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

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.
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.
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.
# TABLE OF CONTENTS

**INTRODUCTION** .......................................................... 2  
  *Nature of the Problem.* ........................................... 2  
  *Background.* ...................................................... 2  
  *Purpose of the Present Work.* ................................... 6  
  *Methods of Approach - Mathematical.* ....................... 6  
  *Methods of Approach - Engineering.* ......................... 8  

**BODY** ........................................................................ 9  
  *Refinement of the Mathematical Technique.* ................... 9  
  *Implementation of the Technique in Hardware.* .......... 13  

**CONCLUSIONS** ........................................................... 15  

**REFERENCES** ............................................................... 16  

**APPENDIX** .................................................................. 18
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. Convolving 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; Verlegers et al, 1982; Gratton, 1983; Fortgens & De Bruin, 1982; Woestenburg et al 1982; 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, 1982; 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, 1982; 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:

\[
v_o [t] = \int v_i [\mu] h(t-\mu) d\mu
\]

where:
- \( v_i \) = input potential
- \( v_o \) = output potential
- \( d \) = distance
- \( A \) = amplitude
- \( f \) = frequency
- \( \phi \) = 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; Woestenburg 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), Woestenburg 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 artifact 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 \( g(\omega, C_e) \) as a function of radial frequency \( \omega = 2\pi f \) as measured at the \( C_e \). 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 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 \( \phi \) 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 \( \text{cor} V_{\text{EEG}}(\omega) \) is accomplished in the frequency domain and is based on (a) measurement of the OBServed EEG \( \text{obs} V_{\text{EEG}}(\omega) \) and OBServed EOG \( \text{obs} V_{\text{EOG}}(\omega) \); (b) measurement of the system response to STIMulation \( \text{stim} V_{\text{EEG}}(\omega) \) & \( \text{stim} V_{\text{EOG}}(\omega) \) for direct interrogation, (c) a mathematical model that describes the electrodynamic behavior of the system for Theoretical Eye Movements \( \text{tem} V_{\text{EEG}}(\omega) \) & \( \text{tem} mV_{\text{EOG}}(\omega) \) and Theoretical Direct Interrogation \( \text{tdi} V_{\text{EEG}}(\omega) \) & \( \text{tdi} V_{\text{EOG}}(\omega) \); and (d) measurement of the CALibration of each recording channel \( \text{cal} V_{\text{EEG}}(\omega) \) & \( \text{cal} V_{\text{EOG}}(\omega) \). 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:

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

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

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

\(S\) 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 electrodynamical 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 \(\xi\) 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 electrodynamically.

\(D'\) 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 \(\phi\) of the surface image of the ocular dipole pair. Differentiation of this equation with respect to \(\phi\) yields an equation describing the change in potential at an EOG electrode due to a change in \(\phi\). When the change in potential with respect to \(\phi\) is zero, the potential is at an extremum (maximum or minimum) and the corresponding \(\phi\), at a particular electrode site, can be computed. This value of \(\phi\) 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 \(\phi\)’s. A change in \(\phi\) 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:

$$\text{EEG}^c(t) = \text{EEG}^o(t) - \text{IFT} \left( \text{EOG}^o(s) \times S(s) \times G(s) \times D \right)$$

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 $V=ktq/r$, 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.

$$V = kq[1/(r-L\sin\theta)]+[-n/(r+L\sin\theta)].$$  (1)

Rearranging Equation 1 yields

$$V = kq[(m-n)r+(m+n)L\sin\theta]/(r^2-L^2\sin^2\theta))$$  (2)

Since $r>>L$, we can simplify Equation 2:

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

where $\alpha = m/n$.

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

$$V = (nkq/r^2)L\sin\theta = Ar^2\sin\theta.$$  (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 $\alpha\neq1$, the zero-potential line becomes a circle described by,

$$x^2 + (y+G)^2 = G^2,$$

where $G=\frac{L[(\alpha+1)/(\alpha-1)]}$.  (5)
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 $\beta_L$ from the left eye and $\beta_R$ 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:
\[
\beta_L = [r^2 + t^2 + rZ\cos\theta]^\frac{1}{2} \\
\beta_R = [r^2 + t^2 - rZ\cos\theta]^\frac{1}{2}
\]
\hspace{1cm} (6)
\hspace{1cm} (7)

From the law of sines:
\[
\gamma_L = -\phi + \sin^{-1}[\frac{(r/\beta_L)\sin\theta}{2}] \\
\gamma_R = \pi - \phi - \sin^{-1}[\frac{(r/\beta_R)\sin\theta}{2}]
\]
\hspace{1cm} (8)
\hspace{1cm} (9)

where: $\phi$ = 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:

\[
V = \frac{4}{3} \left( \beta_L^2 - \right. \left. \frac{1}{2} \right) \sin \phi + \gamma_L \cos \phi \right) + \left[ \beta_R^2 \left( \frac{1}{2} \right) \sin \phi + \gamma_R \cos \phi \right] + \left( A/L \right) \left( \alpha - 1 \right) \left( \beta_L^{-1} + \beta_R^{-1} \right),
\]

where:
\[
\beta_L = \frac{1}{(r)^2 + (Z)^2 + rZ \cos \theta}^{1/2}
\]
\[
\beta_R = \frac{1}{(r)^2 + (Z)^2 - rZ \cos \theta}^{1/2}
\]
\[
\gamma_L = \frac{r}{\beta_L} \sin \theta
\]
\[
\gamma_R = \frac{r}{\beta_R} \sin \theta
\]

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 Equation 4:

\[
V = B \left( \frac{1}{2} \sigma \sin \phi \right)
\]

where:
\[
\sigma = \left[ r^2 + h^2 - 2r \sin \theta \right]^{1/2}
\]
\[
\psi = \text{angle of the stimulus dipole (an analog of } \phi)\]

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:

\[
\text{corr } V_{EEG} = \text{obs } V_{EEG} - \text{obs } V_{EOG} \frac{S_{EEG} / S_{EOG}}{D}
\]

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.

$$D = \frac{(\alpha+1)[\beta_L^{-2} \left(- (1-\gamma_L)^{2/3} \sin\phi + \gamma_L \cos\phi \right) + \beta_R^{-2} \left( (1-\gamma_R)^{2/3} \sin\phi + \gamma_R \cos\phi \right)]}{(\alpha+1)[\gamma_L^{-2} \left(- (1-\eta_L)^{2/3} \sin\phi + \gamma_L \cos\phi \right) + \gamma_R^{-2} \left( (1-\eta_R)^{2/3} \sin\phi + \gamma_R \cos\phi \right)] + ((\alpha-1)/L)[\beta_L^{-1} + \beta_R^{-1}]}$$

where,

$\beta_L = [(r)^2 + (z)^2 + rZ \cos\theta]^{1/2}$,
$\beta_R = [(r)^2 + (z)^2 - rZ \cos\theta]^{1/2}$,
$\gamma_L = (r/\beta_L) \sin\theta$,
$\gamma_R = (r/\beta_R) \sin\theta$,
$\sigma = [r^2 + h^2 - 2rh \sin\theta]^{1/2}$,
$\eta = (-r/\sigma) \cos\theta$,

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

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

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

We measure $z$ and $h$, as well as $\beta_L$ and $\beta_R$ for each electrode (EEG and EOG). We then calculate $r$ and $\theta$ for each electrode, and then calculate $\gamma_L$, $\gamma_R$, $\sigma$, and $\eta$ 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


APPENDIX

The following appendix is the operations manual for the EEG Artifact Rejection System (EARS) device.
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 ELECTROENCEPHALOGRAM 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: 57 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 1.1: ELECTROENCEPHALOGRAM 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, an 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. Whe: 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 Chann...
#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.

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

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 artifacts will be corrected.
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.

<table>
<thead>
<tr>
<th>Output #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Real Time EEG</td>
</tr>
<tr>
<td>#2</td>
<td>Real Time Event Trigger</td>
</tr>
<tr>
<td>#3</td>
<td>Delayed, Corrected EEG</td>
</tr>
<tr>
<td>#4</td>
<td>Delayed Event Trigger</td>
</tr>
<tr>
<td>#5</td>
<td>Delayed, Uncorrected EEG</td>
</tr>
<tr>
<td>#6</td>
<td>Delayed Horizontal EOG</td>
</tr>
<tr>
<td>#7</td>
<td>Delayed Vertical Upper EOG</td>
</tr>
<tr>
<td>#8</td>
<td>Delayed Vertical Lower EOG</td>
</tr>
</tbody>
</table>

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/r². 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.

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

(1)
Rearranging Equation 1 yields

\[ V = kq\frac{((m-n)r+(m+n)L\sin\theta)}{(r^2-L^2\sin^2\theta)}. \]  \hspace{1cm} (2)

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

\[ V = n\left(\frac{kq}{r}\right)\frac{L\sin(\alpha+1) + (kq/r)(\alpha-1)}{r} \]

where \( \alpha = m/n \).

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

\[ V = (nkq/r^2)L\sin\theta = Ar^2\sin\theta. \]  \hspace{1cm} (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,

\[ x^2 + (y+G)^2 = G^2, \]  \hspace{1cm} (5)

where \( G = L[(\alpha+1)/(\alpha-1)] \).
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 $\beta_L$ from the left eye and $\beta_R$ 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:

\[ \beta_L = \left[ r^2 + \frac{1}{4}Z^2 + rz\cos\theta \right]^{1/2} \]  
\[ \beta_R = \left[ r^2 + \frac{1}{4}Z^2 - rz\cos\theta \right]^{1/2} \]  

From the law of sines:

\[ \gamma_L = -\phi + \sin^{-1}\left( \frac{r}{\beta_L}\sin\theta \right) \]  
\[ \gamma_R = \pi - \phi - \sin^{-1}\left( \frac{r}{\beta_R}\sin\theta \right) \]

where: \( \phi \) = 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:

\[ V = i(\alpha + 1)A\left[ \beta_L^{-1}(1-\gamma_L^2)^{1/2}\sin\phi + \gamma_L\cos\phi \right] + \beta_R^{-1}(1-\gamma_R^2)^{1/2}\sin\phi + \gamma_R\cos\phi \] + \((A/L)(\alpha - 1)(\beta_L^{-1} + \beta_R^{-1})\),

where:

\[ \beta_L = \left[ (r)^2 + \frac{1}{4}(Z)^2 + rz\cos\theta \right]^{1/2} \]  
\[ \beta_R = \left[ (r)^2 + \frac{1}{4}(Z)^2 - rz\cos\theta \right]^{1/2} \]  
\[ \gamma_L = \left( \frac{r}{\beta_L}\right)\sin\theta \]  
\[ \gamma_R = \left( \frac{r}{\beta_R}\right)\sin\theta \]
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:

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

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:

\[ \text{corr } V_{\text{EEG}} = \frac{v_{\text{EEG}}}{v_{\text{EOG}} [S_{\text{EEG}}/S_{\text{EOG}}]} V_{\text{EOG}} \]  

where \( D \) is the geometrical correction factor between the stimulus dipole and the ocular dipoles. \( S \) denotes the surface dipole stimulation response and \( v_{\text{obs}} \) 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.

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

(13)

where,

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

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

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

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

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

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

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

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

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

We measure $z$ and $h$, as well as $\beta_L$ and $\beta_R$ for each electrode (EEG and EOG). We then calculate $r$ and $\theta$ for each electrode, and then calculate $\gamma_L$, $\gamma_R$, $\sigma$, and $\eta$ 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
FIGURE VII.B.2: ISOSWITCH BOARD SCHEMATIC
FIGURE VII.B.3: ISOSWITCH BOARD
OUT
ISO-SWITCH
BOARD
FIGURE VII.B.4: INSTRUMENTATION AMP & HORIZONTAL STIMULUS SCHEMATIC
FIGURE VII.B.5: INSTRUMENTATION AMP & HORIZONTAL STIMULUS BOARD
FIGURE VII.B.6: ANALOG AMPLIFIERS SCHEMATIC
FIGURE VII.B.7: AUTOCALIBRATION & VERTICAL AUTOSTIMULATION

SCHEMATIC
FIGURE VII.B.8: ANALOG AMPLIFIER/AUTOCAL/AUTOSTIM BOARD SILKSCREEN
FIGURE VII.B.9: DIGITAL CIRCUIT SCHEMATIC
FIGURE VII.B.10: ANALOG INPUT SIGNAL LIMITER/BUFFER SCHEMATIC
FIGURE VII.B.11: DIGITAL BOARD SILKSCREEN
FIGURE VII.B.12: RELAY BOX AUXILIARY CIRCUIT SCHEMATICS
FIGURE VII.B.13: RELAY BOARDS
FIGURE VII.B.14: RELAY BOX CONNECTOR SCHEMATIC
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 C 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
APPENDIX A: PROC 1 SOFTWARE LISTING
stmt level incl
1 */

EEG.C96 - EOG Artifact Removal System
Implemented in Intel iC96 for the 80C196KA

Steven M. Falk
Jeffrey C. Sigl

Created: August 22, 1989
Version No.: 1.0
Last Update: February 12, 1990

GMS Engineering Corporation
8940-D Route 108
Columbia, MD 21045
(301) 995-0508

-----------------------
SP,SFRS,REGISTERS

-----------------------
EXTERNAL MEMORY

-----------
0000
0100

-----------
SP,SFRS,REGISTERS

-----------
EXTERNAL MEMORY

-----------
0FFE

-----------
I/O PORTS 3 & 4

-----------
LOW INTRPT VECTORS

-----------
RESERVED (INTEL)

-----------
RESERVED (INTEL)

-----------
RESERVED (INTEL)

-----------
EPROM SECURITY KEY

-----------
HIGH INTRPT VECTORS

-----------
RESERVED (INTEL)

-----------
CODE

-----------
6000
Port pin assignments

<table>
<thead>
<tr>
<th>Pin</th>
<th>Direction</th>
<th>Function</th>
<th>P1.x=1</th>
<th>P1.x=0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.0</td>
<td>output</td>
<td>Coin</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>P1.1</td>
<td>output</td>
<td>Stim1</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>P1.2</td>
<td>output</td>
<td>Stim2</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>P1.3</td>
<td>output</td>
<td>Relay Address (LSB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.4</td>
<td>output</td>
<td>Relay Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.5</td>
<td>output</td>
<td>Relay Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.6</td>
<td>output</td>
<td>Relay Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.7</td>
<td>output</td>
<td>Relay Address (MSB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.5</td>
<td>output</td>
<td>LED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2.2</td>
<td>input</td>
<td>Event Trigger In</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input Channels

<table>
<thead>
<tr>
<th>#</th>
<th>EEG</th>
<th>Input Channels</th>
<th>Stored in data_buffer[#][1]</th>
<th>Output Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EEG</td>
<td>uEEG(t)</td>
<td></td>
<td>ETO(t)</td>
</tr>
<tr>
<td>1</td>
<td>HEOG</td>
<td>HEOG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>VEOGu</td>
<td>VEOGu</td>
<td>cEEG(t-T)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>VEOGl</td>
<td>VEOGl</td>
<td>ETO(t-T)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Event Trigger Out</td>
<td>Event Trigger Out</td>
<td>HEOG(t-T)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>VEOGu</td>
<td>VEOGu(t-T)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>VEOGl</td>
<td>VEOGl(t-T)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This routine must be linked with:

e_int.obj (macro El)
CCR.ABS (cbb)
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 &
ra(1AH-1FFDH(STACK),6000H-0DFFFH)
rtine_flag = 0 = real-time
rtine_flag = 1 = interrogation
rtine_flag = 2 = collection of epochs for fine tuning
rtine_flag = 3 = correction
rtine_flag = 7 = real-time with cal pulses
rtine_flag = 99 = stop

