LABORATORY NOTE NO. 81

The Summer Apprentice Program
1992

Katherine Ellen Renn
Thornton Samuel Mu
David M. Dahle
Vincent K. Lee
Renee M. Ward

In Cooperation with George Washington University
School of Engineering and Applied Science Summer Apprentice Program

Summer 1992
LABORATORY NOTE NO. 81

DTIC QUALITY REPRODUCED B

The Summer Apprentice Program 1992

Katherine Ellen Renn
Thornton Samuel Mu
David M. Dahle
Vincent K. Lee
Renee M. Ward

In Cooperation with George Washington University
School of Engineering and Applied Science Summer Apprentice Program

Summer 1992
This document has been approved for public release and sale; its distribution is unlimited.

Destroy this report when it is no longer needed. Do not return to the originator.

Citation of trade names in this report does not constitute an official endorsement or approval of the use of such items.

This material has been reviewed by Letterman Army Institute of Research and there is no objection to its presentation and/or publication. The opinions or assertions contained herein are the private views of the author(s) and are not to be construed as official nor as reflecting the views of the Department of the Army or the Department of Defense. (AR 360-5)

John R. Hess (date)
COL, MC
Commander
14 June 93
During the summer of 1992, Letterman Army Institute of Research hosted five inquisitive and energetic teenagers as part of the George Washington University School of Engineering and Applied Science Summer Apprentice Program. Each participant was assigned to a LAIR researcher who served as mentor. The pace was vigorous, and the work was challenging. On August 6, 1992, the students presented their final reports to the Commander of LAIR and other dignitaries from the Presidio of San Francisco. A few days later they flew to Washington, D.C. and presented their projects to the other program participants.

The final reports prepared by the summer apprentices are presented in this LAIR Laboratory Note.
Summer Apprentice Program

1992

During the summer of 1992, Letterman Army Institute of Research hosted five inquisitive and energetic teenagers as part of the George Washington University School of Engineering and Applied Science Summer Apprentice Program. Each participant was assigned to a LAIR researcher who served as mentor. The pace was vigorous, and the work was challenging. On August 6, 1992, the students presented their final reports to the Commander of LAIR and other dignitaries from the Presidio of San Francisco. A few days later they flew to Washington, D.C. and presented their projects to the other program participants.

The final reports prepared by the summer apprentices are presented in this LAIR Laboratory Note.
Katherine Ellen Renn

Bloomingdale Senior High School

Randolph-Macon Women’s College

August 5, 1992

Letterman Army Institute of Research

Mentor - Raymond Regan, M.D.

The Effect of Nitric Oxide on NMDA Neurotoxicity
ABSTRACT

During N-methyl-D-aspartate (NMDA) neurotoxicity, NMDA receptors, in response to glutamate, open their channels allowing an influx of intracellular calcium. The processes leading to cell death after an influx of calcium are unknown, but experimental evidence suggests it may be due to nitric oxide (NO). We showed, in our system, that the nitric oxide synthase inhibitors, \( \text{N}^{\text{\text{O}}} \)-nitro-L-arginine and \( \text{N}^{\text{\text{O}}} \)-monomethyl-L-arginine, and NO binding hemoglobin do not prevent NMDA neurotoxicity. These agents also blocked the formation of cyclic guanosine monophosphate. These data suggest that NO does not mediate the neurotoxicity of NMDA.
The effect of nitric oxide on NMDA neurotoxicity

Katherine E. Renn

Stroke is the third leading cause of death in the United States, claiming approximately 500,000 victims each year (1). Thirty percent of stroke victims die and 20% to 30% become permanently and severely disabled (1). Experimental evidence has supported the hypothesis that during stroke, excessive amounts of glutamate are released by ischemic neurons. Studies have suggested that in response to glutamate, N-methyl-D-aspartate (NMDA) and DL-α-amino-3-hydroxy-5-methylisoxazole-propionic acid hydrobromide salt/kainate (AMPA/kainate), types of glutamate receptors, open their channels allowing the influx of calcium and sodium (1). AMPA/kainate receptors only allow the passage of sodium. An abnormal build-up of both ions may occur. The increased build-up of calcium may trigger the release of glutamate, thus spreading the calcium cascade to other cells (1). The excess calcium may activate proteases, lipases, and endonucleases. These enzymes may degrade nucleic acids, proteins, and lipids. The metabolism of arachidonic acid, a by-product of the degradation of phospholipids, leads to the formation of oxygen-free radicals (1). An overabundance of calcium also stimulates the formation of nitric oxide (2).

The mechanism for the formation of nitric oxide in the central nervous system (CNS) is still not fully understood. It is theorized that glutamate released from presynaptic terminals binds to NMDA
2 - Renn

receptors. Calcium rushes in and binds with calmodulin, thereby activating NO synthase (2,3). Arginine, already present in the CNS, is converted, by NO synthase, to equal amounts of NO and citrulline (3).

In a recent study Dawson et al., reported that the NO antagonists N-\(\omega\)-nitro-L-arginine (N-arg) and N-\(\omega\)-monomethyl-L-arginine (M-arg) attenuated glutamate neurotoxicity in cortical cell cultures (4). In this study, NMDA neurotoxicity was associated with an increase in NO, which was likely mediated by the entry of calcium ions (4). Nitric oxide was also shown to mediate ischemic neuronal injury in rats (5). This study, conducted by Nowicki et al., showed that N-arg, a competitive inhibitor of NO synthase, reduced the volume of infarction (5). In a contrasting study by Demerle-Pallardy et al., the NO pathway was not involved in the neurotoxicity in the brain because the production of cyclic guanosine monophosphate (cGMP) by glutamate was totally blocked by the NO synthase inhibitors with no attenuation of neuronal injury (6). In another study conducted by Manzoni et al., nitric oxide was shown to protect against NMDA induced currents and the associated increase in intracellular calcium (7). Manzoni tested the effects of NO-induced ionic fluxes on NMDA by using drugs that produced NO. These drugs all blocked NMDA-induced currents.

The conflicting evidence suggests that more research needs to be done to determine the precise role of NO in the CNS. The discrepancy in results may be due to different methods, procedures, or distinct origins of neuronal cells. It is worthwhile to develop a detailed understanding of
the precise role of NO in NMDA neurotoxicity. This information could have implications to limit stroke-related brain damage. We performed this study in neuronal cell cultures to assess the role of nitric oxide in NMDA neurotoxicity.

MATERIALS AND METHODS

Cell culture: Cortical cell cultures consisted of both neuronal and glial cells. Cells were obtained according to the method of Regan et al. (8). Neuronal cells were obtained from fetal Swiss-Webster mice at 15-17 days gestation. The neocortex was removed, minced, and placed in media containing 0.08% acetylated trypsin at 37°C for one hour. After centrifugation, the cells were placed in plating media consisting of Eagle's minimal essential media (MEM), 10% heat-inactivated horse serum, 10% fetal bovine serum, glutamine (2 mM), and glucose (21 mM). The cells were triturated through a flamed Pasteur pipette, diluted in plating media, and plated on pure glial cultures. The cells were plated in 15 mm multiwell plates (2.5 x 10^-5 cells/well). The cultures were maintained at 37°C in a 5% CO2 incubator, and the media was changed twice weekly with growth media lacking fetal serum.

Glial cultures were obtained from neonatal Swiss-Webster mice. The dissection and dissociation was the same as the above method. The cells were plated on 15 mm Primaria (Falcon) multiwell plates at a
density of 0.5 hemispheres per plate. Glial cultures were used primarily as a feeder cell-layer for neurons.

**Animal care and handling:** Pregnant female Swiss-Webster mice were anesthetized with halothane and euthanized by means of cervical dislocation. The fetuses were removed, using sterile techniques, and immediately decapitated.

**Cytotoxicity:** Cells 15-19 days in vitro (DIV) were exposed to excitatory amino acids (EAA’s) according to the methods of Regan et al. (8). Prior to exposure, the cells were washed with a HEPES (N-[2-hydroxyethyl]piperazine-N’-[2-ethanesulfonic acid])-buffered controlled salt solution (HCSS) to remove MEM. HCSS has the following composition: NaCl, 120 mM; KCl, 5.4 mM; MgCl₂, 0.8 mM; CaCl₂, 1.8 mM; HEPES, 20 mM; glucose, 15 mM. The cells were exposed to NMDA in HCSS for 5 minutes. The experimental solution was then washed with MEM. Cells were placed in an incubator for 20-24 hours. Long exposures to EAA’s were performed in MEM with added glucose (25 mM).

**LDH assay:** Cell injury was quantified by measurement of the lactate dehydrogenase (LDH) released by damaged cells. Each LDH value was scaled to the mean value obtained by the control NMDA exposure. The LDH test was performed using the methods described by Regan et al. (8).
cGMP assay: After the application of NMDA 300 µm (5 min), the cell culture was placed in ice cold 65% ethanol for 5 min. The supernatant was extracted twice and lyophilized. The dried extracts were dissolved in the assay buffer, and the cGMP was measured with a cGMP enzyme assay kit purchased from Amersham.

NADPH-diaphorase staining: Control cultures and cultures exposed to EAA's were stained for reduced nicotinamide adenine dinucleotide phosphate diaphorase (NDP), described by Koh et al. (9). Cultures were fixed for 30 min in 4% paraformaldehyde at room temperature, and incubated in medium containing 1 mM NADPH and 0.2 mM nitro blue tetrazolium in 0.1 M Tris buffer (pH 8.2) at 37°C for 30 min to 1 hr. The staining action was ended by washing with water.

Reagents: All chemicals were purchased from Sigma (St. Louis, MO). The cGMP assay kit was obtained from Amersham (Amersham, UK).

RESULTS

By DIV 15, neurons in culture could be readily identified by their phase-bright cell bodies and extensive processes. The neurons were stained for NDP as an indirect measurement of cells containing NOS (9). Less than 1% of the neurons in these cultures were stained (Fig. 1).
Within minutes after exposure to NMDA 300 μM (5 min), neurons were acutely swollen. Over the next few hours, swollen neurons were beginning to deteriorate. At 24 hours the neurons were phase-dark and fragmented. The neurons were also releasing substantial amounts of LDH into the surrounding medium. Neuronal cell death was recorded 24 hours after NMDA (300 μm) exposure (Fig. 2). About 80% of the neurons were injured; the glial cell layer was left uninjured.

The simultaneous application of the NOS inhibitor 1 mM N-arg with NMDA 300 μM (5 min) did not provide protection against neuronal death (Fig 2). The application of 1 mM M-arg also did not provide protection. Approximately 75% of the neurons were injured with each application. To further ascertain the role of nitric oxide in NMDA neurotoxicity, we applied reduced human A_o hemoglobin (Hb), which binds NO, simultaneously with NMDA (300 μM). Hemoglobin provided no protection against NMDA-mediated neuronal injury (Fig. 3). Simultaneous application of the noncompetitive NMDA antagonist MK-801 completely blocked neuronal injury.

To determine if NO production was effectively blocked by the concentration of NOS inhibitors used, we assayed cGMP levels. The application of NMDA 300 μM (5 min) stimulated the formation of cGMP (Fig. 4). The formation of cGMP was completely blocked by the application of N-arg (1 mM) and Hb (500 μM). In fact, these cGMP levels were lower than baseline levels. Methyl-arginine, another NOS
inhibitor, less potent than N-arg (4), also partially blocked the formation of cGMP levels (Fig. 4).

**DISCUSSION**

Although Demerle-Pallardy used whole rat brain, our results were in agreement with their findings that NO was not shown to be involved in NMDA neurotoxicity (6). In contrast to Dawson et al., N-Arg and M-Arg (inhibitors of NOS) did not block NMDA-induced neuronal death (Fig. 2). Also, Hb (100 μM or 500 μM) did not protect against neuronal injury (Fig. 3). The discrepancy in results may be due to different methods, procedures, or distinct origins of neuron cells.

The cells exposed to NMDA 300 μM (5 min) resulted in approximately 80% neuronal death. A study by Dawson et al., suggests that N-arg and M-arg (NOS inhibitors) would block NO mediated neuronal death because they inhibit NOS (4). This inhibition would stop NOS from converting arginine into NO. Yet, in our system, N-arg and M-arg were not effective in blocking neuronal death. Approximately 70% neurons died with the simultaneous application of NMDA 300 μM (5 min) and the NOS inhibitors. Dawson et al's. evidence also suggested that Hb (which binds to NO) would block neuronal death (4). Our cultures were not protected against injury; approximately 80% neuronal death was recorded with this application. To prove that the concentrations of N-arg, M-arg, and Hb which we used were effective in
blocking the production of NO, we assayed the cGMP levels. The results suggest that N-arg, M-arg, and Hb were effective in blocking the formation of cGMP. These results suggest that NMDA-mediated neuronal death is independent of NO.

CONCLUSION

We showed that the nitric oxide synthase (NOS) inhibitors, nitro-arginine (N-arg) and methyl-arginine (M-arg) do not prevent NMDA neurotoxicity. Also, the application of hemoglobin (Hb) did not prevent neuronal injury. These data establish that NO does not mediate the neurotoxicity of NMDA. The discrepancy in results may be due to different methods, procedures, or distinct origins of neuron cells.
ACKNOWLEDGMENTS

Thanks to

Dr. Regan, Dr. Panter, Dr. Wheeler, COL Brown, SSG Thomas,
Susan Siefert, Donna Shepard, and Andre Akers.

ANIMAL CARE AND HANDLING STATEMENT

The opinions and assertions contained herein are the private views of the authors and are not to be construed as official nor do they reflect the views of the Army or the Department of Defense. (AR 360-5)

The experimental studies of the author described in this report were reviewed and approved by the Institutional Review Committee/Animal Care and Use Committee at Letterman Army Institute of Research. The manuscript was peer reviewed for compliance prior to submission for publication. In conducting the research described here, the author adhered to the "Guide for the Care and Use of Laboratory Animals," DHEW Publication (NIH) 85-23.
REFERENCES


Fig. 1, NADPH-diaphorase neuron stain. Three NDP neurons with darkly stained cell bodies can be seen against a background of other largely unstained neurons and glia.

Fig. 2, Protection provided by N-arg (1 mM) and M-arg (1mM). Amount of LDH present in bathing media was measured 24 hours after the application of NMDA 300 µm (5 min) and antagonists (mean ± S.E.M., n=4 cultures at each point). LDH release was scaled to the mean value evoked by exposure to 300 µm NMDA, 24 hrs (=100).

Fig. 3, Protection provided by Hb (100 µm and 500 µm). Amount of LDH present in bathing media was measured 24 hrs after the application of NMDA and antagonists (mean ± S.E.M., n=4 cultures at each point). LDH release was scaled to the mean value evoked by exposure to 300 µm NMDA, 24 hrs (=100).

Fig. 4, Formation of cGMP after the application of NMDA with or without N-arg 1 mM, M-arg 1 mM, or Hb 500 µm. Data are means ± S.E.M.
FIGURE 2

LDH Released (% Control)

<table>
<thead>
<tr>
<th></th>
<th>CONTROL</th>
<th>NMDA</th>
<th>NA</th>
<th>MA</th>
<th>MK801</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Indicates significant difference from control.
FIGURE 3

LDH RELEASE (% CONTROL)

CONTROL  NMDA  Hb,100 μM  Hb,500 μM  MK801
FIGURE 4
Thornton Samuel Mu
Lowell High School
August 12, 1992
Letterman Army Institute of Research
Mentor - Michael A. Dubick, Ph.D.
Pyridoxine deficiency and wound healing in rats:
Assessment of antioxidant enzymes and
immunological status
Abstract

Many nutrients are involved in the complex process of wound healing. Of the vitamins, pyridoxine (vitamin B₆) deficiency has resulted in impaired healing, most likely due to its effect on protein metabolism. This study examined the effects of pyridoxine deficiency on wound healing and its effects on acute phase proteins, immunoglobulins, and antioxidant enzymes in rats. The results showed that despite pyridoxine deficiency, the animals in each dietary group responded the same toward injury.
Pyridoxine deficiency and wound healing in rats:
Assessment of antioxidant enzymes and immunological status -- Thornton Mu

Wound healing is a complex process that takes place in many stages (1) and requires the presence of many important hormones, nutrients, and proteins, such as acute-phase proteins, immunoglobulins, and proteins that regulate dangerous oxidants (2-5). Free radicals formed by these oxidizers can cause serious cell injury, such as lipid peroxidation (6).

Among the nutrients, pyridoxine (vitamin B₆) plays quite an important role in the synthesis and maturation of connective tissue (7). Consequently, pyridoxine deficiency can affect normal wound healing (8). Pyridoxine also possesses an antioxidant function that prevents damage done by free radicals (9), functions as a coenzyme in the metabolism of amino acids (10), and is important for normal immune function (11). Because proteins, such as immunoglobulins, antioxidant enzymes, and acute-phase proteins, are composed of amino acids, it is assumed that a deficiency in vitamin B₆ would adversely affect production of these proteins.

We proposed that pyridoxine deficiency would impair the ability to respond to injury. In this study we examined the effects of various stages of pyridoxine deficiency on the status and presence (activity) of
antioxidant enzymes and on the concentrations of certain acute-phase proteins and immunoglobulins in an incisional wound model in the rat. A better understanding of the role of pyridoxine could lead to improved therapeutic regimens or preventive measures, thus reducing mortality and pain for the injured subject.

Materials and Methods

Experimental protocol

Young adult, female Long-Evans rats, initially weighing 175-200g were employed in these studies. A total of twenty-four rats were used in the study. The rats were randomly assigned to four groups of six rats. The first group was fed a diet devoid of (deficient) vitamin B₆. The second group was fed a diet with 0.25 mg vitamin B₆ per kg. The third group was fed a diet with 1 mg/kg vitamin B₆. The fourth group was fed a control diet with 7 mg/kg vitamin B₆. All four groups were fed their respective diets for a period of five weeks. Three rats within each group underwent surgery in which an incision was made through the full thickness of the skin on the dorsal side. The rats recovered during one week after the surgery and were fed their respective diets during the recovery period.
Biochemical Studies

At the end of six weeks, the rats were overdosed with pentobarbital (120mg/kg). Blood was withdrawn by cardiac puncture into citrated tubes and plasma was recovered by centrifugation for further evaluation of vitamin B₆ levels, immunoglobulin G and M, albumin, and fibrinogen. Liver, kidney, and brain were quickly removed, snap frozen in liquid nitrogen, and stored at -70°C in a biological freezer until assayed.

In preparation for assays to determine glutathione peroxidase, glutathione reductase, superoxide dismutase, and protein content, samples of kidney, liver, and brain were homogenized separately using the Polytron homogenizer (Brinkmann Instruments, Westbury, NY) in a homogenizing buffer composed of 0.25M sucrose and 10mM Tris-HCl, pH 7.4 for ten seconds. Using a Branson sonifier cell disruptor (Branson Ultra Sonic, Danbury, CT), the samples were sonicated three times for a period of five seconds each. The samples were centrifuged for thirty minutes at 10,000g in a RC-5 Superspeed Refrigerated Centrifuge (DuPont Instruments, Newton, CT).

