

DTIC FILE COPY

AD _____

①

AD-A227 360

CONTRACT NO: DAMD17-89-C-9045

TITLE: DEVELOPMENT OF AN EEG ARTIFACT CORRECTION DEVICE

PRINCIPAL INVESTIGATOR: George M. Samaras

CONTRACTING ORGANIZATION: GMS Engineering Corporation
8940-D Route 108
Columbia, Maryland 21045

REPORT DATE: April 16, 1990

TYPE OF REPORT: Final Report

DTIC
ELECTE
OCT 02 1990
S B D

PREPARED FOR: U.S. ARMY MEDICAL RESEARCH AND DEVELOPMENT COMMAND
Fort Detrick, Frederick, Maryland 21702-5012

DISTRIBUTION STATEMENT: Approved for public release;
distribution unlimited

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION GMS Engineering Corporation		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State, and ZIP Code) 8940-D Route 108 Columbia, Maryland 21045			7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION U.S. Army Medical Research & Development Command		8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAMD17-89-C-9045		
8c. ADDRESS (City, State, and ZIP Code) Fort Detrick Frederick, Maryland 21702-5012			10. SOURCE OF FUNDING NUMBERS		
PROGRAM ELEMENT NO. 62787A		PROJECT NO. 3M1 62787A879 ✓	TASK NO. BH	WORK UNIT ACCESSION NO. 101	
11. TITLE (Include Security Classification) Development of an EEG Artifact Correction Device					
12. PERSONAL AUTHOR(S) George M. Samaras					
13a. TYPE OF REPORT Final Report		13b. TIME COVERED FROM 2/13/89 TO 3/16/90		14. DATE OF REPORT (Year, Month, Day) 1990 April 16	
15. PAGE COUNT					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP			
06	05		RA III; EEG, EOG, Artifact Correction, Portable Device;		
23	01		Mathematical Model; Volunteers;		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
The purpose of this research effort was to develop an EEG Artifact Correction Device. This consisted of refining an existing mathematical model and implementing this algorithm on a microprocessor based, battery operated, multichannel unit that would fit in a flight suit pocket. From a scientific point of view, this project was a great success in that the mathematical technique was extended to handle blink artifacts in a non-arbitrary biophysically based manner. From an engineering point of view, the project was not a great success in that technological limitations (computing speed of CMOS processors) prevented the microprocessor from correcting more than one EEG channel in nearly real-time.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Mrs. Virginia M. Miller			22b. TELEPHONE (Include Area Code) (301) 663-7325		22c. OFFICE SYMBOL SGRD-RMT-S

FOREWORD

Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the U.S. Army.

 Where copyrighted material is quoted, permission has been obtained to use such material.

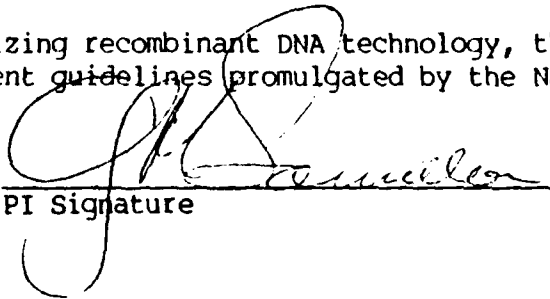
 Where material from documents designated for limited distribution is quoted, permission has been obtained to use the material.

✓ Citations of commercial organizations and trade names in this report do not constitute an official Department of the Army endorsement or approval of the products or services of these organizations.

 In conducting research using animals, the investigator(s) adhered to the "Guide for the Care and Use of Laboratory Animals," prepared by the Committee on Care and Use of Laboratory Animals of the Institute of Laboratory Animal Resources, National Research Council (NIH Publication No. 86-23, Revised 1985).

✓ For the protection of human subjects, the investigator(s) have adhered to policies of applicable Federal Law 45CFR46.

 In conducting research utilizing recombinant DNA technology, the investigator(s) adhered to current guidelines promulgated by the National Institutes of Health.


PI Signature

1/16/90
Date

TABLE OF CONTENTS

INTRODUCTION	2
<i>Nature of the Problem.</i>	2
<i>Background.</i>	2
<i>Purpose of the Present Work.</i>	6
<i>Methods of Approach - Mathematical.</i>	6
<i>Methods of Approach - Engineering.</i>	8
BODY	9
<i>Refinement of the Mathematical Technique.</i>	9
<i>Implementation of the Technique in Hardware.</i>	13
CONCLUSIONS	15
REFERENCES	16
APPENDIX	18



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

INTRODUCTION

Nature of the Problem. Contamination of the observed electroencephalogram (EEG) by physiological artifacts such as eye movements and blinks (electrooculogram, or EOG) is a classical problem in electrophysiological studies. EOG artifacts are a major impediment in the recording and analysis of the EEG; the problem was first reported as early as 1941 and reports and attempted solutions continue to be published as recently as this year. The EEG (both steady state and evoked potential) is an important tool in the diagnosis of neurological dysfunction, such as epilepsy, cerebrovascular trauma and brain tumor, as well as in sleep studies and determining workload, mental state, and limitations of sensory information processing. Our approach uses a direct interrogation (signal injection) technique which is substantially different from the *a posteriori* techniques published by others. Direct interrogation permits on-line, virtually real time correction of eye movement artifacts on the observed EEG. We conducted a research program to add the capability of blink artifact removal to our current artifact rejection technique and implement this in a hardware device. The capability of completely correcting the EEG for EOG artifacts offers a number of significant benefits. It simplifies the acquisition and analysis of the steady state EEG and evoked potentials (EP) in clinical studies. In addition, it simplifies the development of algorithms for machine analysis of the EEG and EP; such analysis has been thwarted by the presence of EOG artifacts.

Background. A biological signal is propagated from its site of origin to a site of measurement through a medium that can, in principle, be described by a transfer function. A transfer function is merely a mathematical construct describing the relationship between the "input" and the "output" of the medium. This description can be accomplished equivalently in the time domain or, after Fourier transformation, in the frequency domain. Theoretically, the transfer function can be completely determined by measuring the response of the medium to a unit impulse; the unit impulse response is the inverse transform of the transfer function. Convolution of the unit impulse response with any input always yields the output (Van Valkenburg, 1964).

It is important to note that unique transfer functions can only be defined for linear systems. All naturally-occurring systems are, in the final theoretical analysis, distributed-parameter, non-stationary, quantized, stochastic, nonlinear systems. However, it has been repeatedly proven that modelling naturally-occurring systems as nearly lumped-parameter, stationary, continuous, deterministic, linear systems can yield practically useful results (Riggs, 1970).

Observed biological signals, such as EEG, are composites of the signal of interest and other unwanted, but nevertheless, real and physiological signals. These unwanted signals are traditionally termed "artifacts". In theory, there are three ways of eliminating unwanted signals - prevention, traditional filtering, and filtering after estimation of the transfer function.

Prevention is possible for many, but not all, unwanted signals. Technical artifacts, such as 60 Hz "hum" and electrochemical effects at the electrode - tissue interface, are preventable. Certainly, artifact prevention is the ideal. However, physiological artifacts are not preventable; it is clearly "undesirable" to prevent eye movements in a visual tracking task OR to stop the maternal heart during an attempt to obtain the fetal ECG!

Traditional filtering is useful for many, but not all, unwanted signals. When there is good frequency separation between the signal of interest and the unwanted signals, filters such as Butterworth, Tschebychev, Bessel, and Cauer are useful. However, in the absence of good frequency separation (or when the bandwidth of the signal of interest overlaps the bandwidth of the unwanted signals), traditional filtering results in loss of data and signal distortion. Steep roll-off of these filters (higher order forms) results in significant phase distortion, which is often overlooked in visual analysis (Johnson et al, 1979). Often, EEG data which is deemed contaminated based on some subjective or pre-programmed criterion is simply rejected (Gratton et al, 1983) or masked (Barlow, 1985).

The problem of determining the state of a system from noisy measurements is called estimation (or filtering). With a state-space approach, the dynamical system is modeled by a finite-

dimensional Markov process; the conditional probability density function of the state embodies all the information, which is available from the measurements (Jazwinski, 1970). All estimates of the state can be constructed from the density function, allowing formulation of linear and nonlinear filters and predictors. Removal of eye movement "artifacts" from observed EEG measurements is an example of such a problem. This is not merely a problem of theoretical interest; eye movements and eye blinks are a permanent source of serious unwanted signals in the measurement and analysis of the electroencephalogram (Lyman, 1941; Case, 1959; Corby & Kopell, 1972; Girton & Kamiya, 1973; Matsuo et al, 1975; Gevins et al, 1977; Whitton et al, 1978; Barlow & Remond, 1981; Verleger et al, 1982; Gratton et al, 1983; Fortgens & De Bruin, 1983; Woestenburg et al 1983; Elbert et al, 1985) and are especially troublesome in event-related brain potential measurements (e.g. CNV and P300) (Hillyard & Galambos, 1970; Wasman et al, 1970; Girton & Kamiya, 1973).

Before we ask the question "What is the transfer function which describes the coupling of the EOG to the observed EEG?", let us first review the biophysics underlying the process.

In electrodynamic terms, the eye can be modeled as a dipole (Mowrer et al, 1935; Barry & Jones, 1965). In this representation, the corneo-retinal potential difference is the result of charge separation, with the corneal aspect being positive and the retinal aspect being negative. The corneo-retinal potential is known to vary considerably between individuals (Shackel, 1967) as well as within individuals as a function of illumination (Rubin & Walls, 1969) and time (Shackel & Davis, 1960). It is further assumed (Fortgens and De Bruin, 1983; Elbert et al, 1985) that under normal conditions both eyes move conjugately. The motion of these linked dipoles creates potentials, which are observable at distant EEG recording sites. In addition to the potentials created by this dipole motion, potentials are also generated by the eyelid, acting like a sliding electrode, picking up positive potential moving across the positively charged corneal surface (Matsuo et al, 1975). The resultant change in charge distribution caused by closure of the eyelid can also be described by a change in dipole moment. Thus, the EOG can be described as an electrical potential resulting from a change in ocular dipole moment (Girton & Kamiya, 1973; Elbert et al, 1985).

The EOG potential is propagated through the medium to sites all over the body; this includes the head, which may be modeled as a four layer sphere of different conductivities (Cuffin & Cohen, 1979). The relationship between the EOG potential (measured, say, near its source) and the propagated EOG potential, at some distant site, is completely described by the transfer function of the medium (assumed linear). Numerous workers have attempted to exploit this relationship in order to remove the unwanted EOG from the observed EEG.

In its simplest form, the transfer function might be assumed to be constant and unity. However, if we merely subtract the measured EOG from an EEG measured at distant scalp sites, it is obvious that this "correction" will yield an erroneous estimate of the real EEG (Gratton et al, 1983). The transfer function could be assumed to be constant, but not unity (Barlow & Remond, 1981); however, this would not take into consideration the known dependence on the distance to the EEG electrode site from the eye (Girton & Kamiya, 1973). The transfer function could be assumed to be distance dependent; however, this does not take into consideration the known frequency and phase angle dependence of propagated volume conductor potentials (Gevins et al, 1977; Whitton et al, 1978; Woestenburg et al, 1983; Elbert et al, 1985).

From a theoretical point of view, the transfer function describing the propagation of the EOG potential ($v_i[t]$) through the medium to the distant recording site ($v_o[t]$), is a function $h[t]=h[d,A,f,\phi,t]$, such that:

$$\text{where:} \quad v_o[t] = \int v_i[\mu] h[t-\mu] d\mu$$

v_i = input potential
 v_o = output potential
 d = distance
 A = amplitude
 f = frequency
 ϕ = phase angle
 t = time

From a practical point of view, distance dependence of the transfer function can be ignored for specific, fixed electrode sites (such as in a single recording session). Frequency dependence (Gevins et al, 1977; Whitton et al, 1978; Elbert et al, 1985) and phase angle dependence (Whitton et al, 1978; Woestenburger et al, 1983) cannot be ignored. We can find no evidence in the literature for amplitude dependence; yet, this does not mean that it can be arbitrarily ignored without investigation. Finally, time dependence of the transfer function should not be ignored (except possibly in very short duration recording sessions), since it is a fundamental premise that biological systems change with time. An intuitive illustration of this might be the temporally-dependent impedance changes resulting from perspiration.

An analysis of the published literature clearly indicates that thinking in this area has been slowly evolving to the aforementioned full theoretical form of the transfer function and the parameters it depends upon. In fact, based on the 1985 work of Elbert et al, it appears that direct interrogation (as described later in this proposal) is the next logical area of investigation for the removal of the EOG from the EEG.

Numerous attempts have been made to estimate the transfer functions of unwanted biological signals. Bergveld & Meijer (1981) have reported a technique for removing the maternal ECG from abdominal electrocardiograms, in order to obtain a fetal ECG as well as a technique for determining the ideal electrode position (Meijer & Bergveld, 1981). They postulate a transfer function composed of the linear combination of three independent observation sites and attempt to estimate the coefficients of this linear combination. Johnson et al (1979) have reported a technique for removal of muscle artifact from the electroencephalogram. They formulate a nonlinear estimator (filter) based on an a priori model of the EEG (represented as the superposition of four lightly damped oscillators, operating in the alpha, beta, theta, and delta bands, driven by independent white Gaussian noises) and an a priori model of the muscle artifact (represented by the superposition of "action potentials" of three different durations generated as impulse responses of three linear systems driven by independent Poisson processes). Techniques for removing the EOG "artifact" from the EEG have been reported by Verleger et al (1982), Gratton et al (1983), Woestenburger et al (1983), and Elbert et al (1985).

Verleger et al report "completely correcting for blink effects", but only partial correction of eye movement artifact; this is in contrast to Weerts & Lang (1973) who "presumably removed the eye movement effect correctly, but overcompensated for the blink effect" (Verleger et al, 1982). They use a regression approach consisting of:

- a. identifying maximum variance EOG segments;
- b. estimating a linear regression coefficient;
- c. estimating a general transmission rate;
- d. correcting the EOG for DC bias; and
- e. subtracting the weighted EOG from the observed EEG.

Gratton et al (1983) use a somewhat different approach. Their procedure consists of:

- a. estimating correction factors derived from EOG and EEG data obtained during, rather than before, the experiment;
- b. estimating separate correction factors for blinks and eye movements;
- c. removing event-related EOG and EEG activity from the data; and
- d. subtracting the weighted EOG from the observed EEG.

They state that their approach has six clear advantages: it distinguishes between blink and eye movement artifact; it provides corrections that are insensitive to stimulus-locked activity; it retains all data for use in subsequent analyses; it does not require special data collection; the subjects need not control or minimize eye movements; and, the estimate is based on a large sample, rather than a

few data obtained from a few prescribed eye movements. They also properly point out that "noise" in the measured EOG may significantly alter the magnitude of the estimated correction factor.

Woestenburg et al (1983) report a technique for removing the eye movement artifact from the EEG by regression analysis in the frequency domain. They explicitly recognize and demonstrate that the transfer of eye movement activity to EEG can have frequency dependent amplitude and phase characteristics and they attempt to determine the transfer function. They assume that the medium is passive and constant and that there is no linear correlation between EOG and EEG activity. Furthermore, they state that "a successful method for removing the EOG artifact from the EEG should be able to handle the following phenomena:

- a. Transfer from EOG on EEG is frequency dependent. Some frequencies may be attenuated more than other frequencies.
- b. The EOG artifact as measured at the scalp can be distorted by phase-shifts.
- c. Both vertical and horizontal eye movements may contribute to the artifact."

Woestenburg et al (1983) applied their technique to simulated data as well as to real data. The principal limitation of their technique is that it is an *a posteriori* approach typically requiring two blocks of 36 complex visual stimulus presentations and about one hour of computer time for data analysis.

Elbert et al (1985) use a biophysical approach to the theoretical formulation of the electrodynamic equations, which allow a complete description of the ocular influence in the EEG. They separate the transfer function, describing the ocular influence in the EEG, into vertical, lateral, and radial components and attempt to identify (but do not adequately support) the minimum necessary and sufficient EOG electrodes and their anatomical positions. Elbert et al explicitly recognize the frequency dependence of the transfer function; they report the form of the vertical component ($g(\omega, C_z)$) as a function of radial frequency ($\omega = 2\pi f$) as measured at the C_z . They both report theoretical and empirical forms. There are two empirical forms reported. One form, attributed to Gasser et al, is derived from naturally occurring ocular artifacts. The other form was derived following application of an (unspecified) artificial drive signal to the EOG electrodes.

The application of this artificial drive signal, by Elbert et al, forms the published "springboard" of our research efforts. The artificial drive signal, applied to the EOG electrodes, is an example of direct interrogation of the biological system under consideration. In keeping with the theoretical approach to determining the transfer function, it allows us to apply a "unit impulse", so as to completely describe the real transfer function. Judicious selection of an externally applied drive signal, when properly utilized, can be a safe, effective, and noninvasive means of determining the transfer function of the ocular influence on the EEG. An artificial drive signal has already been applied by Elbert et al (1985) and by us (unpublished, 1985 and Falk et al, 1987). Sullivan (1965) reported use of a 40 KHz drive signal for measuring impedance in order to determine the direction of the ocular dipole.

All the previously cited literature (with the exception of Elbert et al) attempt to determine the transfer function (correction factor, weighting factor, regression coefficient, etc.) through the use of naturally occurring ocular motions. Since there are, potentially, an infinite number of different ocular motions, selection of specific motions (Weerts & Lang, 1973; Verleger et al, 1982; Fortgens & De Bruin, 1983) obviously lacks generality and completeness. The work of Woestenburg et al (1983), and then Gratton et al (1983), begins to circumvent this problem by basing the estimate on a large sample, rather than a few data obtained from a few prescribed eye movements. But even this approach does not fully address the problem. Our approach is to apply an external drive signal which describes all possible ocular motions; these ocular motions are merely an electrical signature composed of particular amplitudes at particular frequencies with particular phase relations. In fact, because the biopotentials generated by ocular motion are not unbounded, the EOG does NOT contain all possible amplitudes and frequencies; the EOG is constrained to frequencies below, say

for example, 30 Hz and to amplitudes below, say for example, 5 mV. Therefore, practically, the "unit impulse" required to theoretically determine the transfer function need not be an impulse input, $\delta(t)$; instead, it can be a relatively short time duration pulse whose frequency transform includes those frequencies of interest.

Purpose of the Present Work. The purposes of this research study were twofold: first, to refine our existing mathematical technique, and second, to implement it in a portable, battery-operated twelve channel device.

Methods of Approach - Mathematical. The mathematical technique that allows us to remove the eye movements from the on-going EEG is called the direct interrogation technique. Three basic assumptions are made in order to utilize this technique. We assume that the eye movement signal propagates only on the surface of the head to the distant EEG sites. Since the skull is approximately eighty times the resistivity of the scalp, the path of least resistance is the scalp. Depth electrode studies have been conducted and there was no evidence of EOG artifact in the EEG (Cooper, 1971). We also assume that the medium is linear (thus the theory of superposition holds) and the medium is non-dispersive (no shift in frequency). We have tested both these assumptions and we find them to be true. With this as our base, we can model this system as an input (EOG), an output (EOG artifact on the EEG), and a medium (scalp) and its transfer function.

Before we discuss the models of eye movement and eye blink, we must discuss some terminology regarding the ocular dipole. An electric *dipole* is an electric potential source arising from the separation of equal and opposite charges and resulting in an electric field whose magnitude is nonzero at all points in space except those equidistant from both charges. These equidistant points define a unique *zero-potential plane* orthogonal to the line connecting both charges. The *ocular dipole* is an electric dipole with the positive charge on the cornea and the negative charge on the retina. The *conjugate eye dipole pair* is comprised of the two linked ocular dipoles that move in parallel. The *surface image* of a dipole is that portion of the electric field residing on a surface transecting the three dimensional dipole electric field. The image of the zero-potential plane on the surface is a *zero-potential line*. The *surface image of the ocular dipole* is the image on that surface defined by the skin on the head (including the face). The zero-potential line forms an angle ϕ with the x-axis of our geometrical coordinate system. A direct interrogation stimulus dipole or "*surface stimulus dipole*" is the electric source resulting from the application of two spaced surface electrodes driven by a floating voltage source (a floating battery).

We made the following *explicit* assumptions: (a) the EOG signal reaches the EEG recording site via surface propagation (propagation by other means is negligible); (b) the medium is passive and constant (over relatively short time periods); (c) the principle of superposition holds (the system is linear or nearly linear) and a unique transfer function does exist; (d) the medium is non-dispersive (frequencies don't change during propagation); and (e) our mathematical model properly represents the electrodynamic behavior of the conjugate eye dipole pair. While further investigation is required, we presently believe that no other *implicit* assumptions have been made.

CORRection of the observed EEG for EOG artifacts ($^{corr}V_{EEG}[\omega]$) is accomplished in the frequency domain and is based on (a) measurement of the **OBServed** EEG ($^{obs}V_{EEG}[\omega]$) and **OBServed** EOG ($^{obs}V_{EOG}[\omega]$), (b) measurement of the system response to **STIMulation** ($^{stim}V_{EEG}[\omega]$ & $^{stim}V_{EOG}[\omega]$) for direct interrogation, (c) a mathematical model that describes the electrodynamic behavior of the system for **Theoretical Eye Movements** ($^{tem}V_{EEG}[\omega]$ & $^{tem}mV_{EOG}[\omega]$) and **Theoretical Direct Interrogation** ($^{tdi}V_{EEG}[\omega]$ & $^{tdi}V_{EOG}[\omega]$), and (d) measurement of the **CALibration** of each recording channel ($^{cal}V_{EEG}[\omega]$ & $^{cal}V_{EOG}[\omega]$). The mathematical derivation is summarized here.

The formula for implementing the EEG correction, on a frequency per frequency basis, is:

$${}^{\text{corr}} V_{\text{EEG}} [\omega] = {}^{\text{obs}} V_{\text{EEG}} [\omega] - (S \times D / G) {}^{\text{obs}} V_{\text{EOG}} [\omega]$$

where:

$$S = {}^{\text{stim}} V_{\text{EEG}} [\omega] + {}^{\text{stim}} V_{\text{EOG}} [\omega] \quad (\text{using } 20 \mu\text{A stimulus pulse})$$

$$G = {}^{\text{cal}} V_{\text{EEG}} [\omega] + {}^{\text{cal}} V_{\text{EOG}} [\omega] \quad (\text{using } 1 \text{ mV calibration pulse})$$

$$D = \{ {}^{\text{tem}} V_{\text{EEG}} [\omega] + {}^{\text{tem}} V_{\text{EOG}} [\omega] \} + \{ {}^{\text{tdi}} V_{\text{EEG}} [\omega] + {}^{\text{tdi}} V_{\text{EOG}} [\omega] \}$$

S is a measure of the system response to direct interrogation and is the ratio of the signals observed at the EEG and EOG recording sites; it is the putative transfer function. G is a measure of the discrepancy between the recording channels and is the ratio of the calibration signals observed at the EEG and EOG recording sites; G would not be necessary, if and only if the recording channels were absolutely identical. D is a geometrical correction factor that interrelates the theoretical electrodynamic behavior of the (non-collocated) direct interrogation stimulus dipoles and the ocular dipoles; it is, in fact, our mathematical model. It must contain both magnitude and phase information, so it has the form:

$$D = D' e^{i\xi}$$

where D' describes the magnitude correction due to geometry and ξ describes the phase correction due to geometry. The geometrical correction factor D would not be necessary, if and only if the direct interrogation stimulus dipole exactly and completely emulated the ocular dipoles geometrically and electro-dynamically.

D' was derived by obtaining the general solution of the general differential equation that describes the propagation of a potential generated by any source. The general solution was constrained to model a dipole source. Using this equation, the conjugate eye dipoles were resolved into a single equivalent theoretical source located at the origin of our selected coordinate system. Similarly, by coordinate transformation, the stimulus dipoles were converted to an equivalent theoretical source also located at the origin of our coordinate system. With these two source equations, the magnitude relationship of the signals expected at the EEG and EOG recording sites (as a result of eye movements versus surface dipole stimulation) was computed. This permits computation of the magnitude portion of the geometrical correction factor; it is used to correct the empirical transfer function (found by direct interrogation stimulation) for the difference in geometry between the stimulus dipoles and ocular dipoles.

Phase changes due to propagation through the medium and this information is contained in the empirical transfer function obtained by direct interrogation. Additionally, there is a relative phase shift between the EEG and EOG recording sites. It is due solely to the changing geometric orientation of the isopotential lines caused by rotation of the surface image of the ocular dipole. This information is not contained in the direct interrogation data and must be independently corrected. The equation describing the single equivalent theoretical source of the conjugate eye dipole is a function of the angle of rotation ϕ of the surface image of the ocular dipole pair. Differentiation of this equation with respect to ϕ yields an equation describing the change in potential at an EOG electrode due to a change in ϕ . When the change in potential with respect to ϕ is zero, the potential is at an extremum (maximum or minimum) and the corresponding ϕ , at a particular electrode site, can be computed. This value of ϕ is the value of the angle of rotation that creates an extremum at the particular electrode site under consideration. It will have different values for different electrode sites. The geometrically-dependent relative phase shift between an arbitrary pair of electrode sites is the difference of their corresponding ϕ 's. A change in ϕ can not be determined from one EOG electrode; in general, an orthogonal pair is preferred.

Our mathematical technique can be summarized as follows. Integral to our technique are the following four (4) explicit assumptions:

- a. the EOG artifact on the EEG is the result of an electrodynamic process, arising from the movement of the eye dipoles and from the eyelids across their surface (Elbert et al, 1985);
- b. the EOG artifact reaches the EEG recording site primarily via surface propagation (propagation by other means is negligible) (Cooper et al, 1965, 1971; Cuffin and Cohen, 1979);
- c. the surface propagation medium is passive, linear, and non-dispersive (over relatively short time periods); thus, a unique transfer function exists - this was shown in our feasibility demonstration; and
- d. all possible eye movements and blinks are completely described by their Fourier components, and these consist of a bounded set of frequencies, amplitudes, and phases.

Therefore, EOG propagation between the site of EOG generation and the EEG electrodes can be characterized by a transfer function; the transfer function in turn can be characterized by injecting a signal at the EOG generation site and recording the resultant signal at the EEG electrodes (direct interrogation).

This method of rejecting ocular motion artifacts on the EEG recording can be mathematically expressed as:

$$EEG^c(t) = EEG^o(t) - IFT (EOG^o(s) \times S(s) \times G(s) \times D)$$

where: $EEG^c(t)$	=	Corrected EEG (time domain)
$EEG^o(t)$	=	Observed EEG (time domain)
IFT	=	Inverse Fourier Transform
$EOG^o(s)$	=	Observed EOG (frequency domain)
$S(s)$	=	Transfer function (frequency domain)
$G(s)$	=	Channel response correction factor (frequency domain)
D	=	Geometric correction factor

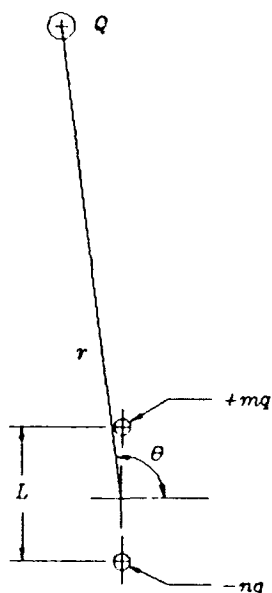
Methods of Approach - Engineering. The mathematical technique is an algorithm for removing the unwanted influence of the EOG on the observed EEG. This can be implemented in hardware by constructing a microprocessor-based device which can be programmed to execute this algorithm. Amplifiers and filters are used to condition biopotential signals which can then be digitized and processed. These processed data can be once again converted to analog signals for display and recording. The digital processing time will introduce a finite delay due to the time required for the microprocessor to execute the necessary computations. Standard engineering techniques permit implementation of the analog and digital circuitry in a form that requires minimal power, and thus can be battery operated.

BODY

Refinement of the Mathematical Technique. In this research study, we have expanded the biophysical model to include the blink. This yields a general electrodynamic model for both the source and the propagating electric field from the eye for all possible eye movements and blinks.

The Biophysical Model. The transection of the face across the three dimensional ocular dipole field (caused by the corneo-retinal potential in the eye) yields a surface image dipole propagating on the scalp. This surface image dipole can be modelled to incorporate both the eye movement and the eye blink. The eye movement produces a symmetric dipole, while the blink produces an asymmetric dipole.

General Dipole Representation. A dipole source, symmetric or asymmetric, is the superposition of two point sources separated by a distance. The point source's electric field propagates as a function of $1/r^2$. The voltage at any point is described by $V=kq/r$, where k is Boltzmann's constant, and q is the amount of charge. The surface image dipole is described here.



The figure on the left shows two point sources separated by a distance (L). The voltage (V) appearing at point Q is derived as follows.

$$V = kq\{[m/(r-\frac{1}{2}L\sin\theta)] + [-n/(r+\frac{1}{2}L\sin\theta)]\}. \quad (1)$$

Rearranging Equation 1 yields

$$V = kq\{[(m-n)r + (m+n)\frac{1}{2}L\sin\theta]/(r^2 - \frac{1}{4}L^2\sin^2\theta)\} \quad (2)$$

Since $r \gg L$, we can simplify Equation 2:

$$V = n[(kq/r^2)\frac{1}{2}L\sin\theta(\alpha+1) + (kq/r)(\alpha-1)], \quad (3)$$

where $\alpha = m/n$.

As a note, if $\alpha=1$ (eye movement) and the dipole is symmetric, Equation 3 reduces to,

$$V = (nkq/r^2)L\sin\theta = Ar^{-2}\sin\theta. \quad (4)$$

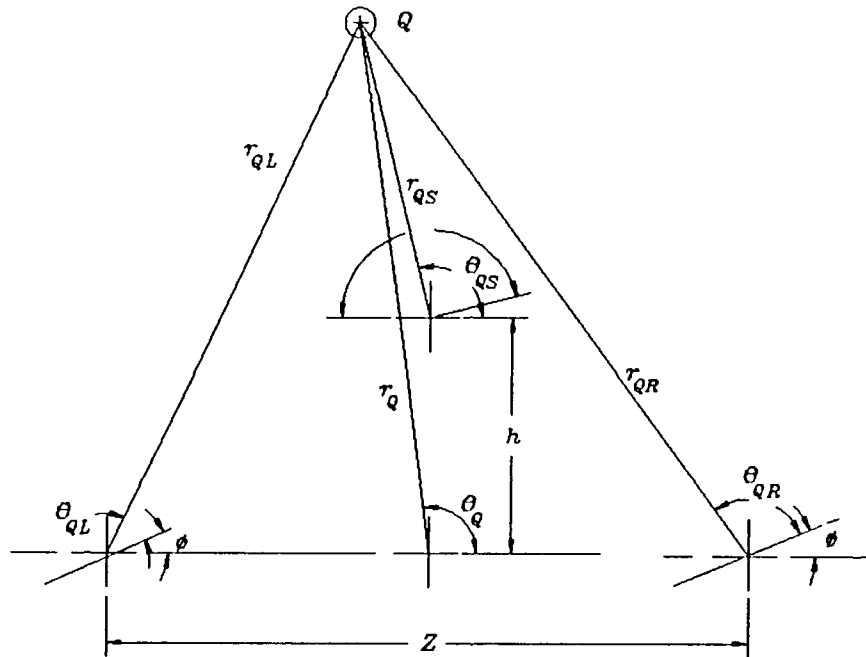
where $A = nrqL$.

Furthermore, it is important to note that the zero-potential line of the dipole is the x-axis when $\alpha=1$ ($\sin\theta=0$). When $\alpha \neq 1$, the zero-potential line becomes a circle described by,

$$x^2 + (y+G)^2 = G^2, \quad (5)$$

where $G = \frac{1}{2}L[(\alpha+1)/(\alpha-1)]$.

Selection of a Facial Coordinate System. In order to spatially represent the ocular dipoles or the stimulus dipole in planar geometry, we must select a coordinate system. This is shown in the figure below. The point, Q, in this figure represents an electrode. The subscript L is used to show reference to the left eye, the subscript R is used to reference the right eye. The electrode is a distance β_L from the left eye and β_R from the right eye. Reference to the stimulation dipole is indicated by S. Our facial coordinate system has its origin at the geometric center of the two eyes (which are separated by a distance z). The stimulus dipole is located a distance h above the origin. The electrode is a distance r from the origin of the coordinate system. The stimulation dipole lies on the y-axis as the eyes lie on the x-axis.



Spatial Resolution of the Bi-ocular Dipoles. Using our coordinate system and dipole representation described above, we will model the two ocular dipoles as one complex mathematical function based at the origin of the coordinate system.

From the law of cosines:

$$\beta_L = [r^2 + \frac{1}{4}z^2 + rZ\cos\theta]^{\frac{1}{2}} \quad (6)$$

$$\beta_R = [r^2 + \frac{1}{4}z^2 - rZ\cos\theta]^{\frac{1}{2}} \quad (7)$$

From the law of sines:

$$\gamma_L = -\phi + \sin^{-1}[(r/\beta_L)\sin\theta] \quad (8)$$

$$\gamma_R = \pi - \phi - \sin^{-1}[(r/\beta_R)\sin\theta] \quad (9)$$

where: ϕ = angle of the ocular dipoles (zero-potential line).

We now substitute these identities (Eqns. 6-9) into the dipole equation defined earlier (Eqn. 3) and sum the two ocular dipoles to result in one function. We obtain this, in a general form, for any electrode:

$$V = \frac{1}{2}(\alpha+1)A\{[\beta_L^{-2}[-(1-\gamma_L^2)^{1/2}]\sin\phi+\gamma_L\cos\phi\}+[\beta_R^{-2}[(1-\gamma_R^2)^{1/2}]\sin\phi+\gamma_R\cos\phi\}+(A/L)(\alpha-1)(\beta_L^{-1}+\beta_R^{-1}), \quad (10)$$

where:

$$\begin{aligned}\beta_L &= [(r)^2+(\frac{1}{2}Z)^2+rZ\cos\theta]^{1/2} \\ \beta_R &= [(r)^2+(\frac{1}{2}Z)^2-rZ\cos\theta]^{1/2} \\ \gamma_L &= (r/\beta_L)\sin\theta \\ \gamma_R &= (r/\beta_R)\sin\theta\end{aligned}$$

Spatial Representation of the Stimulus Dipoles. We can similarly describe the representation of the stimulus dipole in our new coordinate system as we have described the ocular dipoles above. Using Equation 4:

$$V = B\{\sigma^{-2}[\eta\sin\psi-(1-\eta^2)^{1/2}\cos\psi]\} \quad (11)$$

where:

$$\begin{aligned}\sigma &= [r^2 + h^2 - 2rh\sin\theta]^{1/2} \\ \eta &= (-r/\sigma)\cos\theta \\ \psi &= \text{angle of the stimulus dipole (an analog of } \phi)\end{aligned}$$

Equations 10 & 11 and the associated identities are the basis of the mathematical model which will be used in the calculation of the transfer function.

Mathematical Relationship Between Resultant Ocular Dipoles & Stimulus Dipoles. We can now use our basic equations and our coordinate system to correct the putative transfer function measured by surface dipoles. The EOG artifact correction equation in the frequency domain is:

$${}^{\text{corr}} V_{\text{EEG}} = {}^{\text{obs}} V_{\text{EEG}} - {}^{\text{obs}} V_{\text{EOG}} [S_{\text{EEG}} / S_{\text{EOG}}] D \quad (12)$$

where D is the geometrical correction factor between the stimulus dipole and the ocular dipoles. S denotes the surface dipole stimulation response and ${}^{\text{obs}} V$ denotes the naturally occurring response. The subscript "EEG" and "EOG" refer to the electrode recording the response; the superscripts "corr" and "obs" refer to the corrected and observed potential, respectively.