*/

/* Headers */
#include <80C196.h> /* 80C196 I/O registers */
#include <ctype.h>
#include <setjmp.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Definitions */
#define PRESCALE 64
#define BUFF_START 64
#define BUFF_DIFF 384 /* 0.75*SAMP_NUM */
#define CHAN_BASE 0x08 /* ADC Channel base (ADCO) */
#define CHAN_NUM 4 /* number of input (ADC) channels */
#define CHAN_MIN 3 /* CHAN_NUM-1 */
#define CHAN_MAX 5 /* CHAN_NUM+1 */
#define DELTA_T 0x0E371
#define LEFT_LIM 128 /* 0.25*SAMP_NUM */
#define RIGHT_LIM 383 /* (0.75*SAMP_NUM)-1 */
#define SAMP_NUM 512 /* number of samples */
#define TRO_PULSE 4 /* 30 msec */
#define CAL_NUM 60 /* cal/stim samples in buffers */
#define CAL_NUM_MIN 59 /* CAL_NUM-1 */
#define CAL_ON 39 /* Stim on */
#define CAL_OFF 19 /* Stim off */
#define CAL_REP 60 /* cal/stim repetitions for avg */
#define DELTA_T_FAST 0x0F8DE
#define SAMP_LIM 200 /* sample time out limit */
#define SAMP_X2 1024
#define LOW_BATT 650
#define E_LOW_BATT 600
#define V 0
#define R 1
#define I 1

/* Interrupt service function assignments */
#pragma interrupt (nmi_int=31) /* NMI interrupt */
#pragma interrupt (extint=29) /* EXTINT Pin interrupt */
#pragma interrupt (receive=25) /* Serial Port Receive interrupt */
#pragma interrupt (samp=1) /* A2D CONVERSION COMPLETE interrupt */
#pragma interrupt (time1=0) /* TIMER1 OVERFLOW interrupt */

/* Function declarations */
void main(void);
void rint(void); /* Interrupt Service Routines */
void extint(void);
void receive(void);
void time1(void);
void samp(void);
void serial(void);
void senddata(int);
int readdata(void);
void dac(int,int);
void err(void);
void fft_for(void);

/* External functions */
extern void enable_int(void);

/* SFF Images */
unsigned char im_loc0, im_loc1, im_loc2, im_sp_slot, import1, pmm;
register unsigned char status_temp; /* defined in USER.LIB */

/* Global variables */
register long int ttemp1, ttemp2, ttemp3, ttemp4, tdenom;
register int gain_mag, new_pt, alpha1, alpha2, sgmag, sghmag, tune_pt, temp1_reg,
temp2_reg;
register char chan, buff_num, buff_not, chan_end;
register int dv[8];
int cal_counter, cal_mode, rep_count, cal_monitor, cal_cnt, correct_cnt;
char timer_flag, r_flag, correct_flag, error, loop_flag;
char restart_flag, tro_counter, rtime_flag;
int battlevel, battcnt, i_temp;
int xreal[512], ximag[512];
long int lltemp1, lltemp2, lltemp3, lltemp4, denom;
int out_buffer[2][SAMP_NUM];
int alpha[SAMP_NUM];
int eegf[2][2][SAMP_NUM];
int data_buffer[CHAN_P1][2][SAMP_NUM], batt_volt[3];
int data_tuner[CHAN_NUM][SAMP_K2];
int gsum1[CAL_NUM];
int gssum2[CAL_NUM];
int gs[3][2][CAL_NUM];

/* Messages */
const char mess1[] = "Type ENTER to continue...
\n";
const char cls[] = {\'\033\',\'\',\'2',\'J\'};
const char bell = \'007\';
const char stim_port[] = (0x02, 0x04, 0x01); /* i/o lines for stim/cal */

/* Square Root Table */
const int sqrt[] = (100, 121, 144, 169, 196, 225, 256, 289,
324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900,
961, 1024, 1089, 1156, 1225, 1296, 1369, 1444, 1521);

/* Super Gaussian Window, power = 0 */
const int supgau[] = (16, 20, 26, 32, 40, 49, 61, 74,
90, 108, 129, 154, 182, 214, 251, 293,
339, 391, 449, 514, 585, 662, 747, 840,
940, 1048, 1165, 1289, 1423, 1564, 1714, 1872,
2039, 2213, 2396, 2587, 2785, 2991, 3203, 3422,
3647, 3879, 4115, 4357, 4603, 4853, 5106, 5363,
5622, 5883, 6164, 6409, 6674, 6938, 7202, 7465,
7728, 7988, 8246, 8503, 8756, 9006, 9253, 9497,
9736, 9972, 10203, 10429, 10651, 10869, 11081, 11288,
11490, 11687, 11878, 12065, 12245, 12421, 12591, 12756,
12916, 13070, 13220, 13364, 13503, 13657, 13767, 13891,
14011, 14126, 14237, 14343, 14445, 14543, 14636, 14726,
14812, 14984, 15042, 15048, 15120, 15188, 15253, 15316,
15375, 15431, 15485, 15536, 15585, 15631, 15675, 15717,
15756, 15794, 15829, 15863, 15895, 15925, 15953, 15980,
16006, 16030, 16052, 16074, 16094, 16113, 16131, 16148,
16164, 16179, 16193, 16206, 16218, 16230, 16241, 16251,
16260, 16269, 16278, 16285, 16293, 16300, 16306, 16312,
16317, 16323, 16327, 16332, 16336, 16340, 16343, 16347,
16350, 16353, 16355, 16358, 16360, 16362, 16364, 16366,
16367, 16369, 16370, 16371, 16373, 16374, 16375, 16376,
16376, 16377, 16378, 16378, 16379, 16380, 16380, 16380,
16381, 16381, 16381, 16382, 16382, 16382, 16382, 16383,
16383, 16383, 16383, 16383, 16383, 16383, 16383, 16384,
16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
16384, 16384, 16384, 16384, 16384, 16384, 16384, 16384,
void main(void)
{
    int i, j, k;

    /* System Configuration */

    int_mask = 0x00;  /* mask all interrupts */
    imask1 = 0x00;   /* enable global interrupts */
    im_iocl = 0x00;  /* set I/O control registers */
    im_ioc0 = im_ioc0;
    im_ioc1 = im_ioc1;
    im_ioc2 = im_ioc2;
    import1 = 0x00;  /* initial Port1 config. */
    lport1 = import1;
    pmm = 127;
    pwm_control = pmm;
119 1 /* Set up Serial Port */
119 1
120 1 iocl = im_iocl;
121 1 /* select TXD on P2.0 */
122 1 baud_rate = 0x4D;
123 1 /* baud rate of 9600 on 12 MHz */
124 1 sp_con = 0x09;
125 1 /* Mode 1, enable receive, no parity */
126 1 wsr = 0x0F;
127 1 /* alternate window */
128 1 im_sp_stat = 0x20;
129 1 /* set the initial TI bit */
130 1
131 1 restart_flag = 1;
132 1
133 1 /* Initialize flags & pointers */
133 1
134 1
135 1 if ( restart_flag == 1 ) {/*
136 1 restart_flag = 0;
137 1 error = 0;
138 1 correct_flag = 0;
139 1 correct_cnt = 2;
140 1 trow_counter = 0;
141 1 cal_counter = 0;
142 1 cal_cnt = 0;
143 1 cal_monitor = 0;
144 1 battlevel = 0;
145 1 batcnt = 0;
146 1 new_pt = BUFF_START; /* init. data buffer pointers */
147 1 tune_pt = 0;
148 1 buff_num = 0;
149 1 buff_not = 1;
150 1
151 1 for ( i=0; i < 8; i++)
152 1 dec( i, 0 ); /* zero DACs */
153 1
154 1 /* Clear buffers */
155 1
156 1 for ( i=0; i < 2; i++) {
157 1 for ( j=0; j < SAMP_NUM; j++) {
158 1 for ( k=0; k < CHAN_P1; k++)
159 1 data_buffer[k][j][i] = 0;
160 1 /* zero output buffer */
161 1 }
162 1 }
163 1 }
164 1
165 1 for ( i=0; i<CHAN_NUM; i++ ) {
166 1 data_tuner[i][j] = 0;
167 1 }
168 1 /* */
169 1
170 1 for ( i_temp=0; i_temp<256; i_temp++ ) {
for ( j=0; j<8; j++ )
    dac(j, tMem-128);
for ( k=0; k<10; k++ )
    
    for ( i=0; i < 8; i++)
    dac( i, 64 );

/* Go to main menu */
serial();
loop_flag = 0;
ri_flag = 0;
ipendl &= -0x02;

/* Initialize Timer1 */
im_iocl |= 0x04; /* enable TIMER1 overflow intrpt */
loc1 = im_iocl;
timer1_flag = 0; /* timer1 flag */
loop_flag = 0;
wsr = 0x0F; /* alternate window */
timer1 = DELTA_T; /* load 7.813 msec timer */
wsr = 0;
int_mask |= 0x03; /* unmask TIMER1, A2D DONE, */
imask1 |= 0x22; /* EXTINT, and RI intrpt */

/* Loop endlessly............... */

/* Wait for Timer1 Overflow (every 7.813 msec) to start another sampling
sequence. The TIMER1 interrupt handling routine starts the CHAN_NUM channel
sweep: First the last data point is written out to the DAC, then each ADC is
sampled on an interrupt driven basis. */

next:

/* Has the serial port received a character? */

if ( (ri_flag == 1) | (rtine_flag == 99) ) (  
    int_mask &= -0x03; /* mask TIMER1 & A2D DONE intrpts */
    imask1 &= -0x22; /* mask EXTINT & RI intrpts */
    serial();
    loop_flag = 0;
    ri_flag = 0;
    correct_flag = 0;
    correct_cnt = 2;
    timer1_flag = 0;
    ipend1 &= -0x02;
wsr = 0x0F; /* alternate window */
timer1 = DELTA_T;
wsr = 0;
int_mask[0] = 0x03; /* unmask TIMER1 & A2D DONE intrpts */
imask[1] = 0x22; /* unmask EXTINT & RI intrpts */
goto wait;

battlevet = 0;
for (i=0; i<3; i++){
    if (batt_volt[i] < LOW_BATT) 
battlevet = 300;
    if (batt_volt[i] < E_LOW_BATT) 
battlevet = 70;
}

if (restart_flag == 1) 
goto restart;
if (correct_flag != 1) 
goto wait;

/* Load EOG-VU, window the data, multiply by PRESCALE, & transform */
for (itemp=0; itemp < 512; itemp++) {
xreal[i_temp] = (int)(((long)data_buffer[2][buff not][i_temp])*supgau[i_temp])/256;
ximag[i_temp] = 0;
}
fft_for();
if (error != 0) {
    err();
    goto restart;
}

/* Save VU-EOG(w) */
for (i_temp=0; i_temp < 512; i_temp++) {
eog[0][0][i_temp] = (int)(((long)xreal[i_temp]*10)/PRESCALE);
eog[0][1][i_temp] = (int)(((long)ximag[i_temp]*10)/PRESCALE);
}

/* Load EOG-VL, window the data, multiply by PRESCALE, & transform */
xreal[i_temp] = (int)(((long)data_buffer[3][buff not][i_temp])*supgau[i_temp])/256;
ximag[i_temp] = 0;
fft_for();
if (error != 0) {
    err();
    goto restart;
}
C-96  COMPILER EEG
04/06/90 15:25:37  PAGE 10

/* alpha(w) = EOG-VU(w) / EOG-VL(w); alpha(w) is scaled by 10 */

for ( i_temp=0; i_temp < SAMPNUM; i_temp++ ) {
  xreal[i_temp] = (int)(((long) xreal[i_temp] * 10) / PRE_SCALE);
  ximag[i_temp] = (int)(((long) ximag[i_temp] * 10) / PRE_SCALE);

  lltemp1 = gain_mag * (eogf[0][0][i_temp] * (long) xreal[i_temp])
           + (eogf[0][1][i_temp] * (long) ximag[i_temp]);
  denom = ((long) xreal[i_temp]) * xreal[i_temp] +
          ((long) ximag[i_temp]) * ximag[i_temp]) * 10;

  if ( lltemp1 < 0 )
    lltemp1 = -lltemp1;
  else {
    lltemp2 = lltemp1 / denom;
    if ( lltemp2 > 40 )
      lltemp2 = 40;
    if ( lltemp2 < 10 )
      lltemp2 = 10;
    xreal[i_temp] = (int) lltemp2;
  }

  alpha[i_temp] = xreal[i_temp];
}

/* Load EOG-H(L-R), window the data, multiply by PRE_SCALE, & transform */

for ( i_temp=0; i_temp < SAMPNUM; i_temp++ ) {
  xreal[i_temp] = (int)(((long) data_buffer[1][buff_not][i_temp]*supgau[i_temp])/256);
  ximag[i_temp] = 0;
}

fft_for();
if ( error != 0 )
  err();
goto restart;

for ( i_temp=0; i_temp < SAMPNUM; i_temp++ ) {
  eogf[1][0][i_temp] = (int)(((long) xreal[i_temp] * 10) / PRE_SCALE);
  eogf[1][1][i_temp] = (int)(((long) ximag[i_temp] * 10) / PRE_SCALE);
}

/* -----------------------------------------------*/

/* Correct the EEG */

for ( i_temp=0; i_temp < SAMPNUM; i_temp++ ) {
\begin{verbatim}
C-96 COMPILER EEG
  04/06/90 15:25:37 PAGE 11