The samples were tested for glutathione peroxidase by using hydrogen peroxide as a substrate (12) and tested for glutathione reductase activity (13) by measuring the disappearance of beta-nicotinamide adenine dinucleotide phosphate (beta-NADPH) using the
Beckman DU-70 Spectrophotometer (Beckman instruments, Fullerton, CA).

Total superoxide dismutase (SOD) (12) activity was measured by the ability of the enzyme to inhibit auto-oxidation of pyrogallol using the Beckman DU-70 Spectrophotometer. Manganese SOD activity (12) was measured under the same conditions except the buffer contained 1mM KCN. CuZnSOD was calculated by subtracting MnSOD activity from the total SOD activity.

Immunoglobulin G and M (IgG and IgM, respectively) concentrations in plasma were determined in diffusion plates (Kallestad, Chaska, MN) by double immunodiffusion (14). The results were recorded after 24 and 48 hrs incubation but only the 48 hr values were used according to standard protocol in clinical laboratories. Ten μL of antibody was placed in the center well with dilutions of plasma placed in the outer wells. For IgM, neat through 1/32 dilutions were used and for IgG, 1/32 to 1/1024 dilutions were used.

Fibrinogen and albumin concentrations were determined by rocket immunoelectrophoresis (150 using the Bio-Rad horizontal electrophoresis cell (Bio-Rad laboratories, Richmond, CA). Albumin concentrations were determined from standards, while fibrinogen concentrations were calculated as arbitrary units based on dilutions of normal rat plasma.
Protein content was determined with a commercially available kit (Bio-Rad, Richmond, CA) using the Beckman Du-70 spectrophotometer.

Statistical Analysis

Data were analyzed by two way ANOVA with injury and diet as factors. Significant differences were further evaluated by the Newman-Keuls method for multiple comparisons. A p<0.05 was considered statistically significant.

Animal Use Statement

Animal quarantine and prior care were performed in accordance with SOP# OP-ARG-4 and OP-ARG-40 and conforms to the provisions of the NIH as stated in the Guide for the Care and Use of Laboratory Animals (NIH publication #85-23).

Results

Pyridoxine status

Tests to determine pyridoxine status in rats showed that rats with no vitamin B₆ in their diets were indeed deficient in vitamin B₆ whereas the control rats with 7mg/kg of pyridoxine had normal vitamin B₆ concentrations in their plasma. The other two groups had intermediate vitamin B₆ levels in their plasma (data not shown).
Acute-phase proteins (Table 1)

Concentrations of fibrinogen were not significantly affected by either diet or injury whereas albumin concentrations were significantly lower in rats with injury (p=0.0023) and those with pyridoxine deficient diets (p=0.0002).

Immunoglobulin G and M

Concentrations of IgG and IgM were not significantly different among the dietary groups and injury versus noninjury groups, but were relatively constant.

Antioxidant enzyme status (Table 2)

Activity of glutathione peroxidase per gram of tissue (GPx/g) was significantly higher (p=0.0240) in the livers of rats deficient in vitamin B\textsubscript{6} compared to controls. However, in brain and kidney samples, GPx activity was not significantly affected by the different vitamin B\textsubscript{6} diets. In all dietary groups, GPx activity from brain samples from injured rats showed a significant decrease (p=0.0229) compared to their uninjured counterparts whereas in the liver and kidney samples, a nonsignificant trend of decrease was noted.

In the rats that were deficient in vitamin B\textsubscript{6}, activity of glutathione reductase per gram of tissue (GR/g) was significantly lower
(p=0.0278) in liver samples, but significantly higher (p=0.0034) in brain samples compared to controls. Activity of GR/g in kidney was not significantly affected by diet or injury.

Differences in manganese superoxide dismutase activity per gram of tissue (MnSOD/g) were not significant among the diet and injury status of the three tissue groups.

However, activity of copper zinc superoxide dismutase per gram of tissue (CuZnSOD/g) was significantly lower (p=0.0391) in kidney samples from rats fed deficient pyridoxine diets but was significantly higher (p=0.0390) in brain samples with the same deficient diets compared to controls. Liver sample activity remained non-significant. In terms of injury status, CuZnSOD/g activity was non-significant among the three tissues.

Discussion

It has already been shown that vitamin B₆ affects protein metabolism (10), and in our studies, protein levels of rats with diets deficient in vitamin B₆ were significantly reduced (data not shown). However, similar to studies by Schaeffer et al. (16), body weight of all rats remained relatively constant despite the varying amount of pyridoxine in their diets (data not shown). Therefore, to insure more
stable readings, activity of the antioxidant enzymes measured were expressed per grams of tissue rather than grams of protein.

Our results have shown that there were few significant decreases in enzyme activity due to changes in pyridoxine diet or injury status. One could possibly assume that with a shortfall of protein, amounts of other proteins and enzymes could also decrease. Yet this assumption did not hold for the enzymes and proteins measured in our studies because fibrinogen, IgG, IgM, GPx, GR, MnSOD, and CuZnSOD concentrations did not decrease significantly. Instead, in tissues in which antioxidant enzymes activity was significantly affected by diet, the values were usually higher in the deficient group than in the controls.

One possible explanation for these findings is that the one week allowed for recovery from surgery may have been enough time to synthesize the enzymes and proteins listed above. In addition, a previous study done by Black et al. showed that rats with pyridoxine-deficient diets maintained a vitamin B₆ reserve in their gastrocnemius muscle (17). Therefore, although the plasma revealed a pyridoxine deficiency, the rats may have been relying on this storage to provide tissues with enough pyridoxine to synthesize the needed enzymes and proteins to respond to the stress of the injury induced.

Albumin levels, on the other hand, showed a significant decrease
both in the vitamin B₆ group and the injured group. Albumin in plasma was affected by pyridoxine deficiency, but the reaction of albumin as an acute-phase protein following injury was not impaired.

In conclusion, pyridoxine deficiency did not significantly affect the rat's response to surgically-induced wound. The antioxidant status of the tissues examined in the rats remained at control level, or higher, and the plasma concentrations of immunoglobulins measured were unaffected by the pyridoxine deficiency. Also, the acute-phase protein response to injury was not adversely affected by the pyridoxine deficiency. More research is needed to gain a better understanding of the effects of vitamin B₆ in the realm of wound healing.
Acknowledgements

I would like to thank Dr. Michael Dubick and Dr. X.Q. Yuan for their sharing their vast knowledge, learning, and experience with me; Dr. Virginia Gildengorin for assisting with the statistical analyses; Ms. Susan Siefert for helping me prepare this paper; and the Military Trauma Research Division and COL Brown, MAJ McCollum and Mr. William Ward of the Command Division at the Letterman Army Institute of Research for giving me this tremendous opportunity.
REFERENCES


12. Zidenberg-Cherr S, Han B, Dubick MA, Keen CL. Influence of dietary-induced copper and manganese deficiency on ozone-induced...
12 -- Mu

...changes in lung and liver antioxidant systems. Toxicol Lett 1991;57:81-90.


David M. Dahle
Tamalpais High School
August 5, 1992
Letterman Army Institute of Research
Mentor - Kim Vandegriff, Ph.D.
Data Acquisition for Experiments in
Hemoglobin Oxygen Binding
ABSTRACT

This report presents an Assembly Language program that was designed to collect experimental data in the form of analog voltage signals and convert them into digital values for computer storage and analysis. These voltage signals represent the concentration of oxygen in a reaction cell for an experiment which tests the function of hemoglobin by measuring the color change of the hemoglobin solution as it transforms from deoxyhemoglobin to oxyhemoglobin upon the addition of oxygen. The program interfaces the experiment with an analog-to-digital converter that improves the accuracy of the oxygen concentration measurements. The program also incorporates specific timing sequences to allow synchronous data acquisition. The Assembly Language program employs user-friendly interfacing and several automated features that result in saving of time and effort on the part of the researcher.
DATA ACQUISITION FOR EXPERIMENTS IN HEMOGLOBIN OXYGEN BINDING

David M. Dahle

This project in computer science was designed to translate experimental data signals into digital files for data analysis. The data translation procedure will be used in experiments to aid in the development of a hemoglobin-based blood substitute.

The biochemical experiment involves testing the structure and function of hemoglobin by measuring its transition from deoxyhemoglobin to oxyhemoglobin as oxygen is added and the protein changes color from purple to red. The change in color of hemoglobin is measured optically by rapid-scanning spectrophotometry, which records the amount of light absorbed by hemoglobin at each wavelength in the visible spectrum from 400 to 800 nanometers. The absorbance values for each wavelength are measured as rapidly scanning monochromatic light is transmitted through a reaction chamber containing the hemoglobin solution. Oxygen concentrations are recorded simultaneously as a voltage level, which is produced by an electrode fitted in the reaction cell. This new methodology measures the
The change in the hemoglobin spectrum at 5 scans per second.

This project involves the aspect of the experiment that records the oxygen concentrations in the reaction cell containing the hemoglobin, which is done simultaneously with the recording of the absorbance values. The program that records the oxygen concentrations receives this information as an analog voltage signal. It converts the voltage to digital values using an analog-to-digital converter. It synchronizes its operations with the recording of the absorbance values using a digital signal sent by the spectrophotometric system. The oxygen concentration values are then combined with the absorbance values and analyzed. Figure 1 shows these data graphed in 3-dimensions. The graph begins with deoxyhemoglobin and ends with oxyhemoglobin.

This project was undertaken to upgrade an outdated data acquisition system in an effort to improve the accuracy of the oxygen concentration readings. Other goals of the project were to incorporate a user-friendly interface and a flexible design so that the program would be easier to operate, making data
acquisition as simple as possible. The new data acquisition system arrived in the laboratory just before I did, so I began the project by first installing the computer board, fabricating new cables, and then developing the program.

EQUIPMENT USED

COMPUTERS

MS-DOS 80286.
CompuAdd Tower MS-DOS 80486.
GoldStar MS-DOS 80386.

Digital Equipment Corporation network system and software.

PROGRAM DEVELOPMENT

WordPerfect Version 5.1.
Microsoft Windows Version 3.1.
Microsoft DOS Version 5.0.
WordStar Professional Version 5.0.

COMPILERS/ASSEMBLERS

Borland International Turbo Assembler Version 1.0 and utilities.

Microsoft C Optimizing Compiler Version 5.0, Version 5.1 and utilities.
EXPERIMENTAL APPARATUS

LT Quantum 1200 Spectrophotometric System (LT Industries Rockville, Maryland) is used to make the spectral measurements of the hemoglobin.

YSI Model 5300 Biological Oxygen Monitor System (Yellow Springs Instrument Co., Inc. Yellow Springs, Ohio) is used to create the oxygen voltage concentration signal.

Analog Connection MINI-16 Data Acquisition and Control System Analog to Digital Converter (Strawberry Tree Computers, Inc. Sunnyvale, California) is used to convert the analog signal to a digital value so it can be stored and used in a computer.

EXPERIMENTAL METHOD

The experiment has two components which run synchronously. (See Figure 2). The first component is the spectrophotometric system which makes the absorbance value readings. The software for this system controls the rapid-scanning spectrophotometer and collects data from it. The data are recorded as absorbance values for each wavelength in the visible light spectrum from 400 nanometers to 800 nanometers. Absorbance values at each wavelength are measured as
rapidly-scanning monochromatic light is transmitted through the hemoglobin solution. The system also signals the oxygen concentration recorder via a TTL (Transistor to Transistor Logic) pulse, so the oxygen concentration recordings can be synchronized with the spectral scans. This is a digital signal that represents two states, on or off, or, in this case, "record" or "don't record" oxygen concentrations.

The second component of the experiment measures the amount of oxygen in the reaction chamber simultaneously with the spectral readings. The system begins with a Clark-type, fast response polarographic electrode that is fitted in the reaction cell containing the hemoglobin solution. It generates an analog voltage signal based on the concentration of oxygen in the reaction cell. The signal goes to an oxygen monitor which amplifies and calibrates it, then to a filter to reduce line noise. A voltmeter gives a display of the approximate voltage signal. The signal then travels to the analog-to-digital converter in the computer. This converter has 8 analog input/outputs and 16 digital input/outputs. An analog port is used to read the oxygen voltage concentration signals and
convert them to digital values which only then can be stored in the computer. A digital port is used to read the TTL signal so that the oxygen concentration recorder will run synchronously with the spectrophotometric system.

PROGRAM DESCRIPTION

The program relies on a device driver to run the hardware, so that the driver must be installed before the program can be run. When the program starts it verifies that the driver has been installed. If it has not, the program displays an error message, waits for the user to hit any key, then exits. If the device driver has been installed, the program accesses the functions by placing parameters on the stack and calling interrupt vector 60.

After the test for the presence of the software, the program loads a default configuration file which contains the settings for all the configuration fields and the date they were saved. This date is used by the Automatic File Names feature. This feature enables the program to automatically create names for the files based on the date and the current experiment number. It creates file names by generating an 8-character
numerical string, the first two characters being the month of the year, the next two the day of the month, the next two the last two digits in the year, and the last two the experiment number (i.e., 12259250 for December 25, 1992, experiment number 50). After each experiment is run without errors, the program automatically increments the experiment number. When a configuration file is written, the date to which the experiment number corresponds is also saved. When the file is loaded again, the program compares the date in the file with the current date. If they are the same, it restores the experiment number from that file, if they are different, it resets it to 1. This allows the experiment number to act as an automatic counter for the number of experiments run in a day, even if the program is run several separate times in the same day.

After the configuration file is loaded, the user is given a list of several functions which can be performed. The user makes selections from these by using the left and right arrow keys. Each item will become highlighted as it is selected. The user can execute the currently highlighted item by hitting the return key. If the user presses the up or down keys
he/she can scroll through the configuration parameters. Highlighted configuration items can be toggled with the return key. String entry fields are used to read file names from the user and function in the same way as any other string input field.

The Average File Name field specifies the file in which the averages of all the readings for each data set will be written. The Data File Names field specifies the name for the data files in which every reading from each set will be written. An extension number corresponding to the current set is used so the data for each set can be distinguished, and any extension given will be replaced. If no file name is given for either of the two file name fields, then no file will be created. The Experiment Number field specifies the number of the current experiment, which is used by the Automatic File Names feature. This number is ignored by the program if this feature is turned off. The Automatic File Names field allows the user to turn on or off the automatic file naming feature. The Print Elapsed Time field allows the user to specify if he/she wants the program to compute the length of time that the TTL pulse was active and the
data rate in Hertz (Hz). The Analog Input Range field is used by the device driver to indicate the projected voltage range of the input signal. The Analog Input Resolution field is also used by the device driver to determine the precision of the readings. This field is also used to set the speed at which data are sampled. The less precision specified, the faster the data are converted.

The first item from the list of functions is the Begin item which is used to start data acquisition. Once started, the function list will be overwritten by a message informing the user that he/she can end data acquisition by pressing any key. If any file or memory errors occur, then an error message is displayed and acquisition is aborted. If the Automatic File Names feature has been activated, then the file names are given just before acquisition begins.

Acquisition begins by waiting for a drop in the TTL line from the spectrophotometric software. This line is connected to one of the digital input ports on the analog-to-digital converter. It remains in an active state (low-voltage mode) until the specified number of scans has been completed. When spectral data
collection is finished, the signal returns to its inactive state (high-voltage mode). The program senses the change in the TTL signal and stops recording voltage readings. An average of all of the readings in each set is computed and written to an output file, if a file name was given in the Average File field. If the user specified a file name in the Data File Names field, then each reading in the set will be written to disk using the name given plus an extension corresponding to the current set number. This process is repeated until the user cancels the operation by hitting any key, allowing sets of readings to be taken.

The next item in the function list is Instant which continuously reads both the analog voltage and the TTL line state so that the user can verify the connections before an experiment begins. This can be done by comparing the voltage readings with those on the voltmeter.

The Load function enables the user to read a configuration file with a name other than the default name. A string input field appears just below the function list in which the user can enter the new file name. He/she can abort by hitting the escape key.
When done, the user presses the return key to write the file.

The Save function allows the user to write a configuration file with a name other than the default one. A string input field appears and functions the same as the Load function.

The last item in the function list is the Exit option. When this function is selected, the program writes the current configuration settings to the default configuration file, then returns to the parent process, normally the MS-DOS command prompt.

**EXPERIMENTAL DATA**

Figure 3 shows the fractional saturation of hemoglobin calculated as a function of the partial pressure of oxygen. The fractional saturation of hemoglobin (Hb) is computed using the following formula:

\[
\text{Fractional Saturation} = \frac{\%\text{oxyHb}}{\%\text{oxyHb} + \%\text{deoxyHb}}.
\]

**DISCUSSION**

The program went through two versions. The first was written in the C Programming Language (see Appendix A for source listing). The second version was written almost entirely in 80286 Assembly Language, only
functions which involve the processing of floating-point variables were written in C (see Appendix B for source listing). The rewriting of the program solved several problems and allowed me to discover better solutions for old ones.

A few improvements were also made from the old system which was in place previously, beyond the increased accuracy provided by the new data acquisition system. These include the use of 8-byte doubles instead of 4-byte floats in computing the averages. This increased the number of decimal places used in the calculation and, therefore, the precision of the computation.

A major difference was discovered between the new data acquisition and the old one. In the old system, the converter would start recording as usual with the TTL pulse. Samples were then taken at a specified rate until the TTL pulse went inactive. The number of readings might vary if the timing of the spectrophotometric system was not always accurate, but the samples were always taken at the same rate and within the correct time period. A specific sampling frequency in Hz could be approximated by setting the
rate at which readings were taken and by computing the time that spectrophotometric system would take, which is known. This was required so that a sampling frequency could be set to minimize line noise.

On the new system there is no direct way to set the sampling frequency. It can only be set indirectly through the resolution of the analog-to-digital conversion. It also has no feature to automatically start or stop recording with a TTL signal. This problem at first led us in the direction of attempting to compute the rate at which samples would be taken based on the resolution and the clock speed of the computer. We thought we could then set the number of readings so that the sampling time would cover the same time that the spectrophotometric system was recording, which could be computed as before. But problems were found with this solution. The first was that this method left no room for error on the part of the spectrophotometric system in terms of the time it takes in making its measurements. If for some reason it took more or less time, the oxygen recorder would have no way of knowing this, and the data would not all be from the same time period. The manual for the new converter
also stated that the rate of data conversion would increase with the clock speed of the computer, but it gave no further information and no method for calculation of this function. A formula for computing the rate was found in an example program, but worked only for 4.77 MHz machines, and we are using a faster machine.