We can obtain D by manipulation of the equations described above.

$$D = \frac{\frac{\{\frac{1}{2}(\alpha+1)[\beta_L^{-2}[-(1-\gamma_L^2)^{1/2}]\sin\phi + \gamma_L\cos\phi] + [\beta_R^{-2}[(1-\gamma_R^2)^{1/2}]\sin\phi + \gamma_R\cos\phi] + ((\alpha-1)/L)[\beta_L^{-1} + \beta_R^{-1}]\}_{\text{EEG}}}{\{\frac{1}{2}(\alpha+1)[\beta_L^{-2}[-(1-\gamma_L^2)^{1/2}]\sin\phi + \gamma_L\cos\phi] + [\beta_R^{-2}[(1-\gamma_R^2)^{1/2}]\sin\phi + \gamma_R\cos\phi] + ((\alpha-1)/L)[\beta_L^{-1} + \beta_R^{-1}]\}_{\text{EOG}}}}{\frac{\sigma^{-2}[\eta\sin\psi - (1-\eta^2)^{1/2}\cos\psi]_{\text{EEG}}}{\sigma^{-2}[\eta\sin\psi - (1-\eta^2)^{1/2}\cos\psi]_{\text{EOG}}}} \quad (13)$$

where,

$$\beta_L = [(r)^2 + (\frac{1}{2}Z)^2 + rZ\cos\theta]^{1/2},$$

$$\beta_R = [(r)^2 + (\frac{1}{2}Z)^2 - rZ\cos\theta]^{1/2},$$

$$\gamma_L = (r/\beta_L)\sin\theta,$$

$$\gamma_R = (r/\beta_R)\sin\theta,$$

$$\sigma = [r^2 + h^2 - 2rh\sin\theta]^{1/2},$$

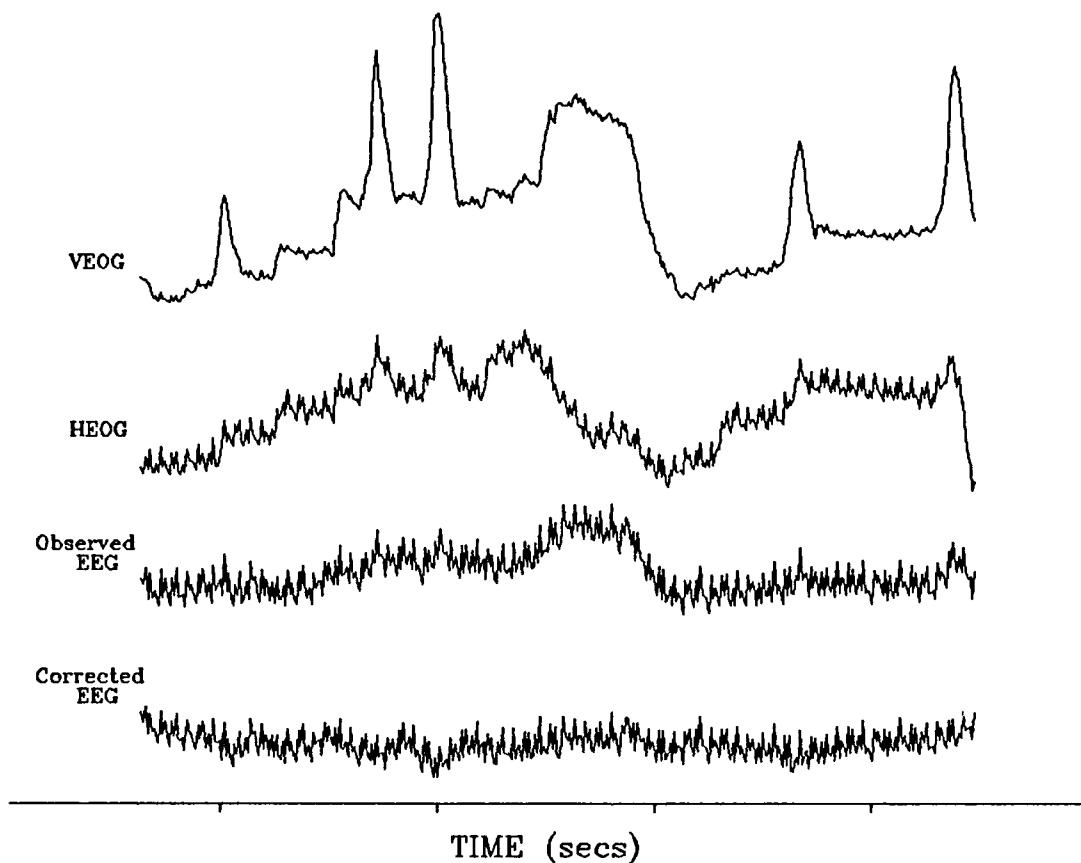
$$\eta = (-r/\sigma)\cos\theta,$$

$$\phi = \begin{cases} 0^\circ & \text{for vertical EOG} \\ 90^\circ & \text{for horizontal EOG} \end{cases}$$

$$\psi = \begin{cases} 0^\circ & \text{for vertical interrogation pulse} \\ 90^\circ & \text{for horizontal interrogation pulse} \end{cases}$$

$$\alpha = m/n = (\text{obtained in real time as } \text{EOG}_{\text{vertical}}^{\text{upper}} / \text{EOG}_{\text{vertical}}^{\text{lower}}).$$

We measure z and h , as well as β_L and β_R for each electrode (EEG and EOG). We then calculate r and θ for each electrode, and then calculate γ_L , γ_R , σ , and η for each electrode. Finally, we calculate D 's for each EEG/EOG electrode combination. This permits correction of the observed EEG in accordance with Equation 12. The next figure shows an example of a correction. The cross-correlation between the observed EEG and the vertical EOG was 0.75; the cross-correlation between the corrected EEG and the vertical EOG was 0.017.



Implementation of the Technique in Hardware. The implementation of the technique described above required an extremely fast microprocessor. The specification that the portable, light-weight device must fit in a flight suit pocket required a low-power CMOS microcontroller. These specifications resulted in the selection of the Intel 80C196 microcontroller. This chip contains a very fast microprocessor, an on-board analog-to-digital converter, extremely low power consumption, and an already written and tested Fast Fourier algorithm.

There were many obstacles encountered with the use of this microcontroller. There is a design flaw in the chip. Intel has since published this flaw and has an updated chip. The flaw is in the unsigned divide instruction. The result from this instruction is either the correct answer or one least significant bit away from the correct answer. This doesn't seem like a major problem on the surface, however in a thirty two bit divide algorithm, the unsigned divide is used. What intermittently occurs is an incorrect answer which is off by one least significant bit in the **HIGH** word; the result is that the numerical answer is off by 65,536!

Another flaw in the chip is that the on-board eighty bytes of RAM is sporadically overwritten. If variables located in the onboard RAM are forced into the external RAM space, the problem seems to disappear.

There is a flaw in the C compiler written for the 80C196. A locally defined variable is being overwritten by a subroutine containing the same, but locally defined, variable. Using identically defined but locally defined variables is standard and "legal" in C, yet this compiler does not seem to properly handle this situation.

There is another flaw in Intel's system. The in-circuit emulation system, used to develop software for the C196, defines the ROM as zero wait state memory. This causes major timing problems, because the ROM should be activated with the user programmable wait states, which can be either one, two, or three. Yet, the emulator disregards this programmable wait state number and accesses the ROM in zero wait states. Intel has been notified of this timing flaw. This undocumented discrepancy makes the software created on the emulator incompatible with the Intel target hardware that would be used in a portable device.

There also have been other general problems plaguing this effort. This microprocessor is an integer based machine. This leads to two hurdles. First, the resolution of the mathematics is truncated to digital steps and not continuous functions. The ratio of one to one half is two in the continuous world, yet the ratio of one to zero (one half is truncated to zero) is infinity in the digital world. The second hurdle is that the integer set is bounded at -32,768 to 32,767. This constraint causes the programmer to scale numbers down as they grow close to the bounds. This is a double-edged sword, since the function of scaling is division, which leads to truncation!

This constraint of integer math caused us to require the use of a host PC and to perform the calculations of the model parameters and the transfer function on the host PC (since floating point arithmetic is necessary here). This eliminates the option of continually interrogating the medium while the subject is ambulatory.

The blink component of the model, although an excellent advance in the biophysical model, added complexity to the correction technique. This complexity added a significant amount of computation time to the microprocessor based program. This result was that there was only time for one channel to be corrected with the full model.

In order to achieve this correction of one channel in the allotted (real) time, there were several "shortcuts" that were necessary. The vertical and horizontal transfer functions were reduced from an array of complex numbers (one for each frequency) to one complex number. We showed that the transfer function varied less than ten percent over the frequency spectrum. This allowed us to reduce these arrays, yet it is a practical variation from the theoretical ideal. Another shortcut was the elimination of the square root. In calculating the absolute value of the ratio of the upper VEOG to the lower VEOG, a square root was necessary. We showed that the imaginary component of the complex ratio was very close to zero, so we took the real component of the ratio instead of the absolute value.

There were many technical obstacles that complicated this research study and prevented us from correcting twelve channels of EEG for EOG artifact. We have successfully fabricated a device that will accurately correct one channel of EEG for EOG artifact. As indicated in the previous discussion, it is susceptible to sporadic failures caused by the Intel 80C196. Application specific circuits and chips can be used to implement this correction technique on multiple channels, however the power consumption will cause the battery size and weight to increase significantly. This would result in a device too large and heavy to place in a pocket or wear on the body.

CONCLUSIONS

From a scientific point of view, this project was a great success in that the mathematical technique was extended to handle blink artifacts in a non-arbitrary biophysically based manner. From an engineering point of view, the project was not a great success in that technological limitations (computing speed of CMOS processors) prevented the microprocessor from correcting more than one EEG channel in nearly real-time.

There were many technical obstacles that complicated this research study and prevented us from correcting twelve channels of EEG for EOG artifact. We have successfully fabricated a device that will correct one channel of EEG for EOG artifact. Application specific circuits and chips could be used to implement this correction technique on multiple channels, however the power consumption will cause the battery size and weight to increase significantly. This would result in a device too large and heavy to place in a pocket or wear on the body. Full implementation of a multichannel man-borne device must wait advances in computer hardware technology.

REFERENCES

- Barry, W. and Jones, M., Influence of eye lid movement upon electrooculographic recording of vertical eye movements, *Aerospace Med.*, 36:855-888, 1965.
- Barlow, J.S., A general-purpose automatic multichannel electronic switch for EEG artifact elimination, *Electroenceph. Clin. Neurophysiol.*, 60:174-176, 1985.
- Barlow, J.S. and Remond, A., Eye movement artifact nulling in EEGs by multichannel on-line EOG subtraction, *Electroenceph. Clin. Neurophysiol.*, 52:418-423, 1981.
- Bergveld, P. and Meijer, W.J., A new technique for the suppression of the MEOG, *IEEE Trans. Biomed. Eng.*, 28:348-354, 1981.
- Cooper, R., Recording changes in electrical properties in the brain: the EEG, in *Methods in Psychobiology, Laboratory Techniques in Neuropsychology and Neurobiology*, Myers, R.D. (Ed), New York, Academic Press, pp 191-194, 1971.
- Corby, J.C. and Kopell, B.S., Differential Contributions of blinks and vertical eye movements as artifacts in EEG recordings, *Psychophysiology*, 9:640-644, 1972.
- Cuffin, B.N. and Cohen, D., Comparison of the magnetoencephalogram and electroencephalogram, *Electroenceph. Clin. Neurophysiol.*, 47:132-146, 1979.
- Elbert, T., Lutzenberger, W., Rockstroh, B., and Birbaumer, N., Removal of ocular artifacts from the EEG - a biophysical approach to the EOG, *Electroenceph. Clin. Neurophysiol.*, 60:455-463, 1985.
- Fortgens, C. and De Bruin, M.P., Removal of eye movement and ECG artifacts from the non-cephalic reference EEG, *Electroenceph. Clin. Neurophysiol.*, 56:90-96, 1983.
- Gevens, A.S., Yeager, C.L., Zeitlin, G.M., Ancoli, S., and Dedon, M.F., On-line computer rejection of EEG artifact, *Electroenceph. Clin. Neurophysiol.*, 42:267-274, 1977.
- Girton, D.G., and Kamiya, J., A simple on-line technique for removing eye movement artifacts from the EEG, *Electroenceph. Clin. Neurophysiol.*, 34:212-216, 1973.
- Gratton, G., Coles, M.G.H., and Donchin, E., A new method for off-line removal of ocular artifact, *Electroenceph. Clin. Neurophysiol.*, 55:468-484, 1983.
- Hillyard, S.A. and Galambos, R., Eye movement artifact in the CNV, *Electroenceph. Clin. Neurophysiol.*, 28:173-182, 1970.
- Jazwinski, A.H., *Stochastic Processes and Filtering Theory*, New York, Academic Press, 1970.
- Johnson, T.L., Wright, S.C. and Segall, A., Filtering of muscle artifact from the electroencephalogram, *IEEE Trans. Biomed. Eng.*, 26:556-563, 1979.
- Lyman, R.S., Eye movements in the electroencephalogram, *Johns Hopk. Hosp. Bull.*, 68:1-31, 1941.
- Matsuo, F., Peters, J.F., and Reilly, E.L. Electrical phenomena associated with movements of the eyelid., *Electroenceph. Clin. Neurophysiol.*, 38:507-512, 1975.
- Meijer, W.J. and Bergveld, P., The simulation of the abdominal MEOG, *IEEE Trans. Biomed. Eng.*, 28:354-357, 1981.
- Mowrer, O.H., Ruch, T.C., and Miller, N.E., The corneo-retinal potential difference as the basis of the galvanic method of recording eye movements, *Am. J. Physiol.*, 114:423-428, 1935.
- Riggs, D. S., *Control Theory and Physiological Feedback Mechanisms*, Baltimore, Williams and Wilkins, 1970, 599 pgs.
- Rubin, M.L. and Walls, G.L., *Fundamentals of Visual Science*, Springfield, Thomas, 1969.
- Shackel, B., Eye movement recording by electro-oculography, in P.H. Venable and I. Martin (Eds.) *A Manual of Psychophysiological Methods*, Amsterdam, North Holland Publ. Co., 1967.
- Shackel, B. and Davis, J.R., A second survey with electro-oculography, *Brit. J. Ophthalmol.*, 44:337-346, 1960.
- Van Valkenburg, M.E., *Network Analysis*, New Jersey: Prentice Hall, 1964.
- Verleger, R., Gasser, T., and Mocks, J., Correction of EOG artifacts in event-related potentials of the EEG: Aspects of reliability and validity, *Psychophysiology*, 19:472-480, 1982.
- Wasman, M., Morehead, S.D., Lee, H., and Rowland, V., Interaction of electro-ocular potentials with the contingent negative variation, *Psychophysiology*, 7:103-111, 1970.

Weerts, T.C. and Lang, P.J., The effects of eye fixation and stimulus and response location on the contingent negative variation (CNV), **Biol. Psych.**, 1:1-19, 1973.

Whitton, J.L., Lue, F. and Moldofsky, H., A spectral method for removing eye movement artifacts from the EEG, **Electroenceph. Clin. Neurophysiol.**, 44:735-741, 1978.

Woestenburg, J.C., Verbaten, M.N., Slangen, J.L., The removal of the eye-movement artifact from the EEG by regression analysis in the frequency domain, **Biol. Psych.**, 16:127-147. 1983.

APPENDIX

The following appendix is the operations manual for the EEG Artifact Rejection System (EARS) device.

GMS Engineering Corporation

P.I.: G.M. Samaras

LIST OF PERSONNEL RECEIVING PAY ON CONTRACT #DAMD17-89-C-9045

\

**GEORGE M. SAMARAS, PhD, P.I.
OTIS R. BLAUMANIS, PhD
STEVEN M. FALK, MSE
JACK C. CRYSTAL, BSEE
JEFFREY C. SIGL, PhD
LINDA C. MAAS, BSME**

**OPERATING & MAINTENANCE
INSTRUCTIONS
for**

**PROTOTYPE
ELECTROENCEPHALOGRAPH
ARTIFACT REJECTION
SYSTEM (EARS)**

Prepared for:

**Department of the Army
U.S. Army Medical Research Acquisition Activity
Fort Detrick, Frederick, Maryland 21701**

Prepared by:

**GMS Engineering Corporation
Columbia, Maryland 21045**

Contract No.: DAMD17-89-C-9045

March 1990

CONTENTS

I.	PREFACE	5
II.	DESCRIPTION	6
III.	INSTALLATION	8
IV.	OPERATION	11
V.	STORAGE	15
VI.	THEORY OF OPERATION	16
VII.	TROUBLESHOOTING GUIDE	23
VIII.	CALIBRATION PROCEDURES	56
IX.	INDEX	57
X.	APPENDICES	58

LIST OF FIGURES

FIGURE I.1: ELECTROENCEPHALOGRAM ARTIFACT REJECTION SYSTEM	5
FIGURE II.1: ELECTRODE INPUT SELECTOR UNIT (RELAY BOX)	7
FIGURE III.1: POSITIONING OF THREE 9V BATTERIES	9
FIGURE III.2: BATTERY CONNECTORS FOR UNINTERRUPTED OPERATION	10
FIGURE IV.1: CHANNEL NUMBERING SYSTEM	12
FIGURE VI.1: TWO POINT SOURCES SEPARATED BY DISTANCE L.	17
FIGURE VI.2: COORDINATE TRANSFORMATION APPLIED TO POINT Q.	18
FIGURE VI.3: OBSERVED AND CORRECTED EEG.	22
FIGURE VII.B.1: HARDWARE BLOCK DIAGRAM	24
FIGURE VII.B.2: ISOSWITCH BOARD SCHEMATIC	26
FIGURE VII.B.3: ISOSWITCH BOARD	28
FIGURE VII.B.4: INSTRUMENTATION AMP & HORIZONTAL STIMULUS SCHEMATIC	30
FIGURE VII.B.5: INSTRUMENTATION AMP & HORIZONTAL STIMULUS BOARD	32
FIGURE VII.B.6: ANALOG AMPLIFIERS SCHEMATIC	34
FIGURE VII.B.7: AUTOCALIBRATION & VERTICAL AUTOSTIMULATION	36
FIGURE VII.B.8: ANALOG AMPLIFIER/AUTOCAL/AUTOSTIM BOARD SILKSCREEN	38
FIGURE VII.B.9: DIGITAL CIRCUIT SCHEMATIC	40
FIGURE VII.B.10: ANALOG INPUT SIGNAL LIMITER/BUFFER SCHEMATIC	42
FIGURE VII.B.11: DIGITAL BOARD SILKSCREEN	44
FIGURE VII.B.12: RELAY BOX AUXILIARY CIRCUIT SCHEMATICS	46
FIGURE VII.B.13: RELAY BOARDS	48
FIGURE VII.B.14: RELAY BOX CONNECTOR SCHEMATIC	50
FIGURE VII.B.15: 37 PIN "D" CONNECTOR PINOUT DESIGNATION	52
FIGURE VII.C.1: SOFTWARE BLOCK DIAGRAM.	55

I. PREFACE

This manual contains information needed for the operation and maintenance of an experimental prototype device, which provides nearly real-time correction of artifacts on the steady state electroencephalogram (EEG). The prototype EEG Artifact Rejection System (EARS) is a battery-operated, portable device that is designed to operate in a variety of experimental operational settings (laboratory, simulators, and aircraft) and is intended to fit in the calf pocket of a flight suit. While the experimental device has been tested in a laboratory setting, it has **NOT** been evaluated in a simulator or on an aircraft. Furthermore, while it has been subjected to limited human testing, it is not an approved clinical device - it is an experimental prototype. The EARS device should only be used on humans under the auspices of an experimental protocol approved by a duly constituted Internal Review Board.

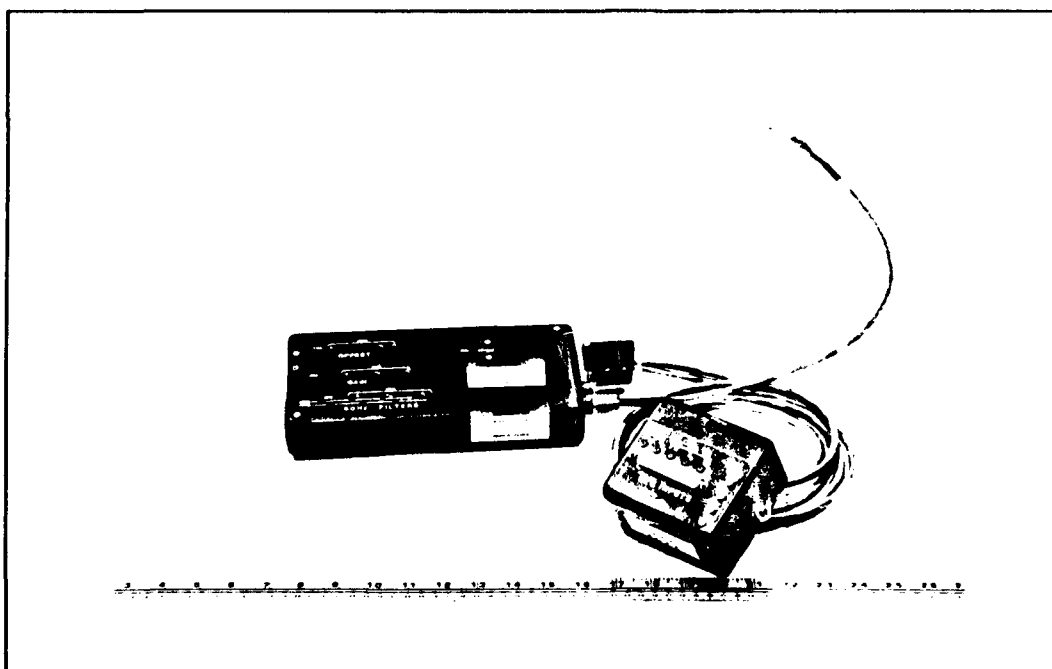


FIGURE I.1: ELECTROENCEPHALOGRAPH ARTIFACT REJECTION SYSTEM

II. DESCRIPTION

The EARS device is a portable unit consisting of a processor unit (which fits in a flight suit calf pocket) and an electrode input selector unit (a "relay box", which is intended to be slung about a subject's neck, as shown in Figure II.1).

The relay box accepts a standard multipin EEG electrode input connector as well as nine (9) additional electrode connections (one pair of ear electrodes, one pair vertical stimulation electrodes, one pair horizontal stimulation electrodes, one pair vertical EOG electrodes, and one single horizontal EOG electrode). These nine electrodes and any one of the 24 EEG electrodes (see Figure IV.1) are transmitted to the processing unit via a cable using standard 25 pin D connectors.

The processing unit communicates with the relay box via its 25 pin D connector, with a host PC via its 9 pin D connector, and with an analog output signal recorder via its 9 pin round connector. The operating mode of the processing unit can be selected from a menu, displayed when the device is connected to a host computer. The processing unit operates in one of three major modes: real-time mode, interrogation mode, and correction mode. In the "real-time" mode, the unit acts as a conventional biopotential amplifier system. In the "interrogation" mode, the unit acquires EOG data and EEG data (only from the one selected channel) for use by the host PC to compute model parameters for the "correction" mode. In the "correction" mode, the unit acquires EEG and EOG data, computes the EOG contribution to the EEG data (using the model and model parameters), subtracts the EOG contribution from the EEG signal, and outputs the corrected signals with a few seconds delay.



FIGURE II.1: ELECTRODE INPUT SELECTOR UNIT (RELAY BOX)

III.INSTALLATION

The hardware consists of four components. There is the EARS main processor unit, an electrode input selector unit, a interconnection cable that connects these two units, and an analog output cable. A separate RS-232 9-pin "D" communications cable must be provided for the communication with the host PC. The "relay box" has two screws on the top, that when unscrewed, allow the lid to open and the batteries to be replaced as shown in Figure III.1. Figure III.2 shows the parallel battery terminal connectors that permits replacing the batteries without interrupting operation of the unit.

Software for the host PC is contained on a 3.5" disk. One can install this software by copying the disk onto a hard disk. The PC must be an IBM PC/AT/XT with a numeric coprocessor. This is done by typing "copy a:*.*)" when in the desired directory on the hard disk. The user is now ready to operate the EARS system.

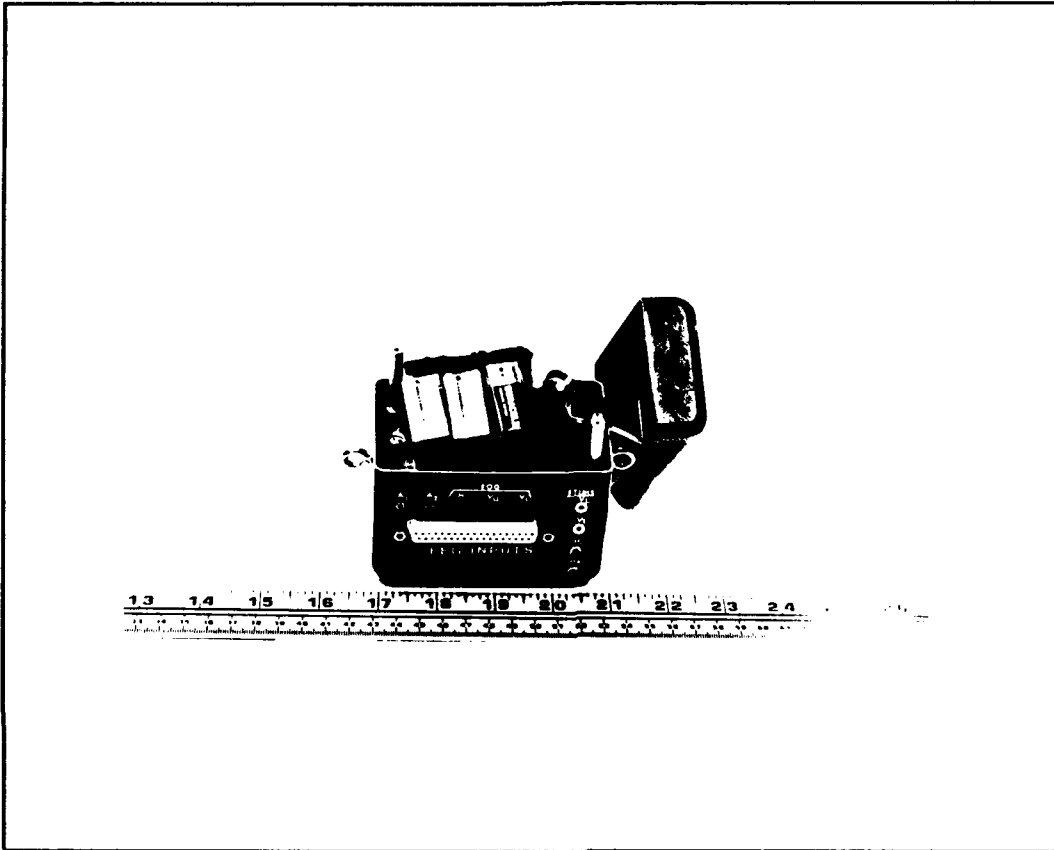


FIGURE III.1: POSITIONING OF THREE 9V BATTERIES

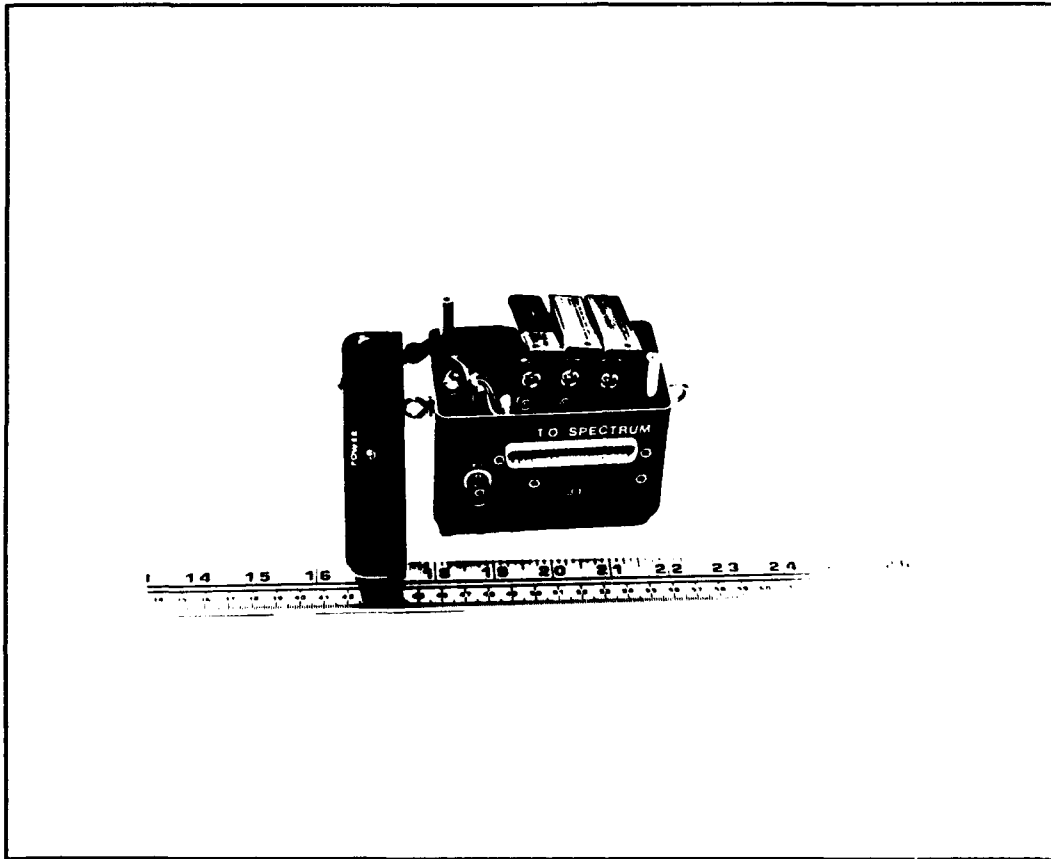


FIGURE III.2: BATTERY CONNECTORS FOR UNINTERRUPTED OPERATION

IV. OPERATION

The operation of the EEG Artifact Rejection System is relatively simple, yet there is a specific protocol that must be followed to ensure proper function.

Once the EEG and reference electrodes (Figure VII.B.15) and the nine additional electrodes (shown in Figure II.1) are attached to the subject, the unit can be powered up. This is achieved by connecting the "relay box" to the main processor unit using the cable provided. When this is done, the power indicator on the "relay box" will illuminate. A flashing light indicates a low battery. The RS-232 cable must now be connected from the processor unit to a IBM PC/XT/AT personal computer with a numeric coprocessor. A numeric coprocessor is required for the Fortran software to operate.

Type "EARS" and then a carriage return to enter the first of the two programs (see Section VII.C for software description). When the screen goes completely blank (approximately five seconds later), push the carriage return once again. The EARS menu will appear on the screen.

GMS Engineering Corporation
EEG Artifact Rejection System

N - Channel Number Selection
L - LED Light Level
R - Real Time Monitoring
P - Calibration Pulses
I - Interrogation
C - Correction

Enter RESPONSE >

One can select the channel that the processor unit will correct by typing "N". The system will prompt the user for the desired channel number (1-24). If a carriage return is pushed without entering a number, the current channel number is selected. The default is Channel

#1: Figure IV.1 delineates the correspondence between the channel numbers (1-24), the 37-pin "D" connector pins and the normal EEG derivations connected to those pins.

GMS EEG CHANNEL #	37 PIN D CONNECTOR PIN	NORMAL EEG DERIVATION
1	1	FP1
2	20	FP2
3	2	F3
4	21	F4
5	3	C3
6	22	C4
7	4	P3
8	23	P4
9	5	O1
10	24	O2
11	6	F7
12	25	F8
13	7	T3
14	26	T4
15	8	T5
16	27	T6
17	28	Cz
18	10	Fz
19	29	Pz
20	11	Fpz
21	30	Oz
22	14	C3'
23	33	Cz'
24	15	C4'

FIGURE IV.1: CHANNEL NUMBERING SYSTEM

One can change the power indicator light level that appears on the "relay box" by typing "L". This will permit low level light operations. The system will prompt the user for the desired light level (1-255). The smaller the number the less intense is the light. If a carriage return is pushed without entering a number, the current light level is selected. The default is 128.

The option "R" will allow the user to monitor the three EOG channels and the selected EEG channel from the outputs on the EARS unit. To exit this routine, just push any key on the PC keyboard. This will bring the user back to the main menu.

The option "P" will allow the user to monitor the three EOG channels and the selected EEG channel from the outputs on the processor unit. A train of calibration pulses will ride on the outputs for approximately five minutes. This is caused by application of a single calibration pulse applied to all the input channels. The amplitude of this calibration pulse is 1 mV. Each pulse is fifty milliseconds in duration, and there is approximately one second between pulses. This aids the user in adjusting the desired gain for each channel. To exit this routine, just push any key on the PC keyboard. This will bring the user back to the main menu.

The option "I" is used for interrogating the medium (subject). This routine requires approximately two minutes. The direct drive signals will be output on the interrogation quadropole (the four electrodes on the forehead). After this interrogation process is completed, the host PC screen will prompt the user to store the appropriate data for processing. The prompts provided on the PC screen are: push 'PgDn', then type "7", and then type "drive" and carriage return. Then push the uppercase "A". The data will stream across the screen and into a file on the disk.

When the screen prompts the user to exit EARS, push 'ALT-X' and then "Y". The user will now be in DOS. The second program should be run by typing "EEG" and a carriage return. This program will prompt the user to enter the geometrical distances (in mm) from the eyes to the EOG and selected EEG electrodes, as well as the distance between the eyes and the distance from the center of the eyes to the quadropole. These should be carefully measured using a soft cloth tape measure. When these parameters are entered, the program then calculates and fine tunes the model coefficients and the medium transfer function.

When this program is finished, the screen will prompt the user to run the EARS program once again. One does this by following the same instructions as above. When the main menu appears, choose the "C" option to begin correcting the selected EEG channel. The user will be instructed to push 'PgUp', "7", and type "correct" and a carriage return. The appropriate

model parameter data will be transferred to the main EARS processor unit, and the EEG correction will begin.

The RS-232 cable can now be disconnected. The analog outputs are as follows.

Output #1	-	Real Time EEG
Output #2	-	Real Time Event Trigger
Output #3	-	Delayed, Corrected EEG
Output #4	-	Delayed Event Trigger
Output #5	-	Delayed, Uncorrected EEG
Output #6	-	Delayed Horizontal EOG
Output #7	-	Delayed Vertical Upper EOG
Output #8	-	Delayed Vertical Lower EOG

The blinking of the power light on the "relay box" means that the batteries are getting low, and must be changed within the hour. The batteries can be changed WITHOUT interrupting operation by putting three new batteries on the reverse side of the battery clip, and then taking out the three old batteries. The power light should then be continuously on.

V. STORAGE

Turn off the battery power to the device by disconnecting the cable between the processor unit and the relay box. The LED indicator will extinguish.

Disconnect all electrodes and cables from the device.

Wipe off any debris from the external surfaces of the EARS unit before storage. A soft cloth dampened with water or a mild soap and water solution can be used. Do not apply organic solvents to this prototype unit.

To conserve battery life, remove the three 9V batteries from the unit. Do not leave the batteries in the unit, if long term storage is intended.

Return the unit to its original transport container or another equivalent storage/protection container.

VI. THEORY OF OPERATION

The EARS device is based on the idea that eye movements and blinks contribute to the observed EEG signals. If the electrical signal characteristics of these eye movements and blinks are known and the medium through which these signals propagate to the EEG observation sites (EEG electrode sites) is characterized, then this unwanted influence can be mathematically removed. The removal of this influence is the correction process. The mathematical model describing this correction process which is implemented in the software in the EARS device is described in this section.

The Biophysical Model. The transection of the face across the three dimensional ocular dipole field (caused by the corneo-retinal potential in the eye) yields a surface image dipole propagating on the scalp. This surface image dipole can be modelled to incorporate both the eye movement and the eye blink. The eye movement produces a symmetric dipole, while the blink produces an asymmetric dipole.

General Dipole Representation. A dipole source, symmetric or asymmetric, is the superposition of two point sources separated by a distance. The point source's electric field propagates as a function of $1/r^2$. The voltage at any point is described by $V=kq/r$, where k is Boltzmann's constant, and q is the amount of charge. The surface image dipole is described here.