246 2 alpha1 = (alpha[i_temp] + 10)/2;    /* alpha + 1 */
247 2 alpha2 = (alpha[i_temp] - 10)/2;    /* alpha - 1 */
248 2 */
249 2 /* (alpha+1)*D1v + (alpha-1)*D2v */
250 2 DSGv(w) = ----------------- * SGv(w)
251 2 (alpha+1)*D3v + (alpha-1)*D4v
252 2 */
253 2
254 2 lltemp1 = (((long) dv[0])*alpha1) + (((long) dv[1])*alpha2);
255 2 lltemp2 = (((long) dv[2])*alpha1) + (((long) dv[3])*alpha2);
256 2 if ( lltemp2 == 0 ) {
257 3    error = 100;
258 3    err();
259 3    goto restart;
260 3 }
261 3 */
262 2 /* DSGv(w) * EOGv(w) */
263 2 lltemp3 = lltemp1 * eogf[0][0][i_temp];
264 2 lltemp4 = lltemp1 * eogf[0][1][i_temp];
265 2 xreal[i_temp] = (int) ( lltemp3 / lltemp2 & 65535 );
266 2 ximag[i_temp] = (int) ( lltemp4 / lltemp2 & 65535 );
267 2 */
268 2 /* (alpha+1)*D1h + (alpha-1)*D2h */
269 2 DSGh(w) = ----------------- * SGh(w)
270 2 (alpha+1)*D3h + (alpha-1)*D4h
271 2 */
272 2
273 2 lltemp3 = ((long) dv[4])*alpha1) + (((long) dv[5])*alpha2);
274 2 lltemp4 = ((long) dv[6])*alpha1) + (((long) dv[7])*alpha2);
275 2 if ( lltemp2 == 0 ) {
276 3    error = 101;
277 3    err();
278 3    goto restart;
279 3 }
280 3 */
281 2 /* DSGh(w) * EOGh(w) */
282 2 lltemp3 = lltemp1 * eogf[1][0][i_temp];
283 2 lltemp4 = lltemp1 * eogf[1][1][i_temp];
284 2 */
285 2 /* (DSGv(w) * EOGvu(w)) + (DSGh(w) * EOGh(w)) */
286 2
287 2 xreal[i_temp] += (int) ( lltemp3 / lltemp2 & 65535 );
288 2 ximag[i_temp] += (int) ( lltemp4 / lltemp2 & 65535 );
289 2 */
290 2 /* Complex Conjugate */
291 2
292 2 ximag[i_temp] = -ximag[i_temp];
293 2 */
294 2 /* Inverse Transform */
\end{verbatim}
269 1  fft_for();                           /* inverse FFT */
270 1  if (error != 0) {
271 2  err();
272 2  goto restart;
273 2  }
274 1  /* Remove Window */
275 1  for (i_temp=BUFF_START; i_temp < (SAMP_NUM-BUFF_START); i_temp++) {
276 2  xreal[i_temp] = (int)((long)xreal[i_temp] * 83886) / supgau[i_temp];
277 2  }
278 1  /* Moving Average Filter (7 point) of corrector */
279 1  for (i_temp=BUFF_START; i_temp < (SAMP_NUM-BUFF_START); i_temp++) {
281 2  }
282 1  /* EEGcorr(t) = EEGobs(t) - IFFT((SGv * EOGv) + (SGh * EOGh)) */
283 1  for (i_temp=BUFF_START; i_temp < (SAMP_NUM-BUFF_START); i_temp++) {
284 2  out_buffer[buff_not][i_temp] = data_buffer[0][buff_not][i_temp] * alpha[i_temp];
285 2  }
286 1  correct_flag = 0;
287 1  wait:
288 1  loop_flag--;  
289 1  while (timer1_flag == 0)  
290 1   ;  
291 1  timer1_flag = 0;
292 1  
293 1  /* Loop again */
294 1  goto next;                           /* wait for another TIMER1 INTERRUPT */
295 1  
296 1  /*-------------------------------------------------------------------*/
297 1  /* nmi_int function - NMI interrupt handler */
298 1  void nmi_int(void)
299 1  {
300 1    imask1 = 0x00;  
301 1    int_mask = 0x00;  
302 1    error = 3;  
303 1    err();
304 1  }
/* extint Function - EXTINT interrupt handler */

void extint(void)
{
  tro_counter = TRO_PULSE;    /* load TRO counter */
}

/* Receive Function - Serial Port Receive interrupt handler */

void receive(void)
{
  ri_flag = 1;
  getchar();
}

/* TIME1 - TIMER1 overflow interrupt handler 
    - Operates in conjunction with SAMP */

void timel(void)
{
  int i, j, k;

  /* Reset TIMER1 */

  if (rtine_flag != 1) {
    wsr = OxOF;
    timer1 = DELTA_I;
    wsr = 0;
  }
  else {
    wsr = OxOF;
    timer1 = DELTA_I_FAST;
    wsr = 0;
  }

  if ( (loop_flag > 0) & (run_flag != 3) ) {
    error = 88;
    err();
  }

  /* Is TRO high? */

  if ( tro_counter > 0 )
    tro_counter--;
battcnt++;  
if (battcnt >= 32000)  
battcnt = 0;  
if (battlevel i == 0) {  
  if ((battcnt % battlevel) == 0) {  
    if (((battcnt / battlevel) % 2) == 0)  
      pwm_control = pwm;  
    else  
      pwm_control = 0;  
  }  
  }  
}  
/* Interrogate */  
if (rtine_flag == 1) {  
i = CAL_NUM_M1 - cal_counter;  
gssum1[i] += data_buffer[1](i)/CAL__mode*2][buff_num][new_pt+1];  
gssum2[i] += data_buffer[1](i)/CAL__mode*2][2-(cal_mode/2)][buff_num][new_pt+1];  
if (cal_counter > CAL_ON) {  
  import1 &= -stim_port[3-cal_mode];  
  ioport1 = import1;  
}  
else if (cal_counter > CAL_OFF) {  
  import1 &= -stim_port[3-cal_mode];  
  ioport1 = import1;  
}  
else if (cal_counter == 0) {  
  import1 &= -stim_port[3-cal_mode];  
  ioport1 = import1;  
}  
cal_counter--;  
if (cal_counter < 0) {  
cal_counter = CAL_NUM_M1;  
rep_count--;  
if (rep_count == 0) {  
  int_mask &= -0x01;  
  /* mask TIMER1 intrpt */  
cal_mode--;  
if (cal_mode == 0) {  
  rtine_flag = 2;  
tune_pt = 0;  
new_pt = BUFF__START;  
buf_num = 0;  
buf_not = 1;  
}  
}  
rep_count = CAL_REP;  
/* reset average counter */  
for (i=0; i < CAL_NUM; i++) {  
  /* average */  
gs[2-cal_mode][0][1] = (gssum[i] / CAL_REP);  
gs[2-cal_mode][1][1] = (gssum2[i] / CAL_REP);  
}  
temp1_reg = (gs[2-cal_mode][0][CAL_NUM_M1] + gs[2-cal_mode][0][0]) / 2;  
temp2_reg = (gs[2-cal_mode][1][CAL_NUM_M1] + gs[2-cal_mode][1][0]) / 2;  
for (i=0; i < CAL_NUM; i++) {
gs[2-cal_mode][0][i] = temp1_reg;
gs[2-cal_mode][1][i] = temp2_reg;
gssum[i] = 0;
gssum2[i] = 0;

loop_flag = 0;

int_mask | = 0x01;
goto skiploop;

if (rtine_flag == 7) {
    if (cal_cnt > 100) {
        import1 &~=(stim_port[2]);
        ioport1 = import1;
    }
    else if (cal_cnt > 90) {
        import1 |=(stim_port[2]);
        ioport1 = import1;
    }
    else if (cal_cnt <= 0) {
        import1 |=(stim_port[2]);
        ioport1 = import1;
    }
    cal_cnt--;
}

if (cal_cnt < 0) {
    cal_cnt = 150;
    cal_monitor--;
    if (cal_monitor == 0) {
        int_mask | = -0x01;
        rtine_flag = 0;
        printf("\n\n\n\n\n REAL TIME MONITORING...\n\n\n\n\n");
        loop_flag = 0;
        int_mask | = 0x01;
        goto skiploop;
    }
}

*/ Initiate the first A/D Conversion */

chan = 0; /* load ADC chan offset */
int_mask | = 0x02;
ad_command = CHAN_BASE; /* start 1st ADC conversion */

loop_flag = 2;

skiploop:
timer1_flag = 1;

/* ................................................................. */
/* SAMP - A2D CONVERSION COMPLETE interrupt handler. 
   * Operates in conjunction with TIME1. 
   * The last sample in the data_buffer is written out to the 
   * corresponding DAC, converted to 8 bits. 
   * reads the sample from the ADC. 
   * returns a 10 bit value. 
   * start a conversion on the next channel of the sweep. */

void samp(void)
{
    register int temp_reg;

    /* Write out the data to the DAC 
       * convert raw data from 10 bits to 8 bits; */

    if (chan == 0) {
        dac(0, (data_buffer[0][buff_num][new_pt-1] >> 2));
        if (runtime_flag == 3) {
            dac(1, (data_buffer[1][buff_num][new_pt-1] >> 2));
            dac(2, (out_buffer[buff_num][new_pt] >> 2));
            dac(3, (data_buffer[3][buff_num][new_pt] >> 2));
            dac(4, (data_buffer[4][buff_num][new_pt] >> 2));
            dac(5, (data_buffer[5][buff_num][new_pt] >> 2));
            dac(6, (data_buffer[2][buff_num][new_pt] >> 2));
            dac(7, (data_buffer[3][buff_num][new_pt] >> 2));
        }
        else {
            dac(1, (data_buffer[1][buff_num][new_pt-1] >> 2));
            dac(2, (data_buffer[2][buff_num][new_pt-1] >> 2));
            dac(3, (data_buffer[3][buff_num][new_pt-1] >> 2));
        }
    }
    else if (chan < 7) {
        temp_reg = ad_result_hi;
        temp_reg = ((temp_reg << 8) + ad_result_lo) >> 6;
        data_buffer[chan][buff_num][new_pt] = temp_reg;
    }
    else if (chan == 3) {
        data_buffer[4][buff_num][new_pt] = -200;
        if (tro_counter > 0)
            data_buffer[4][buff_num][new_pt] = 200;
    }

    /* If necessary, store it in the other buffer as well */

    if ((runtime_flag == 2) & (chan < CHANNUM))
        data_tuner[chan][tune_pt] = data_buffer[chan][buff_num][new_pt];
if (new_pt < LEFT_LIM)
  data_buffer[chan][buff_not][new_pt+BUFF_DIFF] = temp_reg;
else if (new_pt > RIGHT_LIM)
  data_buffer[chan][buff_not][new_pt-BUFF_DIFF] = temp_reg;

/* Either start a new conversion, or update the data pointers */

if (rtine_flag == 1)
  chan_end = 8;
if (chan < chan_end)
  ad_command = CHAN_BASE + chan; /* start next conversion */
else /* Update data buffer pointers */
  if (new_pt < (SAMP_NUM-BUFF_START))
    new_pt++;
    tune_pt++; 
    if ( (tune_pt == SAMP_X2) & (rtine_flag == 2) )
      rtine_flag = 99;
    if (rtine_flag == 3)
      if (new_pt == 129)
        if (correct_flag == 1)
          error = 80;
          err();
        else
          /*
          Serial Port Communications Function
          */
          void serial(void)
          {
int resp, i, j, k, resptmp, jk, tempwait;
int_mask = 0;
pwm_control = pmwm;
if ( rtine_flag == 98 )
goto menuu;
if ( rtine_flag == 99 ){
    printf("Press 'PgDn', select '7', and then type 'drive'.
        Then type upper case 'A'.
    ");
    printf("Interrogation data will take fifteen minutes to download."
    ");
    sndgs:
    resp = getchar();
    printf("Xc",resp);
    if ( resp != 0 )
        goto sndgs;
    for ( i=0; i<3; i++ ){
        for ( j=0; j<2; j++ ){
            for ( k=0; k<5; k++ )
                senddata(data_tunel[i][j][k]);
            tempwait = 0;
        }
    }
    for ( i=0; i<CHAN_NUM; i++ ){
        for ( j=0; j<SAMP_X; j++ ){
            senddata(data_tunel[i][j]);
            tempwait = 0;
        }
    }
    printf("Push 'ESC', ALT-X, 'Y', and then run the program EEG."
    ");
    goto menuu;
}
menu:
printf("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 > ");
resp = getchar();
printf("Xc",resp);
if ( (resp == 'L') | (resp == 'l') )
    lv14:
    resptmp = pmwm + 1;
    printf("Current light level is: %d Enter new level (1-255) > ",resptmp);
    resptmp = 0;
    lv11l;
508     resp = getchar();
509     printf("%c", resp);
510     if (resp == 13)
511         goto lvl1;
512     if ((resp<48) | (resp>58))
513         goto lvl1;
514     resptmp = (resptmp*10) + (resp-48);
515     goto lvl11;
516 
517     lvl1:
518     if (resptmp > 255)
519         goto lvl4;
520     if (resptmp == 0)
521         resptmp = pwm + 1;
522     pwm_control = resptmp;
523     
524     else if ((resp == 'R') | (resp == 'r')) {
525         printf("REAL TIME MONITORING...
526         
527         
528         
529         rtine_flag = 0;
530         goto menu_end;
531     }
532     
533     else if ((resp == 'P') | (resp == 'p')) {
534         printf("Calibrate on Pulses
535         
536         
537         cal_monitor = 300;
538         cal_cnt = 150;
539         goto menu_end;
540     }
541     
542     else if ( (resp == 'I') | (resp == 'i') ) {
543         cal_counter = CAL_NUM_N1;
544         cal_mode = 3;
545         rep_count = CAL_REP;
546         rtine_flag = 1;
547         for (i=0; i<8; i++) {
548             gssum1[i] = 0;
549             gssum2[i] = 0;
550         }
551         printf("INTERROGATING...
552         
553         
554         
555         resptmp = 0;
556         goto menu_end;
557     }
558     
559     else if ( (resp == 'C') | (resp == 'c') ) {
560         printf("Press 'PgUp', select '7', and then type 'correct'.
561         
562         
563         sgv_mag = recvdata();
564         sgh_mag = recvdata();
565         for (i=0; i<8; i++)
566             dv[i] = recvdata();
567             dv[0] = (int)((long) dv[0] * sgv_mag / 10) & 65535;
569             rtine_flag = 3;
570             printf("CORRECTING...
571         
572         
573         
574         resptmp = 0;
575         goto menu_end;
576     }
560 2     goto menu_end;
561 1     else if ( (resp == 'N') | (resp == 'n')) {
563 2     chann4:
564 2     resptmp = (import1 >> 3) + 1;
568 2     printf("\n\rCurrent channel number is: %d\n\rEnter new channel number (1-24) >
573 1     resptmp = 0;
574 2     chann:  
576 2     resp = getchar();
577 2     printf("%c",resp);
579 2     if ( (resp == 13) )
580 2     goto chann;
581 2     if ( (resp<48) | (resp>=58) )
582 2     goto channn;
583 2     resptmp = (resptmp*10) + (resp-48);
584 2     goto chann;
585 2     chann:
586 2     if ( resptmp > 24 )
587 2     goto chann4;
588 2     if ( resptmp == 0 )
589 2     resptmp = (import1 >> 3) + 1;
590 2     resptmp+=;
591 2     import1 &= 0x07;
592 2     import1 |= (resptmp << 3);
598 2     goto menu;
599 1     goto menu;
600 1 }
601 1 }
602 1 menu_end;
603 1 ;
604 1 }
605 1
606 1 /*---------------------------------------------*/
607 1 /*  SENDOUT DATA Function */
608 1 void senddata(data_out)
609 1
610 2 int     data_out;
611 2
612 1 {
613 1     int     i, j, temp_data, temp_sign, data2_out, data_first;
614 1     char    temp_chr;
615 1     temp_sign = 0;
616 1     if ( data_out < 0 ) {
617 2     temp_sign = 1;
618 2     data_out = -data_out;
619 2 }
620 1     data2_out = 0;
621 1     data_first = data_out;
622 1     for ( i=4; i>0; i-- ) {
623 2     data_first -= data2_out;
624 2     temp_data = data_first;
625 2     for ( j=1; j>0; j-- )
626 2     temp_data /= 10;
data2_out = temp_data;
for ( j=i; j>0; j-- )
data2_out *= 10;
printf("%d", temp_data);
}
printf("%d\n", temp_sign);
}