Because the issue of primary importance is to record the data simultaneously, a trade off was made and a method similar to the old one was used. The data recording would stop only when the TTL pulse returned to an inactive state. This increased program overhead, and the sampling frequency was reduced by approximately 20% because the TTL pulse had to be checked between each reading. This was an acceptable solution because only the averages of the readings in a set are commonly used during data analysis. During testing, a resolution was found that would produce a Hz close to that used in the old system (i.e., 120 Hz), so the new solution worked, though not elegantly.

The first version of the program uses the first method described to solve the timing problem. It is capable of taking more samples per set, but the timing
is unsure. The second version uses the second method, it takes fewer readings per set, but the timing is always correct. Neither version is capable of implementing a specific Hz, due to the hardware limitations.

There is also a minor concern in computing the elapsed time. The clock used to read the time updates only 18.2 times per second, which allows for approximately a 5.4-hundredths of a second variance. But the values computed from this time are only for reference and are not used in any other calculations, so this may not be of much consequence.

A bug was found in the driver routines which caused the program to crash under certain conditions. Whenever a certain function in the driver was called and the direction flag of the processor was set, the driver ended up attempting to execute instructions in the interrupt vector table, the beginning of memory. The direction flag is used by the processor during string instructions to determine the direction of the strings in memory, either forwards or backwards. Normally this flag is set properly before any string instruction is used and does not need to be in any
predetermined state, but the device driver function does not reset this flag. A solution was found by inserting an instruction to clear this flag before calling any device driver function. Forgetting to reset the direction flag is a common programming error on the 80x86.

IMPLICATIONS

This project has accomplished its two objectives. The first was to improve the accuracy of data acquisition, which is mostly a result of the new data acquisition system. The second was to make the acquisition of data as simple as possible for the user. This has been accomplished by designing a user-friendly interface and automating as much of the process as possible, leaving more time and energy for the experiment. Much improvement has been made in this area as compared with the old program. As a result, the data will not only be more accurate but easier to collect.

CONCLUSIONS

This project only took five of the eight weeks that I spent as an apprentice at LAIR. I also wrote a program to process the spectral data from the
spectrophotometric system (see Appendix C for source listing). The original two programs run on a VAX network system because they require such a large amount of memory. Because the experiment is going to be moved next year to a location where a network may not be available, a program was needed that would not require the network. The new program uses disk buffers instead of memory and is therefore much slower, but if a RAM disk is used, program performance is greatly enhanced. At any rate, slowly acquired data are better than no data at all.

Many other smaller tasks were undertaken along the way. These include but are not limited to the installation and trouble shooting of several programs, the optimization of (though not understanding of) a Fortran program that models oxygen diffusion in and out of red blood cells, and the successful installation of OS/2 on one computer. One of the most significant accomplishments was learning Intel 80x86 Assembly Language during the first weeks, which enabled the second version of the data acquisition program to be written.
ACKNOWLEDGEMENTS

I especially wish to thank my mentor, Dr. Kim Vandegriff, Ph.D., for accepting me into this program and for taking so much time away from her work to assist me.

I would like to thank Yvonne Le Tellier for helping me so closely during the first days, when I most needed it.

I would also like to thank Turney Steward for all the insightful discussions on programming languages and philosophies he had for me.

Finally, I would like to express my gratitude to Susan E. Siefert for her assistance in the writing and editing of this paper.
RESOURCES


FIGURES

1. This graph represents all the data collected in this experiment graphed in 3-dimensions. The x-axis represents the wavelength values from 480 to 650 nanometers, in 1-nanometer steps, the y-axis the absorbance values, and the z-axis the partial pressure of oxygen over time.

2. This figure shows the different components of the experimental system. The reaction chamber, oxygen concentration electrode, monitor and recorder, the spectrophotometric system and its control of the oxygen system with the TTL pulse, the gas-flow system to oxygenate the hemoglobin, and the system to keep the temperature in the reaction chamber constant.

3. This graph represents the fractional saturation of hemoglobin computed as a function of the partial pressure of oxygen in mm Hg. The fractional saturation of hemoglobin is computed using the following formula:

\[
\text{Fractional Saturation} = \frac{\%\text{oxyHb}}{(\%\text{oxyHb} + \%\text{deoxyHb})}.
\]
Appendix A

Version 1 of the oxygen voltage concentration recorder.
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <signal.h>
#include <ctype.h>
#include <dos.h>
#include <string.h>
extern void SHCALL(char *, float *, unsigned short *);
#define ANACNT 8  /* Number of analog channels */
#define DIGCNT 12 /* Number of digital channels */
#define TRUE 1L
#define FALSE NULL
unsigned short digital[DIGCNT];  /* Digital channel array */
float analog[ANACNT];  /* Analog channel array */
signed int exitflag=FALSE;  /* exit flag */

/* initial - sets up the hardware, reads calibration settings and set
 * the channels */

int initialize(void)
{
  register int i;

  /* Read CALIB.DAT file and calibrate the analog inputs */
  SHCALL("Fn", analog, digital);  /* Call A-D driver */

  if (digital[0]==0 && digital[2]==0) {
    printf("\nERROR: Driver, ADRIVE.COM, not installed; ");
    printf("or analog card not installed.\n");
    return(1);
  }

  if (digital[0]==0 && digital[2]!=0) {
    printf("\nERROR: No analog card selected. BRD SEL switch set to 0.\n");
    return(1);
  }

  if (digital[0]!=0 && digital[6]==0) {
    printf("\nERROR: CALIB.DAT file not correct or FIND.EXE was not run.\n");
    return(1);
  }

  if (digital[0] > digital[6]) {
    printf("\nERROR: Calibration numbers are not correct.\n");
    return(1);
  }

  if (digital[0]>ANACNT || digital[2]>DIGCNT) {
    printf("\nERROR: Too many channels installed.\n");
    printf("Change size of ANACNT and/or DIGCNT in heading\n");
    return(1);
  }

  /* set number of analog input channels to use */
  digital[0]=1;  /* use one analog channel */
  SHCALL("N", analog, digital);

  /* Set analog channel resolution */
  digital[0]=16;  /* 16 bit mode */
}
/ * Set up analog channel range and calibrate */
digital[0]=2; /* 10 volts */
SH_CALL("rc",analog,digital);

/ * Set up digital channel direction */
for (i=0; i<DIGCNT; i++) digital[i]=0; /* set for input */
SH_CALL("S",analog,digital);

/ * Set delay time to zero */
digital[0]=0;
SH_CALL("d",analog,digital); /* set delay value */

/ *
* Usage - prints out switches and tells their usage
*/

void Usage(char **argv)
{
    printf("\nUsage: %s [switches]\n\n",argv[0]);
    printf(" -a[filename] write 02 average to file\n\n");
    printf(" -d[filename] write all data to file (.xxx added)\n\n");
    printf(" -h[number] sampling rate in hertz (default=173)\n\n");
    printf(" -n[number] number of sets to collect\n\n");
    printf(" -t default: read until user interrupts\n\n");
    printf(" -w print time taken to read oxygen voltages\n\n");
    printf(" -z[number] scans per sample (default=8)\n\n");
}

/ *
* mkdfname - adds an extension to the oxygen voltage reading file
* based on the number of the SET that we are on
*
* name=name of the file (without extension)
* dfnum=SET number we are on
*/

void mkdfname(char *name, int dfnum)
{
    register int i;
    char ext[4];
    sprintf(&ext[0], "%03d", dfnum);
    for (i=0; i<13; i++) {
        if (name[i]=='.') {
            strcpy(&name[i+1],&ext[0]);
            break;
        } else if (name[i]==0) {
            name[i]='.';
            strcpy(&name[i+1],&ext[0]);
            break;
        }
    }
}

/ *
* chandler - function called when Control-C is pressed. Sets the exit flag to TRUE.
*
int chandler(void)
{
    signal(SIGINT, SIG_IGN); /* disallow Ctr-C during handler */
    exitflag=TRUE;
    signal(SIGINT, chandler); /* reattach handler to CTR-C */
    return 0;
}

int main(int argc, char **argv)
{
    register int i; /* general purpose loop counter */
    int pdata=FALSE; /* print all voltage readings to screen */
    int petime=FALSE; /* print time taken to read oxygen voltages */
    int scansps=8; /* scans per sample */
    int setlim=-1; /* number of sets */
    int dfon=FALSE; /* write all voltage readings to a file */
    int hertz=173; /* hertz */
    int samplecount; /* number of voltage readings to make avr */
    int dfnum; /* number of SETS of voltage readings taken */
    FILE *datafile=NULL,*avrfile=NULL; /* file handles */
    char dfname[50],afname[50]; /* file names */
    float sum,avr,buf; /* variables for computing the avr V */
    struct dostime t timel,tizre2; /* variables used for elapsed time */
    signed int s,h;

    /* set CTR-C handler so we can exit properly */
    if (signal(SIGINT, chandler)==(int(*)(int))1) {
        printf("ERROR: Unable to set CTR-C handler");
        return(1);
    }

    /* do hardware initialization and make sure driver is installed */
    if (initializeo) return(1);

    /* process switches */
    for (i=1; i<argc; i++) {
        if (!(argv[i][0]=='-')) {
            printf("ERROR: Switch error\n");
            Usage(argv);
            goto end;
        } else {
            switch(tolower(argv[i][1])) {
                case 'a':
                    if (argv[i][2]==0) /* default name */
                        strcpy(&afname[0], "O2AVR.DAT");
                    else /* use user-defined name */
                        strcpy(&afname[0], &argv[i][2]);
                    avrfile=fopen(&afname[0], "w");
                    if (avrfile) {
                        printf("ERROR: Unable to open file \%s\n", &afname[0]);
                        goto end;
                    }
                    break;
                case 'd':
                    if (argv[i][2]==0) /* default name */
                        strcpy(&dfname[0], "O2VDAT");
                    else /* use user-defined name */
                        strcpy(&dfname[0], &argv[i][2]);
                    dfnum=fopen(&dfname[0], "w");
                    if (dfnum) {
                        printf("ERROR: Unable to open file \%s\n", &dfname[0]);
                        goto end;
                    }
                    break;
            }
        }
    }

end:
}
break;
  
case 'h':
    hertz=atoi(&argv[i][2]);
    break;

  case 'n':
    setlim=atoi(&argv[i][2]);
    break;

  case 't':
    petime=TRUE;
    break;

  case 'w':
    pdata=TRUE;
    break;

  case 'z':
    scansps=atoi(&argv[i][2]);
    break;

  case '?':
    usage(argv);
    goto end;

default:
  printf("ERROR: Unknown Switch 'c', argv[i][1]\n");
  usage(argv);
  goto end;

  }

  /* compute number of readings to take */
  samplecount=scansps*O.2*hertz;

  /* allocate memory to read voltage data into */
  buf=(float *)malloc(samplecount*sizeof(float));
  if (!buf) {
    printf("ERROR: Not enough memory\n");
    goto end;
  }

  /* print out information for user */
  printf("Hertz: %d\n",hertz);
  printf("Scans Per Sample: %d\n",scansps);
  printf("Number of Voltage Readings to take: %d\n",samplecount);
  if (avrfile) printf("Writing Averages to file: %s\n",&afname[0]);
  if (dfon) printf("Writing Voltage Readings to files: %s.xxx\n",&dfname[0]);

  if (setlim==1) {
    printf("Hit Control-C to Stop.\n");
  } else {
    printf("Number of Sets to read: %d\n",setlim);
    printf("Hit Control-C to Abort.\n");
  }

  printf("Waiting for TTL Signal...\n");
  for (dfnum=1; (setlim==1)?(1):(dfnum<=setlim); dfnum++) {
    if (exitflag) { printf("Terminating...\n"); goto end; }
    if (dfon) {
      mkdfname(&dfname[0],dfnum);
      datafile=fopen(&dfname[0],"w");
      if (!datafile) {
        printf("ERROR: Unable to open file %s\n",&dfname[0]);
        goto end;
      }
    }
  }

  do { /* wait for TTL signal */
    if (exitflag) { printf("Terminating...\n"); goto end; }
while (digital[0] == 1);  
if (exitflag) { printf("Terminating...\n"); goto end; }

/* collect O2 voltages */
digital[0] = samplecount;
digital[1] = 0;
if (petime) {
  _dos_gettime(&time1);
  SH_CALL("M", buf, digital);
  _dos_gettime(&time2);
  s = (int) time2.second - time1.second;
  h = (int) time2.hsecond - time1.hsecond;
  if (h<0) { h=100+h; s--; }
  printf("Elapsed Time = %d.%d seconds\n", s, h);
} else {
  SH_CALL("M", buf, digital);
}
if (exitflag) { printf("Terminating...\n"); goto end; }

/* compute average average oxygen voltage */
if (pdata) printf("Oxygen Voltage Data: #%d\n", dfnum);
for (i=0, sum=(float)0; i<samplecount; i++) {
  if (pdata) printf("%20e\n", buf[i]);
  if (datafile) fprintf(datafile, "%20e\n", buf[i]);
  sum = sum + buf[i];
}
avr = sum/samplecount;
printf("Average Oxygen Voltage for %04d readings (%04d): %20e\n", samplecount, dfnum, avr);
if (avrfile) fprintf(avrfile, "%20e\n", avr);

/* close data file for oxygen voltage readings */
if (datafile) {
  fclose(datafile);
  datafile = NULL;
}
end:
if (buf) free(buf);
if (datafile) fclose(datafile);
if (avrfile) fclose(avrfile);

return(0);

/* end of file - 'o2volt.c' */
Appendix B

Version 2 of the oxygen voltage concentration recorder.
BASE=o2volt
OBJ=obj
SOURCE=source
LST=listings
ASMOPT=/ml /z /n /c /l /Ik: \o2volt\source
COPT=/AS /FPi87 /c /Fo$(OBJ)\fmath.obj /Fa$(OBJ)\fmath.asm /W3 /Gs \
/Fs$(LST)\fmath.lst
FILES=$(OBJ)\main.obj $(OBJ)\hard.obj $(OBJ)\util.obj $(OBJ)\fmath.obj \\
$(OBJ)\help.obj

$(BASE).exe: $(FILES)
   link /NOI $(FILES),$(BASE).exe;

$(OBJ)\fmath.obj: $(SOURCE)\fmath.c
   cl $(COPT) $(SOURCE)\fmath.c

$(OBJ)\main.obj: $(SOURCE)\main.asm $(SOURCE)\header.i
   tasm $(ASMOPT) $(SOURCE)\main.asm, $(OBJ)\main.obj,$(LST)\main.lst

$(OBJ)\hard.obj: $(SOURCE)\hard.asm $(SOURCE)\header.i
   tasm $(ASMOPT) $(SOURCE)\hard.asm, $(OBJ)\hard.obj,$(LST)\hard.lst

$(OBJ)\util.obj: $(SOURCE)\util.asm $(SOURCE)\header.i
   tasm $(ASMOPT) $(SOURCE)\util.asm, $(OBJ)\util.obj,$(LST)\util.lst

$(OBJ)\help.obj: $(SOURCE)\help.asm $(SOURCE)\header.i
   tasm $(ASMOPT) $(SOURCE)\help.asm, $(OBJ)\help.obj,$(LST)\help.lst

$(SOURCE)\header.i:

## end of file 'o2volt.lmk'
callsys macro interrupt, service
  mov ah, service
  int interrupt .
endm

FNAME_LEN equ 127
NUMBUF_LEN equ 127
DACQ_ENTRIES equ 3000
AVRFLIDTH equ 23
DATFLIDTH equ 25
MIDDLE_LINE equ 13
DISPLAY_PAGE equ 1h
DISPLAY_HEIGHT equ 25
DISPLAY_WIDTH equ 80

DISPLAY_NORMAL equ 00010111b
DISPLAY_HILIGHT equ 00101111b

; end of file 'header.i'
dosseg
locals
include header.i

.model small

.data

--- Program: o2volt
--- File: help.asm
--- Author: David Dahle (summer apprentice)
--- Date: July-August 1992
--- Purpose: help text and functions for oxygen voltage recorder
--- Letterman Army Institute of Research
--- Persidio of San Francisco, CA 94129-6800

dosseg
locals
include header.i

.model small

.data

;-------------------123456789012345678901234567890123-------------------
begin db 'The Begin function is used to',0
   db 'start data acquisition. Once',0
   db 'started, acquisition may be',0
   db 'terminated by pressing any key.',0
   db 0
instant db 'The Instant function displays',0
   db 'the current analog voltage level',0
   db 'and the current state of the',0
   db 'TTL signal. This function can',0
   db 'be terminated by pressing any',0
   db 'key.',0
   db 0
load db 'The Load function allows a',0
   db 'config file with a name other',0
   db 'than the default name to be',0
   db 'loaded. A string input field',0
   db 'will appear below the command',0
   db 'list where the name can be',0
   db 'entered. ESCAPE aborts',0
   db 0
save db 'The Save function allows a',0
   db 'config file to be saved to a',0
   db 'file with a name other than the',0
   db 'default name. A string',0
   db 'will appear below the command',0
   db 'bar where the new name can be',0
   db 'entered. ESCAPE aborts',0
   db 0
exit db 'The Exit function writes',0
   db 'the configuration settings',0
   db 'to the default config file',0
   db 'and then exits.',0
   db 0
avrfn db 'The Average File field',0
   db 'specifies the name of the',0
   db 'file in which the averages',0
   db 'from each set will be written.',0
   db 'If no name is given, then no',0
   db 'file will be created.',0
   db 0
datfn db 'The Data Files field',0
   db 'specifies the names of files',0
'in which each reading from',
'each set will be written.',
'If no name is given, then no',
'files will be created.',

'files will be written.',

The Experiment Number field’,
'specifies the number of the',
'current experiment. If the',
'Automatic File Names option',
'is turned off, this number is',
'ignored.',

The Automatic File Names',
'field lets you turn this',
'feature on or off. If on, the',
'program will create file names',
'based on the current date and',
'the experiment number.',

The Print Time field lets you’,
'specify if you want the elapsed',
'time that the TTL pulse was',
'active and the data rate in',
'hertz to be computed.',

The Analog Input Range field’,
'lets you specify the projected',
'range of the analog voltage',
'signal. These are listed on’,
'page 185 in the manual and are',
'used by the "r" command.',

The Analog Input Resolution’,
'field lets you specify the',
'resolution of the readings.',
'These are listed on page 148’,
in the manual and are used by’,
the "a" command.',

...
je   short __continue
__strloop:
   mov   al, [si]
   inc   si
   cmp   al, 0
   jne   short __strloop
__continue:
   inc   dh
   loop   short __loop
__endOfText:
   ret