Figure VI.1 shows two point sources separated by a distance (L). The voltage (V) appearing at point Q is derived as follows.

$$V = kq\left\{\frac{m}{(r-\frac{1}{2}L\sin\theta)} + \frac{-n}{(r+\frac{1}{2}L\sin\theta)}\right\}. \quad (1)$$

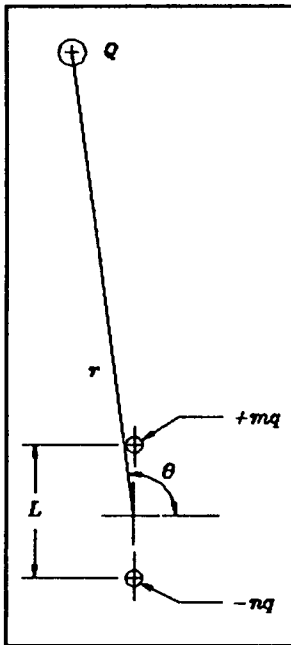


FIGURE VI.1: TWO POINT SOURCES SEPARATED BY DISTANCE L.

Rearranging Equation 1 yields

$$V = kq\{[(m-n)r + (m+n)\frac{1}{2}L\sin\theta]/(r^2 - \frac{1}{4}L^2\sin^2\theta)\}. \quad (2)$$

Since $r \gg L$, we can simplify Equation 2:

$$V = n[(kq/r^2)\frac{1}{2}L\sin\theta(\alpha+1) + (kq/r)(\alpha-1)], \quad (3)$$

where $\alpha = m/n$.

As a note, if $\alpha=1$ (eye movement) and the dipole is symmetric, Equation 3 reduces to,

$$V = (nkq/r^2)L\sin\theta = Ar^2\sin\theta. \quad (4)$$

where $A = nrqL$.

Furthermore, it is important to note that the zero-potential line of the dipole is the x-axis when $\alpha=1$ ($\sin\theta=0$). When $\alpha \neq 1$, the zero-potential line becomes a circle described by,

$$x^2 + (y+G)^2 = G^2, \quad (5)$$

where $G = \frac{1}{2}L[(\alpha+1)/(\alpha-1)]$.

Selection of a Facial Coordinate System. In order to spatially represent the ocular dipoles or the stimulus dipole in planar geometry, we must select a coordinate system. This is shown in Figure VI.2. The point, Q, in this figure represents an electrode. The subscript L is used to show reference to the left eye, the subscript R is used to reference the right eye. The electrode is a distance β_L from the left eye and β_R from the right eye. Reference to the stimulation dipole is indicated by S. Our facial coordinate system has its origin at the geometric center of the two eyes (which are separated by a distance z). The stimulus dipole is located a distance h above the origin. The electrode is a distance r from the origin of the coordinate system. The stimulation dipole lies on the y-axis as the eyes lie on the x-axis.

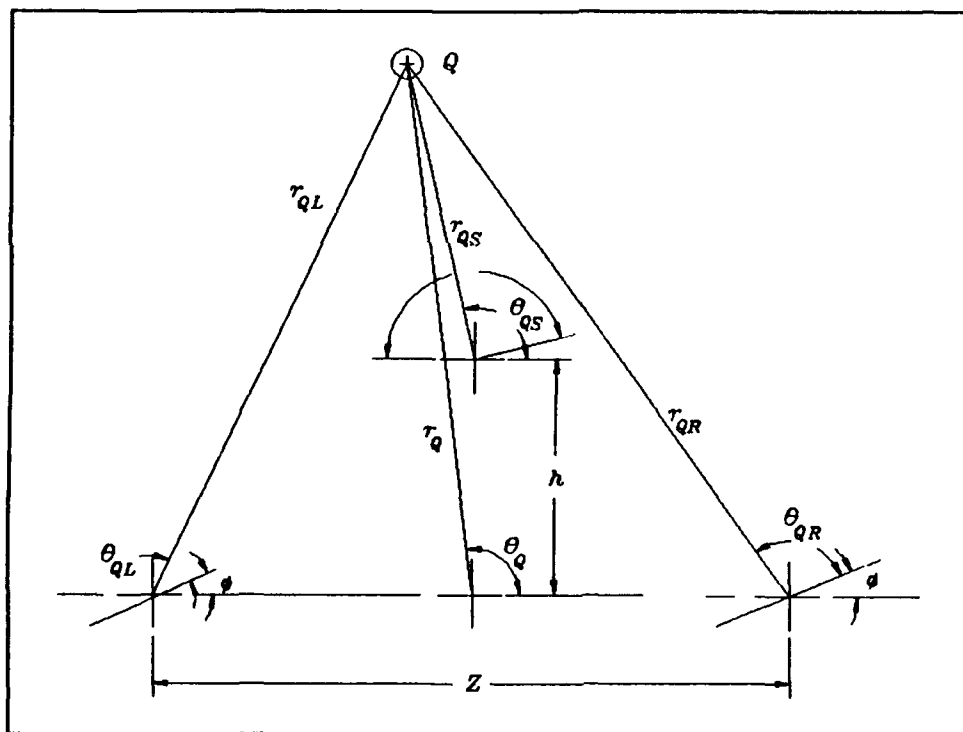


FIGURE VI.2: COORDINATE TRANSFORMATION APPLIED TO POINT Q.

Spatial Resolution of the Bi-ocular Dipoles. Using our coordinate system and dipole representation described above, we will model the two ocular dipoles as one complex mathematical function based at the origin of the coordinate system.

From the law of cosines:

$$\beta_L = [r^2 + \frac{1}{4}Z^2 + rZ\cos\theta]^{\frac{1}{2}} \quad (6)$$

$$\beta_R = [r^2 + \frac{1}{4}Z^2 - rZ\cos\theta]^{\frac{1}{2}} \quad (7)$$

From the law of sines:

$$\gamma_L = -\phi + \sin^{-1}[(r/\beta_L)\sin\theta] \quad (8)$$

$$\gamma_R = \pi - \phi - \sin^{-1}[(r/\beta_R)\sin\theta] \quad (9)$$

where: ϕ = angle of the ocular dipoles (zero-potential line).

We now substitute these identities (Eqns. 6-9) into the dipole equation defined earlier (Eqn. 3) and sum the two ocular dipoles to result in one function. We obtain this, in a general form, for any electrode:

$$V = \frac{1}{2}(\alpha+1)A\{[\beta_L^{-2}[-(1-\gamma_L^2)^{1/2}]\sin\phi+\gamma_L\cos\phi\}+[\beta_R^{-2}[(1-\gamma_R^2)^{1/2}]\sin\phi+\gamma_R\cos\phi\}+(A/L)(\alpha-1)\{\beta_L^{-1}+\beta_R^{-1}\}, \quad (10)$$

where:

$$\beta_L = [(r)^2+(\frac{1}{4}Z)^2+rZ\cos\theta]^{1/2}$$

$$\beta_R = [(r)^2+(\frac{1}{4}Z)^2-rZ\cos\theta]^{1/2}$$

$$\gamma_L = (r/\beta_L)\sin\theta$$

$$\gamma_R = (r/\beta_R)\sin\theta$$

Spatial Representation of the Stimulus Dipoles. We can similarly describe the representation of the stimulus dipole in our new coordinate system as we have described the ocular dipoles above. Using Figure VI.2 and Equation 4:

$$V = B\{\sigma^2[\eta\sin\psi - (1-\eta^2)^{1/2}\cos\psi]\} \quad (11)$$

where:

$$\sigma = [r^2 + h^2 - 2rh\sin\theta]^{1/2}$$

$$\eta = (-r/\sigma)\cos\theta$$

$$\psi = \text{angle of the stimulus dipole (an analog of } \phi)$$

Equations 10 & 11 and the associated identities are the basis of the mathematical model which will be used in the calculation of the transfer function.

Mathematical Relationship Between Resultant Ocular Dipoles & Stimulus Dipoles. We can now use our basic equations and our coordinate system to correct the putative transfer function measured by surface dipoles. The EOG artifact correction equation in the frequency domain is:

$${}^{\text{corr}}V_{\text{EEG}} = {}^{\text{obs}}V_{\text{EEG}} - {}^{\text{obs}}V_{\text{EOG}} [S_{\text{EEG}}/S_{\text{EOG}}] D \quad (12)$$

where D is the geometrical correction factor between the stimulus dipole and the ocular dipoles. S denotes the surface dipole stimulation response and ${}^{\text{obs}}V$ denotes the naturally occurring response. The subscript "EEG" and "EOG" refer to the electrode recording the response; the superscripts "corr" and "obs" refer to the corrected and observed potential, respectively.

We can obtain D by manipulation of the equations described above.

$$D = \frac{\frac{\{\frac{1}{2}(\alpha+1)[\beta_L^{-2}-(1-\gamma_L^2)^{1/2}]\sin\phi+\gamma_L\cos\phi\}+[\beta_R^{-2}\{(1-\gamma_R^2)^{1/2}\}\sin\phi+\gamma_R\cos\phi]\{+(\alpha-1)/L\}[\beta_L^{-1}+\beta_R^{-1}]\}_{EEG}}{\frac{\{\frac{1}{2}(\alpha+1)[\beta_L^{-2}-(1-\gamma_L^2)^{1/2}]\sin\phi+\gamma_L\cos\phi\}+[\beta_R^{-2}\{(1-\gamma_R^2)^{1/2}\}\sin\phi+\gamma_R\cos\phi]\{+(\alpha-1)/L\}[\beta_L^{-1}+\beta_R^{-1}]\}_{EOG}}{\frac{\sigma^2[\eta\sin\psi-(1-\eta^2)^{1/2}\cos\psi]_{EEG}}{\sigma^2[\eta\sin\psi-(1-\eta^2)^{1/2}\cos\psi]_{EOG}}} \quad (13)$$

where,

$$\beta_L = [(r)^2 + (\frac{1}{2}Z)^2 + rZ\cos\theta]^{1/2},$$

$$\beta_R = [(r)^2 + (\frac{1}{2}Z)^2 - rZ\cos\theta]^{1/2},$$

$$\gamma_L = (r/\beta_L)\sin\theta,$$

$$\gamma_R = (r/\beta_R)\sin\theta,$$

$$\sigma = [r^2 + h^2 - 2rh\sin\theta]^{1/2},$$

$$\eta = (-r/\sigma)\cos\theta,$$

$$\phi = \begin{cases} 0^\circ & \text{for vertical EOG} \\ 90^\circ & \text{for horizontal EOG} \end{cases}$$

$$\psi = \begin{cases} 0^\circ & \text{for vertical interrogation pulse} \\ 90^\circ & \text{for horizontal interrogation pulse} \end{cases}$$

$$\alpha = m/n = (\text{obtained in real time as } EOG_{\text{vertical}}^{\text{upper}} / EOG_{\text{vertical}}^{\text{lower}}).$$

We measure z and h, as well as β_L and β_R for each electrode (EEG and EOG). We then calculate r and θ for each electrode, and then calculate γ_L , γ_R , σ , and η for each electrode. Finally, we calculate D's for each EEG/EOG electrode combination. This permits correction of the observed EEG in accordance with Equation 12. Figure VI.3 shows an example of a correction. The cross-correlation between the observed EEG and the vertical EOG was 0.75; the cross-correlation between the corrected EEG and the vertical EOG was 0.017.

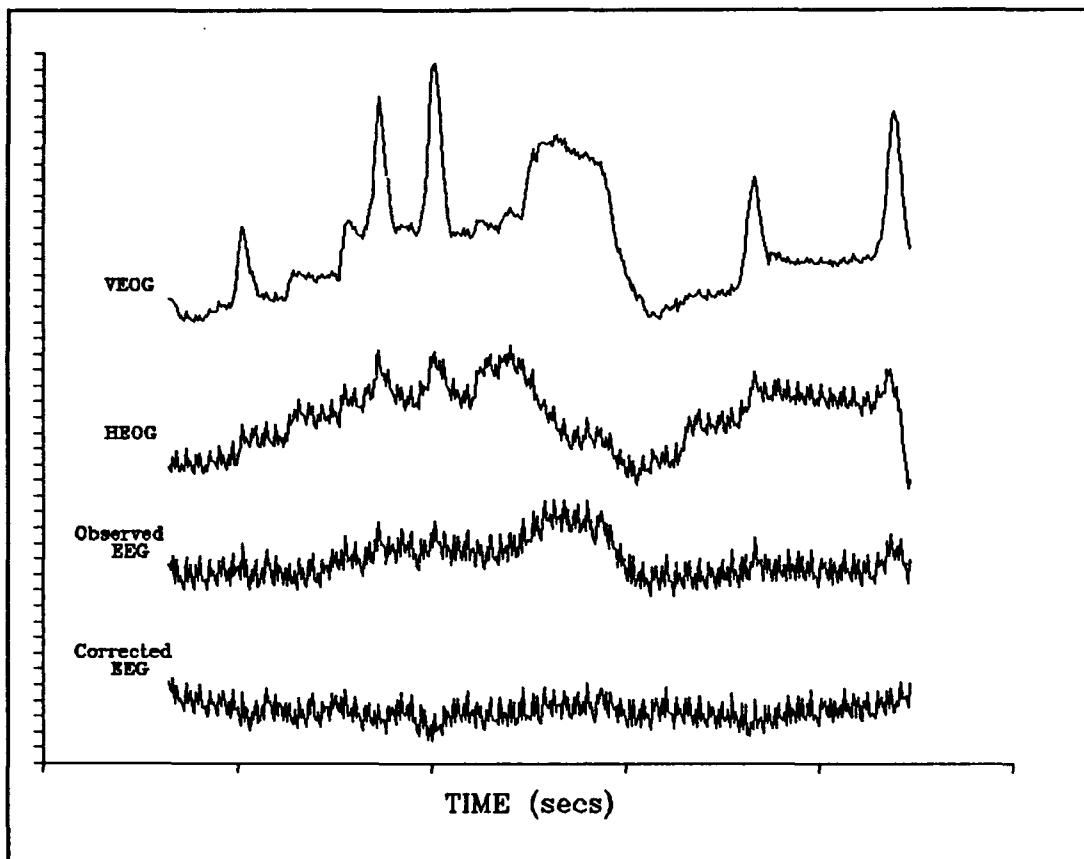


FIGURE VI.3: OBSERVED AND CORRECTED EEG.

VII. TROUBLESHOOTING GUIDE

A. GENERAL

The general troubleshooting protocol of the EARS system is extremely simple. If the power light is flashing, the batteries need to be changed. If the output levels become "flat" (no signal), the batteries need to be changed. If the batteries are new, then the system must be serviced by authorized personnel.

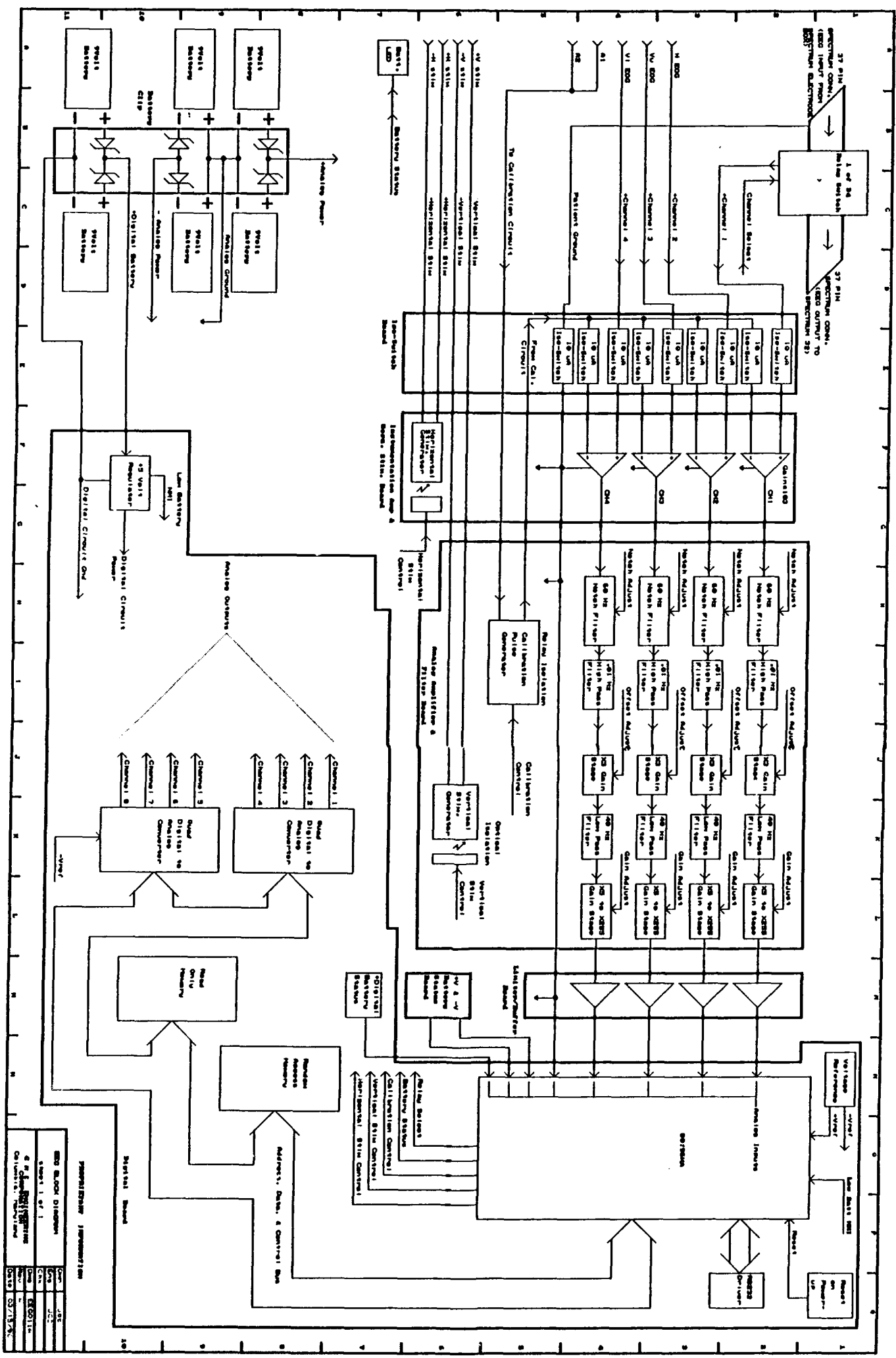
B. HARDWARE DESCRIPTION

The hardware schematics are shown in Figures VII.B.2 through VII.B.15. Figure VII.B.1 is a block diagram of the complete system. Each analog channel contains four user-adjustable potentiometers. One controls the gain of the channel; and one controls the offset of the channel. The other two are for fine tuning the 60 Hz notch filter. These are factory calibrated, and should hold their calibration for several months or longer.

The calibration circuit is a floating voltage source that is switched into series with the inverting input of the instrumentation amplifier. This level can be changed by the user with a potentiometer. See Section VIII for further details.

The digital circuitry consists of a microcontroller and memory. There are digital-to-analog output converters/amplifiers which allow the user to monitor the EEG and EOG channels.

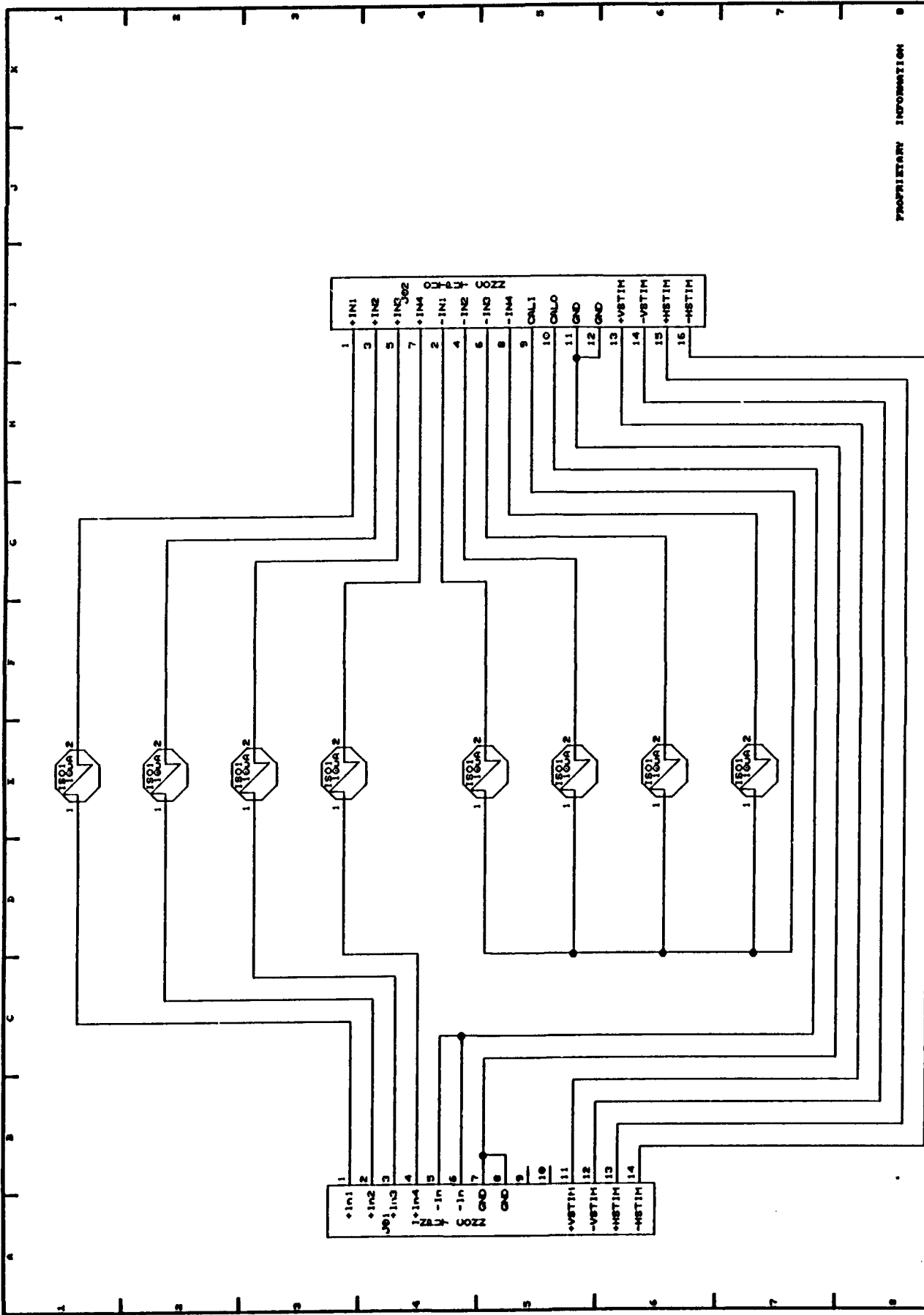
FIGURE VII.B.1: HARDWARE BLOCK DIAGRAM



POWER SUPPLY INFORMATION

Item	Value
REG. BLOCK DIAGRAM	1-25
Sheet 1 of 1	1-25
DATE	6/20/71
DESIGNED BY	W. J. B. / J. B. B.
CHECKED BY	W. J. B. / J. B. B.
DATE	6/20/71
REVISION	02/15/76

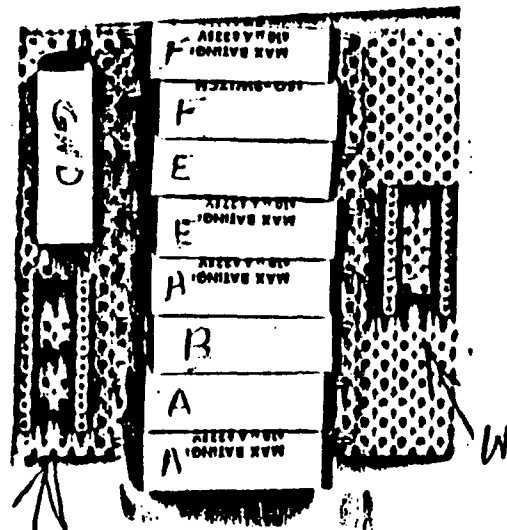
FIGURE VII.B.2: ISOSWITCH BOARD SCHEMATIC



PROPRIETARY INFORMATION

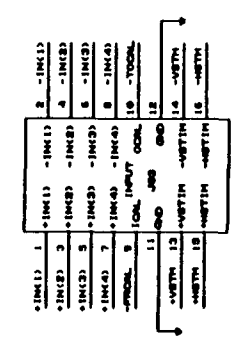
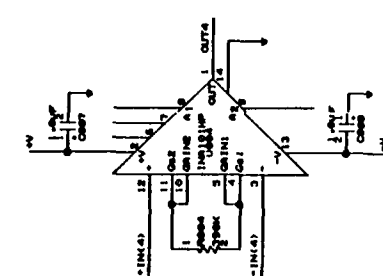
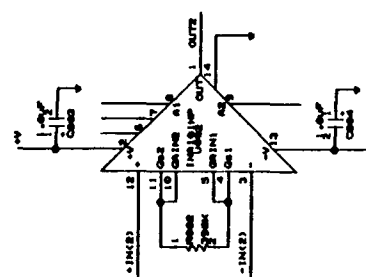
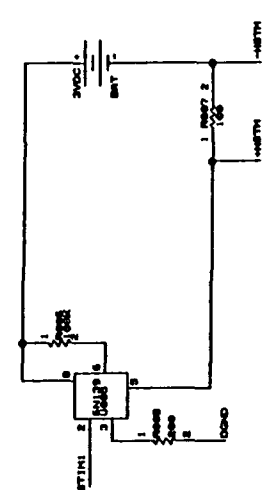
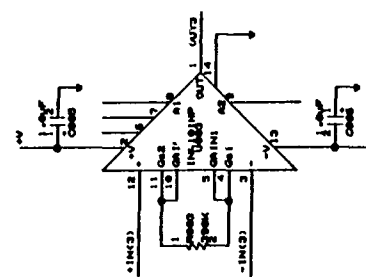
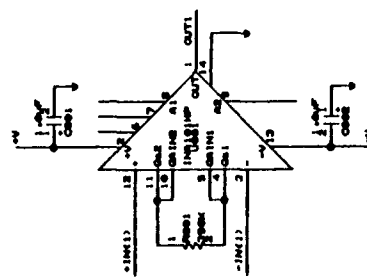
EEG ISO-SWITCH BOARD	Doc	JCC
Sheet 1 of 1	Enr	JCC
C. H. S. ENGINEERING CORPORATION	Chk	EEG0084
Columbia, Maryland	Rev	A
	Date	2/23/90

FIGURE VII.B.3: ISOSWITCH BOARD



OUT
150-SWITCH
BOARD

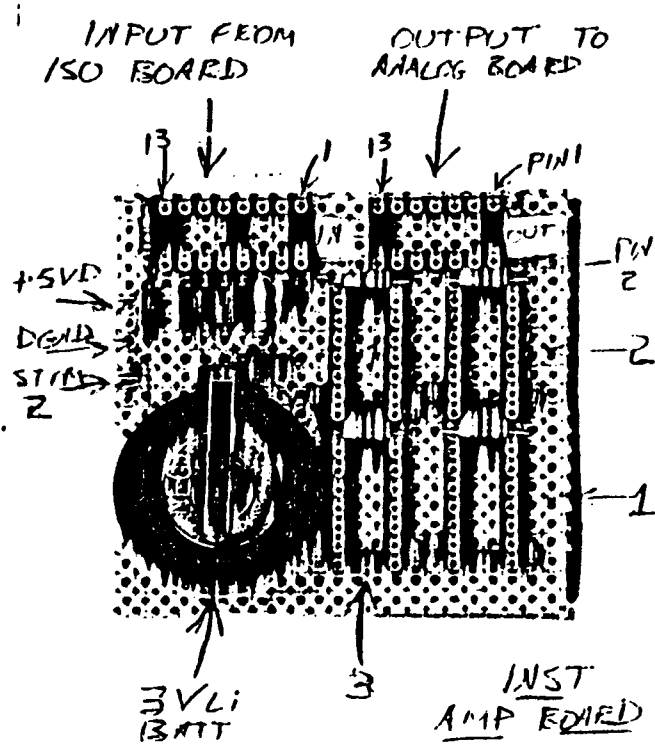
FIGURE VII.B.4: INSTRUMENTATION AMP & HORIZONTAL STIMULUS SCHEMATIC



PROPOSED STAFFING IMPROVEMENTS

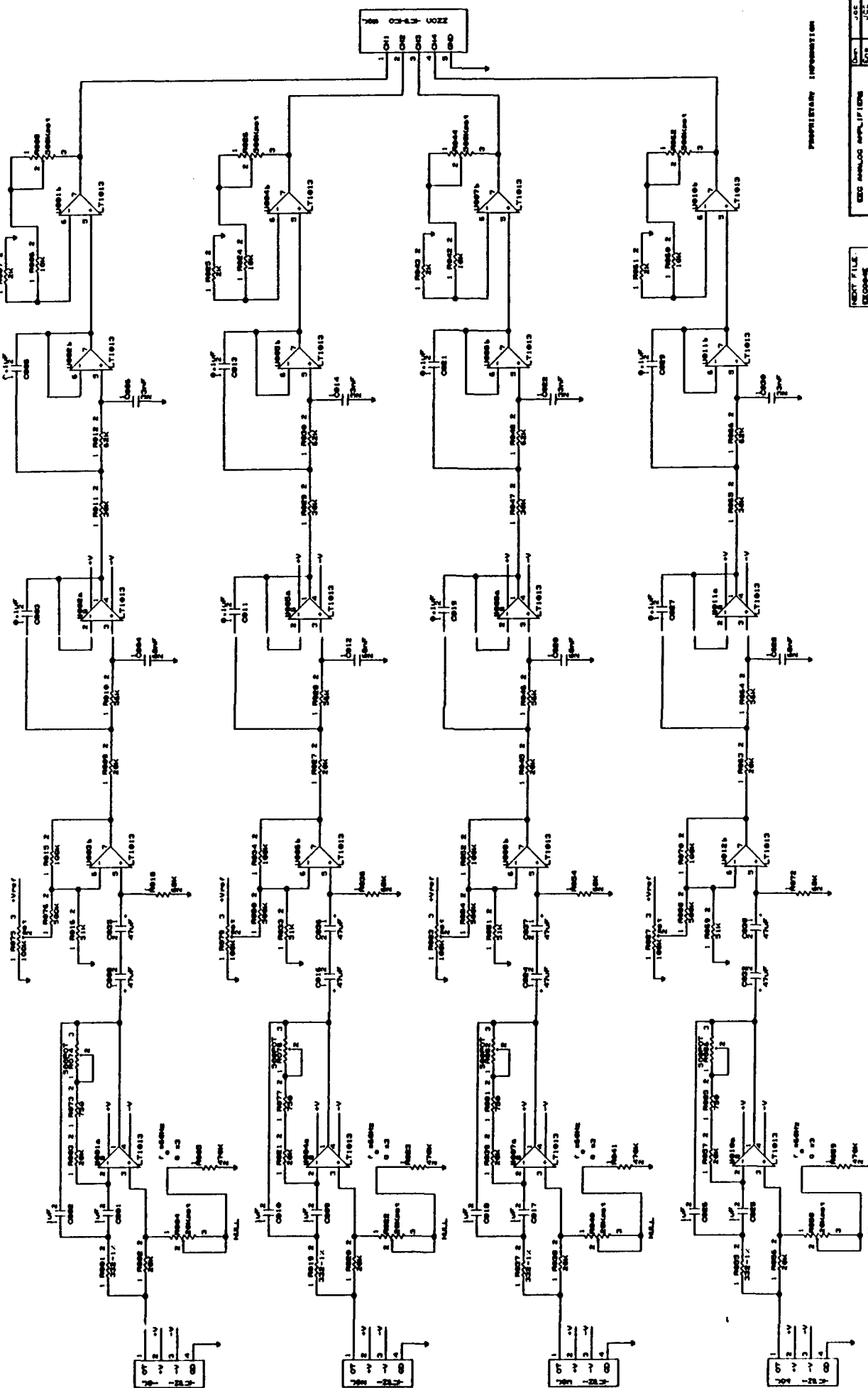
DESIGNER	DATE	REV
10/10/2010	10/10/2010	1
10/10/2010	10/10/2010	2
10/10/2010	10/10/2010	3
10/10/2010	10/10/2010	4
10/10/2010	10/10/2010	5
10/10/2010	10/10/2010	6
10/10/2010	10/10/2010	7
10/10/2010	10/10/2010	8
10/10/2010	10/10/2010	9
10/10/2010	10/10/2010	10
10/10/2010	10/10/2010	11
10/10/2010	10/10/2010	12
10/10/2010	10/10/2010	13
10/10/2010	10/10/2010	14
10/10/2010	10/10/2010	15
10/10/2010	10/10/2010	16
10/10/2010	10/10/2010	17
10/10/2010	10/10/2010	18
10/10/2010	10/10/2010	19
10/10/2010	10/10/2010	20
10/10/2010	10/10/2010	21
10/10/2010	10/10/2010	22
10/10/2010	10/10/2010	23
10/10/2010	10/10/2010	24
10/10/2010	10/10/2010	25
10/10/2010	10/10/2010	26
10/10/2010	10/10/2010	27
10/10/2010	10/10/2010	28
10/10/2010	10/10/2010	29
10/10/2010	10/10/2010	30
10/10/2010	10/10/2010	31
10/10/2010	10/10/2010	32
10/10/2010	10/10/2010	33
10/10/2010	10/10/2010	34
10/10/2010	10/10/2010	35
10/10/2010	10/10/2010	36
10/10/2010	10/10/2010	37
10/10/2010	10/10/2010	38
10/10/2010	10/10/2010	39
10/10/2010	10/10/2010	40
10/10/2010	10/10/2010	41
10/10/2010	10/10/2010	42
10/10/2010	10/10/2010	43
10/10/2010	10/10/2010	44
10/10/2010	10/10/2010	45
10/10/2010	10/10/2010	46
10/10/2010	10/10/2010	47
10/10/2010	10/10/2010	48
10/10/2010	10/10/2010	49
10/10/2010	10/10/2010	50
10/10/2010	10/10/2010	51
10/10/2010	10/10/2010	52
10/10/2010	10/10/2010	53
10/10/2010	10/10/2010	54
10/10/2010	10/10/2010	55
10/10/2010	10/10/2010	56
10/10/2010	10/10/2010	57
10/10/2010	10/10/2010	58
10/10/2010	10/10/2010	59
10/10/2010	10/10/2010	60
10/10/2010	10/10/2010	61
10/10/2010	10/10/2010	62
10/10/2010	10/10/2010	63
10/10/2010	10/10/2010	64
10/10/2010	10/10/2010	65
10/10/2010	10/10/2010	66
10/10/2010	10/10/2010	67
10/10/2010	10/10/2010	68
10/10/2010	10/10/2010	69
10/10/2010	10/10/2010	70
10/10/2010	10/10/2010	71
10/10/2010	10/10/2010	72
10/10/2010	10/10/2010	73
10/10/2010	10/10/2010	74
10/10/2010	10/10/2010	75
10/10/2010	10/10/2010	76
10/10/2010	10/10/2010	77
10/10/2010	10/10/2010	78
10/10/2010	10/10/2010	79
10/10/2010	10/10/2010	80
10/10/2010	10/10/2010	81
10/10/2010	10/10/2010	82
10/10/2010	10/10/2010	83
10/10/2010	10/10/2010	84
10/10/2010	10/10/2010	85
10/10/2010	10/10/2010	86
10/10/2010	10/10/2010	87
10/10/2010	10/10/2010	88
10/10/2010	10/10/2010	89
10/10/2010	10/10/2010	90
10/10/2010	10/10/2010	91
10/10/2010	10/10/2010	92
10/10/2010	10/10/2010	93
10/10/2010	10/10/2010	94
10/10/2010	10/10/2010	95
10/10/2010	10/10/2010	96
10/10/2010	10/10/2010	97
10/10/2010	10/10/2010	98
10/10/2010	10/10/2010	99
10/10/2010	10/10/2010	100

FIGURE VII.B.5: INSTRUMENTATION AMP & HORIZONTAL STIMULUS BOARD



INSTRUMENTATION
AMP BOARD

FIGURE VII.B.6: ANALOG AMPLIFIERS SCHEMATIC

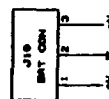
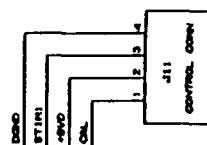
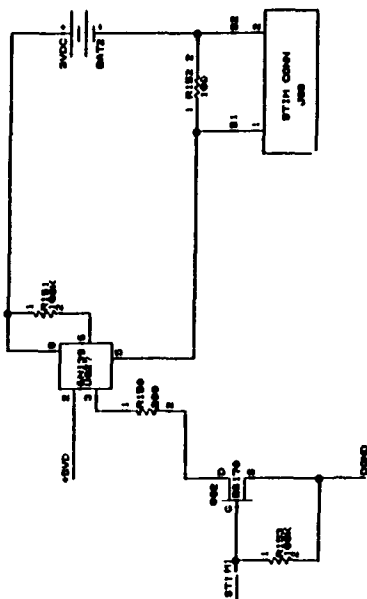