/**************************************************************************/
 RECV DATA Function */

int recvdata(void)
{
    int i, j, temp_data, data_in;
    char temp_chr;
    temp_data = 0;
    for ( i=4; i>=0; i-- )
    {
       retry1:
        temp_chr = getchar();
        if ( temp_chr == 32 )
            temp_chr = 48;
        if ( (temp_chr < 48) | (temp_chr >= 58) )
            goto retry1;
        data_in = (int) temp_chr - 48;
        for ( j=0; j<i; j++ )
            data_in *= 10;
        temp_data += data_in;
        retry2:
        temp_chr = getchar();
        if ( (temp_chr < 48) | (temp_chr > 49) )
            goto retry2;
        if ( temp_chr == 49 )
            temp_data = -temp_data;
        return ( temp_data );
    }

/**************************************************************************/
 DAC Function */

void dac(channel, level)
{
    int channel, level;
    (*/

int dacc;

/* Decode DAC address */
if ( channel <= 3 )
dacc = 0xE013 + channel;
else
dacc = 0xE007 + channel;
if ( (dacc % 2) == 0)
dacc -= 2;

/* Write out the sample to the DAC */
if ( level > 127 )
level = 127;
else if ( level < -128 )
level = -128;
memset( dacc, (level+128), 1 );
/* +/- 128 into 0 - 256 */

/*...........................................................................*/
/* Error function */
void err(void)
{
    int i;
    int_mask = 0x00; /* mask all interrupts */
imask1 = 0x02;
for ( i=0; i < CHAN_NUM; i++ )
    dac( i, 0 ); /* zero DACs */
printf(c(ts);
    printf("\n\n\t\t\tERROR\n\n\t\t\tNo.\%i\n",error);
    printf("\n\t\t\t%\n",error);
imask1 &= -0x03; /* mask TIMER1 & A2D DONE intrpts */
imask1 &= -0x22; /* mask EXTINT & RI intrpts */
rtine_flag = 98;
serial();
ri_flag = 0;
ipend1 &= -0x02;
int_mask |= 0x03; /* unmask TIMER1 & A2D DONE intrpts */
imask1 |= 0x22; /* unmask EXTINT & RI intrpts */
restart_flag = 1;
}
MODULE INFORMATION:

<table>
<thead>
<tr>
<th>AREA</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE AREA SIZE</td>
<td>16DFH</td>
</tr>
<tr>
<td>CONSTANT AREA SIZE</td>
<td>080EH</td>
</tr>
<tr>
<td>DATA AREA SIZE</td>
<td>6FDBH</td>
</tr>
<tr>
<td>STATIC REGS AREA SIZE</td>
<td>005AH</td>
</tr>
<tr>
<td>OVERLAYABLE REGS AREA SIZE</td>
<td>000AH</td>
</tr>
<tr>
<td>MAXIMUM STACK SIZE</td>
<td>0094H</td>
</tr>
</tbody>
</table>

C-96 COMPILATION COMPLETE. 0 WARNINGS, 0 ERRORS
SOURCE FILE: FFT.FOR.A96
OBJECT FILE: FFT.FOR.OBJ

ERR LOC  OBJECT   LINE    SOURCE STATEMENT

0000     RSEG   1      FFT_FO   MODULE STACKSIZE(6)
0002     2      ;FFT ALGORITHM FROM INTEL APPLICATIONS NOTE, AP-275, BY IRA HORDON

0000     4      RSEG   1
0002     5      EXTRN   error
0006     6      OSEG   1
0024     7      OSEG at 24H
0024     8      TMPR:   dsl   1
0028     9      TMPI:   dsl   1
0032     10     TMP1:   dsl   1
0036     11     TMPR1:  dsl   1
0040     12     TMPI1:  dsl   1
0044     13     XRTMP:  dsl   1
0048     14     TMPI2:  dsl   1
0052     15     XIIMP:  dsl   1
0056     16     WRP:   dsw   1
0060     17     WIP:   dsw   1
0064     18     PUR:   dsw   1
0068     19     IN_CNT: dsw   1
0072     20     NDIV2: dsw   1
0076     21     INP:   dsw   1
0080     22     KPTR:  dsw   1
0084     23     K22:   dsw   1
0088     24     N_SUB_K: dsw   1
0092     25     RK:    dsw   1
0096     26     RNK:   dsw   1
00A0     27     SHF_CNT: dsb   1
00A4     28     LOOP_CNT: dsb   1

0000     29     DSEG   1
0002     30     EXTRN   XREAL, XIMAG
0005     31     ; XREAL, XIMAG: Base addresses for 512 16-bit signed
0007     32     ; entries for real and imaginary numbers, respectively.
0009     33     $EJECT
ERR LOC  OBJECT   SOURCE STATEMENT
            LINE
0000        CSEG
39          PUBLIC fft_for; Starting point for FFT algorithm
40          ; ; ; ; ; START FOURIER CALCULATIONS
41          ; ; ; ; ; 400 'INITIALIZATION OF LOOP
42
0000 1100   cLB error
43
0002 FC     cLRT
44          ldb loop_cnt, #1
0003 B10151 45          ldb shft_cnt, #8
0006 B10850 46          ld ndiv2, #512
47
0009 A1000244 48          ; ; ; ; ; 410 K=0
49
000D        OUT_LOOP:
50          clr kptr
51          cmpb loop_cnt, #9 ; 912 = 2^9
52
000F 990951 53          bgt UNWEAVE
54
0012 DA0220A3 55          ; ; ; ; ; 420 IF LOOP > EXP THEN 700
56
0016        MID_LOOP:
57          clr incnt
58          ; ; ; ; ; 430 INCNT=0
59
0018        IN_LOOP:
60          add in_cnt, #2 ; 450 INCNT=INCNT+1
61          ; ; ; ; ; 440 'CALCULATIONS BEGIN HERE
62
001C A04660 63          ld pwr, kptr
64          ; ; ; ; ; 470 WRP=WR(P) : WIP=W1(P) : KN2=K+N2
65
001F 0850A0 66          shr pwr,shft_cnt ; Calculate multi factors
67
0022 71FE40 68          andb pwr, #11111110b
69          ; ; ; ; ; 490 TMPR=(WRP*XR(KN2)-WIP*XI(KN2))/2
70          ; ; ; ; ; 490 TMPI=(WRP*XI(KN2)+WIP*XR(KN2))/2
71          ; ; ; ; ; 490 TMPL=WIP=WI(P) : KN2=K+N2
72          ; ; ; ; ; 490 TMPL=WIP=WI(P) : KN2=K+N2
73
0025 A341FC040 74          add kn2, kptr, ndiv2
75          ; ; Complex multiplication follows
76
002A A341FC043C 77          ; ; ; ; ; 480 TMPL=WIP=WI(P) : KN2=K+N2
78          ; ; ; ; ; 480 TMPL=WIP=WI(P) : KN2=K+N2
79          ; ; ; ; ; 480 TMPL=WIP=WI(P) : KN2=K+N2
80
002F A341FE083E 81          ; ; ; ; ; 490 TMPI=(WRP*XI(KN2)+WIP*XR(KN2))/2
82          ; ; ; ; ; 490 TMPI=(WRP*XI(KN2)+WIP*XR(KN2))/2
83          ; ; ; ; ; 490 TMPI=(WRP*XI(KN2)+WIP*XR(KN2))/2
84
0034 44444648 85          ; ; ; ; ; 490 TMPI=(WRP*XI(KN2)+WIP*XR(KN2))/2
86          ; ; ; ; ; 490 TMPI=(WRP*XI(KN2)+WIP*XR(KN2))/2
87
ERR LOC  OBJECT       LINE   SOURCE STATEMENT

86    87    88    89    90    91
005A DC55               BVT    ERR1    ; branch on error
005C A34700002C E 92    ld     tmp1, xreal[kptr] ; 500 TMP11=XR(K)/2
0061 A012C             shra  tmp1, #1   ;  TMP11=XI(K)/2
0064 A347000030 E 94    ld     tmp1, ximag[kptr]
0069 A0130             shra  tmp1, #1   
0090 DC23               BVT    ERR2    ; Branch on error
0092 65020046           ik:    add    kptr, #2
0096 B84442             cmp    in_cnt, ndiv2
0099 D60277B             blt    IN_LOOP
009D 644446           114     ;;;;; 570 IF INCNT < N2 THEN GOTO 450
00A8 1751             118     ;;;;; 570 IF K < N1 THEN GOTO 430
00A9 D0144             cmp    kptr, #1022 ; ; N1 = 2 *(N-1)
00AD 1550             123     blt    MID_LOOP
00A9 1550             124     ;;;;; 600 LOOP = LOOP + 1 : N2 = N2 / 2
00AF 275C             125     ;;;;; 610 GOTO 400
00B1 010100           130     br     OUT_LOOP
00B4 F0             131     ERR1:  ldb    error, #01 ; overflow error
00B5 810200          133     ret
00B8 F0             134     ERR2:  ldb    error, #02 ; overflow error
00B8 F0             137
008E 910100           138     $EJECT
ERR LOC OBJECT LINE SOURCE STATEMENT

00B9 00146 144 clr kptr
008A A100044A 145 ld n_sub_k, #1024
00BF 147 UN_LOOP: ;; Bit reversal of the transformed array
00BF A347FC004C R 149 ld rk, brev[kptr]
00C4 884C46 151 cmp kptr, rk
00CD 6D28 152 bge ENDL
00C9 A347000024 E 154 ld tmpr, xreal[kptr]
00CE A34700002B E 155 ld tmpl, ximag[kptr]
00D3 A340000030 E 157 ld tmprl, xreal[rk]
00D0 C340000024 E 159 st tmpr, xreal[rk]
00D2 C34000002B E 160 st tmpl, ximag[rk]
00D7 C34700002C E 161 st tmprl, ximag[rk]
00D9 C347000030 E 162 st tmpl, ximag[kptr]

ENDL: add kptr, #2

sub n_sub_k, #2

bne UN_LOOP

bge ENDL

mov kptr, #2

add kptr, #2
<table>
<thead>
<tr>
<th>OBJ.</th>
<th>DCW</th>
<th>STMT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>049B</td>
<td>253</td>
<td>2<em>231, 2</em>487, 2<em>23, 2</em>27, 2<em>151, 2</em>407</td>
</tr>
<tr>
<td>04A4</td>
<td>254</td>
<td>2<em>87, 2</em>343, 2<em>215, 2</em>471, 2<em>55, 2</em>311</td>
</tr>
<tr>
<td>04B0</td>
<td>255</td>
<td>2<em>183, 2</em>439, 2<em>119, 2</em>375, 2<em>247, 2</em>503</td>
</tr>
<tr>
<td>04BC</td>
<td>256</td>
<td>2<em>15, 2</em>271, 2<em>143, 2</em>399, 2<em>79, 2</em>335</td>
</tr>
<tr>
<td>04CB</td>
<td>257</td>
<td>2<em>207, 2</em>463, 2<em>47, 2</em>303, 2<em>175, 2</em>431</td>
</tr>
<tr>
<td>04DE</td>
<td>258</td>
<td>2<em>111, 2</em>367, 2<em>239, 2</em>495, 2<em>31, 2</em>287</td>
</tr>
<tr>
<td>04EO</td>
<td>259</td>
<td>2<em>159, 2</em>415, 2<em>95, 2</em>351, 2<em>223, 2</em>479</td>
</tr>
<tr>
<td>04EC</td>
<td>260</td>
<td>2<em>63, 2</em>319, 2<em>191, 2</em>447, 2<em>127, 2</em>383</td>
</tr>
<tr>
<td>04FB</td>
<td>261</td>
<td>2<em>255, 2</em>511</td>
</tr>
<tr>
<td>ERR LOC</td>
<td>OBJECT</td>
<td>LINE</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>0C2E</td>
<td>7C7A057A89779079</td>
<td>424</td>
</tr>
<tr>
<td>0C3A</td>
<td>6877B6764176A575</td>
<td>425</td>
</tr>
<tr>
<td>0C46</td>
<td>5B73077354729071</td>
<td>426</td>
</tr>
<tr>
<td>0C52</td>
<td>5E6F966C960F86C</td>
<td>427</td>
</tr>
<tr>
<td>0C5E</td>
<td>6D6A869A668BC67</td>
<td>428</td>
</tr>
<tr>
<td>0C6A</td>
<td>5B646E61F62F061</td>
<td>429</td>
</tr>
<tr>
<td>0C76</td>
<td>575EC75D835C9CB5</td>
<td>430</td>
</tr>
<tr>
<td>0C8B</td>
<td>42581057F555C954</td>
<td>431</td>
</tr>
<tr>
<td>0CBE</td>
<td>335F66F84E814D</td>
<td>432</td>
</tr>
<tr>
<td>0C9A</td>
<td>5B495981C74DC45</td>
<td>433</td>
</tr>
<tr>
<td>0CDA</td>
<td>CE417340173FB83D</td>
<td>434</td>
</tr>
<tr>
<td>0CB2</td>
<td>8C3924388A364D5</td>
<td>435</td>
</tr>
<tr>
<td>0CBE</td>
<td>FB30872F112E992C</td>
<td>436</td>
</tr>
<tr>
<td>0CCA</td>
<td>2628B8262825A623</td>
<td>437</td>
</tr>
<tr>
<td>0CD6</td>
<td>1A1F931D81C021A</td>
<td>438</td>
</tr>
<tr>
<td>0CE2</td>
<td>E2155514CB123A11</td>
<td>439</td>
</tr>
<tr>
<td>0CEE</td>
<td>8C0CF80A6A09B07</td>
<td>440</td>
</tr>
<tr>
<td>0CF0</td>
<td>240392010000</td>
<td>441</td>
</tr>
<tr>
<td>0D00</td>
<td></td>
<td>442</td>
</tr>
</tbody>
</table>

**Note:** The table lists error locations, object code, line numbers, and corresponding source statements.
# Symbol Table Listing