; --- Procedure: displayCmdHelp
; --- Input: al=[command]
; --- Output:

   public displayCmdHelp
displayCmdHelp:
   cmp   al, 0
   jne   __not1
   call  displayBeginHelp
   jmp   __end
__not1:
   cmp   al, 1
   jne   __not2
   call  displayInstantHelp
   jmp   __end
__not2:
   cmp   al, 2
   jne   __not3
   call  displayLoadHelp
   jmp   __end
__not3:
   cmp   al, 3
   jne   __not4
   call  displaySaveHelp
   jmp   __end
__not4:
   cmp   al, 4
   jne   __not5
   call  displayExitHelp
   jmp   __end
__not5:
   __end:   ret

   public displayBeginHelp
displayBeginHelp:
   lea   si, begin
   call  displayHelpText
   ret
   public displayInstantHelp
displayInstantHelp:
   lea   si, instant
   call  displayHelpText
   ret
   public displayLoadHelp
displayLoadHelp:
   lea   si, load
   call  displayHelpText
   ret
   public displaySaveHelp
displaySaveHelp:
   lea   si, save
   call  displayHelpText
PUBLIC displayExitHelp
    displayExitHelp:
        lea    si,exit
        call  displayHelpText
        ret
PUBLIC displayAvrfnHelp
    displayAvrfnHelp:
        lea    si,avrfn
        call  displayHelpText
        ret
PUBLIC displayDatfnHelp
    displayDatfnHelp:
        lea    si,datfn
        call  displayHelpText
        ret
PUBLIC displayExpnumHelp
    displayExpnumHelp:
        lea    si,expnum
        call  displayHelpText
        ret
PUBLIC displayAutofnsHelp
    displayAutofnsHelp:
        lea    si,autofns
        call  displayHelpText
        ret
PUBLIC displayPrttHelp
    displayPrttHelp:
        lea    si,prtt
        call  displayHelpText
        ret
PUBLIC displayRangeHelp
    displayRangeHelp:
        lea    si,range
        call  displayHelpText
        ret
PUBLIC displayResolHelp
    displayResolHelp:
        lea    si,rsol
        call  displayHelpText
        ret

    END
    . END OF FILE 'HELP ASM'
dosseg
locals
include header.i

.model small

.data
; display text
ptitle db 'Data Acquisition Oxygen Voltage Recorder ',0
pline db ' ',0
pbttop db 'Program',0
pbttop db 'Dialog',0
pbmid db 'Program',0
pbmid db 'Dialog',0
pbbot db 'Help',0
pcbot db 'Help',0
avrft db 'Average File: ',0
datfl db 'Data Files: ',0
autnu db 'Experiment Number: ',0
autfn db 'Automatic File Naming: ',0
etime db 'Print Elapsed Time: ',0
range db 'Analog Input Range: ',0
resol db 'Analog Input Resolution: ',0
etime db 'ON ',0
etime db 'OFF',0
range db '50 mV ',0
range db '500 mV ',0
range db '10 V ',0
range db '+/− 25 mV ',0
range db '+/− 250 mV ',0
range db '+/− 5 V ',0
range db '2 mA ',0
range db '20 mA ',0
range db '+/− 1 mA ',0
range db '+/− 10 mA ',0
range db '+/− 50 mA ',0
range db 'AR / w univ ',0
range db 'AR / w b1V ',0
range db 'AR / w b1C ',0
resol db 'low noise',0
resol db '16 bits ',0
resol db '15 bits ',0
resol db '14 bits ',0
resol db '13 bits ',0
resol db '12 bits ',0
numcom db 4
com0 db 'Begin ',0
com1 db 'Instant ',0

; these must be spaces here!!
; startup message
.startup db 'Data Acquisition Program - Oxygen Voltage Recorder',0
    db 'Strawberry Tree Incorporated Analog-to-Digital Converter',0
    db 'Letterman Army Institute of Research',0
    db 'Presidio of San Francisco, CA 94129-6800',0
.blank1 db ',0
    db 'Author: David Dahle',0
    db 'Version: 2.1.0 Assembled: ',0
    db '0
    db 'Press any key to exit...',0
    db 'Setup completed successfully...',0

; program variables
.curdate dw 2 DUP (0)
.avrf1 db FNAMLEN DUP (0)
.datfl db FNAMLEN DUP (0)
.ne 'buf db NUMBUF_LEN DUP (0)
.etime dw 1 ; 1=ON, 2=OFF
.range dw 5 ; range of analog input
.resol dw 2 ; resolution of analog input
.autofn dw 2 ; 1=ON 2=OFF
.autonum dw 1 ; number for automatic increment
.config_defaultName db 'o2volt.cnf',0

; program status
.mode db 0 ; 0=com, 1=config
.comands db 0 ; 0=begin, i=instant, 2=hardware info, 3=exit
.com_save db 0 ; saved copy of command
.config db 1 ; config line 1-7

.global resol:byte, avrf1:byte, datfl:byte, range:word, etime:byte
.global startup:byte, curdate:word, autonum:word, blank1:byte
.global autofn:word

.code

; labels that appear in 'util.asm'
extrn printTextToScreen:near
extrn setUpStringGadget:near, setDisplayAndCursorPos:near
extrn removeStringGadget:near, addStringGadgetChar:near
extrn delStringGadgetChar:near, backStringGadgetChar:near
extrn leftStringGadgetChar:near, rightStringGadgetChar:near
extrn printCursorString:near, printString:near
extrn printCursorStringAttrib:near, printStringAttrib:near
extrn printCursorStringLimit:near, printStringLimit:near
extrn convertltoA:near, convertLtoA:near, convertAtoI:near
extrn loadConfigurationFile:near, saveConfigurationFile:near
extrn getConfigFileName:near, clearKeyBoardBuffer:near

; labels that appear in 'hard.asm'
extrn setUpHardware:near, instant Readings:near
extrn initializeHardware:near, beginDataAcquisition:near

; labels that appear in 'help.asm'
extrn displayResolHelp:near, displayRangeHelp:near
extrn displayPrntHelp:near, displayAutofnHelp:near
extrn displayExpnumHelp:near, displayDatflnHelp:near
extrn displayAvrflHelp:near, displayCmdHelp:near

... procedure main
--- output:  
--- Preserves:  

_called by C-Language startup code  

```

public _main

_main:
; get current date
    callsys 21h,2ah ; get date
    mov [curdate], cx ; year
    mov [curdate+2], dx ; month,day

; setup screen display
    call initDisplay ; color, borders
    call print_allConfig ; configuration
    call print_startupMessage ; dialog window

; setup hardware
    call setUpHardware
    cmp ax, 0
    je short __noHardwareErrors

; error with A/D board, exit program
    mov si, dx
    call printTextToScreen ; print error text
    call print_cmdLineBusy
    callsys 16h,0
    jmp short __exitProgram

__noHardwareErrors:
    call initializeHardware
    call print_cmdLine
    mov si, offset hard_sc
    call printTextToScreen

__main:
    mov al, [command]
    call displayCmdHelp
    lea dx, config_defaultName
    call loadConfigurationFile
    call mainLoop
    lea dx, config_defaultName
    call saveConfigurationFile

__exitProgram:
    call clearKeyboardBuffer
    call resetDisplay
    mov ax, 0 ; set a return code
    ret

--- Procedure: print startupMessage  
--- Input: none  
--- Output: none  
--- Preserves:  

print startupMessage:
    mov si, OFFSET startup_message
    push si
    call printTextToScreen
```
; ___ Procedure: mainLoop
; ___ Input: none
; ___ Output: none

mainLoop:
callsys 16h, 0        ; wait for next character
 cmp al, 0
 jne short _extended
 cmp al, 13
 jne short _notReturn
 cmp [mode], 0
 jne short _notCommRet
 mov al, [command]
 cmp al, 0
 jne short _notComm0
 call beginDataAcquisition
 jmp mainLoop

_notComm0:
 cmp al, 1
 jne short _notComm1
 call instant_Readings
 jmp mainLoop

_notComm1:
 cmp al, 2
 jne short _notComm2
 call getConfigFileName
 cmp ax, 0
 je short _comm2Exit
 call loadConfigurationFile
 jmp mainLoop

_comm2Exit:
 jmp mainLoop

_notComm2:
 cmp al, 3
 jne short _notComm3
 call getConfigFileName
 cmp ax, 0
 je short _comm3Exit
 call saveConfigurationFile
 jmp mainLoop

_comm3Exit:
 jmp mainLoop

_notComm3:
 jmp short _exit

_notCommRet:
call handleConfigGadgetReturn
 jmp mainLoop

_notReturn:
cmp al, 8
 jne short _return

cmp    [mode], 4
jne    short ___notBackConfigGadget
cmp    [config], 4
jg     short ___notBackStringGadget
call   backStringGadgetChar
___notBackStringGadget:
___notBackConfigGadget:
jmp     mainLoop

___notBackspace:
cmp    al, 32           ; a printable character?
jl     short ___notPrintChar
cmp    al, 126
jg     short ___notPrintChar
call   addStringGadgetChar
jmp     mainLoop

___notPrintChar:
jmp     mainLoop

___exit:    ret

___extended:
cmp    ah, 75              ; check for Cursor Left
jne    short ___notLeft
cmp    byte ptr [mode], 0
jne    short ___notLeftMode0
mov     al, [command]
cmp    al, 0
jne    short ___notUnder
mov     al, [numcom]
jmp     short ___under
___notUnder:
dec     al
___under:
mov     [command], al
call   print_cmdLine
mov     al, [command]
call   displayCmdHelp
jmp     mainLoop
___notLeftMode0:
cmp    [config], 4
jg     short ___notLeftStringGadget
call   leftStringGadgetChar
___notLeftStringGadget:
jmp     mainLoop

___notLeft:
cmp    ah, 77              ; check for Cursor Right
jne    short ___notRight
cmp    byte ptr [mode], 0
jne    short ___notRightMode0
mov     al, [command]
inc     al
mov     al, [numcom]
jle    short ___notOver
mov     al, 0
___notOver:
mov     [command], al
call   print_cmdLine
mov     al, [command]
call   displayCmdHelp
jmp     mainLoop
___notRightMode0
cmp    [config], 4
call rightStringGadgetSetUp
__notRightStringGadget:
    jmp mainLoop
__notRight:
    cmp ah, 72
    jne short __notUp
    cmp [mode], 0
    jne short __notCommUp
    mov al, [command]
    mov [com_save], al
    mov [command], -1
    mov [mode], 1
    mov [config], 7
    ; bottom editing gadget
    call print_cmdLine
    call handleConfigGadgetSetUp
    jmp mainLoop
__notCommUp:
    cmp [config], 1
    jne short __notAtTheTop
    call handleConfigGadgetRemoval
    mov byte ptr [mode], 0
    mov al, [com_save]
    mov [command], al
    call print_cmdLine
    mov al, [command]
    call displayCmdHelp
    jmp mainLoop
__notAtTheTop:
    call handleConfigGadgetRemoval
    dec [config]
    call handleConfigGadgetSetUp
    jmp mainLoop
__notUp:
    cmp ah, 80
    jne short __notDown
    cmp [mode], 0
    jne short __notCommDown
    mov al, [command]
    mov [com_save], al
    mov [command], -1
    mov [mode], 1
    mov [config], 1
    ; bottom editing gadget
    call print_cmdLine
    call handleConfigGadgetSetUp
    jmp mainLoop
__notCommDown:
    cmp [config], 7
    jne short __notAtTheBottom
    call handleConfigGadgetRemoval
    mov [mode], 0
    mov al, [com_save]
    mov [command], al
    call print_cmdLine
    mov al, [command]
    call displayCmdHelp
    jmp mainLoop
__notAtTheBottom:
    call handleConfigGadgetRemoval
    inc [config]
    call handleConfigGadgetSetUp
    jmp mainLoop
__notDown:
; --- Procedure: handleConfigGadgetSetUp
; --- Input: none
; --- Output: none
; --- Preserves:

handleConfigGadgetSetUp:
  mov al, [config]
  cmp al, 1
  jne short _notConfig1
  call displayAvrfnHelp
  mov al, 0
  mov bx, offset avrfl
  mov cl, FNAME_LEN
  mov ch, AVRFL_DWIDTH
  mov dl, 17
  mov dh, 3
  call setUpStringGadget
  jmp __exit

__notConfig1:
  cmp al, 2
  jne short _notConfig2
  call displayDatfnHelp
  mov al, 0
  mov bx, offset datfl
  mov cl, FNAME_LEN
  mov ch, DATFL_DWIDTH
  mov dl, 15
  mov dh, 4
  call setUpStringGadget
  jmp __exit

__notConfig2:
  cmp al, 3
  jne short _notConfig3
  call displayExpnumHelp
  mov ax, [autonum]
  call convertIoA
  mov si, offset nedbuf
  mov al, [bx]
  inc bx
  mov [si], al
  inc si
  cmp al, 0
  jne __config3Loop
  mov al, 1
  mov bx, offset nedbuf

__config3Loop:
mov ch, 0
mov dl, 22
mov dh, 5
call setUpStringGadget
jmp __exit

__notConfig3:
cmp al, 4
jne short __notConfig4
call displayAutofnHelp
mov bl, DISPLAY_HILIGHT
call print Autofn
jmp __exit

__notConfig4:
cmp al, 5
jne short __notConfig5
call displayPrttHelp
mov bl, DISPLAY_HILIGHT
call print_etime
jmp __exit

__notConfig5:
cmp al, 6
jne short __notConfig6
call displayRangeHelp
mov bl, DISPLAY_HILIGHT
call print_range
jmp __exit

__notConfig6:
cmp al, 7
jne short __notConfig7
call displayResolHelp
mov bl, DISPLAY_HILIGHT
call print_resol

__notConfig7:
__exit:
ret

Procedure: handleContigGadgetRemoval
-- Input: none
-- Output: none

handleContigGadgetRemoval:
mov al, [config]
cmp al, 1
jne short __notConfig1
call removeStringGadget
jmp short __exit

__notConfig1:
cmp al, 2
jne short __notConfig2
call removeStringGadget
jmp short __exit

__notConfig2:
cmp al, 3
jne short __notConfig3

__notConfig3:
cmp al, 4
jne short __notConfig4
call displayAutofnHelp
mov bl, DISPLAY_HILIGHT
call print Autofn
jmp __exit

__notConfig4:
cmp al, 5
jne short __notConfig5
call displayPrttHelp
mov bl, DISPLAY_HILIGHT
call print_etime
jmp __exit

__notConfig5:
cmp al, 6
jne short __notConfig6
call displayRangeHelp
mov bl, DISPLAY_HILIGHT
call print_range
jmp __exit

__notConfig6:
cmp al, 7
jne short __notConfig7
call displayResolHelp
mov bl, DISPLAY_HILIGHT
call print_resol

__notConfig7:
__exit:
ret
mov $1,01leq_number
call convertAtoI
mov [autonum],ax
call print_autonum
jmp short __exit

__notConfig3:
cmp al,4
jne short __notConfig4
mov bl,DISPLAY_NORMAL
call print_autofn
jmp short __exit

__notConfig4:
cmp al,5
jne short __notConfig5
mov bl,DISPLAY_NORMAL
call print_etime
jmp short __exit

__notConfig5:
cmp al,6
jne short __notConfig6
mov bl,DISPLAY_NORMAL
call print_range
jmp short __exit

__notConfig6:
cmp al,7
jne short __notConfig7
mov bl,DISPLAY_NORMAL
call print_resoI
 jmp short __exit

__notConfig7:
__exit:
ret

; --- Procedure: handleConfigGadgetReturn
; --- Input: none
; --- Output: none

handleConfigGadgetReturn:
 mov al,[config]

__notConfig3:
cmp al,4
jne short __notConfig4
mov bl,DISPLAY_NORMAL
call print_autofn
mov ax,[autofn]
cmp ax,2
jne short __notAutofn1
mov ax,1
jmp short __notAutofn2

__notAutofn1:
inc ax
__notAutofn2:
mov [autofn],ax
mov bl,DISPLAY_HILIGHT
call print_autofn
_notConfig4:
    cmp al, 5
    jne short _notConfig5
    mov bl, DISPLAY_NORMAL
    call print_etime
    mov ax, [etime]
    cmp ax, 2
    jne short _notEtime1
    mov ax, 1
    jmp short _notEtime2

_notEtime1:
    inc ax

_notEtime2:
    mov [etime], ax
    mov bl, DISPLAY_HILIGHT
    call print_etime
    jmp short _exit

_notConfig5:
    cmp al, 6
    jne short _notConfig6
    mov bl, DISPLAY_NORMAL
    call print_range
    mov ax, [range]
    cmp ax, 10
    jg short _notC6Less16
    cmp ax, 10
    jne short _notC6LoopAround
    mov ax, 16
    jmp short _config6End

_notC6Less16:
    cmp ax, 19
    jne short _notC6LoopAround
    mov ax, 0
    jmp short _config6End

_notC6LoopAround:
    inc ax

_config6End:
    mov [range], ax
    mov bl, DISPLAY_HILIGHT
    call print_range
    jmp short _exit

_notConfig6:
    cmp al, 7
    jne short _notConfig7
    mov bl, DISPLAY_NORMAL
    call print_resol
    mov ax, [resol]
    cmp ax, 6
    jne short _notResol1
    mov ax, 1
    jmp short _notResol2

_notResol1:
    inc ax

_notResol2:
    mov [resol], ax
    mov bl, DISPLAY_HILIGHT
    call print_resol
    jmp short _exit
notConfig:
_exit:
   ret

; --- Procedure: print_cmdLineBusy
; --- Input:
; --- Output:
; --- Preserves:

public print_cmdLineBusy
print_cmdLineBusy:
   mov    bh,DISPLAY_PAGE
   mov    dl,3
   mov    dh,11
   callsys 10h,2
   mov    ah,9
   mov    al,''
   mov    bh,DISPLAY_PAGE
   mov    bl,DISPLAY_NORMAL
   mov    cx,40
   callsys 10h,9          ; BIOS write character
   lea    si,hard_er
   mov    dl,3
   mov    dh,11
   call   printString
   ret

; --- Procedure: print_cmdLine
; --- Input: none
; --- Output: none
; --- Preserves: ds,es,si,di

public print_cmdLine
print_cmdLine:
   mov    bh,DISPLAY_PAGE
   mov    dl,3
   mov    dh,11
   callsys 10h,2          ; set cursor position
   mov    si,offset comm_0
   mov    cx,0
   __loop: push cx
   cmp    [command],cl
   je     short __jmp0
   mov    bl,DISPLAY_NORMAL
   jmp    short __jmp1
__jmp0:   mov    bl,DISPLAY_HIGHLIGHT
__jmp1:   call   printCursorString
   pop    cx
   inc    cx
   cmp    [numcom],cl
   jge    __loop
   ret

