUT CHAR IN MSG
04940 094A 08             INX
04950 094B 20 F6          BRA    MSGPU1
04960 094D 31      MSGPU2 INS             UPDATE STK WITH NEXT TRNSFR L
04970 094E BF 095D        STS    FRMLOC   STR BACK IN FRMLOC
04980 0951 BE 1C96        LDS    STKSAV   PICK UP ENTRY STACK PTR
04990 0954 39             RTS
05000              *
05010              * END MSGPUT
05020              *
05030              ************
05040              *
05050              *FUNC: HXASC
05060              *INPUTS:A,X (DATA TO BE CONV'D,ADDR TO STORE ASCII O
05070              *CALLS: HSASC ROUTINE IN EKG-EXEC
05080              *DISTROYS: A,X,CC
05090              *PURPOSE: THIS IS A RELOC PASS ROUTINE TO PICK UP
05100              * ADDRO OF HXASC IN EKG-EXEC AND JUMP TO IT
05110              *
05120 0955 FF 1CA7 HXASC  STX    HXBUF    SAVE X IN PARAMATER BUFFER
05130 0958 FE 1CA5        LDX    HXASLC   GET ADDR OF CONV ROUTINE IN E
05140 095B 6E 00          JMP    0,X      JUMP TO IT
05150              *
05160              * END OF HXASC
05170              *
05180              **************
05190              *
05200 095D 0002    FRMLOC RMB    2        DATA FROM LOC FOR MSGPUT
05210              *
05220 095F CE 096B ERROR  LDX    #ERRMSG
05230 0962 BD CA8F        JSR    OUTNCR
05240 0965 BD CA36        JSR    KEYED0
05250 0968 7E 1D00        JMP    START
05260              *
05270 096B 0707    ERRMSG FDB    $0707,$0D0A
05280 096F 4D             FCC    /MEMORY FILE COMPRESSION TYPE /
05290 098C 4E             FCC    /NOT RECOGNIZED. PRESS RETURN/
05300 09A8 04             FCB    4
05310              *
05320              ************************************************
05330              * END PRSTAT
05340              ************************************************
05350              *
05360              ************************************************
05370              *
05380              * OUTPUT STRING TO LIST TO CONSOLE
05390              *
05400              ************************************************
05410              *
05420 09A9 4E      NOCOMP FCC    /NO COMPRESSION PERFORMED/
```

```
05430 09C1 04            FCB   4
05440 09C2 54      TOLA  FCC   /TOLAN-A/
05450 09C9 04            FCB   4
05460 09CA 54      TOLB  FCC   /TOLAN-B/
05470 09D1 04            FCB   4
05480 09D2 44      DOWR  FCC   /DOWER/
05490 09D7 04            FCB   4
05500 09D8 54      TURNPT FCC  /TURNING POINT/
05510 09E5 04            FCB   4
05520 09E6 32      INTER FCC   /2ND ORDER INTERPOLATOR/
05530 09FC 04            FCB   4
05540 09FD 43      ENTCLR FCC  /CALCULATION NOT MADE/
05550 0A11 04            FCB   4
05560 0A12 1A07    HDRMSG FDB  $1A07,$0C0D,$0A0A,$0A0A,$0A0A,$0A0A
05570 0A1E 3A            FCC   /:EKG SAMPLE COLLECTION STATISTICS/
05580 0A3F 20            FCC   / : PAGE 1/
05590 0A48 0D0A          FDB   $0D0A,$0D0A
05600 0A4C 46            FCC   /FILENAME. . . . . /
05610 0A66 0008    NAME  RMB   8
05620 0A6E 0D0A          FDB   $0D0A,$0D0A
05630 0A72 53            FCC   /SUBJECT . . . . . /
05640 0A8C 0035    SUBJ  RMB   53
05650 0AC1 0D0A          FDB   $0D0A,$0D0A
05660 0AC5 53            FCC   /SAMPLING RATE . . . . . /
05670 0ADF 000A    RATE  RMB   10
05680 0AE9 0D0A          FDB   $0D0A,$0D0A
05690 0AED 44            FCC   /DATE OF COLLECTION. . . . /
05700 0B07 0014    DATE  RMB   20
05710 0B1B 0D0A          FDB   $0D0A,$0D0A
05720 0B1F 54            FCC   /TIME OF COLLECTION. . . . /
05730 0B39 0008    TIME  RMB   8
05740 0B41 0D0A          FDB   $0D0A,$0D0A
05750 0B45 43            FCC   /COMPRESSION USED. . . . . /
05760 0B5F 0019    TYPE  RMB   25
05770 0B78 0D0A          FDB   $0D0A,$0D0A
05780 0B7C 43            FCC   /CHANNEL X ENTROPY . . . . /
05790 0B96 0008    XENT  RMB   8
05800 0B9E 20            FCC   / BITS       /
05810 0BAA 0D0A          FDB   $0D0A,$0D0A
05820 0BAE 43            FCC   /CHANNEL Y ENTROPY . . . . /
05830 0BC8 0008    YENT  RMB   8
05840 0BD0 20            FCC   / BITS       /
05850 0BDC 0D0A          FDB   $0D0A,$0D0A
05860 0BE0 43            FCC   /CHANNEL Z ENTROPY . . . . /
05870 0BFA 0008    ZENT  RMB   8
05880 0C02 20            FCC   / BITS       /
05890 0C0E 0D0A          FDB   $0D0A,$0D0A
05900 0C12 54            FCC   /TOTAL SOURCE ENTROPY. . . /
05910 0C2C 0008    TENT  RMB   8
05920 0C34 20            FCC   / BITS       /
05930 0C40 0D0A          FDB   $0D0A,$0D0A
05940 0C44 50            FCC   /PRESS RETURN FOR PAGE 2 OF STATISTIC
05950 0C6A 04            FCB   4
05960 0C6B 1A07    HDRMS2 FDB  $1A07,$0D0A
```

```
05970 0C6F 45              FCC   /EKG SAMPLE COLLECTION STATISTICS/
05980 0C8F 20              FCC   / : PAGE 2/
[illegible]                FDB   [illegible]
06000 0C9C 41              FCC   /[illegible]/
06010 0CB2 0D0A            FDB   $0D0A
06020 0CB4 52              FCC   /RATIO POSSIBLE. . . . . /
06030 0CCE 0008  MAXCPR    RMB   8
06040 0CD6 20              FCC   / : 1        /
06050 0CE2 0D0A            FDB   $0D0A
06060 0CE4 43              FCC   /COMPRESSION RATIO/
06070 0CF5 0D0A            FDB   $0D0A
06080 0CF7 41              FCC   /ACHIEVED. . . . . /
06090 0D11 0008  CPRACH    RMB   8
06100 0D19 20              FCC   / : 1        /
06110 0D25 0D0A            FDB   $0D0A
06120 0D27 41              FCC   /ACHIEVED COMPRESSION/
06130 0D3B 0D0A            FDB   $0D0A
06140 0D3D 45              FCC   /EFFICIENCY. . . . . /
06150 0D57 0005  CPREFF    RMB   5
06160 0D5C 20              FCC   / % OF MAXIMUM  /
06170 0D6B 0D0A            FDB   $0D0A
06180 0D6D 43              FCC   /COMPRESSION TIME/
06190 0D7D 0D0A            FDB   $0D0A
06200 0D7F 45              FCC   /EFFICIENCY OBTAINED . . . /
06210 0D99 0005  TIMEFF    RMB   5
06220 0D9E 20              FCC   / % SMP INTERVAL/
06230 0DAD 0D0A            FDB   $0D0A,$0D0A
06240 0DB1 43              FCC   /COLLECTION DURATION . . . /
06250 0DCB 0005  COLDUR    RMB   5
06260 0DD0 20              FCC   / SECONDS       /
06270 0DDF 0D0A            FDB   $0D0A,$0D0A
06280 0DE3 43              FCC   /CHANNEL X MAXIMUM . . . . /
06290 0DED 0008  AMAXX     RMB   8
06300 0E05 20              FCC   / VOLTS       /
06310 0E11 0D0A            FDB   $0D0A,$0D0A
06320 0E15 43              FCC   /CHANNEL X MINIMUM . . . . /
06330 0E2F 0008  AMINX     RMB   8
06340 0E37 20              FCC   / VOLTS       /
06350 0E43 0D0A            FDB   $0D0A,$0D0A
06360 0E47 43              FCC   /CHANNEL Y MAXIMUM . . . . /
06370 0E61 0008  AMAXY     RMB   8
06380 0E69 20              FCC   / VOLTS       /
06390 0E75 0D0A            FDB   $0D0A,$0D0A
06400 0E79 43              FCC   /CHANNEL Y MINIMUM . . . . /
06410 0E93 0008  AMINY     RMB   8
06420 0E9B 20              FCC   / VOLTS       /
06430 0EA7 0D0A            FDB   $0D0A,$0D0A
06440 0EAB 43              FCC   /CHANNEL Z MAXIMUM . . . . /
06450 0EC5 0008  AMAXZ     RMB   8
06460 0ECD 20              FCC   / VOLTS       /
06470 0ED9 0D0A            FDB   $0D0A,$0D0A
06480 0EDD 43              FCC   /CHANNEL Z MINIMUM . . . . /
06490 0EF7 0008  AMINZ     RMB   8
06500 0EFF 20              FCC   / VOLTS       /
```

```
06510 0F0B 0D0A          FDB    $0D0A,$0D0A
06520 0F0F 43            FCC    /COMMENTS. . . . . . . . . /
                         (S...
06540 0F2A 0D0A          FDB    $0D0A,$0D0A
06550 0F62 50            FCC    /PRESS RETURN FOR PAGE 3 OF STATISTIC
06560 0F88 04            FCB    4
06570 0F89 1A07  COSTAT  FDB    $1A07,$0C0D,$0A0A,$0A0A,$0A0A,$0A0A
06580 0F95 45            FCC    /EKG SAMPLE COLLECTION STATISTICS: PA
06590 0FBD 0D0A          FDB    $0D0A    CR/LF
06600 0FBF 20            FCC    / NUMBER OF SAMPLES TAKEN /
06610 0FD8 28            FCC    /(SAMPNO)= /
06620 0FE2 0004  NUMSMP  RMB    4
06630 0FE6 20            FCC    / (HEX)/
06640 0FEC 0D0A          FDB    $0D0A    CR/LF
06650 0FEE 20            FCC    / MAXIMUM LOOP COUNT PER/
06660 1005 20            FCC    / INTERRUPT (LPCNT)= /
06670 1019 0002  NUMLPS  RMB    2
06680 101B 20            FCC    / (HEX)/
06690 1021 0D0A          FDB    $0D0A    CR/LF
06700 1023 20            FCC    / TOTAL WAITING LOOP COUNTS DURING/
06710 1044 20            FCC    / COLLECTION (LOOPCT)= /
06720 105A 0008  TOTLUP  RMB    8
06730 1062 20            FCC    / (HEX)/
06740 1068 0D0A          FDB    $0D0A
06750 106A 54            FCC    /TIME EFFICIENCY = (1-(LOOPCT/(SAMPNO
06760 1097 2A            FCC    /*100/
06770 109B 0D0A          FDB    $0D0A,$0D0A
06780 109F 43            FCC    /CHANNEL MAXIMUMS AND MINIMUMS/
06790 10BC 0D0A          FDB    $0D0A
06800 10BE 20            FCC    / XMAX= /
06810 10C5 0002  XMAXNM  RMB    2
06820 10C7 20            FCC    / (HEX) AT SAMPLE NUMBER /
06830 10DF 0004  XMAXLO  RMB    4
06840 10E3 20            FCC    / (HEX)/
06850 10E9 0D0A          FDB    $0D0A
06860 10EB 20            FCC    / XMIN= /
06870 10F2 0002  XMINNM  RMB    2
06880 10F4 20            FCC    / (HEX) AT SAMPLE NUMBER /
06890 110C 0004  XMINLO  RMB    4
06900 1110 20            FCC    / (HEX)/
06910 1116 0D0A          FDB    $0D0A
06920 1118 20            FCC    / YMAX= /
06930 111F 0002  YMAXNM  RMB    2
06940 1121 20            FCC    / (HEX) AT SAMPLE NUMBER /
06950 1139 0004  YMAXLO  RMB    4
06960 113D 20            FCC    / (HEX)/
06970 1143 0D0A          FDB    $0D0A
06980 1145 20            FCC    / YMIN= /
06990 114C 0002  YMINNM  RMB    2
07000 114E 20            FCC    / (HEX) AT SAMPLE NUMBER /
07010 1166 0004  YMINLO  RMB    4
07020 116A 20            FCC    / (HEX)/
07030 1170 0D0A          FDB    $0D0A
07040 1172 20            FCC    / ZMAX= /
```

```
07050 1179 0002   ZMAXNM RMB   2
                         FCC   / (HEX) AT SAMPLE NUMBER /
07080 1197 20            FCC   / (HEX)/
07090 119D 0D0A          FDB   $0D0A
07100 119F 20            FCC   / ZMIN= /
07110 11A6 0002   ZMINNM RMB   2
07120 11A8 20            FCC   / (HEX) AT SAMPLE NUMBER /
07130 11C0 0004   ZMINLO RMB   4
07140 11C4 20            FCC   / (HEX)/
07150 11CA 0D0A          FDB   $0D0A,$0D0A
07160 11CE 43            FCC   /COMPRESSION STATISTICS:/
07170 11E5 0D0A          FDB   $0D0A
07180 11E7 20            FCC   / NUMBER OF MEMORY BITS /
07190 11FE 41            FCC   /AVAILABLE = /
07200 120A 0006   AVABIT RMB   6
07210 1210 20            FCC   / (HEX)/
07220 1216 0D0A          FDB   $0D0A
07230 1218 20            FCC   / NUMBER OF BITS AVAILABLE/
07240 1231 20            FCC   / TO VAR LEN CODER = /
07250 1245 0008   TCPBIT RMB   8
07260 124D 20            FCC   / (HEX)/
07270 1253 0D0A          FDB   $0D0A
07280 1255 20            FCC   / TOTAL NUMBER OF DATA BITS/
07290 126F 20            FCC   / STORED = /
07300 1279 0008   TOTBIT RMB   8
07310 1281 20            FCC   / (HEX)/
07320 1287 0D0A          FDB   $0D0A
07330 1289 20            FCC   / NUMBER OF BITS USED TO/
07340 12A0 20            FCC   / STORE CHANNEL X = /
07350 12B3 0006   XBITNM RMB   6
07360 12B9 20            FCC   / (HEX)/
07370 12BF 0D0A          FDB   $0D0A
07380 12C1 20            FCC   / NUMBER OF BITS USED TO/
07390 12D8 20            FCC   / STORE CHANNEL Y = /
07400 12EB 0006   YBITNM RMB   6
07410 12F1 20            FCC   / (HEX)/
07420 12F7 0D0A          FDB   $0D0A
07430 12F9 20            FCC   / NUMBER OF BITS USED TO/
07440 1310 20            FCC   / STORE CHANNEL Z = /
07450 1323 0006   ZBITNM RMB   6
07460 1329 20            FCC   / (HEX)/
07470 132F 0D0A          FDB   $0D0A
07480 1331 20            FCC   / NUMBER OF BITS USED TO/
07490 1348 20            FCC   / STORE TIME OR OTHER PARAM/
07500 1362 45            FCC   /ETER = /
07510 1369 0006   TBITNM RMB   6
07520 136F 20            FCC   / (HEX)/
07530 1375 0D0A          FDB   $0D0A
07540 1377 43            FCC   /COMPRESSION RATIO = TOTAL /
07550 1391 44            FCC   /DATA BITS STORED PER MEM /
07560 13AA 42            FCC   /BITS AVAILABLE/
07570 13B8 0D0A          FDB   $0D0A,$0D23
07580 13BC 50            FCC   /PRESS RETURN =/
```

```
07590 13CA 04                FCB    4
07600                *
07610                *  END OF PRSTAT
07620                *
07630                      END
```

```
00030                ****************************************************
00040                *
00050                * CALLED UNIT :  NOCPRS
00060                * AUTHOR :  CAPT. MEL TOWNSEND
00070                * VERSION : 1.7
00080                * VERSION DATE : 3 OCT 80
00090                *
00100                ****************************************************
00110                *
00120                * OVERLAY DESCRIPTION
00130                *
00140                * THIS OVERLAY SAMPLES THE EKG DATA VIA THE
00150                * A/D CONVERTERS AND STORES THE 8 BIT ROUNDED
00160                * VALUES INTO MEMORY LOCATIONS 3C00-7FFF. THE
00170                * THE X,Y,Z CHANNELS ARE SAMPLED AND STORED
00180                * SEQUENTIALLY STARTING AT 3C00 AND WORKING UP.
00190                *
00200                ****************************************************
00210                ****************************************************
00220                *
00230                * START OF NOCPRS
00240                *
00250                ****************************************************
00260                *
00270 0100                 ORG     $0100     OVERLAY START ADDRESS
00280                *
00290                      OPT     O         ASSB OPT-GEN CRN FILE
00300                      OPT     NOG       ASSB OPT-SUPPRESS FULL PCC LI
00310                *
00320                ****************************************************
00330                *
00340                * LABLE DECLARATIONS
00350                *
00360                * SUPPORT SUBROUTINE ADDRESSES
00370                *
00380      CA87      OUTPUT EQU     $CA87     EPROMDOS. ALPH STRING TO CONS
00390      CA8F      OUTCR  EQU     $CA8F     EPROMDOS. ALPH STRING,NO CRLF
00400      CA2C      KEYIN  EQU     $CA2C     EPROMDOS. CONSOLE INPUT NO TI
00410      CA36      KEYIN0 EQU     $CA36     EPROMDOS. CONSOLE INPUT, NO ?
00420      1D00      START  EQU     $1D00     EKG-EXEC. START EKG-EXEC
00430                *
00440                * DATA BUFFERS
00450                *
00460      3400      HDRSTR EQU     $3400     DATA MEMORY BUFFER HEADER
00470      FFF8      IRQVEC EQU     $FFF8     INTERRUPT VECTOR ADDR
00480      1C96      STKSAV EQU     $1C96     STACK SAVE BUFFER
00490      1C98      CPPTYP EQU     $1C98     CONFIGURATION ANALYSIS
00500      3002      ENDBUF EQU     $3002     ADDR OF LAST CHIP RAM
00510      1C9D      VECSAV EQU     $1C9D     IRQ VECTOR SAVE
00520      1CA1      FILHLC EQU     $1CA1     FILLER ADDR HOLDER
00530      1CA3      SAVHLC EQU     $1CA3     SAVFIL ADDR HOLDER
00540      3490      LOOPCT EQU     $3490     TOTAL NUMBER OF LOOPS
00550      3494      SAMPNO EQU     $3494     NUMBER OF SAMPLES
00560      3496      LOCAL  EQU     $3496     AVGD MAX LOOP COUNT
```

```
00570        3497    MAXZ    EQU     $3497     MAX VLU IN CH Z
00580        3498    MAXZLO  EQU     $3498     MAX VLU LOC IN CH Z
00590        3499    MINZ    EQU     $3499     MIN VLU IN CH Z
00600        349A    MINZLO  EQU     $349A     MIN VLU LOC IN CH Z
00610        349D    MAXY    EQU     $349D     MAX VLU IN CH Y
00620        349E    MAXYLO  EQU     $349E     MAX VLU LOC IN CH Y
00630        34A0    MINY    EQU     $34A0     MIN VLU IN CH Y
00640        34A1    MINYLO  EQU     $34A1     MIN VLU LOC IN CH Y
00650        34A3    MAXX    EQU     $34A3     MAX VLU IN CH X
00660        34A4    MAXXLO  EQU     $34A4     MAX VLUE LOC IN CH X
00670        34A6    MINX    EQU     $34A6     MIN VLU IN CH X
00680        34A7    MINXLO  EQU     $34A7     MIN VLU LOC IN CH X
00690        34AC    NUMBIT  EQU     $34AC     NUM OF BITS USED TO STR DEV.
00700        34B0    XBITS   EQU     $34B0     NUM OF BITS USED TO STR X
00710        34B3    YBITS   EQU     $34B3     NUM OF BITS USED TO STR Y
00720        34B6    ZBITS   EQU     $34B6     NUM OF BITS USED TO STR Z
00730        34B9    TBITS   EQU     $34B9     NUM OF BITS USED TO STR T
00740        34BC    ACELCT  EQU     $34BC     # BITS SHF TO VLU IN CTR ACC
00750        3600    XPDFM   EQU     $3600     0 VLU LOC OF X PDF
00760        3800    YPDFM   EQU     $3800     0 VLU LOC OF Y PDF
00770        3A00    ZPDFM   EQU     $3A00     0 VLU LOC OF Z PDF
00780        3B00    TPDF    EQU     $3B00     0 VLU LOC TO TIME PDF
00790        3C00    SECZRO  EQU     $3C00     START OF DATA MEM FILE
00800                *
00810                * HARDWARE ADDRESSES
00820                *
00830        E400    ADCZRO  EQU     $E400     ADC CHANNEL ZERO
00840        E404    ADCTWO  EQU     $E404     ADC CHANNEL TWO
00850        E500    DACZRO  EQU     $E500     DAC CHANNEL ZERO
00860                *
00870                *
00880                *FUNCTION :NOCPRS
00890                *INPUTS :STATUS REGISTERS FLAG EKG-EXEC
00900                *OUTPUTS :DATA TO DISK
00910                *CALLS  :OUTPUT,PRINTER,KEYED,OUTSTR
00920                * KEYBD0,SAVTTL,START
00930                *DESTROYS :ALL REGISTERS
00940                *PURPOSE :TO COLLECT 3 CHANNELS OF EKG
00950                * DATA AND STORE INTO MEMORY.
00960                *
00970                *
00980 0100 0F       NOCPRS SEI
00990 0101 CE 0233         LDX    #STRTSG
01000 0104 BD CA8F         JSR    OUTSTR   "THIS PROGRAM SAMPLES TWO..."
01010 0107 BD CA2C         JSR    KEYED    GET RESPONSE FROM CONSOLE
01020 010A FE 3002         LDX    ENDDAT
01030 010D 09              DEX
01040 010E F6 00           LDA B  0,X
01050 0110 C1 59           CMP B  #'Y      IS INPUT YES?
01060 0112 27 03           BEQ    NOCPR1   YES. KEEP EXECUTING THIS ROUT
01070 0114 7E 1D00         JMP    START    NO. RTN TO EKG-EXEC
01080 0117 CE 4F43 NOCPR1  LDX    #$4F43   FLAG COMPRESSION TYPE (EC)
01090 011A FF 1C98         STX    CMPTYP
01100 011D BD 051A         JSR    FILHDR   SET UP DATA FILE HEADER
```

```
01110 0120 CE 0285          LDX    #IRQMSG
[illegible]                 [illegible]    "[illegible]"
[illegible]                 [illegible]
01140 0129 CE 3C00 NOCPGO   LDX    #SEC2RO   INITIALIZE REF TO START LOC
01150 012C FF 0230          STX    BUFPTR    STORE IN BUFFER POINTER
01160 012F 86 03            LDA A  #3        PICK UP COUNTER FOR TIME CAL
01170 0131 B7 0227          STA A  CALCNT    STORE IN BUFFER
01180 0134 FE FFF8          LDX    IRQVEC    PICK UP CURRENT IRQ VECTOR
01190 0137 FF 1C9D          STX    VECSAV    SAVE IN BUFFER
01200 013A CE 051F TIMCAL   LDX    #CALINT   GET INTR VECTOR ADDR FOR CAL
01210 013D FF FFF8          STX    IRQVEC    PUT IN VECTOR ADDRESS
01220 0140 86 00            LDA A  #0        INIT COUNTER FOR 256 TEST LOO
01230 0142 B7 0232          STA A  DONTST    STORE IN DONTST BUFFER
01240 0145 CE 0000          LDX    #0        CLEAR LOOP COUNTER
01250 0148 FF 022C          STX    IRQCNT
01260 014B FF 022E          STX    IRQCNT+2
01270 014E FF E500          STX    DACZRO    PULSE INT ENABLE CIRCUIT
01280 0151 20 09            BRA    SPOO1     BRA TO COUNTING LOOP
01290                *
01300                **********************************************
01310                *
01320                * BASIC TIMING LOOP FOR EFFICIENCY TEST
01330                *
01340                **********************************************
01350                *
01360 0153 01       SPLOOP NOP               DELAY TO MATCH TIME
01370 0154 01              NOP               IT TAKES TO EXECUTE THE
01380 0155 01              NOP               INCRITMENT OF BYTES 3&4
01390 0156 01              NOP               OF THE LOOP COUNTER
01400 0157 01              NOP               WHEN COUNT CARRIES TO
01410 0158 01              NOP               HIGH 2 BYTES OF 4 BYTE
01420 0159 7E 015F         JMP    SPOO3      JUMP TO CONTINUE LOOP
01430 015C 0E       SPOO1  CLI               PREPARE FOR TIMER INT
01440 015D 3E              WAI               STOP PROCESSOR & WAIT
01450 015E 0E       SPOO4  CLI
01460 015F FE 022E         LDX    IRQCNT+2  RET'ED FROM INT, INC COUNT
01470 0162 08              INX
01480 0163 FF 022E         STX    IRQCNT+2  SAVE IT
01490 0166 26 EB           BNE    SPLOOP    COUNT GONE FFFF TO 0000?
01500 0168 FE 022C         LDX    IRQCNT    YES. INC BYTES 3&4
01510 016B 08              INX
01520 016C FF 022C         STX    IRQCNT    SAVE COUNT
01530 016F B6 0232 SPOO3   LDA A  DONTST    IS DONE TEST SATISFIED?
01540 0172 81 00           CMP A  #0
01550 0174 26 E8           BNE    SPOO4     NO. KEEP LOOPING
01560                *
01570                **********************************************
01580                *
01590                * END BASIC TIMING LOOP FOR EFF TEST
01600                *
01610                **********************************************
01620                *
01630 0176 0F              SEI               PREVENT SER TO PEND INTR
01640 0177 CE 4000         LDX    #$4000     PICK UP INTR CNT ADDR
```

```
017A FF E500          STX    DACZRO   DISABLE INTR FROM TIMER
017D B7 E400          STA A  ADCZRO   CLR MC6800 INT FLIP FLOP
[illegible]
[illegible] 7A 0227   DEC    CALCNT   [illegible]
0184 27 38            BEQ    SPDONE   IS COUNT 3? YES, GO SAVE DATA
0186 B6 0227          LDA A  CALCNT   NO. GET COUNT
0189 81 02            CMP A  #2       IS COUNT 2?
018B 26 22            BNE    PARSAV   NO. SAVE PARAMETER FROM INPUT
018D FE 022E          LDX    IRQCNT+2 YES. SAVE TIME LOOP COUNTER
0190 FF 022A          STX    CALONE
0193 CE 0396          LDX    #SAMPLE  NOW PUT SAMPLE ADDR IN IRQVEC
0196 FF FFF8          STX    IRQVEC
0199 86 AA            LDA A  #$AA     SET UP DONTST
019B B7 0232          STA A  DONTST
019E CE 0000          LDX    #0       CLR LOOP COUNT AND SAMPLE COU
01A1 FF 022C          STX    IRQCNT
01A4 FF 022E          STX    IRQCNT+2
01A7 FF 3494          STX    SAMPNO
01AA FF E500          STX    DACZRO   ENABLE TIMER INTERRUPTS
01AD 20 AD            BRA    SPOO1    GO WAIT FOR INTERRUPT
01AF FE 022C PARSAV   LDX    IRQCNT   SAVE LOOP COUNT
01B2 FF 3490          STX    LOOPCT
01B5 FE 022E          LDX    IRQCNT+2
01B8 FF 3492          STX    LOOPCT+2
01BB 7E 013A          JMP    TIMCAL   GO EXECUTE ANOTHER TIME CAL
01BE B6 022B SPDONE   LDA A  CALONE+1 AVERAGE TWO TIME CAL RUNS
01C1 BB 022F          ADD A  IRQCNT+3 ADD LSB BYTES
01C4 B7 0229          STA A  CALZRO+1 STORE IN BUFFER
01C7 B6 022A          LDA A  CALONE   PICK UP LSB OF FIRST
01CA B9 022E          ADC A  IRQCNT+2 ADD WITH CARRY
01CD B7 0228          STA A  CALZRO   STORE IN BUFFER
01D0 77 0228          ASR    CALZRO   DIVIDE BY 2 TO AVERAGE
01D3 76 0229          ROR    CALZRO+1
01D6 86 08            LDA A  #8       SET UP COUNTER FOR /256
01D8 77 0228 CALSHF   ASR    CALZRO   NOW DIVIDE BY 256 FOR
01DB 76 0229          ROR    CALZRO+1 LOOPS/INTERRUPT
01DE 4A               DEC A
01DF 26 F7            BNE    CALSHF   8 SHIFTS YET?
01E1 B6 0229          LDA A  CALZRO+1 PICK UP SHIFTED RESULT
01E4 B7 3496          STA A  LPCAL    STORE IN FILE PAR BUFFER
01E7 B6 34AF          LDA A  DTABIT+3 PUT DATA BIT COUNT INTO ACBLC
01EA B7 34FF          STA A  ACBLCT+3 SINCE NO COMPRESSION PERFORMED
01ED B6 34AE          LDA A  DTABIT+2
01F0 B7 34FE          STA A  ACBLCT+2
01F3 B6 34AD          LDA A  DTABIT+1
01F6 B7 34FD          STA A  ACBLCT+1
01F9 B6 34AC          LDA A  DTABIT
01FC B7 34FC          STA A  ACBLCT
01FF FE 1C9D          LDX    VECSAV   YES. RESTORE IRQ VECTOR
0202 FF FFF8          STX    IRQVEC
0205 CE 0312          LDX    #SAVMSG
0208 BD CA8F          JSR    OUTMCR   "SAMPLING COMPLETE..."
020B BD C9EC          JSR    KEYPD    GET SAVE DECISION
020E FE 3002          LDX    ENDBUF
```