PROFESSIONAL INFORMATION

NEXT FILE -
110000

REC ANALOG AMPLIFIERS	Don	JEC
	Eve	JEC
	Chu	
Sheet 1 of 2	Don	EEG000E
4 M Classification	Rev	E QG12790
Columbia, Maryland	Date	12/20/85

**FIGURE VII.B.7: AUTOCALIBRATION & VERTICAL AUTOSTIMULATION
SCHEMATIC**



IND 1.1 (continued) **ANNUAL REPORT**

BEG AUTOCL. & AUTOSTIM					
sheet 2 of 2					
C H Case/EIS/Williams					
Columbus, Maryland					
Date:	Jan	EIS Good!	JEC		
Sat:	Jan		JEC		
Ch:					
Time:					
Page:					
Case:					

FIGURE VII.B.8: ANALOG AMPLIFIER/AUTOCAL/AUTOSTIM BOARD SILKSCREEN

GMS ENGINEERING CORP.
 ANALOG BOARD
 AUTOMATED DESIGN SERVICES 05-15-89
 SILKSCREEN 1 1 V028

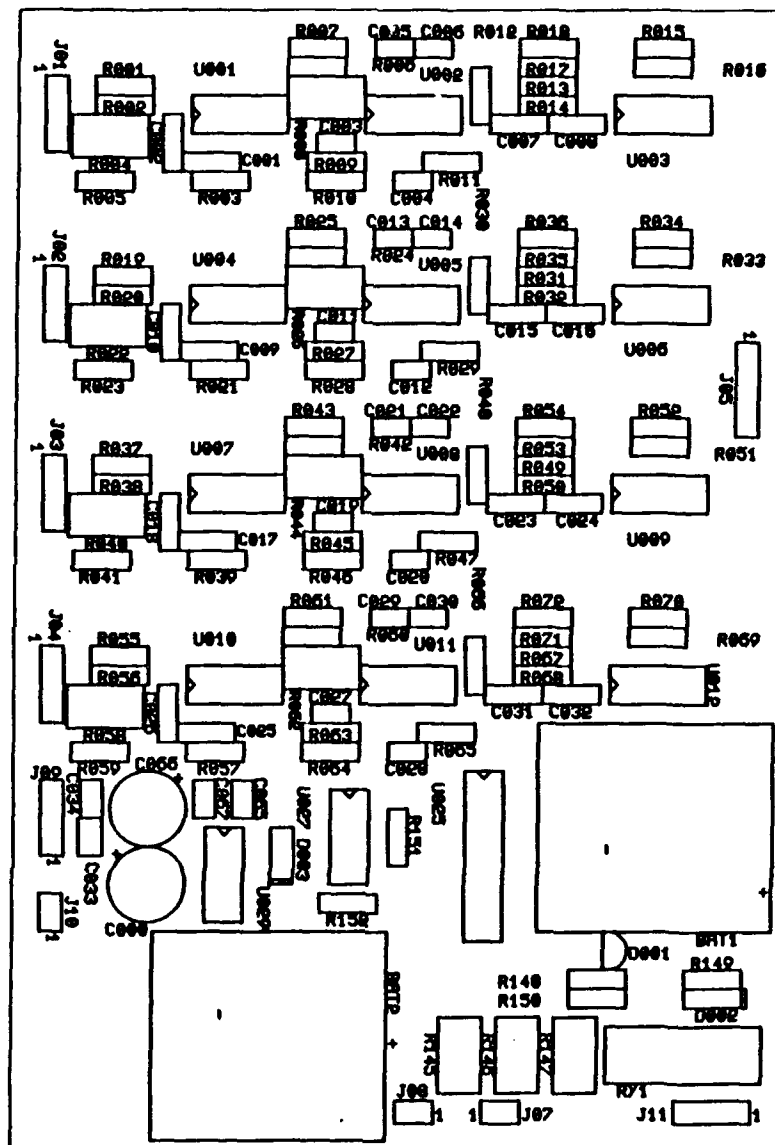
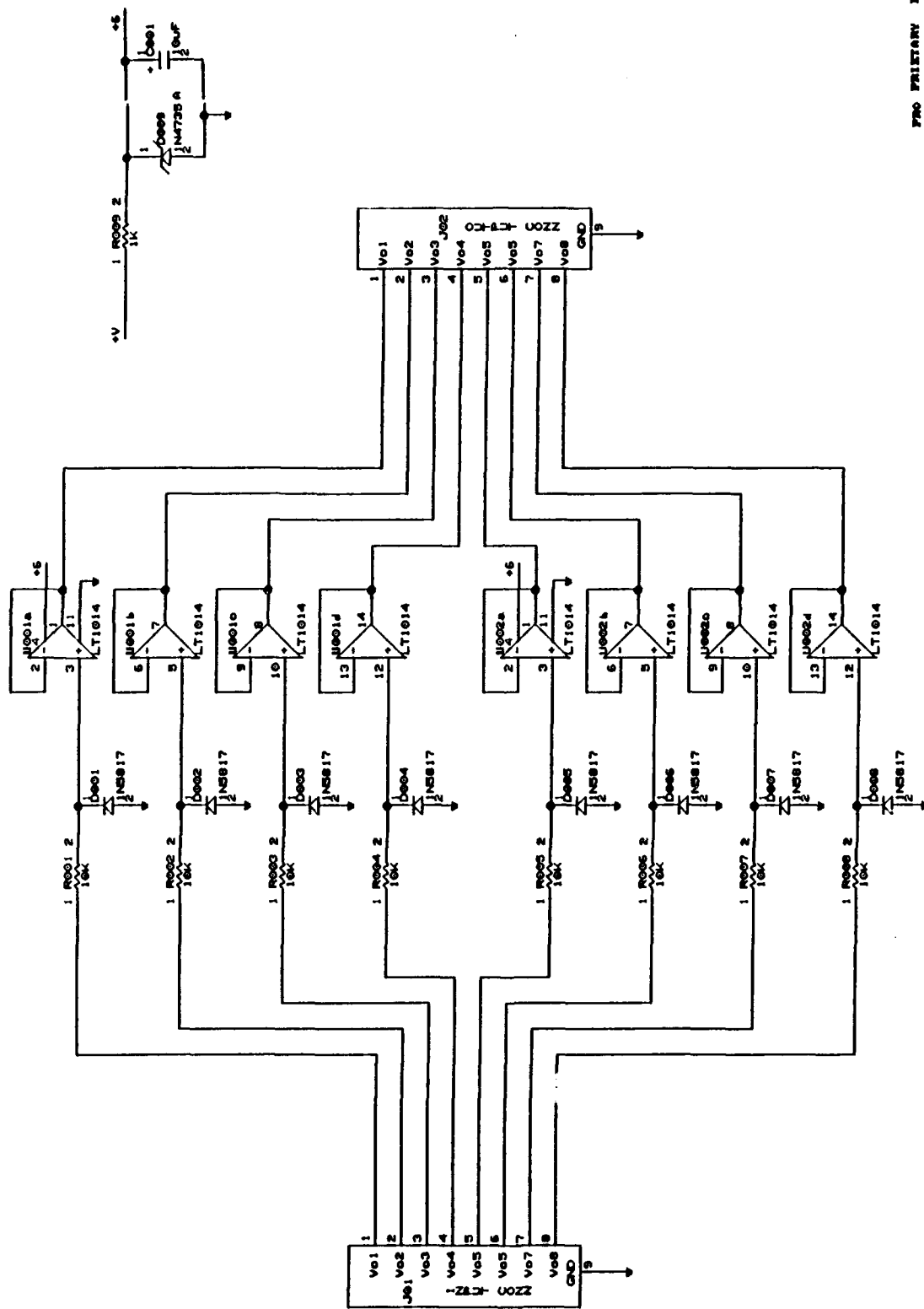


FIGURE VII.B.9: DIGITAL CIRCUIT SCHEMATIC

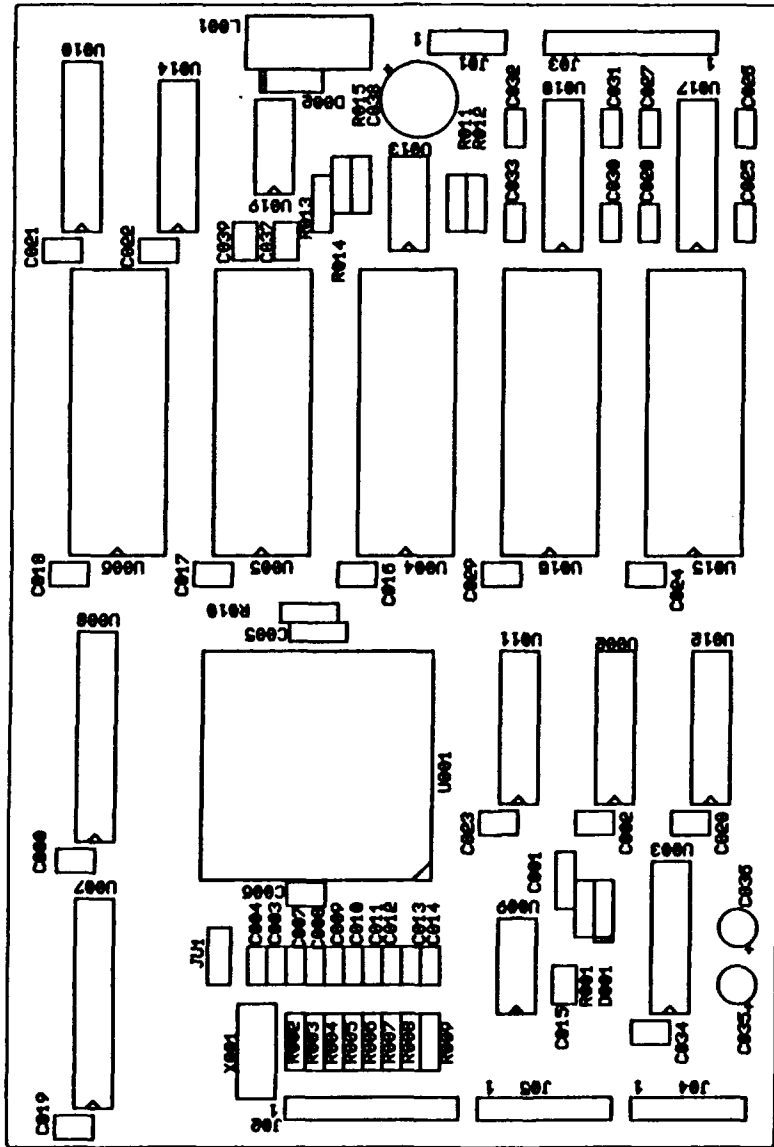
FIGURE VII.B.10: ANALOG INPUT SIGNAL LIMITER/BUFFER SCHEMATIC



PRO PRIMARY INFORMATION

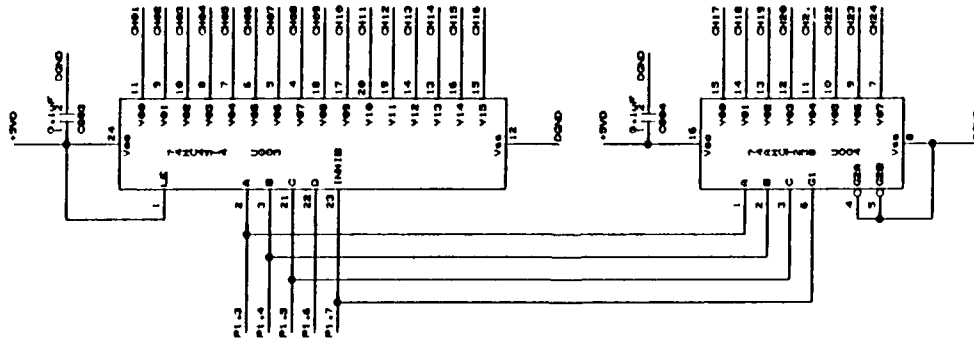
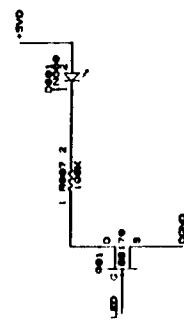
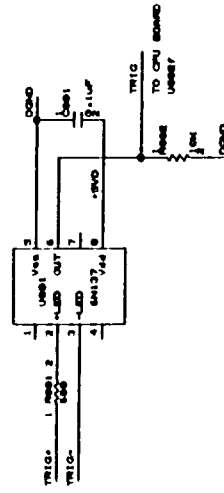
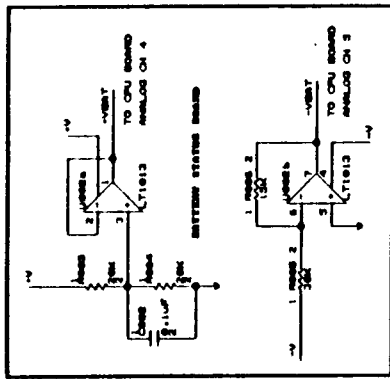
DESIGN	DATE	BY	CHK	APP	REV	DATE
10001A	10/10/85	JCC	JCC	JCC	JCC	JCC
10001B	10/10/85	JCC	JCC	JCC	JCC	JCC
10001C	10/10/85	JCC	JCC	JCC	JCC	JCC
10001D	10/10/85	JCC	JCC	JCC	JCC	JCC
10001E	10/10/85	JCC	JCC	JCC	JCC	JCC
10001F	10/10/85	JCC	JCC	JCC	JCC	JCC
10001G	10/10/85	JCC	JCC	JCC	JCC	JCC
10001H	10/10/85	JCC	JCC	JCC	JCC	JCC
10001I	10/10/85	JCC	JCC	JCC	JCC	JCC
10001J	10/10/85	JCC	JCC	JCC	JCC	JCC
10001K	10/10/85	JCC	JCC	JCC	JCC	JCC
10001L	10/10/85	JCC	JCC	JCC	JCC	JCC
10001M	10/10/85	JCC	JCC	JCC	JCC	JCC
10001N	10/10/85	JCC	JCC	JCC	JCC	JCC
10001O	10/10/85	JCC	JCC	JCC	JCC	JCC
10001P	10/10/85	JCC	JCC	JCC	JCC	JCC
10001Q	10/10/85	JCC	JCC	JCC	JCC	JCC
10001R	10/10/85	JCC	JCC	JCC	JCC	JCC
10001S	10/10/85	JCC	JCC	JCC	JCC	JCC
10001T	10/10/85	JCC	JCC	JCC	JCC	JCC
10001U	10/10/85	JCC	JCC	JCC	JCC	JCC
10001V	10/10/85	JCC	JCC	JCC	JCC	JCC
10001W	10/10/85	JCC	JCC	JCC	JCC	JCC
10001X	10/10/85	JCC	JCC	JCC	JCC	JCC
10001Y	10/10/85	JCC	JCC	JCC	JCC	JCC
10001Z	10/10/85	JCC	JCC	JCC	JCC	JCC

FIGURE VII.B.11: DIGITAL BOARD SILKSCREEN



GMS ENGINEERING CORP.	
DIGITAL BOARD	
AUTOMATED DESIGN SERVICES 05-15-80	
SILKSCREEN	V929

FIGURE VII.B.12: RELAY BOX AUXILIARY CIRCUIT SCHEMATICS



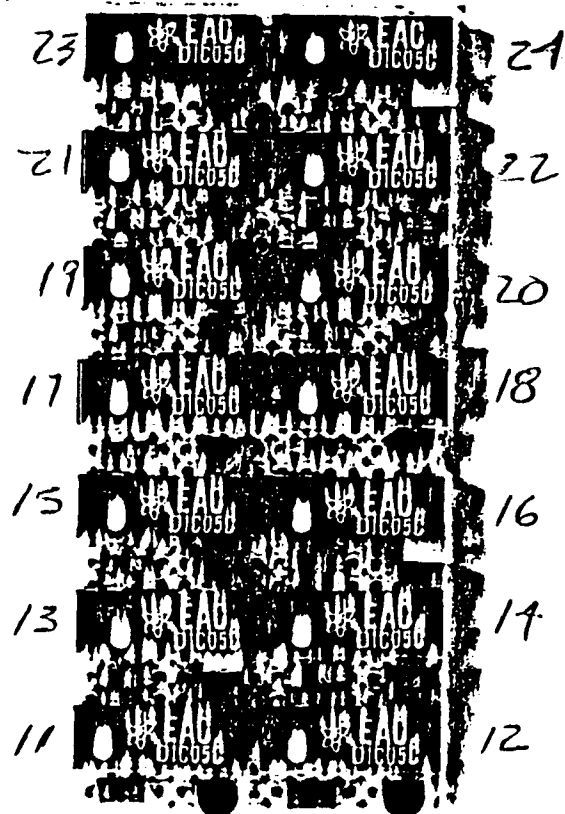
NOTE: ALL PETS AND TYPE 88179

Rev	Date	By	App
1.0	10/1/79	J. J. J.	J. J. J.
1.1	10/1/79	J. J. J.	J. J. J.
1.2	10/1/79	J. J. J.	J. J. J.
1.3	10/1/79	J. J. J.	J. J. J.
1.4	10/1/79	J. J. J.	J. J. J.
1.5	10/1/79	J. J. J.	J. J. J.
1.6	10/1/79	J. J. J.	J. J. J.
1.7	10/1/79	J. J. J.	J. J. J.
1.8	10/1/79	J. J. J.	J. J. J.
1.9	10/1/79	J. J. J.	J. J. J.
2.0	10/1/79	J. J. J.	J. J. J.

PROPRIETARY INFORMATION

Rev	Date	By	App
1.0	10/1/79	J. J. J.	J. J. J.
1.1	10/1/79	J. J. J.	J. J. J.
1.2	10/1/79	J. J. J.	J. J. J.
1.3	10/1/79	J. J. J.	J. J. J.
1.4	10/1/79	J. J. J.	J. J. J.
1.5	10/1/79	J. J. J.	J. J. J.
1.6	10/1/79	J. J. J.	J. J. J.
1.7	10/1/79	J. J. J.	J. J. J.
1.8	10/1/79	J. J. J.	J. J. J.
1.9	10/1/79	J. J. J.	J. J. J.
2.0	10/1/79	J. J. J.	J. J. J.

FIGURE VII.B.13: RELAY BOARDS



RELAY BOX

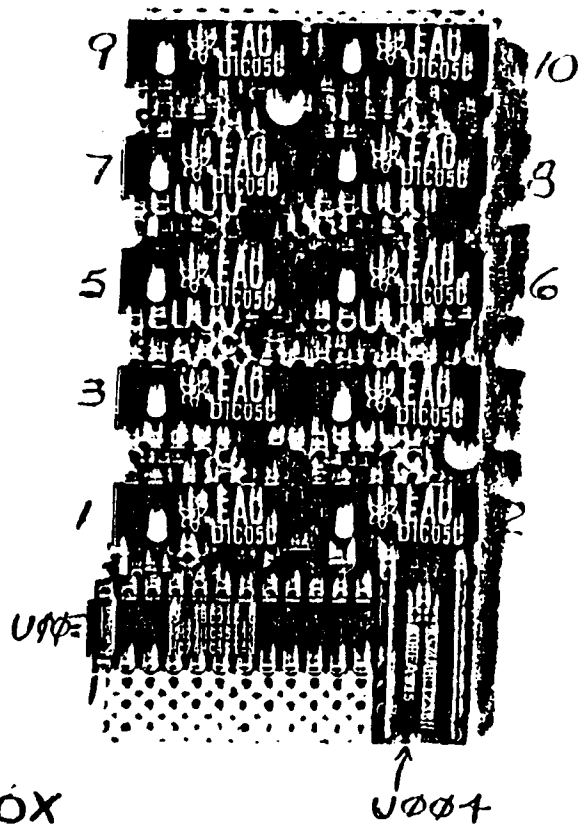
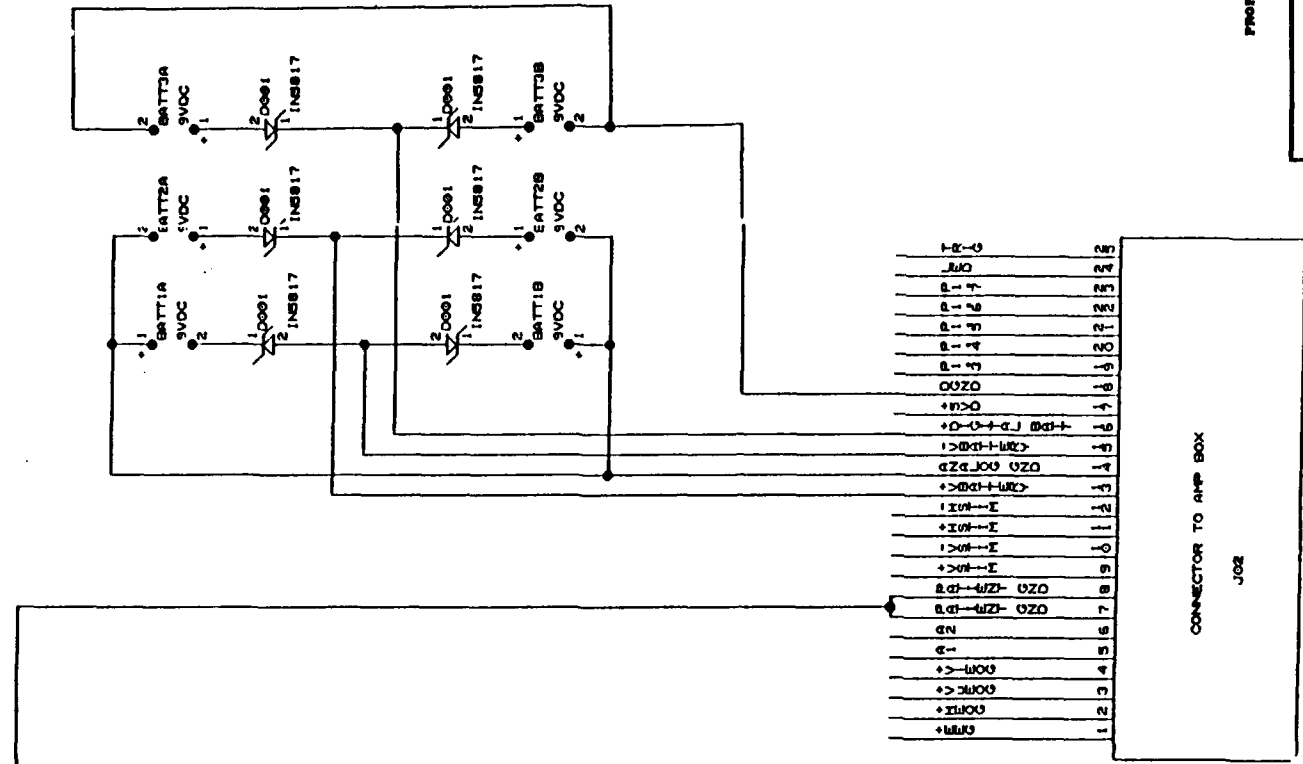
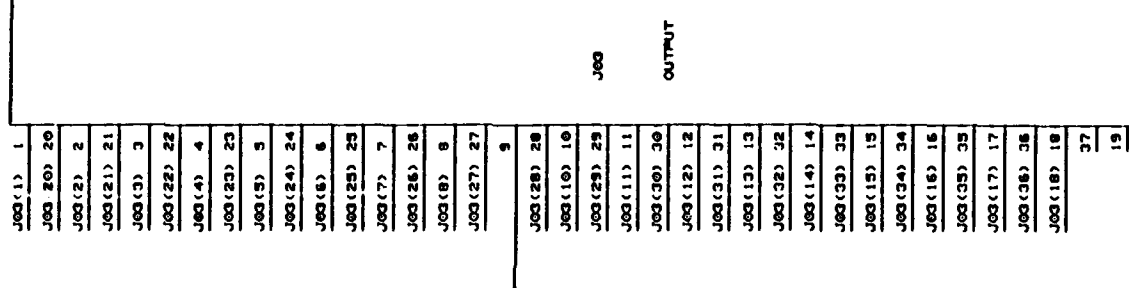
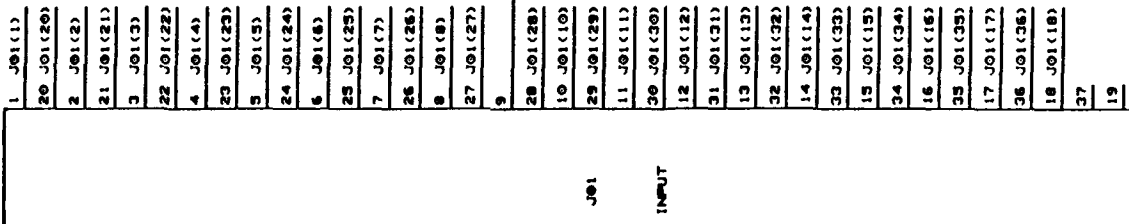


FIGURE VII.B.14: RELAY BOX CONNECTOR SCHEMATIC

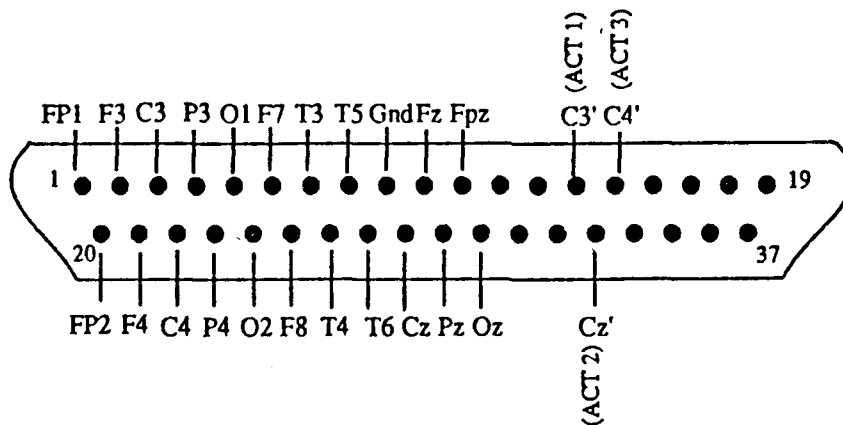
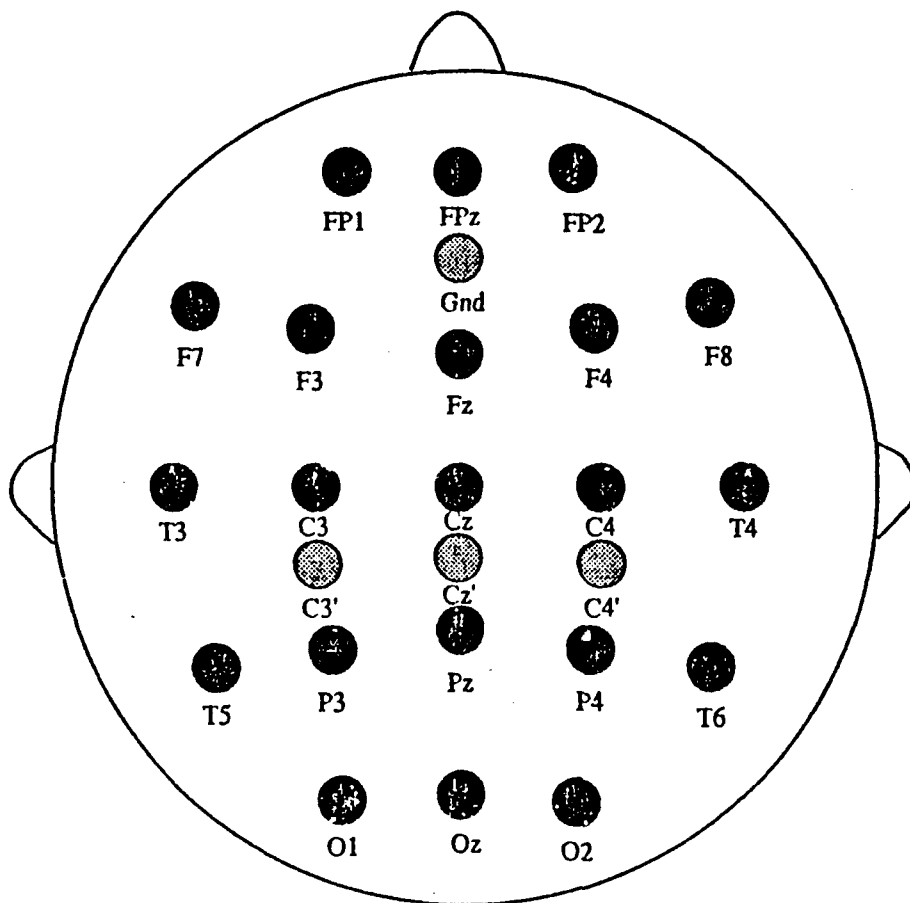


PROPRIETARY INFORMATION

SEC RELAY BOX	Don	J00
Sheet 2 of 2	Enr	JCC
G M S ENGINEERING	Chr	
CORPORATION	One	EEG10A
Columbia, Maryland	Rev	A
	Date	3/15/90

FIGURE VII.B.15: 37 PIN "D" CONNECTOR PINOUT DESIGNATION

SPECTRUM 32 HEAD CAP CONFIGURATION



37 PIN "D" HEAD CAP CONNECTOR

C3' = ACT 1 on Spectrum 32 Headbox.
 Cz' = ACT 2 on Spectrum 32 Headbox.
 C4' = ACT 3 on Spectrum 32 Headbox.
 Gnd = Iso Gnd on Spectrum 32 Headbox.

NOTE: In order to use the Prime electrodes (C3', Cz', or C4') you must specify the correct active input (ACT 1, 2, or 3) on the Spectrum 32.

C. SOFTWARE DESCRIPTION

There are two components of the software for the EEG Artifact Rejection System. These components are the EPROM code which is contained in the unit and the PC based code which is contained on the disk. The complete software are appended here.

The EPROM code ("EEG" written in C, and "FFT_FOR" written in assembler) consists of six parts. The first part is the serial communications routine. This allows the user to control what the unit does as well as selection of the desired channel and the LED level. This also is the vehicle for data transfer. The second part is the service of the hardware and housekeeping routines. This part is necessary for all software. The processor, memory, and peripheral circuits require specific signals and protocols. This is accomplished in the housekeeping and hardware service routines. The batteries are checked here as well. The third and fourth parts are similar to each other. They perform the real-time monitoring function. The fourth part adds the calibration pulses if the user desires them. The fifth part performs the direct interrogation. The vertical, horizontal, and calibration drives are output, and the electrode signals are recorded and stored for later processing. The last part is the application of the model for correction of the EEG.

The PC-based code ("EARS" and "EEG" which consists of "SHELL", "DCALC2", "SGM", and "FTUNE2" all written in Fortran) consists of four parts. The first part is the serial communications routine. This is the PC side of the communications described above. This program is "EARS". The second (DCALC2), third (SGM), and fourth (FTUNE2) parts are all contained in "EEG". The second part allows the user to enter the geometrical measurements, and calculates the model parameters from these measurements. The third part takes the interrogation data and calculates the medium transfer function for the specific test subject. The last part statistically fine tunes the model coefficients obtained in the second

part using the transfer function obtained in the third part. The coefficients and the transfer function must be then sent back to the main processor unit before the correction can commence. This is accomplished by option "C" in "EARS".

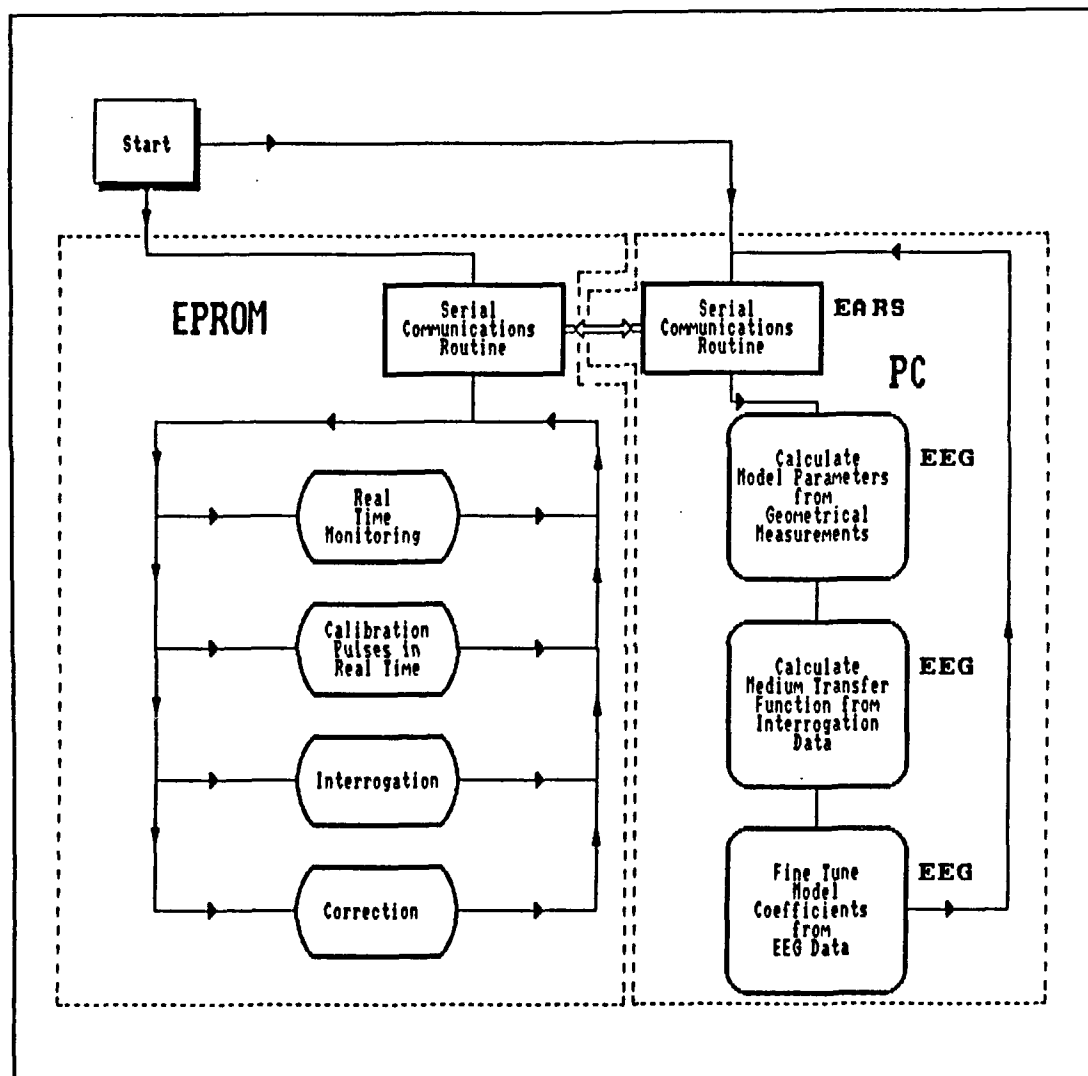


FIGURE VII.C.1: SOFTWARE BLOCK DIAGRAM.

VIII. CALIBRATION PROCEDURES

By choosing the "P" (Calibration Pulses) option on the main menu, the user can tune the gains and offsets while viewing the input signal with a common calibration pulse riding on all the channels. This amplitude of this calibration pulse can be adjusted by the user. This may be required when the channel gains are changed to the upper or lower extremes. This procedure is as follows.