; --- Procedure: print null

public  print_datfl, print_avrfl, print_autonum, print_allConfig

.print_allConfig:
call  print_avrfl
call  print_datfl
call  print_autonum
mov   bl, DISPLAY_NORMAL
call  print_autoIn
mov   bl, DISPLAY_NORMAL
call  print_etimc
mov   bl, DISPLAY_NORMAL
call  print_range
mov   bl, DISPLAY_NORMAL
call  print_resol
ret

print_avrfl:
  mov   si, offset avrfl_t
  mov   dh, 3
  mov   dl, 3
  call  printStrfng
  mov   si, offset avrfl
  mov   cl, AVRFL_WIDTH
  call  printCursorStringLimit
  ret

print_datfl:
  mov   si, offset datfl_t
  mov   dh, 4
  mov   dl, 3
  call  printStrfng
  mov   si, offset datfl
  mov   cl, DATFL_WIDTH
  call  printCursorStringLimit
  ret

print_autonum:
  mov   si, offset autonum_t
  mov   dh, 5
  mov   dl, 3
  call  printStrfng
  mov   ax, [autonum]
  call  convertloA
  mov   si, bx
  mov   cl, 4 ; max length of string
  call  printCursorStringLimit
  ret

print_autoIn:
  mov   si, offset autfn_t
  mov   dh, 6
  mov   dl, 3
  call  printStrfng
  mov   ax, [autofn]
  cmp   al, 1
  jne   short __jmpl
  mov   si, offset etime_1
  jmp   short __jmp2
__jmpl:  mov   si, offset etime_2
__jmp2:  call  printCursorStringAttrib
  ret

print_etimc:
  mov   si, offset etimc_t
  mov   dh, 7
  mov   dl, 3
  call  printStrfng
  ret
cmp al, 1
jne short __jmp1
mov si, offset etime_1
jmp short __jmp2
__jmp1: mov si, offset etime_2
__jmp2: call printCursorStringAttrib
ret

print_range:
  mov si, offset range_t
  mov dh, 8
  mov dl, 3
  call printString
  mov ax, [range]
  cmp ax, 10
  jle __notOver
  sub ax, 5
__notOver:
  mov cl, 11
  mul cl
  lea si, range_0
  add si, ax
  call printCursorStringAttrib
  ret

print_resol:
  mov si, offset resol_t
  mov dh, 9
  mov dl, 3
  call printString
  mov ax, [resol]
  cmp al, 1
  jne short __jmp1
  mov si, offset resol_1
  jmp short __jmpf
__jmp1:
  cmp al, 2
  jne short __jmp2
  mov si, offset resol_2
  jmp short __jmpf
__jmp2:
  cmp al, 3
  jne short __jmp3
  mov si, offset resol_3
  jmp short __jmpf
__jmp3:
  cmp al, 4
  jne short __jmp4
  mov si, offset resol_4
  jmp short __jmpf
__jmp4:
  cmp al, 5
  jne short __jmp5
  mov si, offset resol_5
  jmp short __jmpf
__jmp5:
  mov si, offset resol_6
__jmpf:
  call printCursorStringAttrib
  ret

; --- Procedure: initDisplay
; --- Input: none
; --- Output: none
; --- Preserves: ds, es ...?
;
initDisplay:
; make page the current display page
  mov al, DISPLAY_PAGE

; set screen colors
        mov    bh, DISPLAY_PAGE
        mov    dx, 0
        callsys 10h, 2          ; Set Cursor Position (BIOS)
        mov    ah, 9
        mov    al, ' 
        mov    bh, DISPLAY_PAGE
        mov    bl, DISPLAY_NORMAL
        mov    cx, DISPLAY_WIDTH*DISPLAY_HEIGHT
        callsys 10h, 9          ; BIOS write char - color screen

        mov    bh, DISPLAY_PAGE
        mov    dx, 0
        callsys 10h, 2          ; Set Cursor Position (BIOS)
        mov    ah, 9
        mov    al, ' 
        mov    bh, DISPLAY_PAGE
        mov    bl, DISPLAY_HILIGHT
        mov    cx, DISPLAY_WIDTH
        callsys 10h, 9          ; BIOS write char - color top line

; set up title bar text
        mov    si, offset ptitle
        mov    dx, 0            ; cursor position
        call    printString

; set up screen text borders
        mov    si, offset pbtop
        mov    dl, 0            ; dl, dh (x, y)
        mov    dh, 1
        call    printString

        mov    cx, MIDDLE_LINE-2
        mov    dh, 2

    _loop1:
        mov    si, offset pline
        mov    dl, 0
        call    printString
        mov    si, offset pline
        mov    dl, DISPLAY_WIDTH-1
        call    printString
        inc    dh
        loop    _loop1

        mov    si, offset pbmid
        mov    dl, 0
        call    printString

        mov    cx, (DISPLAY_HEIGHT-2) - MIDDLE_LINE

    _loop2:
        mov    si, offset pline
        mov    dl, 0
        inc    dh
        call    printString
        mov    si, offset pline
        mov    dl, DISPLAY_WIDTH-1
        call    printString
        loop    _loop2

        mov    si, offset pbbot
        mov    dl, 0
        inc    dh
        call    printString
mov al, 43
mov si, offset pctop
call printString

__loop3:
  mov cx, 7
  mov si, offset pline
  mov dl, 43
  inc dh
  call printString
  mov si, offset pline
  mov dl, DISPLAY_WIDTH-3
  call printString
  loop __loop3

mov dl, 43
inc dh
mov si, offset pcbot
call printString

; make page the current display page
ret

; --- Procedure: resetDisplay
; --- Input: none
; --- Output: none
; --- Preserves: none
resetDisplay:
  mov al, 0
  callsys 10h, 5 ; BIOS Set Active Page back to zero
ret

end
; end of file 'main.asm'
dosseg
locals
include header.i

.model small

.data

; string gadget variables
  g_flags db 0 ; gadget flags 0=string 1=num
  scrn_x db 0
  scrn_y db 0 ; x,y position of gadget
  scrn_w db 0 ; onscreen width of gadget
  buf_len db 0 ; characters in buffer
  buf_off db 0 ; display offset into buffer
  buf_cur db 0 ; cursor offset into buffer
  buf_wid db 0 ; max length of buffer
  buf_seg dw 0 ; segment buffer is in
  buf_ptr dw 0 ; segment offset to buffer

; number to ascii text buffer
nconbuf db NUMBUF_LEN DUP (0)

; display window parameters
  curlin db 0 ; cursor position in status window

; dos error texts
  diskfer db 'Disk Full',0
  derrmsg db 'Invalid function number',0
  db 'File not found',0
  db 'Path not found',0
  db 'No more handles (to many files)',0
  db 'Access denied',0
  db 'Invalid handle',0
  db 'Memory control blocks destroyed',0
  db 'Not enough memory',0
  db 'Invalid memory-block address',0
  db 'Invalid environment block',0
  db 'Invalid format',0
  db 'Invalid file-access code',0
  db 'Invalid data',0
  db 'You should never see this one!!!?!@#|@',0
  db 'Invalid drive specification',0
  db 'Attempt to remove the current directory',0
  db 'Not the same device',0
  db 'No more files',0

MAX_DOSERROR_NUMBER equ 18
DOSERROR_STRING_WIDTH equ 40
CONFIG_LENGTH equ 4+(FNAME_LEN*2)+NUMBUF_LEN+(2*5)

cnfoper db 'ERROR: Unable to open configuration file.',0
cnfder db 'ERROR: DOS Error while reading configuration file.',0
cnfwter db 'ERROR: DOS Error while writing configuration file.',0
; Configuration file written.
; Configuration File Read.
(Configuration LENGTH) DUP (0)
; Name of Configuration File: ', 0

global nconbuf:byte, curdate:word, autonum:word, startup:byte
global blank1:byte

.code

; labels in file 'main.asm'
extrn print_allConfig:near

; --- Procedure: getConfigFileName
; --- Input: none
; --- Output: ax=0, don’t load file
; ---     ax!=0, load file, ds:dx=filename
; --- Preserves:

public getConfigFileName
getConfigFileName:
    mov dl,3
    mov dh,12
    lea si,cnfany1
    call printString

mainLoop:
callsys 16h,0

    cmp al,0
    jne short __extended
    cmp al,13
    jne short __notReturn
    call removeStringGadget

    mov cl,74
    mov dl,3
    movdh,12
    lea si,blank1
    call printStringLimit
    mov ax,1
    lea dx,cnfbuf
    ret

__notReturn:
cmp al,8
    jne short __notBackspace
    call backStringGadgetChar
    jmp mainLoop

__notBackspace:
cmp al,27
    jne short __notEscape
    call removeStringGadget

_
; --- Procedure: loadConfigurationFile
; --- Input: dx=name of config file to load
; --- Output:
; --- Preserves:
;
public loadConfigurationFile
loadConfigurationFile:
    mov  al,0          ; open for reading
    callsys 21h,3dh
    jnc short __noOpenError
    lea  si,cnfoper
    call  doslOError
    jmp   __exit

__noOpenError:
    push ax            ; save handle
    mov  bx,ax
    mov  cx,51         ; length of init string
    lea  dx,nconbuf    ; destination buffer
    callsys 21h,3fh
    jnc short __noFileError1
    lea  si,cnfrder    
    call  doslOError
    jmp   __closeExit

__noFileError1:
    cmp  ax,cx
798
98
lea    si, cnfrder
call   dosIOError
jmp    _closeExit

_noFileError2:
lea    di, startup
mov    si, dx

cmpLoop:
    mov    al, [si] ; string read
    inc    si
    mov    ah, [di] ; source string
    inc    di
    cmp    al, ah
    jne    short _notConfigFile
    cmp    ah, 0 ; equal at null, string ok
    jne    short _cmpLoop

    pop    bx
    push   bx
    mov    cx, CONFIG_LENGTH
    lea    dx, cnfbuf
    callsys 21h, 3fh
    jnc    short noReadError1
    lea    si, cnfrder
    call   dosIOError
    jmp    short _closeExit

_noReadError1:
    cmp    ax, cx
    je     short _noReadError2
    lea    si, cnfrder
    call   dosIOError
    jmp    short _closeExit

_noReadError2:
    lea    si, cnfbuf
    lea    di, curdate
    mov    ax, [si]
    cmp    [di], ax
    je     short _dateSame1
    jmp    short _dateNotSame

_dateSame1:
    mov    ax, [si+2]
    cmp    [di+2], ax
    jne    short _dateNotSame
    mov    ax, [autonum]
    jmp    short _allHere

_dateNotSame:
    mov    ax, -1 ; set -1, reset autonum to 1

_allHere:
    add    si, 4
    add    di, 4
    mov    dx, ds
    mov    es, dx
    mov    cx, CONFIG_LENGTH - 4
    cld

copyLoop:
    movsb
    loop   __copyLoop
    cmp    ax, -1
    jne    short _noResetAuto
    mov    [autonum], 1
; Procedure: saveConfigurationFile
; Input: dx=file name
; Output:
; Preserves:

public  saveConfigurationFile
saveConfigurationFile:
    mov    cx, 0
    callsys 21h, 3ch
    jnc    short _noOpenError
    lea    si, cnfoper
    call   dosIOError
    jmp    short _closeExit

_noOpenError:
    push   ax
    mov    bx, ax
    lea    dx, startup
    mov    cx, 51
    callsys 21h, 40h
    jnc    short _noFileWriteError1
    lea    si, cnfwter
    call   dosIOError
    jmp    short _closeExit

_noFileWriteError1:
    cmp    ax, cx
    je     short _noFileWriteError2
    mov    al, -1
    call   dosIOError
    jmp    short _closeExit

_noFileWriteError2:
    pop    bx
    push   bx
    lea    dx, curdate
    mov    cx, CONFIG_LENGTH
    callsys 21h, 40h
    jnc    short _noFileWriteError3
    lea    si, cnfwter
    call   dosIOError
    jmp    short _closeExit
```assembly
; --- Procedure: dosIOError
; --- Input: si=pointer to initial string to output
; --- al=error number returned by dos (-1=diskfull)
; --- Output:
; --- Preserves:

public dosIOError

dosIOError:
    push ax
    call printTextToScreen
    pop ax
    cmp al,-1
    jne short __noDiskFullError
    lea si,diskfer
    jmp short __printText

__noDiskFullError:
    cmp al,MAX_DOSERROR_NUMBER
    ja short __noErrorMessage ; unsigned, please
    dec al
    mov bl,DOSERROR_STRING_WIDTH
    mul bl
    lea si,derrmsg
    add si",aK

__printText:
    call printTextToScreen
    ret

__noErrorMessage:
    mov ah,0
    call convertItxA
    mov si,bx
    call printTextToScreen
    ret
```

clearKeyBoardBuffer:
callsys 16h,1
    je _noCharacter
    callsys 16h,0
    jmp short clearKeyBoardBuffer
_noCharacter:
    ret

; --- Procedure: printTextToScreen
; --- Input: ds:si=points to a null-terminated string to output
; --- Output:
; --- Preserves:

PUBLIC printTextToScreen
printTextToScreen:
    mov dl,1 ; x position to print
    mov dh, [curlin]
    add dh, MIDDLE_LINE+1
    mov cl, 78 ; width of screen we have
    call printStringLimit
    inc [curlin]
    cmp [curlin], 10
    jne short _noNeedToScroll
    dec [curlin]
    mov al, 1 ; number of lines to scroll
    mov bh, DISPLAY_NORMAL ; attribute of new blank line
    mov cl, 1
    mov ch, MIDDLE_LINE+1
    mov dl, 78
    mov dh, DISPLAY_HEIGHT-2
    callsys 10h, 6

_noNeedToScroll:
    ret

; --- Procedure: printTtsBackLine
; --- Input:
; --- Output:

public printTtsBackLine
printTtsBackLine:
    cmp [curlin], 0
    je short _atTopLine
    dec [curlin]
_atTopLine:
    ret

; --- Procedure: setUpStringGadget
; --- Inputs: al=flags
;            ds:bx=pointer to buffer of an asciiz string
;            cl=max length of buffer including null
;            ch=length of onscreen display
public setUpStringGadget

setUpStringGadget:

```
    mov    [g_flags], al     ; save parameters
    mov    [scrn_x], dl
    mov    [scrn_y], dh
    mov    [scrn_w], ch
    mov    [buf_wid], cl
    mov    [buf_seg], ds
    mov    [buf_ptr], bx

    mov    cl, 0             ; get the number of chars in this buffer

    _loop1:
    mov    al, [bx]
    inc    bx
    inc    cl
    cmp    al, 0
    jne    _loop1
    dec    cl
    mov    [buf_len], cl
    mov    [buf_off], 0
    mov    [buf_cur], 0

    call   setDisplayAndCursorPos
    ret
```


```

public setDisplayAndCursorPos

setDisplayAndCursorPos:

```

```
    push   ds
    mov    si, [buf_ptr]     ; buffer offset into segment
    mov    ds, [buf_seg]     ; buffer segment
    mov    al, [buf_off]
    mov    ah, 0
    add    si, ax
    mov    cl, [scrn_w]      ; gadget width
    mov    dl, [scrn_x]
    mov    dh, [scrn_y]
    call   printStringLimit
    pop    ds

    mov    dl, [scrn_x]
    add    dl, [buf_cur]
    mov    dh, [scrn_y]
    mov    bh, DISPLAY_PAGE
    callsys 10h, 2           ; BIOS set cursor pos
    ret
```


```

public removeStringGadget

removeStringGadget:

```

```


```
```java
public removeStringGadget
.removeStringGadget:
push ds
mov si,[buf_ptr]
mov ds,[buf_seg]
mov cl,[scrn_w]
mov dl,[scrn_x]
mov dh,[scrn_y]
call printStringLimit
pop ds
ret

; --- Procedure: addStringGadgetChar
; --- Input: al=ascii character to insert
; --- Output: none

public addStringGadgetChar
addStringGadgetChar:
mov cl,[buf_len]
mov ch,0
inc cx
cmp [buf_wid],cl ; including null
jbe __bufferFull ; is the buffer already full
 unsigned please!!!
cmp [g_flags],0 ; 0=string
je short __notNumberGadget

cmp al, '+'
je short __numberGadgetChar
cmp al, '-'
je short __numberGadgetChar
cmp al, '.'
je short __numberGadgetChar
cmp al, '0'
jl short __notNumberGadgetChar
cmp al, '9'
jg short __notNumberGadgetChar

__numberGadgetChar:
__notNumberGadget:
push ds
push es
push ax
mov bl,[buf_off]
add bl,[buf_cur]
mov bh,0 ; bx=position in buffer
mov ax,[buf_seg]
mov ds,ax
mov es,ax
mov si,[buf_ptr]
push si
add si,cx ; cx=characters in buffer
dec si ; (old length)
mov di,si
inc di ; destination address one byte later
sub cx,bx ; cx=characters remaining
std ; backwards!!
```

```
loop                     __loop
    pop            si
    pop            ax
    mov           [si+bx],al    ; bx=buffer pos, still
    pop            es
    pop            ds
    inc           [buf_len]
call      setDisplayAndCursorPos
call      rightStringGadgetChar
_bufferFull:
_notNumberGadgetChar:
    ret

;       --- Procedure: delStringGadgetChar
;       --- Input: none
;       --- Output: none
;
public  delStringGadgetChar
delStringGadgetChar:
    mov           al,[buf_len]
    mov           ah,[buf_off]
    add           ah,[buf_cur]
    cmp           al,ah
    jbe          __nothingToDelete
    push          ds
    mov           bx,[buf_seg]
    mov           ds,bx
    mov           es,bx
    mov           di,[buf_ptr]
    mov           cl,ah
    mov           ch,0
    add           di,cx
    mov           si,di
    inc           si
    sub           al,ah
    mov           cl,al
    mov           ch,0
    cld
__loop:  movsb
_loop   __loop
    dec           [buf_len]
    pop           ds
call      setDisplayAndCursorPos
 nothingToDelete:
    ret

;       --- Procedure: backStringGadgetChar
;       --- Input: none
;       --- Output: none
;
public  backStringGadgetChar
```
mov al, [buf_cur]
cmp al, 0
jne short __goFunction
mov al, [buf_off]
cmp al, 0
je short __exit
__goFunction:
call leftStringGadgetChar
call delStringGadgetChar
__exit: ret

; --- Procedure: leftStringGadgetChar
; --- Input: none
; --- Output: none
;
public leftStringGadgetChar
leftStringGadgetChar:
  mov al, [buf_cur]
  mov ah, [buf_off]
  cmp al, 0
  jne short __notCursorAtStart
  cmp ah, 0
  je short __exit
  dec ah
  jmp short __setNewPos
__notCursorAtStart:
  dec al
__setNewPos:
  mov [buf_cur], al
  mov [buf_off], ah
  call setDisplayAndCursorPos
__exit: ret