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREV</td>
<td>00FCH</td>
<td>CODE REL WORD</td>
</tr>
<tr>
<td>ENDL</td>
<td>00F1H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>ERR1</td>
<td>00B1H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>ERR2</td>
<td>0085H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>ERROR</td>
<td></td>
<td>REG EXTERNAL</td>
</tr>
<tr>
<td>FFT_FO</td>
<td>0000H</td>
<td>CODE REL PUBLIC ENTRY</td>
</tr>
<tr>
<td>GM</td>
<td>003BH</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>GR2</td>
<td>006CH</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>GW</td>
<td>002AH</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>GX</td>
<td>0087H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>GX2</td>
<td>0075H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>IK</td>
<td>0092H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>IN_CNT</td>
<td>0042H</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>IN_LOOP</td>
<td>0018H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>KW2</td>
<td>0048H</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>KPTR</td>
<td>0046H</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>LOOP_CNT</td>
<td>0051H</td>
<td>OVERLAY ABS BYTE</td>
</tr>
<tr>
<td>MID_LOOP</td>
<td>0016H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>N_SUB_K</td>
<td>004AH</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>NDIV2</td>
<td>0044H</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>OUT_LOOP</td>
<td>0000H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>PWR</td>
<td>0040H</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>RK</td>
<td>004CH</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>RNK</td>
<td>004EH</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>SHFT_CNT</td>
<td>0050H</td>
<td>OVERLAY ABS BYTE</td>
</tr>
<tr>
<td>TMP1</td>
<td>0028H</td>
<td>OVERLAY ABS LONG</td>
</tr>
<tr>
<td>TMP11</td>
<td>0030H</td>
<td>OVERLAY ABS LONG</td>
</tr>
<tr>
<td>TMPR</td>
<td>0024H</td>
<td>OVERLAY ABS LONG</td>
</tr>
<tr>
<td>TMPW</td>
<td>002CH</td>
<td>OVERLAY ABS LONG</td>
</tr>
<tr>
<td>UN_LOOP</td>
<td>008FH</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>UNWEAVE</td>
<td>0089H</td>
<td>CODE REL ENTRY</td>
</tr>
<tr>
<td>W1</td>
<td>08FEN</td>
<td>CODE REL WORD</td>
</tr>
<tr>
<td>WIP</td>
<td>003EH</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>WR</td>
<td>04FCH</td>
<td>CODE REL WORD</td>
</tr>
<tr>
<td>WRP</td>
<td>003CH</td>
<td>OVERLAY ABS WORD</td>
</tr>
<tr>
<td>XIMAG</td>
<td></td>
<td>DATA EXTERNAL</td>
</tr>
<tr>
<td>XITMP</td>
<td>0038H</td>
<td>OVERLAY ABS LONG</td>
</tr>
<tr>
<td>XREAL</td>
<td></td>
<td>DATA EXTERNAL</td>
</tr>
<tr>
<td>XRTMP</td>
<td>0034H</td>
<td>OVERLAY ABS LONG</td>
</tr>
</tbody>
</table>

Assembly completed, no error(s) found.
$PAGELENGTH(51)
$TITLE("ENABLE 80C196 GLOBAL INTERRUPTS")
E_INTERNAL MODULE STACKSIZE(12)
PUBLIC ENAB_INT

; E_INT.A96

; Version 1.0 July 26, 1989
; Jeffrey C. Sigl

; GMS Engineering Corporation
; 8940-D Route 108
; Columbia, Maryland 21045

; COMMON DEFINITIONS

$INCLUDE(8096.INC) ;80C196 REGISTER DEFINITIONS

;---------------------------------------------------------------------
=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 28 ;

;---------------------------------------------------------------------
=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_L EQU 02H:BYTE ; R
0003 =1 36 AD_RESULT_H 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 ; R
0006 =1 41 HSI_STATUS EQU 06H:BYTE ; R
0007 =1 42 SBUF EQU 07H:BYTE ; R/W
<table>
<thead>
<tr>
<th>ERR LOC</th>
<th>OBJECT</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>=1</td>
<td>43</td>
<td>INT_MASK EQU 0BH:BYTE ; R/W</td>
</tr>
<tr>
<td>0009</td>
<td>=1</td>
<td>44</td>
<td>INT_PENDING EQU 09H:BYTE ; R/W</td>
</tr>
<tr>
<td>000A</td>
<td>=1</td>
<td>45</td>
<td>WATCHDOG EQU 0AH:BYTE ; W WATCHDOG TIMER</td>
</tr>
<tr>
<td>000A</td>
<td>=1</td>
<td>46</td>
<td>TIMER1  EQU 0AH:WORD ; R</td>
</tr>
<tr>
<td>000C</td>
<td>=1</td>
<td>47</td>
<td>TIMER2  EQU 0CH:WORD ; R</td>
</tr>
<tr>
<td>000E</td>
<td>=1</td>
<td>48</td>
<td>BAUD_RATE EQU 0EH:BYTE ; W</td>
</tr>
<tr>
<td>000E</td>
<td>=1</td>
<td>49</td>
<td>IPORT0   EQU 0EH:BYTE ; R</td>
</tr>
<tr>
<td>000F</td>
<td>=1</td>
<td>50</td>
<td>IPORT1   EQU 0FH:BYTE ; R/W</td>
</tr>
<tr>
<td>0010</td>
<td>=1</td>
<td>51</td>
<td>IPORT2   EQU 10H:BYTE ; R/W</td>
</tr>
<tr>
<td>0011</td>
<td>=1</td>
<td>52</td>
<td>SP_CON   EQU 11H:BYTE ; W</td>
</tr>
<tr>
<td>0011</td>
<td>=1</td>
<td>53</td>
<td>SP_STAT  EQU 11H:BYTE ; R</td>
</tr>
<tr>
<td>0015</td>
<td>=1</td>
<td>54</td>
<td>IOC0     EQU 15H:BYTE ; W</td>
</tr>
<tr>
<td>0015</td>
<td>=1</td>
<td>55</td>
<td>IOS0     EQU 15H:BYTE ; R</td>
</tr>
<tr>
<td>0016</td>
<td>=1</td>
<td>56</td>
<td>IOC1     EQU 16H:BYTE ; W</td>
</tr>
<tr>
<td>0016</td>
<td>=1</td>
<td>57</td>
<td>IOS1     EQU 16H:BYTE ; R</td>
</tr>
<tr>
<td>0017</td>
<td>=1</td>
<td>58</td>
<td>PWM_CONTROL EQU 17H:BYTE ; W</td>
</tr>
<tr>
<td>0018</td>
<td>=1</td>
<td>59</td>
<td>SP       EQU 18H:WORD ; R/W</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>60</td>
<td>; /*</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>61</td>
<td>; */ 80C196 SFR's</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>62</td>
<td>; *</td>
</tr>
<tr>
<td>0008</td>
<td>=1</td>
<td>64</td>
<td>IOC2     EQU 0BH:BYTE ; W</td>
</tr>
<tr>
<td>0012</td>
<td>=1</td>
<td>65</td>
<td>;TIMER2   EQU 0CH:WORD ; R/W</td>
</tr>
<tr>
<td>0013</td>
<td>=1</td>
<td>66</td>
<td>IPEND1   EQU 12H:BYTE ; R/W</td>
</tr>
<tr>
<td>0014</td>
<td>=1</td>
<td>67</td>
<td>IMASK1   EQU 13H:BYTE ; R/W</td>
</tr>
<tr>
<td>0017</td>
<td>=1</td>
<td>68</td>
<td>WSR      EQU 14H:BYTE ; R/W</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>69</td>
<td>IOS2     EQU 17H:BYTE ; R</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>70</td>
<td>; ===================SEGMENTS; CODE SEGMENT</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>71</td>
<td>;</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>73</td>
<td>CODE SEGMENT</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>75</td>
<td>CSEG</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>77</td>
<td>ENAB_INT;</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>79</td>
<td>EI ;ENABLE</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>80</td>
<td>RET</td>
</tr>
<tr>
<td></td>
<td>=1</td>
<td>81</td>
<td>END</td>
</tr>
</tbody>
</table>
## Symbol Table Listing

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD_COMMAND</td>
<td>0002H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>AD_RESULT_HI</td>
<td>0003H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>AD_RESULT_LO</td>
<td>0002H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>BAUD_RATE</td>
<td>000CH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>E_INT</td>
<td></td>
<td>MODULE STACKSIZE(12)</td>
</tr>
<tr>
<td>ENABLE_INT</td>
<td>0000H</td>
<td>CODE REL PUBLIC ENTRY</td>
</tr>
<tr>
<td>HSI_MODE</td>
<td>0003H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>HSI_STATUS</td>
<td>0006H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>HSI_TIME</td>
<td>0004H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>HSO_COMMAND</td>
<td>0006H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>HSO_TIME</td>
<td>0004H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>IMASK1</td>
<td>0013H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>INT_MASK</td>
<td>0008H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>INT_PENDING</td>
<td>0009H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC0</td>
<td>0015H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC1</td>
<td>0016H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC2</td>
<td>0008H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOPORT0</td>
<td>000EH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOPORT1</td>
<td>000FH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOPORT2</td>
<td>0010H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS0</td>
<td>0015H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS1</td>
<td>0016H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS2</td>
<td>0017H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IPEND1</td>
<td>0012H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>PWM_CONTROL</td>
<td>0017H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>RO</td>
<td>0000H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>SBUF</td>
<td>0007H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>SP</td>
<td>0018H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>SP_COM</td>
<td>0011H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>SP_STAT</td>
<td>0011H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>TIMER1</td>
<td>000AH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>TIMER2</td>
<td>000CH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>WATCHDOG</td>
<td>000AH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>WSR</td>
<td>0014H</td>
<td>NULL ABS BYTE</td>
</tr>
</tbody>
</table>

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.
MCS-96 MACRO ASSEMBLER STATUS

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
0000 public statustemp 1
0000 status_temp rseg 2
0000 status_temp: DSB 1 ;Global status register
0001 end

SYMBOL TABLE LISTING

NAME VALUE ATTRIBUTES

STATUS_TEMP 0000H REG REL PUBLIC BYTE

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.
### MCS-96 MACRO ASSEMBLER

**GETCHAR**

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>

<table>
<thead>
<tr>
<th>ERR LOC</th>
<th>OBJECT</th>
<th>LINE</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>$debug</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>$nolist include (8096.inc)</td>
</tr>
<tr>
<td>001C</td>
<td></td>
<td>52</td>
<td>tmpO equ 1CH:word</td>
</tr>
<tr>
<td>0006</td>
<td></td>
<td>53</td>
<td>RI_pos equ 06H:byte</td>
</tr>
<tr>
<td>00BF</td>
<td></td>
<td>54</td>
<td>RI_mask equ 0BFH:byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>56</td>
<td>extrn status_temp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>57</td>
<td>public getchar</td>
</tr>
<tr>
<td>0000</td>
<td></td>
<td>59</td>
<td>CSEG</td>
</tr>
<tr>
<td>0000 901100</td>
<td>E 61</td>
<td>getchar: orb status_temp, SP_STAT</td>
<td></td>
</tr>
<tr>
<td>0003 3600FA</td>
<td>E 62</td>
<td>jbc status_temp, RI_pos, getchar</td>
<td></td>
</tr>
<tr>
<td>0006 B0071C</td>
<td>63</td>
<td>ldb tmp0, sbuf</td>
<td></td>
</tr>
<tr>
<td>0009 71BF00</td>
<td>E 64</td>
<td>andb status_temp, #RI_mask</td>
<td></td>
</tr>
<tr>
<td>000C F0</td>
<td>65</td>
<td>ret</td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td>66</td>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>
## Symbol Table Listing

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD_COMMAND</td>
<td>0002H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>AD_RESULT_HI</td>
<td>0003H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>AD_RESULT_LO</td>
<td>0002H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>BAUD_RATE</td>
<td>000EH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>GETCHAR</td>
<td>0000H</td>
<td>CODE REL PUBLIC ENTRY</td>
</tr>
<tr>
<td>HSI_MODE</td>
<td>0003H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>HSI_STATUS</td>
<td>0006H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>HSI_TIME</td>
<td>0004H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>HSIO_COMMAND</td>
<td>0006H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>HSITIME</td>
<td>0004H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>IMASK1</td>
<td>0013H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>INT_MASK</td>
<td>0008H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>INT_PENDING</td>
<td>0009H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC0</td>
<td>0015H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC1</td>
<td>0016H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC2</td>
<td>000BH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IPORTO</td>
<td>000EH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IPORT1</td>
<td>000FH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IPORT2</td>
<td>0010H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS0</td>
<td>0015H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS1</td>
<td>0016H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS2</td>
<td>0017H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IPEND1</td>
<td>0012H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>PWM_CONTROL</td>
<td>0017H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>RO</td>
<td>0000H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>RI_MASK</td>
<td>0008H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>RI_POS</td>
<td>0006H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>SBIF</td>
<td>0007H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>SF</td>
<td>0011H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>STATU</td>
<td></td>
<td>NULL EXTERNAL</td>
</tr>
<tr>
<td>TIMER1</td>
<td>000AH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>TIMER2</td>
<td>000CH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>TMP0</td>
<td>001CH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>WATCHDOG</td>
<td>000AH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>WSR</td>
<td>0014H</td>
<td>NULL ABS BYTE</td>
</tr>
</tbody>
</table>

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 Tl_pos equ 05H:byte
  00DF 53 Tl_mask equ 0DFH:byte
  54
  55 extrn status_temp
  56 public putchar
  57
  0000 58 CSEG
  0000 901100 E 60 putchar: orb status_temp, SP STAT
  0003 3500FA E 61 jbc status_temp, Tl_pos, putchar
  0006 B3180207 62 lab sbuf, 2[sp]
  000A 71DF00 E 63 andb status_temp, #TImask
  000D F0 64 ret
  000E 65 end
<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD_COMMAND.</td>
<td>0002H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>AD_RESULT_HI</td>
<td>0003H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>AD_RESULT_LO</td>
<td>0002H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>BAUD_RATE</td>
<td>000EH</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>HSI_MODE</td>
<td>0003H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>HSI_STATUS</td>
<td>0006H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>HSI_TIME</td>
<td>0004H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>HSO_COMMAND</td>
<td>0006H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>HSO_TIME</td>
<td>0004H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>IMASK1</td>
<td>0013H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>INT_MASK</td>
<td>0008H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>INT_PENDING</td>
<td>0009H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC0</td>
<td>0015H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC1</td>
<td>0016H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOC2</td>
<td>0008H</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>IOPORT0</td>
<td>000EH</td>
<td>NULL ABS Byte</td>
</tr>
<tr>
<td>IOPORT1</td>
<td>000FH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOPORT2</td>
<td>0010H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS0</td>
<td>0015H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS1</td>
<td>0016H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IOS2</td>
<td>0017H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>IPEND1</td>
<td>0012H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>PUTCHAR</td>
<td>0000H</td>
<td>CODE REL PUBLIC ENTRY</td>
</tr>
<tr>
<td>PCON</td>
<td>0017H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>RO</td>
<td>0000H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>SBUF</td>
<td>0007H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>SP</td>
<td>0018H</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>SP Tháng</td>
<td>0011H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>SP_STAT</td>
<td>0011H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>STATUS_TEMP</td>
<td>CODFH</td>
<td>NULL EXTERNAL</td>
</tr>
<tr>
<td>TI_MASK</td>
<td>0005H</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>TI_POS</td>
<td>000AH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>TIMER1</td>
<td>000AH</td>
<td>NULL ABS BYTE</td>
</tr>
<tr>
<td>TIMER2</td>
<td>000CH</td>
<td>NULL ABS WORD</td>
</tr>
<tr>
<td>WSR</td>
<td>0014H</td>
<td>NULL ABS BYTE</td>
</tr>
</tbody>
</table>