```
02190 0211 09            DEX
02200 0212 F6 00         LDA B  0,X
02210 0214 C1 59         CMP B  #'Y       IS DIVISION YES?
02220 0216 26 0C         BNE    EXJMP     NO. EXIT TO EKG-EXEC
02230 0218 BD 0515       JSR    SAVEFL    YES. SAVE THE FILE ON DISK
02240 021B CE 0381       LDX    #SDONE
02250 021E BD CA8F       JSR    OUTSTR
02260 0221 BD CA2C       JSR    KEYDD
02270 0224 7E 1D00 EXJMP JMP    START     JUMP BACK TO EKG-EXEC
02280                    *
02290 0227 0001    CALCNT RMB    1         TEST CAL LOOP COUNTER
02300 0228 0002    CALBUF RMB    2         AVG'D CAL LOOPS DURING DAT CO
02310 022A 0002    CALONE RMB    2         TEMP BUF FOR PRE CALLBC CAL
02320 022C 0004    IRQCNT RMB    4         TEMP INTR LOOP COUNTER
02330 0230 0002    BUFPTR RMB    2         BUFFER POINTER OF NEXT AVAIL
02340 0232 0001    DONTST RMB    1         DONE TEST FLG FOR MEM FULL
02350                    *
02360 0233 1A07    STRMSG FDB    $1A07
02370 0235 54             FCC    /THIS MODULE SAMPLES THE EKG /
02380 0251 49             FCC    /INPUT AND STORES THE DATA WITH/
02390 026F 0D0A           FDB    $0D0A
02400 0271 4E             FCC    /NO DATA COMPRESSION./
02410 0285 0D0A           FDB    $0D0A,$0D0A
02420 0289 44             FCC    /DO YOU WISH TO EXECUTE THIS /
02430 02A5 4D             FCC    /MODULE (Y OR N)/
02440 02B4 04             FCB    4
02450 02B5 1A      IRQMSG FCB    $1A
02460 02D5 49             FCC    /INSURE SUBJECT AND EKG DEVICE /
02470 02D4 52             FCC    /READY!/
02480 02DA 0D0A           FDB    $0D0A,$0D0A,$0D0A
02490 02E0 50             FCC    /PRESS RETURN, THEN CLOSE INTERRUPT/
02500 0302 20             FCC    / ENABLE SWITCH./
02510 0311 04             FCB    4
02520 0312 1A07    SAVMSG FDB    $1A07
02530 0314 53             FCC    /SAMPLING COMPLETE.  PLEASE OPEN /
02540 0334 49             FCC    /INTERRUPT ENABLE SWITCH./
02550 034C 0D0A           FDB    $0D0A,$0D0A,$0D0A
02560 0352 44             FCC    /DO YOU WISH TO SAVE THIS DATA ON /
02570 0373 44             FCC    /DISK (Y OR N)/
02580 0380 04             FCB    4
02590 0381 20      SDONE  FCC    / PRESS RETURN/
02600 038E 04             FCB    4
02610                    *
02620                       ************************************************
02630                       ND NOCERS
02640                    ************************************************
02650                    *
02660                    *
02670                    *FUNCTION :SAMPLE
02680                    *INPUTS   :STATUS BUFFERS
02690                    *OUTPUTS  :COMPRESSED,ROUNDED DATA IN MEM BUFF
02700                    *CALLS    :NOTHING
02710                    *PURPOSE  :THIS ROUTINE SAMPLES THE EKG LEVEL,
02720                    * ROUNDS THE VALUES TO 3 BITS (FROM 12)
```

```
02730                  * CALCULATES MAX,MIN,# OF BITS, SAMPLES
02740                  * ETC., AND SAVES THESE PARAM AND DATA
02750                  * TO [illegible]
02760                  *
02770                  *
02780 038F 0001     SHFBUF RMB    1          TEMP SHIFT BUFFER
02790 0390 0001     TEMPA1 RMB    1          TEMP REG, MSB CH 0
02800 0391 0001     TEMPB1 RMB    1          TEMP REG, LSB CH 0
02810 0392 0001     TEMPA2 RMB    1          TEMP REG, MSB CH 1
02820 0393 0001     TEMPB2 RMB    1          TEMP REG, LSB CH 1
02830 0394 0001     TEMPA3 RMB    1          TEMP REG, MSB CH 2
02840 0395 0001     TEMPB3 RMB    1          TEMP REG, LSB CH 2
02850                  *
02860 0396 FE 3494 SAMPLE LDX    SAMPNO     GET CUR SAMPL COUNT
02870 0399 08             INX
02880 039A FF 3494        STX    SAMPNO
02890 039D BF 1C96        STS    STKSAV     SAVE STACK PRT
02900 03A0 8E 3280        LDS    #$3280     INITIALIZE STACK IN UNUSED MEM
02910 03A3 CE B400        LDX    #ADCZRO    NOW PULSE A/D TO START CONV
02920 03A6 A7 00          STA A  0,X        FOR CHANNEL 0
02930 03A8 01             NOP
02940 03A9 A6 00          LDA A  0,X
02950 03AB E6 01          LDA B  1,X
02960 03AD B7 0390        STA A  TEMPA1
02970 03B0 F7 0391        STA B  TEMPB1
02980 03B3 08             INX
02990 03B4 08             INX
03000 03B5 A7 00          STA A  0,X        NOW PULSE A/D FOR CONV
03010 03B7 01             NOP               ON CHANNEL 1
03020 03B8 A6 00          LDA A  0,X
03030 03BA E6 01          LDA B  1,X
03040 03BC B7 0392        STA A  TEMPA2
03050 03BF F7 0393        STA B  TEMPB2
03060 03C2 08             INX
03070 03C3 08             INX
03080 03C4 A7 00          STA A  0,X        NOW PULSE A/D FOR CONV
03090 03C6 01             NOP               ON CHANNEL 2
03100 03C7 A6 00          LDA A  0,X
03110 03C9 E6 01          LDA B  1,X
03120 03CB B7 0394        STA A  TEMPA3
03130 03CE F7 0395        STA B  TEMPB3
03140 03D1 CE 0390        LDX    #TEMPA1
03150 03D4 A6 00    SAMPL1 LDA A  0,X
03160 03D6 E6 01          LDA B  1,X
03170 03D8 47             ASR A             NOW SHIFT 2 BYTE VALUE [illegible]
03180 03D9 56             ROR B             4 POSITIONS TO RIGHT [illegible]
03190 03DA 47             ASR A             12 BIT TO 8 BIT ROUTINE CONV
03200 03DB 56             ROR B
03210 03DC 47             ASR A
03220 03DD 56             ROR B
03230 03DE 47             ASR A
03240 03DF 56             ROR B
03250 03E0 F7 038F        STA B  SHFBUF     SAVE 8 BIT RESULT OF SHIFT
03260 03E3 86 00          LDA A  #0         AND ADD CARRY OUT OF LAST
```

```
03270 03F5 B9 039F          ADC A  SHIFBUF   SHIFT ROUNDING UP OR DOWN
03280 03F8 36               PSH A           SAVE VLU IN STACK TEMPORARILY
[illegible]                 PSH
03300 03FA C6               PSH
03310 03FB 8C 0396          CPX    #TEMPA3+2  CHANNEL Z ROUNDED TO 8 BITS
03320 03FE 26 E4            BNE    SAMPL1   NO. GO SAMPLE NEXT CHANNEL
03330 03F0 32               PUL A           GET Z DATA
03340 03F1 FE 0230          LDX    BUFPTR   PICK UP CUR MEM FILE PTR
03350 03F4 A7 02            STA A  2,X      SAVE DATA TO MEM FILE
03360 03F6 FE 3494          LDX    SAMPNO   GET CUR SAMPLE COUNT
03370 03F9 B1 3497          CMP A  MAXZ     IS CUR Z MAX OVE SAMPLE SET?
03380 03FC 2F 06            BLE    SPZMIN   NO. GO CHECK FOR MIN
03390 03FE B7 3497          STA A  MAXZ     YES. KEEP CUR VLU
03400 0401 FF 3498          STX    MAXZLO   KEEP CUR SAMPLE NUM
03410 0404 B1 349A SPZMIN   CMP A  MINZ     IS CUR Z MIN OVER SAMPLE SET?
03420 0407 2C 06            BGE    SPYMAX   NO. GO CHECK FOR Y MAX
03430 0409 B7 349A          STA A  MINZ     YES. KEEP CUR VLU
03440 040C FF 349B          STX    MINZLO   KEEP CUR SAMPLE NUM
03450 040F 32       SPYMAX  PUL A           GET CUR Y DATA
03460 0410 FE 0230          LDX    BUFPTR   PICK UP CURRENT MEM FILE PTR
03470 0413 A7 01            STA A  1,X      SAVE DATA TO MEM FILE
03480 0415 FE 3494          LDX    SAMPNO   GET CUR SAMPLE COUNT
03490 0418 B1 349D          CMP A  MAXY     IS CUR Y MAX OVER SAMPLE SET
03500 041B 2F 06            BLE    SPYMIN   NO. GO CHEK FOR MIN
03510 041D B7 349D          STA A  MAXY     YES. KEEP CUR Y VLU
03520 0420 FF 349E          STX    MAXYLO   KEEP CUR SAMPLE NUM
03530 0423 B1 34A0 SPYMIN   CMP A  MINY     IS CUR Y MIN OVER SAMPLE SET?
03540 0426 2C 06            BGE    SPXMAX   NO. GO CHEK FOR X MAX
03550 0428 B7 34A0          STA A  MINY     YES. KEEP CUR Y VLU
03560 042B FF 34A1          STX    MINYLO   KEEP CUR SAMPLE NUM
03570 042E 32       SPXMAX  PUL A           GET CUR X DATA
03580 042F FE 0230          LDX    BUFPTR   PICK UP CURRENT MEM FILE PTR
03590 0432 A7 00            STA A  0,X      SAVE DATA TO MEM FILE
03600 0434 FE 3494          LDX    SAMPNO   GET CUR SAMPLE COUNT
03610 0437 B1 34A3          CMP A  MAXX     IS CUR X MAX OVER SAMPLE SET?
03620 043A 2F 06            BLE    SPXMIN   NO. GO CHECK X MIN
03630 043C B7 34A3          STA A  MAXX     YES. KEEP CUR X VLU
03640 043F FF 34A4          STX    MAXXLO   KEEP CUR SAMPLE VLU
03650 0442 B1 34A6 SPXMIN   CMP A  MINX     IS CUR X MIN OVER SAMPLE SET?
03660 0445 2C 06            BGE    MMDONE   NO. EXIT MAX,MIN UPDATE
03670 0447 B7 34A6          STA A  MINX     YES. KEEP CUR X VLU
03680 044A FF 34A7          STX    MINXLO   KEEP CUR SAMPLE COUNT
03690 044D 0C       MMDONE  CLC             DONE WITH MAX & MINS. NOW
03700 044E 86 18            LDA A  #24      UPDATE ACTUAL DATA BIT STORED
03710 0450 C6 00            LDA B  #0
03720 0452 BB 34AF          ADD A  DTABIT+3
03730 0455 B7 34AF          STA A  DTABIT+3
03740 0458 86 00            LDA A  #0
03750 045A F9 34AE          ADC B  DTABIT+2
03760 045D F7 34AE          STA B  DTABIT+2
03770 0460 C6 00            LDA B  #0
03780 0462 B9 34AD          ADC A  DTABIT+1
03790 0465 B7 34AD          STA A  DTABIT+1
03800 0468 F9 34AC          ADC B  DTABIT
```

```
03810 046B F7 34AC          STA B  DTABIT
03820 046E 86 08            LDA A  #8          NOW UPDATE XBIT,YBIT,ZBIT COU
[illegible]
05840 0472 BB 34B2          ADD A  XBITS+2     X,Y,Z COUNTER BITS.
03850 0475 B7 34B2          STA A  XBITS+2
03860 0478 B7 34B5          STA A  YBITS+2
03870 047B B7 34B8          STA A  ZBITS+2
03880 047E 86 00            LDA A  #0
03890 0480 F9 34B1          ADC B  XBITS+1
03900 0483 F7 34B1          STA B  XBITS+1
03910 0486 F7 34B4          STA B  YBITS+1
03920 0489 F7 34B7          STA B  ZBITS+1
03930 048C B9 34B0          ADC A  XBITS
03940 048F B7 34B0          STA A  XBITS
03950 0492 B7 34B3          STA A  YBITS
03960 0495 B7 34B6          STA A  ZBITS
03970 0498 7F 34BB          CLR    TBITS+2
03980 049B 7F 34BA          CLR    TBITS+1
03990 049E 7F 34B9          CLR    TBITS
04000 04A1 34               DES                NOW POINT STACK BACK TO DATA
04010 04A2 34               DES
04020 04A3 34               DES
04030 04A4 32               PUL A              GET Z DATA
04040 04A5 CE 3A00          LDX    #ZPDFM1
04050 04A8 BD 04D2          JSR    PDFSTR      UPDATE Z PDF BIN COUNTER
04060 04AB 32               PUL A              GET Y DATA
04070 04AC CE 3800          LDX    #YPDFM
04080 04AF BD 04D2          JSR    PDFSTR      UPDATE Y PDF BIN CONTER
04090 04B2 32               PUL A
04100 04B3 CE 3600          LDX    #XPDFM
04110 04B6 BD 04D2          JSR    PDFSTR
04120 04B9 FE 0230          LDX    BUFPTR      NOW GET & UPDATE MEM BUFF PTR
04130 04BC 08               INX
04140 04BD 08               INX
04150 04BE 08               INX
04160 04BF FF 0230          STX    BUFPTR
04170 04C2 BE 1C96          LDS    STKSAV      RETRIEV ENTRY STACK POINTER
04180 04C5 8C 7FFE          CPX    #$7FFE      IS MEM BUF FULL?
04190 04C8 26 03            BNE    SAMRTI      NO. RET FROM INTR & SAMPLE AG
04200 04CA 7F 0232          CLR    DONTST      YES. RESET MEM FULL FLAG
04210 04CD 3B       SAMRTI  RTI
04220                       *
04230 04CE 0002     TEMPDF  RMB    2           TEMP WORKING BUFFER
04240 04D0 0002     TEMPST  RMB    2           TEMP STACK SAVE BUFFER
04250                       *
04260 04D2 BF 04D0  PDFSTR  STS    TEMPST      SAVE STACK PRT
04270 04D5 FF 04CE          STX    TEMPDF      SAVE PDF PTR
04280 04D8 16               TAB                SAVE INPUT DATA VLU
04290 04D9 BB 04CF          ADD A  TEMPDF+1    NOW CALCULATE ADDR [illegible]
04300 04DC B7 04CF          STA A  TEMPDF+1    IN THE PDF MEM BUFF [illegible]
04310 04DF 86 00            LDA A  #0
04320 04E1 C5 80            BIT B  #$80        IS THE DATA VLU NEG?
04330 04E3 2B 05            BMI    PDFST1      YES. SBC MSB VS ADC
04340 04E5 B9 04CE          ADC A  TEMPDF      NO. ADD WITH CARRY MSB
```

```
04350 04E8 20 03          BRA    PDFST2   AND STORE
04360 04EA B2 04CE PDFST1 SBC A  TEMPDF   VLU IS NEG. SBC
[illegible]
04380 04F0 17             TBA             [illegible]
04390 04F1 BB 04CF        ADD A  TEMPDF+1 FOR PROPER ADDR CALCULATION
04400 04F4 B7 04CF        STA A  TEMPDF+1
04410 04F7 86 00          LDA A  #0
04420 04F9 C5 80          BIT B  #$80     IS THE DATA NEG?
04430 04FB 2B 05          BMI    PDFST3   YES. SBC MSB VS ADC
04440 04FD B9 04CE        ADC A  TEMPDF   NO. ADD WITH CARRY MSB BYTE
04450 0500 20 03          BRA    PDFST4   AND STORE
04460 0502 B2 04CE PDFST3 SBC A  TEMPDF   VLU IS NEG. SUB WITH CARRY
04470 0505 B7 04CE PDFST4 STA A  TEMPDF   AND STORE
04480 0508 FE 04CE        LDX    TEMPDF   NOW LOAD CALC ADDR FOR INDEX
04490 050B AE 00          LDS    0,X      GRAB VLU IN CALC ADDR
04500 050D 31             INS             INCREMENT IT
04510 050E AF 00          STS    0,X      AND STORE IT BACK IN BUFFER
04520 0510 BE 04D0        LDS    TEMPST   RECOVER STACK POINTER
04530 0513 17             TBA             RECOVER ORIGINAL DATA
04540 0514 39             RTS
04550              *
04560              *FUNC: EXEC JUMPS
04570              *INPUTS: ACCUMULATORS
04580              *OUTPUTS: NONE
04590              *CALLS: SAVEFL,FILHDR (EKG-EXEC) VIA ADDR BUFFERS
04600              *DESTROYS: X,A,B,CC
04610              *PURPOSE: TO JUMP TO DESIRED ROUTINES VIA RELOC ADDR
04620              *
04630 0515 FE 1CA3 SAVEFL LDX    SAVELC   GET ADDR OF SAVEFL
04640 0518 6E 00          JMP    0,X      JUMP TO SAVEFL
04650              *
04660 051A FE 1CA1 FILHDR LDX    FILHLC   GET ADDR OF FILHLC
04670 051D 6E 00          JMP    0,X      JUMP TO SAVEFL
04680              *
04690              ************************************************
04700              * END SAMPLE
04710              ************************************************
04720              *
04730              *
04740              *FUNCTION :CALINT
04750              *INPUTS (REG)  :NONE
04760              *OUTPUTS (REG) :NONE
04770              *CALLS  :NOTHING
04780              *DESTROYS (REG):NONE (INTERRUPT HANDLER)
04790              *PURPOSE :THIS ROUTINE IS USED FOR CALIBRATING
04800              * THE MAX NUMBER OF LOOPS POSSIBLE
04810              * DURING AN INTERRUPT PERIOD.  THIS
04820              * ROUTINE UPDATES THE DONTST FLAG AND
04830              * RESETS THE ST6800 INT FLIP FLOP &
04840              * THE RETURNS.
04850              *
04860 051F 7C 0232 CALINT INC    DONTST   INCREMENT DONE TEST FLAG
04870 0522 B7 E400        STA A  ADCZRO   CLR ST6800 INTR FLIP FLOP
04880 0525 01             NOP
```

```
04890 0526 3B           RTI
04900                   *
                        ***************************************************
04920                   *  END CALINT
04930                   ***************************************************
04940                   *
04950                   *
04960                   ***************************************************
04970                   *
04980                   *  END OF NOCIPRS OVERLAY ROUTINES
04990                   *
05000                   ***************************************************
05010                   *
05020                           END
```

```
00030                ***************************************************
                     *
                     * OVERLAY NAME: TOLAN-A
00060                * AUTHOR : CAPT. NEL TOLNCHID
00070                * VERSION : 1.8
00080                * VERSION DATE : 22 OCT 80
00090                *
00100                ***************************************************
00110                *
00120                * OVERLAY DESCRIPTION
00130                *
00140                * THIS OVERLAY SAMPLES THE EKG DATA VIA THE
00150                * A/D CONVERTERS, ROUNDS THE DATA TO 8 BITS,
00160                * AND THEN COMPRESSES THE DATA VIA THE TOLAN-A
00170                * ALGORITHM. THE COMPRESSED DATA IS THEN STORED INTO
00180                * MEMORY DATA FILE FROM 3C00-7FFF.
00190                *
00200                ***************************************************
00210                *
00220                * START OF TOLAN-A
00230                *
00240                ***************************************************
00250                *
00260 0100                 ORG     $0100     OVERLAY START ADDRESS
00270                *
00280                      OPT     O         ASSB OPT-GEN OBJ FILE
00290                      OPT     NOG       ASSB OPT-SUPPRESS FULL PCC LI
00300                *
00310                ***************************************************
00320                *
00330                * LABLE DECLARATIONS
00340                *
00350                * SUPPORT SUBROUTINE ADDRESSES
00360                *
00370      CA87      OUTPUT EQU     $CA87     EPROMDOS. ALPH STRING TO CONS
00380      CA8F      OUTNCR EQU     $CA8F     EPROMDOS. ALPH STRING,NO CRLF
00390      CA2C      KEYRD  EQU     $CA2C     EPROMDOS. CONSOLE INPUT ROUTI
00400      CA36      KEYBD0 EQU     $CA36     EPROMDOS. CONSOLE INPUT. NO ?
00410      1D00      START  EQU     $1D00     EKG-EXEC. FILE CREATE ROUTINE
00420                *
00430                * DATA BUFFERS
00440                *
00450      3400      HDRSTR EQU     $3400     DATA MEMORY BUFFER HEADER
00460      FFF8      IRQVEC EQU     $FFF8     INTERRUPT VECTOR ADDR
00470      1C96      STKSAV EQU     $1C96     STACK SAVE BUFFER
00480      1C98      CPRTYP EQU     $1C98     COMPRESSION ALGOR FLAG
00490      3002      ENDBUF EQU     $3002     ADDR OF LAST CHAR INPUT
00500      1C9D      VECSAV EQU     $1C9D     IRQ VECTOR SAVE BUFFER
00510      1CA1      FILHDC EQU     $1CA1     FILEHDR ADDR DATA BUFFER
00520      1CA3      SAVHDC EQU     $1CA3     SAVFIL ADDR DATA BUFFER
00530      3490      LOOPCT EQU     $3490     TOTAL NUM TIMES LOOP EXEC'D
00540      3494      SAMPNO EQU     $3494     NUMBER OF SAMPLES TAKEN
00550      3496      LPCML  EQU     $3496     AVG'D MAX LOOP COUNT/INTR
00560      3497      MAXZ   EQU     $3497     MAX VLU IN CH Z
```

```
00[illegible]  3498   MAXZLO EQU   $3498   MAX VLU LOC IN CH Z
00[illegible]  349A   MINZ   EQU   $349A   MIN VLU IN CH Z
[illegible]    349B   MINZLO EQU   $349B   MIN VLU LOC IN CH Z
[illegible]    349D   MAXY   EQU   $349D   MAX VLU IN CH Y
00610          349E   MAXYLO EQU   $349E   MAX VLU LOC IN CH Y
00620          34A0   MINY   EQU   $34A0   MIN VLU IN CH Y
00630          34A1   MINYLO EQU   $34A1   MIN VLU LOC IN CH Y
00640          34A3   MAXX   EQU   $34A3   MAX VLU IN CH X
00650          34A4   MAXXLO EQU   $34A4   MAX VLUE LOC IN CH X
00660          34A6   MINX   EQU   $34A6   MIN VLU IN CH X
00670          34A7   MINXLO EQU   $34A7   MIN VLU LOC IN CH X
00680          34AC   DTABIT EQU   $34AC   NUM OF BITS USED TO STR DTA
00690          34B0   XBITS  EQU   $34B0   NUM OF BITS USED TO STR X
00700          34B3   YBITS  EQU   $34B3   NUM OF BITS USED TO STR Y
00710          34B6   ZBITS  EQU   $34B6   NUM OF BITS USED TO STR Z
00720          34B9   TBITS  EQU   $34B9   NUM OF BITS USED TO STR T
00730          34BC   ACELCT EQU   $34BC   # BITS FED TO VAR LEN CODE
00740          3600   XPDFM  EQU   $3600   0 VLU LOC OF X PDF
00750          3800   YPDFM  EQU   $3800   0 VLU LOC OF Y PDF
00760          3A00   ZPDFM  EQU   $3A00   0 VLU LOC OF Z PDF
00770          3B00   TPDF   EQU   $3B00   0 VAL LOC OF TIME VAR HIST
00780          3C00   SECZRO EQU   $3C00   DATA STORE ADDR START
00790                 *
00800                 * HARDWARE ADDRESSES
00810                 *
00820          E400   ADCZRO EQU   $E400   ADC CHANNEL ZERO
00830          E404   ADCTWO EQU   $E404   ADC CHANNEL TWO
00840          E500   DACZRO EQU   $E500   DAC CHANNEL ZERO
00850                 *
00860                 *
00870                 *FUNCTION :TOLAN-A
00880                 *INPUTS :STATUS BUFFERS FROM EKG-EXEC
00890                 *OUTPUTS :DATA TO DISK
00900                 *CALLS :OUTPUT,FILHDR,KEYBD,OUTNCR
00910                 * KEYBD0,SAVEFL,START
00920                 *DESTROYS :ALL REGISTERS
00930                 *PURPOSE :TO COLLECT 3 CHANNELS OF EKG
00940                 * DATA AND STORE INTO MEMORY.
00950                 *
00960                 *
00970 0100 0F         TOLANA SEI
00980 0101 CE 0252           LDX    #STRMSG
00990 0104 BD CA8F           JSR    OUTNCR   "THIS MODULE SAMPLES EKG..."
01000 0107 BD CA2C           JSR    KEYBD    GET RESPONSE FROM CONSOLE
01010 010A FE 3002           LDX    ENDBUF
01020 010D 09                DEX
01030 010E E6 00             LDA B  0,X
01040 0110 C1 59             CMP B  #'Y      IS INPUT YES?
01050 0112 27 03             BEQ    TOLAN1   YES. KEEP EXECUTING THIS ROUT
01060 0114 7E 1D00           JMP    START    NO. BIN TO EKG-EXEC
01070 0117 CE 5441 TOLAN1    LDX    #$5441   FLAG COMPRESSION TYPE (TA)
01080 011A FF 1C98           STX    CMPTYP
01090 011D BD 023B           JSR    FILHDR   SET UP DATA FILE HEADER
01100 0120 CE 02D9           LDX    #IRQMSG
```

```
01110 0123 BD CA8F        JSR    OUTNCR   "INSURE SUBSEC" AND REG..."
01120 0126 BD CA36        JSR    KEYPD0
0113  0129 CE 3000 TCLAGO LDX    #$3000   INITIALIZE BUF PTR AT $3000
0114  012C FF 024E        STX    BUFPTR   STORE IN BUFFER PNTR LOC
01150 012F 86 04          LDA A  #4       PICK UP COUNTER FOR TIME CAL
01160 0131 B7 0245        STA A  CALCNT   STORE IN BUFFER
01170 0134 86 80          LDA A  #$80     INITIALIZE BIT POINTER TO LEF
01180 0136 B7 03C7        STA A  BITPTR
01190 0139 4F             CLR A          INIT COMPRESS VAR
01200 013A B7 0251        STA A  FIRST    CLEAR FIRST SAMPLE FLAG
01210 013D B7 03BC        STA A  XSLOPE
01220 0140 B7 03BD        STA A  YSLOPE
01230 0143 B7 03BE        STA A  ZSLOPE
01240 0146 FE FFF8        LDX    IRQVEC   PICK UP CURREN IRQ VECTOR
01250 0149 FF 1C9D        STX    VECSAV   SAVE IN BUFFER
01260 014C CE 06AE TIMCAL LDX    #CALINT  GET INTR VECTOR ADDR FOR CAL
01270 014F FF FFF8        STX    IRQVEC   PUT IN VECTOR ADDRESS
01280 0152 86 00          LDA A  #0       INIT COUNTER FOR 256 TEST LOO
01290 0154 B7 0250        STA A  DONTST   STORE IN DONTST BUFFER
01300 0157 CE 0000        LDX    #0       CLEAR LOOP COUNTER
01310 015A FF 024A        STX    IRQCNT
01320 015D FF 024C        STX    IRQCNT+2
01330 0160 FF E500        STX    DACZRO   PULSE INT ENABLE CIRCUIT
01340 0163 20 09          BRA    SPOO1    BRA TO COUNTING LOOP
01350              *
01360              ********************************************
01370              *
01380              * BASIC TIMING LOOP FOR EFFICIENCY TEST
01390              *
01400              ********************************************
01410              *
01420 0165 01      SPLOOP NOP              DELAY TO MATCH TIME
01430 0166 01             NOP              IT TAKES TO EXECUTE THE
01440 0167 01             NOP              INCREMENT OF BYTES 3&4
01450 0168 01             NOP              OF THE LOOP COUNTER
01460 0169 01             NOP              WHEN COUNT CARRIES TO
01470 016A 01             NOP              HIGH 2 BYTES OF 4 BYTE
01480 016B 7E 0181        JMP    SPOO3     JUMP TO CONTINUE LOOP
01490 016E 0E      SPOO1  CLI              PREPARE FOR TIMER INT
01500 016F 3E             WAI              STOP PROCESSOR & WAIT
01510 0170 0E      SPOO4  CLI
01520 0171 FE 024C        LDX    IRQCNT+2 RET'ED FROM INT, INC COUNT
01530 0174 08             INX
01540 0175 FF 024C        STX    IRQCNT+2 SAVE IT
01550 0178 26 E8          BNE    SPLOOP   COUNT COME FFFF TO 0000?
01560 017A FE 024A        LDX    IRQCNT   YES, INC BYTES 3&4
01570 017D 08             INX
01580 017E FF 024A        STX    IRQCNT   SAVE COUNT
01590 0181 B6 0250 SPOO3  LDA A  DONTST   IS DONE TEST SATISFIED?
01600 0184 81 00          CMP A  #0
01610 0186 26 E8          BNE    SPOO4    NO, KEEP LOOPING
01620              *
01630              ********************************************
01640              *
```