; --- Procedure: rightStringGadgetChar
; --- Input: none
; --- Output: none
;
public rightStringGadgetChar
rightStringGadgetChar:
  mov bl, [buf_len]
  mov bh, [buf_off]
  mov al, [buf_cur]
  mov ah, [scrn_w]
  mov cl, bh ; is the cursor at the end of the text
  add cl, al
  cmp cl, bl
  je short __exit ; equal, at the end of the line
  inc al ; advance cursor pos by movement or scrolling
  cmp al, ah
  jb short __notAtTheEnd
  mov al, ah ; max pos is under the last character
  inc bh ; buffer offset
__notAtTheEnd:
  mov [buf_off], bh ; save new parameters
  mov [buf_cur], al
  call setDisplayAndCursorPos
; --- Procedure: printString / printCursorString
; --- Input: si=segment offset to null terminated string
; --- (dl,dh=(x,y) position to start at)
; --- Output: si=points to next byte after string's null
; --- Preserves: bx,cx,dh,di,bp,ds,es
;
public printCursorString,printString
printCursorString:
push cx
push bx
mov bh,DISPLAY_PAGE
callsys 10h,3 ; get current cursor pos
pop bx
pop cx ; ch,cl gets set to new cursor mode

printString:
push cx
push bx
mov bh,DISPLAY_PAGE
callsys 10h,2 ; already in dx - cursor pos

_loop:
    mov al,[si] ; character to print
    inc si
    cmp al,0 ; zero, end of string
    je short _end
    mov bh,DISPLAY_PAGE
    mov cx,1
callsys 10h,0ah ; BIOS print single character

    inc dl
    mov bh,DISPLAY_PAGE
callsys 10h,2 ; BIOS set cursor pos
    jmp _loop

_end: pop bx
pop cx
ret

; --- Procedure: printStringAttrib / printCursorStringAttrib
; --- Input: si=segment offset to null terminated string
; --- bl=attribute to make string
; --- (dl,dh=(x,y) position to start at)
; --- Output: si=points to next byte after string's null
; --- Preserves: cx,dh,di,ds,es
;
public printCursorStringAttrib,printStringAttrib
printCursorStringAttrib:
push cx
mov bh,DISPLAY_PAGE
callsys 10h,3 ; get current cursor pos
pop cx

printStringAttrib:
push cx ; save cx register
mov bh,DISPLAY_PAGE
_loop:
    mov al, [si] ; character to print
    inc si
    cmp al, 0 ; zero, end of string
    je _end
    mov bh, DISPLAY_PAGE
    mov cx, 1
    callsys 10h, 09h ; BIOS print single character
    inc dl
    mov bh, DISPLAY_PAGE
    callsys 10h, 2 ; BIOS set cursor pos
    jmp _loop

_end: pop cx
ret

; --- Procedure: printStringLimit / printCursorStringLimit
; --- Input: si=segment offset to null terminated string
; --- cl=length of string to print (pad with ' ' if shorter)
; --- (dl,dh=(x,y) position to start at)
; --- Output: none
; --- Preserves: dh,di,ds,es
;
public printCursorStringLimit, printStringLimit
printCursorStringLimit:
    push cx
    mov bh, DISPLAY_PAGE
    callsys 10h, 3 ; get cursor pos
    pop cx

printStringLimit:
    mov ch, 0
    push cx ; save cx register
    mov bp, sp ; to base ptr register
    mov bh, DISPLAY_PAGE
    callsys 10h, 2 ; already in dx - cursor pos

_loop:
    mov al, [si] ; character to print
    cmp al, 0 ; zero, end of string
    jne short __jmp1
    mov al, ' ' 
    jmp short __jmp2

__jmp1: inc si
__jmp2: mov bh, DISPLAY_PAGE
    mov cx, 1
    callsys 10h, 0ah ; BIOS print single character
    inc dl
    mov bh, DISPLAY_PAGE
    callsys 10h, 2 ; BIOS set cursor pos
    dec word ptr [bp]
    cmp word ptr [bp], 0
    jne _loop ; test if we should continue
    pop cx
    ret
; --- Procedure: convertItoA / convertLtoA
; --- Input: (dx): ax=unsigned (long) word to convert
; --- Output: ds: bx=pointer to an asciiiz string
; --- Preserves: si, bp, ds, es

public convertItoA, convertLtoA
convertItoA:
    mov dx, 0
convertLtoA:
    push ax
    push dx
    ; get the number of characters we will need to convert this number
    mov bx, 10
    mov cx, 0
    _loop1:
        div bx
        mov dx, 0
        inc cx
        cmp ax, 0
        jne _loop1
    ; convert this string to a ASCII string (base(bx)=10)
    pop dx
    pop ax
    mov di, offset nconbuf
    add di, cx
    mov byte ptr [di], 0 ; null terminate string
    _loop2:
        div bx
        add dx, '0'
        dec di ; pre-decrement mode -(di)
        mov [di], dl ; remainder will be less than 10
        mov dx, 0
        cmp ax, 0
        jne _loop2
    mov bx, offset nconbuf ; return value
    ret

; --- Procedure: convertAtoI
; --- Input: ds: si=asciiiz string to convert to an unsigned int
; --- Output: ax=unsigned word
; --- Preserves: di, si, ds, es
; Note: ignores '-', '+', and '.' characters

public convertAtoI
convertAtoI:
    ; find the end of this string
    mov cx, 0
    _loop1:
        mov dl, [si]
        inc cx
        inc si
        cmp dl, 0
        jne _loop1
    ; convert this string into a number
    mov bx, 1 ; base offset value
    ; base offset value
dec    si
sub    sp, 2
mov    bp, sp
mov    word ptr [bp], 0
jcxz   __exit

__loop2:
dec    si
mov    ax, 0
mov    al, [si]
cmp    al, '0'
jl     short __notNumberChar
cmp    al, '9'
jg     short __notNumberChar
sub    al, '0'
mul    bx
add    [bp], ax
mov    ax, 10
mul    bx
mov    bx, ax

__notNumberChar:
loop    __loop2

__exit:
pop    ax
ret

; end of file 'util.asm'
dosseg
locals
include header.i

.model small

.data

.error text
dw 16 dup(0) ; 2-byte integer array of 16 elements
db 64 dup(0) ; 4-byte readl array of 16 elements
db 10 dup(0) ; 10 bytes of string space
; Change these numbers if more channels are added -----------
da db SN ; number of analog channels
db SM ; number of digital I/O's

avrh...
; --- Procedure: setUpHardware
; --- Input: none
; --- Output: ax=0 if no error
; ax=1 if error, ds:dx points to error text
; --- Preserves:

public setUpHardware

setUpHardware:
mov [C], 'F'
; driver commands: read CALIB file
mov [C+1], 'n'
; get no. of channels
mov [C+2], 0
; end with null
call driver_CMD
; call driver
mov ax, word ptr [A]
; 1st integer element returned
mov bx, word ptr [A+4]
; 3rd integer element returned
mov cx, word ptr [A+12]
; 7th integer element returned
cmp ax, 0
jne short _set2
cmp bx, 0
jne short _set1
mov dx, offset nocrd
; error
jmp short _serr

_set1: mov dx, offset notsel
; error
jmp short _serr

_set2: cmp cx, 0
jne short _set3
jmp short _serr

_set3: cmp ax, cx
jle short _set4
jmp short _serr

_set4: cmp ax, 16
jg short _set5
jmp short _serr

_set5: mov bx, offset SM
jmp short _serr

_set6: mov ax, 0
ret

_serr: mov ax, -1
ret

; --- Procedure: beginDataAcquisition
; --- Input:
; --- Output:
; --- Preserves: nothing

public beginDataAcquisition
call initializeHardware
call print_cmdLineBusy

cmp [autofn],2
je short __noAutoFileName
mov ax,[autonum]
push ax
lea ax,curdate
push ax
lea ax,strbuf
push ax
call _createAutomaticFileName
add sp,6
lea ax,strbuf
push ax
lea ax,avrfl
push ax
call _insertAutoFileName
add sp,4
call print_avrfl
lea ax,strbuf
push ax
lea ax,datfl
push ax
call _insertAutoFileName
add sp,4
call print_datfl

__noAutoFileName:
  mov bx, (DACQ_ENTRIES*4)/16 ; room for DACQ_ENTRIES num of readings
  callsys 21h,48h ; allocate memory
  jnc __memoryAllocated ; returns SEGMENT pointer
  lea si,malloece
  call dosIOError
  jmp __exitDataAcq

__memoryAllocated:
  mov [memhand],ax ; save memory segment pointer
  cmp [avrfl],0
  je short __noAvrFileNec
  lea dx,avrfl
  mov cx,0
  callsys 21h,3ch ; Create new/Replace old file
  jnc __avrFileOpened
  lea si,avrfener
  call dosIOError
  jmp __exitNoAvrFile

__avrFileOpened:
  mov [avrhand],ax

__noAvrFileNec:
  mov [cursam],1 ; init current sample number

__dataAcq_MainLoop:
  cmp [datfl],0
  je short __noDatFileNec
  mov ax,[cursam]
push ax
  lea ax,[datfl]
push ax
  ...
add sp, 4

call print_datfl

lea dx, datfl
mov cx, 0 ; create new file

callsys 21h, 3ch ; Create new/Replace old file
jnc short _datFileOpened
lea si, datfl

callsys 21h, 3ch ; Create new/Replace old file

_jnc short _datFileOpened

lea si, datfl

callsys 21h, 3ch ; Create new/Replace old file

jmp _exitErrorDatFile

_datFileOpened:

mov [dathand], ax

__noDatFileNec:

mov [C], 'c'
mov [C+1], 0

call driver.Cmd
mov dx, 0 ; flag to make sure we start inactive

_waitingForTTL:

callsys 16h, 1 ; character ready
je short _waitingNoCharacter
lea si, colterm

call printTextToScreen
jmp _endDataAcquisition

_waitingNoCharacter:

mov [C], 'I'
mov [C+1], 0

call driver.Cmd
cmp [A], 0
je short _signalActiveNow
mov dx, 1 ; found at least once in inactive state
jmp _waitingForTTL

_signalActiveNow:

cmp dx, 1
je short _foundInactive
lea si, ttlLow

call printTextToScreen ; output error

_waitingForInactiveLoop:

callsys 16h, 1 ; character ready
je short _waitNoChar
lea si, colterm

call printTextToScreen
jmp _endDataAcquisition

_waitNoChar:

mov [C], 'I'
mov [C+1], 0

call driver_Cmd
cmp [A], 0
je short _waitForInactiveLoop
mov dx, 1
jmp short _noDatFileNec

_foundInactive:

callsys 21h, 2ch ; get current time
mov byte ptr [strtime], ch
mov byte ptr [strtime+1], cl
mov byte ptr [strtime+2], dh
mov byte ptr [strtime+3], dl

mov ax, [memhand] ; set registers for movsw
mov es, ax
mov di, 0
mov [numread], 0

__readValuesLoop:
```assembly
jb short __receiveBufNotFull
lea si,bufover
call printTextToScreen
jmp short __dataSetCompleted

__receiveBufNotFull:
  inc [numread]
  mov [A],1
  mov [C], 'h'
  mov [C+1], 0
  call driver_CMD
  cld
  lea si,B ; where value is
  movsw
  movsw
  mov [C], 'I'
  mov [C+1], 0
  call driver_CMD
  cmp [A], 0
  je __readValuesLoop ; keep reading values

__dataSetCompleted:
  callsys 21h,2ch ; get time
  mov byte ptr [endtime], ch
  mov byte ptr [endtime+1], cl
  mov byte ptr [endtime+2], dh
  mov byte ptr [endtime+3], dl
  mov ax, [numread] ; number read
  push ax
  push es
  mov ax, 0 ; far pointer to buffer
  push ax
  lea ax, nconbuf
  push ax ; buffer
  call _computeAverage
  add sp, 8
  mov ax, [numread]
  push ax
  mov ax, [cursam]
  push ax
  lea ax, nconbuf
  push ax
  lea ax, strbuf
  push ax
  call _createStatusString
  add sp, 8
  lea si, strbuf
  call printTextToScreen
  cmp [etime], 2
  je short __noPrintTimeTaken
  mov ax, [numread]
  push ax
  lea ax, endtime
  push ax
  lea ax, strtime
  push ax
  lea ax, strbuf
  push ax
  call _createTimeTakenString
  add sp, 8
  lea si, strbuf
  call printTextToScreen
```

115
cmp [avrr11], 0
je short __noAvrFileWrite
lea dx, nconbuf
call setStringForWrite
mov bx, [avrhand]
callsys 21h, 40h
jnc short __noAvrWriteError
lea si, awrter
call dosIOError
jmp __endDataAcquisitiofl
__noAvrWriteError:
cmp ax, cx
je short __noAvrWriteError2
mov al, -1
lea si, awrter
call dosIOError
jmp __endDataAcquisitiofl
__noAvrWriteError2:
__noAvrFileWrite:
cmp [datfl], 0
je short __noDatFileWrite
mov cx, 0
mov bx, 0
__datFileWriteLoop:
push bx
push cx
mov ax, [memhand]
mov es, ax
lea ax, nconbuf
push ax
lea di, B
mov ax, es: [bx]
mov [di], ax
mov ax, es: [bx + 2]
mov [di + 2], ax
push di
call __convertFtoA
add sp, 4
lea dx, nconbuf
call setStringForWrite
mov bx, [datehand]
callsys 21h, 40h
mov dx, cx
pop cx
pop bx
jnc short __noDatWriteError
lea si, dwrter
call dosIOError
jmp __endDataAcquisitiofl
__noDatWriteError:
cmp dx, ax
je short __noDatWriteError2
mov al, -1
lea si, dwrter
call dosIOError
jmp __endDataAcquisitiofl
__noDatWriteError2:
add bx, 4
inc cx
cmp [numread], cx
jne short __datFileWriteLoop
__noDatFileWrite:
mov bx,[datathand]
callsys 21h,3eh ; close data file
jmp __dataAcq_MainLoop

__endDataAcquisition:
cmp [autofn],2
je short _noAutoExpInc
mov ax,[autonum]
cmp ax,99
je short __numberNoCarry
inc ax
jmp short __numberCarryContinue
__numberNoCarry:
mov ax,1
__numberCarryContinue:
mov [autonum],ax
call print_autonum
__noAutoExpInc:
cmp [dataf1],0
je short _exitErrorDatFile
mov bx,[dathand]
callsys 21h,3eh ; close file handle
lea dx,datfl
callsys 21h,41h ; delete file
__exitErrorDatFile:
cmp [avrf1],0
je short _exitNoAvrFile
mov bx,[avrhand]
callsys 21h,3eh
__exitNoAvrFile:
mov ax,[memhand]
mov es,ax
callsys 21h,49h
__exitDataAcq:
call clearKeyboardBuffer
call print_cmdLine
ret

; --- Procedure: setStringForWrite
; --- Input:
; --- Output:
; --- Preserves:
;
setStringForWrite:
    mov bx,dx
    mov cx,0
__strLengthLoop:
    mov al,[bx]
cmp al,0
je short __endOfString
inc cx
inc bx
jmp short __strLengthLoop
__endOfString:
    mov byte ptr [bx],13
    inc cx
    mov byte ptr [bx+1],10
    inc cx
; --- Procedure: instant_readings
; --- Input: none
; --- Output: none
; --- Preserves:

    public instant_readings
instant_readings:
    call initializeHardware
    call print_cmdLineBusy

    mov [C],'c'
    mov [C+1],0
    call driver_CMD

loop:
    mov [C],'I'
    mov [C+1],0
    call driver_CMD
    mov ax,[A]
    push ax

    mov [A],1
    mov [C],'h'
    mov [C+1],0
    call driver_CMD

    lea ax,nconbuf
    push ax
    lea ax,B
    push ax
    call _createInstantDisplayLine
    add sp,6
    lea si,nconbuf
    call printTextToScreen

    callsys 16h,1 ; check if character is ready
    jne short _exitLoop
    call printTtsBackLine
    jmp short loop

_exitLoop:
    call clearKeyBoardBuffer
    call print_cmdLine
    ret

; --- Procedure: initializeHardware
; --- Input: none
; --- Output: none
; --- Preserves:

    PUBLIC initializeHardware
initializeHardware:
    mov ax,1 ; use one analog channel
    mov word ptr [A],ax
    mov [C],'N' ; set number of analog input channels
    mov [C+1],0

mov al, [resoll]
cmp al, 1
jne short __notResoll
mov ax, 18 ;  low noise mode
jmp short __endResol

__notResoll:
    mov ax, 2
    cmp al, 2
    jne short __notResol2
    mov ax, 16 ;  16-bit mode
    jmp short __endResol

__notResol2:
    mov ax, 3
    cmp al, 3
    jne short __notResol3
    mov ax, 15 ;  15-bit mode
    jmp short __endResol

__notResol3:
    mov ax, 4
    cmp al, 4
    jne short __notResol4
    mov ax, 14 ;  14-bit mode
    jmp short __endResol

__notResol4:
    mov ax, 5
    cmp al, 5
    jne short __notResol5
    mov ax, 13 ;  13-bit mode
    jmp short __endResol

__notResol5:
    mov ax, 12
        ;  12-bit mode
__endResol:
    mov [A], ax
    mov [C], 'a'
        ;  Set resolution of all analog input channels
    mov [C+1], 0
    call driver_CMD

    mov ax, [range]
    mov [A], ax
    mov [C], 'r'
        ;  Set range of analog input
    mov [C+1], 'c'
    mov [C+2], 0
    call driver_CMD

    mov ax, 0
        ;  set for input (0) (1= output)
    mov [A], ax
    mov [C], 'S'
        ;  set Digital I/O to input or output
    mov [C+1], 0
    call driver_CMD

    mov ax, 0
        ;  set channel switch delay time to 0
    mov [A], ax
    mov [C], 'd'
    mov [C+1], 0
    call driver_CMD

    ret

; --- Procedure: driver_CMD
; --- Input: Arrays A,B, and C initialized with command and parameters
; --- Output:
; --- Preserves: all general purpose registers

    push bp
    mov bp, sp
    ; Code follows...
mov ax, offset A
push ax
push ds
mov ax, offset B
push ax
push ds
mov ax, offset C
push ax
mov bp, sp
call check ; check if driver is there
cmp ax, -l
jz short ndrv

; clear direction flag
clcd
clei
mov ax, 'CA'
int 60H ; call trap 60h to do the command