ASSEMBLY COMPLETED, NO ERROR(S) FOUND.
APPENDIX B: PC SOFTWARE LISTING
$declare$

```
program shell

  character*6 word
  character*1 chr(6), cts(4), capps, cappi, cr, cappr, cappz
  character*1 bell, resp
  integer i, j
  integer*4 tmpvb, tmpvbb, tmpva
  integer step, iresp, ichanl, ichann
  real*4 dlv, d2v, dhl, d2h
  real*4 d3v, d4v, d3h, d4h
  real*4 vmag, hmag, gain, ccmax
  real*4 tmpvbt(11)

  integer ichar
  integer*4 int4

  equivalence (word,chr)
```

```
c Functions

c Data Relations

equivalence (word,chr)
```

```
c Data Initialization

data word /*6H */
data capps, cappi, cappr, cappz /*S', 'I', 'R', 'A'*/
data ichanl, ichann /*, 0/ */
```

```
cls(1) = $#33
cls(2) = $#133
cls(3) = $#62
cls(4) = $#112
bell = $#7
```

```
step = 0
dl = 0.
d2v = 0.
dhl = 0.
```
Line# Source Line
57 d2h = 0.
58 d3v = 0.
59 d4v = 0.
60 d3h = 0.
61 d4h = 0.
62
63 c Clear the screen
64 2 write(*,3)(cls(i),i=1,4)
65 3 format( ' ',4a1)
66 66 open(11,file='drive',status='old')
67 66 read(11,67)resp
68 67 format(a1)
69 if ( resp .ne. cappz ) then
70 73 goto 66
71 endif
72
73 call dcalc(cls,d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,step,bell,ccmax)
74 call sgm(vmag,0)
75 call sgm(hmag,1)
76 call sgm(gain,2)
77 call ftune(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,ccmax)
78 close(11)
79 if ( gain .eq. -9999. ) then
80 1357 goto
81 endif
82 open(12,file='correct',status='new')
83 tmpvbl(1) = vmag
84 tmpvbl(2) = hmag
85 tmpvbl(3) = gain
86 tmpvbl(4) = d1v
87 tmpvbl(5) = d2v
88 tmpvbl(6) = d3v
89 tmpvbl(7) = d4v
90 tmpvbl(8) = d1h
91 tmpvbl(9) = d2h
92 tmpvbl(10) = d3h
93 tmpvbl(11) = d4h
94 do 152 i=1,11
95 tmpv = int4(tmpvbl(i))
96 tmpvb = tmpv * 10
97 if ( tmpvb .lt. 0 ) then
98 tmpvb = -tmpvb
99 tmpvb = tmpvb + 1
100 endif
101 write(12,154)tmpvb
102 154 format(i6)
103 152 continue
close(12)
write(*,1356)
format(' Correction Matrix now computed for the selected channel.'
        + '/' EARS Correction option can now be run (main menu option C).'
continue
stop
end

main Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4H</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0002</td>
</tr>
<tr>
<td>IRESP</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0006</td>
</tr>
<tr>
<td>TMPVA</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>000a</td>
</tr>
<tr>
<td>WORD</td>
<td>local</td>
<td>CHAR*6</td>
<td>6</td>
<td>000c</td>
</tr>
<tr>
<td>TMPVB</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>000e</td>
</tr>
<tr>
<td>CAPPS</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0012</td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0012</td>
</tr>
<tr>
<td>CAPP1</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0013</td>
</tr>
<tr>
<td>CAPP2</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0014</td>
</tr>
<tr>
<td>J</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0015</td>
</tr>
<tr>
<td>ICHANL</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0016</td>
</tr>
<tr>
<td>ICHANN</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0016</td>
</tr>
<tr>
<td>D1V</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>001a</td>
</tr>
<tr>
<td>TMPVBB</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>001e</td>
</tr>
<tr>
<td>D2V</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0022</td>
</tr>
<tr>
<td>D3V</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0026</td>
</tr>
<tr>
<td>D4V</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>002a</td>
</tr>
<tr>
<td>TMPVBL</td>
<td>local</td>
<td>REAL*4</td>
<td>44</td>
<td>002e</td>
</tr>
<tr>
<td>CR</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>005a</td>
</tr>
<tr>
<td>HMAG</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>005c</td>
</tr>
<tr>
<td>GAIN</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0060</td>
</tr>
<tr>
<td>BELL</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0064</td>
</tr>
<tr>
<td>CLS</td>
<td>local</td>
<td>CHAR*1</td>
<td>4</td>
<td>0066</td>
</tr>
<tr>
<td>ICHAR</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>006a</td>
</tr>
<tr>
<td>VMAG</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>006e</td>
</tr>
<tr>
<td>CCMAX</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0072</td>
</tr>
<tr>
<td>RESP</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0076</td>
</tr>
<tr>
<td>STEP</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0076</td>
</tr>
<tr>
<td>D1H</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>007c</td>
</tr>
<tr>
<td>D2H</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0080</td>
</tr>
<tr>
<td>D3H</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0084</td>
</tr>
<tr>
<td>CHR</td>
<td>equiv</td>
<td>CHAR*1</td>
<td>6</td>
<td>000c</td>
</tr>
</tbody>
</table>
Microsoft FORTRAN Optimizing Compiler Version 4.00

Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCALC</td>
<td>extern</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>FTUNE</td>
<td>extern</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>SGM</td>
<td>extern</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>main</td>
<td>FSUBRT</td>
<td>***</td>
<td>***</td>
<td>0000</td>
</tr>
</tbody>
</table>

Code size = 03a8 (936)
Data size = 006b (107)
Bss size = 0088 (136)

No errors detected
subroutine dcalc(cls, d1v, d2v, d1h, d2h, d3v, d4v, d3h, d4h, step, bell, + ccmax)

data structures

character*1 cts(4)
character*1 bell, resp
integer i, j, n, zline, ezline, step, iresp
integer stepl
real*4 dlv, d2v, dlh, d2h
real*4 d3v, d4v, d3h, d4h, ccmax
double precision z, h, phi, psi, pi2, capt, raddeg
double precision betel, betar, gamat, gamar
double precision theta, r, sigma, eta

double precision betatv, betarv, gamav, gamarv
double precision thetav, rv, sigmav, etav
double precision betath, betarh, gamath, gamarh
double precision thetah, rh, sigmah, etah
double precision denomv, denomh, temp, temp2, term1, term2
double precision delta, jeff, negftag, max, scale

functions

integer ichar
double precision dacos, dsin, dcos, dsqrt, dabs

data initialization

pi2 = 1.5707963268000
raddeg = 57.2957795131D0
n = 1
z = 111.
h = 70.
capt = 8.
betalv = 33.
betarv = 94.
betalh = 49.
betarh = 144.
zline = 0
bell = 208.
betar = 211.
ccmax = 0.05
ezline = 0
57   c Clear the screen
58   write(*,3)(cls(i),i=1,4)
59   format(' ',4a1)
60   c Write the Main Menu
61   write(*,50)z,h,capl,betalv,betalv,betarv,betarv,zline,betal, +betal,ccmax
62   format(/20x,' GMS Engineering EEG-EOG Artifact Removal'// +/20x,' Parameter Menu'////,
63   +/ ' 1',5x, 'Distance between the eyes (mm).................',' +/ ' 2',5x, 'Distance from the origin to the stim electrode (mm),'
64   +/ ' 3',5x, 'Corneal-retinal distance (mm)....................',
65   +/ ' 4',5x, 'Distance from Left Eye to VJ-EOG electrode (mm)....',
66   +/ ' 5',5x, 'Distance from Right Eye to VJ-EOG electrode (mm)....',
67   +/ ' 6',5x, 'Distance from Left Eye to H-EOG electrode (mm)....',
68   +/ ' 7',5x, 'Distance from Right Eye to H-EOG electrode (mm)....',
69   +/ ' 8',5x, 'H-EOG electrode above(O)/below(1) the "Z" line?.... ',
70   +/ ' 9',5x, 'Distance from Left Eye to EEG electrode (mm)....',
71   +/ '10',5x, 'Distance from Right Eye to EEG electrode (mm)....',
72   +/ '11',5x, 'Maximum cross-correlation function for correction..',
73   +/ '12',5x, 'Physical data entered; compute Correction Matrix'/,
74   +/ Enter Response >
75   read(*,15)iresp
76   if (iresp .eq. 1) then
77       write(*,110)
78       read(*,'(f12.5)')z
79       if ( z .lt. 0. ) then
80           z = 0.
81           call error(bell)
82       endif
83   elseif (iresp .eq. 2) then
84       write(*,110)
85       read(*,'(f12.5)')h
86       if ( h .lt. 0. ) then
87           h = 0.
88           call error(bell)
89       endif
elseif (iresp .eq. 3) then
  write(*,110)
  format(/13x,'Enter new value >'
  read(*,'(f12.5)')capt
  if ( capt .lt. 0. ) then
    capt = 8.
    call error(bell)
  endif

elseif (iresp .eq. 4) then
  write(*,110)
  read(*,'(f12.5)')betatv
  if ( betatv .lt. 0. ) then
    betatv = 0.
    call error(bell)
  endif

elseif (iresp .eq. 5) then
  write(*,110)
  read(*,'(f12.5)')betarv
  if ( betarv .lt. 0. ) then
    betarv = 0.
    call error(bell)
  endif

elseif (iresp .eq. 6) then
  write(*,110)
  read(*,'(f12.5)')betalh
  if ( betalh .lt. 0. ) then
    betalh = 0.
    call error(bell)
  endif

elseif (iresp .eq. 7) then
  write(*,110)
  read(*,'(f12.5)')betarh
  if ( betarh .lt. 0. ) then
    betarh = 0.
    call error(bell)
  endif

elseif (iresp .eq. 8) then
  write(*,110)
  read(*,'(f15)')zline
  if ( (zline .ne. 0) .and. (zline .ne. 1)) then
    zline = 0
    call error(bell)
  endif

elseif (iresp .eq. 9) then
  write(*,110)
  read(*,'(f12.5)')betaln
  if ( betaln .lt. 0. ) then
    betaln = 0.
    call error(bell)
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 ((betat.eq.0.).or.(betar.eq.0.)) then
189 write(*,155)bell
190 155 format(//,'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 endif
201
202 goto 2
203
204
205 c Check if the VEOG electrode is below the stim electrode;
206 c if so, negate the Dv's
207 c
208 98 delta = dacos ( (x**2 - betalv**2 + betarv**2)
209 # / (2.*x*betarv) )
210 if ( (betarv*dsin(delta)) .lt. h ) then
211 negflag = -1.000
212 write(*,170)
213 c170 format('/ The EOGv electrode is below the STIM electrode.')
214 else
215 negflag = 1.000
216 write(*,172)
217 c172 format('/ The EOGv electrode is above the STIM electrode.')
218 endif
219
220 c Compute Vertical EOG parameters
221 c
222 23 write(*,23)
223 c23 format('/ EOGV')
rv = dsqrt( 0.5 * ( betaLV**2 + betarV**2 - ((z**2)/2.) ) )
c write(*,200)rv

c temp = (rv**2 + (z/2)**2 - betarV**2) / (rv * z)
if ( temp .gt. 1. ) then
temp = 1.0
elseif ( temp .lt. -1. ) then
temp = -1.0
endif

gamay = ( rv/betaLV) * dsin( thetav )
c write(*,230)gamay

gamavr = ( rv/betarV) * dsin( thetav )
c write(*,240)gamavr

gammaV = (rv/betavr) * dsin( thetav )
c write(*,250)gammaV

sigmav = dsqrt( rv**2 + h**2 - 2.*rv*h*dsin(thetav) )
c write(*,260)sigmav

etav = (-rv/sigmav) * dcos( thetaV )
c write(*,270)etav

c Compute Horizontal EOG parameters

c rh = dsqrt( 0.5 * ( betah**2 + betarh**2 - ((z**2)/2.) ) )
c write(*,200)rh

c temp = (rh**2 + (z/2)**2 - betarh**2) / (rh * z)
if ( temp .gt. 1. ) then
temp = 1.0
elseif ( temp .lt. -1. ) then
temp = -1.0
endif

thetah = dacos( temp )
if ( zline .eq. 1 ) thetah = -thetah

c gamah = ( rh/betah) * dsin( thetah )
c write(*,230)gamah

gamahr = ( rh/betarh) * dsin( thetah )
c write(*,240)gamahr

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 c Compute EEG electrode parameters
288 289 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
296 297 temp = 1.0
298 299 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 )
305 306 gamar = ( r/betar ) * dsin( theta )
307 c write(*,230) gamal
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 c Compute the denominators (V & H)
316 317 psi = 0.
318 319 temp = (sigma**(-2)) * ( dsqrt( 1.-(eta**2) ) )
320 temp2 = (sigmav**(-2)) * ( dsqrt( 1.-(etav**2) ) )
321 denomv = temp / temp2
322 c write(*,1212) denomv
323 324 325 psi = p12
temp = (sigma**(-2)) * eta
temp2 = (sigmav**(-2)) * etah
denomh = temp / temp2
328 329 c write(*,1213) denomh
331 c1213 format( , denomh = ',f16.8)
332 333 c Divi
334 phi = 0.
term1 = (beta1**(-2)) * gamal
335 term2 = (beta2**(-2)) * gamar
d1v = 0.5 * negflag * (term1 + term2) / denomv

phi = pi2

term1 = (betal**(-2)) * (-dsqrt1.-(gaml**(2))

term2 = (betal**(-2)) * (dsqrt1.-(gaml**(2))

d1h = 0.5 * (term1 + term2) / denomh

phi = pi2

term1 = (betal**(-2)) * (-dsqrt1.-(gaml**(2))

term2 = (betal**(-2)) * (dsqrt1.-(gaml**(2))

d1h = 0.5 * (term1 + term2) / denomh

d2v = (1./betal) + (1./betal) / capl

d2v = (d2v / denomv)

d2h = (1./betal) + (1./betal) / capl

d2h = d2h / denomh

1000 continue

phi = 0.

term1 = (betalv**(-2)) * gamalv

term2 = (betalv**(-2)) * gamalv

d3v = 0.5 * (term1 + term2)

phi = pi2

term1 = (betahl**(-2)) * (-dsqrt1.-(gamlh**(2))

term2 = (betahl**(-2)) * (dsqrt1.-(gamlh**(2))

d3h = 0.5 * (term1 + term2)

d4v = (1./betalv) + (1./betalv) / capl

d4v = (d4v / denomv)

d4h = (1./betahl) + (1./betahl) / capl

d4h = d4h / denomh

Print out the data, as well as writing it to the ASCII file

write(*,2000)

write(*,2000) format(///' The D1's are:,,/

write(*,2000) i, dlv, d1h, d2v, d2h

write(*,2000) format(///' EEG Electrode # i,13,/' D1V = ',f16.5,

c + ' D1H = ',f16.5,' D2V = ',f16.5,' D2H = ',f16.5)

c write(*,2010) d5v, d5h, d4v, d4h

c write(*,2010) format(///' D3V = ',f16.5,' D3H = ',f16.5,

c + ' D4V = ',f16.5,' D4H = ',f16.5)

c Scale D's

if (dabs(dlv) .gt. 10.) dlv = (d1v/dabs(dlv))*10.