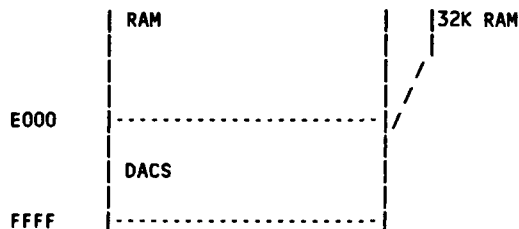
The EARS main processor unit must be opened by removing the seven screws on the face opposite the potentiometers. Once the top is removed, the several printed circuit boards in the unit will become visible. On the side of the main printed circuit board closest to the RS-232 connector is a jumper and two monitoring pins. The jumper can be moved to the next position, which bypasses the relay and continuously applies the calibration battery to the circuit. The monitoring pins can be used to measure the exact voltage of the calibration pulse. A microvolt meter must be used for this purpose. The potentiometer control on the outside opposite face of the unit can be used to adjust the voltage. The jumper **MUST** be placed back in the factory position after this is complete and before the unit is closed.

IX. INDEX

aircraft	5
batteries	8, 9, 14, 15, 23, 54
battery	5, 8, 10, 11, 14, 15, 56
bi-ocular dipoles	19
biopotential amplifier	6
calibration	11, 13, 23, 54, 56
correction	5, 6, 11, 14, 16, 20, 21, 54, 55
dipole	16-20
EARS	1, 5, 6, 8, 11-16, 23, 54-56
EARS menu	11
EEG	5, 6, 11-14, 16, 20-23, 54
electrodes	6, 11, 13, 15
EOG	6, 12-14, 20, 21, 23
EPROM	54
eye movements	16
facial coordinate system	18
install	8
interrogating the medium	13
interrogation	6, 11, 13, 21, 54
light level	11, 12
ocular dipoles	18-20
operation	5, 8, 10, 11, 14, 16
power light	14, 23
processor	6, 8, 11, 13-15, 54-56
real-time	5, 6, 54
relay box	6-8, 11, 12, 14, 15, 46, 50
selector	6-8
simulator	5
software	8, 11, 16, 54, 55, 58, 59
stimulus dipoles	20
storage	15

X. APPENDICES

APPENDIX A: PROGRAM SOFTWARE LISTING



Port pin assignments

Pin	Direction	Function	P1.x=1	P1.x=0
P1.0	output	Cal	ON	OFF
P1.1	output	Stim1	ON	OFF
P1.2	output	Stim2	ON	OFF
P1.3	output	Relay Address (LSB)		
P1.4	output	Relay Address		
P1.5	output	Relay Address		
P1.6	output	Relay Address		
P1.7	output	Relay Address (MSB)		
P2.5	output	LED		
P2.2	input	Event Trigger In		

#	Input Channels	#	Stored in data_buffer[#] [] []	Output Chan.
0	EEG	EEG	uEEG(t)	
1	HEOG	HEOG		ETO(t)
2	VEOGu	VEOGu		cEEG(t-T)
3	VEOGL	VEOGL		ETO(t-T)
4		Event Trigger Out		HEOG(t-T)
5				VEOGu(t-T)
6				VEOGL(t-T)
7				

This routine must be linked with:

```
e_int.obj (macro EI)
CCR.ABS (ccb)
user.lib (patched c96.lib)
fft_for.obj (the FFT)
cstart.obj (main module)
plm96.lib
```

i.e.,

```
eeg.obj, cstart.obj, ccr.abs, e_int.obj, fft_for.obj, &
user.lib, plm96.lib to eeg ixref &
ro(2000H-2013H, 2018H-2018H, 2030H-203FH, 2080H-5FFFH) &
ra(1AH-1FFDH(STACK), 6000H-0DFFFH)
```

```

1
1      rtine_flag = 0      =      real-time
1      rtine_flag = 1      =      interrogation
1      rtine_flag = 2      =      collection of epochs for fine tuning
1      rtine_flag = 3      =      correction
1      rtine_flag = 7      =      real-time with cal pulses
1      rtine_flag = 99     =      stop
1
1      */
1
1
1
1      /* Headers */
1
1      #include <80C196.h>          /* 80C196 I/O registers */
34     #include <ctype.h>
35     #include <setjmp.h>
36     #include <stdio.h>
44     #include <stdlib.h>
52     #include <string.h>
65
65
65
65
65     /* Definitions */
65
65     #define      PRE_SCALE      64
65
65     #define      BUFF_START      64
65     #define      BUFF_DIFF      384      /* 0.75*SAMP_NUM */
65     #define      CHAN_BASE      0x08      /* ADC Channel base (ADC0) */
65     #define      CHAN_NUM      4      /* number of input (ADC) channels */
65     #define      CHAN_M1      3      /* CHAN_NUM-1 */
65     #define      CHAN_P1      5      /* CHAN_NUM+1 */
65     #define      DELTA_T      0x0E371
65     #define      LEFT_LIM      128      /* 0.25*SAMP_NUM */
65     #define      RIGHT_LIM      383      /* (0.75*SAMP_NUM)-1 */
65     #define      SAMP_NUM      512      /* number of samples */
65     #define      TRO_PULSE      4      /* 30 msec */
65     #define      CAL_NUM      60      /* cal/stim samples in buffers */
65     #define      CAL_NUM_M1      59      /* CAL_NUM-1 */
65     #define      CAL_ON      39      /* Stim on */
65     #define      CAL_OFF      19      /* Stim off */
65     #define      CAL_REP      60      /* cal/stim repetitions for avg */
65     #define      DELTA_T_FAST      0x0F8DE
65     #define      SAMP_LIM      200      /* sample time out limit */
65     #define      SAMP_X2      1024
65     #define      LOW_BATT      650
65     #define      E_LOW_BATT      600
65     #define      H      0
65     #define      V      1
65     #define      R      0
65     #define      I      1
65
65     /* Interrupt service function assignments */
65
65     #pragma interrupt (nmi_int=31)      /* NMI interrupt */

```

```

65      #pragma interrupt (extint=29)          /* EXTINT Pin interrupt */
65      #pragma interrupt (receive=25)         /* Serial Port Receive interrupt */
65      #pragma interrupt (samp=1)            /* A2D CONVERSION COMPLETE interrupt */
65      #pragma interrupt (time1=0)           /* TIMER1 OVERFLOW interrupt */
65
65      /* Function declarations */
65
65      void          main(void);
66
65      void          nmi_int(void);           /* Interrupt Service Routines */
66      void          extint(void);
67      void          receive(void);
68      void          time1(void);
69      void          samp(void);
70
71      void          serial(void);
72      void          senddata(int);
73      int           recvdata(void);
74      void          dac(int,int);
75      void          err(void);
76      void          fft_for(void);
77
77      /* External functions */
77
77      extern void    enab_int(void);
78
78      /* SFP Images */
78
78      unsigned char  im_ioc0, im_ioc1, im_ioc2, im_sp_stat, import1, pmwm;
79      register unsigned char  status_temp;      /* defined in USER.LIB */
80
80      /* Global variables */
80
80      register long int  tltemp1, tltemp2, tltemp3, tltemp4, tdenom;
81      register int       gain_mag, new_pt, alpha1, alpha2, sgv_mag, sgh_mag, tune_pt, temp1_reg,
temp2_reg;
82      register char      chan, buff_num, buff_not, chan_end;
83
83      register int       dv[8];
84
84      int               cal_counter, cal_mode, rep_count, cal_monitor, cal_cnt, correct_cnt;
85      char              timer1_flag, ri_flag, correct_flag, error, loop_flag;
86      char              restart_flag, tro_counter, rtine_flag;
87      int               battlevel, battcnt, i_temp;
88      int               xreal[512], ximag[512];
89      long int          lltemp1, lltemp2, lltemp3, lltemp4, denom;
90      int               out_buffer[2][SAMP_NUM];
91      int               alpha[SAMP_NUM];
92      int               eogf[2][2][SAMP_NUM];
93      int               data_buffer[CHAN_P1][2][SAMP_NUM], batt_volt[3];
94      int               data_tuner[CHAN_NUM][SAMP_X2];
95      int               gssum1[CAL_NUM];

```

[illegible]

```

103      16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
103      16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
103      16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
103      16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
103      16384, 16383, 16383, 16383, 16383, 16383, 16383, 16383,
103      16383, 16382, 16382, 16382, 16382, 16381, 16381, 16381,
103      16380, 16380, 16380, 16379, 16378, 16378, 16377, 16376,
103      16376, 16375, 16374, 16373, 16371, 16370, 16369, 16367,
103      16366, 16364, 16362, 16360, 16358, 16355, 16353, 16350,
103      16347, 16343, 16340, 16336, 16332, 16327, 16323, 16317,
103      16312, 16306, 16300, 16293, 16285, 16278, 16269, 16260,
103      16251, 16241, 16230, 16218, 16206, 16193, 16179, 16164,
103      16148, 16131, 16113, 16094, 16074, 16052, 16030, 16006,
103      15980, 15953, 15925, 15895, 15863, 15829, 15794, 15756,
103      15717, 15675, 15631, 15585, 15536, 15485, 15431, 15375,
103      15316, 15253, 15188, 15120, 15048, 14973, 14894, 14812,
103      14726, 14636, 14543, 14445, 14343, 14237, 14126, 14011,
103      13891, 13767, 13637, 13503, 13364, 13220, 13070, 12916,
103      12756, 12591, 12421, 12245, 12065, 11878, 11687, 11490,
103      11288, 11081, 10869, 10651, 10429, 10203, 9972, 9736,
103      9497, 9253, 9006, 8756, 8503, 8246, 7988, 7728,
103      7465, 7202, 6938, 6674, 6409, 6146, 5883, 5622,
103      5363, 5106, 4853, 4603, 4357, 4115, 3879, 3647,
103      3422, 3203, 2991, 2785, 2587, 2396, 2213, 2039,
103      1872, 1714, 1564, 1423, 1289, 1165, 1048, 940,
103      840, 747, 662, 585, 514, 449, 391, 339,
103      293, 251, 214, 182, 154, 129, 108, 90,
103      74, 61, 49, 40, 32, 26, 20, 16};
104
104
104      /*=====*/
104
104      void main(void)
104      (
105          1      int      i, j, k;
106          1
106          1      /* System Configuration */
106          1
106          1          int_mask = 0x00;          /* mask all interrupts */
107          1          imask1 = 0x00;
108          1          enab_int();          /* enable global interrupts */
109          1
109          1          im_ioc0 = 0x00;          /* set I/O control registers */
110          1          ioc0 = im_ioc0;
111          1
111          1          im_ioc1 = 0x00;
112          1          ioc1 = im_ioc1;
113          1
113          1          im_ioc2 = 0x00;
114          1          ioc2 = im_ioc2;
115          1
115          1          import1 = 0x00;          /* initial Port1 config. */
116          1          ioport1 = import1;
117          1          pwm = 127;
118          1          pwm_control = pwm;
119          1

```

```

119 1      /* Set up Serial Port */
119 1
119 1      im_ioc1 |= 0x21;          /* select TXD on P2.0 */
120 1      ioc1 = im_ioc1;
121 1      baud_rate = 0x40;
122 1      baud_rate = 0x80;      /* baud rate of 9600 on 12 MHz */
123 1      sp_con = 0x09;
124 1      wsr = 0x0F;          /* Mode 1, enable receive, no parity */
125 1      im_sp_stat = 0x20;    /* alternate window */
126 1      status_temp = im_sp_stat; /* set the initial TI bit */
127 1      wsr = 0;
128 1      gain_mag = 1;
129 1      sgv_mag = 1;
130 1      sgh_mag = 1;
131 1      rtine_flag = 0;
132 1
132 1      restart_flag = 1;
133 1
133 1      /* Initialize flags & pointers */
133 1
133 1      restart:
133 1      /*      if ( restart_flag == 1 ) {*/
133 1          restart_flag = 0;
134 1          error = 0;
135 1          correct_flag = 0;
136 1          correct_cnt = 2;
137 1          tro_counter = 0;
138 1          cal_counter = 0;
139 1          cal_cnt = 0;
140 1          cal_monitor = 0;
141 1          battlevel = 0;
142 1          battcnt = 0;
143 1          new_pt = BUFF_START; /* init. data buffer pointers */
144 1          tune_pt = 0;
145 1          buff_num = 0;
146 1          buff_not = 1;
147 1
147 1          for ( i=0; i < 8; i++)
148 1              dac( i, 0 ); /* zero DACs */
149 1
149 1          /* Clear buffers */
149 1
149 1          for ( i=0; i < 2; i++) {
150 2              for ( j=0; j < SAMP_NUM; j++) {
151 3                  for ( k=0; k < CHAN_P1; k++)
152 3                      data_buffer[k][i][j] = 0;
153 3                      out_buffer[i][j] = 0; /* zero output buffer */
154 3                  }
154 2              }
154 1          for ( i=0; i<CHAN_NUM; i++ ) {
155 2              for ( j=0; j<SAMP_X2; j++ )
156 2                  data_tuner[i][j] = 0;
157 2              }
157 1          /*      }*/
157 1
157 1          for ( i_temp=0; i_temp<256; i_temp++ ) {

```

```

158 2          for ( j=0; j<8; j++ ) (
159 3              dac(j,i_temp-128);
160 3              for ( k=0; k<10; k++ )
161 3                  ;
161 3          )
161 2      )
161 1
161 1      for ( i=0; i < 8; i++)
162 1          dac( i, 64 );
163 1
163 1
163 1      /* Go to main menu */
163 1
163 1      serial();
164 1      loop_flag = 0;
165 1      ri_flag = 0;
166 1      ipend1 &= ~0x02;
167 1
167 1      /* Initialize Timer1 */
167 1
167 1      im_ioc1 |= 0x04;          /* enable TIMER1 overflow intrpt */
168 1      ioc1 = im_ioc1;
169 1
169 1      timer1_flag = 0;        /* timer1 flag */
170 1      loop_flag = 0;
171 1      wsr = 0x0F;             /* alternate window */
172 1      timer1 = DELTA_T;      /* load 7.813 msec timer */
173 1      wsr = 0;
174 1
174 1      int_mask |= 0x03;        /* unmask TIMER1, A2D DONE, */
175 1      imask1 |= 0x22;         /* EXTINT, and RI intrpt */
176 1
176 1
176 1      /* Loop endlessly..... */
176 1
176 1      /* Wait for Timer1 Overflow (every 7.813 msec) to start another sampling
176 1      sequence. The TIMER1 interrupt handling routine starts the CHAN_NUM channel
176 1      sweep: First the last data point is written out to the DAC, then each ADC is
176 1      sampled on an interrupt driven basis. */
176 1
176 1
176 1      next:
176 1
176 1      /* Has the serial port received a character? */
176 1
176 1      if ( (ri_flag == 1) | (rtine_flag == 99) ) (
177 2          int_mask &= ~0x03;    /* mask TIMER1 & A2D DONE intrpts */
178 2          imask1 &= ~0x22;     /* mask EXTINT & RI intrpts */
179 2          serial();
180 2          loop_flag = 0;
181 2          ri_flag = 0;
182 2          correct_flag = 0;
183 2          correct_cnt = 2;
184 2          timer1_flag = 0;
185 2          ipend1 &= ~0x02;

```

```

186 2      wsr = 0x0F;          /* alternate window */
187 2      timer1 = DELTA_T;
188 2      wsr = 0;
189 2      int_mask |= 0x03;      /* unmask TIMER1 & A2D DONE intrpts */
190 2      imask1 |= 0x22;      /* unmask EXTINT & RI intrpts */
191 2      goto wait;
192 2  )
192 1
192 1      battlevel = 0;
193 1      for ( i=0; i<3; i++ ) {
194 2          if ( batt_volt[i] < LOW_BATT )
195 2              battlevel = 300;
196 2          if ( batt_volt[i] < E_LOW_BATT )
197 2              battlevel = 70;
198 2      }
198 1
198 1      if ( restart_flag == 1 )
199 1          goto restart;
200 1      if ( correct_flag != 1 )
201 1          goto wait;
202 1
202 1      /* Load EOG-VU, window the data, multiply by PRE_SCALE, & transform */
202 1      for ( i_temp=0; i_temp < 512; i_temp++ ) {
203 2          xreal[i_temp] = (int) (((long) data_buffer[2][buff_not][i_temp])*supgau[i_temp])/256);
204 2          ximag[i_temp] = 0;
205 2      }
205 1
205 1      fft_for();
206 1      if ( error != 0 ) {
207 2          err();
208 2          goto restart;
209 2      }
209 1
209 1      /* Save VU-EOG(w) */
209 1
209 1      for ( i_temp=0; i_temp < 512; i_temp++ ) {
210 2          eogf[0][0][i_temp] = (int) (((long) xreal[i_temp] * 10) / PRE_SCALE);
211 2          eogf[0][1][i_temp] = (int) (((long) ximag[i_temp] * 10) / PRE_SCALE);
212 2
212 2      /* Load EOG-VL, window the data, multiply by PRE_SCALE, & transform */
212 2      xreal[i_temp] = (int) (((long) data_buffer[3][buff_not][i_temp])*supgau[i_temp])/256);
213 2      ximag[i_temp] = 0;
214 2  }
214 1
214 1      fft_for();
215 1      if ( error != 0 ) {
216 2          err();
217 2          goto restart;
218 2      }
218 1
218 1

```

```

218 1      /* alpha(w) = EOG-VU(w) / EOG-VL(w) ; alpha(w) is scaled by 10 */
218 1
218 1      for ( i_temp=0; i_temp < 512; i_temp++ ) {
219 2
219 2          xreal[i_temp] = (int) (((long) xreal[i_temp] * 10) / PRE_SCALE);
220 2          ximag[i_temp] = (int) (((long) ximag[i_temp] * 10) / PRE_SCALE);
221 2
221 2          lltemp1 = gain_mag * ((eogf[0][0][i_temp] * (long) xreal[i_temp])
221 2              + (eogf[0][1][i_temp] * (long) ximag[i_temp]));
222 2          denom = ( (((long) xreal[i_temp]) * xreal[i_temp]) +
222 2              (((long) ximag[i_temp]) * ximag[i_temp]) ) * 10;
223 2
223 2          if ( lltemp1 < 0 )
224 2              lltemp1 = -lltemp1;
225 2
225 2          if ( denom == 0 )
226 2              xreal[i_temp] = 40;
227 2          else {
228 3              lltemp2 = lltemp1 / denom;
229 3              if ( lltemp2 > 40 )
230 3                  lltemp2 = 40;
231 3              if ( lltemp2 < 10 )
232 3                  lltemp2 = 10;
233 3              xreal[i_temp] = (int) lltemp2;
234 3          }
234 2          alpha[i_temp] = xreal[i_temp];
235 2      }
235 1
235 1
235 1
235 1
235 1      /* Load EOG-H(L-R), window the data, multiply by PRE_SCALE, & transform */
235 1
235 1      for ( i_temp=0; i_temp < 512; i_temp++ ) {
236 2          xreal[i_temp] = (int) (((long) data_buffer[1][buff_not][i_temp]) * supgau[i_temp]) / 256;
237 2          ximag[i_temp] = 0;
238 2      }
238 1
238 1      fft_for();
239 1      if ( error != 0 ) {
240 2          err();
241 2          goto restart;
242 2      }
242 1
242 1      for ( i_temp=0; i_temp < 512; i_temp++ ) {
243 2          eogf[1][0][i_temp] = (int) (((long) xreal[i_temp] * 10) / PRE_SCALE);
244 2          eogf[1][1][i_temp] = (int) (((long) ximag[i_temp] * 10) / PRE_SCALE);
245 2      }
245 1
245 1
245 1
245 1      /*-----*/
245 1
245 1      /* Correct the EEG */
245 1
245 1      for ( i_temp=0; i_temp < SAMP_NUM; i_temp++ ) {
246 2

```

```

246 2      alpha1 = (alpha[i_temp] + 10)/2;          /* alpha + 1 */
247 2      alpha2 = (alpha[i_temp] - 10)/2;          /* alpha - 1 */
248 2
248 2      /*      (alpha+1)*D1v + (alpha-1)*D2v
248 2      DSGv(w) = ----- * SGv(w)
248 2      (alpha+1)*D3v + (alpha-1)*D4v
248 2      */
248 2
248 2      lltemp1 = (((long) dv[0])*alpha1) + (((long) dv[1])*alpha2);
249 2      lltemp2 = (((long) dv[2])*alpha1) + (((long) dv[3])*alpha2);
250 2      if ( lltemp2 == 0 ) {
251 3          error = 100;
252 3          err();
253 3          goto restart;
254 3      }
254 2
254 2      /*      DSGv(w) * EOGvu(w)          */
254 2
254 2      lltemp3 = lltemp1 * eogf[0][0][i_temp];
255 2      lltemp4 = lltemp1 * eogf[0][1][i_temp];
256 2
256 2      xreal[i_temp] = (int) ( (lltemp3 / lltemp2) & 65535 );
257 2      ximag[i_temp] = (int) ( (lltemp4 / lltemp2) & 65535 );
258 2
258 2
258 2      /*      (alpha+1)*D1h + (alpha-1)*D2h
258 2      DSGh(w) = ----- * SGh(w)
258 2      (alpha+1)*D3h + (alpha-1)*D4h
258 2      */
258 2
258 2      lltemp1 = (((long) dv[4])*alpha1) + (((long) dv[5])*alpha2);
259 2      lltemp2 = (((long) dv[6])*alpha1) + (((long) dv[7])*alpha2);
260 2      if ( lltemp2 == 0 ) {
261 3          error = 101;
262 3          err();
263 3          goto restart;
264 3      }
264 2
264 2      /*      DSGh(w) * EOGh(w)          */
264 2
264 2      lltemp3 = lltemp1 * eogf[1][0][i_temp];
265 2      lltemp4 = lltemp1 * eogf[1][1][i_temp];
266 2
266 2      /*      (DSGv(w) * EOGvu(w)) + (DSGh(w) * EOGh(w))          */
266 2
266 2      xreal[i_temp] += (int) ( (lltemp3 / lltemp2) & 65535 );
267 2      ximag[i_temp] += (int) ( (lltemp4 / lltemp2) & 65535 );
268 2
268 2
268 2      /* Complex Conjugate */
268 2
268 2      ximag[i_temp] = -ximag[i_temp];
269 2      }
269 1
269 1
269 1      /* Inverse Transform */

```

```

269 1
269 1      fft_for();                      /* inverse FFT */
270 1      if ( error != 0 ) {
271 2          err();
272 2          goto restart;
273 2      }
273 1
273 1      /* Remove Window */
273 1
273 1      for ( i_temp=BUFF_START; i_temp < (SAMP_NUM-BUFF_START); i_temp++ ) {
274 2          xreal[i_temp] = (int)(( (long) xreal[i_temp] ) * 83886) / supgau[i_temp]);
275 2      }
275 1
275 1
275 1      /* Moving Average Filter (7 point) of corrector */
275 1
275 1      for ( i_temp=BUFF_START; i_temp < (SAMP_NUM-BUFF_START); i_temp++ ) {
276 2          alpha[i_temp] = (xreal[i_temp-3] + xreal[i_temp-2] + xreal[i_temp-1] + xreal[i_temp]
+
276 2          xreal[i_temp+1] + xreal[i_temp+2] + xreal[i_temp+3]) / 7;
277 2      }
277 1
277 1
277 1      /*      EEGcorr(t) = EEGobs(t) - IFFT( (SGv * EOGvu) + (SGh * EOGh) ) */
277 1
277 1      for ( i_temp=BUFF_START; i_temp < (SAMP_NUM-BUFF_START); i_temp++ ) {
278 2          out_buffer[buf_not][i_temp] = data_buffer[0][buf_not][i_temp] - alpha[i_temp];
279 2      }
279 1
279 1      correct_flag = 0;
280 1
280 1      wait:
280 1          loop_flag--;
281 1          while ( timer1_flag == 0)
282 1              ;
282 1          timer1_flag = 0;
283 1
283 1      /* Loop again */
283 1
283 1      goto next;                      /* wait for another TIMER1 INTERRUPT */
284 1  }
286
286
286
286      /*-----*/
286      /* nmi_int Function - NMI interrupt handler */
286
286      void nmi_int(void)
286      {
287 1          imask1 = 0x00;                /* disable interrupts */
288 1          int_mask = 0x00;
289 1          error = 3;                    /* error */
290 1          err();
291 1      }
293
293
293
293

```

```

293
293      /*-----*/
293      /* extint Function - EXTINT interrupt handler */
293      /*          - senses an external trigger on TRI */
293
293      void    extint(void)
293      {
294          1      tro_counter = TRO_PULSE;          /* load TRO counter */
295          1      }
297
297
297
297
297      /*-----*/
297      /* Receive Function - Serial Port Receive interrupt handler */
297
297      void    receive(void)
297      {
298          1      ri_flag = 1;
299          1      getchar();
300          1      }
302
302
302
302
302
302      /*-----*/
302      /* TIME1 - TIMER1 overflow interrupt handler
302          - Operates in conjunction with SAMP */
302
302      void    time1(void)
302      {
303          1      int      i, j, k;
304          1
304          1      /* Reset TIMER1 */
304          1
304          1      if ( rtine_flag != 1 ) {
305          2          wsr = 0x0F;
306          2          timer1 = DELTA_T;
307          2          wsr = 0;
308          2      }
308          1      else {
309          2          wsr = 0x0F;
310          2          timer1 = DELTA_T_FAST;
311          2          wsr = 0;
312          2      }
312          1
312          1      if ( (loop_flag > 0) & (rtine_flag != 3) ) {
313          2          error = 88;
314          2          err();
315          2      }
315          1
315          1      /* Is TRO high ? */
315          1
315          1      if ( tro_counter > 0 )
316          1          tro_counter--;

```

```

317 1
317 1 battcnt++;
318 1 if ( battcnt >= 32000 )
319 1 battcnt = 0;
320 1 if ( battlevel != 0 ) {
321 2 if ( (battcnt % battlevel) == 0 ) {
322 3 if ( ((battcnt / battlevel) % 2) == 0 )
323 3 pwm_control = pmwm;
324 3 else
325 3 pwm_control = 0;
326 3 }
326 2 }
326 1
326 1 /* Interrogate ? */
326 1
326 1 if ( rtine_flag == 1 ) {
327 2 i = CAL_NUM_M1 - cal_counter;
328 2 gssum1[i] += data_buffer[((cal_mode%2)/cal_mode)*2][buff_num][new_pt-1];
329 2 gssum2[i] += data_buffer[((cal_mode%2)+2-(cal_mode/2)][buff_num][new_pt-1];
330 2
330 2 if ( cal_counter > CAL_ON ) {
331 3 import1 &= ~(stim_port[3-cal_mode]);
332 3 ioport1 = import1;
333 3 }
333 2 else if ( cal_counter > CAL_OFF ) {
335 3 import1 |= (stim_port[3-cal_mode]);
336 3 ioport1 = import1;
337 3 }
337 2 else if ( cal_counter >= 0 ) {
339 3 import1 &= ~(stim_port[3-cal_mode]);
340 3 ioport1 = import1;
341 3 }
341 2 cal_counter--;
342 2
342 2 if ( cal_counter < 0 ) {
343 3 cal_counter = CAL_NUM_M1;
344 3 rep_count--;
345 3
345 3 if ( rep_count == 0 ) {
346 4 int_mask &= ~0x01; /* mask TIMER1 intrpt */
347 4 cal_mode--;
348 4 if ( cal_mode == 0 ) {
349 5 rtine_flag = 2;
350 5 tune_pt = 0;
351 5 new_pt = BUFF_START;
352 5 buff_num = 0;
353 5 buff_not = 1;
354 5 }
354 4 rep_count = CAL_REP; /* reset average counter */
355 4
355 4 for ( i=0; i < CAL_NUM; i++) { /* average */
356 5 gs[2-cal_mode][0][i] = (gssum1[i] / CAL_REP);
357 5 gs[2-cal_mode][1][i] = (gssum2[i] / CAL_REP);
358 5 }
358 4 temp1_reg = (gs[2-cal_mode][0][CAL_NUM_M1] + gs[2-cal_mode][0][0]) / 2;
359 4 temp2_reg = (gs[2-cal_mode][1][CAL_NUM_M1] + gs[2-cal_mode][1][0]) / 2;
359 4 for ( i=0; i < CAL_NUM; i++) {

```



```

398      /* SAMP - A2D CONVERSION COMPLETE interrupt handler.
398      - Operates in conjunction with TIME1.
398      - The last sample in the data_buffer is written out to the
398      corresponding DAC, converted to 8 bits.
398      - reads the sample from the ADC.
398      - returns a 10 bit value.
398      - start a conversion on the next channel of the sweep. */
398
398      void    samp(void)
398
398      {
399          1      register int    temp_reg;
400          1
400          1      /*      Write out the data to the DAC
400          1      - convert raw data from 10 bits to 8 bits; */
400          1
400          1      if ( chan == 0 ) {
401          2          dac( 0, (data_buffer[0][buff_num][new_pt-1] >> 2) );
402          2          if ( rtine_flag == 3 ) {
403          3              dac( 1, (data_buffer[4][buff_num][new_pt-1] >> 2) );
404          3              dac( 2, (out_buffer[buff_num][new_pt] >> 2) );
405          3              dac( 3, (data_buffer[4][buff_num][new_pt] >> 2) );
406          3              dac( 4, (data_buffer[0][buff_num][new_pt] >> 2) );
407          3              dac( 5, (data_buffer[1][buff_num][new_pt] >> 2) );
408          3              dac( 6, (data_buffer[2][buff_num][new_pt] >> 2) );
409          3              dac( 7, (data_buffer[3][buff_num][new_pt] >> 2) );
410          3          }
410          2          else {
411          3              dac( 1, (data_buffer[1][buff_num][new_pt-1] >> 2) );
412          3              dac( 2, (data_buffer[2][buff_num][new_pt-1] >> 2) );
413          3              dac( 3, (data_buffer[3][buff_num][new_pt-1] >> 2) );
414          3          }
414          2      }
414          1
414          1
414          1      /* Read a new sample from the ADC; convert from 0 - 1023 to +- 512 */
414          1
414          1      if ( chan < 4 ) {
415          2          temp_reg = ad_result_hi;
416          2          temp_reg = ((( temp_reg << 8) + ad_result_lo) >> 6) - 512;
417          2          data_buffer[chan][buff_num][new_pt] = temp_reg;
418          2      }
418          1      else if ( chan < 7 ) {
420          2          temp_reg = ad_result_hi;
421          2          temp_reg = ((( temp_reg << 8) + ad_result_lo) >> 6);
422          2          batt_volt[chan-4] = temp_reg;
423          2      }
423          1      if ( chan == 3 ) {
424          2          data_buffer[4][buff_num][new_pt] = -200;
425          2          if ( tro_counter > 0 )
426          2              data_buffer[4][buff_num][new_pt] = 200;
427          2      }
427          1
427          1      /* If necessary, store it in the other buffer as well */
427          1
427          1      if ( (rtine_flag == 2) & (chan < CHAN_NUM) )
428          1          data_tuner[chan][tune_pt] = data_buffer[chan][buff_num][new_pt];

```

```

429 1      if ( new_pt < LEFT_LIM )
430 1          data_buffer[chan][buff_not][new_pt+BUFF_DIFF] = temp_reg;
431 1      else if ( new_pt > RIGHT_LIM )
432 1          data_buffer[chan][buff_not][new_pt-BUFF_DIFF] = temp_reg;
433 1
434 1
434 1      /* Either start a new conversion, or update the data pointers */
434 1
434 1      chan_end = 8;
435 1      if ( rtine_flag == 1 )
436 1          chan_end = 4;
437 1      chan++;
438 1      if ( chan < chan_end )
439 1          ad_command = CHAN_BASE + chan;          /* start next conversion */
440 1      else {                                     /* Update data buffer pointers */
441 2          if ( new_pt < (SAMP_NUM-BUFF_START) ) {
442 3              new_pt++;
443 3              tune_pt++;
444 3              if ( (tune_pt == SAMP_X2) & (rtine_flag == 2) )
445 3                  rtine_flag = 99;
446 3              if ( rtine_flag == 3 ) {
447 4                  if ( new_pt == 129 ) {
448 5                      if ( correct_flag == 1 ) {
449 6                          error = 80;
450 6                          err();
451 6                      }
452 5                      else {
453 6                          if ( correct_cnt > 0 )
454 6                              correct_cnt--;
455 6                          if ( correct_cnt == 0 )
456 6                              correct_flag = 1;
457 6                      }
458 5                  }
459 4              }
460 3          else {
461 3              new_pt = BUFF_START;
462 3              temp_reg = buff_not;
463 3              buff_not = buff_num;
464 3              buff_num = temp_reg;
465 3          }
466 2          loop_flag--;
467 2          int_mask &= ~0x02;
468 2      }
469 1
470 1  }
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

[illegible]

```

508 2      resp = getchar();
509 2      printf("%c",resp);
510 2      if ( resp == 13 )
511 2          goto lvl1;
512 2      if ( (resp<48) | (resp>=58) )
513 2          goto lvl1;
514 2      resptmp = (resptmp*10) + (resp-48);
515 2      goto lvl1;
516 2  lvl1:
516 2      if ( resptmp > 255 )
517 2          goto lvl4;
518 2      if ( resptmp == 0 )
519 2          resptmp = pmwm + 1;
520 2      resptmp--;
521 2      pmwm = resptmp;
522 2      pwm_control = pmwm;
523 2  }
523 1
523 1  else if ( (resp == 'R') | (resp == 'r') ) {
525 2      printf("\n\r\n\r\n\r      REAL TIME MONITORING...\n\r\n\r\n\r");
526 2      rtine_flag = 0;
527 2      goto menu_end;
528 2  }
528 1
528 1  else if ( (resp == 'P') | (resp == 'p') ) {
530 2      printf("\n\r\n\r\n\r      Calibration Pulses\n\r\n\r\n\r");
531 2      rtine_flag = 7;
532 2      cal_monitor = 300;
533 2      cal_cnt = 150;
534 2      goto menu_end;
535 2  }
535 1
535 1  else if ( (resp == 'I') | (resp == 'i') ) {
537 2      cal_counter = CAL_NUM_M1;
538 2      cal_mode = 3;
539 2      rep_count = CAL_REP;
540 2      rtine_flag = 1;
541 2      for ( i=0; i<CAL_NUM; i++ ) {
542 3          gssum1[i] = 0;
543 3          gssum2[i] = 0;
544 3      }
544 2      printf("\n\r\n\r\n\r      INTERROGATING...\n\r\n\r\n\r");
545 2      goto menu_end;
546 2  }
546 1
546 1  else if ( (resp == 'C') | (resp == 'c') ) {
548 2      printf("\n\r\n\r\n\r");
549 2      printf("Press 'PgUp', select '7', and then type 'correct'.\n\r");
550 2      sgv_mag = recvdata();
551 2      sgh_mag = recvdata();
552 2      gain_mag = recvdata();
553 2      for ( i=0; i<8; i++ )
554 2          dv[i] = recvdata();
555 2      dv[0] = ((int) (((long) dv[0] * sgv_mag) / 10)) & 65535;
556 2      dv[4] = ((int) (((long) dv[4] * sgh_mag) / 10)) & 65535;
557 2      rtine_flag = 3;
558 2      printf("\n\r\n\r\n\r      CORRECTING...\n\r\n\r\n\r");
559 2      resp = getchar();

```

```

560      2          goto menu_end;
561      2      }
561      1      else if ( (resp == 'N') | (resp == 'n') ) {
563      2      chan4:
563      2          resptmp = (import1 >> 3) + 1;
564      2          printf("\n\rCurrent channel number is: %d\n\rEnter new channel number (1-24) >
",resptmp);
565      2          resptmp = 0;
566      2      channn:
566      2          resp = getchar();
567      2          printf("%c",resp);
568      2          if ( resp == 13 )
569      2              goto chann;
570      2          if ( (resp<48) | (resp>=58) )
571      2              goto channn;
572      2          resptmp = (resptmp*10) + (resp-48);
573      2          goto channn;
574      2      chann:
574      2          if ( resptmp > 24 )
575      2              goto chan4;
576      2          if ( resptmp == 0 )
577      2              resptmp = (import1 >> 3) + 1;
578      2          resptmp--;
579      2          import1 &= 0x07;
580      2          import1 |= (resptmp << 3);
581      2          ioport1 = import1;
582      2      }
582      1
582      1      goto menu;
583      1
583      1      menu_end:
583      1      ;
583      1      }
585
585      /*-----*/
585      /*      SENDOUT DATA Function */
585
585      void      senddata(data_out)
585
585      int      data_out;
587
587      {
587      1          int      i, j, temp_data, temp_sign, data2_out, data_first;
588      1          char      temp_chr;
589      1
589      1          temp_sign = 0;
590      1          if ( data_out < 0 ) {
591      2              temp_sign = 1;
592      2              data_out = -data_out;
593      2          }
593      1          data2_out = 0;
594      1          data_first = data_out;
595      1          for ( i=4; i>=0; i-- ) {
596      2              data_first -= data2_out;
597      2              temp_data = data_first;
598      2              for ( j=i; j>0; j-- )
599      2                  temp_data /= 10;

```