; just for leg room
nopl
sti

; correct stack pointer
ndrv: add sp, 12
pop bp
ret

; check to make sure the driver is actually present

check:
push si
push es
mov ax, 0
mov es, ax
mov ax, es: [060h*4]
cmp ax, 0
je short ndrv
mov si, ax
mov ax, es: [060h*4+2]
cmp ax, 0
je short ndrv
mov es, ax
mov al, es: [si]
cmp al, 03dh
jne short ndrv
mov ax, 0
pop es
pop si
ret

ndrv: mov ax, -1
pop es
pop si
ret

end ; end of file 'hard.asm'
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <dos.h>

struct dosdate me {
    unsigned int year;
    unsigned char day;
    unsigned char month;
};

int insertAutoFileName(char *file, char *date) {
    char ext[127], c;
    signed int i;

    ext[0]=0;
    if (file[0]==0) goto enda;

    for (i=strlen(file); (i);) {
        c=file[--i];
        if (c=='\') {
            strcpy(ext,&file[i]);
            file[i]=0;
        } else if (c=='\') {
            strcpy(&file[i+1],date);
            goto endf;
        }
    }

    strcpy(file,date);
endf : strcat(file,ext);
enda : return 0;
}

/*
 * createAutomaticFileName - creates a string for the automatic file name
 * feature
 * buf=destination buffer for string
 * date=pointer to date structure
 * experiment=number of the current experiment
 */
{  
    sprintf(buf,"%d%02d%02d%02d",
            (int)date->month,(int)date->day,(int)(date->year-1900),experiment);
}

/************************************************************************
* createTimeTakenString - compute time and hertz and create display string
* * strbuf=destination buffer for output string
* * strtime=first time reading
* * endtime=second time reading
* * numread=number of readings taken
*/
void createTimeTakenString(char *strbuf,struct dostime_t *strtime,
                           struct dostime_t *endtime,int numread)
{
    signed char s, h;
    unsigned char buf[20];
    double time;
    int hertz;

    s=(signed char)endtime->second-strtime->second;
    h=(signed char)endtime->hsecond-strtime->hsecond;
    if (h<0) {
        h=h+100;
        s--;
    }
    sprintf(buf,"%d.%02d.E+000",(int)s,(int)h);
    time=atof(buf);
    if (!time)
        hertz=0;
    else
        hertz=(int)((double)numread/time);

    sprintf(strbuf,"Time: %d.%02d seconds. Hertz: %d",
            (int)s,(int)h,hertz);
}

/************************************************************************
* convertFtoA - converts a float to an ascii string for printing
* * variable=pointer to a float
* * buffer=destination buffer for ascii string
*/
void convertFtoA(float *variable,char *buffer)
{
    sprintf(buffer,"%e",*variable);
}

/************************************************************************
* createStatusString - create status string for printing
* * buffer=destination for string
* * average=pointer to a string for the average
* * cursam=current set number
* * numread=readingds in the current set
*/
void createStatusString(char *buffer,char *average,int cursam, int numread)
{
/* computeAverage - computes the average of all the o2 voltages read */
void computeAverage(char *buffer, float far *values, unsigned int numread)
{
    register i;
    double average, temp;
    for (i=0; i<numread; i++) {
        temp=(double)values[i]; /* does it do this anyway?? */
        average=average+temp;
    }
    temp=(double)numread;
    average=average/temp;
    sprintf(buffer," %e",average);
}

/* createInstantDisplayLine - create string for instant function */
void createInstantDisplayLine(float *variable, char *buffer, int digit)
{
    char digs[5];
    if (digit==1)
        strcpy(digs,"HIGH");
    else
        strcpy(digs,"LOW");
    sprintf(buffer,"Voltage: %e Digital: %s",*variable,digs);
}

/* mkdfname - adds an extension to the oxygen voltage reading file */
void mkdfname(char *name, int cursam)
{
    register int i;
    char ext[4];
    sprintf(&ext[0],"%03d",cursam);
    for (i=0; (1); i++) {
        if (name[i]=='.' ) {
            strcpy(&name[i+1],&ext[0]);
        }
    }
}
} else if (name[i]==0) {
    name[i]='.';
    strcpy(name[i+1], ext[0]);
    break;
}

/* end of file 'fmath.c' */
Appendix C

Program that processes spectral data from the spectrophotometric system.
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <signal.h>
#include <ctype.h>
#include <dos.h>
#include <string.h>
#include <io.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

#define MAX_LINE_BUF 256
#define FILEBUF_SIZE 0x800
#define FALSE 0
#define TRUE 1

int exitflag=FALSE;  /* flag set when a CTRL-C is hit */
char sourcefn[50], destfn[50]; /* source/dest file names */
char linebuffer[FILEBUF_SIZE]; /* output buffer for writing to file */
int linebuf_size;  /* number of chars currently in buffer */

/*
 * Usage - printf switch options to the screen
 */
void Usage(char **argv) {
    printf("\nUsage: %s [switches]\n", argv[0]);
    printf(" -binumber] starting wavelength\n");
    printf(" -d[filename) destination file name\n");
    printf(" -e [number] ending wavelength\n");
    printf(" -n[number] samples per file\n");
    printf(" -s[filename] source file name\n");
    printf(" -w[1|2] read whole number wavelengths or all\n");
    printf(" 1=read only whole number wavelengths\n");
    printf(" 2=read all wave lengths\n");
}

/*
 * chandler - Control-C interrupt handler. sets the exitflag=TRUE
 */
int chandler(void) {
    signal(SIGINT, SIG_IGN);  /* disallow Ctrl-C during handler */
    exitflag=TRUE;
    signal(SIGINT, chandler);  /* reattach handler to CTR-C */
    return 0;
}