if (dabs(d2v) .gt. 10.) d2v = (d2v/dabs(d2v))*10.

if (dabs(d1h) .gt. 10.) d1h = (d1h/dabs(d1h))*10.
if (dabs(d2h) .gt. 10.)
d2h = (d2h/dabs(d2h))*10.
if (dabs(d3h) .gt. 10.)
d3h = (d3h/dabs(d3h))*10.
if (dabs(d4h) .gt. 10.)
d4h = (d4h/dabs(d4h))*10.
max = 0.000
if (dabs(dlv) .gt. max)
max = dabs(dlv)
if (dabs(d2v) .gt. max)
max = dabs(d2v)
if (dabs(d3v) .gt. max)
max = dabs(d3v)
if (dabs(d4v) .gt. max)
max = dabs(d4v)
if (dabs(d3h) .gt. max)
max = dabs(d3h)
if (dabs(d4h) .gt. max)
max = dabs(d4h)
scale = 8192.000 / max
dlv = dlv * (-scale)
d2v = d2v * (-scale)
d3v = d3v * scale
d4v = d4v * scale
dlh = dlh * scale
d2h = d2h * scale * 0.01
d3h = d3h * scale
d4h = d4h * scale
c write(*,2100)
c2100 format(///'The D's for EEG.C96 are (in order):',/)
c write(*,2070) scale*dlv, scale*d2v, scale*d3v, scale*d4v, scale*dlh, scale*d2h, scale*d3h, scale*d4h
c2070 format(/8C',f9.0))
c write(*,3333)
c3333 format(///'/Please wait.../)
c3333 + '/'/
step = 3
return
dend

DCALC Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCMAX</td>
<td></td>
<td>param</td>
<td>0006</td>
<td></td>
</tr>
<tr>
<td>BELL</td>
<td></td>
<td>param</td>
<td>000a</td>
<td></td>
</tr>
<tr>
<td>STEP</td>
<td></td>
<td>param</td>
<td>000e</td>
<td></td>
</tr>
<tr>
<td>D4H</td>
<td></td>
<td>param</td>
<td>0012</td>
<td></td>
</tr>
<tr>
<td>D3H</td>
<td></td>
<td>param</td>
<td>0016</td>
<td></td>
</tr>
<tr>
<td>D4V</td>
<td></td>
<td>param</td>
<td>001a</td>
<td></td>
</tr>
<tr>
<td>D3V</td>
<td></td>
<td>param</td>
<td>001e</td>
<td></td>
</tr>
<tr>
<td>D2H</td>
<td></td>
<td>param</td>
<td>0022</td>
<td></td>
</tr>
</tbody>
</table>
DCALC Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1H</td>
<td>param</td>
<td></td>
<td>0026</td>
<td></td>
</tr>
<tr>
<td>D2V</td>
<td>param</td>
<td></td>
<td>002a</td>
<td></td>
</tr>
<tr>
<td>DIV</td>
<td>param</td>
<td></td>
<td>002e</td>
<td></td>
</tr>
<tr>
<td>CLS</td>
<td>param</td>
<td></td>
<td>0032</td>
<td></td>
</tr>
<tr>
<td>DLINE</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0002</td>
</tr>
<tr>
<td>IRESP</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0004</td>
</tr>
<tr>
<td>BETARV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0006</td>
</tr>
<tr>
<td>SIGMAV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>000e</td>
</tr>
<tr>
<td>EZLINE</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0016</td>
</tr>
<tr>
<td>H</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0018</td>
</tr>
<tr>
<td>DENOMV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0020</td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0028</td>
</tr>
<tr>
<td>J</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>002a</td>
</tr>
<tr>
<td>P12</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>002c</td>
</tr>
<tr>
<td>THETV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0034</td>
</tr>
<tr>
<td>W</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>003c</td>
</tr>
<tr>
<td>R</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>003e</td>
</tr>
<tr>
<td>RH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0046</td>
</tr>
<tr>
<td>ETA</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>004e</td>
</tr>
<tr>
<td>Z</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0056</td>
</tr>
<tr>
<td>JEFF</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>005e</td>
</tr>
<tr>
<td>CAPL</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0066</td>
</tr>
<tr>
<td>PHI</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>006e</td>
</tr>
<tr>
<td>ETAH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0076</td>
</tr>
<tr>
<td>GAMAL</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>007e</td>
</tr>
<tr>
<td>MAX</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0086</td>
</tr>
<tr>
<td>ICHAR</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>008e</td>
</tr>
<tr>
<td>RADDEG</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0090</td>
</tr>
<tr>
<td>SCALE</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0098</td>
</tr>
<tr>
<td>TEMP2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00a0</td>
</tr>
<tr>
<td>RV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00a8</td>
</tr>
<tr>
<td>GAMAR</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00b0</td>
</tr>
<tr>
<td>BETAL</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00b8</td>
</tr>
<tr>
<td>TERM1</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00c0</td>
</tr>
<tr>
<td>DELTA</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00c8</td>
</tr>
<tr>
<td>TERM2</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00d0</td>
</tr>
<tr>
<td>GAMALH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00d8</td>
</tr>
<tr>
<td>PSI</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00e0</td>
</tr>
<tr>
<td>WAV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00e8</td>
</tr>
<tr>
<td>STEP1</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>00f0</td>
</tr>
<tr>
<td>BETAR</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00f2</td>
</tr>
<tr>
<td>GAMARH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>00fa</td>
</tr>
<tr>
<td>BETALH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0102</td>
</tr>
<tr>
<td>ETAH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>010a</td>
</tr>
<tr>
<td>SIGMA</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0112</td>
</tr>
<tr>
<td>TEMP</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>011a</td>
</tr>
<tr>
<td>BETARH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0122</td>
</tr>
<tr>
<td>THETA</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>012a</td>
</tr>
<tr>
<td>GAMALV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0132</td>
</tr>
<tr>
<td>SIGMAH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>013a</td>
</tr>
<tr>
<td>RESP</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0142</td>
</tr>
</tbody>
</table>
DCALC Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DENOMH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0144</td>
</tr>
<tr>
<td>THETAH</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>014c</td>
</tr>
<tr>
<td>GAMARV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>0154</td>
</tr>
<tr>
<td>BETALV</td>
<td>local</td>
<td>REAL*8</td>
<td>8</td>
<td>015c</td>
</tr>
</tbody>
</table>

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

ERROR Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>BELL</td>
<td>param</td>
<td></td>
<td></td>
<td>0006</td>
</tr>
<tr>
<td>RESP</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0164</td>
</tr>
</tbody>
</table>

Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCALC</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
<td>0000</td>
</tr>
<tr>
<td>ERROR</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
<td>1359</td>
</tr>
</tbody>
</table>

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

No errors detected
subroutine sgm(magsun, orient)

character*6 resp
character*1 bite(2)
character*1 cappa, cappb, cappc, cappd, cappe, cappf
character*1 chr(6), one, sp, cr, retran, oktran
integer*2 in, istart, i, j, k, irec, orient
real*4 eeg(4096), magsum
complex*8 sg(4096), sgtemp
integer tmprsp

integer ichar
real*4 float, cabs
complex*8 cmplx

equivalence (resp, chr)
data cappa, cappb, cappc, cappd, cappe, cappf /'A', 'B', 'C', 'D', 'E', 'F'/
data one /49/
data sp, retran, oktran /8#140, 0, 1/
open (9, file='scrtch.dat', status='new', access='sequential',
+ form='formatted')

c Load EEG data - Read 120 words
j = 2018
do 33 i = 1, 60
read(11,105)in
format(i6)
33 17 read(11,105)in
105 format(i6)

write(*,105)in
c orient = in - ((in/10) * 10)
in = in/10
if ( orient .eq. 1 ) then
   in = -in
endif

eeg(j) = float(in)
Program:

```fortran
57  j = j + 1
58
59  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  c Load EOG data - Read 120 words
87
88  j = 2018
89  do 133 i = 1, 60
90    133 continue
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)
113  130 continue
```
continue

c Window & FFT eog

call sgwin (eeg, eeg, 9, 0.001, 12)
do 170 i = 1, 4096
   sg(i) = cfrptx ( eeg(i), 0.)
continue
call cfft ( sg, 12, 0, 1.0 )
c Divide & compute the average
magnitude
magsum = 0.
rewind 9
do 1013 i = 6, 45
   read(9,*) sgtemp
   eeg(i) = 100. * cabs ( sgtemp/sg(i) )
magsum = magsum + eeg(i)
c
write (*,2345)i, eeg(i), magsum
2345 format (' ',i6,2(' ',f15.3))
continue
magsum = magsum / 40.
clos (9, status='delete')
return
end

SGM Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORIENT</td>
<td>param</td>
<td></td>
<td>0006</td>
<td></td>
</tr>
<tr>
<td>MAGSUM</td>
<td>param</td>
<td></td>
<td>000a</td>
<td></td>
</tr>
<tr>
<td>EEG</td>
<td>local</td>
<td>REAL*4</td>
<td>16384</td>
<td>0000</td>
</tr>
<tr>
<td>I</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0002</td>
</tr>
<tr>
<td>J</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0004</td>
</tr>
<tr>
<td>K</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0006</td>
</tr>
<tr>
<td>SGTEMP</td>
<td>local</td>
<td>COMPLEX*8</td>
<td>8</td>
<td>0008</td>
</tr>
<tr>
<td>CAPP</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>000a</td>
</tr>
<tr>
<td>CAPPA</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>000b</td>
</tr>
<tr>
<td>CAPPC</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>000c</td>
</tr>
<tr>
<td>CAPPD</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>000d</td>
</tr>
<tr>
<td>CAPPF</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>000e</td>
</tr>
<tr>
<td>ONE</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>000f</td>
</tr>
<tr>
<td>CR</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0010</td>
</tr>
<tr>
<td>SP</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0011</td>
</tr>
<tr>
<td>RETRAN</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0012</td>
</tr>
<tr>
<td>ISTART</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0012</td>
</tr>
</tbody>
</table>
Microsoft FORTRAN Optimizing Compiler Version 4.00

SGM Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>OKTRAN</td>
<td>local</td>
<td>CHAR*1</td>
<td>1</td>
<td>0013</td>
</tr>
<tr>
<td>IN</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0014</td>
</tr>
<tr>
<td>IREC</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0016</td>
</tr>
<tr>
<td>TMPRSP</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>0018</td>
</tr>
<tr>
<td>ICHAR</td>
<td>local</td>
<td>INTEGER*4</td>
<td>4</td>
<td>001c</td>
</tr>
<tr>
<td>RESP</td>
<td>local</td>
<td>CHAR*6</td>
<td>6</td>
<td>0020</td>
</tr>
<tr>
<td>SG</td>
<td>local</td>
<td>COMPLEX*8</td>
<td>32768</td>
<td>4000</td>
</tr>
<tr>
<td>BITE</td>
<td>local</td>
<td>CHAR*1</td>
<td>2</td>
<td>0000</td>
</tr>
<tr>
<td>CHR</td>
<td>local</td>
<td>CHAR*1</td>
<td>6</td>
<td>0020</td>
</tr>
</tbody>
</table>

Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFFT</td>
<td>extern</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>SGM</td>
<td>extern</td>
<td>FSUBRT</td>
<td>***</td>
<td>0000</td>
</tr>
<tr>
<td>SGWIN</td>
<td>extern</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>

Code size = 04b0 (1200)
Data size = 0060 (96)
Bss size  = 0026 (38)

No errors detected
subroutine ftune(dlv, d2v, d1h, d2h, d3v, d4v, d3h, d4h, vmag, hmag, gain, + cmax)

data structures
character*1 cls(4), chr(6), cbell
character*1 resp, sp, retran, oktran, cr
integer i, j, n, k, l, tmprsp, in, mod
real*4 dlv, d2v, d1h, d2h
real*4 d3v, d4v, d3h, d4h, cmax
real*4 cc, ccrawv, ccrawh, ccold, delta1, delta2, ccertio
integer  ichar
real*4 float, cabs, real, abs;
complex*8 cmplx
character yes, yess
integer*2 nun, iorder, icnst2, ilim, orent, orient
real*4 eogvu(512), eogvl(512), eogh(512), raweeg(512)
real*4 eohvu(512), eohvl(512), eohh(512), raweeh(512)
real*4 alpha(512), teinp(512)
real*4 time, alphal, alpha2, const, intrvl, ceeg
real*4 vmag, hmag, gain, raddeg, order, const1, pi2
complex*8 compl(512), coamp2(512)
double precision scale, max, dabs

equivalence (resp,chr)

data initialization
pi2 = 1.5707963268000
raddeg = 57.295779513100
data yes, yess /'y','Y'/
data sp, retran, oktran /&M140, 0, 1/
data cbell /&M7/
delta1 = 0.5 * dlv
delta1 = 100.
delta2 = 0.5 * d2v
do 401 i = 1, 512
read(11,402)in
format(i6)
orent = in - ((in/10) * 10)
in = in/10
if ( orent .eq. 1 ) then
   in = -in
endif