```
01650                * END BASIC TIMING LOOP FOR EFT TEST
01660                *
                     ********************************************************
01680                *
01690 0188 0F              SEI              PREVENT SER TO MORE INTR
01700 0189 CE 4000         LDX    #$4000    PICK UP INTR OFF WORD
01710 018C FF E500         STX    DACZRO    DISABLE INTR FROM TIMER
01720 018F B7 E400         STA A  ADCZRO    CLR ST6800 INT FLIP FLOP
01730 0192 01              NOP
01740 0193 7A 0245         DEC    CALCNT    DEC TIME CAL LOOP COUNTER
01750 0196 27 4F           BEQ    SPDONE    IS COUNT 3? YES, GO SAVE DATA
01760 0198 B6 0245         LDA A  CALCNT    NO. GET COUNT
01770 019B 81 03           CMP A  #3        IS COUNT 3?
01780 019D 26 25           BNE    MAINSP    NO. GO TO MAIN SPL LOOP
01790 019F FE 024C         LDX    IRQCNT+2  YES. SAVE TIME LOOP COUNTER
01800 01A2 FF 0248         STX    CALONE
01810 01A5 CE 03C8         LDX    #SAMPLE   NOW PUT SAMPLE ADDR IN IRQVEC
01820 01A8 FF FFF8         STX    IRQVEC
01830 01AB 86 00           LDA A  #$0       SET UP DONTST
01840 01AD B7 0250         STA A  DONTST
01850 01B0 7F 0251         CLR    FIRST     RESET FIRST SMP FLG
01860 01B3 CE 0000         LDX    #0        CLR LOOP COUNT AND SAMPLE COU
01870 01B6 FF 024A         STX    IRQCNT
01880 01B9 FF 024C         STX    IRQCNT+2
01890 01BC FF 3494         STX    SAMPNO
01900 01BF FF E500         STX    DACZRO    ENABLE TIMER INTERRUPTS
01910 01C2 20 AA           BRA    SPOOL     GO WAIT FOR INTERRUPT
01920 01C4 81 02   MAINSP  CMP A  #2        IS COUNT 2?
01930 01C6 26 10           BNE    PARSAV    NO. SAVE PARAMETER FROM ERG
01940 01C8 86 55           LDA A  #$55      SET FIRST FLAG
01950 01CA B7 0251         STA A  FIRST
01960 01CD B7 0250         STA A  DONTST    SET DONTST FOR SMPL COLLEC
01970 01D0 CE 0000         LDX    #0        ENABLE INTR CLOCK
01980 01D3 FF E500         STX    DACZRO
01990 01D6 20 96           BRA    SPOOL     GO WAIT FOR INTERRUPT
02000 01D8 FE 024A PARSAV  LDX    IRQCNT    SAVE LOOP COUNT
02010 01DB FF 3490         STX    LOOPCT
02020 01DE FE 024C         LDX    IRQCNT+2
02030 01E1 FF 3492         STX    LOOPCT+2
02040 01E4 7E 017C         JMP    TIMCAL    GO EXECUTE ANOTHER TIME CAL
02050 01E7 B6 0249 SPDONE  LDA A  CALONE+1  AVERAGE TWO TIME CAL RUNS
02060 01EA BB 024D         ADD A  IRQCNT+3  ADD LSB BYTES
02070 01ED B7 0247         STA A  CALZRO+1  STORE IN BUFFER
02080 01F0 B6 0248         LDA A  CALONE    PICK UP MSB OF FIRST
02090 01F3 B9 024C         ADC A  IRQCNT+2  ADD WITH CARRY
02100 01F6 B7 0246         STA A  CALZRO    STORE IN BUFFER
02110 01F9 77 0246         ASR    CALZRO    DIVIDE BY 2 TO AVERAGE
02120 01FC 76 0247         ROR    CALZRO+1
02130 01FF 86 08           LDA A  #8        SET UP COUNTER FOR 256
02140 0201 77 0246 CALSHF  ASR    CALZRO    NOW DIVIDE BY 256 TO
02150 0204 76 0247         ROR    CALZRO+1  LOOPS/INTERRUPT
02160 0207 4A              DEC A
02170 0208 26 F7           BNE    CALSHF    8 SHIFTS YET?
02180 020A B6 0247         LDA A  CALZRO+1  PICK UP SHIFTED RESULT
```

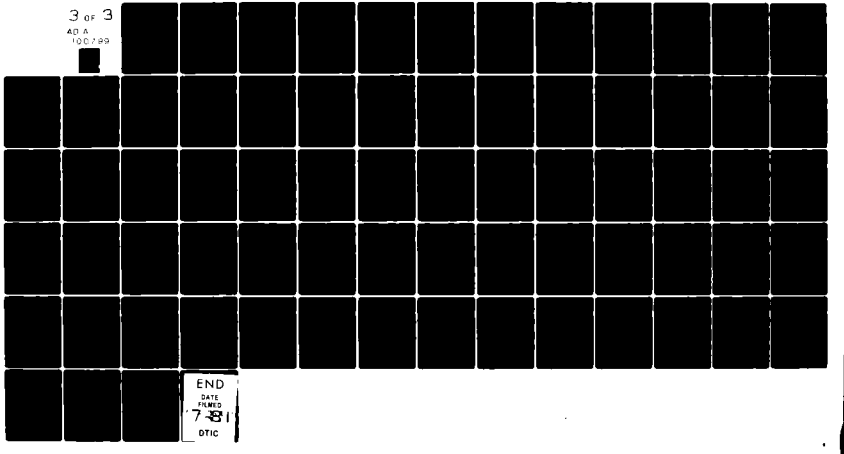
```
02190 020D B7 3496        STA A  LPCAL    STORE IN FILE HDR POINTER
02200 0210 BD 059E        JSR    DTACNT   NOW COUNT # OF DATA BYTES STOR
[illegible]               [illegible]
02220 0216 FF FFF8        STX    INQADR
02230 0219 CE 0336        LDX    #SAVMSG
02240 021C BD CA8F        JSR    OUTNCR   "SAMPLING COMPLETE..."
02250 021F BD CA2C        JSR    KEYBD    GET SAVE DECISION
02260 0222 FE 3002        LDX    ENDBUF
02270 0225 09             DEX
02280 0226 E6 00          LDA B  0,X
02290 0228 C1 59          CMP B  #'Y      IS DECISION YES?
02300 022A 26 0C          BNE    EXJMP    NO. RTN TO EKG-EXEC
02310 022C BD 0240        JSR    SAVEFL   YES. SAVE THE FILE ON DISK
02320 022F CE 03A4        LDX    #SDONE
02330 0232 BD CA8F        JSR    OUTNCR
02340 0235 BD CA2C        JSR    KEYBD
02350 0238 7E 1D00 EXJMP  JMP    START    JUMP BACK TO EKG-EXEC
02360                     *
02370                     *FUNC: RELOC JUMP
02380                     *INPUTS: A
02390                     *OUTPUTS: NONE
02400                     *CALLS: FILHDR,SAVEFL (EKG-EXEC ROUTINES)
02410                     *DESTROYS: A,B,CC,X
02420                     *PURPOSE : THIS ROUTINE ENABLES TOLAN-A TO BE RELOCA
02430                     *WITHOUT WORRY OF CHANGING CALLS TO EKG-EXEC.
02440                     *
02450 023E FE 1CA1 FILHDR LDX    FILHLC   GET FILHDR ADDRESS
02460 023E 6E 00          JMP    0,X      JUMP TO IT
02470                     *
02480 0240 FE 1CA3 SAVEFL LDX    SAVELC   GET SAVEFL ADDRESS
02490 0243 6E 00          JMP    0,X      JUMP TO IT
02500                     *
02510 0245 0001    CALCNT RMB    1        TEST CAL LOOP COUNTER
02520 0246 0002    CALZRO RMB    2        AVG'D CAL LOOPS DURING DAT CO
02530 0248 0002    CALCUE RMB    2        TEMP BUF FOR PRE COLLEC CAL
02540 024A 0004    INQCNT RMB    4        TEMP INTR LOOP COUNTER
02550 024E 0002    BUFPTR RMB    2        BUFFER POINTER OF NEXT AVAIL
02560 0250 0001    DONTST RMB    1        DONE TEST FLG FOR MEM FULL
02570 0251 0001    FIRST  RMB    1        FIRST A/D SAMPL FLG
02580                     *
02590 0252 1A07    STRMSG FDB    $1A07
02600 0254 54             FCC    /THIS MODULE SAMPLES THE EKG /
02610 0270 49             FCC    /INPUT AND STORES THE DATA /
02620 028A 0D0A           FDB    $0D0A
02630 028C 43             FCC    /COMPRESSED VIA MAJOR TOLAN-A./
02640 02A9 0D0A           FDB    $0D0A,$0D0A
02650 02AD 44             FCC    /DO YOU WISH TO EXECUTE THIS
02660 02C9 4D             FCC    /MODULE (Y OR N)/
02670 02D8 04             FCB    4
02680 02D9 1A      IRQMSG FCB    $1A
02690 02DA 49             FCC    /INSURE SUBJECT AND EKG DEVICE
02700 02F8 52             FCC    /READY!/
02710 02FE 0D0A           FDB    $0D0A,$0D0A,$0D0A
02720 0304 50             FCC    /PRESS RETURN, THEN CLOSE DOOR/
```

```
02730 0326 20              FCC    / ENABLE SWITCH./
02740 0335 04              FCB    4
02750 0336 1A07    SAMPING FDB    $1A07
02760 0338 53              FCC    /SAMPLING CONTINUE. ENABLE SWITCH./
02770 0357 .9              FCC    /INTERRUPT ENABLE SWITCH./
02780 036F 0D0A            FDB    $0D0A,$0D0A,$0D0A
02790 0375 44              FCC    /DO YOU WISH TO SAVE THIS DATA ON /
02800 0396 44              FCC    /DISK (Y OR N)/
02810 03A3 04              FCB    4
02820 03A4 20      SDONE   FCC    / PRESS RETURN/
02830 03B1 04              FCB    4
02840                      *
02850                      *
02860                      * END TOLAN-A
02870                      *
02880                      *
02890                      *FUNCTION :SAMPLE
02900                      *INPUTS :STATUS BUFFERS
02910                      *OUTPUTS :COMPRESSED,ROUNDED DATA IN MEM BUFF
02920                      *CALLS :NOTHING
02930                      *PURPOSE :THIS ROUTINE SAMPLES THE EKG LEADS,
02940                      * ROUNDS THE VALUES TO 8 BITS (PROC 12)
02950                      * CALCULATES MAX,MIN,# OF BITS, SAMPLES
02960                      * ETC., AND SAVES THESE PARAM AND DATA
02970                      * TO DATA MEM FILE.
02980                      *
02990                      *
03000 03B2 0001    SHFBUF  RMB    1          TEMP SHIFT BUFFER
03010 03B3 0001    TEMPA1  RMB    1          TEMP REG, MSB CH 0
03020 03B4 0001    TEMPB1  RMB    1          TEMP REG, LSB CH 0
03030 03B5 0001    TEMPA2  RMB    1          TEMP REG, MSB CH 1
03040 03B6 0001    TEMPB2  RMB    1          TEMP REG, LSB CH 1
03050 03B7 0001    TEMPA3  RMB    1          TEMP REG, MSB CH 2
03060 03B8 0001    TEMPB3  RMB    1          TEMP REG, LSB CH 2
03070 03B9 0001    XDATA   RMB    1          TEMP X CH DTA BUF
03080 03BA 0001    YDATA   RMB    1          TEMP Y CH DTA BUF
03090 03BB 0001    ZDATA   RMB    1          TEMP Z CH DTA BUF
03100 03BC 0001    XSLOPE  RMB    1          1ST DIFF X(N)-X(N+1)
03110 03BD 0001    YSLOPE  RMB    1          1ST DIFF Y(N)-Y(N+1)
03120 03BE 0001    ZSLOPE  RMB    1          1ST DIFF Z(N)-Z(N+1)
03130 03BF 0001    XEXP    RMB    1          EXPECTED VAL OF X(N+1)
03140 03C0 0001    YEXP    RMB    1          EXPECTED VAL OF Y(N+1)
03150 03C1 0001    ZEXP    RMB    1          EXPECTED VAL OF Z(N+1)
03160 03C2 0001    XACCEL  RMB    1          DIFF X(N+1)-E(X(N+1))
03170 03C3 0001    YACCEL  RMB    1          DIFF Y(N+1)-E(Y(N+1))
03180 03C4 0001    ZACCEL  RMB    1          DIFF Z(N+1)-E(Z(N+1))
03190 03C5 0001    SIPFLG  RMB    1          I HAVE JUST SAMPLED N-1
03200 03C6 0001    ICNT    RMB    1          TIME CNT VAR (<127)
03210 03C7 0001    BITPTR  RMB    1          BIT POINTER FOR SHIFT OPERATIONS
03220                      *
03230 03C8 FE 3494 SAMPLE  LDX    SAMPNO     GET CUR SAMPL COUNT
03240 03CB 08              INX
03250 03CC FF 3494         STX    SAMPNO
03260 03CF 3F 1C96         STS    STKSAV     SAVE STACK PNT
```

```
03270 03D2 8E 3280          LDS    #$3280   INITIALIZE STACK IN UNUSED MEM
              B7 0251       LDA A  FIRST    CK FIRST FLG
(CL                         (CONT           IF NO, CONT)
03300 03DA BD 05C8          JSR    SAMPL0   YES. SMPL & RTI
03310 03DD FE 024E          LDX    BUFPTR   NOW STR FIRST DATA VALUES
03320 03E0 B6 03F9          LDA A  XDATA
03330 03E3 A7 00            STA A  0,X
03340 03E5 08               INX
03350 03E6 B6 03FA          LDA A  YDATA
03360 03E9 A7 00            STA A  0,X
03370 03EB 08               INX
03380 03EC B6 03FB          LDA A  ZDATA
03390 03EF A7 00            STA A  0,X
03400 03F1 08               INX
03410 03F2 FF 024E          STX    BUFPTR   UPDATE NEW BUFPTR
03420 03F5 86 55            LDA A  #$55     SET STRFLG & FIRST FLAG
03430 03F7 B7 03C5          STA A  STRFLG
03440 03FA B7 0251          STA A  FIRST
03450 03FD 86 08            LDA A  #8       UPDATE X,Y,Z BIT CNTRS FOR IN
03460 03FF B7 34E2          STA A  XBITS+2
03470 0402 B7 34E5          STA A  YBITS+2
03480 0405 B7 34E8          STA A  ZBITS+2
03490 0408 7F 0250          CLR    DONTST   CLEAR DONTST FLAG FOR 1 SMPL
03500 040B BE 1C96          LDS    STKSAV
03510 040E 3B               RTI
03520 040F 7D 03C5 COMPRS   TST    STRFLG   HAS DATA JUST BEEN STORED ?
03530 0412 27 05            BEQ    COMPR1   NO. KEEP COUNTING TIME
03540 0414 86 01            LDA A  #1       YES. SET TIME COUNT TO 1
03550 0416 B7 03C6          STA A  ICNT
03560 0419 B6 03F9 COMPR1   LDA A  XDATA    GET X VLU
03570 041C BB 03FC          ADD A  XSLOPE   ADD {X(N-1)-X(N-2)}
03580 041F B7 03FF          STA A  XEXP     CREATE EXPECTED VAL OF X(N+1)
03590 0422 B6 03FA          LDA A  YDATA    GET LAST Y SMPL VALUE
03600 0425 BB 03FD          ADD A  YSLOPE   ADD {(Y(N-1)-Y(N-2)}
03610 0428 B7 03C0          STA A  YEXP     CREATE EXPECTED VAL OF Y(N+1)
03620 042B B6 03FB          LDA A  ZDATA    GET LAST Z SMPL VALUE
03630 042E BB 03FE          ADD A  ZSLOPE   ADD {(Z(N-1)-Z(N-2)}
03640 0431 B7 03C1          STA A  ZEXP     CREATE EXPECTED VAL OF Z(N+1)
03650 0434 BD 05C8          JSR    SAMPL0   NOW GO GET NEXT SMPL. N=N+1
03660 0437 B6 03F9          LDA A  XDATA    CALC DIF X(N)-E{X(N)}
03670 043A B0 03FF          SUB A  XEXP
03680 043D B7 03C2          STA A  XACCEL   SAVE DIF IN VAR
03690 0440 B6 03FA          LDA A  YDATA    CALC DIF Y(N)-E{Y(N)}
03700 0443 B0 03C0          SUB A  YEXP
03710 0446 B7 03C3          STA A  YACCEL   SAVE DIF IN VAR
03720 0449 B6 03FB          LDA A  ZDATA    CALC DIF Z(N)-E{Z(N)}
03730 044C B0 03C1          SUB A  ZEXP
03740 044F B7 03C4          STA A  ZACCEL   SAVE DIF IN VAR
03750 0452 B6 03C2          LDA A  XACCEL
03760 0455 26 21            BNE    COMPR2   ARE ALL INTERPOLATIONS GOOD?
03770 0457 B6 03C3          LDA A  YACCEL   IF NO, NEED TO STR ALL DIFF'S
03780 045A 26 1C            BNE    COMPR2
03790 045C B6 03C2          LDA A  XACCEL
03800 045F 26 17            BNE    COMPR2
```

```
03810 0461 B6 03C6          LDA A  ICNT     UPDATE RUN TIME COUNTER
03820 0464 4C               INC A
[illegible]
03840 0467 2F 71            BLE    CONTI3   YES. GO [illegible]
03850 0469 B7 03C6          STA A  ICNT     NO. INC ICNT & RTI
03860 046C 7F 03C5          CLR    STRFLG   RESET I HAVE JUST STRED FLG
03870 046F 86 55            LDA A  #$55
03880 0471 B7 0250          STA A  DONTST   KEEP DONE TEST SET
03890 0474 BE 1C96          LDS    STKSAV   RETRIEV STACK
03900 0477 3B               RTI
03910                  *
03920 0478 CE 34B0 COMPR2   LDX    #XBITS   UPDATE X BIT CNTR
03930 047B B6 03C2          LDA A  XACCEL
03940 047E BD 0568          JSR    BITCNT
03950 0481 CE 3600          LDX    #XPDFM   UPDATE CH X FREQ OF OCCUR HIS
03960 0484 B6 03C2          LDA A  XACCEL
03970 0487 BD 066B          JSR    PDFSTR
03980 048A B6 03C2          LDA A  XACCEL
03990 048D B6 03C2          LDA A  XACCEL
04000 0490 BB 03DC          ADD A  XSLOPE   CALC & SAVE X(N-1)-X(N-2)
04010 0493 B7 03DC          STA A  XSLOPE
04020 0496 CE 34B3          LDX    #YBITS   UPDATE Y BIT CNTR
04030 0499 B6 03C3          LDA A  YACCEL
04040 049C BD 0568          JSR    BITCNT
04050 049F CE 3800          LDX    #YPDFM   UPDATE CH Y FREQ OF OCCUR HIS
04060 04A2 B6 03C3          LDA A  YACCEL
04070 04A5 BD 066B          JSR    PDFSTR
04080 04A8 B6 03C3          LDA A  YACCEL
04090 04AB B6 03C3          LDA A  YACCEL
04100 04AE BB 03DD          ADD A  YSLOPE   CALC & SAVE Y(N-1)-Y(N-2)
04110 04B1 B7 03DD          STA A  YSLOPE
04120 04B4 CE 34B6          LDX    #ZBITS   UPDATE Z BIT CNTR
04130 04B7 B6 03C4          LDA A  ZACCEL
04140 04BA BD 0568          JSR    BITCNT
04150 04BD CE [illegible]   LDX    #ZPDFM   UPDATE CH Z FREQ OF OCCUR HIS
04160 04C0 B6 03C4          LDA A  ZACCEL
04170 04C3 BD 066B          JSR    PDFSTR
04180 04C6 B6 03C4          LDA A  ZACCEL
04190 04C9 B6 03C4          LDA A  ZACCEL
04200 04CC BB 03DE          ADD A  ZSLOPE   CALC & SAVE Z(N-1)-Z(N-2)
04210 04CF B7 03DE          STA A  ZSLOPE
04220 04D2 CE 34B9          LDX    #TBITS   NOW UPDATE TIME BIT CNTR
04230 04D5 86 06            LDA A  #6
04240 04D7 BD 0568          JSR    BITCNT
04250 04DA B6 03C2 CONTR3   LDA A  XACCEL   PICK UP XACCEL NOW
04260 04DD BD 051B          JSR    CONSTR   GO CODE & STORE IT
04270 04E0 B6 03C3          LDA A  YACCEL   PICK UP YACCEL NOW
04280 04E3 BD 051B          JSR    CONSTR   GO CODE & STORE IT
04290 04E6 B6 03C4          LDA A  ZACCEL   PICK UP ZACCEL NOW
04300 04E9 BD 051B          JSR    CONSTR   GO CODE & STORE IT
04310 04EC B6 03C6          LDA A  ICNT     NOW GET ICNT FOR [illegible]
04320 04EF CE 3F00          LDX    #TPDF    [illegible]
04330 04F2 BD 066B          JSR    PDFSTR
04340 04F5 BD 055A          JSR    RESET    PUT IT 0 FOR [illegible]
```

AD-A100 799 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 6/5
ANALYSIS AND PERFORMANCE EVALUATION OF ELECTROCARDIOGRAM DATA C--ETC(U)
DEC 80 M D TOWNSEND
UNCLASSIFIED AFIT/GE/EE/80D-46 NL



```
04350 04F8 79 03C6          ROL     ICNT     ROL 7 BIT ICNT TO HS POSITION
04360 04FB 86 08            LDA A   #8       SET CNTER FOR TIME STORE LOOP
04370 04FD 4A       COMPR4  DEC A            DECREMENT LOOP CNTR
04380 04FE 27 0F            BEQ     COMPR6   TIME STRED YET?
04390 0500 79 03C6          ROL     ICNT     NO. ROTATE TIME WORD ONCE LEF
04400 0503 25 05            BCS     COMPR5   1 IN CARRY?
04410 0505 BD 055A          JSR     RESET    NO. PUT 0 IN MEM FILE
04420 0508 20 F3            BRA     COMPR4
04430 050A BD 0534 COMPR5   JSR     SET      YES. PUT 1 IN MEM FILE
04440 050D 20 EE            BRA     COMPR4
04450 050F 86 55   COMPR6   LDA A   #$55
04460 0511 B7 03C5          STA A   STRFLG   NOW SET "I HAVE JUST STORED"
04470 0514 B7 0250          STA A   DONTST   SET DONTST FLAG
04480 0517 BE 1C96          LDS     STKSAV
04490 051A 3B               RTI
04500                *
04510 051B BD 055A COMSTR   JSR     RESET    PUT 0 IN MEM FILE FOR DELIM
04520 051E 4D               TST A            NOW TEST STATUS OF A
04530 051F 2A 06            BPL     COMST1   IS IT +?
04540 0521 40               NEG A            NO. GET 2'S COMP FOR MAG
04550 0522 BD 0534          JSR     SET      NO. STORE 1 FOR SIGN BIT
04560 0525 20 03            BRA     COMST2
04570 0527 BD 055A COMST1   JSR     RESET    YES. STORE 0 FOR SIGN BIT
04580 052A 4D       COMST2  TST A            NOW CNT DOWN & STORE DATA
04590 052B 27 06            BEQ     COMST3   CNT 0?
04600 052D BD 0534          JSR     SET      NO. STORE 1 TO MEM FILE
04610 0530 4A               DEC A            DEC CNTR
04620 0531 20 F7            BRA     COMST2
04630 0533 39       COMST3  RTS
04640                *
04650                *
04660                *FUNC: SET
04670                *INPUTS: BUFPTR,BITPTR
04680                *OUTPUTS: BIT SET PT'ED TO BY ABOVE
04690                *CALLS: NOTHING
04700                *DESTROYS: B,X,CC
04710                *PURPOSE: THIS ROUTINE SETS THE BIT POINTED TO
04720                * BY BITPTR & BUFPTR AND THEN UPDATES THESE COUNTERS
04730                *
04740 0534 FE 024E SET      LDX     BUFPTR   GET CUR MEM WORD FOR DATA STO
04750 0537 F6 03C7          LDA B   BITPTR   GET BIT IN THAT WORD TO BE SE
04760 053A EA 00            ORA B   0,X      SET THE BIT
04770 053C E7 00            STA B   0,X      SAVE IN MEM FILE
04780 053E 0C       SET0    CLC              CLR CARRY FOR PROPER LEFT ROT
04790 053F 76 03C7          ROR     BITPTR   ROLL BITPTR ONCE RIGHT
04800 0542 24 15            BCC     SETRTS   WAS BITPTR DOWN TO 1ST BIT?
04810 0544 76 03C7          ROR     BITPTR   YES. IN BUFPTR $ SET BITPTR T
04820 0547 08               INX
04830 0548 FF 024E          STX     BUFPTR
04840 054B F6 024E          LDA B   BUFPTR
04850 054E C1 80            CMP B   #$80
04860 0550 26 07            BNE     SETRTS
04870 0552 7F 0250          CLR     DONTST   YES. STOP DATA COLLEC & RTI
04880 0555 BE 1C96          LDS     STKSAV
```

```
04890 0558 3B            RTI
04900 0559 39      SETRTS RTS
04910              *
04920              *FUNC: BITSET
04930              *INPUTS: BUFPTR,BITPTR
04940              *OUTPUTS: BIT RESET IN MEM FILE PT'ED TO BY ABOVE
04950              *CALLS: NOTHING
04960              *DESTROYS: B,X,CC
04970              *PURPOSE: THIS ROUTINE RESETS THE BIT IN THE MEMFILE
04980              * POINTED TO BY BUFPTR & BITPTR & UPDATES THESE COUN
04990              *
05000 055A FE 024E RESET  LDX    BUFPTR
05010 055D F6 03C7        LDA B  BITPTR
05020 0560 53             COM B
05030 0561 E4 00          AND B  0,X       RESET BIT IN MEM WORD
05040 0563 E7 00          STA B  0,X       SAVE BIT IN MEM FILE
05050 0565 7E 053E        JMP    SET0      UPDATE CNTR'S AND CK IF MEM F
05060              *
05070              *FUNC: BITCNT
05080              *INPUTS: ACCEL VLU IN A, BIT CNTR IN X
05090              *OUTPUTS: UPDATED BIT COUNT
05100              *CALLS:NOTHING
05110              *DESTROYS:A,B,CC
05120              *PURPOSE: THIS ROUTINE INC 3 BYTE BIT COUNTER WITH
05130              * NUMBER OF BIT USED TO STORE X,Y,Z,& TIME.
05140              *
05150 0568 2A 01   BITCNT BPL    BITCN1    IS ACCEL +
05160 056A 40             NEG A            NO. GET ABS VLU
05170 056B 4C      BITCN1 INC A            ADD 1 FOR DELIM 0 BIT
05180 056C 4C             INC A            ADD 1 FOR SIGN BIT
05190 056D C6 00          LDA B  #0        RESET B WITHOUT AFFECTING CAR
05200 056F AB 02          ADD A  2,X       INC LSB BYTE OF COUNT PTED TO
05210 0571 A7 02          STA A  2,X
05220 0573 86 00          LDA A  #0
05230 0575 E9 01          ADC B  1,X
05240 0577 E7 01          STA B  1,X
05250 0579 C6 08          LDA B  #8        NOW UP COUNT OF ACTUAL DATA B
05260 057B A9 00          ADC A  0,X
05270 057D A7 00          STA A  0,X
05280 057F 86 00          LDA A  #0
05290 0581 FB 34BF        ADD B  ACELCT+3 NO. INC ACELCT COUNT
05300 0584 F7 34BF        STA B  ACELCT+3
05310 0587 C6 00          LDA B  #0
05320 0589 B9 34BE        ADC A  ACELCT+2
05330 058C B7 34BE        STA A  ACELCT+2
05340 058F 86 00          LDA A  #0
05350 0591 F9 34BD        ADC B  ACELCT+1
05360 0594 F7 34BD        STA B  ACELCT+1
05370 0597 B9 34BC        ADC A  ACELCT
05380 059A B7 34BC        STA A  ACELCT
05390 059D 39      BITRTS RTS
05400              *
05410              *FUNC: DTACNT
05420              *INPUTS: NUM ON SMPLS COLLECTED
```