```

600      2          data2_out = temp_data;
601      2          for ( j=i; j>0; j-- )
602      2              data2_out *= 10;
603      2          printf("%d", temp_data);
604      2      }
604      1          printf("%d\n", temp_sign);
605      1
605      1      }
607
607
607
607
607
607      /*-----*/
607      /*      RECEIVE DATA Function */
607
607      int      recvdata(void)
607
607      {
608      1          int      i, j, temp_data, data_in;
609      1          char      temp_chr;
610      1
610      1          temp_data = 0;
611      1          for ( i=4; i>=0; i-- ) {
612      2      retry1:
612      2          temp_chr = getchar();
613      2          if ( temp_chr == 32 )
614      2              temp_chr = 48;
615      2          if ( (temp_chr < 48) | (temp_chr >= 58) )
616      2              goto retry1;
617      2          data_in = (int) temp_chr - 48;
618      2          for ( j=0; j<i; j++ )
619      2              data_in *= 10;
620      2          temp_data += data_in;
621      2      }
621      1      retry2:
621      1          temp_chr = getchar();
622      1          if ( (temp_chr < 48) | (temp_chr > 49) )
623      1              goto retry2;
624      1          if ( temp_chr == 49 )
625      1              temp_data = -temp_data;
626      1          return ( temp_data );
627      1      }
627
629
629
629
629
629      /*-----*/
629      /*      DAC Function */
629
629      void      dac(channel, level)
629
629      int      channel, level;
631
631      {

```

```

631      int    dacc;
632      1
632      /* Decode DAC address */
632      1
        if ( channel <= 3 )
632      1          dacc = 0xE013 + channel;
633      1          else
634      1              dacc = 0xE007 + channel;
635      1
        if ( (dacc % 2) == 0 )
636      1            dacc -= 2;
637      1
638      1
638      1      /* Write out the sample to the DAC */
638      1
        if ( level > 127 )
639      1            level = 127;
640      1          else if ( level < -128 )
641      1            level = -128;
642      1
643      1      memset( dacc, (level+128), 1 );           /* +- 128 into 0 - 256 */
644      1  }
645      1
646      1
646      1
646      1
646      1
646      1      /*-----*/
646      1      /* Error Function */
646      1
        void err(void)
646      1      {
647      1          int i;
648      1
648      1          int_mask = 0x00;           /* mask all interrupts */
649      1          imask1 = 0x02;
650      1
650      1          for ( i=0; i < CHAN_NUM; i++) {
651      2              dac( i, 0 );         /* zero DACs */
652      2          }
652      1          printf(crlf);             /* clear the screen */
653      1          printf("\n\n\r\t\t\t%cERROR\n\n\r",bell);
654      1          printf("\t\t\tNo. %i\n\r",error);
655      1
655      1          int_mask &= ~0x03;         /* mask TIMER1 & A2D DONE intrpts */
656      1          imask1 &= ~0x02;          /* mask EXTINT & RI intrpts */
657      1          rtine_flag = 98;
658      1          serial();
659      1          ri_flag = 0;
660      1          ipend1 &= ~0x02;
661      1          int_mask |= 0x03;          /* unmask TIMER1 & A2D DONE intrpts */
662      1          imask1 |= 0x02;           /* unmask EXTINT & RI intrpts */
663      1          restart_flag = 1;
664      1      }
665      1
666      1

```

MODULE INFORMATION:

CODE AREA SIZE	= 16DFH	5855D
CONSTANT AREA SIZE	= 080EH	2062D
DATA AREA SIZE	= 6FDBH	28635D
STATIC REGS AREA SIZE	= 005AH	90D
OVERLAYABLE REGS AREA SIZE	= 000AH	10D
MAXIMUM STACK SIZE	= 0094H	148D

C-96 COMPILATION COMPLETE. 0 WARNINGS, 0 ERRORS

DOS 3.30 (038-N) MCS-96 MACRO ASSEMBLER, V1.2

SOURCE FILE: FFT_FOR.A96

OBJECT FILE: FFT_FOR.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: <none>

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			1	FFT_FO MODULE STACKSIZE(6)
			2	
			3	;FFT ALGORITHM FROM INTEL APPLICATIONS NOTE, AP-275, BY IRA HORDON
			4	; "EMBEDDED CONTROL APPLICATIONS", INTEL CORP, 1988.
			5	
	0000		6	RSEG
			7	EXTRN error
			8	
	0024		9	OSEG at 24H
	0024		10	TMPI: dsl 1
	0028		11	TMPI: dsl 1
	002C		12	TMPI1: dsl 1
	0030		13	TMPI1: dsl 1
	0034		14	XRTMP: dsl 1
	0038		15	XITMP: dsl 1
	003C		16	WRP: dsw 1
	003E		17	WIP: dsw 1
	0040		18	PWR: dsw 1
	0042		19	IN_CNT: dsw 1
	0044		20	NDIV2: dsw 1
			21	
	0046		22	KPTR: dsw 1
	0048		23	KN2: dsw 1
	004A		24	N_SUB_K: dsw 1
	004C		25	RK: dsw 1
	004E		26	RNK: dsw 1
	0050		27	SHFT_CNT: dsb 1
	0051		28	LOOP_CNT: dsb 1
			29	
			30	
	0000		31	DSEG
			32	
			33	EXTRN XREAL, XIMAG
			34	
			35	; XREAL, XIMAG: Base addresses for 512 16-bit signed
			36	; entries for real and imaginary numbers, respectively.
			37	
			38	\$EJECT

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			39	
	0000		40	CSEG
			41	
			42	PUBLIC fft_for ; Starting point for FFT algorithm
			43	
	0000		44	;
			45	FFT_for: ;;; START FOURIER CALCULATIONS
	0000 1100	E	46	clrb error ;;; 400 ' INITIALIZATION OF LOOP
			47	
	0002 FC		48	clrvt
	0003 B10151		49	ldb loop_cnt, #1
	0006 B10850		50	ldb shft_cnt, #8
	0009 A1000244		51	ld ndiv2, #512
			52	;
	0000		53	OUT_LOOP: ;;; 410 K=0
	0000 0146		54	clr kptr
			55	;
	000F 990951		56	cmpb loop_cnt, #9 ; 512 = 2^9
	0012 DA0220A3		57	bgt UNWEAVE
			58	
			59	
	0016		60	MID_LOOP: ;;; 430 INCRNT=0
	0016 0142		61	clr in_cnt
			62	
			63	
	0018		64	IN_LOOP: ;;; 440 'CALCULATIONS BEGIN HERE
	0018 65020042		65	add in_cnt, #2 ; 450 INCRNT=INCRNT+1
			66	;
	001C A04640		67	ld pwr, kptr
	001F 085040		68	shr pwr, shft_cnt ; Calculate mult factors
	0022 71FE40		69	andb pwr, #11111110b
	0025 A341FC0040	R	70	ld pwr, brev[pwr]
			71	;
	002A A341FC043C	R	72	gm: ld wrp, wr[pwr]
	002F A341FE083E	R	73	ld wip, wi[pwr]
	0034 44444648		74	add kn2, kptr, ndiv2
			75	
			76	;; Complex multiplication follows
			77	
			78	;
	0038 FE4F4900003C24	E	79	gm: mul tmp1, wrp, xreal[kn2]
	003F FE4F4900003E28	E	80	mul tmpi, wip, ximag[kn2]
	0046 682A26		81	sub tmp1+2, tmpi+2
			82	;
	0049 FE4F4900003C2C	E	83	mul tmp1, wrp, ximag[kn2]
	0050 FE4F4900003E28	E	84	mul tmpi, wip, xreal[kn2]
	0057 642E2A		85	add tmp1+2, tmpi+2

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			86	
			87	
			88	;; high byte only of a signed multiply
			89	;; provides an effective divide by two
	005A	DC55	90	BVT ERR1 ; branch on error
			91	
	005C	A34700002C	E 92	ld tmp1, xreal[kptr] ; 500 TMPR1=XR(K)/2
	0061	0A012C	93	shra tmp1, #1 ; TMP11=XI(K)/2
	0064	A347000030	E 94	ld tmp1, ximag[kptr]
	0069	0A0130	95	shra tmp1, #1
			96	
			97	;
	006C	48262C34	98	gr2: sub ;;; 510 XR(KN2) = TMPR1 = TMPR1 -TMPR
	0070	C349000034	E 99	xrtmp, tmp1, tmp1+2
			100	st xrtmp, xreal[kn2]
			101	;; 520 XI(KN2) = TMPR1 = TMP11 -TMP1
	0075	482A3038	102	gx2: sub xitmp, tmp1, tmp1+2
	0079	C349000038	E 103	st xitmp, ximag[kn2]
			104	;
	007E	44262C34	105	add xrtmp, tmp1, tmp1+2
	0082	C347000034	E 106	st xrtmp, xreal[kptr]
			107	;; 540 XI(K) = TMP11 + TMP1
	0087	442A3038	108	gx: add xitmp, tmp1, tmp1+2
	0088	C347000038	E 109	st xitmp, ximag[kptr]
			110	
	0090	DC23	111	BVT ERR2 ; Branch on error
			112	;
			113	ik: add ;;; 560 K = K + 1
	0092	65020046	114	add kptr, #2
			115	;
			116	;; 570 IF INCNT < N2 THEN GOTO 450
	0096	884442	117	cmp in_cnt, ndiv2
	0099	D602277B	118	blt IN_LOOP
			119	;
			120	add kptr, ndiv2 ;;; 580 K = K + N2
			121	;
			122	;; 590 IF K < N1 THEN GOTO 430
	00A0	89E0346	123	cmp kptr, #1022 ; N1 = 2 *(N-1)
	00A4	D602276E	124	blt MID_LOOP
			125	;
			126	;; 600 LOOP = LOOP + 1 : N2 = N2 / 2
	00A8	1751	127	incb loop_cnt ; 605 SHIFT = SHIFT + 1
	00AA	0A0144	128	shra ndiv2, #1
	00AD	1550	129	decb shft_cnt
			130	;
	00AF	275C	131	br OUT_LOOP ;;; 610 GOTO 400
			132	
	00B1	B10100	E 133	ERR1: ldb error, #01 ; overflow error
	00B4	F0	134	ret
	00B5	B10200	E 135	ERR2: ldb error, #02 ; overflow error
	00B8	F0	136	ret
			137	
			138	\$EJECT

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			139	
			140	;
			141	UNWEAVE: ;;; 700 ' REORDERING STARTS HERE
			142	
			143	;
			144	;;; 720 FOR K = 0 TO 511
00B9	0146		144	clr kptr
00BB	A100044A		145	ld n_sub_k, #1024
			146	
00BF			147	UN_LOOP: ;; Bit reversal of the transformed array
			148	
00BF	A347FC004C	R	149	ld rk, brev[kptr]
			150	
00C4	884C46		151	cmp kptr, rk
00C7	D628		152	bge ENDL
			153	
00C9	A347000024	E	154	ld tmp, xreal[kptr]
00CE	A347000028	E	155	ld tmpi, ximag[kptr]
00D3	A34D00002C	E	156	ld tmp1, xreal[rk]
00D8	A34D000030	E	157	ld tmpi1, ximag[rk]
			158	
00DD	C34D000024	E	159	st tmp, xreal[rk]
00E2	C34D000028	E	160	st tmpi, ximag[rk]
00E7	C34700002C	E	161	st tmp1, xreal[kptr]
00EC	C347000030	E	162	st tmpi1, ximag[kptr]
			163	
			164	;
00F1	65020046		165	ENDL: add kptr, #2 ;;; 950 NEXT K
00F5	6902004A		166	sub n_sub_k, #2
00F9	D7C4		167	bne UN_LOOP
			168	
00FB	F0		169	RET
			170	
			171	;\$nolist
00FC			172	CSEG
			173	
00FC			174	BREV:
			175	
00FC	0000000200010003		176	DCW 2*0, 2*256, 2*128, 2*384, 2*64, 2*320
0108	8001800340004002		177	DCW 2*192, 2*448, 2*32, 2*288, 2*160, 2*416
0114	C000C002C001C003		178	DCW 2*96, 2*352, 2*224, 2*480, 2*16, 2*272
0120	20012003A000A002		179	DCW 2*144, 2*400, 2*80, 2*336, 2*208, 2*464
012C	6000600260016003		180	DCW 2*48, 2*304, 2*176, 2*432, 2*112, 2*368
0138	E001E00310001002		181	DCW 2*240, 2*496, 2*8, 2*264, 2*136, 2*392
0144	9000900290019003		182	DCW 2*72, 2*328, 2*200, 2*456, 2*40, 2*296
0150	50015003D000D002		183	DCW 2*168, 2*424, 2*104, 2*360, 2*232, 2*488
015C	3000300230013003		184	DCW 2*24, 2*280, 2*152, 2*408, 2*88, 2*344
0168	B001B00370007002		185	DCW 2*216, 2*472, 2*56, 2*312, 2*184, 2*440
0174	F000F002F001F003		186	DCW 2*120, 2*376, 2*248, 2*504, 2*4, 2*260
0180	0801080388008802		187	DCW 2*132, 2*388, 2*68, 2*324, 2*196, 2*452
018C	4800480248014803		188	DCW 2*36, 2*292, 2*164, 2*420, 2*100, 2*356
0198	C801C80328002802		189	DCW 2*228, 2*484, 2*20, 2*276, 2*148, 2*404
01A4	A800A802A801A803		190	DCW 2*84, 2*340, 2*212, 2*468, 2*52, 2*308
01B0	68016803E800E802		191	DCW 2*180, 2*436, 2*116, 2*372, 2*244, 2*500
01BC	1800180218011803		192	DCW 2*12, 2*268, 2*140, 2*396, 2*76, 2*332
01C8	9801980358005802		193	DCW 2*204, 2*460, 2*44, 2*300, 2*172, 2*428
01D4	D800D802D801D803		194	DCW 2*108, 2*364, 2*236, 2*492, 2*28, 2*284
01E0	38013803B800B802		195	DCW 2*156, 2*412, 2*92, 2*348, 2*220, 2*476

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
01EC	7800780278017803	196	DCW 2*60, 2*316, 2*188, 2*444, 2*124, 2*380
01F8	F801F80304000402	197	DCW 2*252, 2*508, 2*2, 2*258, 2*130, 2*386
0204	8400840284018403	198	DCW 2*66, 2*322, 2*194, 2*450, 2*34, 2*290
0210	44014403C400C402	199	DCW 2*162, 2*418, 2*98, 2*354, 2*226, 2*482
021C	2400240224012403	200	DCW 2*18, 2*274, 2*146, 2*402, 2*82, 2*338
0228	A401A40364006402	201	DCW 2*210, 2*466, 2*50, 2*306, 2*178, 2*434
0234	E400E402E401E403	202	DCW 2*114, 2*370, 2*242, 2*498, 2*10, 2*266
0240	1401140394009402	203	DCW 2*138, 2*394, 2*74, 2*330, 2*202, 2*458
024C	5400540254015403	204	DCW 2*42, 2*298, 2*170, 2*426, 2*106, 2*362
0258	D401D40334003402	205	DCW 2*234, 2*490, 2*26, 2*282, 2*154, 2*410
0264	B400B402B401B403	206	DCW 2*90, 2*346, 2*218, 2*474, 2*58, 2*314
0270	74017403F400F402	207	DCW 2*186, 2*442, 2*122, 2*378, 2*250, 2*506
027C	0C000C020C010C03	208	DCW 2*6, 2*262, 2*134, 2*390, 2*70, 2*326
0288	8C018C034C004C02	209	DCW 2*198, 2*454, 2*38, 2*294, 2*166, 2*422
0294	CC00CC02CC01CC03	210	DCW 2*102, 2*358, 2*230, 2*486, 2*22, 2*278
02A0	2C012C03AC00AC02	211	DCW 2*150, 2*406, 2*86, 2*342, 2*214, 2*470
02AC	6C006C026C016C03	212	DCW 2*54, 2*310, 2*182, 2*438, 2*118, 2*374
02B8	EC01EC031C001C02	213	DCW 2*246, 2*502, 2*14, 2*270, 2*142, 2*398
02C4	9C009C029C019C03	214	DCW 2*78, 2*334, 2*206, 2*462, 2*46, 2*302
02D0	5C015C03DC00DC02	215	DCW 2*174, 2*430, 2*110, 2*366, 2*238, 2*494
02DC	3C003C023C013C03	216	DCW 2*30, 2*286, 2*158, 2*414, 2*94, 2*350
02E8	BC018C037C007C02	217	DCW 2*2, 2*478, 2*62, 2*318, 2*190, 2*446
02F4	FC00FC02FC01FC03	218	DCW 2*126, 2*382, 2*254, 2*510, 2*1, 2*257
0300	0201020382008202	219	DCW 2*129, 2*385, 2*65, 2*321, 2*193, 2*449
030C	4200420242014203	220	DCW 2*33, 2*289, 2*161, 2*417, 2*97, 2*353
0318	C201C20322002202	221	DCW 2*225, 2*481, 2*17, 2*273, 2*145, 2*401
0324	A200A202A201A203	222	DCW 2*81, 2*337, 2*209, 2*465, 2*49, 2*305
0330	62016203E200E202	223	DCW 2*177, 2*433, 2*113, 2*369, 2*241, 2*497
033C	1200120212011203	224	DCW 2*9, 2*265, 2*137, 2*393, 2*73, 2*329
0348	9201920352005202	225	DCW 2*201, 2*457, 2*41, 2*297, 2*169, 2*425
0354	D200D202D201D203	226	DCW 2*105, 2*361, 2*233, 2*489, 2*25, 2*281
0360	32013203B200B202	227	DCW 2*153, 2*409, 2*89, 2*345, 2*217, 2*473
036C	7200720272017203	228	DCW 2*57, 2*313, 2*185, 2*441, 2*121, 2*377
0378	F201F2030A000A02	229	DCW 2*249, 2*505, 2*5, 2*261, 2*133, 2*389
0384	8A008A028A018A03	230	DCW 2*69, 2*325, 2*197, 2*453, 2*37, 2*293
0390	4A014A03CA00CA02	231	DCW 2*165, 2*421, 2*101, 2*357, 2*229, 2*485
039C	2A002A022A012A03	232	DCW 2*21, 2*277, 2*149, 2*405, 2*85, 2*341
03A8	AA01AA036A006A02	233	DCW 2*213, 2*469, 2*53, 2*309, 2*181, 2*437
03B4	EA00EA02EA01EA03	234	DCW 2*117, 2*373, 2*245, 2*501, 2*13, 2*269
03C0	1A011A039A009A02	235	DCW 2*141, 2*397, 2*77, 2*333, 2*205, 2*461
03CC	5A005A025A015A03	236	DCW 2*45, 2*301, 2*173, 2*429, 2*109, 2*365
03D8	DA01DA033A003A02	237	DCW 2*237, 2*493, 2*29, 2*285, 2*157, 2*413
03E4	BA00BA02BA01BA03	238	DCW 2*93, 2*349, 2*221, 2*477, 2*61, 2*317
03F0	7A017A03FA00FA02	239	DCW 2*189, 2*445, 2*125, 2*381, 2*253, 2*509
03FC	0600060206010603	240	DCW 2*3, 2*259, 2*131, 2*387, 2*67, 2*323
0408	8601860346004602	241	DCW 2*195, 2*451, 2*35, 2*291, 2*163, 2*419
0414	C600C602C601C603	242	DCW 2*99, 2*355, 2*227, 2*483, 2*19, 2*275
0420	26012603A600A602	243	DCW 2*147, 2*403, 2*83, 2*339, 2*211, 2*467
042C	6600660266016603	244	DCW 2*51, 2*307, 2*179, 2*435, 2*115, 2*371
0438	E601E60316001602	245	DCW 2*243, 2*499, 2*11, 2*267, 2*139, 2*395
0444	9600960296019603	246	DCW 2*75, 2*331, 2*203, 2*459, 2*43, 2*299
0450	56015603D600D602	247	DCW 2*171, 2*427, 2*107, 2*363, 2*235, 2*491
045C	3600360236013603	248	DCW 2*27, 2*283, 2*155, 2*411, 2*91, 2*347
0468	B601B60376007602	249	DCW 2*219, 2*475, 2*59, 2*315, 2*187, 2*443
0474	F600F602F601F603	250	DCW 2*123, 2*379, 2*251, 2*507, 2*7, 2*263
0480	0E010E038E008E02	251	DCW 2*135, 2*391, 2*71, 2*327, 2*199, 2*455
048C	4E004E024E014E03	252	DCW 2*39, 2*295, 2*167, 2*423, 2*103, 2*359

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
	0498	CE01CE032E002E02	253	DCW 2*231, 2*487, 2*23, 2*279, 2*151, 2*407
	04A4	AE00AE02AE01AE03	254	DCW 2*87, 2*343, 2*215, 2*471, 2*55, 2*311
	0480	6E016E03EE00EE02	255	DCW 2*183, 2*439, 2*119, 2*375, 2*247, 2*503
	048C	1E001E021E011E03	256	DCW 2*15, 2*271, 2*143, 2*399, 2*79, 2*335
	04C8	9E019E035E005E02	257	DCW 2*207, 2*463, 2*47, 2*303, 2*175, 2*431
	04D4	DE00DE02DE01DE03	258	DCW 2*111, 2*367, 2*239, 2*495, 2*31, 2*287
	04E0	3E013E03BE00BE02	259	DCW 2*159, 2*415, 2*95, 2*351, 2*223, 2*479
	04EC	7E007E027E017E03	260	DCW 2*63, 2*319, 2*191, 2*447, 2*127, 2*383
	04F8	FE01FE03	261	DCW 2*255, 2*511
			262	
			263	
	04FC		264	WR:
			265	
	04FC	FF7FFD7FF57FE97F	266	DCW 32767, 32765, 32757, 32745, 32728, 32705
	0508	A67F867F617F377F	267	DCW 32678, 32646, 32609, 32567, 32521, 32469
	0514	9C7E5F7E1D7ED57D	268	DCW 32412, 32351, 32285, 32213, 32137, 32057
	0520	E37C887C297CC57B	269	DCW 31971, 31880, 31785, 31685, 31580, 31470
	052C	7C7A057A89790979	270	DCW 31356, 31237, 31113, 30985, 30852, 30714
	0538	6B77D8764176A575	271	DCW 30571, 30424, 30273, 30117, 29956, 29791
	0544	8573077354729D71	272	DCW 29621, 29447, 29268, 29085, 28898, 28706
	0550	5E6F966EC96DF86C	273	DCW 28510, 28310, 28105, 27896, 27683, 27466
	055C	6D6AB869A668BC67	274	DCW 27245, 27019, 26790, 26556, 26319, 26077
	0568	E864EE63F162F061	275	DCW 25832, 25582, 25329, 25072, 24811, 24547
	0574	D75EC75DB35C9C5B	276	DCW 24279, 24007, 23731, 23452, 23170, 22884
	0580	42581D57F555C954	277	DCW 22594, 22301, 22005, 21705, 21403, 21096
	058C	3351FB4FBF4E814D	278	DCW 20787, 20475, 20159, 19841, 19519, 19195
	0598	B44969481C47CD45	279	DCW 18868, 18537, 18204, 17869, 17530, 17189
	05A4	CE417340173FB83D	280	DCW 16846, 16499, 16151, 15800, 15446, 15090
	05B0	8C392438BA364D35	281	DCW 14732, 14372, 14010, 13645, 13279, 12910
	05BC	FB30872F112E992C	282	DCW 12539, 12167, 11793, 11417, 11039, 10659
	05C8	2628A8262825A623	283	DCW 10278, 9896, 9512, 9126, 8739, 8351
	05D4	1A1F931D0B1C821A	284	DCW 7962, 7571, 7179, 6786, 6393, 5998
	05E0	E2155514C8123A11	285	DCW 5602, 5205, 4808, 4410, 4011, 3612
	05EC	8C0CFB0A6A09D907	286	DCW 3212, 2811, 2410, 2009, 1608, 1206
	05F8	2403920100006EFE	287	DCW 804, 402, 0, -402, -804, -1206
	0604	B8F927F896F605F5	288	DCW -1608, -2009, -2410, -2811, -3212, -3612
	0610	55F0C6EE38EDABEB	289	DCW -4011, -4410, -4808, -5205, -5602, -5998
	061C	07E77EE5F5E36DE2	290	DCW -6393, -6786, -7179, -7571, -7962, -8351
	0628	DDDD5ADCD8DA58D9	291	DCW -8739, -9126, -9512, -9896, -10278, -10659
	0634	E1D467D3EFD179D0	292	DCW -11039, -11417, -11793, -12167, -12539, -12910
	0640	21CCB3CA46C9DC7	293	DCW -13279, -13645, -14010, -14372, -14732, -15090
	064C	AAC348C2E9C08DBF	294	DCW -15446, -15800, -16151, -16499, -16846, -17189
	0658	86BD33BAE48897B7	295	DCW -17530, -17869, -18204, -18537, -18868, -19195
	0664	C1B37FB241B10580	296	DCW -19519, -19841, -20159, -20475, -20787, -21096
	0670	65AC37AB0BAE3A8	297	DCW -21403, -21705, -22005, -22301, -22594, -22884
	067C	7EA564A44DA339A2	298	DCW -23170, -23452, -23731, -24007, -24279, -24547
	0688	159F109E0F9D129C	299	DCW -24811, -25072, -25329, -25582, -25832, -26077
	0694	319944985A977596	300	DCW -26319, -26556, -26790, -27019, -27245, -27466
	06A0	DD93089337926A91	301	DCW -27683, -27896, -28105, -28310, -28510, -28706
	06AC	1E8F638EAC8DF98C	302	DCW -28898, -29085, -29268, -29447, -29621, -29791
	06B8	FC8A588ABF892389	303	DCW -29956, -30117, -30273, -30424, -30571, -30714
	06C4	7C87F7867786FB85	304	DCW -30852, -30985, -31113, -31237, -31356, -31470
	06D0	A4843B84D7837883	305	DCW -31580, -31685, -31785, -31880, -31971, -32057
	06DC	77822B82E381A181	306	DCW -32137, -32213, -32285, -32351, -32412, -32469
	06E8	F780C9809F807A80	307	DCW -32521, -32567, -32609, -32646, -32678, -32705
	06F4	288017800B800380	308	DCW -32728, -32745, -32757, -32765, -32767, -32765
	0700	0B80178028803FB0	309	DCW -32757, -32745, -32728, -32705, -32678, -32646

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
	070C	9F80C980F7802B81	310	DCW -32607, -32567, -32521, -32469, -32412, -32351
	0718	E3812B827782C782	311	DCW -32285, -32213, -32137, -32057, -31971, -31880
	0724	D7833B84A4841285	312	DCW -31785, -31685, -31580, -31470, -31356, -31237
	0730	7786F7867C870688	313	DCW -31113, -30985, -30852, -30714, -30571, -30424
	073C	BF895B8AFC8AA18B	314	DCW -30273, -30117, -29956, -29791, -29621, -29447
	0748	AC80638E1E8FDE8F	315	DCW -29268, -29085, -28898, -28706, -28510, -28310
	0754	37920893DD93B694	316	DCW -28105, -27896, -27683, -27466, -27245, -27019
	0760	5A9744983199239A	317	DCW -26790, -26556, -26319, -26077, -25832, -25582
	076C	0F9D109E159F1DA0	318	DCW -25329, -25072, -24811, -24547, -24279, -24007
	0778	4DA364A47EA59CA6	319	DCW -23731, -23452, -23170, -22884, -22594, -22301
	0784	0BAA37AB65AC98AD	320	DCW -22005, -21705, -21403, -21096, -20787, -20475
	0790	41B17FB2C1B30585	321	DCW -20159, -19841, -19519, -19195, -18868, -18537
	079C	E488338A868BDBBC	322	DCW -18204, -17869, -17530, -17189, -16846, -16499
	07A8	E9C048C2AAC30EC5	323	DCW -16151, -15800, -15446, -15090, -14732, -14372
	07B4	46C9B3CA21CC92CD	324	DCW -14010, -13645, -13279, -12910, -12539, -12167
	07C0	EFD167D3E1D45DD6	325	DCW -11793, -11417, -11039, -10659, -10278, -9896
	07CC	D8DA5ADCD0D61DF	326	DCW -9512, -9126, -8739, -8351, -7962, -7571
	07D8	F5E37EE507E792E8	327	DCW -7179, -6786, -6393, -5998, -5602, -5205
	07E4	38EDC6EE55F0E4F1	328	DCW -4808, -4410, -4011, -3612, -3212, -2811
	07F0	96F627F8B8F94AFB	329	DCW -2410, -2009, -1608, -1206, -804, -402
	07FC	000092012403B604	330	DCW 0, 402, 804, 1206, 1608, 2009
	0808	6A09F80A8C0C1C0E	331	DCW 2410, 2811, 3212, 3612, 4011, 4410
	0814	C8125514E2156E17	332	DCW 4808, 5205, 5602, 5998, 6393, 6786
	0820	081C931D1A1F9F20	333	DCW 7179, 7571, 7962, 8351, 8739, 9126
	082C	2825A8262628A329	334	DCW 9512, 9896, 10278, 10659, 11039, 11417
	0838	112E872FFB306E32	335	DCW 11793, 12167, 12539, 12910, 13279, 13645
	0844	BA3624388C39F23A	336	DCW 14010, 14372, 14732, 15090, 15446, 15800
	0850	173F7340CE412543	337	DCW 16151, 16499, 16846, 17189, 17530, 17869
	085C	1C476948B449FB4A	338	DCW 18204, 18537, 18868, 19195, 19519, 19841
	0868	BF4EF84F33516852	339	DCW 20159, 20475, 20787, 21096, 21403, 21705
	0874	F5551D5742586459	340	DCW 22005, 22301, 22594, 22884, 23170, 23452
	0880	B35CC75DD75EE35F	341	DCW 23731, 24007, 24279, 24547, 24811, 25072
	088C	F162EE63E864DD65	342	DCW 25329, 25582, 25832, 26077, 26319, 26556
	0898	A6688B696D6A4A6B	343	DCW 26790, 27019, 27245, 27466, 27683, 27896
	08A4	C96D966E5E6F2270	344	DCW 28105, 28310, 28510, 28706, 28898, 29085
	08B0	54720773B5735F74	345	DCW 29268, 29447, 29621, 29791, 29956, 30117
	08BC	4176D8766B77FA77	346	DCW 30273, 30424, 30571, 30714, 30852, 30985
	08C8	8979057A7C7AEE7A	347	DCW 31113, 31237, 31356, 31470, 31580, 31685
	08D4	297C887CE37C397D	348	DCW 31785, 31880, 31971, 32057, 32137, 32213
	08E0	1D7E5F7E9C7ED57E	349	DCW 32285, 32351, 32412, 32469, 32521, 32567
	08EC	617F867FA67FC17F	350	DCW 32609, 32646, 32678, 32705, 32728, 32745
	08F8	F57FFD7FFF7F	351	DCW 32757, 32765, 32767
			352	
			353	
	08FE		354	WI:
			355	
	08FE	00006EFEDCFC4AFB	356	DCW 0, -402, -804, -1206, -1608, -2009
	090A	96F605F574F3E4F1	357	DCW -2410, -2811, -3212, -3612, -4011, -4410
	0916	38EDABEB1EEA92E8	358	DCW -4808, -5205, -5602, -5998, -6393, -6786
	0922	F5E36DE2E6E061DF	359	DCW -7179, -7571, -7962, -8351, -8739, -9126
	092E	D8DA58D9DAD75DD6	360	DCW -9512, -9896, -10278, -10659, -11039, -11417
	093A	EFD179D005CF92CD	361	DCW -11793, -12167, -12539, -12910, -13279, -13645
	0946	46C9DCC774C60EC5	362	DCW -14010, -14372, -14732, -15090, -15446, -15800
	0952	E9C08DBF32BEDB8C	363	DCW -16151, -16499, -16846, -17189, -17530, -17869
	095E	E48897B74CB605B5	364	DCW -18204, -18537, -18868, -19195, -19519, -19841
	096A	41B105B0CDAE98AD	365	DCW -20159, -20475, -20787, -21096, -21403, -21705
	0976	0BAAE3A8BEA79CA6	366	DCW -22005, -22301, -22594, -22884, -23170, -23452

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
	0982	4DA339A229A11DA0	367	DCW -23731, -24007, -24279, -24547, -24811, -25072
	098E	0F9D129C189B239A	368	DCW -25329, -25582, -25832, -26077, -26319, -26556
	099A	5A9775969395B694	369	DCW -26790, -27019, -27245, -27456, -27683, -27896
	09A6	37926A91A290DE8F	370	DCW -28105, -28310, -28510, -28706, -28898, -29085
	09B2	AC8DF98C4B8CA18B	371	DCW -29268, -29447, -29621, -29791, -29956, -30117
	09BE	BF89288995880688	372	DCW -30273, -30424, -30571, -30714, -30852, -30985
	09CA	7786FB8584851285	373	DCW -31113, -31237, -31356, -31470, -31580, -31685
	09D6	D78378831D83C782	374	DCW -31785, -31880, -31971, -32057, -32137, -32213
	09E2	E381A18164812B81	375	DCW -32285, -32351, -32412, -32469, -32521, -32567
	09EE	9F807A805A803F80	376	DCW -32609, -32646, -32678, -32705, -32728, -32745
	09FA	0B80038001800380	377	DCW -32757, -32765, -32767, -32765, -32757, -32745
	0A06	28803F805A807A80	378	DCW -32728, -32705, -32678, -32646, -32609, -32567
	0A12	F78028B16481A181	379	DCW -32521, -32469, -32412, -32351, -32285, -32213
	0A1E	7782C7821D837883	380	DCW -32137, -32057, -31971, -31880, -31785, -31685
	0A2A	A48412858485F885	381	DCW -31580, -31470, -31356, -31237, -31113, -30985
	0A36	7C87068895882889	382	DCW -30852, -30714, -30571, -30424, -30273, -30117
	0A42	FC8AA18B4B8CF98C	383	DCW -29956, -29791, -29621, -29447, -29268, -29085
	0A4E	1E8FDE8FA2906A91	384	DCW -28898, -28706, -28510, -28310, -28105, -27896
	0A5A	DD93869493957596	385	DCW -27683, -27466, -27245, -27019, -26790, -26556
	0A66	3199239A189B129C	386	DCW -26319, -26077, -25832, -25582, -25329, -25072
	0A72	159F1DA029A139A2	387	DCW -24811, -24547, -24279, -24007, -23731, -23452
	0A7E	7EA59CA6BEA7E3A8	388	DCW -23170, -22884, -22594, -22301, -22005, -21705
	0A8A	65AC98ADCD4E0580	389	DCW -21403, -21096, -20787, -20475, -20159, -19841
	0A96	C18305B54CB69787	390	DCW -19519, -19195, -18868, -18537, -18204, -17869
	0AA2	86B8DB8C32BE8DBF	391	DCW -17530, -17189, -16846, -16499, -16151, -15800
	0AAE	AAC30EC574C60CC7	392	DCW -15446, -15090, -14732, -14372, -14010, -13645
	0ABA	21CC92CD05CF79D0	393	DCW -13279, -12910, -12539, -12167, -11793, -11417
	0AC6	E1D45DD6AD758D9	394	DCW -11039, -10659, -10278, -9896, -9512, -9126
	0AD2	DDDD61DFE6E06DE2	395	DCW -8739, -8351, -7962, -7571, -7179, -6786
	0ADE	07E792E81EEAABEB	396	DCW -6393, -5998, -5602, -5205, -4808, -4410
	0AEA	55F0E4F174F305F5	397	DCW -4011, -3612, -3212, -2811, -2410, -2009
	0AF6	B8F94AFBDCFC6EFE	398	DCW -1608, -1206, -804, -402, 0, 402
	0B02	2403B6044806D907	399	DCW 804, 1206, 1608, 2009, 2410, 2811
	0B0E	8C0C1C05A80F3A11	400	DCW 3212, 3612, 4011, 4410, 4808, 5205
	0B1A	E2156E17F918821A	401	DCW 5602, 5998, 6393, 6786, 7179, 7571
	0B26	1A1F9F202322A623	402	DCW 7962, 8351, 8739, 9126, 9512, 9896
	0B32	2628A3291F2B992C	403	DCW 10278, 10659, 11039, 11417, 11793, 12167
	0B3E	F8306E32DF334D35	404	DCW 12539, 12910, 13279, 13645, 14010, 14372
	0B4A	8C39F23A563CB83D	405	DCW 14732, 15090, 15446, 15800, 16151, 16499
	0B56	CE4125437A44CD45	406	DCW 16846, 17189, 17530, 17869, 18204, 18537
	0B62	B449FB4A3F4C814D	407	DCW 18868, 19195, 19519, 19841, 20159, 20475
	0B6E	335168529853C954	408	DCW 20787, 21096, 21403, 21705, 22005, 22301
	0B7A	42586459825A9C5B	409	DCW 22594, 22884, 23170, 23452, 23731, 24007
	0B86	D75EE35FEB60F061	410	DCW 24279, 24547, 24811, 25072, 25329, 25582
	0B92	E864DD65CF668C67	411	DCW 25832, 26077, 26319, 26556, 26790, 27019
	0B9E	6D6A4A6B236CF86C	412	DCW 27245, 27466, 27683, 27896, 28105, 28310
	0BAA	5E6F2270E2709D71	413	DCW 28510, 28706, 28898, 29085, 29268, 29447
	0BB6	B5735F740475A575	414	DCW 29621, 29791, 29956, 30117, 30273, 30424
	0BC2	6B77FA7784780979	415	DCW 30571, 30714, 30852, 30985, 31113, 31237
	0BCE	7C7AE7A5C78C57B	416	DCW 31356, 31470, 31580, 31685, 31785, 31880
	0BDA	E37C397D897DD57D	417	DCW 31971, 32057, 32137, 32213, 32285, 32351
	0BE6	9C7ED57E097F377F	418	DCW 32412, 32469, 32521, 32567, 32609, 32646
	0BF2	A67FC17FD87FE97F	419	DCW 32678, 32705, 32728, 32745, 32757, 32765
	0BFE	FF7FD7FF57FE97F	420	DCW 32767, 32765, 32757, 32745, 32728, 32705
	0COA	A67F867F617F377F	421	DCW 32678, 32646, 32609, 32567, 32521, 32469
	0C16	9C7E5F7E1D7ED57D	422	DCW 32412, 32351, 32285, 32213, 32137, 32057
	0C22	E37C887C297CC57B	423	DCW 31971, 31880, 31785, 31685, 31580, 31470

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
	0C2E	7C7A057A89790979	424	DCW 31356, 31237, 31113, 30985, 30852, 30714
	0C3A	6877D8764176A575	425	DCW 30571, 30424, 30273, 30117, 29956, 29791
	0C46	B573077354729D71	426	DCW 29621, 29447, 29268, 29085, 28898, 28706
	0C52	5E6F966EC96DF86C	427	DCW 28510, 28310, 28105, 27896, 27683, 27466
	0C5E	6D6A8869A668BC67	428	DCW 27245, 27019, 26790, 26556, 26319, 26077
	0C6A	E864EE63F162F061	429	DCW 25832, 25582, 25329, 25072, 24811, 24547
	0C76	D75EC75DB35C9C5B	430	DCW 24279, 24007, 23731, 23452, 23170, 22884
	0C82	42581D57F555C954	431	DCW 22594, 22301, 22005, 21705, 21403, 21096
	0C8E	3351FB4FBF4E814D	432	DCW 20787, 20475, 20159, 19841, 19519, 19195
	0C9A	B34969481C47CD45	433	DCW 18867, 18537, 18204, 17869, 17530, 17189
	0CA6	CE417340173FB83D	434	DCW 16846, 16499, 16151, 15800, 15446, 15090
	0CB2	8C392438BA364D35	435	DCW 14732, 14372, 14010, 13645, 13279, 12910
	0CBE	FB30872F112E992C	436	DCW 12539, 12167, 11793, 11417, 11039, 10659
	0CCA	2628A8262825A623	437	DCW 10278, 9896, 9512, 9126, 8739, 8351
	0CD6	1A1F931D081C821A	438	DCW 7962, 7571, 7179, 6786, 6393, 5998
	0CE2	E2155514C8123A11	439	DCW 5602, 5205, 4808, 4410, 4011, 3612
	0CEE	8C0CF80A6A09D907	440	DCW 3212, 2811, 2410, 2009, 1608, 1206
	0CFA	240392010000	441	DCW 804, 402, 0
			442	
	0D00		443	END

SYMBOL TABLE LISTING

N A M E	VALUE	ATTRIBUTES
BREV.	00FCH	CODE REL WORD
ENDL.	00F1H	CODE REL ENTRY
ERR1.	00B1H	CODE REL ENTRY
ERR2.	00B5H	CODE REL ENTRY
ERROR	-----	REG EXTERNAL
FFT_F0.	-----	MODULE STACKSIZE(6)
FFT_FOR	0000H	CODE REL PUBLIC ENTRY
GM.	0038H	CODE REL ENTRY
GR2	006CH	CODE REL ENTRY
GW.	002AH	CODE REL ENTRY
GX.	00B7H	CODE REL ENTRY
GX2	0075H	CODE REL ENTRY
IK.	0092H	CODE REL ENTRY
IN_CNT.	0042H	OVERLAY ABS WORD
IN_LOOP	0018H	CODE REL ENTRY
KN2	0048H	OVERLAY ABS WORD
KPTR.	0046H	OVERLAY ABS WORD
LOOP_CNT.	0051H	OVERLAY ABS BYTE
MID_LOOP.	0016H	CODE REL ENTRY
N_SUB_K	004AH	OVERLAY ABS WORD
NDIV2	0044H	OVERLAY ABS WORD
OUT_LOOP.	000DH	CODE REL ENTRY
PWR	0040H	OVERLAY ABS WORD
RK.	004CH	OVERLAY ABS WORD
RNK	004EH	OVERLAY ABS WORD
SHFT_CNT.	0050H	OVERLAY ABS BYTE
TMPI.	0028H	OVERLAY ABS LONG
TMPI1	0030H	OVERLAY ABS LONG
TMPR.	0024H	OVERLAY ABS LONG
TMPR1	002CH	OVERLAY ABS LONG
UN_LOOP	00BFH	CODE REL ENTRY
UNWEAVE	00B9H	CODE REL ENTRY
WI.	08FEH	CODE REL WORD
WIP	003EH	OVERLAY ABS WORD
WR.	04FCH	CODE REL WORD
WRP	003CH	OVERLAY ABS WORD
XIMAG	-----	DATA EXTERNAL
XITMP	0038H	OVERLAY ABS LONG
XREAL	-----	DATA EXTERNAL
XRTMP	0034H	OVERLAY ABS LONG

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

DOS 3.30 (038-N) MCS-96 MACRO ASSEMBLER, V1.2

SOURCE FILE: E_INT.A96

OBJECT FILE: E_INT.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: <none>