if (abs(float(in)) .gt. 470.) then
Line# | Source Line
--- | ---
57 | c gain = -9999.
58 | c write (*,4433)
59 | c4433 format(' Data is saturated. Interrogate again.')
60 | c goto 2222
61 | c endif
62 | raweeg(i) = float(in)
63 | 
64 | 401 continue
65 | 
66 | do 1401 i = 1, 512
67 | 
68 | 1407 read(11,1402)in
69 | 1402 format(i6)
70 | 
71 | orent = in - ((in/10) * 10)
72 | in = in/10
73 | if ( orent .eq. 1 ) then
74 | in = -in
75 | endif
76 | 
77 | if (abs(float(in)) .gt. 470.) then
78 | gain = -9999.
79 | write (*,4433)
80 | goto 2222
81 | endif
82 | 
83 | raweeg(i) = float(in)
84 | 
85 | 1401 continue
86 | 
87 | do 411 i = 1, 512
88 | 
89 | 417 read(11,412)in
90 | 412 format(i6)
91 | 
92 | orent = in - ((in/10) * 10)
93 | in = in/10
94 | if ( orent .eq. 1 ) then
95 | in = -in
96 | endif
97 | 
98 | if (abs(float(in)) .gt. 470.) then
99 | gain = -9999.
100 | write (*,4433)
101 | goto 2222
102 | endif
103 | 
104 | eagh(i) = float(in)
105 | 
106 | 411 continue
107 | 
108 | do 1411 i = 1, 512
109 | 
110 | 1417 read(11,1412)in
111 | 1412 format(i6)
113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 1411 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 113 orent = in · ((in/10) * 10) 114 in = in/10 115 if ( orent .eq. 1 ) then 116 in = -in 117 endif 118 120 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)
169 170 continue
171 do 431 i = 1, 512
172
174 437 read(11,432)in
175 432 format(i6)
176
177 orent = in - ((in/10) * 10) in = in/10
178 if ( orent .eq. 1 ) then
179 in = -in
180 endif
181
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
eogvl(i) = float(in)
189
191 431 continue
192 do 1431 i = 1, 512
193
195 1437 read(11,1432)in
196 1432 format(i6)
197
198 orent = in - ((in/10) * 10) in = in/10
199 if ( orent .eq. 1 ) then
200 in = -in
201 endif
202
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
eogvl(i) = float(in)
210
212 1431 continue
213
215 c fine tuning algorithm
216 11 continue
217
219 call ccf(eogvu,raweeg,ccrawv)
220 call ccf(eogh,raweeg,ccrawh)
222
call crct(dlv,d2v,d1h,d2h,d3v,d4v,d5h,d4h,vmag,hmag,gain,
223 +eogvu,eogvl,eogh,raweeg,temp)
Line# Source Line
225  call ccf(eogvu,temp,ccold)
226  
227  if (abs(ccold) .gt. abs(ccrawv)) then
228   write(*,1109)cbett
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)cbett
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(cccold) .gt. abs(ccrawh)) then
244   write(*,1109)cbett
245   1109 format(' Data Entry Error!','a1')
246   gain = -9999.
247   goto 2222
248  endif
249  
250  if (abs(ccold) .gt. 0.8) then
251   write(*,1108)cbett
252   1108 format(' Measurement Error!','a1')
253   gain = -9999.
254   goto 2222
255  endif
256  
257  write(*,2460)
258  2460 format(' Fine tuning geometric VERTICAL parameters of model...')
259  write(*,2462)
260  2462 format(' Iteration  D1  D2  CCF')
261  
262  call ccf(eogvu,temp,ccold)
263  
264  l = 0
265  write(*,*)l,d1v,d2v,ccold
266  
267  l = l + 1
268  
269  d1v = d1v + delt1
270  call ccrct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
271  +eogvu,eogvl,eogh,raweeg,temp)
272  
273  call ccf(eogvu,temp,cc)
274  
275  ccratio = cc / ccold
276  if (ccratio .lt. 0.0) then
277     delt1 = delt1 * (-.5)
278     goto 2501
279  endif
280  
281  if (abs(cc) .gt. abs(ccold)) then
282     delt1 = delt1 * (-1.)
283     goto 2501
284  endif
ccold = cc
d2v = d2v + delta2
call crct(d1v,d2v,d1h,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
+eogvu,eogvl,eogh,raweeg,temp)
call ccf(eogvu,temp,cc)
crtio = cc / ccold
if (crtio .lt. 0) then
delta2 = delta2 * (-.5)
goto 2511
endif
if (abs(cc) .gt. abs(ccold)) then
delta2 = delta2 * (-1.)
goto 2511
endif
goto 2500
2511 write(*,*)l,d1v,d2v,cc
j = mod(l,10)
if (j .eq. 0) then
    write(*,2512)
format(' Continue iterations ? (Y/N) ',
read(*,2513)resp
format(a1)
if (resp .ne. 'Y') then
    write(*,2518)
format(' Fine tuning geometric HORIZONTAL parameters',
+ ' of model...')
call ccf(eogh,temp,ccold)
l=0
write(*,2518)
write(*,2462)
call ccf(eogh,temp,ccold)
l=0
write(*,2462)
call ccf(eogh,temp,ccold)
l=0
write(*,2518)
write(*,2518)
call ccf(eogh,temp,ccold)
l=0
write(*,2518)
call ccf(eogh,temp,ccold)
l=0
defs1 = 0.5 * d1h
defs2 = 0.5 * d2h
goto 2600
defsh = d1h + delta1
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  ccrtio = cc / ccold
340  if (ccrtio .lt. 0) then
341    delta1 = delta1 * (-.5)
342  endif
343  goto 2601
344
345  if (abs(cc) .gt. abs(ccold)) then
346    delta1 = delta1 * (-1.)
347  goto 2601
348  endif
349
350  ccotd = cc
351  d2h = d2h + delta2
352  call crct(dlv,d2v,d1h,d2h,d3v,d4v,d5h,d6h,vmag,hmag,gain,
353   +eogvu,eogvl,eogh,raweeg,temp)
354  call ccf(eogh,temp,cc)
355
356  ccrtio = cc / ccold
357  if (ccrtio .lt. 0) then
358    delta2 = delta2 * (-.5)
359  goto 2611
360  endif
361
362  if (abs(cc) .gt. abs(ccold)) then
363    delta2 = delta2 * (-1.)
364  goto 2611
365  endif
366
367  2611 write(*,*)l,d1h,d2h,cc
368  j = mod(l,10)
369  if (j .eq. 0) then
370    write(*,2512)
371    read(*,2513)resp
372    if (resp .ne. 'Y') then
373      goto 2800
374  endif
375  endif
376  ccold = cc
377  if (abs(cc) .lt. ccmax) then
378    goto 2800
379  endif
380  goto 2600
381
382  2800 continue
383  open(12,file='correeg.dat')
384  do 2803 i=64,384
385    write(12,*)i,temp(i),raweeg(i),eogvu(i),eogh(i)
386  2803 continue
387  close(12)
388  c Scale D's
389
390
Line# Source Line

393 \( \text{max} = 0.000 \)
394 if ( \text{dabs}(dlv) .gt. max ) max = \text{dabs}(dlv)
395 if ( \text{dabs}(d2v) .gt. max ) max = \text{dabs}(d2v)
396 if ( \text{dabs}(d3v) .gt. max ) max = \text{dabs}(d3v)
397 if ( \text{dabs}(d4v) .gt. max ) max = \text{dabs}(d4v)
398 if ( \text{dabs}(d5v) .gt. max ) max = \text{dabs}(d5v)
399 if ( \text{dabs}(divy) .gt. max ) max = \text{dabs}(divy)
400 if ( \text{dabs}(d2h) .gt. max ) max = \text{dabs}(d2h)
401 if ( \text{dabs}(d3h) .gt. max ) max = \text{dabs}(d3h)
402 if ( \text{dabs}(d4h) .gt. max ) max = \text{dabs}(d4h)
403 scale = 8192.000 / max
404 dlv = dlv * 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 2222 return
414 end

**FTUNE Local Symbols**

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCMAX</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>0006</td>
</tr>
<tr>
<td>GAIN.</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>000a</td>
</tr>
<tr>
<td>VMAG.</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>000e</td>
</tr>
<tr>
<td>D4H</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>0012</td>
</tr>
<tr>
<td>D3H</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>0016</td>
</tr>
<tr>
<td>D4V</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>001a</td>
</tr>
<tr>
<td>D3V</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>001e</td>
</tr>
<tr>
<td>D2V</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>0022</td>
</tr>
<tr>
<td>D1V</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>0026</td>
</tr>
<tr>
<td>D1V</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>002a</td>
</tr>
<tr>
<td>D1V</td>
<td>param</td>
<td>REAL</td>
<td>4</td>
<td>002e</td>
</tr>
<tr>
<td>CCRAWV.</td>
<td>local</td>
<td>REAL</td>
<td>4</td>
<td>0032</td>
</tr>
<tr>
<td>EOGVJ</td>
<td>local</td>
<td>REAL</td>
<td>4</td>
<td>0048</td>
</tr>
<tr>
<td>CMPLX</td>
<td>local</td>
<td>COMPLEX*8</td>
<td>8</td>
<td>0002</td>
</tr>
<tr>
<td>CCRTO.</td>
<td>local</td>
<td>REAL</td>
<td>4</td>
<td>000a</td>
</tr>
<tr>
<td>YES</td>
<td>local</td>
<td>CHAR</td>
<td>1</td>
<td>000d</td>
</tr>
<tr>
<td>YES.</td>
<td>local</td>
<td>CHAR</td>
<td>1</td>
<td>000e</td>
</tr>
<tr>
<td>IORDER.</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>000e</td>
</tr>
<tr>
<td>SP.</td>
<td>local</td>
<td>CHAR</td>
<td>1</td>
<td>000f</td>
</tr>
<tr>
<td>RETRAN.</td>
<td>local</td>
<td>CHAR</td>
<td>1</td>
<td>0010</td>
</tr>
<tr>
<td>CCRAWV.</td>
<td>local</td>
<td>REAL</td>
<td>4</td>
<td>0010</td>
</tr>
<tr>
<td>OKTRAN.</td>
<td>local</td>
<td>CHAR</td>
<td>1</td>
<td>0011</td>
</tr>
<tr>
<td>CBELL.</td>
<td>local</td>
<td>CHAR</td>
<td>1</td>
<td>0012</td>
</tr>
<tr>
<td>CC.</td>
<td>local</td>
<td>REAL</td>
<td>4</td>
<td>0014</td>
</tr>
<tr>
<td>CONST.</td>
<td>local</td>
<td>REAL</td>
<td>4</td>
<td>0018</td>
</tr>
<tr>
<td>ORENT.</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>001c</td>
</tr>
<tr>
<td>I.</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>001e</td>
</tr>
<tr>
<td>J.</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0022</td>
</tr>
</tbody>
</table>
subroutine ccf(ueeg,ceeg,cc)
real*4 cc, cov, sho
real*4 ueeg(512), ceeg(512)
real*8 a(512,6), b(6)
423  integer*2     ii, jj
424
425
426  do 10 ii=1,512
427       a(ii,1) = ueeg(ii)
428       a(ii,2) = cceg(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 iii=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
454        if ( sho .eq. 0.0 ) then
455           cc = 1.0
456        goto 75
457        endif
458
459        cc = cov / sho
460
461 75     continue
462
463 return
464 end

CCF Local Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>param</td>
<td>INTEGER*2</td>
<td>0006</td>
<td></td>
</tr>
<tr>
<td>CEEG</td>
<td>param</td>
<td>INTEGER*2</td>
<td>000a</td>
<td></td>
</tr>
<tr>
<td>UEEG</td>
<td>param</td>
<td>INTEGER*2</td>
<td>000e</td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0094</td>
</tr>
<tr>
<td>JJ</td>
<td>local</td>
<td>INTEGER*2</td>
<td>2</td>
<td>0096</td>
</tr>
<tr>
<td>COV</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>0098</td>
</tr>
<tr>
<td>SHO</td>
<td>local</td>
<td>REAL*4</td>
<td>4</td>
<td>009c</td>
</tr>
<tr>
<td>A</td>
<td>local</td>
<td>REAL*8</td>
<td>24576</td>
<td>7142</td>
</tr>
<tr>
<td>B</td>
<td>local</td>
<td>REAL*8</td>
<td>48</td>
<td>d142</td>
</tr>
</tbody>
</table>
subroutine crct(dlv,d2v,dlh,d2h,d3v,d4v,d3h,d4h,vmag,hmag,gain,
    +eogvu,eogvl,eogh,raweeg,temp)
    integer i, j, n, k, tmrsp, in
    real*4 dlv, d2v, d1h, d2h
    real*4 d3v, d4v, d3h, d4h
    real*4 float, cabs, real
    complex*8 cplx
    real*4 eogvu(512), eogvl(512), eogh(512), raweeg(512)
    real*4 teogvu(512), teogvl(512), teogh(512)
    real*4 alpha(512), temp(512)
    real*4 time, alpha1, alpha2, const, intrvl, ceeg
    real*4 vmag, hmag, gain, raddeg, order, const1, pi2
    complex*8 comp1(512), comp2(512)
    c Window & FFT EOG-VU
    call sgwin(eogvu, teogvu, 9, 0.001, 9)
    do 60 i = 1, 512
    comp(i) = cplx( teogvu(i), 0. )
    60 continue
    call cfft( comp1, 9, 0, 1.0 )
    c Window & FFT EOG-VL
    call sgwin(eogvl, teogvl, 9, 0.001, 9)
    do 70 i = 1, 512
    comp2(i) = cplx( teogvl(i), 0. )
    70 continue
    call cfft( comp2, 9, 0, 1.0 )
    c Compute Alpha
    do 80 i = 1, 512
    temp(i) = (gain * cabs( comp1(i) / comp2(i) )) / 100.
    if (temp(i) .lt. 1.) then
    temp(i) = 1.
    elseif (temp(i) .gt. 4.) then
    temp(i) = 4.
    endif
    alpha(i) = temp(i)
    80 continue
    c Moving Average Filter of Alpha (7 point)
alpha(1) = ((4.*temp(1)) + temp(2) + temp(3) + temp(4)) / 7.
alpha(2) = ((3.*temp(1)) + temp(2) + temp(3) + temp(4)) +
          + temp(5)) / 7.
alpha(3) = ((2.*temp(1)) + temp(2) + temp(3) + temp(4)) +
          + temp(5) + temp(6)) / 7.
do 90 i = 4, 509
   alpha(i) = (temp(i-3) + temp(i-2) + temp(i-1) + temp(i) +
             + temp(i+1) + temp(i+2) + temp(i+3)) / 7.
90    continue
alpha(510) = (temp(507) + temp(508) + temp(509) +
              + temp(510) + temp(511) + (2.*temp(512))) / 7.
alpha(511) = (temp(508) + temp(509) + temp(510) +
              + temp(511) + (3.*temp(512))) / 7.
alpha(512) = (temp(509) + temp(510) + temp(511) +
              + (4.*temp(512))) / 7.
do 90 i = 1, 512
   alpha1 = alpha(i) + 1.
   alpha2 = alpha(i) - 1.
do 1000 i = 1, 512
   comp1(i) = cmplx. teogh(i), 0. )
1000 continue
c  Inverse Transform
call cifft (comp1, 9, 0, 1.0)
Line# Source Line

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(complex(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
EDGVU          param  0016
GAIN           param  001a
HMAG           param  001e
D4H            param  0022
D3H            param  0026
D4V            param  002a
D3V            param  002e
D2H            param  0032
D1H            param  0036
D2V            param  003a
D1V            param  0042
ALPHA          local  REAL*4   2048  0000
CONST          local  REAL*4   4   00a0
I              local  INTEGER*4 4   00b4
J              local  INTEGER*4 4   00b8
P12            local  REAL*4   4   00c2
K              local  INTEGER*4 4   00c6
N              local  INTEGER*4 4   00d0
CEEG           local  REAL*4   4   00d4
ALPHA1         local  REAL*4   4   00d8
IN             local  INTEGER*4 4   00e2
ALPHA2         local  REAL*4   4   00e4
INTRVL         local  REAL*4   4   00e8
TMPRSSP        local  INTEGER*4 4   00ee
RADDEG         local  REAL*4   4   00f0
TIME           local  REAL*4   4   00f4
FLOAT          local  REAL*4   4   00f8
CONST1         local  REAL*4   4   00fc
ORDER          local  REAL*4   4   0100
TEOGH          local  REAL*4   4   0104
TEOGVL         local  REAL*4   4   0108
TEOGVU         local  REAL*4   4   0112
COMP1          local  COMPLEX*8 4096  0d42
COMP2          local  COMPLEX*8 4096  ex42
Global Symbols

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Type</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCF</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
<td>11f2</td>
</tr>
<tr>
<td>CFFT</td>
<td></td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>CRFT</td>
<td></td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>CRCT</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
<td>1590</td>
</tr>
<tr>
<td>FTUNE</td>
<td></td>
<td>FSUBRT</td>
<td>***</td>
<td>0000</td>
</tr>
<tr>
<td>SGWIN</td>
<td></td>
<td>extern</td>
<td>***</td>
<td>***</td>
</tr>
</tbody>
</table>

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

No errors detected