```
05430                *OUTPUTS: BITCNT=TO SMPLS*24
05440                *CALLS: NOTHING
05450                *DESTROYS: A,B,X,CC
05460                *PURPOSE: TO CALC NUM OF BITS STORED DURING COLLECTI
05470                *
05480 059E FE 3494 DTACNT LDX    SAMPNO
05490 05A1 86 18   DTACN1 LDA A  #24
05500 05A3 C6 00          LDA B  #0
05510 05A5 BB 34AF        ADD A  DTABIT+3
05520 05A8 B7 34AF        STA A  DTABIT+3
05530 05AB 86 00          LDA A  #0
05540 05AD F9 34AE        ADC B  DTABIT+2
05550 05B0 F7 34AE        STA B  DTABIT+2
05560 05B3 C6 00          LDA B  #0
05570 05B5 B9 34AD        ADC A  DTABIT+1
05580 05B8 B7 34AD        STA A  DTABIT+1
05590 05BB F9 34AC        ADC B  DTABIT
05600 05BE F7 34AC        STA B  DTABIT
05610 05C1 09             DEX
05620 05C2 8C 0000        CPX    #0
05630 05C5 26 DA          BNE    DTACN1
05640 05C7 39             RTS
05650                *
05660                *FUNC: SAMPL0
05670                *INPUTS: DATA VIA A/D CONV
05680                *OUTPUTS: ROUNDED DATA TO XDTA,YDTA,ZDTA
05690                *CALLS:PDFSTR (UPDATE PDF HISTOGRAMS)
05700                *DESTROYS:A,B,X,CC
05710                *PURPOSE: THIS ROUTINES SAMPLES THE A/D AND THEN
05720                * ROUNDS THE DATA TO 8 BITS FROM 12. THE DATA
05730                * IS PLACED IN XDTA,YDTA,ZDTA AND THE COLLEC
05740                * STAT ARE UPDATED.
05750                *
05760 05C8 CE E400 SAMPL0 LDX    #ADCZRO  NOW PULSE A/D TO START CONV
05770 05CB A7 00          STA A  0,X      ON CHANNEL 0
05780 05CD 01             NOP
05790 05CE A6 00          LDA A  0,X
05800 05D0 E6 01          LDA B  1,X
05810 05D2 B7 03B3        STA A  TEMPA1
05820 05D5 F7 03B4        STA B  TEMPB1
05830 05D8 08             INX
05840 05D9 08             INX
05850 05DA A7 00          STA A  0,X      NOW PULSE A/D TO CONV
05860 05DC 01             NOP             ON CHANNEL 1
05870 05DD A6 00          LDA A  0,X
05880 05DF E6 01          LDA B  1,X
05890 05E1 B7 03B5        STA A  TEMPA2
05900 05E4 F7 03B6        STA B  TEMPB2
05910 05E7 08             INX
05920 05E8 08             INX
05930 05E9 A7 00          STA A  0,X      NOW PULSE A/D TO CONV
05940 05EB 01             NOP             ON CH 2
05950 05EC A6 00          LDA A  0,X
05960 05EE E6 01          LDA B  1,X
```

```
05970 05F0 B7 03B7          STA A  TEMPA3
05980 05F3 F7 03B8          STA B  TEMPB3
05990 05F6 CE 03B3          LDX    #TEMPA1
06000 05F9 A6 00    SAMPL1  LDA A  0,X
06010 05FB E6 01            LDA B  1,X
06020 05FD 47               ASR A          12 BIT TO 8 BIT ROUNDED CONV
06030 05FE 56               ROR B
06040 05FF 47               ASR A
06050 0600 56               ROR B
06060 0601 47               ASR A
06070 0602 56               ROR B
06080 0603 47               ASR A
06090 0604 56               ROR B
06100 0605 F7 03B2          STA B  SHFBUF  SAVE 8 BIT RESULT OF SHIFT
06110 0608 86 00            LDA A  #0      AND ADD CARRY OUT OF LAST
06120 060A B9 03B2          ADC A  SHFBUF  SHIFT ROUNDING UP OR DOWN
06130 060D 36               PSH A          SAVE TO MEM FILE BUFFER VIA S
06140 060E 08               INX
06150 060F 08               INX
06160 0610 8C 03B9          CPX    #TEMPA3+2  CHANNEL Z ROUNDED TO 8 BITS
06170 0613 26 E4            BNE    SAMPL1  NO. GO SAMPLE NEXT CHANNEL
06180 0615 32               PUL A          GET Z DATA
06190 0616 B7 03BB          STA A  ZDATA
06200 0619 FE 3494          LDX    SAMPNO  GET CUR SAMPLE COUNT
06210 061C B1 3497          CMP A  MAXZ    IS CUR Z MAX OVE SAMPLE SET?
06220 061F 2F 06            BLE    SPZMIN  NO. GO CHECK FOR MIN
06230 0621 B7 3497          STA A  MAXZ    YES. KEEP CUR VLU
06240 0624 FF 3498          STX    MAXZLO  KEEP CUR SAMPLE NUM
06250 0627 B1 349A SPZMIN   CMP A  MINZ    IS CUR Z MIN OVER SAMPLE SET?
06260 062A 2C 06            BGE    SPYMAX  NO. GO CHECK FOR Y MAX
06270 062C B7 349A          STA A  MINZ    YES. KEEP CUR VLU
06280 062F FF 349B          STX    MINZLO  KEEP CUR SAMPLE NUM
06290 0632 32       SPYMAX  PUL A          GET CUR Y DATA
06300 0633 B7 03BA          STA A  YDATA
06310 0636 B1 349D          CMP A  MAXY    IS CUR Y MAX OVER SAMPLE SET
06320 0639 2F 06            BLE    SPYMIN  NO. GO CHEK FOR MIN
06330 063B B7 349D          STA A  MAXY    YES. KEEP CUR Y VLU
06340 063E FF 349E          STX    MAXYLO  KEEP CUR SAMPLE NUM
06350 0641 B1 34A0 SPYMIN   CMP A  MINY    IS CUR Y MIN OVER SAMPLE SET?
06360 0644 2C 06            BGE    SPXMAX  NO. GO CHEK FOR X MAX
06370 0646 B7 34A0          STA A  MINY    YES. KEEP CUR Y VLU
06380 0649 FF 34A1          STX    MINYLO  KEEP CUR SAMPLE NUM
06390 064C 32       SPXMAX  PUL A          GET CUR X DATA
06400 064D B7 03B9          STA A  XDATA
06410 0650 B1 34A3          CMP A  MAXX    IS CUR X MAX OVER SAMPLE SET?
06420 0653 2F 06            BLE    SPXMIN  NO. GO CHECK X MIN
06430 0655 B7 34A3          STA A  MAXX    YES. KEEP CUR X VLU
06440 0658 FF 34A4          STX    MAXXLO  KEEP CUR SAMPLE VLU
06450 065B B1 34A6 SPXMIN   CMP A  MINX    IS CUR X MIN OVER SAMPLE SET?
06460 065E 2C 06            BGE    MMDONE  NO. EXIT MAX,MIN UPDATE
06470 0660 B7 34A6          STA A  MINX    YES. KEEP CUR X VLU
06480 0663 FF 34A7          STX    MINXLO  KEEP CUR SAMPLE COUNT
06490 0666 39       MMDONE  RTS
06500               *
```

```
06510 0667 0002    TEMPDF RMB    2          TEMP WORKING BUFFER
06520 0669 0002    TEMPST RMB    2          TEMP STACK SAVE BUFFER
06530              *
06540 066B BF 0669 PDFSTR STS    TEMPST     SAVE STACK PTR
06550 066E FF 0667        STX    TEMPDF     SAVE PDF PTR
06560 0671 16             TAB
06570 0672 BB 0668        ADD A  TEMPDF+1   NOW CALCULATE ADDRESS IN INC
06580 0675 B7 0668        STA A  TEMPDF+1   IN THE PDF MEM BUFF
06590 0678 86 00          LDA A  #0
06600 067A C5 80          BIT B  #$80       IS THE DATA VLU NEG
06610 067C 2B 05          BMI    PDFST1     YES. SBC MSB VS ADC
06620 067E B9 0667        ADC A  TEMPDF     NO. ADC MSB
06630 0681 20 03          BRA    PDFST2     AND STORE
06640 0683 B2 0667 PDFST1 SBC A  TEMPDF     VLU IS NEG. SBC
06650 0686 B7 0667 PDFST2 STA A  TEMPDF     AND STORE
06660 0689 17             TBA               RECOVER VLU AND ADD (SUB) AGA
06670 068A BB 0668        ADD A  TEMPDF+1   FOR PROPER ADDR CALC
06680 068D B7 0668        STA A  TEMPDF+1
06690 0690 86 00          LDA A  #0
06700 0692 C5 80          BIT B  #$80       IS VLU NEG?
06710 0694 2B 05          BMI    PDFST3     YES. SBC MSB VS ADC MSB
06720 0696 B9 0667        ADC A  TEMPDF     NO. ADC MSB
06730 0699 20 03          BRA    PDFST4
06740 069B B2 0667 PDFST3 SBC A  TEMPDF     VLU IS NEG. SBC
06750 069E B7 0667 PDFST4 STA A  TEMPDF     AND STORE
06760 06A1 FE 0667        LDX    TEMPDF     NO LOAD CALC ADDR FOR INDEX
06770 06A4 AE 00          LDS    0,X        GRAB VLU IN CALC ADDR
06780 06A6 31             INS               INC IT
06790 06A7 AF 00          STS    0,X        AND STORE IT BACK IN BUFFER
06800 06A9 BE 0669        LDS    TEMPST     RECOVER STACK POINTER
06810 06AC 17             TBA
06820 06AD 39             RTS
06830              *
06840              *FUNC: CALINT
06850              *INPUTS: NONE
06860              *OUTPUTS: NONE
06870              *CALLS: NOTHING
06880              *DESTROYS :NO REG (INTR HANDLER)
06890              *PURPOSE: THIS ROUTINE IS USED FOR CALIBRATING
06900              * THE MAX NUMBER OF LOOPS POSSIBLE DURING
06910              * AN INTERRUPT PERIOD.  THIS ROUTINE UPDATES THE
06920              * DONTST FLAG AND RESETS THE ST6800 INT FLIP FLOP
06930              * AND THEN RETURNS.
06940              *
06950 06AE 7C 0250 CALINT INC    DONTST     INCREMENT DONE TEST FLAG
06960 06B1 B7 E400        STA A  ADCZRO     CLR ST6800 INTR FLIP FLOP
06970 06B4 01             NOP
06980 06B5 3B             RTI
06990              *
07000              * END OF TOLAN-A OVERLAY ROUTINES
07010              *
07020                      END
```

```
00030                *
00040                *      PROGRAM NAME:  DECPRS
00050                *      AUTHOR :  CAPT. MEL TOWNSEND
00060                *      VERSION : 1.3
00070                *      VERSION DATE : 29 NOV 80
00080                *
00090                ****************************************************
00100                *
00110                *       PROGRAM DESCRIPTION
00120                *
00130                *    THIS PROGRAM PERFORMS THE DECOMPRESSION OPERATI
00140                * ON THE DATA COMPRESSED BY THE TOLAN-A,TOLAN-B,DOWE
00150                * ETC.  THE ROUTINE ASSUMES THAT THE DATA IS IN MEMO
00160                * AND READS THE COMPRESSION TYPE FROM THE MEMORY FIL
00170                * BUFFER HEADER.  AFTER PROMPT TO THE TERMINAL, THIS
00180                * PROGRAM BEGINS DECOMPRESSION AND OUTPUTS THE DECOM
00190                * DATA TO AN ANALOG DISPLAY DEVICE (IE OSCILLISCOPE)
00200                * D/A CONVERTER 0.  THE ROUTINE REQUIRES AN INTERRUP
00210                * CLOCK VIA THE A/D BOARD (ST6800) AND THE ASSUMPTIO
00220                * IS MADE THAT THE INTERRUPT CLOCK FREQUENCY IS ADJU
00230                * TO THE SAMPLE RATE AT WHICH THE DATA WAS TAKEN.
00240                *
00250                ****************************************************
00260                *
00270                *            START OF DECPRS
00280                *
00290                ****************************************************
00300                *
00310 0500                 ORG    $0500     PROGRAM ORIGIN
00320                *
00330                      OPT    O         ASSB OPT. LIST ASSMBLY
00340                      OPT    NOG       ASSB OPT. SUPRESS FCC LIST
00350                *
00360                ****************************************************
00370                *
00380                *            LABLE DESCRIPTIONS
00390                *
00400                *  SUPPORT SUBROUTINES
00410                *
00420      CA8F      OUTNCR EQU    $CA8F     EPROMDOS. OUTPUT STRING
00430      CA2C      KEYBD0 EQU    $CA2C     EPROMDOS. INPUT ALPH STRING
00440      0100      DISPLA EQU    $0100     EKG-EXEC. TERMINAL INTFC DRIV
00450                *
00460                * DATA BUFFERS
00470                *
00480      0020      CHNLBF EQU    $0020     CHNL SELECT FLG FRM DISPLAY
00490      3C00      XINIT  EQU    $3C00     INITIAL COND CH X
00500      3C01      YINIT  EQU    $3C01     INITIAL COND CH Y
00510      3C02      ZINIT  EQU    $3C02     INITIAL COND CH Z
00520      3002      ENDBUF EQU    $3002     END-OF-STRING FRM TERM INPUT
00530      FFF8      IRQVEC EQU    $FFF8     MASKABLE INTR JMP VECTOR ADDR
00540      3400      HDRSTR EQU    $3400     MEM FILE HEADER ADDR
00550      3C00      SECZRO EQU    $3C00     DATA START LOCATION
00560                *
```

```
00570                 * HARDWARE ADDRESSES
00580                 *
00590          E500   DACZRO EQU    $E500    D/A CH 0
00600          E400   ADCZRO EQU    $E400    A/D CH 0
00610                 *
00620                 *
00630                 *FUNCTION: DECPRS
00640                 *INPUTS : CHNL SELECT VIA CHNLBF
00650                 *OUTPUTS : DECRPRSED DATA VIA D/A
00660                 *CALLS : OUTNCR,KEYBD0,DISPLAY
00670                 *DESTROYS (REG) : A,B,X,CC
00680                 *PURPOSE : THIS IS THE SET UP ROUTINE WHICH INITIALI
00690                 * THE PROGRAM CONSTANTS AND, DETERMINES WHAT TYPE OF
00700                 * COMPRESSION WAS USED, AND ENABLES RECONSTRUCTION
00710                 * INTERUPPTS.
00720                 *
00730 0500 0F       DECPRS SEI
00740 0501 FE 3400         LDX    HDRSTR   GET CMPRS TYPE FROM MEM FILE
00750 0504 FF 0538         STX    NOTDN1   STR CPR TYPE IN ERROR MSG
00760 0507 8C 5441         CPX    #$5441   IS IT TOLAN-A?
00770 050A 27 61           BEQ    TADECP   YES. GO DECPRS IT.
00780 050C CE 0518         LDX    #NOTDNE  NO. CPRS ROUTINE NOT DONE
00790 050F BD CA8F         JSR    OUTNCR   PRINT ERROR MSG & RETURN
00800 0512 BD CA2C         JSR    KEYBD0
00810 0515 7E 0100         JMP    DISPLA   RETURN TO DISPLAY ROUTINE
00820                 *
00830 0518 1A07     NOTDNE FDB    $1A07,$0D0A
00840 051C 44              FCC    /DECOMPRESSION FOR FILE TYPE /
00850 0538 0002     NOTDN1 RMB    2
00860 053A 20              FCC    / NOT YET IMPLEMENTED./
00870 054F 0D0A            FDB    $0D0A,$0D0A
00880 0553 50              FCC    /PRESS ANY KEY TO CONTINUE/
00890 056C 04              FCB    4
00900                 *
00910 056D 96 20    TADECP LDA A  CHNLBF   GET WHICH CHANNEL TO BE DECPR
00920 056F 81 00           CMP A  #0       IS IT CHNL X?
00930 0571 26 07    TADEC0 BNE    TADEC1   NO. CHECK IF Y.
00940 0573 C6 58           LDA B  #'X      PUT ASCII X IN ERROR MSG
00950 0575 F7 0846         STA B  CHNASC
00960 0578 20 10           BRA    TADEC3
00970 057A 81 01    TADEC1 CMP A  #1       IS IT CH Y?
00980 057C 26 07           BNE    TADEC2   NO. MUST BE Z.
00990 057E C6 59           LDA B  #'Y      PUT ASCII Y IN ERROR MSG
01000 0580 F7 0846         STA B  CHNASC
01010 0583 20 05           BRA    TADEC3
01020 0585 C6 5A    TADEC2 LDA B  #'Z      PUT ASCII Z IN ERROR MSG
01030 0587 F7 0846         STA B  CHNASC
01040 058A CE 0811  TADEC3 LDX    #GOMSG   GET PROMPT MSG LOC
01050 058D BD CA8F         JSR    OUTNCR   NOW PRINT PROMPT
01060 0590 BD CA2C         JSR    KEYBD0
01070 0593 FE 3002         LDX    ENDBUF   GET ANSWER FROM TERMINAL
01080 0596 09              DEX
01090 0597 E6 00           LDA B  0,X
01100 0599 C1 4E           CMP B  #'N      WAS IT NO?
```

```
01110 059B 26 03          BNE    TADEC4   YES. EXIT ROUTINE
01120 059D 7E 0100        JMP    DISPLA
01130 05A0 CE 3C03 TADEC4 LDX    #SECTOR+3  POSITION POINTER TO DEL STR
01140 05A3 FF 0611        STX    PUPTR
01150 05A6 86 80          LDA A  #$80      INITIALIZE BITPTR TO LEFT BIT
01160 05A8 B7 0613        STA A  BITPTR
01170 05AB 7F 0617        CLR    XSLOPE    INIT SLOPES TO ZERO (2ND INIT
01180 05AE 7F 0618        CLR    YSLOPE
01190 05B1 7F 0619        CLR    ZSLOPE
01200 05B4 B6 3C00        LDA A  XINIT     GET FIRST INIT COND
01210 05B7 B7 0614        STA A  XDTA      & PUT IN DATA BUF
01220 05BA B6 3C01        LDA A  YINIT
01230 05BD B7 0615        STA A  YDTA
01240 05C0 B6 3C02        LDA A  ZINIT
01250 05C3 B7 0616        STA A  ZDTA
01260 05C6 7F 0629        CLR    TARFLG    CLR DECERSION FLG
01270 05C9 7F 061A        CLR    DELT      INIT TIME COUNTER
01280 05CC FE FFF8        LDX    IRQVEC    PICK UP CURRENT INT VEC
01290 05CF FF 061D        STX    VECSAV    SAVE IT IN TEMP BUFFER
01300 05D2 CE 062F        LDX    #TARECN   PUT NEW INT VEC IN
01310 05D5 FF FFF8        STX    IRQVEC
01320 05D8 86 AA          LDA A  #$AA      SET DONTST FLAG
01330 05DA B7 062A        STA A  DONTST
01340 05DD CE 0000        LDX    #0
01350 05E0 FF E500        STX    DACZRO    SEND ENABLE PLS TO INT CIRCUI
01360                *
01370 05E3 0E      TAINTR CLI               ENABLE CPU FOR RECONS INTR'S
01380 05E4 B6 062A        LDA A  DONTST    CHECK IF DECPRS DONE
01390 05E7 81 00          CMP A  #0        DONE YET?
01400 05E9 26 F8          BNE    TAINTR    NO. KEEP LOOPING
01410 05EB 0F             SEI              INSURE INTR DISABLE
01420 05EC CE 0878        LDX    #GOAGIN   PRINT GO AGAIN MSG
01430 05EF BD CA8F        JSR    OUTNCR    "ENTER CONTINUATION COMMAND"
01440 05F2 BD CA2C        JSR    KEYBD0
01450 05F5 FE 3002        LDX    ENDBUF    NOW GET COMMAND
01460 05F8 09             DEX
01470 05F9 E6 00          LDA B  0,X
01480 05FB 17             TBA              PUT IT IN A
01490 05FC 90 28          SUB A  $28       TRANSFORM X TO 0
01500 05FE 97 20          STA A  CHNLBF    PUT IN CHNL FLAG
01510 0600 F7 0846        STA B  CHNASC    PUT ASCII IN ERROR MSG
01520 0603 C1 58          CMP B  #'X       WAS IT .LT. 'X?
01530 0605 2D 07          BLT    TADEC5    YES. EXIT ROUTINE
01540 0607 C1 5A          CMP B  #'Z       WAS CMMD .GT. 'Z ?
01550 0609 2E 03          BGT    TADEC5    YES. EXIT ROUTINE
01560 060B 7E 05A0        JMP    TADEC4    NO. GO EXECUTE DECPRS AGAIN
01570 060E 7E 0100 TADEC5 JMP    DISPLA    RETURN TO DISPLAY
01580                *
01590                ********************************************************
01600                *       END  DECPRS INIT ROUTINE
01610                ********************************************************
01620                *
01630                *
01640                * PARAMETER BUFFER
```

```
01650                    *
01660 0611 0002    BUFPTR RMB    2        CURRENT MEM FILE WORD POINTER
01670 0613 0001    BITPTR RMB    1        CURRENT BIT IN CURRENT WORD PTR
01680 0614 0001    XDTA   RMB    1        CH X DATA BUF
01690 0615 0001    YDTA   RMB    1        CH Y DATA BUG
01700 0616 0001    ZDTA   RMB    1        CH Z DATA BUF
01710 0617 0001    XSLOPE RMB    1        CH X 1ST DIFF BUFFER
01720 0618 0001    YSLOPE RMB    1        CH Y 1ST DIFF BUF
01730 0619 0001    ZSLOPE RMB    1        CH Z 1ST DIFF
01740 061A 0001    DELT   RMB    1        TIME COMPRESSION RUN LEN BUFF
01750 061B 0001    VALUE  RMB    1        DATA RET BUF FROM GETVLU
01760 061C 0001    CNT    RMB    1        COUNTER IN GETTIM
01770 061D 0002    VECSAV RMB    2        IRQ VECTOR SAVE BUF
01780 061F 0001    BITVLU RMB    1        BIT SET/RESET FLAG FROM GETBI
01790 0620 0001    SIGN   RMB    1        SIGN BIT ON DECODED DATA
01800 0621 0001    XACCEL RMB    1        UNCODED CH X ACCEL DATA
01810 0622 0001    XTEMP  RMB    1        TEMP BUF USED IN DECFRS
01820 0623 0001    YACCEL RMB    1        UNCODED CH Y ACCEL DATA
01830 0624 0001    YTEMP  RMB    1        TEMP BUF USED IN DECFRS
01840 0625 0001    ZACCEL RMB    1        UNCODED CH Z ACCEL DATA
01850 0626 0001    ZTEMP  RMB    1        TEMP BUF USED IN DECFRS
01860 0627 0002    WORKBF RMB    2        TEMP WORKING BUFFER
01870 0629 0001    TARFLG RMB    1        FLG USED IN DECFRS ALGORITHM
01880 062A 0001    DONTST RMB    1        FLG USED TO EXIT INTR LOOP
01890 062B 0002    DABUF  RMB    2        D/A CONV OUTPUT BUFFER
01900 062D 0002    STKSAV RMB    2        TEMP STACK SAVE BUFFER
01910                    *
01920                    *FUNCTION : TARECN
01930                    *INPUTS : BUFFER VALUES
01940                    *OUTPUTS : D/A VALUES VIA ST-6800
01950                    *CALLS : GETVLU,GETTIM,OUTDA
01960                    *DESTROYS        A,B,X,CC
01970                    *PURPOSE : THIS IS THE INTERRUPT SERVICE ROUTINE WHI
01980                    * PERFORMS RECONSTRUCTION ON THE TOLAN-A ENCODED DAT
01990                    * FILE IN MEMORY.  THE DATA IS DECODED AND OUTPUT ON
02000                    * D/A (ST6800) CH 0.
02010                    *
02020                    ****************************************************
02030                    *
02040 062F B7 E400 TARECN STA A  ADCZRO   ACK INTR. RESET INTR FF ON ST
02050 0632 01              NOP
02060 0633 BF 062D         STS    STKSAV   SAVE ENTRY STACK FOR RTI
02070 0636 B6 0846         LDA A  CHNASC   CHECK CHNL SELECT, OUTPUT CUR
02080 0639 81 58           CMP A  #'X      IS CHNNL SELECT X ?
02090 063B 26 08           BNE    TAREC1   NO. CHK IF Y
02100 063D B6 0614         LDA A  XDTA     YES. GET CUR XDTA VALUE.
02110 0640 BD 0796         JSR    OUTDA    OUTPUT VALUE VIA D/A CH 0
02120 0643 20 12           BRA    TARE22
02130 0645 81 59   TAREC1  CMP A  #'Y      IS CHNNL SELECT Y ?
02140 0647 26 08           BNE    TAREC2   NO. MUST BE Z
02150 0649 B6 0615         LDA A  YDTA     GET CUR YDTA VALUE.
02160 064C BD 0796         JSR    OUTDA    OUTPUT Y VALUE VIA D/A CH 0
02170 064F 20 06           BRA    TARE22
02180 0651 B6 0616 TAREC2  LDA A  ZDTA     GET CUR ZDTA VALUE.
```

```
02190 0654 BD 0796          JSR    OUTDA    OUTPUT Z VALUE VIA D/A CH 0
02200 0657 B6 0629 TARE22   LDA A  TARFLG   WAS LAST DATA PT EXTROPOLATED
02210 065A 81 00            CMP A  #0
02220 065C 26 33            BNE    TARE33   YES. DON'T GO GET NEXT ACCEL V
02230 065E 86 58            LDA A  #'X      PUT ASCII X IN GETVLU ERROR M
02240 0660 B7 0A3D          STA A  OVFLCH
02250 0663 BD 0700          JSR    GETVLU   NOW GO GET NEXT XACCEL VALUE
02260 0666 B6 061B          LDA A  VALUE    AND PUT XACCEL
02270 0669 B7 0621          STA A  XACCEL
02280 066C B7 0622          STA A  XTEMP    SAVE XACCEL IN TEMP BUFF
02290 066F 86 59            LDA A  #'Y      PUT 'Y IN GETVLU ERROR MSG
02300 0671 B7 0A3D          STA A  OVFLCH
02310 0674 BD 0700          JSR    GETVLU   GET YACCEL DTA
02320 0677 B6 061B          LDA A  VALUE
02330 067A B7 0623          STA A  YACCEL   PUT IN YACCEL VAR
02340 067D B7 0624          STA A  YTEMP    PUT IN YACCEL TEMP VAR
02350 0680 86 5A            LDA A  #'Z      PUT 'Z IN GETVLU ERROR MSG
02360 0682 B7 0A3D          STA A  OVFLCH
02370 0685 BD 0700          JSR    GETVLU   GET ZACCEL DATA
02380 0688 B7 0625          STA A  ZACCEL   PUT IN ZACCEL VAR
02390 068B B7 0626          STA A  ZTEMP    PUT IN ZACCEL TEMP VAR
02400 068E BD 075E          JSR    GETTIM   NOW GET TIME COPRESSION RUN L
02410 0691 7F 0629 TAREC3   CLR    TARFLG   RESET FLAGS
02420 0694 7A 061A          DEC    DELT     NOW CK IF RUN LEN ONLY 1
02430 0697 27 09            BEQ    TAREC4   YES. ACCEL VAR NOT ZERO
02440 0699 7F 0621          CLR    XACCEL   ACCEL WAS ZRO SO CLR X-Y-ZACC
02450 069C 7F 0623          CLR    YACCEL
02460 069F 7F 0625          CLR    ZACCEL
02470 06A2 B6 0617 TAREC4   LDA A  XSLOPE   NOW GET LAST SLOPE
02480 06A5 BB 0621          ADD A  XACCEL   & CALC NEW SLOPE WITH ACCEL
02490 06A8 B7 0617          STA A  XSLOPE
02500 06AB B6 0618          LDA A  YSLOPE
02510 06AE BB 0623          ADD A  YACCEL
02520 06B1 B7 0618          STA A  YSLOPE
02530 06B4 B6 0619          LDA A  ZSLOPE
02540 06B7 BB 0625          ADD A  ZACCEL
02550 06BA B7 0619          STA A  ZSLOPE
02560 06BD B6 0614          LDA A  XDTA     NOW CALC DATA VALUE USING SLO
02570 06C0 BB 0617          ADD A  XSLOPE
02580 06C3 B7 0614          STA A  XDTA
02590 06C6 B6 0615          LDA A  YDTA
02600 06C9 BB 0618          ADD A  YSLOPE
02610 06CC B7 0615          STA A  YDTA
02620 06CF B6 0616          LDA A  ZDTA
02650 06D8 B6 061A          LDA A  DELT     WAS RUN LEN .GT. 2 PTS ?
02660 06DB 27 22            BEQ    TARRTI   IF YES, DONT GET NEW PTS
02670 06DD 81 01            CMP A  #1       JUST EXTROPOLATE POINTS UNTIL
02680 06DF 27 07            BEQ    TAREC5   DELTA IS REDUCED TO 1
02690 06E1 86 37            LDA A  #55      SET TARFL1
02700 06E3 B7 0629          STA A  TARFLG
02710 06E6 20 17            BRA    TARRTI
02720 06E8 B6 0622 TAREC5   LDA A  XTEMP    RETRIEVE ACCEL VLUS PREVIOUSL
02730 06EB B7 0621          STA A  XACCEL
02740 06EE B6 0624          LDA A  YTEMP
```