```

ERR LOC  OBJECT                LINE    SOURCE STATEMENT
                                1      $PAGELENGTH(51)
                                2      $TITLE(" ENABLE 80C196 GLOBAL INTERRUPTS")
                                3
                                4      E_INT          MODULE  STACKSIZE(12)
                                5                          PUBLIC  ENAB_INT
                                6
                                7      ;      E_INT.A96
                                8      ;
                                9      ;      Version 1.0          July 26, 1989
                               10      ;
                               11      ;      Jeffrey C. Sigl
                               12      ;
                               13      ;      GMS Engineering Corporation
                               14      ;      8940-D Route 108
                               15      ;      Columbia, Maryland 21045
                               16      ;
                               17      ;
                               18      ; =====
                               19      ;
                               20      ;      COMMON DEFINITIONS
                               21
                               22      $INCLUDE(8096.INC)          ;80C196 REGISTER DEFINITIONS
                                = 1                                2 3
;*****
                                =1 24      ;
                                =1 25      ; 8096.INC - DEFINITION OF SYMBOLIC NAMES FOR THE I/O REGISTERS OF THE
                                =1 26      ;      8096 AND THE 80C196
                                =1 27      ;      (C) INTEL CORPORATION 1983
                                = 1                                2 8
;*****
                                =1 29      ;
                                =1 30      ;/*
                                =1 31      ; *      8096 SFR's
                                =1 32      ; */
0000                                =1 33      RO          EQU    00H:WORD      ; R      ZERO REGISTER
0002                                =1 34      AD_COMMAND  EQU    02H:BYTE      ; W
0002                                =1 35      AD_RESULT_LO EQU    02H:BYTE      ; R
0003                                =1 36      AD_RESULT_HI EQU    03H:BYTE      ; R
0003                                =1 37      HSI_MODE     EQU    03H:BYTE      ; W
0004                                =1 38      HSO_TIME     EQU    04H:WORD      ; W
0004                                =1 39      HSI_TIME     EQU    04H:WORD      ; R
0006                                =1 40      HSO_COMMAND  EQU    06H:BYTE      ; W
0006                                =1 41      HSI_STATUS   EQU    06H:BYTE      ; R
0007                                =1 42      SBUF         EQU    07H:BYTE      ; R/W

```

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
	0008		=1 43	INT_MASK EQU 08H:BYTE ; R/W
	0009		=1 44	INT_PENDING EQU 09H:BYTE ; R/W
	000A		=1 45	WATCHDOG EQU 0AH:BYTE ; W WATCHDOG TIMER
	000A		=1 46	TIMER1 EQU 0AH:WORD ; R
	000C		=1 47	TIMER2 EQU 0CH:WORD ; R
	000E		=1 48	BAUD_RATE EQU 0EH:BYTE ; W
	000E		=1 49	IOPORT0 EQU 0EH:BYTE ; R
	000F		=1 50	IOPORT1 EQU 0FH:BYTE ; R/W
	0010		=1 51	IOPORT2 EQU 10H:BYTE ; R/W
	0011		=1 52	SP_CON EQU 11H:BYTE ; W
	0011		=1 53	SP_STAT EQU 11H:BYTE ; R
	0015		=1 54	IOC0 EQU 15H:BYTE ; W
	0015		=1 55	IOS0 EQU 15H:BYTE ; R
	0016		=1 56	IOC1 EQU 16H:BYTE ; W
	0016		=1 57	IOS1 EQU 16H:BYTE ; R
	0017		=1 58	PWM_CONTROL EQU 17H:BYTE ; W
	0018		=1 59	SP EQU 18H:WORD ; R/W
			=1 60	;
			=1 61	;/*
			=1 62	; * 80C196 SFR's
			=1 63	; */
	000B		=1 64	IOC2 EQU 0BH:BYTE ; W
			=1 65	;TIMER2 EQU 0CH:WORD ; R/W
	0012		=1 66	IPEND1 EQU 12H:BYTE ; R/W
	0013		=1 67	IMASK1 EQU 13H:BYTE ; R/W
	0014		=1 68	WSR EQU 14H:BYTE ; R/W
	0017		=1 69	IOS2 EQU 17H:BYTE ; R
			70	
			71	; =====
			72	;
			73	; CODE SEGMENT
			74	
	0000		75	CSEG
			76	
	0000		77	ENAB_INT:
			78	
	0000 FB		79	EI ;ENABLE
INTRPTS				
0001 F0			80	RET
0002			81	END

SYMBOL TABLE LISTING

N A M E	VALUE	ATTRIBUTES
AD_COMMAND.	0002H	NULL ABS BYTE
AD_RESULT_HI.	0003H	NULL ABS BYTE
AD_RESULT_LO.	0002H	NULL ABS BYTE
BAUD_RATE.	000EH	NULL ABS BYTE
E_INT.	-----	MODULE STACKSIZE(12)
ENAB_INT.	0000H	CODE REL PUBLIC ENTRY
HSI_MODE.	0003H	NULL ABS BYTE
HSI_STATUS.	0006H	NULL ABS BYTE
HSI_TIME.	0004H	NULL ABS WORD
HSD_COMMAND.	0006H	NULL ABS BYTE
HSD_TIME.	0004H	NULL ABS WORD
IMASK1.	0013H	NULL ABS BYTE
INT_MASK.	0008H	NULL ABS BYTE
INT_PENDING.	0009H	NULL ABS BYTE
IOC0.	0015H	NULL ABS BYTE
IOC1.	0016H	NULL ABS BYTE
IOC2.	000BH	NULL ABS BYTE
IOPORT0.	000EH	NULL ABS BYTE
IOPORT1.	000FH	NULL ABS BYTE
IOPORT2.	0010H	NULL ABS BYTE
IOS0.	0015H	NULL ABS BYTE
IOS1.	0016H	NULL ABS BYTE
IOS2.	0017H	NULL ABS BYTE
IPEND1.	0012H	NULL ABS BYTE
PWM_CONTROL.	0017H	NULL ABS BYTE
RO.	0000H	NULL ABS WORD
SBUF.	0007H	NULL ABS BYTE
SP.	0018H	NULL ABS WORD
SP_CON.	0011H	NULL ABS BYTE
SP_STAT.	0011H	NULL ABS BYTE
TIMER1.	000AH	NULL ABS WORD
TIMER2.	000CH	NULL ABS WORD
WATCHDOG.	000AH	NULL ABS BYTE
WSR.	0014H	NULL ABS BYTE

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

DOS 3.30 (038-N) MCS-96 MACRO ASSEMBLER, V1.2

SOURCE FILE: STATUS.A96

OBJECT FILE: STATUS.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: <none>

ERR LOC	OBJECT	LINE	SOURCE STATEMENT
		1	public status_temp
0000		2	rseg
0000		3	status_temp: DSB 1 ;Global status register
0001		4	end

SYMBOL TABLE LISTING

N A M E	VALUE	ATTRIBUTES
STATUS_TEMP	0000H	REG REL PUBLIC BYTE

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

DOS 3.30 (038-N) MCS-96 MACRO ASSEMBLER, V1.2

SOURCE FILE: GETCHAR.A96

OBJECT FILE: GETCHAR.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: <none>

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			1	\$debug
			2	\$nolist include (8096.inc)
			51	
	001C		52	tmp0 equ 1CH:word
	0006		53	RI_pos equ 06H:byte
	00BF		54	RI_mask equ 0BFH:byte
			55	
			56	extrn status_temp
			57	public getchar
			58	
	0000		59	CSEG
			60	
	0000 901100	E	61	getchar: orb status_temp, SP_STAT
	0003 3600FA	E	62	jbc status_temp, RI_pos, getchar
	0006 B0071C		63	ldb tmp0, sbuf
	0009 718F00	E	64	andb status_temp, #RI_mask
	000C F0		65	ret
	000D		66	end

SYMBOL TABLE LISTING

N A M E	VALUE	ATTRIBUTES
AD_COMMAND.	0002H	NULL ABS BYTE
AD_RESULT_HI.	0003H	NULL ABS BYTE
AD_RESULT_LO.	0002H	NULL ABS BYTE
BAUD_RATE.	000EH	NULL ABS BYTE
GETCHAR.	0000H	CODE REL PUBLIC ENTRY
HSI_MODE.	0003H	NULL ABS BYTE
HSI_STATUS.	0006H	NULL ABS BYTE
HSI_TIME.	0004H	NULL ABS WORD
HSO_COMMAND.	0006H	NULL ABS BYTE
HSO_TIME.	0004H	NULL ABS WORD
IMASK1.	0013H	NULL ABS BYTE
INT_MASK.	0008H	NULL ABS BYTE
INT_PENDING.	0009H	NULL ABS BYTE
IOC0.	0015H	NULL ABS BYTE
IOC1.	0016H	NULL ABS BYTE
IOC2.	000BH	NULL ABS BYTE
IOPORT0.	000EH	NULL ABS BYTE
IOPORT1.	000FH	NULL ABS BYTE
IOPORT2.	0010H	NULL ABS BYTE
IOS0.	0015H	NULL ABS BYTE
IOS1.	0016H	NULL ABS BYTE
IOS2.	0017H	NULL ABS BYTE
IPEND1.	0012H	NULL ABS BYTE
PWM_CONTROL.	0017H	NULL ABS BYTE
RO.	0000H	NULL ABS WORD
RI_MASK.	00BFH	NULL ABS BYTE
RI_POS.	0006H	NULL ABS BYTE
SBUF.	0007H	NULL ABS BYTE
.	0018H	NULL ABS WORD
.	0011H	NULL ABS BYTE
SP.	0011H	NULL ABS BYTE
STATU. MP.	-----	NULL EXTERNAL
TIMER1.	000AH	NULL ABS WORD
TIMER2.	000CH	NULL ABS WORD
TMP0.	001CH	NULL ABS WORD
WATCHDOG.	000AH	NULL ABS BYTE
WSR.	0014H	NULL ABS BYTE

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

DOS 3.30 (038-N) MCS-96 MACRO ASSEMBLER, V1.2

SOURCE FILE: PUTCHAR.A96

OBJECT FILE: PUTCHAR.OBJ

CONTROLS SPECIFIED IN INVOCATION COMMAND: <none>

ERR	LOC	OBJECT	LINE	SOURCE STATEMENT
			1	\$debug
			2	\$nolist include (8096.inc)
			51	
	0005		52	TI_pos equ 05H:byte
	000F		53	TI_mask equ 0DFH:byte
			54	
			55	extrn status_temp
			56	public putchar
			57	
	0000		58	CSEG
			59	
	0000 901100	E	60	putchar: orb status_temp, SP_STAT
	0003 3500FA	E	61	jbc status_temp, TI_pos, putchar
	0006 83180207		62	ldb sbuf, 2[sp]
	000A 71DF00	E	63	andb status_temp, #TI_mask
	000D F0		64	ret
	000E		65	end

SYMBOL TABLE LISTING

N A M E	VALUE	ATTRIBUTES
AD_COMMAND.	0002H	NULL ABS BYTE
AD_RESULT_HI.	0003H	NULL ABS BYTE
AD_RESULT_LO.	0002H	NULL ABS BYTE
BAUD_RATE	000EH	NULL ABS BYTE
HSI_MODE.	0003H	NULL ABS BYTE
HSI_STATUS.	0006H	NULL ABS BYTE
HSI_TIME.	0004H	NULL ABS WORD
HSO_COMMAND	0006H	NULL ABS BYTE
HSO_TIME.	0004H	NULL ABS WORD
IMASK1.	0013H	NULL ABS BYTE
INT_MASK.	0008H	NULL ABS BYTE
INT_PENDING	0009H	NULL ABS BYTE
IOC0.	0015H	NULL ABS BYTE
IOC1.	0016H	NULL ABS BYTE
IOC2.	000BH	NULL ABS BYTE
IOPORT0	000EH	NULL ABS BYTE
IOPORT1	000FH	NULL ABS BYTE
IOPORT2	0010H	NULL ABS BYTE
IOS0.	0015H	NULL ABS BYTE
IOS1.	0016H	NULL ABS BYTE
IOS2.	0017H	NULL ABS BYTE
IPEND1.	0012H	NULL ABS BYTE
PUTCHAR	0000H	CODE REL PUBLIC ENTRY
PWM_CONTROL	0017H	NULL ABS BYTE
RO.	0000H	NULL ABS WORD
SBUF.	0007H	NULL ABS BYTE
SP.	0018H	NULL ABS WORD
SP_CON.	0011H	NULL ABS BYTE
SP_STAT	0011H	NULL ABS BYTE
STATUS_TEMP	-----	NULL EXTERNAL
TI_MASK	00DFH	NULL ABS BYTE
TI_POS.	0005H	NULL ABS BYTE
TIMER1.	000AH	NULL ABS WORD
TIMER2.	000CH	NULL ABS WORD
WATCHDOG.	000AH	NULL ABS BYTE
WSR	0014H	NULL ABS BYTE

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.

APPENDIX B: PC SOFTWARE LISTING

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```
1 $declare
2 c      shell.for
3
4 c      Driver for the EEG artifact correction system.
5
6
7 c      Created: December 19, 1989
8 c      Last Update:      February 14, 1989
9
10 c      Steven M. Falk
11 c      Jeffrey C. Sigl
12 c      GMS Engineering Corporation
13 c      8940-D Route 108
14 c      Columbia, MD 21045
15 c      (301) 995-0508
16
17      program shell
18
19 c Data Structures
20
21      character*6      word
22      character*1      chr(6), cls(4), capps, cappi, cr, cappr, cappz
23      character*1      bell, resp
24      integer          i, j
25      integer*4        tmpvb, tmpvbb, tmpva
26      integer          step, iresp, ichanl, ichann
27      real*4           d1v, d2v, d1h, d2h
28      real*4           d3v, d4v, d3h, d4h
29      real*4           vmag, hmag, gain, ccmag
30      real*4           tmpvbl(11)
31
32 c Functions
33
34      integer          ichar
35      integer*4        int4
36
37 c Data Relations
38
39      equivalence      (word, chr)
40
41 c Data Initialization
42
43      data word /6H      /
44      data capps, cappi, cappr, cappz /'S', 'I', 'R', 'A'/
45      data ichanl, ichann /1, 0/
46      cls(1) = 8#33
47      cls(2) = 8#133
48      cls(3) = 8#62
49      cls(4) = 8#112
50      bell = 8#7
51
52      step = 0
53
54      d1v = 0.
55      d2v = 0.
56      d1h = 0.
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```
57      d2h = 0.
58      d3v = 0.
59      d4v = 0.
60      d3h = 0.
61      d4h = 0.
62
63
64      c Clear the screen
65
66      2      write(*,3)(cls(i),i=1,4)
67      3      format(' ',4a1)
68
69      open(11,file='drive',status='old')
70      66      read(11,67)resp
71      67      format(a1)
72      if ( resp .ne. cappz ) then
73          goto 66
74      endif
75
76      call dcalc(cls,d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,step,bell,ccmax)
77
78      call sgm(vmag,0)
79      call sgm(hmag,1)
80      call sgm(gain,2)
81
82      call ftune(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,ccmax)
83
84      close(11)
85
86      if ( gain .eq. -9999. ) then
87          goto 1357
88      endif
89
90      open(12,file='correct',status='new')
91
92      tmpvbl(1) = vmag
93      tmpvbl(2) = hmag
94      tmpvbl(3) = gain
95      tmpvbl(4) = d1v
96      tmpvbl(5) = d2v
97      tmpvbl(6) = d3v
98      tmpvbl(7) = d4v
99      tmpvbl(8) = d1h
100     tmpvbl(9) = d2h
101     tmpvbl(10) = d3h
102     tmpvbl(11) = d4h
103     do 152 i=1,11
104         tmpva = int4(tmpvbl(i))
105         tmpvb = tmpva * 10
106         if ( tmpva .lt. 0 ) then
107             tmpvb = -tmpvb
108             tmpvb = tmpvb + 1
109         endif
110         write(12,154)tmpvb
111     154     format(i6)
112     152     continue
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

113
114      close(12)
115
116      write(*,1356)
117 1356      format(' Correction Matrix now computed for the selected channel.'
118      + /' EARS Correction option can now be run (main menu option C).')
119
120 1357      continue
121
122      stop
123      end

```

main Local Symbols

Name	Class	Type	Size	Offset
D4H	local	REAL*4	4	0002
IRESP	local	INTEGER*4	4	0006
TMPVA	local	INTEGER*4	4	000a
WORD.	local	CHAR*6	6	000c
TMPVB	local	INTEGER*4	4	000e
CAPPS	local	CHAR*1	1	0012
I	local	INTEGER*4	4	0012
CAPPI	local	CHAR*1	1	0013
CAPPR	local	CHAR*1	1	0014
CAPPZ	local	CHAR*1	1	0015
J	local	INTEGER*4	4	0016
ICHANL.	local	INTEGER*4	4	0016
ICHANN.	local	INTEGER*4	4	001a
D1V	local	REAL*4	4	001a
TMPVBB.	local	INTEGER*4	4	001e
D2V	local	REAL*4	4	0022
D3V	local	REAL*4	4	0026
D4V	local	REAL*4	4	002a
TMPVBL.	local	REAL*4	44	002e
CR.	local	CHAR*1	1	005a
HMAG.	local	REAL*4	4	005c
GAIN.	local	REAL*4	4	0060
BELL.	local	CHAR*1	1	0064
CLS	local	CHAR*1	4	0066
ICHAR	local	INTEGER*4	4	006a
VMAG.	local	REAL*4	4	006e
CCMAX	local	REAL*4	4	0072
RESP.	local	CHAR*1	1	0076
STEP.	local	INTEGER*4	4	0078
D1H	local	REAL*4	4	007c
D2H	local	REAL*4	4	0080
D3H	local	REAL*4	4	0084
CHR	equiv	CHAR*1	6	000c

PAGE 4
03-22-90
17:43:47

Microsoft FORTRAN Optimizing Compiler Version 4.00

Global Symbols

Name	Class	Type	Size	Offset
DCALC	extern	***	***	***
FTUNE	extern	***	***	***
SGM	extern	***	***	***
main.	FSUBRT	***	***	0000

Code size = 03a8 (936)

Data size = 006b (107)

Bss size = 0088 (136)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```
1 $storage:2
2 $declare
3 c      dcalc.for
4 c
5 c      a program to compute the D's for the EEG correction algorithm
6 c
7 c      October 12, 1989
8 c
9 c      Jeffrey C. Sigl
10
11      subroutine dcalc(cls,d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,step,bell,
12      +ccmax)
13
14 c Data Structures
15
16      character*1      cls(4)
17      character*1      bell, resp
18      integer          i, j, n, zline, ezline, step, iresp
19      integer          step1
20      real*4           d1v, d2v, d1h, d2h
21      real*4           d3v, d4v, d3h, d4h, ccmax
22      double precision z, h, phi, psi, pi2, capl, raddeg
23      double precision betal, betar, gamal, gamar
24      double precision theta, r, sigma, eta
25      double precision betalv, betarv, gamalv, gamarv
26      double precision thetav, rv, sigmav, etav
27      double precision betalh, betarh, gamalh, gamarh
28      double precision thetah, rh, sigmah, etah
29      double precision denomv, denomh, temp, temp2, term1, term2
30      double precision delta, jeff, negflag, max, scale
31
32 c Functions
33
34      integer          ichar
35      double precision dacos, dsin, dcos, dsqrt, dabs
36
37
38 c Data Initialization
39
40      pi2 = 1.5707963268000
41      raddeg = 57.295779513100
42
43      n = 1
44      z = 111.
45      h = 70.
46      capl = 8.
47      betalv = 33.
48      betarv = 94.
49      betalh = 49.
50      betarh = 144.
51      zline = 0
52      betal = 208.
53      betar = 211.
54      ccmax = 0.05
55      ezline = 0
56
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

57 c Clear the screen
58
59 2      write(*,3)(cls(i),i=1,4)
60 3      format(' ',4a1)
61
62 c Write the Main Menu
63
64      write(*,50)z,h,capl,betalv,betarv,betalh,betarh,zline,betal,
65      +betar,camax
66 50      format(/20x,' GMS Engineering EEG-EOG Artifact Removal'///,
67      +20x,'          Parameter Menu'////,
68      +' 1',5x,'Distance between the eyes (mm).....',
69      +f9.2,/,
70      +' 2',5x,'Distance from the origin to the stim electrode (mm)',
71      +f9.2,/,
72      +' 3',5x,'Corneo-retinal distance (mm).....',
73      +f9.2,/,
74      +' 4',5x,'Distance from Left Eye to VU-EOG electrode (mm)....',
75      +f9.2,/,
76      +' 5',5x,'Distance from Right Eye to VU-EOG electrode (mm)...',
77      +f9.2,/,
78      +' 6',5x,'Distance from Left Eye to H-EOG electrode (mm).....',
79      +f9.2,/,
80      +' 7',5x,'Distance from Right Eye to H-EOG electrode (mm)....',
81      +f9.2,/,
82      +' 8',5x,'H-EOG electrode above(0)/below(1) the "Z" line?.... ',
83      +i5,/,
84      +' 9',5x,'Distance from Left Eye to EEG electrode (mm).....',
85      +f9.2,/,
86      +' 10',5x,'Distance from Right Eye to EEG electrode (mm).....',
87      +f9.2,/,
88      +' 11',5x,'Maximum cross-correlation function for correction..',
89      +f9.2,/,
90      +' 12',5x,'Physical data entered; compute Correction Matrix'/,
91      +/12x,' Enter Response      >'\\)
92
93
94 c Read the response & take appropriate action
95
96      read(*,'(i5)')iresp
97
98      if (iresp .eq. 1) then
99          write(*,110)
100          read(*,'(f12.5)')z
101          if ( z .lt. 0. ) then
102              z = 0.
103              call error(bell)
104          endif
105
106      elseif (iresp .eq. 2) then
107          write(*,110)
108          read(*,'(f12.5)')h
109          if ( h .lt. 0. ) then
110              h = 0.
111              call error(bell)
112          endif

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```
113
114     elseif (iresp .eq. 3) then
115         write(*,110)
116 110     format(/13x,'Enter new value >'\)
117         read(*,'(f12.5)')capl
118         if ( capl .lt. 0. ) then
119             capl = 8.
120             call error(bell)
121         endif
122
123     elseif (iresp .eq. 4) then
124         write(*,110)
125         read(*,'(f12.5)')betalv
126         if ( betalv .lt. 0. ) then
127             betalv = 0.
128             call error(bell)
129         endif
130
131     elseif (iresp .eq. 5) then
132         write(*,110)
133         read(*,'(f12.5)')betarv
134         if ( betarv .lt. 0. ) then
135             betarv = 0.
136             call error(bell)
137         endif
138
139     elseif (iresp .eq. 6) then
140         write(*,110)
141         read(*,'(f12.5)')betalh
142         if ( betalh .lt. 0. ) then
143             betalh = 0.
144             call error(bell)
145         endif
146
147     elseif (iresp .eq. 7) then
148         write(*,110)
149         read(*,'(f12.5)')betarh
150         if ( betarh .lt. 0. ) then
151             betarh = 0.
152             call error(bell)
153         endif
154
155     elseif (iresp .eq. 8) then
156         write(*,110)
157         read(*,'(i5)')zline
158         if ( (zline .ne. 0) .and. (zline .ne. 1)) then
159             zline = 0
160             call error(bell)
161         endif
162
163     elseif (iresp .eq. 9) then
164         write(*,110)
165         read(*,'(f12.5)')betal
166         if ( betal .lt. 0. ) then
167             betal = 0.
168             call error(bell)
```

```

Line# Source Line      Microsoft FORTRAN Optimizing Compiler Version 4.00

169          endif
170
171      elseif (iresp .eq. 10) then
172          write(*,110)
173          read(*,'(f12.5)')betar
174          if ( betar .lt. 0. ) then
175              betar = 0.
176              call error(bell)
177          endif
178
179      elseif (iresp .eq. 11) then
180          write(*,110)
181          read(*,'(f12.5)')ccmax
182          if ( ccmax .lt. 0. ) then
183              ccmax = 0.
184              call error(bell)
185          endif
186
187      elseif (iresp .eq. 12) then
188          if ((betal.eq.0.).or.(betar.eq.0.)) then
189              write(*,155)bell
190 155          format(//,12x,'EEG electrode distances',
191 +              ' must be entered! '//,a1,12x,
192 +              'Type ENTER to continue...')
193              read(*,'(bn,a1)')resp
194              goto 2
195          endif
196          goto 98
197
198      else
199          call error(bell)
200
201      endif
202
203      goto 2
204
205
206 c Check if the VEOG electrode is below the stim electrode;
207 c     if so, negate the Dv's
208
209 98      delta = dacos( (z**2 - betalv**2 + betarv**2)
210 #          / (2.*z*betarv) )
211      if ( (betarv*dsin(delta)) .lt. h ) then
212          negflag = -1.000
213 c          write(*,170)
214 c170          format('/ The EOGv electrode is below the STIM electrode.')
215      else
216          negflag = 1.000
217 c          write(*,172)
218 c172          format(' The EOGv electrode is above the STIM electrode.')
219      endif
220
221 c Compute Vertical EOG parameters
222
223 c          write(*,23)
224 c23          format('/ EOGV')

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

225
226      rv = dsqrt( 0.5 * ( betalv**2 + betarv**2 - ((z**2)/2.) ) )
227 c      write(*,200)rv
228 c200  format(' r ',f16.8)
229
230      temp = (rv**2 + (z/2)**2 - betarv**2) / (rv * z)
231      if ( temp .gt. 1. ) then
232          temp = 1.0
233      elseif ( temp .lt. -1. ) then
234          temp = -1.0
235      endif
236      thetav = dacos( temp )
237 c      write(*,210)thetav*raddeg
238 c210  format(' theta ',f16.8,' degrees')
239
240      gamalv = ( rv/betalv ) * dsin( thetav )
241 c      write(*,230)gamalv
242 c230  format(' gamal ',f16.8)
243
244      gamarv = ( rv/betarv ) * dsin( thetav )
245 c      write(*,240)gamarv
246 c240  format(' gamar ',f16.8)
247
248      sigmav = dsqrt( rv**2 + h**2 - 2.*rv*h*dsin(thetav) )
249 c      write(*,250)sigmav
250 c250  format(' sigma ',f16.8)
251
252      etav = (-rv/sigmav) * dcos( thetav )
253 c      write(*,260)etav
254 c260  format(' eta ',f16.8)
255
256 c Compute Horizontal EOG parameters
257
258 c      write(*,24)
259 c24   format('/ EOGH')
260
261      rh = dsqrt( 0.5 * ( betalh**2 + betarh**2 - ((z**2)/2.) ) )
262 c      write(*,200)rh
263
264      temp = (rh**2 + (z/2)**2 - betarh**2) / (rh * z)
265      if ( temp .gt. 1. ) then
266          temp = 1.0
267      elseif ( temp .lt. -1. ) then
268          temp = -1.0
269      endif
270      thetah = dacos( temp )
271      if ( zline .eq. 1 ) thetah = -thetah
272 c      write(*,210)thetah*raddeg
273
274      gamalh = ( rh/betalh ) * dsin( thetah )
275 c      write(*,230)gamalh
276
277      gamarh = ( rh/betarh ) * dsin( thetah )
278 c      write(*,240)gamarh
279
280      sigmah = dsqrt( rh**2 + h**2 - 2.*rh*h*dsin(thetah) )

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

281 c      write(*,250)sigmah
282
283      etah = (-rh/sigmah) * dcos( thetah )
284 c      write(*,260)etah
285
286
287 c Compute EEG electrode parameters
288
289 998      do 1000 i = 1, n
290
291          r = dsqrt(0.5*(betal**2 + betar**2 - ((z**2)/2.)))
292 c          write(*,200)r
293
294          temp = ( r**2 + (z/2)**2 - betar**2 ) / ( r * z )
295          if ( temp .gt. 1. ) then
296              temp = 1.0
297          elseif ( temp .lt. -1. ) then
298              temp = -1.0
299          endif
300          theta = dacos( temp )
301          if ( ezline .eq. 1 ) theta = -theta
302 c          write(*,210)theta*raddeg
303
304          gamal = ( r/betal ) * dsin( theta )
305          gamar = ( r/betar ) * dsin( theta )
306 c          write(*,230)gamal
307 c          write(*,240)gamar
308
309          sigma = dsqrt( r**2 + h**2 - 2.*r*h*dsin(theta) )
310          eta = (-r/sigma) * dcos( theta )
311 c          write(*,250)sigma
312 c          write(*,260)eta
313
314
315 c Compute the demoninators ( V & H )
316
317          psi = 0.
318          temp = (sigma**(-2)) * ( -(dsqrt( 1.-(eta**2) ) ) )
319          temp2 = (sigmah**(-2)) * ( -(dsqrt( 1.-(etav**2) ) ) )
320          denomv = temp / temp2
321
322 c          write(*,1212)denomv
323 c1212      format(' denomv = ',f16.8)
324
325          psi = pi2
326          temp = (sigma**(-2)) * eta
327          temp2 = (sigmah**(-2)) * etah
328          denomh = temp / temp2
329
330 c          write(*,1213)denomh
331 c1213      format(' denomh = ',f16.8)
332
333 c D1Vi
334
335          phi = 0.
336          term1 = (betal**(-2)) * gamal
          term2 = (betar**(-2)) * gamar

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

337      d1v = 0.5 * negflag * ( term1 + term2 ) / denomv
338
339      c D1Hi
340      phi = pi2
341      term1 = (betal**(-2)) * (-dsqrt(1.-(gamal**2)))
342      term2 = (betar**(-2)) * (dsqrt(1.-(gamar**2)))
343      d1h = 0.5 * ( term1 + term2 ) / denomh
344
345      c D2Vi
346      d2v = ( (1./betal) + (1./betar) ) / capl
347      d2v = ( d2v / denomv )
348
349      c D2Hi
350      d2h = ( (1./betal) + (1./betar) ) / capl
351      d2h = d2h / denomh
352
353      1000 continue
354
355
356      c D3V
357      phi = 0.
358      term1 = (betalv**(-2)) * gamalv
359      term2 = (betarv**(-2)) * gamarv
360      d3v = 0.5 * ( term1 + term2 )
361
362      c D3H
363      phi = pi2
364      term1 = (betalh**(-2)) * (-dsqrt(1.-(gamalh**2)))
365      term2 = (betarh**(-2)) * (dsqrt(1.-(gamarh**2)))
366      d3h = 0.5 * ( term1 + term2 )
367
368      c D4V
369      d4v = ( (1./betalv) + (1./betarv) ) / capl
370
371      c D4H
372      d4h = ( (1./betalh) + (1./betarh) ) / capl
373
374
375      c Print out the data, as well as writing it to the ASCII file
376
377      c      write(*,2000)
378      c2000 format(///' The D's are:',/)
379
380      c      write(*,2060)i, d1v, d1h, d2v, d2h
381      c2060 format(/' EEG Electrode # ',i3,/' D1V = ',f16.5,/
382      c      +' D1H = ',f16.5,/' D2V = ',f16.5,/' D2H = ',g16.5)
383
384      c      write(*,2010) d3v, d3h, d4v, d4h
385      c2010 format(/' D3V = ',f16.5,/' D3H = ',f16.5,
386      c      +' /' D4V = ',f16.5,/' D4H = ',f16.5,/)
387
388      c Scale D's
389
390      if ( dabs(d1v) .gt. 10. ) d1v = (d1v/dabs(d1v))*10.
391      if ( dabs(d2v) .gt. 10. ) d2v = (d2v/dabs(d2v))*10.
392      if ( dabs(d1h) .gt. 10. ) d1h = (d1h/dabs(d1h))*10.