```
02750 06F1 B7 0623          STA A  YACCEL
02760 06F4 B6 0626          LDA A  ZTEMP
02770 06F7 B7 0625          STA A  ZACCTL
02780 06FA 86 55            LDA A  #$55      NOW SET TARFL2 FLAG
02790 06FC B7 0629          STA A  TARFLG
02800 06FF 3B         TARRTI RTI
02810                       *
02820                       ****************************************************
02830                       *          END OF INTERRUPT HANDLER
02840                       ****************************************************
02850                       *
02860                       *                SUBROUTINES
02870                       *
02880                       *
02890                       *FUNCTION : GETVLU
02900                       *INPUTS : BUFPTR,BITPTR,SIGN BUFFERS
02910                       *OUTPUTS :VALUE BUFFER
02920                       *CALLS : GETBIT
02930                       *DESTROYS : A,X,CC
02940                       *PURPOSE : THIS ROUTINE SCANS MEMORY STARTING AT POI
02950                       * DEFINED BY BUFPTR,BITPTR AND GETS ACCEL VALUE. BUF
02960                       * AND BITPTR ARE UPDATED.
02970                       *
02980                       *
02990 0700 7F 061B GETVLU CLR    VALUE    INSURE VALUE IS CLEAR ON STAR
03000 0703 BD 07ED        JSR    GETBIT   GET DELIMITER BIT
03010 0706 B6 061F        LDA A  BITVLU
03020 0709 81 00          CMP A  #0       IS DELIM BIT =0 ?
03030 070B 27 06          BEQ    GETVL1   YES. GET NEXT BIT
03040 070D CE 097D        LDX    #SYNCMS  NO. SYNC ERROR HAS OCCURED. P
03050 0710 7E 07F5        JMP    ERROR
03060 0713 7F 0620 GETVL1 CLR    SIGN     INSURE SIGN CLR
03070 0716 BD 07ED        JSR    GETBIT   GET SIGN BIT
03080 0719 B6 061F        LDA A  BITVLU
03090 071C 81 00          CMP A  #0       IS ACCEL POS?
03100 071E 27 03          BEQ    GETVL2   YES. LEAVE SIGN CLR
03110 0720 73 0620        COM    SIGN     NO. SET SIGN FLAG
03120 0723 BD 07ED GETVL2 JSR    GETBIT   GET FIRST CODE BIT
03130 0726 B6 061F        LDA A  BITVLU
03140 0729 26 11          BNE    GETVL4
03150 072B FE 0611 GETVL3 LDX    BUFPTR   END OF VALUE DECODE. PACK UP
03160 072E 78 0613        ASL    BITPTR   FOR CORRECT POSITIONING FOR N
03170 0731 24 2A          BCC    GETVL6   VALUE FETCH
03180 0733 79 0613        ROL    BITPTR
03190 0736 09             DEX
03200 0737 FF 0611        STX    BUFPTR
03210 073A 20 21          BRA    GETVL6
03220 073C 7C 061B GETVL4 INC    VALUE    NOW PAST DELIM & SIGN. NOW CO
03230 073F 2A 06          BPL    GETVL5   BITS. IS CODE WORD > 127 ?
03240 0741 CE 09FB        LDX    #OVFLMS  YES. OVERFLOW HAS OCCURED. PR
03250 0744 7E 07F5        JMP    ERROR
03260 0747 BD 07ED GETVL5 JSR    GETBIT   NOW GET DATA VALUE CODE BITS
03270 074A B6 061F        LDA A  BITVLU
03280 074D 81 00          CMP A  #0       DELIM DETECTED YET?
```

```
03290 074F 26 EB          BNE    GETVL4   NO. KEEP FETCHING BITS
03300 0751 B6 0620        LDA A  SIGN     YES. GET SIGN AND ADJUST CNT
03310 0754 81 00          CMP A  #0
03320 0756 27 03          BEQ    GETVL3   SIGN 0. VALUE CORR. GO ADJ PT
03330 0758 70 061B        NEG    VALUE    SIGN 1. TAKE 2'S COMP, ADJ PT
03340 075B 20 CE          BRA    GETVL3
03350 075D 39       GETVL6 RTS
03360                 *
03370                 ********************************************************
03380                 *           END OF SUBROUTINE GETVLU
03390                 ********************************************************
03400                 *
03410                 *
03420                 *FUNCTION : GETTIM
03430                 *INPUTS : BITVLU (CURR BIT VALUE PT'ED TO)
03440                 *OUTPUTS : DELT (CURRENT TIME COUNT)
03450                 *CALLS : GETBIT, ERROR
03460                 *DESTROYS : A,X,CC REGISTERS
03470                 *PURPOSE : THIS ROUTINE, WHEN CALLED AFTER 3 GETVLU
03480                 * RETURNS THE VALUE OF THE RUN LENGTH TIME COUNTER I
03490                 * VARIABLE DELT.
03500                 *
03510                 *
03520 075E BD 07ED GETTIM JSR    GETBIT   GET DELIM BIT
03530 0761 B6 061F        LDA A  BITVLU
03540 0764 81 00          CMP A  #0       IS BIT 0?
03550 0766 27 06          BEQ    GETTI1   YES. PROPER DELIM. CONTINUE
03560 0768 CE 09AA        LDX    #SYNCTI  NO. SYNC ERROR. PRINT & ERR O
03570 076B 7E 07F5        JMP    ERROR
03580 076E 7F 061A GETTI1 CLR    DELT     INSURE TIME CNT INITIALLY CLR
03590 0771 86 07          LDA A  #7       INIT CNT FOR 7 BIT TIME
03600 0773 B7 061C        STA A  CNT
03610 0776 BD 07ED GETTI2 JSR    GETBIT   GET DELTA T BIT
03620 0779 B6 061F        LDA A  BITVLU
03630 077C BA 061A        ORA A  DELT     SET LSB BIT ACCORD TO T BIT
03640 077F B7 061A        STA A  DELT
03650 0782 78 061A        ASL    DELT     SHIFT DELT FOR NEXT BIT
03660 0785 7A 061C        DEC    CNT      DONE WITH TIME YET?
03670 0788 26 EC          BNE    GETTI2   NO. KEEP FETCHING TIME BITS
03680 078A B6 061A        LDA A  DELT     YES. GET TIME VALUE
03690 078D 26 06          BNE    GETTI3   IS TIME CNT 0 ?
03700 078F CE 0A3F        LDX    #TIMERR  YES. ERROR. PRNT ERR MSG
03710 0792 7E 07F5        JMP    ERROR    AND ERROR OFF
03720 0795 39       GETTI3 RTS
03730                 *
03740                 ********************************************************
03750                 *           END GETTIM SUBROUTINE
03760                 ********************************************************
03770                 *
03780                 *
03790                 *FUNCTION : OUTDA
03800                 *INPUTS : DATA VALUE TO BE D/A'ED IN ACC A
03810                 *OUTPUTS : VALUE IN ACC A VIA D/A CH 0
03820                 *CALLS : NOTHING
```

```
03830                *DESTROYS : B,CC
03840                *PURPOSE : THIS ROUTINE OUTPUTS THE DATA IN
03850                * ACCUMULATOR A TO THE D/A FOR CONVERSION AND
03860                * TRANSMISSION.  THE 8 BIT VALUE IS SHIFTED
03870                * TO FORM THE 12 BIT OPERAND REQUIRED BY THE
03880                * D/A (ST6800).
03890                *
03900                *
03910 0796 36        OUTDA  PSH A            SAVE VLU IN A ACC
03920 0797 B7 062C          STA A  DABUF+1   PUT VALUE IN D/A OUT BUFF
03930 079A 7F 062B          CLR    DABUF     CLR MSB BYTE OF D/A OUT BUFF
03940 079D 78 062C          ASL    DABUF+1   LEFT SHIFT 2 BYTE D/A WORD
03950 07A0 79 062B          ROL    DABUF     FOR CONVERSION OF 8 BIT
03960 07A3 78 062C          ASL    DABUF+1   VALUE TO 12 BIT
03970 07A6 79 062B          ROL    DABUF
03980 07A9 78 062C          ASL    DABUF+1
03990 07AC 79 062B          ROL    DABUF
04000 07AF 78 062C          ASL    DABUF+1
04010 07B2 79 062B          ROL    DABUF
04020 07B5 FE 062B          LDX    DABUF     NOW LD INDEX WITH 12 BIT VLU
04030 07B8 FF E500          STX    DACZRO    AND OUTPUT TO D/A
04040 07BB 32               PUL A           RETRIEV ENTY A ACC VLUE
04050 07BC 39               RTS
04060                *
04070                ********************************************************
04080                *         END SUBROUTINE OUTDA
04090                ********************************************************
04100                *
04110                *
04120                *FUNCTION : GETBIT
04130                *INPUTS : BUFPTR,BITPTR POINTERS
04140                *OUTPUTS : BITVLU, DONIST FLAGS
04150                *CALLS :NOTHING
04160                *DESTROYS :A,CC
04170                *PURPOSE : THIS ROUTINE CHECKS THE STATE OF THE BIT
04180                * POINTED TO BY THE BUFPTR,BITPTR PAIR AND SETS THE
04190                * THE BITVLU FLAG ACCORDINGLY.  THIS ROUTINE CHECK
04200                * FOR END OF MEMORY AND FORCES EXIT OF RECONSTRUCTIO
04210                * ROUTINES IF EOM IS DETECTED.
04220                *
04230                *
04240 07BD 7F 061F GETBIT CLR    BITVLU   INSURE BITVLU STARTS RESET
04250 07C0 FE 0611        LDX    BUFPTR   GET MEM WORD POINTER
04260 07C3 F6 0613        LDA B  BITPTR   GET BIT POINTER IN THAT MEM W
04270 07C6 E5 00          BIT B  0,X      CHECK IF BIT SET
04280 07C8 27 03          BEQ    GETBI1   NO IT WAS RESET. RESET BITVLU
04290 07CA 7C 061F        INC    BITVLU   YES. BIT WAS SET. SET BITVLU
04300 07CD 74 0613 GETBI1 LSR    BITPTR   NOW UPDATE POINTERS
04310 07D0 24 22          BCC    GETRTS
04320 07D2 76 0613        ROR    BITPTR
04330 07D5 08             INX
04340 07D6 FF 0611        STX    BUFPTR
04350 07D9 F6 0611        LDA B  BUFPTR   CHECK FOR END OF MEMORY
04360 07DC C1 80          CMP B  #$80
```

```
04370 07DE 26 14          BNE    GETRTS   NO, CONTINUE
04380 07E0 0F             SEI             NOW TURN OFF INTR AND RET
0[illegible] CE 4000      LDX    #$4000
04[illegible] FF E500     STX    DACZRO
04410 07E7 FE 061D        LDX    VECSAV
04420 07EA FF FFF8        STX    IRQVEC
04430 07ED 7F 062A        CLR    DONTST   YES. CLR DONTST FLAG AND RTI
04440 07F0 BE 062D        LDS    STKSAV   RETRIEVE INTR STACK PTR
04450 07F3 3B             RTI
04460 07F4 39       GETRTS RTS
04470               *
04480               ****************************************************
04490               *           END OF SUBROUTINE GETBIT
04500               ****************************************************
04510               *
04520               *
04530               *FUNCTION : ERROR
04540               *INPUTS : MESSAGE STRING POINTED TO BY INDX REG
04550               *OUTPUTS : ERROR MESSAGE TO CONSOLE DEVICE
04560               *CALLS : OUTCR,KEYBD0,DISPLA
04570               *DESTROYS : A,B,X,CC
04580               *PURPOSE : THIS ROUTINE PRINTS ERROR MESSAGES UPON
04590               * ERROR DETECTION BY THE DECOMPRESSION ROUTINES.
04600               *
04610               *
04620 07F5 BD CA8F ERROR  JSR    OUTCR
04630 07F8 CE 09DF        LDX    #ANYKEY  GET ANYKEY MESSAGE
04640 07FB BD CA8F        JSR    OUTCR    PRINT "TO CONTINUE PRESS ANY
04650 07FE BD CA2C        JSR    KEYBD0
04660 0801 0F             SEI             NOW CLR INTR CONDITION
04670 0802 CE 4000        LDX    #$4000   AND RETURN TO DISPLAY
04680 0805 FF E500        STX    DACZRO
04690 0808 FE 061D        LDX    VECSAV   RETRIEV IRQ VEC ADDR
04700 080B FF FFF8        STX    IRQVEC
04710 080E 7E 0100        JMP    DISPLA   RETURN TO "DISPLAY"
04720               *
04730               ****************************************************
04740               *       END ERROR HANDLING ROUTINE
04750               ****************************************************
04760               *
04770               *        MESSAGE STRINGS
04780               *
04790 0811 1A0D   GOMSG  FDB    $1A0D,$0A07
04800 0815 44             FCC    /DECOMPRESSION OVERLAY/
04810 082A 0D0A           FDB    $0D0A,$0D0A
04820 082E 43             FCC    /CURRENT CHANNEL FLAG IS /
04830 0846 0001   CHNASC RMB    1
04840 0847 0D0A           FDB    $0D0A,$0D0A
04850 084B 44             FCC    /DO YOU WISH TO EXECUTE THIS MODULE (
04860 0877 04             FCB    4
04870 0878 0D0A   GOAGIN FDB    $0D0A,$0A07
04880 087C 44             FCC    /DECOMPRESSION AND DISPLAY COMPLETE/
04890 089E 0D0A           FDB    $0D0A,$0D0A
04900 08A2 45             FCC    /ENTER CONTINUATION COMMAND/
```

```
04910 098C 0D0A           FDB    $0D0A
04920 098E 20             FCC    /   X=DISPLAY CHANNEL X ON D-A CHANNE
[illegible]               FDB    $0D0A
[illegible]               FCC    /   Y=DISPLAY CHANNEL Y ON D-A CHANNE
04950 090E 0D0A           FDB    $0D0A
04960 0910 20             FCC    /   Z=DISPLAY CHANNEL Z ON D-A CHANNE
04970 0937 0D0A           FDB    $0D0A,$0D0A
04980 093B 41             FCC    /ANY OTHER KEY RETURNS CONTROL TO "DI
04990 0966 0D0A           FDB    $0D0A,$0D0A
05000 096A 45             FCC    /ENTER COMMAND NOW=/
05010 097C 04             FCB    4
05020 097D 1A07  SYNCMS   FDB    $1A07,$0D0A
05030 0981 53             FCC    /SYNC ERROR DETECTED BY TOLAN-A DECOD
05040 09A7 0D0A           FDB    $0D0A
05050 09A9 04             FCB    4
05060 09AA 1A07  SYNCTI   FDB    $1A07,$0D0A
05070 09AE 53             FCC    /SYNC ERROR DETECTED BY /
05080 09C5 54             FCC    /TOLAN-A DURING TIME FETCH/
05090 09DE 04             FCB    4
05100 09DF 0D0A  ANYKEY   FDB    $0D0A
05110 09E1 54             FCC    /TO CONTINUE PRESS ANY KEY/
05120 09FA 04             FCB    4
05130 09FB 1A07  OVFLMS   FDB    $1A07,$0D0A
05140 09FF 41             FCC    /ACCEL VALUE OVERFLOW DETECTED BY TOL
05150 0A2F 0D0A           FDB    $0D0A
05160 0A31 20             FCC    / IN CHANNEL /
05170 0A3D 0001  OVFLCH   RMB    1
05180 0A3E 04             FCB    4
05190 0A3F 1A07  TIMERR   FDB    $1A07,$0D0A
05200 0A43 54             FCC    /TIME COUNT ERROR DETECTED BY TOLAN-A
05210 0A6F 0D0A           FDB    $0D0A
05220 0A71 54             FCC    /TIME CNT (DELT)=0/
05230 0A82 04             FCB    4
05240                     *
05250                     ********************************************************
05260                     *
05270                     *      END OF DECPRS OVERLAY ROUTINES
05280                     *
05290                     ********************************************************
05300                     *
05310                           END
```

ENTROPY

```
0100 REM ****************************************
0110 REM *
0120 REM *       EKG ENTROPY CALCULATION PROGRAM
0130 REM *
0140 REM ****************************************
0150 REM *
0160 REM *     THIS PROGRAM READS THE FREQUENCY
0170 REM * OF OCCURENCE DATA IN MEMORY FROM AN
0180 REM * EKG DATA COLLECTION AND CALCULATES
0190 REM * THE ENTROPY OF THE X,Y,AND Z DATA
0200 REM * SOURCES.  THE CALCULATED ENTROPY IS
0210 REM * STORED BACK IN THE MEMORY FILE HEADER
0220 REM * IN ASCII. THIS HEADER CAN THEN BE
0230 REM * INSERTED INTO THE DISK FILE USING
0240 REM * MINIDOS.
0250 REM *
0260 REM ****************************************
0270 REM
0280 DIM D(255),N$(8),T(255),M9(6),T$(3)
0290 STRING= 8
0291 DIGITS= 4
0292  T$(1)="X"
0293  T$(2)="Y"
0294  T$(3)="Z"
0295 LINE= 80
0296  L7=6
0297  E1=13507
0313 REM
0325 REM ************************************************
0326 REM * NOW GET FILENAME FROM MEMORY & PRINT IT
0327 REM ************************************************
0328 REM
0330 FOR I=1 TO 8
0340  J=13313+I
0350  N$(I)=CHR$(PEEK(J))
0360  A$=A$+N$(I)
0370 NEXT I
0372  P$=CHR$(12)
0373 PRINT P$
0374 PRINT
0375 PRINT
0376 PRINT
0377 PRINT
0378 PRINT
0380 PRINT
0385 PRINT "EKG ENTROPY CALCULATION"
0386 PRINT
0387 PRINT
0390 PRINT "FILENAME.............";A$
0400 PRINT
0401 PRINT
0402 GOSUB 1696
0403 GOSUB 1900
0404 IF G$ = "S" THEN GO TO 1120
```

ENTROPY

```
0410 REM
0411 REM ************************************************
0412 REM * E1=MEM FILE DESTINATION FOR ASCII RESULTS
0413 REM * 13856=$3620 (HEX)=FLAG NUM USED BY BASIC. DATA REPLACED
0414 REM * 13567=34FF (HEX)=XPDF BUFFER-1
0415 REM * 14079=36FF (HEX)=YPDF BUFFER-1
0416 REM * 14591=38FF (HEX)=ZPDF BUFFER-1
0417 REM * 13500=34BC (HEX)=BASSAV. MEM BUF FOR $3620 DATA SAVE
0419 REM ************************************************
0420 REM
0421 REM
0424 REM
0425 REM ****************************************************
0426 REM *  NOW GET VLU IN BASSAV & STR TO $3620 (HEX)
0427 REM ****************************************************
0428 REM
0429  B=13856
0430  C1=1
0432  N=PEEK(13500)
0433 POKE( B,N)
0435 FOR I=1 TO 255
0436  T(I)=0
0437 NEXT I
0438  T9=0
0439  S2=0
0440  S=0
0441 REM
0442 REM ****************************************************
0443 REM *  NOW BEGIN LOOP TO CALC X,Y,Z & TOTAL ENTROPY
0444 REM ****************************************************
0445 REM
0450 IF C1 <> 1 THEN GO TO 480
0460  K=13567
0480 IF C1 <> 2 THEN GO TO 510
0490  K=14079
0510 IF C1 <> 3 THEN GO TO 540
0520  K=14591
0540 IF C1>= 4 THEN GO TO 800
0542 REM
0543 REM ********************************************************
0544 REM * GET 2 BYTE PDF DATA & MERGE INTO FLTING POINT NUMBER
0545 REM ********************************************************
0546 REM
0550 FOR I=1 TO 509 STEP 2
0560  M=(I+1)/2
0570  D(M)=256*PEEK(I+K)+PEEK(I+K+1)
0572  T(M)=T(M)+D(M)
0590  S=D(M)+S
0592  S2=D(M)+S2
0600 NEXT I
0605  D9=256*PEEK(K+511)+PEEK(K+512)
0606  S=S+D9
0607  T9=T9+D9
0608  S2=S2+D9
```

```
0610  E=0
0612 REM
0613 REM ************************************************************
0614 REM * NOW CALCULATE PROBABILITY OF OCCURRENCE OF SPECIFIC
0615 REM * VLU'S FROM NUM OF OCCURENCES. THEN CALC ENTROPY SUM.
0616 REM ************************************************************
0617 REM
0620 FOR I=1 TO 255
0630  D(I)=D(I)/S
0640 IF D(I)=0 THEN GO TO 660
0650  E=E+(D(I)*(-(LOG(D(I))/.693147)))
0660 NEXT I
0663  D9=D9/S
0664 IF D9=0 THEN GO TO 666
0665  E=E+(D9*(-(LOG(D9)/.693147)))
0666 IF E=0 THEN GOTO 672
0667  C2=8/E
0668 GOTO 683
0672  C2=1
0673 GOTO 680
0674 REM
0675 REM ************************************************************
0676 REM *  PRINT RESULTS OF CALC TO TERMINAL (PRINTER).
0677 REM ************************************************************
0678 REM
0680 PRINT
0681 PRINT "MAXIMUM COMPRESSION FOR CHANNEL ";T$(C1);" IS INFINITE"
0682 GOTO 690
0683 PRINT
0684 PRINT "MAX COMPRESSION RATIO FOR CHANNEL ";T$(C1);" IS ";C2;" ; 1"
0690 PRINT
0700 PRINT "EKG LEAD ";T$(C1);" ENTROPY = ";E;" BITS."
0710 PRINT
0712 REM
0713 REM ************************************************************
0714 REM *  NOW CONVERT CALCULATED ENTROPY TO ASCII & STORE BACK
0715 REM *  IN MEMORY BUFFER
0716 REM ************************************************************
0717 REM
0720 LET E$=STR$(E)
0740 GOSUB 1750
0785  C1=C1+1
0790 GOTO 440
0800 REM
0801 REM ************************************************************
0802 REM * WITH X,Y,Z ENTROPY CALCULATED, NOW CALCULATE TOTAL,
0803 REM * COMBINED ENTROPY BY ADDING X,Y,Z BIN COUNTS & DIVIDING
0804 REM * BY TOTAL SAMPLE COUNT
0805 REM ************************************************************
0806 REM
0807  E5=0
0810 FOR I=1 TO 255
0820  T(I)=T(I)/S2
0822 IF T(I)=0 THEN GO TO 840
```

```
0830  E5=E5+(T(I)*(-(LOG(T(I))/.693147)))
0840 NEXT I
0851  T9=E5/S2
0852 IF T9=0 THEN GO TO 876
0860  E5=E5+(T9*(-(LOG(T9)/.693147)))
0870  C3=8/E5
0872 REM
0873 REM ************************************************************
0874 REM * PRINT COMBINED ENTROPY TO TERM & STR RESULT IN MEM BUFFER
0875 REM ************************************************************
0876 REM
0880 PRINT
0890 PRINT "MAX COMPRESSION RATIO FOR 3 LEAD EKG SYSTEM ";C3;" : 1"
0910 PRINT
0920 PRINT "3 LEAD EKG SOURCE ENTROPY = ";E5;" BITS."
0930 PRINT
0940 LET E$=STR$(E5)
0950 GOSUB 1750
1000 LET E$=STR$(C3)
1005 GOSUB 1750
1010 REM
1020 REM ***********************************************
1030 REM * NOW GET OTHER STATISTICAL VARIABLES &
1040 REM * CALCULATE CHANNEL MAX,MINS,COMPRESSION RATIO
1050 REM * OBTAINED, AND COMPRESSION TIME EFFICIENCY
1060 REM ***********************************************
1070 REM
1080 REM ***********************************************
1082 REM *  START WITH ACHIEVED COMPRESSION RATIO. THEN
1090 REM *  PRINT  THE TIME COMPRESSION EFFICIENCY. THIS IS
1091 REM *  THE PERCENTAGE OF THE TIME AVALIABLE THAT WAS USED
1092 REM *  TO COLLECT,COMPRESS, & CALCULATE STAT VARIABLES.
1100 REM ***********************************************
1110 REM
1120 DIGITS= 4
1130  L7=6
1140  M4=PEEK(13483)+PEEK(13482)*256+PEEK(13481)*256*256
1150  D4=PEEK(13487)+PEEK(13486)*256+PEEK(13485)*256*256
1160  D4=PEEK(13484)*256*256*256+D4
1170  C4=D4/M4
1180 LET E$=STR$(C4)
1190 GOSUB 1750
1200 PRINT
1201 PRINT "COMPRESSION RATIO ACHIEVED = ";C4;" : 1"
1210 IF G$="S" THEN GOTO 1260
1211 REM
1212 REM ***********************************************
1213 REM * CALCULATE OVERALL COMPRESSION EFFICIENCY
1214 REM ***********************************************
1215 REM
1216 DIGITS= 1
1217  C6=((C4-1)/(C3-1))*100
1218 PRINT
1219 PRINT "ACHIEVED COMPRESSION EFFICIENCY = ";C6;" %"
```

```
1220 LET E$=STR$(C6)
1221  L7=4
1222 GOSUB 1750
1229 REM
1230 REM ************************************************
1240 REM *  NOW CALCULATE COMPRESSION TIME EFFECIENCY
1250 REM ************************************************
1252 REM
1260  S4=PEEK(13461)+PEEK(13460)*256
1262  L4=PEEK(13462)
1264  L5=PEEK(13459)+PEEK(13458)*256+PEEK(13457)*256*256
1266  L5=PEEK(13456)*256*256*256+L5
1270  T4=(1-(L5/(L4*S4)))*100
1272 DIGITS= 1
1274  L7=4
1276 LET E$=STR$(T4)
1278 GOSUB 1750
1280 PRINT
1282 PRINT "COLLECTION TIME EFFICIENCY = ";T4;" %"
1284 PRINT
1290 REM
1291 REM ************************************************
1292 REM * CALCULATE COLLECTION DURATION
1293 REM ************************************************
1294 REM
1295  T6=S4/S8
1296 PRINT
1297 PRINT "COLLECTION DURATION = ";T6;" SECONDS"
1298 PRINT "AT A SAMPLE RATE OF ";S8;" HZ"
1299 LET E$=STR$(T6)
1300 GOSUB 1750
1301 REM
1302 REM ************************************************
1310 REM *  RETRIEVE DATA MAX & MINS & CALC VOLTS
1320 REM ************************************************
1330 REM
1339 DIGITS= 5
1340 FOR I=6 TO 1 STEP -1
1350  M9(I)=PEEK(13460+I*3)
1360 IF M9(I) > 127 THEN GOTO 1390
1370  M9(I)=M9(I)*.0390625
1380 GOTO 1400
1390  M9(I)=-(10-M9(I)*.0390625)
1400 NEXT I
1401  E1=13408
1402 FOR I4=5 TO 1 STEP -2
1403 LET E$=STR$(M9(I4))
1404  L7=7
1405 GOSUB 1750
1406  I3=I4+1
1407 LET E$=STR$(M9(I3))
1408 GOSUB 1750
1409 NEXT I4
1410 REM
```

```
1420 REM ************************************************
1430 REM *  NOW PRINT MAX & MINS TO TERM (PRINTER)
1440 REM ************************************************
1450 REM
1460 PRINT
1470 FOR I=1 TO 6 STEP 2
1480 PRINT
1490 PRINT "CHANNEL ";T$((I+1)/2);" MINIMUM = ";M9(7-I);" VOLTS."
1500 PRINT
1510 PRINT "CHANNEL ";T$((I+1)/2);" MAXIMUM = ";M9(6-I);" VOLTS."
1520 PRINT
1530 NEXT I
1540 REM
1550 REM ************************************************
1560 REM *  FINISH WITH A PRINT OF NUM OF BITS USED TO
1570 REM *  STORE X,Y,Z IN MEM WITH THIS COMPRESSION TYPE
1580 REM ************************************************
1590 REM
1600  X3=PEEK(13490)+PEEK(13489)*256+PEEK(13488)*256*256
1610  Y3=PEEK(13493)+PEEK(13492)*256+PEEK(13491)*256*256
1620  Z3=PEEK(13496)+PEEK(13495)*256+PEEK(13494)*256*256
1630  T3=PEEK(13499)+PEEK(13498)*256+PEEK(13497)*256*256
1632 DIGITS= 0
1640 PRINT
1650 PRINT "NUMBER OF BITS USED TO STORE CHANNEL X =";X3
1652 PRINT
1660 PRINT "NUMBER OF BITS USED TO STORE CHANNEL Y =";Y3
1662 PRINT
1670 PRINT "NUMBER OF BITS USED TO STORE CHANNEL Z =";Z3
1672 PRINT
1680 PRINT "NUMBER OF BITS USED TO STORE TIME =";T3
1682 PRINT
1683 PRINT
1690 GOTO 2000
1691 REM
1692 REM ************************************************
1693 REM * SUB TO PRINT OUT COMPRESSION TYPE
1694 REM ************************************************
1695 REM
1696  H=PEEK(13312)*256+PEEK(13313)
1697 IF H=20035 THEN H$(1)="NOT COMP"
1698 IF H=20035 THEN H$(2)="RESSED  "
1699 IF H=21569 THEN H$(1)="TOLAN A "
1700 IF H=21569 THEN H$(2)="        "
1701 IF H=21570 THEN H$(1)="TOLAN B "
1702 IF H=21570 THEN H$(2)="        "
1703 IF H=17487 THEN H$(1)="DOWER   "
1704 IF H=17487 THEN H$(2)="        "
1705 IF H=21584 THEN H$(1)="TURNING "
1706 IF H=21584 THEN H$(2)="POINT   "
1707 IF H=18766 THEN H$(1)="2ND ORD "
1708 IF H=18766 THEN H$(2)="INTERPOL"
1709 PRINT "COMPRESSION USED......";H$(1);H$(2)
1710 PRINT
```

```
1711 RETURN
1720 REM
1730 REM ***********************************************
1732 REM * SUBROUTINE TO WRITE ASCII DATA IN E$ TO MEM
1733 REM * LOCATION POINTED TO BY E1.  WILL WRITE 8 CHAR
1734 REM * STRINGS TO MEMORY.
1739 REM ***********************************************
1740 REM
1750 FOR I=1 TO L7
1760  F$=MID$(E$,I,L7)
1770  F=ASC(F$)
1780 POKE( E1,F)
1781  E1=E1+1
1790 NEXT I
1800 POKE( E1,4)
1801  E1=E1+1
1810 RETURN
1820 REM
1830 REM ***********************************************
1840 REM * SUBROUTINE TO PROMPT COMMAND FOR SHORT RUN OR LONG
1850 REM * RUN. SHORT DOES NOT CALCULATE ENTROPY OR COMPRESSION
1860 REM * RATIOS.
1870 REM ************************************************************
1880 REM
1900 PRINT
1910 PRINT "ENTER S FOR SHORT RUN (NO ENTROPY CALCULATED)"
1920 PRINT "ENTER L FOR LONG RUN (WITH ENTROPY CALCULATED)"
1930 INPUT G$
1932 PRINT "ENTER COLLECTION SAMPLE RATE (IE. 500)"
1934 INPUT O$
1935  S8=VAL(O$)
1940 RETURN
1950 REM
1960 REM ***************************************************
1970 REM *  END OF ENTROPY CALCULATION PROGRAM
1980 REM ***************************************************
1990 REM
2000 PRINT
2010 PRINT
2020 PRINT "ENTROPY CALCULATION & STATISTICS PRINTOUT COMPLETE"
2030 END
```

Appendix D

This appendix contains a listing of the data compressed by the TOLAN-A8 module for the thesis experiment.

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1359PA
SUBJECT . . . . . PARNELL
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1359
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 2.4871   BITS
CHANNEL Y ENTROPY . . . . 2.4751   BITS
CHANNEL Z ENTROPY . . . . 2.4217   BITS
TOTAL SOURCE ENTROPY. . . 2.4796   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 3.2263   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 2.2595   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 56.5  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 77.8  % SMP INTERVAL
COLLECTION DURATION . . . 26.2  SECONDS
CHANNEL X MAXIMUM . . . . 2.50000  VOLTS
CHANNEL X MINIMUM . . . . -0.1171  VOLTS
CHANNEL Y MAXIMUM . . . . 2.50000  VOLTS
CHANNEL Y MINIMUM . . . . -0.1171  VOLTS
CHANNEL Z MAXIMUM . . . . 2.50000  VOLTS
CHANNEL Z MINIMUM . . . . -0.1562  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 3336 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0001B9DB (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 40 (HEX) AT SAMPLE NUMBER 0DA7 (HEX)
XMIN= FD (HEX) AT SAMPLE NUMBER 0171 (HEX)
YMAX= 40 (HEX) AT SAMPLE NUMBER 0DA7 (HEX)
YMIN= FD (HEX) AT SAMPLE NUMBER 0171 (HEX)
ZMAX= 40 (HEX) AT SAMPLE NUMBER 0DA7 (HEX)
ZMIN= FC (HEX) AT SAMPLE NUMBER 0171 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00040000 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 0004CD10 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006652 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 0066A8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006160 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 010000 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER HEX BITS AVALIABLE
```

FILE TA1359PA
CHANNEL X AMPLITUDE DISTRIBUTION
BIN VALUE NUMBER OF OCCURRENCES

80.....0000 (HEX)
81.....0000 (HEX)
82.....0000 (HEX)
83.....0000 (HEX)
84.....0000 (HEX)
85.....0000 (HEX)
86.....0000 (HEX)
87.....0000 (HEX)
88.....0000 (HEX)
89.....0000 (HEX)
8A.....0000 (HEX)
8B.....0000 (HEX)
8C.....0000 (HEX)
8D.....0000 (HEX)
8E.....0000 (HEX)
8F.....0000 (HEX)
90.....0000 (HEX)
91.....0000 (HEX)
92.....0000 (HEX)
93.....0000 (HEX)
94.....0000 (HEX)
95.....0000 (HEX)
96.....0000 (HEX)
97.....0000 (HEX)
98.....0000 (HEX)
99.....0000 (HEX)
9A.....0000 (HEX)
9B.....0000 (HEX)
9C.....0000 (HEX)
9D.....0000 (HEX)
9E.....0000 (HEX)
9F.....0000 (HEX)
A0.....0000 (HEX)
A1.....0000 (HEX)
A2.....0000 (HEX)
A3.....0000 (HEX)
A4.....0000 (HEX)
A5.....0000 (HEX)
A6.....0000 (HEX)
A7.....0000 (HEX)
A8.....0000 (HEX)
A9.....0000 (HEX)
AA.....0000 (HEX)
AB.....0000 (HEX)
AC.....0000 (HEX)
AD.....0000 (HEX)
AE.....0000 (HEX)
AF.....0000 (HEX)
B0.....0000 (HEX)
B1.....0000 (HEX)
B2.....0000 (HEX)

```
B3.............0000 (HEX)
B4.............0000 (HEX)
B5.............0000 (HEX)
B6.............0000 (HEX)
B7.............0000 (HEX)
B8.............0000 (HEX)
B9.............0000 (HEX)
BA.............0000 (HEX)
BB.............0000 (HEX)
BC.............0000 (HEX)
BD.............0000 (HEX)
BE.............0000 (HEX)
BF.............0000 (HEX)
C0.............0000 (HEX)
C1.............0000 (HEX)
C2.............0000 (HEX)
C3.............0000 (HEX)
C4.............0000 (HEX)
C5.............0000 (HEX)
C6.............0000 (HEX)
C7.............0000 (HEX)
C8.............0000 (HEX)
C9.............0000 (HEX)
CA.............0000 (HEX)
CB.............0000 (HEX)
CC.............0000 (HEX)
CD.............0000 (HEX)
CE.............0000 (HEX)
CF.............0000 (HEX)
D0.............0000 (HEX)
D1.............0000 (HEX)
D2.............0000 (HEX)
D3.............0000 (HEX)
D4.............0000 (HEX)
D5.............0000 (HEX)
D6.............0000 (HEX)
D7.............0000 (HEX)
D8.............0000 (HEX)
D9.............0000 (HEX)
DA.............0000 (HEX)
DB.............0000 (HEX)
DC.............0000 (HEX)
DD.............0000 (HEX)
DE.............0000 (HEX)
DF.............0000 (HEX)
E0.............0000 (HEX)
E1.............0000 (HEX)
E2.............0000 (HEX)
E3.............0000 (HEX)
E4.............0000 (HEX)
E5.............0000 (HEX)
E6.............0000 (HEX)
E7.............0000 (HEX)
E8.............0000 (HEX)
```

```
E9.............0000 (HEX)
FA.............0000 (HEX)
FB.............0000 (HEX)
EC.............0000 (HEX)
ED.............0000 (HEX)
EE.............0000 (HEX)
EF.............0000 (HEX)
F0.............0000 (HEX)
F1.............0000 (HEX)
F2.............0001 (HEX)
F3.............0001 (HEX)
F4.............0003 (HEX)
F5.............000C (HEX)
F6.............0012 (HEX)
F7.............0016 (HEX)
F8.............000F (HEX)
F9.............000E (HEX)
FA.............000E (HEX)
FB.............000E (HEX)
FC.............001B (HEX)
FD.............004D (HEX)
FE.............0260 (HEX)
FF.............0961 (HEX)
00.............06A2 (HEX)
01.............09A5 (HEX)
02.............024A (HEX)
03.............005F (HEX)
04.............0027 (HEX)
05.............0011 (HEX)
06.............0003 (HEX)
07.............0008 (HEX)
08.............0008 (HEX)
09.............0004 (HEX)
0A.............0001 (HEX)
0B.............0001 (HEX)
0C.............0006 (HEX)
0D.............0003 (HEX)
0E.............0003 (HEX)
0F.............000A (HEX)
10.............0009 (HEX)
11.............0000 (HEX)
12.............0003 (HEX)
13.............0002 (HEX)
14.............0000 (HEX)
15.............0000 (HEX)
16.............0000 (HEX)
17.............0000 (HEX)
18.............0000 (HEX)
19.............0000 (HEX)
1A.............0000 (HEX)
1B.............0000 (HEX)
1C.............0000 (HEX)
1D.............0000 (HEX)
1E.............0000 (HEX)
```

```
1F.............0000 (HEX)
20.............0000 (HEX)
21.............0000 (HEX)
22.............0000 (HEX)
23.............0000 (HEX)
24.............0000 (HEX)
25.............0000 (HEX)
26.............0000 (HEX)
27.............0000 (HEX)
28.............0000 (HEX)
29.............0000 (HEX)
2A.............0000 (HEX)
2B.............0000 (HEX)
2C.............0000 (HEX)
2D.............0000 (HEX)
2E.............0000 (HEX)
2F.............0000 (HEX)
30.............0000 (HEX)
31.............0000 (HEX)
32.............0000 (HEX)
33.............0000 (HEX)
34.............0000 (HEX)
35.............0000 (HEX)
36.............0000 (HEX)
37.............0000 (HEX)
38.............0000 (HEX)
39.............0000 (HEX)
3A.............0000 (HEX)
3B.............0000 (HEX)
3C.............0000 (HEX)
3D.............0000 (HEX)
3E.............0000 (HEX)
3F.............0000 (HEX)
40.............0000 (HEX)
41.............0000 (HEX)
42.............0000 (HEX)
43.............0000 (HEX)
44.............0000 (HEX)
45.............0000 (HEX)
46.............0000 (HEX)
47.............0000 (HEX)
48.............0000 (HEX)
49.............0000 (HEX)
4A.............0000 (HEX)
4B.............0000 (HEX)
4C.............0000 (HEX)
4D.............0000 (HEX)
4E.............0000 (HEX)
4F.............0000 (HEX)
50.............0000 (HEX)
51.............0000 (HEX)
52.............0000 (HEX)
53.............0000 (HEX)
54.............0000 (HEX)
```

```
55.............0000 (HEX)
56.............0000 (HEX)
57.............0000 (HEX)
58.............0000 (HEX)
59.............0000 (HEX)
5A.............0000 (HEX)
5B.............0000 (HEX)
5C.............0000 (HEX)
5D.............0000 (HEX)
5E.............0000 (HEX)
5F.............0000 (HEX)
60.............0000 (HEX)
61.............0000 (HEX)
62.............0000 (HEX)
63.............0000 (HEX)
64.............0000 (HEX)
65.............0000 (HEX)
66.............0000 (HEX)
67.............0000 (HEX)
68.............0000 (HEX)
69.............0000 (HEX)
6A.............0000 (HEX)
6B.............0000 (HEX)
6C.............0000 (HEX)
6D.............0000 (HEX)
6E.............0000 (HEX)
6F.............0000 (HEX)
70.............0000 (HEX)
71.............0000 (HEX)
72.............0000 (HEX)
73.............0000 (HEX)
74.............0000 (HEX)
75.............0000 (HEX)
76.............0000 (HEX)
77.............0000 (HEX)
78.............0000 (HEX)
79.............0000 (HEX)
7A.............0000 (HEX)
7B.............0000 (HEX)
7C.............0000 (HEX)
7D.............0000 (HEX)
7E.............0000 (HEX)
7F.............0000 (HEX)
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1413LU
SUBJECT . . . . . LUTZ
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1413
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.7826   BITS
CHANNEL Y ENTROPY . . . . 3.7950   BITS
CHANNEL Z ENTROPY . . . . 3.8149   BITS
TOTAL SOURCE ENTROPY. . . 3.8028   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.1036   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.2519   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 22.8  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 95.4  % SMP INTERVAL
COLLECTION DURATION . . . 14.5  SECONDS
CHANNEL X MAXIMUM . . . . 1.79687  VOLTS
CHANNEL X MINIMUM . . . . -1.0156  VOLTS
CHANNEL Y MAXIMUM . . . . 1.75781  VOLTS
CHANNEL Y MINIMUM . . . . -1.0156  VOLTS
CHANNEL Z MAXIMUM . . . . 1.75781  VOLTS
CHANNEL Z MINIMUM . . . . -1.0156  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON. LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 1C60 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 000031E3 (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 2E (HEX) AT SAMPLE NUMBER 0E7F (HEX)
XMIN= E6 (HEX) AT SAMPLE NUMBER 11C4 (HEX)
YMAX= 2D (HEX) AT SAMPLE NUMBER 0E7F (HEX)
YMIN= E6 (HEX) AT SAMPLE NUMBER 11C4 (HEX)
ZMAX= 2D (HEX) AT SAMPLE NUMBER 0E7F (HEX)
ZMIN= E6 (HEX) AT SAMPLE NUMBER 11C4 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00032040 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 0002A900 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 007786 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 0077F8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 00764C (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETERS = 00CA50 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

FILE TA1413LU
CHANNEL X AMPLITUDE DISTRIBUTION
DATA VALUE NUMBER OF OCCURENCES

80.....0000 (HEX)
81.....0000 (HEX)
82.....0000 (HEX)
83.....0000 (HEX)
84.....0000 (HEX)
85.....0000 (HEX)
86.....0000 (HEX)
87.....0000 (HEX)
88.....0000 (HEX)
89.....0000 (HEX)
8A.....0000 (HEX)
8B.....0000 (HEX)
8C.....0000 (HEX)
8D.....0000 (HEX)
8E.....0000 (HEX)
8F.....0000 (HEX)
90.....0000 (HEX)
91.....0000 (HEX)
92.....0000 (HEX)
93.....0000 (HEX)
94.....0000 (HEX)
95.....0000 (HEX)
96.....0000 (HEX)
97.....0000 (HEX)
98.....0000 (HEX)
99.....0000 (HEX)
9A.....0000 (HEX)
9B.....0000 (HEX)
9C.....0000 (HEX)
9D.....0000 (HEX)
9E.....0000 (HEX)
9F.....0000 (HEX)
A0.....0000 (HEX)
A1.....0000 (HEX)
A2.....0000 (HEX)
A3.....0000 (HEX)
A4.....0000 (HEX)
A5.....0000 (HEX)
A6.....0000 (HEX)
A7.....0000 (HEX)
A8.....0000 (HEX)
A9.....0000 (HEX)
AA.....0000 (HEX)
AB.....0000 (HEX)
AC.....0000 (HEX)
AD.....0000 (HEX)
AE.....0000 (HEX)
AF.....0000 (HEX)
B0.....0000 (HEX)
B1.....0000 (HEX)
B2.....0000 (HEX)

```
B3.............0000 (HEX)
B4.............0000 (HEX)
B5.............0000 (HEX)
B6.............0000 (HEX)
B7.............0000 (HEX)
B8.............0000 (HEX)
B9.............0000 (HEX)
BA.............0000 (HEX)
BB.............0000 (HEX)
BC.............0000 (HEX)
BD.............0000 (HEX)
BE.............0000 (HEX)
BF.............0000 (HEX)
C0.............0000 (HEX)
C1.............0000 (HEX)
C2.............0000 (HEX)
C3.............0000 (HEX)
C4.............0000 (HEX)
C5.............0000 (HEX)
C6.............0000 (HEX)
C7.............0000 (HEX)
C8.............0000 (HEX)
C9.............0000 (HEX)
CA.............0000 (HEX)
CB.............0000 (HEX)
CC.............0000 (HEX)
CD.............0000 (HEX)
CE.............0000 (HEX)
CF.............0000 (HEX)
D0.............0000 (HEX)
D1.............0000 (HEX)
D2.............0000 (HEX)
D3.............0000 (HEX)
D4.............0000 (HEX)
D5.............0000 (HEX)
D6.............0000 (HEX)
D7.............0000 (HEX)
D8.............0000 (HEX)
D9.............0000 (HEX)
DA.............0000 (HEX)
DB.............0000 (HEX)
DC.............0000 (HEX)
DD.............0000 (HEX)
DE.............0000 (HEX)
DF.............0000 (HEX)
E0.............0000 (HEX)
E1.............0000 (HEX)
E2.............0000 (HEX)
E3.............0000 (HEX)
E4.............0000 (HEX)
E5.............0000 (HEX)
E6.............0000 (HEX)
E7.............0000 (HEX)
E8.............0000 (HEX)
```

E9.....0000 (HEX)
EA.....0001 (HEX)
EB.....0000 (HEX)
EC.....0000 (HEX)
ED.....0000 (HEX)
EE.....0000 (HEX)
EF.....0001 (HEX)
F0.....0000 (HEX)
F1.....0000 (HEX)
F2.....0001 (HEX)
F3.....0006 (HEX)
F4.....0007 (HEX)
F5.....0012 (HEX)
F6.....0012 (HEX)
F7.....0022 (HEX)
F8.....0035 (HEX)
F9.....0050 (HEX)
FA.....0082 (HEX)
FB.....00CC (HEX)
FC.....0136 (HEX)
FD.....01D9 (HEX)
FE.....02EF (HEX)
FF.....03D0 (HEX)
00.....0164 (HEX)
01.....040A (HEX)
02.....02BB (HEX)
03.....01E4 (HEX)
04.....0132 (HEX)
05.....00AD (HEX)
06.....006B (HEX)
07.....0058 (HEX)
08.....0041 (HEX)
09.....001F (HEX)
0A.....0015 (HEX)
0B.....0013 (HEX)
0C.....000C (HEX)
0D.....000A (HEX)
0E.....0002 (HEX)
0F.....0002 (HEX)
10.....0000 (HEX)
11.....0000 (HEX)
12.....0001 (HEX)
13.....0000 (HEX)
14.....0000 (HEX)
15.....0000 (HEX)
16.....0000 (HEX)
17.....0001 (HEX)
18.....0000 (HEX)
19.....0000 (HEX)
1A.....0000 (HEX)
1B.....0000 (HEX)
1C.....0000 (HEX)
1D.....0000 (HEX)
1E.....0000 (HEX)

```
1F............0000 (HEX)
20............0000 (HEX)
21............0000 (HEX)
22............0000 (HEX)
23............0000 (HEX)
24............0000 (HEX)
25............0000 (HEX)
26............0000 (HEX)
27............0000 (HEX)
28............0000 (HEX)
29............0000 (HEX)
2A............0000 (HEX)
2B............0000 (HEX)
2C............0000 (HEX)
2D............0000 (HEX)
2E............0000 (HEX)
2F............0000 (HEX)
30............0000 (HEX)
31............0000 (HEX)
32............0000 (HEX)
33............0000 (HEX)
34............0000 (HEX)
35............0000 (HEX)
36............0000 (HEX)
37............0000 (HEX)
38............0000 (HEX)
39............0000 (HEX)
3A............0000 (HEX)
3B............0000 (HEX)
3C............0000 (HEX)
3D............0000 (HEX)
3E............0000 (HEX)
3F............0000 (HEX)
40............0000 (HEX)
41............0000 (HEX)
42............0000 (HEX)
43............0000 (HEX)
44............0000 (HEX)
45............0000 (HEX)
46............0000 (HEX)
47............0000 (HEX)
48............0000 (HEX)
49............0000 (HEX)
4A............0000 (HEX)
4B............0000 (HEX)
4C............0000 (HEX)
4D............0000 (HEX)
4E............0000 (HEX)
4F............0000 (HEX)
50............0000 (HEX)
51............0000 (HEX)
52............0000 (HEX)
53............0000 (HEX)
54............0000 (HEX)
```

```
55.............0000 (HEX)
56.............0000 (HEX)
57.............0000 (HEX)
58.............0000 (HEX)
59.............0000 (HEX)
5A.............0000 (HEX)
5B.............0000 (HEX)
5C.............0000 (HEX)
5D.............0000 (HEX)
5E.............0000 (HEX)
5F.............0000 (HEX)
60.............0000 (HEX)
61.............0000 (HEX)
62.............0000 (HEX)
63.............0000 (HEX)
64.............0000 (HEX)
65.............0000 (HEX)
66.............0000 (HEX)
67.............0000 (HEX)
68.............0000 (HEX)
69.............0000 (HEX)
6A.............0000 (HEX)
6B.............0000 (HEX)
6C.............0000 (HEX)
6D.............0000 (HEX)
6E.............0000 (HEX)
6F.............0000 (HEX)
70.............0000 (HEX)
71.............0000 (HEX)
72.............0000 (HEX)
73.............0000 (HEX)
74.............0000 (HEX)
75.............0000 (HEX)
76.............0000 (HEX)
77.............0000 (HEX)
78.............0000 (HEX)
79.............0000 (HEX)
7A.............0000 (HEX)
7B.............0000 (HEX)
7C.............0000 (HEX)
7D.............0000 (HEX)
7E.............0000 (HEX)
7F.............0000 (HEX)
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1448L
[illegible] . . . . . [illegible]
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1449
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 2.9299   BITS
CHANNEL Y ENTROPY . . . . 2.9243   BITS
CHANNEL Z ENTROPY . . . . 2.9564   BITS
TOTAL SOURCE ENTROPY. . . 2.9436   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.7176   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.6008   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 34.9  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 90.6  % SMP INTERVAL
COLLECTION DURATION . . . 18.5  SECONDS
CHANNEL X MAXIMUM . . . . 1.79687  VOLTS
CHANNEL X MINIMUM . . . . -0.7421  VOLTS
CHANNEL Y MAXIMUM . . . . 1.79687  VOLTS
CHANNEL Y MINIMUM . . . . -0.7421  VOLTS
CHANNEL Z MAXIMUM . . . . 1.79687  VOLTS
CHANNEL Z MINIMUM . . . . -0.7812  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON. LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2448 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0000845C (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 2E (HEX) AT SAMPLE NUMBER 0D0A (HEX)
XMIN= ED (HEX) AT SAMPLE NUMBER 1D0A (HEX)
YMAX= 2E (HEX) AT SAMPLE NUMBER 0D0A (HEX)
YMIN= ED (HEX) AT SAMPLE NUMBER 1D0A (HEX)
ZMAX= 2E (HEX) AT SAMPLE NUMBER 0D0A (HEX)
ZMIN= EC (HEX) AT SAMPLE NUMBER 1D0A (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 0003A[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 000366C0 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 0069AA (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 00693E (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006739 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00EB98 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

FILE TA1448LT
CHANNEL X AMPLITUDE DISTRIBUTION

DATA VALUE	NUMBER OF OCCURENCES
80	0000 (HEX)
81	0000 (HEX)
82	0000 (HEX)
83	0000 (HEX)
84	0000 (HEX)
85	0000 (HEX)
86	0000 (HEX)
87	0000 (HEX)
88	0000 (HEX)
89	0000 (HEX)
8A	0000 (HEX)
8B	0000 (HEX)
8C	0000 (HEX)
8D	0000 (HEX)
8E	0000 (HEX)
8F	0000 (HEX)
90	0000 (HEX)
91	0000 (HEX)
92	0000 (HEX)
93	0000 (HEX)
94	0000 (HEX)
95	0000 (HEX)
96	0000 (HEX)
97	0000 (HEX)
98	0000 (HEX)
99	0000 (HEX)
9A	0000 (HEX)
9B	0000 (HEX)
9C	0000 (HEX)
9D	0000 (HEX)
9E	0000 (HEX)
9F	0000 (HEX)
A0	0000 (HEX)
A1	0000 (HEX)
A2	0000 (HEX)
A3	0000 (HEX)
A4	0000 (HEX)
A5	0000 (HEX)
A6	0000 (HEX)
A7	0000 (HEX)
A8	0000 (HEX)
A9	0000 (HEX)
AA	0000 (HEX)
AB	0000 (HEX)
AC	0000 (HEX)
AD	0000 (HEX)
AE	0000 (HEX)
AF	0000 (HEX)
B0	0000 (HEX)
B1	0000 (HEX)
B2	0000 (HEX)

```
B3.............0000 (HEX)
B4.............0000 (HEX)
B5.............0000 (HEX)
B6.............0000 (HEX)
B7.............0000 (HEX)
B8.............0000 (HEX)
B9.............0000 (HEX)
BA.............0000 (HEX)
BB.............0000 (HEX)
BC.............0000 (HEX)
BD.............0000 (HEX)
BE.............0000 (HEX)
BF.............0000 (HEX)
C0.............0000 (HEX)
C1.............0000 (HEX)
C2.............0000 (HEX)
C3.............0000 (HEX)
C4.............0000 (HEX)
C5.............0000 (HEX)
C6.............0000 (HEX)
C7.............0000 (HEX)
C8.............0000 (HEX)
C9.............0000 (HEX)
CA.............0000 (HEX)
CB.............0000 (HEX)
CC.............0000 (HEX)
CD.............0000 (HEX)
CE.............0000 (HEX)
CF.............0000 (HEX)
D0.............0000 (HEX)
D1.............0000 (HEX)
D2.............0000 (HEX)
D3.............0000 (HEX)
D4.............0000 (HEX)
D5.............0000 (HEX)
D6.............0000 (HEX)
D7.............0000 (HEX)
D8.............0000 (HEX)
D9.............0000 (HEX)
DA.............0000 (HEX)
DB.............0000 (HEX)
DC.............0000 (HEX)
DD.............0000 (HEX)
DE.............0000 (HEX)
DF.............0000 (HEX)
E0.............0000 (HEX)
E1.............0000 (HEX)
E2.............0000 (HEX)
E3.............0000 (HEX)
E4.............0000 (HEX)
E5.............0000 (HEX)
E6.............0000 (HEX)
E7.............0000 (HEX)
E8.............0000 (HEX)
```