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

393      if ( dabs(d2h) .gt. 10. ) d2h = (d2h/dabs(d2h))*10.
394      if ( dabs(d3v) .gt. 10. ) d3v = (d3v/dabs(d3v))*10.
395      if ( dabs(d4v) .gt. 10. ) d4v = (d4v/dabs(d4v))*10.
396      if ( dabs(d3h) .gt. 10. ) d3h = (d3h/dabs(d3h))*10.
397      if ( dabs(d4h) .gt. 10. ) d4h = (d4h/dabs(d4h))*10.
398
399      max = 0.000
400      if ( dabs(d1v) .gt. max ) max = dabs(d1v)
401      if ( dabs(d2v) .gt. max ) max = dabs(d2v)
402      if ( dabs(d1h) .gt. max ) max = dabs(d1h)
403      if ( dabs(d2h) .gt. max ) max = dabs(d2h)
404      if ( dabs(d3v) .gt. max ) max = dabs(d3v)
405      if ( dabs(d4v) .gt. max ) max = dabs(d4v)
406      if ( dabs(d3h) .gt. max ) max = dabs(d3h)
407      if ( dabs(d4h) .gt. max ) max = dabs(d4h)
408
409      scale = 8192.000 / max
410      d1v = d1v * (-scale)
411      d2v = d2v * (-scale)
412      d3v = d3v * scale
413      d4v = d4v * scale
414      d1h = d1h * scale
415      d2h = d2h * scale * 0.01
416      d3h = d3h * scale
417      d4h = d4h * scale
418
419      c      write(*,2100)
420      c2100  format('///' The D"s for EEG.C96 are (in order):',/)
421
422      c      write(*,2070) scale*d1v, scale*d2v, scale*d3v,
423      c      + scale*d4v, scale*d1h, scale*d2h, scale*d3h,
424      c      + scale*d4h
425      c2070  format(/8(' ',f9.0))
426
427      write(*,3333)
428      3333   format('//////////')
429      +      '//////////')
430
431      step = 3
432
433      return
434      end

```

Please wait...

DCALC Local Symbols

Name	Class	Type	Size	Offset
CCMAX	param			0006
BELL.	param			000a
STEP.	param			000e
D4H	param			0012
D3H	param			0016
D4V	param			001a
D3V	param			001e
D2H	param			0022

Microsoft FORTRAN Optimizing Compiler Version 4.00

DCALC Local Symbols

Name	Class	Type	Size	Offset
D1H	param			0026
D2V	param			002a
D1V	param			002e
CLS	param			0032
ZLINE	local	INTEGER*2	2	0002
IRESP	local	INTEGER*2	2	0004
BETARV.	local	REAL*8	8	0006
SIGMAV.	local	REAL*8	8	000e
EZLINE.	local	INTEGER*2	2	0016
H	local	REAL*8	8	0018
DENOMV.	local	REAL*8	8	0020
I	local	INTEGER*2	2	0028
J	local	INTEGER*2	2	002a
PI2	local	REAL*8	8	002c
THETAV.	local	REAL*8	8	0034
N	local	INTEGER*2	2	003c
R	local	REAL*8	8	003e
RH.	local	REAL*8	8	0046
ETA	local	REAL*8	8	004e
Z	local	REAL*8	8	0056
JEFF.	local	REAL*8	8	005e
CAPL.	local	REAL*8	8	0066
PHI	local	REAL*8	8	006e
ETAH.	local	REAL*8	8	0076
GAMAL	local	REAL*8	8	007e
MAX	local	REAL*8	8	0086
ICHAR	local	INTEGER*2	2	008e
RADDEG.	local	REAL*8	8	0090
SCALE	local	REAL*8	8	0098
TEMP2	local	REAL*8	8	00a0
RV.	local	REAL*8	8	00a8
GAMAR	local	REAL*8	8	00b0
BETAL	local	REAL*8	8	00b8
TERM1	local	REAL*8	8	00c0
DELTA	local	REAL*8	8	00c8
TERM2	local	REAL*8	8	00d0
GAMALH.	local	REAL*8	8	00d8
PS1	local	REAL*8	8	00e0
NEGFLA.	local	REAL*8	8	00e8
STEP1	local	INTEGER*2	2	00f0
BETAR	local	REAL*8	8	00f2
GAMARH.	local	REAL*8	8	00fa
BETALH.	local	REAL*8	8	0102
ETAV.	local	REAL*8	8	010a
SIGMA	local	REAL*8	8	0112
TEMP.	local	REAL*8	8	011a
BETARH.	local	REAL*8	8	0122
THETA	local	REAL*8	8	012a
GAMALV.	local	REAL*8	8	0132
SIGMAH.	local	REAL*8	8	013a
RESP.	local	CHAR*1	1	0142

Microsoft FORTRAN Optimizing Compiler Version 4.00

DCALC Local Symbols

Name	Class	Type	Size	Offset
DENOMH.	local	REAL*8	8	0144
THETAH.	local	REAL*8	8	014c
GAMARV.	local	REAL*8	8	0154
BETALV.	local	REAL*8	8	015c

```

435
436 c -----
437
438      subroutine error(bell)
439
440      character*1      resp, bell
441
442      write(*,100)bell
443 100      format(///12x,a1,'Invalid Response !'//,13x,
444      +'Type ENTER to continue...')
445      read(*,20)resp
446 20      format(bn,a1)
447
448      return
449      end

```

ERROR Local Symbols

Name	Class	Type	Size	Offset
BELL.	param			0006
RESP.	local	CHAR*1	1	0164

Global Symbols

Name	Class	Type	Size	Offset
DCALC	FSUBRT	***	***	0000
ERROR	FSUBRT	***	***	1359

Code size = 1359 (4953)
Data size = 0206 (518)
Bss size = 0165 (357)

No errors detected

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

1 $DECLARE
2 $LARGE
3
4      subroutine sgm(magsum, orient)
5
6 c      Jeffrey C. Sigl      December 26, 1989
7
8
9 c Data Structures
10
11      character*6      resp
12      character*1      bite(2)
13      character*1      cappa, cappb, cappc, cappd, cappe, cappf
14      character*1      chr(6), one, sp, cr, retran, oktran
15      integer*2        in, istart, i, j, k, irec, orient
16      real*4           eeg(4096), magsum
17      complex*8        sg(4096), sgtemp
18      integer          tmprsp
19
20 c Functions
21
22      integer          ichar
23      real*4           float, cabs
24      complex*8        cmplx
25
26 c Data Relations
27
28      equivalence      (resp,chr)
29
30      data cappa,cappb,cappc,cappd,cappe,cappf/'A','B','C','D','E','F'/
31      data one /49/
32      data sp, retran, oktran /8#140, 0, 1/
33
34      open (9,file='scrtch.dat',status='new',access='sequential',
35      +form='formatted')
36
37
38 c Load EEG data - Read 120 words
39
40      j = 2018
41
42      do 33 i = 1, 60
43
44 17          read(11,105)in
45 105          format(i6)
46
47 c          write(*,105)in
48
49          orient = in - ((in/10) * 10)
50          in = in/10
51          if ( orient .eq. 1 ) then
52              in = -in
53          endif
54
55          eeg(j) = float(in)
56

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

57          j=j+1
58
59 33      continue
60
61  c Extend ends
62
63          do 20 i = 1, 2017
64              eeg(i) = eeg(2018)
65 20      continue
66
67          do 30 i = 2078, 4096
68              eeg(i) = eeg(2077)
69 30      continue
70
71  c Window & FFT eeg
72
73          call sgwin(eeg, eeg, 9, 0.001, 12)
74
75          do 60 i = 1, 4096
76              sg(i) = cmplx( eeg(i), 0. )
77 60      continue
78
79          call cfft( sg, 12, 0, 1.0 )
80
81          do 2323 i = 1, 4096
82              write(9,*)sg(i)
83 2323   continue
84
85
86
87  c Load EOG data - Read 120 words
88
89          j = 2018
90          do 133 i = 1, 60
91
92 117      read(11,105)in
93
94          orient = in - ((in/10) * 10)
95          in = in/10
96          if ( orient .eq. 1 ) then
97              in = -in
98          endif
99
100         eeg(j) = float(in)
101
102         j=j+1
103 133      continue
104
105  c Extend ends
106
107          do 120 i = 1, 2017
108              eeg(i) = eeg(2018)
109 120      continue
110
111          do 130 i = 2078, 4096
112              eeg(i) = eeg(2077)

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

113 130    continue
114
115 c Window & FFT eeg
116
117      call sgwin(eeg, eeg, 9, 0.001, 12)
118
119      do 170 i = 1, 4096
120          sg(i) = cmplx( eeg(i), 0. )
121 170    continue
122
123      call cfft( sg, 12, 0, 1.0 )
124
125
126 c Divide & compute the average magnitude
127
128      magsum = 0.
129      rewind 9
130      do 1013 i = 6, 45
131          read(9,*)sgtemp
132          eeg(i) = 100. * cabs( sgtemp/sg(i) )
133          magsum = magsum + eeg(i)
134
135 c          write(*,2345)i,eeg(i),magsum
136 c2345      format(' ',i6,2(' ',f15.3,))
137
138 1013    continue
139      magsum = magsum / 40.
140
141 9999    close(9,status='delete')
142
143      return
144      end

```

SGM Local Symbols

Name	Class	Type	Size	Offset
ORIENT.	param			0006
MAGSUM.	param			000a
EEG	local	REAL*4	16384	0000
I	local	INTEGER*2	2	0002
J	local	INTEGER*2	2	0004
K	local	INTEGER*2	2	0006
SGTEMP.	local	COMPLEX*8	8	0008
CAPPA	local	CHAR*1	1	000a
CAPPB	local	CHAR*1	1	000b
CAPPC	local	CHAR*1	1	000c
CAPPD	local	CHAR*1	1	000d
CAPPE	local	CHAR*1	1	000e
CAPPF	local	CHAR*1	1	000f
ONE	local	CHAR*1	1	0010
CR.	local	CHAR*1	1	0010
SP.	local	CHAR*1	1	0011
RETRAN.	local	CHAR*1	1	0012
ISTART.	local	INTEGER*2	2	0012

PAGE 4
03-22-90
17:43:57

Microsoft FORTRAN Optimizing Compiler Version 4.00

SGM Local Symbols

Name	Class	Type	Size	Offset
OKTRAN.	local	CHAR*1	1	0013
IN.	local	INTEGER*2	2	0014
IREC.	local	INTEGER*2	2	0016
TMPRSP.	local	INTEGER*4	4	0018
ICHAR.	local	INTEGER*4	4	001c
RESP.	local	CHAR*6	6	0020
SG.	local	COMPLEX*8	32768	4000
BITE.	local	CHAR*1	2	c000
CHR.	local	CHAR*1	6	0020

Global Symbols

Name	Class	Type	Size	Offset
CFFT.	extern	***	***	***
SGM.	FSUBRT	***	***	0000
SGWIN.	extern	***	***	***

Code size = 04b0 (1200)

Data size = 0060 (96)

Bss size = 0026 (38)

No errors detected

```

Line# Source Line      Microsoft FORTRAN Optimizing Compiler Version 4.00

 1 $LARGE
 2 $declare
 3 c      ftune.for
 4 c
 5 c      Steven M. Falk
 6
 7      subroutine ftune(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
 8      +ccmax)
 9
10 c Data Structures
11
12      character*1    cls(4), chr(6), cbell
13      character*1    resp, sp, retran, oktran, cr
14      integer        i, j, n, k, i, tmprsp, in, mod
15      real*4         d1v, d2v, d1h, d2h
16      real*4         d3v, d4v, d3h, d4h, ccmax
17      real*4         cc, ccrawv, ccrawh, ccold, delta1, delta2, ccrtio
18      integer        ichar
19      real*4         float, cabs, real, abs
20      complex*8      cmplx
21      character      yes, yess
22      integer*2      num, iorder, icnst2, ilim, orient, orient
23      real*4         eogvu(512), eogvl(512), eogh(512), raweeg(512)
24      real*4         eohvu(512), eohvl(512), eohh(512), raweeh(512)
25      real*4         alpha(512), temp(512)
26      real*4         time, alpha1, alpha2, const, intrvl, ceeg
27      real*4         vmag, hmag, gain, raddeg, order, const1, pi2
28      complex*8      comp1(512), comp2(512)
29      double precision scale, max, dabs
30
31      equivalence    (resp,chr)
32
33 c Data Initialization
34
35      pi2 = 1.5707963268000
36      raddeg = 57.295779513100
37      data yes, yess /'Y','Y'/
38      data sp, retran, oktran /8#140, 0, 1/
39      data cbell / 8#7/
40
41 c      delta1 = 0.5 * d1v
42      delta1 = 100.
43      delta2 = 0.5 * d2v
44      do 401 i = 1, 512
45
46 407          read(11,402)in
47 402          format(i6)
48
49          orient = in - ((in/10) * 10)
50          in = in/10
51          if ( orient .eq. 1 ) then
52              in = -in
53          endif
54
55
56 c          if (abs(float(in)) .gt. 470.) then

```

```
Line# Source Line      Microsoft FORTRAN Optimizing Compiler Version 4.00

57 c                  gain = -9999.
58 c                  write (*,4433)
59 c4433              format(' Data is saturated. Interrogate again.')
```

```
60 c                  goto 2222
61 c                  endif
62
63                  raweeg(i) = float(in)
64
65 401      continue
66
67      do 1401 i = 1, 512
68
69 1407          read(11,1402)in
70 1402          format(i6)
71
72              orent = in - ((in/10) * 10)
73              in = in/10
74              if ( orent .eq. 1 ) then
75                  in = -in
76              endif
77
78 c              if (abs(float(in)) .gt. 470.) then
79 c                  gain = -9999.
80 c                  write (*,4433)
81 c                  goto 2222
82 c              endif
83
84              raweeg(i) = float(in)
85
86 1401      continue
87
88      do 411 i = 1, 512
89
90 417          read(11,412)in
91 412          format(i6)
92
93              orent = in - ((in/10) * 10)
94              in = in/10
95              if ( orent .eq. 1 ) then
96                  in = -in
97              endif
98
99 c              if (abs(float(in)) .gt. 470.) then
100 c                  gain = -9999.
101 c                  write (*,4433)
102 c                  goto 2222
103 c              endif
104
105              eogh(i) = float(in)
106
107 411      continue
108
109      do 1411 i = 1, 512
110
111 1417          read(11,1412)in
112 1412          format(i6)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```
113
114      orent = in - ((in/10) * 10)
115      in = in/10
116      if ( orent .eq. 1 ) then
117          in = -in
118      endif
119
120 c      if (abs(float(in)) .gt. 470.) then
121 c          gain = -9999.
122 c          write (*,4433)
123 c          goto 2222
124 c      endif
125
126      eogh(i) = float(in)
127
128 1411      continue
129
130      do 421 i = 1, 512
131
132 427          read(11,422)in
133 422          format(i6)
134
135          orent = in - ((in/10) * 10)
136          in = in/10
137          if ( orent .eq. 1 ) then
138              in = -in
139          endif
140
141 c          if (abs(float(in)) .gt. 470.) then
142 c              gain = -9999.
143 c              write (*,4433)
144 c              goto 2222
145 c          endif
146
147          eogvu(i) = float(in)
148
149 421      continue
150
151      do 1421 i = 1, 512
152
153 1427          read(11,1422)in
154 1422          format(i6)
155
156          orent = in - ((in/10) * 10)
157          in = in/10
158          if ( orent .eq. 1 ) then
159              in = -in
160          endif
161
162 c          if (abs(float(in)) .gt. 470.) then
163 c              gain = -9999.
164 c              write (*,4433)
165 c              goto 2222
166 c          endif
167
168          eogvu(i) = float(in)
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```
169
170 1421 continue
171
172 do 431 i = 1, 512
173
174 437 read(11,432)in
175 432 format(i6)
176
177 orent = in - ((in/10) * 10)
178 in = in/10
179 if ( orent .eq. 1 ) then
180 in = -in
181 endif
182
183 c if (abs(float(in)) .gt. 470.) then
184 c gain = -9999.
185 c write (*,4433)
186 c goto 2222
187 c endif
188
189 eogvl(i) = float(in)
190
191 431 continue
192
193 do 1431 i = 1, 512
194
195 1437 read(11,1432)in
196 1432 format(i6)
197
198 orent = in - ((in/10) * 10)
199 in = in/10
200 if ( orent .eq. 1 ) then
201 in = -in
202 endif
203
204 c if (abs(float(in)) .gt. 470.) then
205 c gain = -9999.
206 c write (*,4433)
207 c goto 2222
208 c endif
209
210 eogvl(i) = float(in)
211
212 1431 continue
213
214
215 c fine tuning algorithm
216
217 11 continue
218
219 call ccf(eogvu,raweeg,ccrawv)
220 call ccf(eogh,raweeg,ccrawh)
221
222 call crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
223 +eogvu,eogvl,eogh,raweeg,temp)
224
```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

225      call ccf(eogvu,temp,ccold)
226
227      if (abs(ccold) .gt. abs(ccrawv)) then
228          write(*,1109)cbell
229 1109      format(' Data Entry Error!',a1)
230          gain = -9999.
231          goto 2222
232      endif
233
234      if (abs(ccold) .gt. 0.8) then
235          write(*,1108)cbell
236 1108      format(' Measurement Error!',a1)
237          gain = -9999.
238          goto 2222
239      endif
240
241      call ccf(eogh,temp,ccold)
242
243      if (abs(ccold) .gt. abs(ccrawh)) then
244          write(*,1109)cbell
245          gain = -9999.
246          goto 2222
247      endif
248
249      if (abs(ccold) .gt. 0.8) then
250          write(*,1108)cbell
251          gain = -9999.
252          goto 2222
253      endif
254
255 2431      write(*,2460)
256 2460      format(' Fine tuning geometric VERTICAL parameters of model...')
257          write(*,2462)
258 2462      format('      Iteration      D1      D2      CCF')
259
260      call ccf(eogvu,temp,ccold)
261
262      l = 0
263      write(*,*)l,d1v,d2v,ccold
264 2500      l = l + 1
265
266      d1v = d1v + delta1
267      call crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
268 +eogvu,eogvl,eogh,raweeg,temp)
269      call ccf(eogvu,temp,cc)
270
271      ccrtio = cc / ccold
272      if (ccrtio .lt. 0) then
273          delta1 = delta1 * (-.5)
274          goto 2501
275      endif
276
277      if (abs(cc) .gt. abs(ccold)) then
278          delta1 = delta1 * (-1.)
279          goto 2501
280      endif

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

281
282 2501 ccold = cc
283      d2v = d2v + delta2
284      call crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
285 +eogvu,eogvl,eogh,raweeg,temp)
286      call ccf(eogvu,temp,cc)
287
288      ccrtio = cc / ccold
289      if (ccrtio .lt. 0) then
290          delta2 = delta2 * (-.5)
291          goto 2511
292      endif
293
294      if (abs(cc) .gt. abs(ccold)) then
295          delta2 = delta2 * (-1.)
296          goto 2511
297      endif
298
299 2511 write(*,*)l,d1v,d2v,cc
300      j = mod(l,10)
301      if (j .eq. 0) then
302          write(*,2512)
303 2512 format(' Continue iterations ? (Y/N) ',\ )
304          read(*,2513)resp
305 2513 format(a1)
306          if (resp .ne. 'Y') then
307              write(*,2518)
308 2518 format(' Fine tuning geometric HORIZONTAL parameters',
309 + ' of model...')
310              write(*,2462)
311              call ccf(eogh,temp,ccold)
312              l=0
313              write(*,*)l,d1h,d2h,ccold
314              delta1 = 0.5 * d1h
315              delta2 = 0.5 * d2h
316              goto 2600
317          endif
318      endif
319      ccold = cc
320      if (abs(cc) .lt. ccmx) then
321          write(*,2518)
322          write(*,2462)
323          call ccf(eogh,temp,ccold)
324          l=0
325          write(*,*)l,d1h,d2h,ccold
326          delta1 = 0.5 * d1h
327          delta2 = 0.5 * d2h
328          goto 2600
329      endif
330      goto 2500
331
332 2600 l = l + 1
333
334
335      d1h = d1h + delta1
336      call crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

337      +eogvu,eogvl,eogh,raweeg,temp)
338      call ccf(eogh,temp,cc)
339
340      ccrtio = cc / ccold
341      if (ccrtio .lt. 0) then
342          delta1 = delta1 * (-.5)
343          goto 2601
344      endif
345
346      if (abs(cc) .gt. abs(ccold)) then
347          delta1 = delta1 * (-1.)
348          goto 2601
349      endif
350
351 2601      ccold = cc
352          d2h = d2h + delta2
353          call crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
354      +eogvu,eogvl,eogh,raweeg,temp)
355          call ccf(eogh,temp,cc)
356
357          ccrtio = cc / ccold
358          if (ccrtio .lt. 0) then
359              delta2 = delta2 * (-.5)
360              goto 2611
361          endif
362
363          if (abs(cc) .gt. abs(ccold)) then
364              delta2 = delta2 * (-1.)
365              goto 2611
366          endif
367
368 2611      write(*,*)l,d1h,d2h,cc
369          j = mod(l,10)
370          if (j .eq. 0) then
371              write(*,2512)
372              read(*,2513)resp
373              if (resp .ne. 'Y') then
374                  goto 2800
375              endif
376          endif
377          ccold = cc
378          if (abs(cc) .lt. ccmx) then
379              goto 2800
380          endif
381          goto 2600
382
383 2800      continue
384
385          open(12,file='correeg.dat')
386          do 2803 i=64,384
387              write(12,*)i,temp(i),raweeg(i),eogvu(i),eogh(i)
388 2803      continue
389          close(12)
390
391      c Scale D's
392

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

393      max = 0.000
394      if ( dabs(d1v) .gt. max ) max = dabs(d1v)
395      if ( dabs(d2v) .gt. max ) max = dabs(d2v)
396      if ( dabs(d1h) .gt. max ) max = dabs(d1h)
397      if ( dabs(d2h) .gt. max ) max = dabs(d2h)
398      if ( dabs(d3v) .gt. max ) max = dabs(d3v)
399      if ( dabs(d4v) .gt. max ) max = dabs(d4v)
400      if ( dabs(d3h) .gt. max ) max = dabs(d3h)
401      if ( dabs(d4h) .gt. max ) max = dabs(d4h)
402
403      scale = 8192.000 / max
404      d1v = d1v * scale
405      d2v = d2v * scale
406      d3v = d3v * scale
407      d4v = d4v * scale
408      d1h = d1h * scale
409      d2h = d2h * scale
410      d3h = d3h * scale
411      d4h = d4h * scale
412
413 2222      return
414          end

```

FTUNE Local Symbols

Name	Class	Type	Size	Offset
CCMAX	param			0006
GAIN.	param			000a
HMAG.	param			000e
VMAG.	param			0012
D4H	param			0016
D3H	param			001a
D4V	param			001e
D3V	param			0022
D2H	param			0026
D1H	param			002a
D2V	param			002e
D1V	param			0032
EOGVU	local	REAL*4	2048	0000
CMPLX	local	COMPLEX*8	8	0002
CCRTIO.	local	REAL*4	4	000a
YES	local	CHAR*1	1	000d
YESS.	local	CHAR*1	1	000e
IORDER.	local	INTEGER*2	2	000e
SP.	local	CHAR*1	1	000f
RETRAN.	local	CHAR*1	1	0010
CCRAWV.	local	REAL*4	4	0010
OKTRAN.	local	CHAR*1	1	0011
CBELL	local	CHAR*1	1	0012
CC.	local	REAL*4	4	0014
CONST	local	REAL*4	4	0018
ORENT	local	INTEGER*2	2	001c
I	local	INTEGER*4	4	001e
J	local	INTEGER*4	4	0022

Microsoft FORTRAN Optimizing Compiler Version 4.00

FTUNE Local Symbols

Name	Class	Type	Size	Offset
PI2	local	REAL*4	4	0026
K	local	INTEGER*4	4	002a
L	local	INTEGER*4	4	002e
N	local	INTEGER*4	4	0032
ORIENT.	local	INTEGER*2	2	0036
CEEG.	local	REAL*4	4	0038
CR.	local	CHAR*1	1	003c
ALPHA1.	local	REAL*4	4	003e
IN.	local	INTEGER*4	4	0042
ALPHA2.	local	REAL*4	4	0046
CABS.	local	REAL*4	4	004a
DELTA1.	local	REAL*4	4	004e
DELTA2.	local	REAL*4	4	0052
INTRVL.	local	REAL*4	4	0056
REAL.	local	REAL*4	4	005a
CCOLD	local	REAL*4	4	005e
MAX	local	REAL*8	8	0062
TMPRSP.	local	INTEGER*4	4	006a
RADDEG.	local	REAL*4	4	006e
ICHAR	local	INTEGER*4	4	0072
SCALE	local	REAL*8	8	0076
ILIM.	local	INTEGER*2	2	007e
TIME.	local	REAL*4	4	0080
NUM	local	INTEGER*2	2	0084
ICNST2.	local	INTEGER*2	2	0086
CONST1.	local	REAL*4	4	0088
CCRAWH.	local	REAL*4	4	008c
ORDER	local	REAL*4	4	0090
EOHVU	local	REAL*4	2048	0800
CHR	local	CHAR*1	6	1000
COMP1	local	COMPLEX*8	4096	1006
COMP2	local	COMPLEX*8	4096	2006
CLS	local	CHAR*1	4	3006
EOGH.	local	REAL*4	2048	300a
EOHH.	local	REAL*4	2048	380a
ALPHA	local	REAL*4	2048	400a
TEMP.	local	REAL*4	2048	480a
RAWEEG.	local	REAL*4	2048	500a
RAWEEH.	local	REAL*4	2048	580a
EOGVL	local	REAL*4	2048	600a
EOHVL	local	REAL*4	2048	680a
RESP.	equiv	CHAR*1	1	1000

415

416 c

417

418 subroutine ccf(ueeg,ceeg,cc)

419

420 real*4 cc,cov,sho

421 real*4 ueeg(512),ceeg(512)

422 real*8 a(512,6),b(6)

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

423      integer*2      ii,jj
424
425
426      do 10 ii=1,512
427          a(ii,1) = ueeg(ii)
428          a(ii,2) = ceeg(ii)
429          a(ii,3) = a(ii,1)*a(ii,2)
430          a(ii,4) = a(ii,1)+a(ii,2)
431          a(ii,5) = a(ii,1)*a(ii,1)
432          a(ii,6) = a(ii,2)*a(ii,2)
433 10      continue
434
435      do 20 ii=1,6
436          b(ii) = 0.0
437 20      continue
438
439      do 30 ii=64,448
440          do 31 jj=1,6
441              b(jj) = b(jj) + a(ii,jj)
442 31      continue
443 30      continue
444      do 40 ii=1,6
445          b(ii) = b(ii) / 384.0
446 40      continue
447
448      cov = b(3) - (b(1) * b(2))
449      sho = (b(5)-(b(1)*b(1))) * (b(6)-(b(2)*b(2)))
450      if ( sho .le. 0.0 ) then
451          sho = 0.0
452      endif
453      sho = sho*.5
454      if ( sho .eq. 0.0 ) then
455          cc = 1.0
456          goto 75
457      endif
458      cc = cov / sho
459
460 75      continue
461
462      return
463      end

```

CCF Local Symbols

Name	Class	Type	Size	Offset
CC.	param			0006
CEEG.	param			000a
UEEG.	param			000e
II.	local	INTEGER*2	2	0094
JJ.	local	INTEGER*2	2	0096
COV.	local	REAL*4	4	0098
SHO.	local	REAL*4	4	009c
A.	local	REAL*8	24576	7142
B.	local	REAL*8	48	d142

Microsoft FORTRAN Optimizing Compiler Version 4.00

```

464
465
466 c-----
467
468      subroutine crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
469 +eogvu,eogvl,eogh,raweeg,temp)
470
471      integer      i, j, n, k, tmprsp, in
472      real*4       d1v, d2v, d1h, d2h
473      real*4       d3v, d4v, d3h, d4h
474      real*4       float, cabs, real
475      complex*8    cmplx
476      real*4       eogvu(512), eogvl(512), eogh(512), raweeg(512)
477      real*4       teogvu(52), teogvl(512), teogh(512)
478      real*4       alpha(512), temp(512)
479      real*4       time, alpha1, alpha2, const, intrvl, ceeg
480      real*4       vmag, hmag, gain, raddeg, order, const1, pi2
481      complex*8    comp1(512), comp2(512)
482
483
484 c Window & FFT EOG-VU
485
486      call sgwin(eogvu, teogvu, 9, 0.001, 9)
487      do 60 i = 1, 512
488          comp1(i) = cmplx( teogvu(i), 0. )
489 60      continue
490      call cfft( comp1, 9, 0, 1.0 )
491
492
493 c Window & FFT EOG-VL
494
495      call sgwin(eogvl, teogvl, 9, 0.001, 9)
496      do 70 i = 1, 512
497          comp2(i) = cmplx( teogvl(i), 0. )
498 70      continue
499      call cfft( comp2, 9, 0, 1.0 )
500
501
502 c Compute Alpha
503
504      do 80 i = 1, 512
505          temp(i) = (gain * cabs( comp1(i) / comp2(i) )) / 100.
506
507          if (temp(i) .lt. 1.) then
508              temp(i) = 1.
509          elseif (temp(i) .gt. 4.) then
510              temp(i) = 4.
511          endif
512
513          alpha(i) = temp(i)
514 80      continue
515
516      goto 8989
517
518 c Moving Average Filter of Alpha (7 point)

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

519
520     alpha(1) = ((4.*temp(1)) + temp(2) + temp(3) + temp(4)) / 7.
521     alpha(2) = ((3.*temp(1)) + temp(2) + temp(3) + temp(4)
522 +           + temp(5)) / 7.
523     alpha(3) = ((2.*temp(1)) + temp(2) + temp(3) + temp(4)
524 +           + temp(5) + temp(6)) / 7.
525     do 90 i = 4, 509
526         alpha(i) = (temp(i-3) + temp(i-2) + temp(i-1) + temp(i) +
527 +           temp(i+1) + temp(i+2) + temp(i+3)) / 7.
528 90    continue
529     alpha(510) = (temp(507) + temp(508) + temp(509) +
530 +           temp(510) + temp(511) + (2.*temp(512))) / 7.
531     alpha(511) = (temp(508) + temp(509) + temp(510) +
532 +           temp(511) + (3.*temp(512))) / 7.
533     alpha(512) = (temp(509) + temp(510) + temp(511) +
534 +           (4.*temp(512))) / 7.
535
536
537 c Window & FFT EOG-H
538
539 8989  call sgwin(eogh, teogh, 9, 0.001, 9)
540     do 100 i = 1, 512
541         comp2(i) = cmplx( teogh(i), 0. )
542 100    continue
543     call cfft( comp2, 9, 0, 1.0 )
544
545
546 c =====
547 c                      Correction
548 c =====
549
550     do 1000 i = 1, 512
551
552         alpha1 = alpha(i) + 1.
553         alpha2 = alpha(i) - 1.
554
555 c Vertical Component
556
557         const = (d1v*alpha1 + d2v*alpha2) / (d3v*alpha1 + d4v*alpha2)
558         comp1(i) = const * vmag * comp1(i) / 100.
559
560 c Horizontal Component
561
562         const = (d1h*alpha1 + d2h*alpha2) / (d3h*alpha1 + d4h*alpha2)
563         comp1(i) = comp1(i) + (const * hmag * comp2(i)) / 100.
564
565         temp(i) = 1.
566
567 1000  continue
568
569
570 c Inverse Transform
571
572     call cift( comp1, 9, 0, 1.0 )
573
574

```

Line# Source Line Microsoft FORTRAN Optimizing Compiler Version 4.00

```

575 c Dewindow Corrector & Subtract from Raw EEG
576
577     call sgwin(temp, temp, 9, 0.001, 9)
578     do 1100 i = 1, 512
579         temp(i) = real(comp1(i)) / temp(i)
580         temp(i) = raweeg(i) - temp(i)
581
582 1100 continue
583
584     return
585 end

```

CRCT Local Symbols

Name	Class	Type	Size	Offset
TEMP.	param			0006
RAWEEG.	param			000a
EOGH.	param			000e
EOGVL	param			0012
EOGVU	param			0016
GAIN.	param			001a
HMAG.	param			001e
VMAG.	param			0022
D4H	param			0026
D3H	param			002a
D4V	param			002e
D3V	param			0032
D2H	param			0036
D1H	param			003a
D2V	param			003e
D1V	param			0042
ALPHA	local	REAL*4	2048	0000
CONST	local	REAL*4	4	00a0
I	local	INTEGER*4	4	00a4
J	local	INTEGER*4	4	00a8
P12	local	REAL*4	4	00ac
K	local	INTEGER*4	4	00b0
N	local	INTEGER*4	4	00b4
CEEG.	local	REAL*4	4	00b8
ALPHA1.	local	REAL*4	4	00bc
IN.	local	INTEGER*4	4	00c0
ALPHA2.	local	REAL*4	4	00c4
INTRVL.	local	REAL*4	4	00c8
TMPRSP.	local	INTEGER*4	4	00cc
RADDEG.	local	REAL*4	4	00d0
TIME.	local	REAL*4	4	00d4
FLOAT	local	REAL*4	4	00d8
CONST1.	local	REAL*4	4	00dc
ORDER	local	REAL*4	4	00e0
TEOGH	local	REAL*4	2048	0800
TEOGVL.	local	REAL*4	2048	d172
TEOGVU.	local	REAL*4	208	d972
COMP1	local	COMPLEX*8	4096	da42
COMP2	local	COMPLEX*8	4096	ea42

PAGE 14
03-22-90
17:44:21

Microsoft FORTRAN Optimizing Compiler Version 4.00

Global Symbols

Name	Class	Type	Size	Offset
CCF	FSUBRT	***	***	11f2
CFFT.	extern	***	***	***
CIFT.	extern	***	***	***
CRCT.	FSUBRT	***	***	1590
FTUNE	FSUBRT	***	***	0000
SGWIN	extern	***	***	***

Code size = 2282 (8834)
Data size = 01b3 (435)
Bss size = 00e4 (228)

No errors detected