E9.....0000 (HEX)
EA.....0000 (HEX)
EB.....0000 (HEX)
EC.....0000 (HEX)
ED.....0000 (HEX)
EE.....0000 (HEX)
EF.....0000 (HEX)
F0.....0000 (HEX)
F1.....0000 (HEX)
F2.....0000 (HEX)
F3.....0000 (HEX)
F4.....0000 (HEX)
F5.....0000 (HEX)
F6.....0002 (HEX)
F7.....0001 (HEX)
F8.....0003 (HEX)
F9.....0007 (HEX)
FA.....001C (HEX)
FB.....0034 (HEX)
FC.....008F (HEX)
FD.....0180 (HEX)
FE.....03F7 (HEX)
FF.....06A2 (HEX)
00.....030D (HEX)
01.....0714 (HEX)
02.....03F0 (HEX)
03.....018A (HEX)
04.....008A (HEX)
05.....002A (HEX)
06.....0013 (HEX)
07.....0006 (HEX)
08.....0002 (HEX)
09.....0002 (HEX)
0A.....0002 (HEX)
0B.....0000 (HEX)
0C.....0000 (HEX)
0D.....0000 (HEX)
0E.....0000 (HEX)
0F.....0000 (HEX)
10.....0000 (HEX)
11.....0000 (HEX)
12.....0000 (HEX)
13.....0000 (HEX)
14.....0000 (HEX)
15.....0000 (HEX)
16.....0000 (HEX)
17.....0000 (HEX)
18.....0000 (HEX)
19.....0000 (HEX)
1A.....0000 (HEX)
1B.....0000 (HEX)
1C.....0000 (HEX)
1D.....0000 (HEX)
1E.....0000 (HEX)

```
1F.............0000 (HEX)
20.............0000 (HEX)
21.............0000 (HEX)
22.............0000 (HEX)
23.............0000 (HEX)
24.............0000 (HEX)
25.............0000 (HEX)
26.............0000 (HEX)
27.............0000 (HEX)
28.............0000 (HEX)
29.............0000 (HEX)
2A.............0000 (HEX)
2B.............0000 (HEX)
2C.............0000 (HEX)
2D.............0000 (HEX)
2E.............0000 (HEX)
2F.............0000 (HEX)
30.............0000 (HEX)
31.............0000 (HEX)
32.............0000 (HEX)
33.............0000 (HEX)
34.............0000 (HEX)
35.............0000 (HEX)
36.............0000 (HEX)
37.............0000 (HEX)
38.............0000 (HEX)
39.............0000 (HEX)
3A.............0000 (HEX)
3B.............0000 (HEX)
3C.............0000 (HEX)
3D.............0000 (HEX)
3E.............0000 (HEX)
3F.............0000 (HEX)
40.............0000 (HEX)
41.............0000 (HEX)
42.............0000 (HEX)
43.............0000 (HEX)
44.............0000 (HEX)
45.............0000 (HEX)
46.............0000 (HEX)
47.............0000 (HEX)
48.............0000 (HEX)
49.............0000 (HEX)
4A.............0000 (HEX)
4B.............0000 (HEX)
4C.............0000 (HEX)
4D.............0000 (HEX)
4E.............0000 (HEX)
4F.............0000 (HEX)
50.............0000 (HEX)
51.............0000 (HEX)
52.............0000 (HEX)
53.............0000 (HEX)
54.............0000 (HEX)
```

```
55.............0000 (HEX)
56.............0000 (HEX)
57.............0000 (HEX)
58.............0000 (HEX)
59.............0000 (HEX)
5A.............0000 (HEX)
5B.............0000 (HEX)
5C.............0000 (HEX)
5D.............0000 (HEX)
5E.............0000 (HEX)
5F.............0000 (HEX)
60.............0000 (HEX)
61.............0000 (HEX)
62.............0000 (HEX)
63.............0000 (HEX)
64.............0000 (HEX)
65.............0000 (HEX)
66.............0000 (HEX)
67.............0000 (HEX)
68.............0000 (HEX)
69.............0000 (HEX)
6A.............0000 (HEX)
6B.............0000 (HEX)
6C.............0000 (HEX)
6D.............0000 (HEX)
6E.............0000 (HEX)
6F.............0000 (HEX)
70.............0000 (HEX)
71.............0000 (HEX)
72.............0000 (HEX)
73.............0000 (HEX)
74.............0000 (HEX)
75.............0000 (HEX)
76.............0000 (HEX)
77.............0000 (HEX)
78.............0000 (HEX)
79.............0000 (HEX)
7A.............0000 (HEX)
7B.............0000 (HEX)
7C.............0000 (HEX)
7D.............0000 (HEX)
7E.............0000 (HEX)
7F.............0000 (HEX)
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1545T
[illegible] . . . . . [illegible]
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1546
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.2801   BITS
CHANNEL Y ENTROPY . . . . 3.2716   BITS
CHANNEL Z ENTROPY . . . . 3.3188   BITS
TOTAL SOURCE ENTROPY. . . 3.3026   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.4223   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.3617   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 25.4  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 93.9  % SMP INTERVAL
COLLECTION DURATION . . . 15.8  SECONDS
CHANNEL X MAXIMUM . . . . 3.59375  VOLTS
CHANNEL X MINIMUM . . . . -1.4453  VOLTS
CHANNEL Y MAXIMUM . . . . 3.59375  VOLTS
CHANNEL Y MINIMUM . . . . -1.4453  VOLTS
CHANNEL Z MAXIMUM . . . . 3.55468  VOLTS
CHANNEL Z MINIMUM . . . . -1.4453  VOLTS
COMMENTS. . . . . X,Y,Z, IN COMMON, LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 1EDD (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 26 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0000470E (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 5C (HEX) AT SAMPLE NUMBER 03DD (HEX)
XMIN= DB (HEX) AT SAMPLE NUMBER 145D (HEX)
YMAX= 5C (HEX) AT SAMPLE NUMBER 03DD (HEX)
YMIN= DB (HEX) AT SAMPLE NUMBER 145D (HEX)
ZMAX= 5B (HEX) AT SAMPLE NUMBER 03DD (HEX)
ZMIN= DB (HEX) AT SAMPLE NUMBER 145D (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 000343F0 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 0002E4B8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 0071DF (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 007199 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 007013 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00D220 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1546T
[illegible] . . . . . [illegible]
SAMPLING RATE . . . . . [illegible]
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1548
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.0171   BITS
CHANNEL Y ENTROPY . . . . 3.0114   BITS
CHANNEL Z ENTROPY . . . . 3.0373   BITS
TOTAL SOURCE ENTROPY. . . 3.0355   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.6354   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.4998   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 30.5  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 91.4  % SMP INTERVAL
COLLECTION DURATION . . . 17.4  SECONDS
CHANNEL X MAXIMUM . . . . 2.77343  VOLTS
CHANNEL X MINIMUM . . . . -1.3671  VOLTS
CHANNEL Y MAXIMUM . . . . 2.73437  VOLTS
CHANNEL Y MINIMUM . . . . -1.3671  VOLTS
CHANNEL Z MAXIMUM . . . . 2.73437  VOLTS
CHANNEL Z MINIMUM . . . . -1.3671  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON, LEAD AVL
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 21FE (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00007134 (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 47 (HEX) AT SAMPLE NUMBER 0A9E (HEX)
XMIN= DD (HEX) AT SAMPLE NUMBER 1447 (HEX)
YMAX= 46 (HEX) AT SAMPLE NUMBER 095A (HEX)
YMIN= DD (HEX) AT SAMPLE NUMBER 1447 (HEX)
ZMAX= 46 (HEX) AT SAMPLE NUMBER 095A (HEX)
ZMIN= DD (HEX) AT SAMPLE NUMBER 1447 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00036[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00032FD0 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006EE8 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006E90 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006F88 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00DB48 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1559B
[illegible] . . . . . BLANK
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1559
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.0165   BITS
CHANNEL Y ENTROPY . . . . 3.0038   BITS
CHANNEL Z ENTROPY . . . . 3.0261   BITS
TOTAL SOURCE ENTROPY. . . 3.0233   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.6460   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.5280   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 32.0  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 93.0  % SMP INTERVAL
COLLECTION DURATION . . . 17.7  SECONDS
CHANNEL X MAXIMUM . . . . 2.92968  VOLTS
CHANNEL X MINIMUM . . . . 0.19531  VOLTS
CHANNEL Y MAXIMUM . . . . 2.89062  VOLTS
CHANNEL Y MINIMUM . . . . 0.19531  VOLTS
CHANNEL Z MAXIMUM . . . . 2.89062  VOLTS
CHANNEL Z MINIMUM . . . . 0.15625  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON, LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 22A2 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 26 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00005B1E (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 4B (HEX) AT SAMPLE NUMBER 1FA2 (HEX)
XMIN= 05 (HEX) AT SAMPLE NUMBER 00BD (HEX)
YMAX= 4A (HEX) AT SAMPLE NUMBER 1FA2 (HEX)
YMIN= 05 (HEX) AT SAMPLE NUMBER 005C (HEX)
ZMAX= 4A (HEX) AT SAMPLE NUMBER 1FA2 (HEX)
ZMIN= 04 (HEX) AT SAMPLE NUMBER 01D5 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 0003A780 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00033F30 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006BE5 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006AC9 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006857 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00E9E0 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVAILABLE
```

```
:EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1511S
[illegible] . . . . . [illegible]
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 1511
TIME OF COLLECTION. . . . 1511
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.3255   BITS
CHANNEL Y ENTROPY . . . . 3.3341   BITS
CHANNEL Z ENTROPY . . . . 3.3560   BITS
TOTAL SOURCE ENTROPY. . . 3.3457   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.3911   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.4319   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 31.0  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 94.0  % SMP INTERVAL
COLLECTION DURATION . . . 16.6  SECONDS
CHANNEL X MAXIMUM . . . . 1.32812  VOLTS
CHANNEL X MINIMUM . . . . -2.0703  VOLTS
CHANNEL Y MAXIMUM . . . . 1.32812  VOLTS
CHANNEL Y MINIMUM . . . . -2.0703  VOLTS
CHANNEL Z MAXIMUM . . . . 1.32812  VOLTS
CHANNEL Z MINIMUM . . . . -2.0703  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON. LEAD AVL
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2074 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00004ADC (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 22 (HEX) AT SAMPLE NUMBER 1171 (HEX)
XMIN= CB (HEX) AT SAMPLE NUMBER 021D (HEX)
YMAX= 22 (HEX) AT SAMPLE NUMBER 1423 (HEX)
YMIN= CB (HEX) AT SAMPLE NUMBER 021D (HEX)
ZMAX= 22 (HEX) AT SAMPLE NUMBER 1423 (HEX)
ZMIN= CB (HEX) AT SAMPLE NUMBER 00E8 (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 000390[illegible] (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00030AE0 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 00717A (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 00713D (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 006[illegible]7 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 001000 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM BITS AVALIABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1520B
SUBJECT . . . . . BALSALO
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1520
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 2.6012   BITS
CHANNEL Y ENTROPY . . . . 2.5808   BITS
CHANNEL Z ENTROPY . . . . 2.6229   BITS
TOTAL SOURCE ENTROPY. . . 2.6104   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 3.0645   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.7247   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 35.1  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 90.9  % SMP INTERVAL
COLLECTION DURATION . . . 20.0  SECONDS
CHANNEL X MAXIMUM . . . . 2.07031  VOLTS
CHANNEL X MINIMUM . . . . -0.5468  VOLTS
CHANNEL Y MAXIMUM . . . . 2.07031  VOLTS
CHANNEL Y MINIMUM . . . . -0.5468  VOLTS
CHANNEL Z MAXIMUM . . . . 2.07031  VOLTS
CHANNEL Z MINIMUM . . . . -0.5468  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON. LEA D 2
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2717 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 00008A6D (HEX)
TIME EFFICIENCY = (1-(LOOPCT/(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 35 (HEX) AT SAMPLE NUMBER 1F9E (HEX)
XMIN= F2 (HEX) AT SAMPLE NUMBER 0A63 (HEX)
YMAX= 35 (HEX) AT SAMPLE NUMBER 1F9E (HEX)
YMIN= F2 (HEX) AT SAMPLE NUMBER 054B (HEX)
ZMAX= 35 (HEX) AT SAMPLE NUMBER 1F9E (HEX)
ZMIN= F2 (HEX) AT SAMPLE NUMBER 054B (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 000[illegible]0 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00[illegible] (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 0066BE (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006666 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = 0063CC (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00F630 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED [illegible] BITS AVAILABLE
```

```
EKG SAMPLE COLLECTION STATISTICS : PAGE 1
FILENAME. . . . . TA1439ST
SUBJECT . . . . . STROUP
SAMPLING RATE . . . . . 500 HZ
DATE OF COLLECTION. . . . 23 OCT 80
TIME OF COLLECTION. . . . 1439
COMPRESSION USED. . . . . TOLAN-A
CHANNEL X ENTROPY . . . . 3.2669   BITS
CHANNEL Y ENTROPY . . . . 3.2756   BITS
CHANNEL Z ENTROPY . . . . 3.3094   BITS
TOTAL SOURCE ENTROPY. . . 3.2912   BITS
PRESS RETURN FOR PAGE 2 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS : PAGE 2
APPROX MAX COMPRESSION
RATIO POSSIBLE. . . . . 2.4307   : 1
COMPRESSION RATIO
ACHIEVED. . . . . 1.4314   : 1
ACHIEVED COMPRESSION
EFFICIENCY. . . . . 30.1  % OF MAXIMUM
COMPRESSION TIME
EFFICIENCY OBTAINED . . . 93.4  % SMP INTERVAL
COLLECTION DURATION . . . 16.6  SECONDS
CHANNEL X MAXIMUM . . . . 3.35937  VOLTS
CHANNEL X MINIMUM . . . . -0.5468  VOLTS
CHANNEL Y MAXIMUM . . . . 3.35937  VOLTS
CHANNEL Y MINIMUM . . . . -0.5468  VOLTS
CHANNEL Z MAXIMUM . . . . 3.35937  VOLTS
CHANNEL Z MINIMUM . . . . -0.5859  VOLTS
COMMENTS. . . . . X,Y,Z IN COMMON. LEAD 1
PRESS RETURN FOR PAGE 3 OF STATISTICS=

EKG SAMPLE COLLECTION STATISTICS: PAGE 3
NUMBER OF SAMPLES TAKEN (SAMPNO)= 2071 (HEX)
MAXIMUM LOOP COUNT PER INTERRUPT (LPCAL)= 27 (HEX)
TOTAL WAITING LOOP COUNTS DURING COLLECTION (LOOPCT)= 0000526C (HEX)
TIME EFFICIENCY = (1-(LOOPCT%(SAMPNO*LPCAL)))*100
CHANNEL MAXIMUMS AND MINIMUMS
XMAX= 56 (HEX) AT SAMPLE NUMBER 1472 (HEX)
XMIN= F2 (HEX) AT SAMPLE NUMBER 1E5A (HEX)
YMAX= 56 (HEX) AT SAMPLE NUMBER 1472 (HEX)
YMIN= F2 (HEX) AT SAMPLE NUMBER 0FD8 (HEX)
ZMAX= 56 (HEX) AT SAMPLE NUMBER 1472 (HEX)
ZMIN= F1 (HEX) AT SAMPLE NUMBER 1E5A (HEX)
COMPRESSION STATISTICS:
NUMBER OF MEMORY BITS AVAILABLE = 021FF0 (HEX)
NUMBER OF BITS AVAILABLE TO VAR LEN CODER = 00037200 (HEX)
TOTAL NUMBER OF DATA BITS STORED = 00030A98 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL X = 006E88 (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Y = 006EDC (HEX)
NUMBER OF BITS USED TO STORE CHANNEL Z = C06CF0 (HEX)
NUMBER OF BITS USED TO STORE TIME OR OTHER PARAMETER = 00DDC0 (HEX)
COMPRESSION RATIO = TOTAL DATA BITS STORED PER MEM. BITS AVAILABLE
```

Appendix E

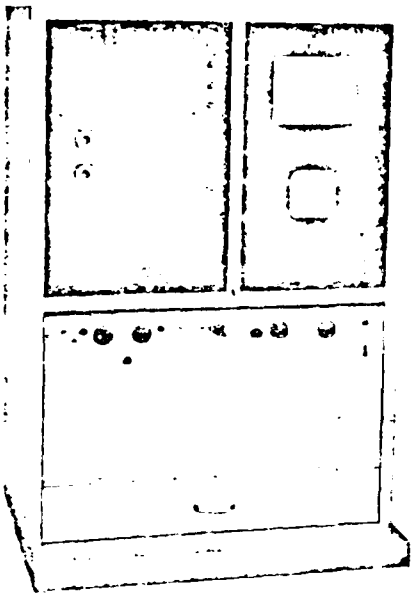
This appendix contains a photocopy of the specifications for the pertinent equipment used in this thesis.

OR-VC

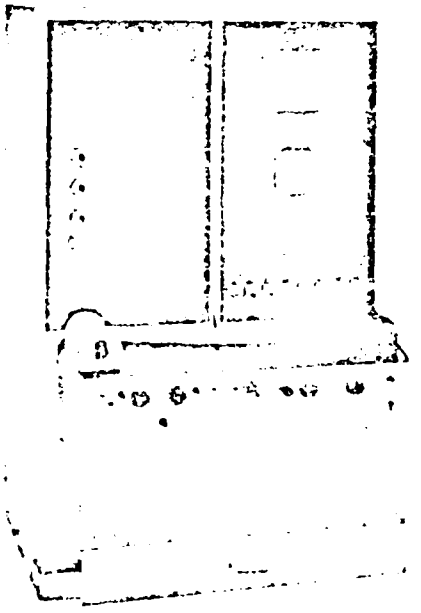
Copied from Dr. VanSlyck's Diary. 1/11

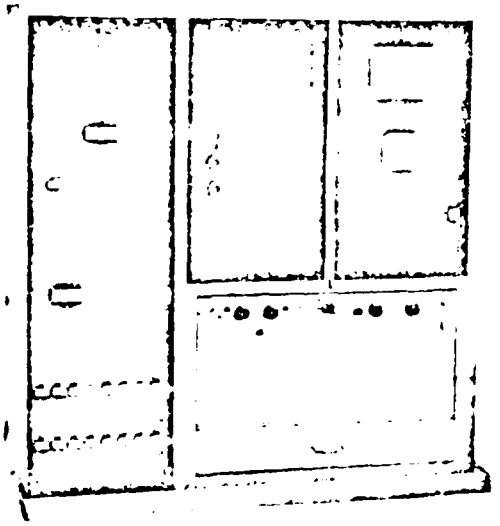
Bld 79
550/150

RESEARCH RECORDER



OPERATING INSTRUCTIONS





ELECTRONICS FOR MEDICINE INC

30 VIRGINIA ROAD, WHITE PLAINS, N.Y. 10603 TEL. 914 WH 6-1700

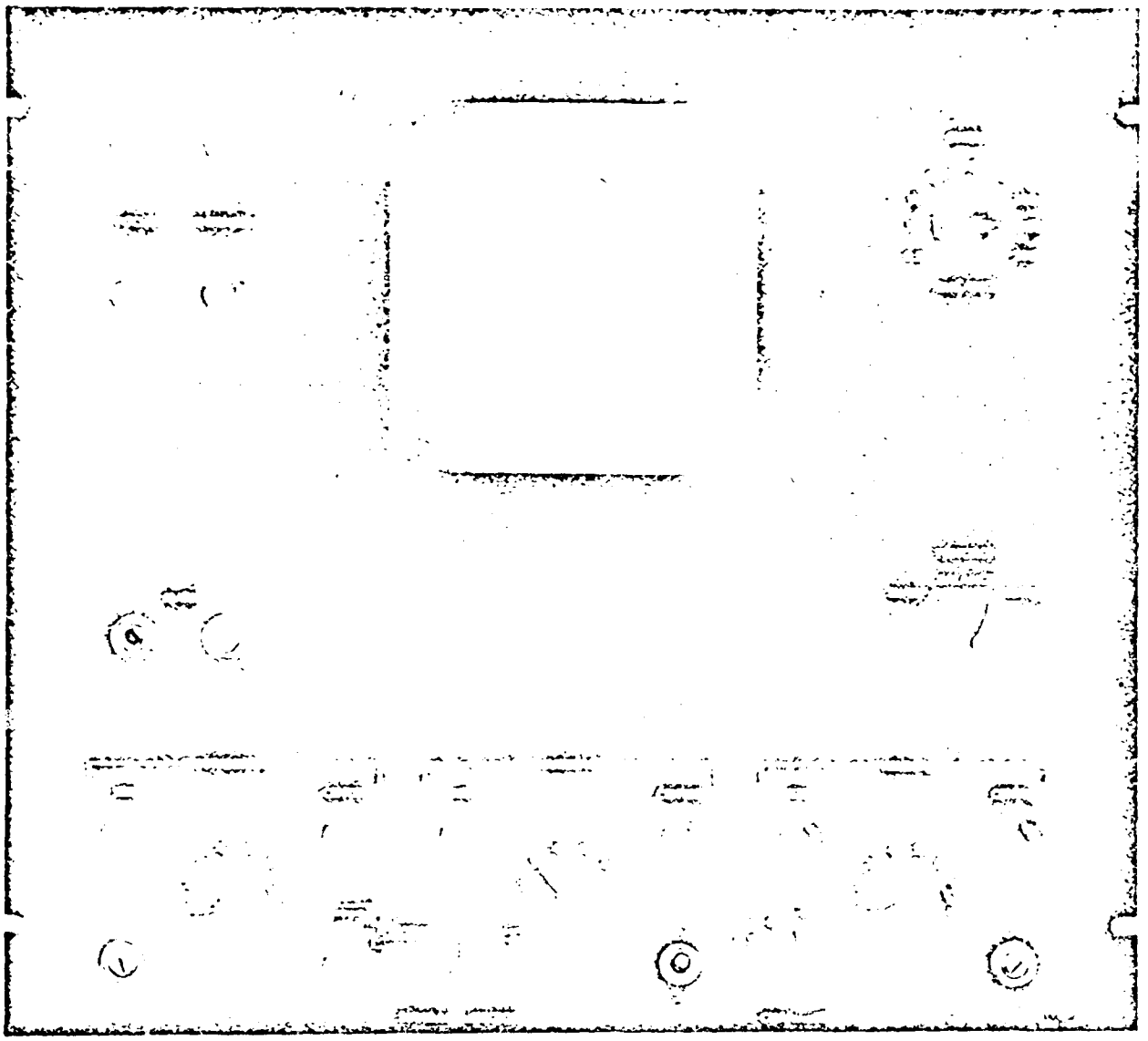
DR-8 RESEARCH RECORDER MONITOR

The Model DR-8 Research Recorder is designed to operate from a 100-130 volt, 60 Hz line. Power consumption is about 700 watts. The circuit is protected by a 10 ampere circuit breaker which is incorporated in the power ON-OFF switch on the camera panel.

It is essential for safety and to minimize 60 Hz interference in the tracings that the Research Recorder be connected to a good "ground". A third wire is brought out at the power plug for this purpose. A good "ground" can usually be obtained at a cold water pipe.

The instrument combines as many as eight amplifiers with a cathode ray tube recording camera and two cathode ray tube monitors. The multitrace monitor displays any number of traces up to eight. The single monitor displays any one of the traces enlarged or reduced in size. It is also used for the presentation of loops, such as the vectorcardiogram or the lung compliance and lung resistance loops, and for balancing strain gauge amplifiers. All phenomena can be monitored continuously. Recording is done at will, merely by turning a switch. The display on the cathode ray tube in the camera can be identical to that on either the multitrace tube or the single trace tube. It is therefore possible to photograph either scalar tracings or loops. In actual operation, size adjustments and preliminary measurements are made by observing the monitor screens without recording. Thus, the camera runs for only short intervals.

The multitrace tube has only one beam, but because the amplifiers are connected to it through an electronic switch operating at a rate of 45,000 Hz (sec., it is possible to display as many as eight traces simultaneously. The beam switches rapidly from one trace to the next, with blanking of the cathode ray tube taking place during the switching period. Thus, each beam is on screen [illegible] of the time, although giving the impression of being continuous. A switch is provided to make possible more frequent sampling for high speed phenomena, by reducing the number of channels displayed.



POWER switch, located on the right side of camera panel, turns the Research Recorder on. A minute or two is required before the traces on the monitor screens are stabilized. Adjustments may now be made.

INTENSITY and FOCUS control adjust the cathode ray tube beams. The upper two controls are for the multitrace tube, the lower two for the single trace tube. The intensity is set according to the brightness of the room in which the instrument is being used. The traces will be sharper at lower intensities. Focus is adjusted for the sharpest spot and should be done when the beams are approximately 1/3rd of the distance from either side of the tube to allow best overall focusing.

FUNCTION selector switch determines the presentation on the single trace tube; the multitrace tube will display scalar traces continuously. The single trace tube will display the trace from any amplifier when the FUNCTION selector switch is in the SWEEP position. In the BALANCE position, a pattern is displayed on the single trace screen which permits rapid balancing of the pressure transducers. In LOOPS, any channel can be presented on the horizontal axis of the single trace tube, and at the same time any other channel can be presented on the vertical axis. This position is used for the VCG loops (X-Y plot).

SWEEP switch determines the speed at which the traces move across the screen. The proper setting depends on the repetition rate of the phenomena displayed. Nine monitoring speeds are available: 2.5, 5, 10, 25, 50, 75, 100, 150 and 200 mm/sec.

The MULTITRACE MONITOR permits the display of up to eight channels at one time. The positions of the individual traces on the multitrace tube are controlled by the CENTER knob on the individual channels. Each of the traces can be positioned anywhere on the screen, superimposed on other traces, or can be moved off screen in either direction if it is not being used. The amplitude of the traces on the multitrace screen will depend upon the setting of the AMPLITUDE controls on the individual amplifiers.

The CHANNELS switch determines the number of amplifiers that will be displayed. The multitrace presentation is the result of an 8 channel electron switch, whose basic frequency is 48 K Hz. In the 1-8 position, each amplifier will be sampled equally 6,000 times/second, or every 166 microseconds. In the 1-5 position, channels 1, 2 and 3 are sampled at 12,000 times/second or every 82 microseconds. Channels 4 and 5 are sampled at the normal 6,000 times and channels 6, 7 and 8 are not displayed at all. Placing the multitrace switch at 1-6 will sample channels 1 and 2 at the 12,000 rate and 3, 4, 5 and 6 at the normal 6,000 rate. Channels 7 and 8 will not appear on the multitrace screen. Amplifiers providing fast rise time phenomena, such as heart sounds, action and nerve potentials, should be in positions 1, 2 and 3 to take advantage of the faster sampling rate and to minimize the dotting effect.

When the SUBTRACT SELECTOR switch is in the OFF position, the built-in baseline, gradient and subtract functions are inoperative. Turning switch to BASELINE will allow built in baseline to be positioned to any point on screen with POSITION control. An event MARK button and jack (for marking from a remote position) are provided. To display either baseline or gradient 1-7, 1-6, 1-8 switch must be in 1-8, since baseline or gradient trace replace channel 8. When BASELINE or GRADIENT position is selected channel 8 amplifier will not be seen.

A PRESSURE GRADIENT can be displayed on the multitrace screen together with the two pressures. The adjustment procedure is as follows:

1) Attach pressure transducers to pressure amplifiers; allow them to warm up, balance the amplifiers. Position the traces to baseline from which measurements are to be made; equalize the amplitudes. (See SGA-SGM instructions)

2) Turn the SUBTRACT SELECTOR switch to "GRADIENT". The gradient trace will now be substituted for Channel 8 and its position controlled by the baseline POSITION control.

3) Turn HORIZONTAL SIZE and SUBTRACT SIZE to zero (counterclockwise)

4) Select the channel to be subtracted from with the Horizontal switch; the channel to be subtracted with the Subtract switch.

5) Position the gradient trace to the baseline position of the two pressure channels. There are now 3 or 4 lines superimposed i.e., 2 pressure amplifiers, the difference trace (gradient) and optionally a baseline, which can be a baseline, marker trace, or any unused pressure amplifier.

6) Turn the +/- switch to +.

7) Turn HORIZONTAL SIZE slowly clockwise; note in which direction trace moves. If it moves upwards, turn HORIZONTAL CENTER counter-clockwise to return trace to its original position. Advance HORIZONTAL SIZE fully clockwise and readjust HORIZONTAL CENTER, if necessary. When the center is optimally adjusted, turning HORIZONTAL SIZE should not affect the position of trace. Return HORIZONTAL SIZE to zero. Repeat procedure with SUBTRACT SIZE and CENTER controls, returning SIZE to zero after obtaining balance.

8) Turn CALIBRATE control on pressure amplifier which was selected by HORIZONTAL SWITCH so that its trace moves approximately full screen; adjust HORIZONTAL SIZE until gradient trace coincides with the pressure trace. Remove the CALIBRATE signal from the pressure amplifier and both lines will return to their original positions.

9) Turn the CALIBRATE on that amplifier selected by SUBTRACT switch until the pressure trace moves approximately full screen. Adjust SUBTRACT SIZE until lines are superimposed. Return CALIBRATE switch to zero and both lines will return to their original positions.

10) The "Gradient" trace will now represent the sum of the two pressures. To observe the gradient, turn the + - switch to -. Readjust gradient POSITION, if necessary.

For the H and V sections of SUBTRACT SELECTOR switch, see loops.

SINGLE TRACE MONITOR presentation is determined by the FUNCTION switch. Any individual amplifier can be displayed when the FUNCTION switch is in the SWEEP position. Turn the VERTICAL selector switch to the channel number representing the desired amplifier. Make certain this amplifier is visible on multitrace screen. The position of the selected trace can be controlled by the CENTER control on the channel, or by the VERTICAL CENTER control. The amplitude on the screen is adjusted by the VERTICAL SIZE control and can be 2 1/2 times the amplitude of the corresponding trace on the multitrace tube, or it

can be reduced as low as zero amplitude. When the SIZE control is at zero (completely counterclockwise), the CENTER control will have no effect. If trace is more than 1" from mid-screen, (SIZE control fully counterclockwise) turn screwdriver adjust below vertical CENTER to return baseline to mid-screen. Turning the SIZE control clockwise may move trace off screen. Note in which direction it moves (up or down) and return trace to screen with VERTICAL CENTER control. If trace moves upward, turn CENTER counterclockwise; if trace moves downward, turn CENTER clockwise.

The BALANCE position of the FUNCTION switch will display a lissajous figure, which is used as an aid in balancing a transducer in a pressure amplifier.

Turn VERTICAL selector switch to pressure amplifier (with transducer connected) to be balanced. The horizontal deflection represents the transducer excitation voltage, the vertical represents the output voltage from the transducer through a pressure amplifier.

The vertical SIZE and CENTER controls do not have any effect in the BALANCE position. When the transducer is brought into balance, there will not be any output voltage, and the lissajous presentation will be a straight horizontal line.

In LOOPS position, the VERTICAL selector switch selects the channel to be presented on the vertical axis, and the vertical SIZE and CENTER controls are used to adjust vertical amplitude and position.

The HORIZONTAL switch selects the channel to be presented on the horizontal axis, and the horizontal SIZE and CENTER controls are used to adjust horizontal amplitude and position. When the HORIZONTAL (or VERTICAL) SIZE control is turned to minimum counterclockwise, the CENTER controls will not have any effect. A dot should appear at mid-screen. If it is more than 1" from mid-screen, adjust screwdriver controls below VERTICAL CENTER or HORIZONTAL SIZE until dot is at mid-screen.

Advance HORIZONTAL SIZE clockwise and adjust HORIZONTAL CENTER to return dot to its original position near mid-screen. Proper adjustment of center is achieved when it is possible to turn horizontal SIZE from minimum counterclockwise to maximum clockwise without moving dot when there isn't any applied signal. Do the same with the VERTICAL and SUBTRACT switches: return the subtract SIZE to minimum counterclockwise.

The H or V positions of SUBTRACT SELECTOR make it possible to add or subtract the signals from two channels on either the horizontal or vertical axis when in the LOOPS loops. This is especially useful when measuring Lung Compliance or Lung Resistance. For example: Let us suppose it is desired to subtract from the horizontal axis. The HORIZONTAL switch selects the amplifier to be subtracted from. Then, the SUBTRACT selector switch is used to select the channel to be subtracted. The - switch is turned to -. The SUBTRACT SELECTOR is turned from OFF to H (horizontal). The SUBTRACT. SIZE and CENTER controls will now also control horizontal amplitude and position but in a direction opposite to that of the HORIZONTAL controls. See instructions for Lung Resistance and Lung Compliance tests.

Timing and direction marking can be introduced to the loop patterns by turning the camera selector to LOOPS and choosing the appropriate TIMING rate. The camera ON-OFF switch need not be ON to display timing on loop patterns, only when photographing loop. Camera timing should be in .004 or .02 to properly display LC/LR loops; or .004 for VCG.

PMB
9/69

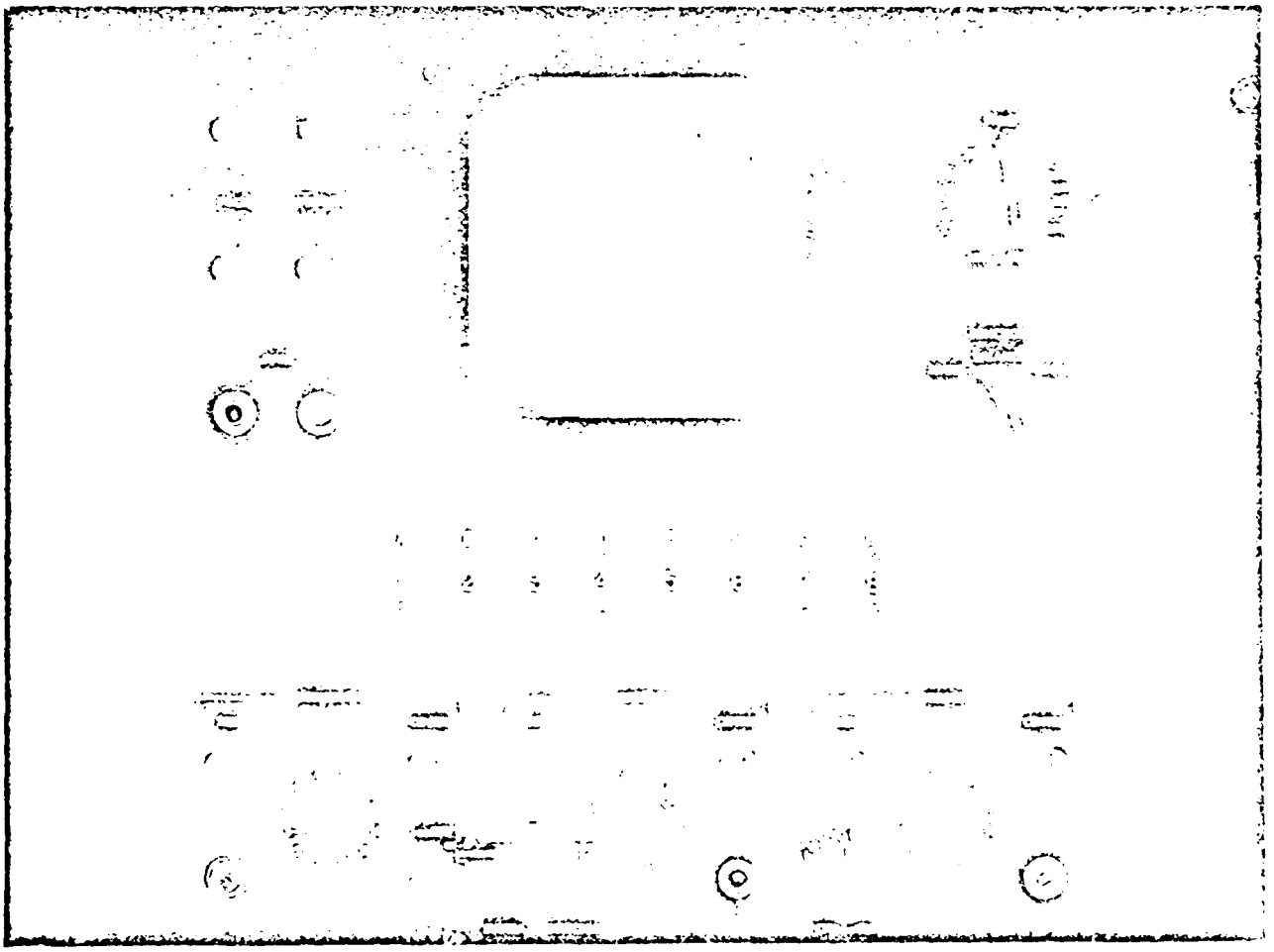
ELECTRONICS for MEDICINE, Inc., 30 Virginia Rd., White Plains, N.Y.

MONITOR

MODEL DR-12

The DR-12 recorder is a modified version of the DR-8. The DR-8 instruction plus the following information covers the operation of the DR-12.

The amplifiers in the DR-12 are numbered from 1 through 12. Ten of these, 1-10, starting from the top, are located on the left side of the recorder; numbers 11 and 12 are located beneath the monitor panel. The eight displayable channels of the recorder are lettered from A through H with a dial type switch supplied for each channel which permits any of the 12 amplifiers to be selected on any of the 8 channels. Since it is possible to select the same amplifier on more than one channel, the user should be certain that each of the dials is set to a different number or the same trace will appear on the screen more than once.



The A-E, A-F, A-H switch on the monitor panel can be used to sample two or three channels more frequently than the others. The electronic switching rate is 48,000 Hz/second; each of the 8 channels is sampled 6,000 times per second when the switch is in the A-H position. In the A-E position, channels A, B, and C are sampled 12,000 times/second while channels D and E are sampled 6,000 times/second. Only

5 channels are operative in this position. In the A-F position, channels A and B are sampled 12,000 times/second while channels C, D, E and F are sampled [illegible] times/second. Operation is limited to [illegible] in this position. [illegible] be sampled at these higher rates by selecting them on channels A or B.

SUBTRACT SELECTOR switch marked BASELINE, GRADIENT, OFF, H and V, when positioned on BASELINE or GRADIENT, will remove channel H (A-E, A-F, A-H switch must be on A-H) from display on multitrace screen and replace it with a baseline or gradient trace. With switch in BASELINE, the POSITION control, below the switch, can be used to set the baseline to any desired point. A MARK push button and remote MARK jack can be used as an event marker. A hand foot switch can be patched into MARK jack, to mark events from a remote point. Placing switch on GRADIENT position will permit the display of gradients using built in gradient amplifier (POSITION control is the same as for baseline). The subtraction is done horizontally. See section on "subtraction" in DR-8 instructions. The H and V position of the SUBTRACT SELECTOR switch, permit subtraction on vector screen, either vertically or horizontally in the OFF position, subtract or GRADIENTS and BASELINE are inoperative. See DR-8 instructions.

SINETRAC SERIES

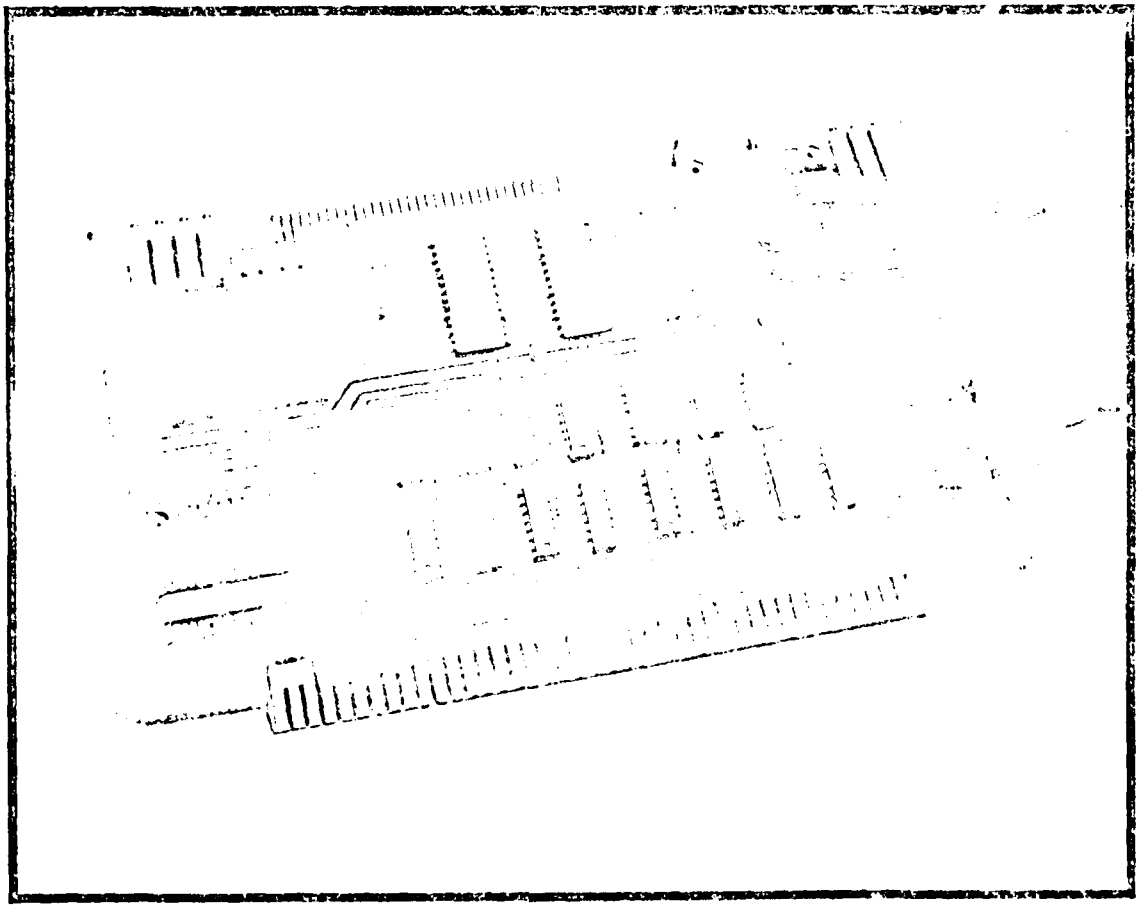
MODEL ST-6800

A/D-D/A PERIPHERAL SYSTEMS

INSTRUCTION MANUAL

Part No. 58-12140-25

JANUARY 1979



SYSTEMS, INC.
[illegible] CABOT BOULEVARD, MANSFIELD, MASS. 02048 TEL. (617)339-9341 and (617) [illegible]
TWX (710)346-[illegible] TELEX [illegible]

SPECIFICATIONS

DATA ACQUISITION SECTION

Specifications

Typical @ +25°C, dynamic conditions, unless otherwise specified.

Analog Inputs

Number of Channels	32 Single-ended or 16 differential
Channel Expansion	Up to 128 single ended or 112 differential using ADC-Slave Expander Cards (Model ST-6[illegible] ADX)
Full Scale Input Ranges	0 to +5 Volts 0 to +10 Volts -5 to +5 Volts -10 to +10 Volts
Current Input Channel Range (8 Channels)	4-20 mA type
Common Mode Range	±10 Volts
Input Overvoltage	±35 Volts Max. continuous
Input Impedance	100 Megohms differential or to ground
Input Bias Current	3nA typ., 10nA max.
Input Capacitance	5pF, OFF CHANNEL, 100 pF ON CHANNEL to ground

Performance

Accuracy @ +25C	Within ±0.025% of input range
Resolution	12 Binary bits (1 part in 4096)
Nonlinearity	±1/2 LSB maximum
Differential non-linearity	±1/2 LSB maximum
Gain Error	Adjustable to zero
Offset or Zero Error	Adjustable to zero
Gain Temperature Drift (Bipolar)	Within ±10 ppm of FSR/°C
Zero Temperature Drift (Unipolar)	Within ±5 ppm of FSR/°C max.
Common Mode Rejection	70 dB min, DC to 1 kHz with 1kΩ unbalance
Power Supply Rejection	100 dB to +5V bus

Dynamic Characteristics

Typical Data Transfer	
I/O Period (Total)	36 microseconds
Throughput Period	20 microseconds
Acquisition Time	8 microseconds
A/D Conversion Time	12 microseconds
Aperture Time	100 nanoseconds
Sample/Hold Switch Feedthrough	.01% Max.
MUX Crosstalk from OFF Channels	.007% @ 1 kHz, Rs = 1K

Digital Outputs

Output Coding	Straight Binary (Unipolar) Offset Binary (Bipolar) 2's Complement (Bipolar) } Jumper Selected
Output Format	2-Byte group electrically compatible to Motorola's EXORciser bidirectional bus. Sign extension is jumper selected on bits 12 thru 15 for 2's complement units. Bits 12 thru 15 are logic zero for all other units.
Channel Addressing	Random channel addressing may be started by external interrupt input for event operation or by internal program control.
Base Address	Prewired by PC Board jumpers for one of 128 base addresses.

Data Distribution Section (D/A Analog Outputs)

Number of Channels	2 (Expandable only by stand-alone ST-6800VA Boards.)
Resolution	12 Bits
Full Scale Output	0 to +5 Volts
Voltage Ranges	0 to +10 Volts -5 to +5 Volts -10 to +10 Volts
Input Coding	Straight Binary (Unipolar) Offset Binary (Bipolar) 2's Complement (Bipolar)
Output Impedance	.05 ohm
Output Current	±5 mA min

Performance

Non linearity	± 1/2 LSB, maximum
Differential Nonlinearity	± 1/2 LSB, maximum
Gain Error	Adjustable to zero using pot. for each channel
Offset or Zero Error	Adjustable to zero using pot for each channel
Gain Temperature Drift	±20 ppm of output/°C
Zero Temperature Drift	(Unipolar output) ±5 ppm of FSR/°C
Offset Temperature Drift	(Bipolar output) ±10 ppm of FSR/°C
Settling Time (20V change)	4 microseconds to ± 1/2 LSB
Slew Rate	20V/usec
Power Supply Rejection	±0.02% of FSR per 1% variation

Power Consumption

1.2 ampstypical @ +5 vdc supplied from MPU bus connector. On-board DC to DC Convert supplies ±15 vdc to linear circuits.

Physical

Operating Temperature Range	0° to +70°C
Storage Temperature Range	-25°C to +85°C
Card Size	9.75"W x 5.75"H x .062"D

BRUSH

OPERATING INSTRUCTIONS

Brush Recorder Mark II
Model RD 2522 20

Specifications

Sensitivity	10 millivolts per chart line (mm). Full scale deflection from chart center = 200 millivolts.
Sensitivity steps	.01, .02, .05, .1, .2, .5, 1, 2, 5 and 10 volts per chart line (mm). Maximum attenuator error 1% with balanced input.
Measurement range	
Single-ended input (5 megohm)	.010 volt to 400 volts
Balanced input (10 megohm)	.010 volt to 400 volts, side to side, allowable voltage off-ground at any attenuator step is 1600 x volts per chart line switch setting up to 500 volts off-ground.
Common mode rejection	Better than 1000 to 1, attenuator set in .01 volt per chart line position.
Zero line stability	Less than 1/4 chart line (mm) per hour. Total drift over eight hour period not more than 1/2 chart line (mm).
Noise	Not noticeable on chart with shorted input.
Frequency response	The recorded peak-to-peak amplitude of a constant voltage sine wave will be within 5% chart line (mm) of a nominal 10 lines from D.C. to 100 cps.
Maximum amplitude	40 lines (mm) peak-to-peak, D.C. to 40 cps. 20 lines (mm) peak-to-peak, D.C. to 70 cps. 10 lines (mm) peak-to-peak, D.C. to 100 cps.

Specifications

Pen Bias	Permits positioning of pen on chart, ± 20 chart lines (mm). Effective for either single-ended or balanced input.
Trace linearity	D. C. within 2% full chart width, A. C. within 3% full chart width, any frequency within limits of maximum amplitude for electric writing.
Trace width	0.006 with Model RA 2822-31 pen.
Writing method	Electric stylus.
Number of recording channels	2
Number of event channels	1 actuated by external switch. 1 actuated by panel switch.
Channel width	40 mm (40 divisions).
Chart Supply	150 feet
Chart speeds	1, 5, 25, and 125 mm per second
Chart speed regulation	Synchronous motor, direct drive.

Specifications

Operating Temperature range, ambient	0°C to 55°C.
Power requirements	105-125 volts, 60 cps, 135 watts at 115 volts.
Transistors	1-5651 and 2-5687
Tubes	1-6BW4, 2-12B4A, 2-12AT7, 2-12AX7
Input terminals	
Front (signal)	Binding posts.
Rear (event Marker)	Binding posts.

Supplies

DESCRIPTION	BRUSH PART NO.
Chart Paper, 2 channel	RA 2922-22
Electric Styli (4)	RA 2822-31
Gram Gage Assembly	227416-910
Pen Mounting Tool	126717

CHAPTER 1 GENERAL DESCRIPTION

1-1 INTRODUCTION

The M6800 EXORciser (M68SDT) is a system development tool used in the design and development of M6800 Microcomputer Systems. The EXORciser Debug and the user's system are built around the M6800 Microcomputer Family of Parts. The M6800 Microcomputer Family of Parts are discussed in the following documents.

- M6800 Microprocessor Programming Manual
- M6800 Microprocessor Applications Manual
- M6800 Microprocessor Family of Parts Data Sheets

The EXORciser may be configured in a variety of applications and with various EXORciser options. This manual, rather than discussing every possible configuration and option, discusses only the basic EXORciser with each option except the Wirewrap and Extender Modules discussed in a supplement to this manual. The basic EXORciser is discussed in Paragraph 1-5 and the options are identified in Paragraph 1-6.

This manual provides general information, installation instructions, applications information, operating procedures, and theory of operation, for Motorola's M6800 EXORciser. The M6800 EXORciser is illustrated in Figure 1-1.

1-2 EXORciser FEATURES

The features of the basic EXORciser include.

- Flexible, adaptable, and expandable design development tool
- Easy to use
- Provides the Microprocessing Unit capability for both the EXORciser and the user's system
- Saves system design and development time
- Decreases system design and development costs
- Evaluates and debugs the user's program
- Evaluates and debugs the user's system hardware

1-3 EXORciser SPECIFICATIONS

Table 1-1 identifies the basic EXORciser specifications.

TABLE 1-1. Basic EXORciser Specifications

CHARACTERISTICS	SPECIFICATIONS
Power Requirements	95-135 205-250 V AC 47-420 Hz, 250W
Word Size Data Address Instructions	 8 bits 16 bits 8, 16, or 24 bits
Memory Size	64K bytes
Instruction Set	72 variable length instructions
Clock cycle time	Selectable, 1μs or external clock between 1 and 2 MHz
Interrupt	Maskable and non-maskable
Physical Characteristics Table top Length Depth Height	 19.25 in 17.50 in 7.00 in
Rack Mountable Length Depth Height	 19.00 in 17.00 in 7.00 in
Baud Rates (Switch Selectable)	110, 150, 300, 600, 1200, 2400, 4800, or 9600

1-4 FUNCTIONAL DESCRIPTION

The M6800 EXORciser, illustrated in Figure 1-2, [illegible] of Paris. The EXORciser may be easily tailored to meet the user's needs in the design and development of his system. Its modular design reduces the time required to develop a system and at the same time, provides great flexibility in configuring an emulation (functional representation) of the user's system. The EXORciser's EXbug firmware, through its debug and program control features, minimizes the time required to develop user's systems. The EXbug firmware provides the EXORciser with the capability to:

- Display the contents of the MPU registers
- Step through the program
- Trace through user's programs to locate problem areas
- Stop the program on a selected program step
- Provide an oscilloscope trigger signal on a selected program step
- Abort from the user's program and return to the EXbug control program on command
- Reinitialize the EXORciser on command
- Change the contents of memory

The user communicates with the EXORciser in one of two ways:

- Through a RS-232C or TTY terminal
- Through the EXORciser front panel controls and indicators

The terminal device permits the user to communicate directly with the EXORciser's EXbug Firmware. The EXORciser's front panel permits the user to apply power to the EXORciser, to abort (exit) the EXORciser from a routine, and to initialize and restart the EXORciser. The EXORciser's unique front panel was designed to incorporate future EXORciser options such as data keys and displays.

1-5 BASIC EXORciser SYSTEM (FIGURES 1-2 AND 1-3)

The basic EXORciser (M68SDT) consists of the MPU Module, the Debug Module, the Baud Rate Module, the Power Supply, and the chassis. These modules are built around the M6800 Microcomputer Family of Parts (MC6800 Microprocessing Unit, MC6820 Peripheral Interface Adapter, MC6850 Asynchronous Communications Interface Adapter, MCM6810 Random Access Memory, and MCM6830 Read Only Memory devices).

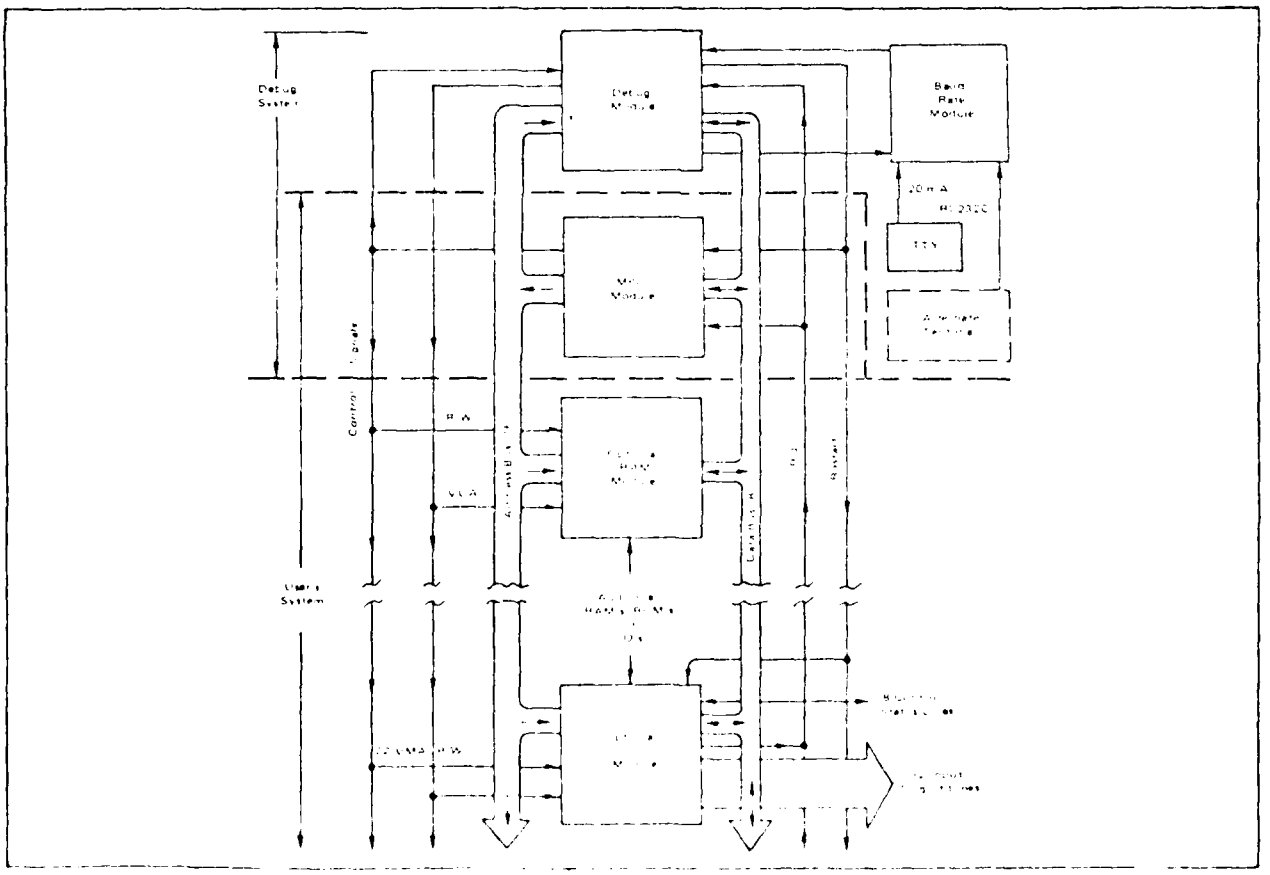


FIGURE 1-2 EXORciser Simplified Block Diagram

The MPU Module (Figure 1-4) incorporates the MC6800 Microcomputer Unit (MPU) and the system clock. This module, as illustrated in Figure 1-2, serves a dual function in the EXORciser system. The MPU [illegible] system. The MPU Module also initiates an EXORciser restart operation and initializes the EXORciser. The MC6800 Microprocessing Unit is an 8-bit parallel device capable of addressing 64k bytes of memory. In addition, the MPU addresses its input and output [illegible] ory. The MPU also provides the EXORciser with 72 variable length instructions and the [illegible] EXORciser [illegible] (delayed); the MPU Module appears exactly like a MC6800 Microprocessing Unit with enhanced TTL bus drive capability.

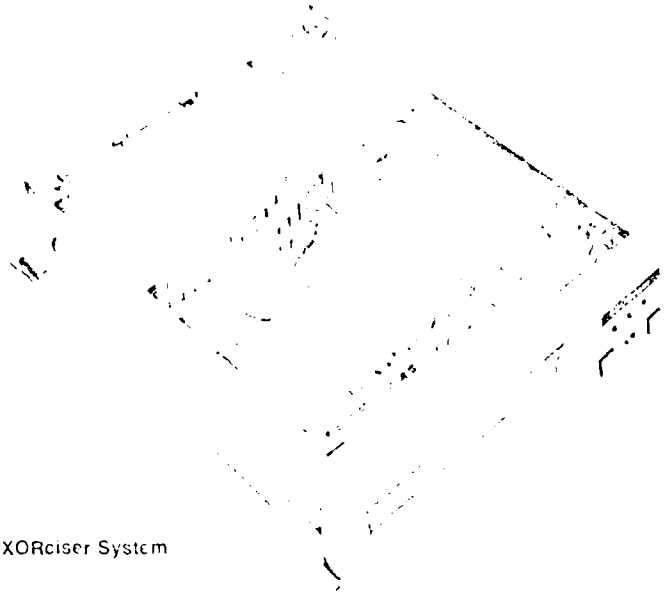


FIGURE 1-3. Basic EXORciser System

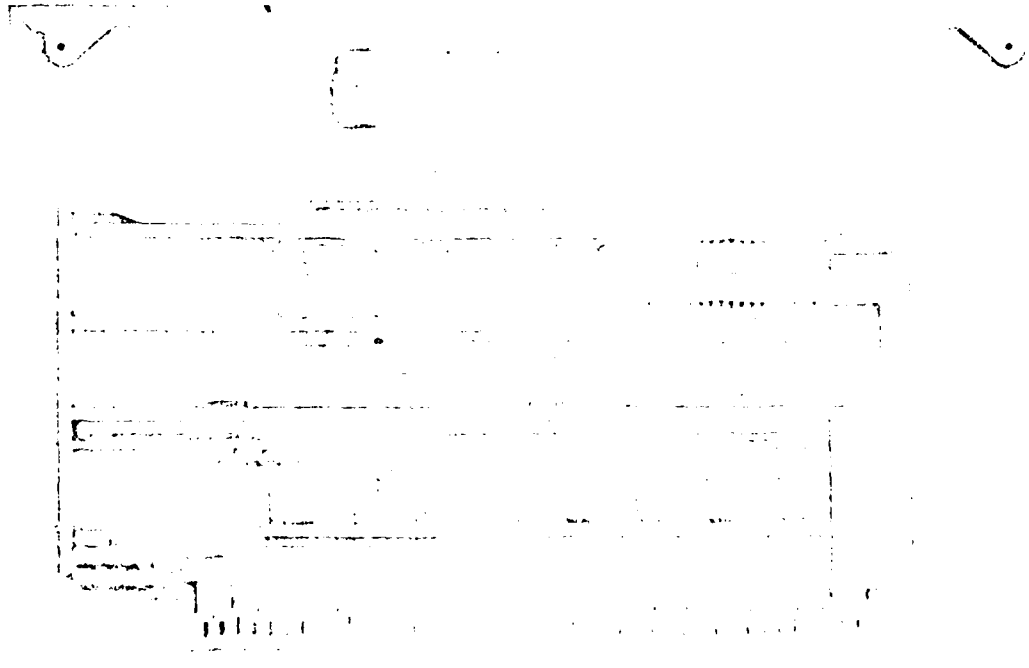


FIGURE 1-4. MPU Module

Vita

Melvin D. Townsend was born 5 April 1[illegible] in [illegible] Springs, Colorado. Upon graduation from Gen. William Mitchell High School in 1970, he enrolled in undergraduate engineering study at Colorado State University. In December, 1974, he was awarded the BSEE degree and commissioned as a second lieutenant, USAF, via AFROTC. Entering active duty immediately, he was assigned to the Foreign Technology Division (AFSC), WPAFB,Ohio. After three and one half years working technical intelligence engineering, he was transfered PCA to the AF Avionics Laboratory (AFSC) where he performed R&D engineering on electro-optical target acquisition systems. In June, 1979, he entered the Air Force Institute of Technology. Captain Townsend is a member of Eta Kappa Nu and IEEE.

Permanent Address
621 Prairie Rd.
Colorado Springs, CO
80909

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: AFIT/EN/10D-46	2 GOVT ACCESSION NO.	3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle): Analysis and Performance Evaluation of Electrocardiogram Data Compression Techniques	5 TYPE OF REPORT & PERIOD COVERED: MS Thesis; 6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s): MELVIN D. TOWNSEND, Captain USAF	8 CONTRACT OR GRANT NUMBER(s)
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS: Clincal Sciences Divison USAF School of Aerospace Medicine/NSP (UAAFSAM/NSF), Brooks AFB, Texas 78235	12 REPORT DATE: December 1980; 13 NUMBER OF PAGES: 252
14 MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15 SECURITY CLASS. (of this report): Unclassified; 15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

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

18 SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

16 JUN 1981

FREDIC C. LYNCH, Major, USAF
Director of Public Affairs

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

Electrocardiogram — Data Reduction
EKG Data Compression — Source Encoding

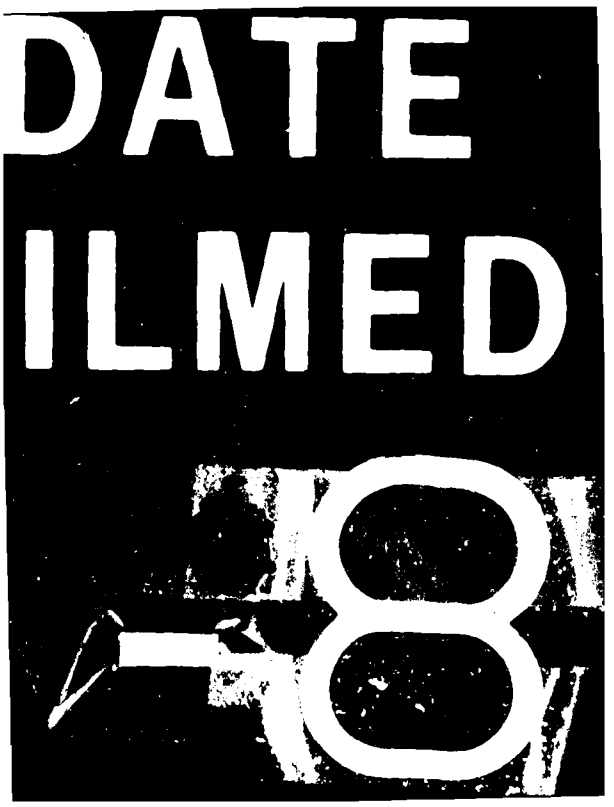
20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

EKG data compression techniques were investigated for potential real time implementation on an 8 bit Motorola 6800 microprocessor. Research indicated entropy reduction transform technique such as the Fast Fourier Transform and the discrete Karhunen-Loeve Transform were not feasible for implementation on the 6800. Two redundancy reduction (ER) techniques (TOLAY and DOWNE) utilizing 2nd order difference

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

operations in conjunction with variable length encoding were studied in detail. One such technique (DPCM) [illegible] revealed compression ratios ranging from 1.25:1 to 2.26:1. Investigation of the poor performance of the compression algorithm showed significant degradation of the 2nd order difference "decorrelator" due to a noisy collection environment. It was concluded that real time EKG data compression is feasible on the 6800 but that time compression techniques which store a zero value sequence counter versus the value of zero are not efficient in a high noise environment.


DATE
ILMED
8