/*
 * readline - read in a block of text ending with a linefeed \n"=10
 */
int readline(int source, char *linebuf) {
    register int x=0;
    do {  if (read(source, &linebuf[x], 1)!=1) {
        sprintf(linebuf, "Error reading '%s', sourcefn);
        perror(linebuf);
        return 1;
    } else 
```
while (linebuf[x++] != 'n');
    linebuf[x] = 0;
    return 0;
}

/*
   * writeline - write file data to disk
   */

int writeline(int filehandle)
{
    if ((write(filehandle, linebuffer, linebuf_size)) == -1) {
        sprintf(linebuffer, "Error writing to '%s'", dest_fn);
        perror(linebuffer);
        return 1;
    }
    linebuf_size = 0;
    return 0;
}

/*
   * saveline - writes the current data to memory for a future disk write
   */

int saveline(int filehandle, char *linebuf, int length)
{
    register int x = 0;
    if (length + linebuf_size > FILEBUF_SIZE) {
        if (writeline(filehandle)) return 1;
        linebuf_size = 0;
    }
    while (x < length)
        linebuffer[linebuf_size++] = linebuf[x++];
    return 0;
}

/*
   * main -
   */

int main(int argc, char **argv)
{
    char linebuf[MAX_LINE_BUF];
    /* file read buffer */
    char absorbance[25];
    /* buffer for reading in absorbance */
    int source=NULL, dest=NULL;
    /* file handles */
    int startwvlen=0, endwvlen=0;
    /* starting end ending wavelengths */
    int curwvlen, cwvlen_dec;
    /* current wavelength we are at */
    int fstwvlen, fendwvlen;
    /* wavelengths in current file */
    int fstwvl_dec, fendwvl_dec;
    /* decimal of wavelengths in file */
    int cwvlen_val;
    /* value of offset for fseek calc. */
    int incwvlen=0;
    /* number of sets in this file */
    int samplecount=0;
    /* number of samples in this file */
    struct dostime_t timel, time2;
    /* variables used for computing time */
    int m, s;
    /* more variables for computing time */
    register int x;
    /* general purpose index variable */
    sourcefn[0] = destfn[0] = 0;
    /* null file names */

    /* set CTR-C handler so we can exit properly */
    if (signal(SIGINT, chandler) == (int(*))() - 1) {
        printf("ERROR: Unable to set CTR-C handler");
        goto end;
    }

    /* process switches */
if (!(argv[0][0]=='-')) {
    printf("ERROR: Switch error\n");
    Usage(argv);
    goto end;
} else {
    switch(tolower(argv[1][1])) {
    case 'b':
        startwvlen=(int)atoi(&argv[x][2]);
        break;
    case 'd':
        strcpy(destfn,&argv[x][2]);
        break;
    case 'e':
        endwvlen=(int)atoi(&argv[x][2]);
        break;
    case 'n':
        samplecount=(int)atoi(&argv[x][2]);
        break;
    case 's':
        strcpy(sourcefn,&argv[x][2]);
        break;
    case 'w':
        incwvlen=(int)atoi(&argv[x][2]);
        if (!(incwvlen==1 || incwvlen==2)) {
            printf("ERROR: -w switch: Invalid number given\n");
            goto end;
        }
        break;
    case '?':
        Usage(argv);
        goto end;
    default:
        printf("ERROR: Unknown Switch '\%c'\n",argv[x][1]);
        Usage(argv);
        goto end;
    }
}

/* if the user hasn’t already given the parameters, get them now */
if (exitflag) { printf("\n ** Aborted **\n"); goto end; }
if (!sourcefn[0]) {
    printf("Name of file with .ASC data file: ");
    scanf("%s",sourcefn);
} else printf("Name of file with .ASC data file: %s\n",sourcefn);
if (exitflag) { printf("\n ** Aborted **\n"); goto end; }
if (!destfn[0]) {
    printf("Name of file to create: ");
    scanf("%s",destfn);
} else printf("Name of file to create: %s\n",destfn);
if (exitflag) { printf("\n ** Aborted **\n"); goto end; }
if (!samplecount) {
    printf("Number of samples in this file: ");
    scanf("%d",&samplecount);
} else printf("Number of samples in this file: %d\n",samplecount);
if (exitflag) { printf("\n ** Aborted **\n"); goto end; }
if (!startwvlen) {
    printf("Starting wavelength: ");
    scanf("%d",&startwvlen);
} else printf("Starting wavelength: %d\n",startwvlen);
if (exitflag) { printf("\n ** Aborted **\n"); goto end; }
if (!endwvlen) {
    printf("Ending wavelength: ");
    scanf("%d",&endwvlen);
} else printf("Ending wavelength: %d\n",endwvlen);
if (!incwvlen) {
    printf("Read only whole number wavelengths (y/n): ");
    loop2: if (exitflag) { printf("n ** Aborted **\n"); goto end; }
    incwvlen=(int)getchar();
    if (incwvlen=='y')
        incwvlen=1;
    else if (incwvlen=='n')
        incwvlen=2;
    else goto loop2;
} else if (incwvlen=='1')
    printf("Reading only whole number wavelengths.\n");
else printf("Reading all wavelengths.\n");
if (exitflag) { printf("n ** Aborted **\n"); goto end; }

if (startwvlen>endwvlen) {
    printf("ERROR: Starting wavelength > Ending wavelength\n");
    goto end;
}
source=open(sourcefn, 0_BINARY|O_RDONLY, 0);
if (source== -1) {
    sprintf(linebuf, "Unable to open '%s': ", sourcefn);
    perror(linebuf);
    goto end;
}
dest=open(destfn, 0_CREAT|O_TRUNC|O_BINARY|O_WRONLY, S_IWRITE);
if (dest== -1) {
    sprintf(linebuf, "Unable to open '%s': ", destfn);
    perror(linebuf);
    goto end;
}

/* set default file range and check to make sure the ranges are ok */
fstwvlen=400; fstwvl_dec=2; fendwvlen=800; fendwvl_dec=0;
if (startwvlen<fstwv~len)
    printf("ERROR: Starting wavelength too small.", fstwvlen);
    goto end;
if (endwvlen>fendwvlen)
    printf("ERROR: Ending Wavelength too large.", fendwvlen);
    goto end;

_dos_gettime(&timel);
_curwvlen=startwvlen;
cwvlen_dec=0;
for (;;) { /* forever */
    if (exitflag) { printf("n ** Aborted **\n"); goto end;
    lseek(source, OL, SEEK_SET);
    for (x=0; x<4; x++)
        if ( readline(source, linebuf) ) goto end;

    /* move file position to next line we want to read */
    if (cwvlen_dec==0) cwvlen_val=0;
    else if (cwvlen_dec==33) cwvlen_val=1;
    else if (cwvlen_dec==67) cwvlen_val=2;
    lseek(source, (long)(((curwvlen-fstwvlen) *4i+cwvlen_val) *24) + (fstwvl_dec*24), SEEK_CUR);
    printf("Wavelength: %d.%02d\r", curwvlen, cwvlen_dec, absorbance);
}

/* process current line data */
if ( readline(source, linebuf) ) goto end;
sscanf(linebuf," %*d %*c %*d %s", absorbance);
    sprintf(linebuf, "%d.%02d %s", curwvlen, cwvlen_dec, absorbance);
    if (saveline(dest, linebuf, strlen(linebuf)) ) goto end;
for (x=1; x<samplecount; x++) {
   lseek(source, (long) (((fendwvlen-curwvlen)*3)-cwvlen_val)*24
   + (fendwvl_dec*24), SEEK_CUR);
   do { if (readline(source, linebuf)) goto end;
   } while(linebuf[0]!="S");

   lseek(source, (long) (((curwvlen-fstwvlen)*3)+cwvlen_val)*24)
   + (fstwvl_dec*24), SEEK_CUR);
   if (getline(source, linebuf)) goto end;
   sscanf(linebuf,"%*d %*c %*d %s", absorbance);
   sprintf(linebuf,"%s", absorbance);
   if (saveline(dest, linebuf,strlen(linebuf))) goto end;
}
   sprintf(linebuf,"\r\n");
   if (saveline(dest, linebuf,2)) goto end;
   if (curwvlen==endwvlen) goto complete;
   if (incwvlen==2) {
      if (cwvlen_dec==0) cwvlen_dec=33;
      else if (cwvlen_dec==33) cwvlen_dec=67;
      else if (cwvlen_dec==67) { cwvlen_dec=0; curwvlen++; }
   } else curwvlen++;

complete:
   if (writeline(dest)) goto end;
   _dos_gettime(&time2);
   m=(int)time2.minute-timel.minute;
   s=(int)time2.second-timel.second;
   if (s<0) { s=s+60; m--; } printf("Elapsed Time = %02d:%02d\r\n",m,s);
end:
   if (dest) close(dest);
   if (source) close(source);

return 0;

} /* end of file 'matrix.c' */
Vincent K. Lee

Lowell High School

August 6, 1992

Letterman Army Institute of Research

Mentor - Victor W. MacDonald, Ph.D.

Mechanisms of Hypertension from
Hemoglobin-based Solutions
ABSTRACT

We examined the effects of hemoglobin-based solutions as blood substitutes and resuscitation fluids. Swine were subjected to hemorrhagic shock and then resuscitated with cell-free hemoglobin-based resuscitation fluids. Isolated rabbit hearts were perfused with hemoglobin-based solutions to test the mechanisms of hypertension observed in resuscitated swine. Hemoglobin’s affinity for nitric oxide appears to be the primary cause of hypertension.
The idea of a red cell substitute has been around for more than a century (1). The first reports of using hemoglobin (Hb) as a blood substitute date back to 1868 (1). Since that time, scientists and researchers have been scrambling to find a safe blood substitute.

The need for blood substitutes has intensified in recent years (2). Blood substitutes could relieve the limited supplies of blood available for emergency blood transfusion during a major disaster. Hemoglobin-based solutions used as blood substitutes would eliminate the problem of matching blood types during blood transfusion. Blood substitutes are being tested as potential resuscitation fluids to revive traumatized patients in situations in which access to uncompromised whole blood supplies are limited or absent. This oxygen-supplying blood substitute could save lives that might otherwise be lost.

Many successful experiments have been done to demonstrate the efficacy of hemoglobin-based solutions as possible blood substitutes (3,4). However, many safety issues remain to be resolved before this material can become a product. Hemoglobin-based
solutions have been associated with toxic side effects after being injected into patients (5,6). Vascular reactivity, leading to constriction of blood vessels, is of primary concern. Systemic vasoconstriction may cause hypertension that could inflict permanent and severe damage to the body.

The purpose of the isolated perfused heart experiment in rabbits is to examine the mechanisms whereby different types of hemoglobin-based solutions induce coronary vasoconstriction. Nitric oxide (NO) is a gas and a normal component of vascular control mechanisms that dilates blood vessels. However, hemoglobin has a high affinity for NO, and can cause vasoconstriction by interfering with normal NO-mediated vasodilation (7). Hemoglobin administration may also induce production of harmful oxygen-free radicals, which can damage tissues by attacking cell membrane lipids.

The objective of the hemorrhaged pig experiment is to determine the capabilities and hemodynamic toxicities of various hemoglobin-based solutions as resuscitation fluids. We are testing to see if hypertension exists after resuscitating a water-deprived swine with hemoglobin-based solutions.
This report will provide an overview of the testing done in the Transfusion Physiology Lab of the Blood Research Division at Letterman Army Institute of Research in San Francisco to define the potential systemic and cardiovascular toxicity of hemoglobin-based blood substitutes.

MATERIALS AND METHODS

Stroma-free hemoglobin and ααHb, formulated in Ringer's acetate, were prepared in a sterile hemoglobin production facility according to previously described methods (8).

Sterile bottles of human serum albumin in Ringer's acetate, prepared from commercial salt-poor albumin, concentrated salts, and water for injection, were used as control solutions.

Ringer's lactate was purchased from approved medical supply houses.
A. Isolated Perfused Rabbit Heart

Eight male New Zealand white rabbits, weighing 2.4 to 3.0 kg, were acquired from a commercial breeder. Animals were anesthetized with 3-1/2 cc of ketamine/Rompin/acepromazine. The thorax was opened, the heart removed immediately, and attached to an aortic cannula as previously described (9). The apparatus is shown in Figure 1. The heart was perfused with a modified Krebs-Henseleit bicarbonate (KH) buffer with a pH of 7.41. The buffer, at 37°C, was pumped through a filter. Acetylcholine and hemoglobin-based solutions were infused through a valve immediately after the filter, mixing into the buffer. The buffer then continued into an oxygenator consisting of a closed cylinder in which 25 feet of Silastic tubing (0.058 inch ID X 0.077 inch OD) was wrapped around a temperature-controlled central core. Gas mixtures (5% CO₂ plus 95% O₂) were introduced into the outer chamber around the Silastic tubing to equilibrate with the buffer, which then flowed to the heart.

A ventricular drain was created at the apex of the left ventricle to minimize ventricular filling by way of the Thebesian circulation. A
saline-filled latex balloon was inserted and secured into the left ventricle by an incision in the left atrium. The balloon pressure was monitored by a pressure transducer (P23XL, Gould, Cleveland, OH) and recorded on a strip-chart recorder (2400S, Gould, Cleveland, OH). The coronary venous outflow was monitored via a pulmonary artery cannula. A reservoir of 37°C KH buffer was raised under the heart until the ventricles were fully immersed.

Acetylcholine dose-response curves were generated by monitoring aortic pressure and titrating the drug in doses of: .2μM, .5μM, 1μM, 2μM, 4μM, 6μM, 8μM, and 10μM. The stability of the aortic pressure determined when the next dose of acetylcholine was administered. A set of baseline acetylcholine concentration versus aortic pressure measurements was established. After a brief period of 5 to 10 minutes of circulating pure buffer, the hemoglobin-based solution was introduced. Another acetylcholine titration curve was determined in the presence of the hemoglobin-based solution. After another period of circulating pure buffer through the system, a recovery dose-response curve for acetylcholine was determined.
At the end of the experiment, the atria and the right ventricle were trimmed. The left ventricle was opened and lightly blotted. A wet weight was immediately taken and a dry weight was taken after drying overnight at 110°C.

The left ventricular developed pressure (LVDP), derivative of LVDP, coronary inflow PO₂, and coronary outflow PO₂ were monitored and recorded on a strip chart recorder. Oxygen consumption was determined by measuring the difference between the content of perfusate (buffer) entering and exiting the coronary circulation, using in-line Clarke-type oxygen electrodes (Instech, Horsham, PA). All the collected data were entered into a computer spreadsheet (RS1, BBN Software Products Corp., Cambridge, MA) (9).

B. Hemorrhaged Pigs

Six 8- to 12-week old female swine weighing 13 to 18 kg, were acquired from a commercial breeder. Animals were surgically prepared by methods previously described (8). The animal preparation is shown in Figure 2. Each animal was fasted overnight.
It was given anesthesia with a mixture of isoflurane, nitrous oxide, and oxygen. A carotid artery catheter (1 mm ID) was placed in one carotid artery through a midline neck incision. A central venous catheter (1 mm ID) was inserted into one internal jugular vein, and a Swan-Ganz catheter (Pentacath, Viggo-Spectramed, Oxnard, CA) was inserted through the other internal jugular vein into the pulmonary artery. The incision was closed after the catheters were brought through the skin on the dorsal surface of the neck. A midline laparotomy was performed, following the same procedure, to remove the spleen (8,10).

Finally, an aortic catheter (2 mm ID) was introduced, and a left renal artery flow probe (6R, Transonic Systems, Inc., Ithaca, NY) was attached through a left retroperitoneal incision and brought through the skin on the dorsal midline. Velcro pouches were sewn to the skin to hold the external ends of the catheters. The catheters were flushed with heparinized saline and capped. Anesthesia was discontinued, and the animal was moved to a transfer cage, where it was observed until it awoke and then it was returned to the animal care facility (8,10).
Two days before the hemorrhage, the swine was transferred to a metabolic cage. The swine was water-deprived for two days, and fasted for 12 hours before the hemorrhage. The animal was weighed daily. Blood samples were drawn daily for blood chemistries and urine samples were collected for volume and creatinine concentration.

The swine was moved to a laboratory in its metabolic cage on the day of hemorrhage. The catheters and the flow probe were connected to pressure transducers and recording machines. Two sets of baseline blood samples were taken 15 min apart after the readings of the aortic, pulmonary artery, and central venous pressures stabilized. When hemorrhage began, aortic blood (25 ml/kg) was withdrawn by a syringe pump (Model 22, Harvard Apparatus, South Natick, MA) over the next hour at 15 min intervals. Blood samples were taken at each of these 15 min intervals. All blood samples were taken for blood chemistries and the pH, PCO₂, and PO₂, using the blood gas analyzer (Model 170 pH/Blood Gas Analyzer, Corning, Medfield, MA). Withdrawn blood was stored in CPDA-1 at room temperature for later reinfusion.
One hour after hemorrhage began, the swine was given the resuscitation fluid through the central venous catheter. Blood samples were taken at five more time points following the resuscitation. Fifteen minutes after the last blood sample, the swine was transfused with its own filtered blood, which had previously been withdrawn. A final set of blood samples were obtained. The catheters were finally flushed with heparin and placed back into the dorsal pouches, along with the flow probe wires. The swine was given free access to food and water.

Pressure transducers (P23XL, Gould, Cleveland, OH), connected to the catheters, measured the aortic, central venous, and pulmonary artery pressures. Outputs were recorded on a strip-chart recorder (3800S, Gould, Cleveland, OH). Cardiac output was measured in triplicate by thermodilution with the Swan-Ganz catheter and a cardiac output computer (COM-1, American Edwards Laboratory, Irving, CA). The ultrasonic flow probe, connected to the flow meter (Model 201, Transonic Systems, Inc., Ithaca, NY), measured the left renal artery flow. The filtered urine, collected from the bottom of the metabolic cage was measured for volume in a graduated cylinder. All
data were then entered into a computer spreadsheet (RS1, BBN Software Products Corp., Cambridge, MA).

The next day, the swine was taken to the necropsy room. It was anesthetized with sodium pentobarbital and urine specimens were collected from the dissected right ureter. Euthanasia was performed with an overdose of sodium pentobarbital. Body tissues and cavities were examined and abnormalities were recorded. Sections of various organs, including the eye, heart, lung, liver, kidney, and cerebrum, were removed and stored in 10% neutral buffered formalin and tissues were taken and prepared for light microscopic examination. All abnormalities were recorded, and then all tissues were graded for all lesions (8,10).

RESULTS

A. Isolated Perfused Rabbit Heart

It has been shown that aortic pressure increases as acetylcholine is administered to hearts in progressively increasing
concentrations (Fig. 3). When hearts are treated with ααHb, sensitivity to acetylcholine increases, indicated by a shift to the left of the dose-response curve. This increased sensitivity does not return completely to normal after ααHb is removed unless deferoxamine is also present as shown in Figure 4.

Four hearts treated with cyan-met Hb (CNmetHb) showed a slight shift toward the left in the dose-response curve (Fig. 5). However, when CNmetHb was removed the recovery curve indicated even higher sensitivity to the drug than the other two curves.

Two hearts with the CNmetHb and deferoxamine had entirely different results (Fig. 6). The control, experimental, and recovery values were so close that no differences were observed.

One heart was administered nitroglycerin, a nitrovasodilator drug (Fig. 7). Sensitivity to acetylcholine appeared to be unaffected by this treatment.

Another heart received ααHb with 2-Mercaptopropionyl glycerin, another scavenger of oxygen-free radicals that, unlike deferoxamine, does not bind free iron. This compound did not affect the increased sensitivity to acetylcholine due to ααHb (Fig. 8).
Renée M. Ward
Tamalpais High School - Pomona College
August 5, 1992
Letterman Army Institute of Research
Mentors - Gerald L. Moore, Ph.D. and Mary Edith Moore
Development of New Methods for Post-Thaw
Red Blood Cell Preservation
Abstract

Fresh human blood was collected in CPDA-1, the red blood cells (RBCs) were frozen by the Valeri high glycerol technique and stored at -80°C. Later, the RBCs were thawed, washed, and an additive solution developed at the Letterman Army Institute of Research (LAIR) was added. The RBCs were stored at 4°C for 14 or 21 days and assayed for ATP, 2,3-DPG, glucose, pH, P50, lysis, crenation, sodium, potassium, and osmolarity. The RBCs were tagged with radioactive chromium, and infused into the original blood donors. Further assaying in a gamma counter determined the percentage of RBCs retained by the donors 24 hours later. Research to date looks promising, but further studies are needed to determine if this additive solution can be used for extending the shelf life of thawed RBCs to 21 days.
Development of New Methods for Post-Thaw
Red Blood Cell Preservation
Renée M. Ward

The freezing of red blood cells (RBCs) could be an ideal way for blood banks to store rare blood types and stock blood for use in treating casualties of war or local emergencies, such as an airplane crash. However, cryopreservation of RBCs is currently impractical for wide-scale use because of the expense of freezing blood, which is four times that of storing fresh blood, and the limited 24-hour shelf life after thawing (1). The goal of our research is to make cryopreservation more practical by extending the 24-hour post-thaw life of the RBCs to 14, and perhaps even 21, days.

One of the current problems with frozen red blood cells is transportation. With only 24 hours of post-thaw shelf life, RBCs must currently be kept at -80°C while being transported (2). This requires special generators in addition to extra time and money. Two weeks of RBC viability after thawing would not only give blood banks enough time for transportation, but would vastly increase the now limited shelf life. This improved accessibility and extended viability could save the lives of people with rare blood types. In addition, these improvements would benefit the military, which has several frozen blood banks around the world intended for use during the initial stages of an armed conflict (3). Presently, these blood reserves have limited practicality because the thawed RBCs must reach the Deployable Medical System at the site of combat and be infused within 24 hours.
Our research, however, hopes to improve the effectiveness of these blood banks by providing 14 to 21 days of post-thaw shelf life.

Another benefit of our research is that it has the capability of making the nation's blood supplies safer (2). Currently in the United States, the annual number of clinical cases of post-transfusion infection with the human immunodeficiency virus (HIV) is estimated at 70, while the estimated risk of contracting HIV from one unit of transfused blood is 0.0007% (4). However, some diseases, such as HIV, cytomegalovirus, and human T-lymphotropic retrovirus, are screened by identifying antibodies to these viruses. Several months can pass after the initial exposure before an antibody response is evident, and thus infected blood with undetected antibodies passes screening each year (4). Because most donors give blood repeatedly, their RBCs could be frozen until the next occasion they gave blood, at which time it would be screened. If any diseases with long incubation periods, such as HIV, appeared, the frozen blood would be disposed of, with the possibility that it, too, was contaminated.

The potential for frozen RBCs has been overlooked due to its expense and limited post-thaw shelf life. However, the cost is steadily decreasing as new and better freezing techniques are developed (2); and, with an extended shelf life, the motivation to find cheaper methods for freezing RBCs will be even greater. This study hopes to shed light on the benefits of cryopreservation of RBCs, allowing for its use in practicality and efficacy.
Materials and Methods

Freezing

To obtain the frozen RBCs needed for our research, whole blood was collected in oversized 800 mL bags containing 63 mL of CPDA-1 (manufactured by Baxter Healthcare Corporation, Deerfield, IL). One sample was taken from the bag at this time for use in a P50 assay.

The whole blood was then centrifuged, in a Beckman J6-HC (Fullerton, CA), for 10 min at 3532 x g (5) at room temperature to separate the plasma from the RBCs. The plasma was then squeezed into a connecting bag with a plasma press, clamped off and saved for use in micro-Drabkins, sodium, potassium, and osmolarity assays. A sample of the red blood cells was used for hematocrit, morphology, hemoglobin, blood pH, P50, and fluorometric assays (See assay section of materials and methods for procedures.)

To prepare the packed RBCs for freezing, we added a total of 400 mL of Fenwal Glycerolyte 57 solution (Deerfield, IL). The top of each glycerol bottle was first swabbed with rubbing alcohol and then punctured with a plasma transfer set tube (Fenwal #4C2243; Deerfield, IL) and a vent tube (Becton Dickinson #5200 19G11/2 TW; Rutherford, NJ). The RBC bag was placed on an Eberbach shaker machine (Ann Arbor, MI) and the bottle of glycerol inverted. As the RBCs were gently being shaken, 50 mL of glycerol were added. After a 5 min rest with the shaker turned off, an additional 50 mL of glycerol were added (shaker on). After 2 min of rest (shaker off), the remaining glycerol was added (while being mixed by hand). Special care was taken to shut off the flow of...
glycerol immediately before the last few drops were added. This last step prevents air from getting into the bag.

Next, we centrifuged the glycerolized RBCs at 1250 x g (5) for 10 min at room temperature. Once again, we used a plasma press to squeeze off the supernatant, excess glycerol, into a connected bag, clamped off the bag, and discarded it. The RBCs were then placed in a sealed freezer bag and cardboard box and frozen at -80°C.

**Thawing and Washing**

Each bag of frozen RBCs was placed in a Precision Coliform Incubator Bath (Fisher Scientific; Santa Clara, CA), and heated to a temperature between 37°C and 42°C. The bag remained in the wash bath until the blood thawed and reached 37°C; this takes approximately 35 min.

A 2000 mL bag of 0.2% dextrose and 0.9% sodium chloride processing solution (Fenwal; Deerfield, IL), a 150 mL bag of 12% sodium chloride processing solution (Fenwal; Deerfield, IL), a 5 L waste bag (Haemonetics; Braintree, MA), and a 600 mL transfer pack (Fenwal; Deerfield, IL) were then attached to a wash set for the Haemonetics Blood Processor 115 (Braintree, MA). The RBCs, saline (12% sodium chloride) and wash solution (0.2% dextrose and 0.9% sodium chloride) should be positioned according to the method of Valeri (5). The RBCs were then mixed on the platform of the Haemonetics Blood Processor 115 while 50 mL of the 12% saline were added. Two minutes of rest followed with the mixer off. (If at this time any pooling, dark spots, occurs in the RBCs, the cells must be mixed by hand.) With the mixer on, 100 mL of wash
solution were added, 2 min rest (mixer off), 150 mL wash solution added (mixer on), and 2 min rest (mixer off). The centrifuge in the cell washer was then turned on and the dilute RBCs added. The flow rate of the blood into the bowl should be approximately 75 mL per minute, and the bowl should not fill in less than 5 min. When effluent appeared in the waste line, the rest of the wash solution was added. If the waste solution appears murky in the tubing, the RBC bag should be lowered to prevent the RBCs from spilling over into the waste bag. After 1000-1200 mL of wash solution have been added, the effluent is normally clear, registering less than 150 mg % free hemoglobin. This can be evaluated with a Free Hemoglobin Visual Comparator (Haemonetics Corporation; Braintree, MA). If more free hemoglobin is present, the RBCs need further washing with wash solution. Once all of the wash solution was added, the centrifuge was stopped and the RBCs were forced out of the bowl and siphoned into the 600 mL transfer pack (Fenwal; Deerfield, IL). The tubing between the bowl and transfer pack was clamped off.

The washed RBCs were centrifuged at 1500 x g, room temperature for 10 min to remove excess wash solution. The supernatant was squeezed off with a plasma press and discarded. One hundred milliliters of an additive solution developed at LAIR was then added to the RBCs and mixed by hand. A sample was taken for assaying and the remaining RBCs were stored at 4°C.
The following assays were conducted on the same day (day zero) that the RBCs were thawed and washed.

**Osmolarity Assay (Appendix I)**

We assayed 0.3 mL of plasma from each sample in triplicate on an Osmometer Model 3DII (Advanced Instruments Inc., Needham Heights, MA.) (6).

**P50 Assay (Appendix II)**

In a cuvette, we prepared a buffer solution of 5 mL Hemox solution and 20 µL of anti-foam. We covered the cuvette with parafilm and then warmed it in a heating block to 38°C. Once the buffer reached its desired temperature, we poured it into the sample chamber of a Model B Hemox-Analyzer manufactured by TCS Medical Products Company (Huntingdon Valley, PA). We added 50 µL of sample to the buffer solution. The rest of the P50 assay was done according to the Operation Manual for the Hemox-Analyzer (7).

**Hematocrit (Appendix III)**

Two heparinized hematocrit tubes were filled with sample by capillary action. They were centrifuged in a micro hematocrit IEC MB centrifuge (Needham Heights, MA) for 5 min, and then read with an IEC International Micro-Capillary Reader (Needham Heights, MA).
Morphology (Appendix IV)

We fixed a sample of RBCs with a solution containing 2% glutaraldehyde, 0.1 mol/L Na-cacodylate HCl (pH 7.2), and 0.1 mol/L sucrose (8). A 1:2 dilution of sample to fixative was made. The sample can now be stored at 4°C until the cells are viewed or counted under the microscope.

To do the morphology, we made a 1:120 dilution of sample to diluent (8.5g sodium chloride and 0.1g sodium azide per liter of distilled water) for easier cell counting. This dilution is based on a hematocrit of 40, higher hematocrits need a greater dilution factor, lower hematocrits require less dilution. Twenty microliters of the dilute fixed cells were then placed in a hemocytometer and viewed through a Universal Zeiss Microscope (Germany) with a blue filter and a 40-power objective. Three slide pictures of each sample were taken with an attached Nikon 2000 camera (Tokyo, Japan). Each slide was of a different set of sixteen squares on the hemocytometer.

Later, the slides were viewed and the RBCs counted and categorized according to the level of degradation of the RBCs (Fig. 1) (9). Each category was then placed in the following equation to rate the RBCs (8):

\[
(A\frac{1100}{E}) + (B\frac{800}{E}) + (C\frac{400}{E}) + (D\frac{100}{E})
\]

Micro-Drabkins Assay (Appendix V)

An aliquot of sample was centrifuged, and the supernatant removed. We determined the amount of hemoglobin in the supernatant with a Cobas Fara
(Roche Diagnostic Systems; Montclair, NJ). The temperature was set at 25°C, wavelength at 540 nm, and 200 μL of Drabkins solution were used as a reagent. Seven readings were taken, one every 60 seconds (10).

Sodium and Potassium Assay (Appendix VI)

Small aliquots of sample were centrifuged and the supernatant was drawn off. The supernatant was then analyzed in a Corning 480 Flame Photometer (Medfield, MA) with Corning Lithium Internal Standard 3000 (Medfield, MA). We used the method described in the instruction manual (11).

Hemoglobin Assay (Appendix VII)

Prepared in triplicate, 20 μL of sample were added to 5 mL of Drabkins reagent and gently mixed. A Beckman Spectrophotometer DU-62 at 540 nm (Fullerton, CA) was zeroed with a blank containing Drabkins reagent. A standard of cyanmethemoglobin at known concentration was prepared in a corresponding cuvette. Both the standard and sample were read in the spectrophotometer. (Readings for the sample and standard should be within 0.005 of each other, if not, repeat assay) (12).

Blood pH (Appendix VIII)

In a Corning 170 pH/Blood Gas Analyzer (Medfield, MA), the temperature was set at 37°C, the gas cylinder pressure at 300 psi, the regulator pressure to 3.5–4.5 psi, and the barometric pressure entered. A sample was put into the sample port of the pH/Blood Gas analyzer via syringe. The procedure followed the guidelines in the instruction manual (13).
Fluorometric Assays

Due to time constraints, the fluorometric assays were not conducted during my tenure at LAIR. These assays determine the amount of adenosine triphosphate, glucose, and 2,3-diphosphoglycerate in the RBCs. According to the protocol of the experiment, this procedure is done using the Moore technique (14).

Infuse Patient

After 14 or 21 days, we took 20 mL of sample and added 20 μL of radioactive sodium chromate. The sample was then allowed to sit for 20 min at room temperature with periodic mixing. At the end of 20 min, we added two doses of 30 mL 0.9% saline solution. Next, we centrifuged the sample for 10 min at 1000 x g. After centrifuging the sample, a spinal needle was inserted into the bottle to draw off the supernatant. We weighed the sample, measured its radioactivity with a dosimeter, put it in a syringe, and injected it into the original donor. Five minutes later we withdrew a 5 mL sample of the donor's blood and repeated this step five times, once every 2.5 min. Twenty-four hours later, another 5 mL sample was withdrawn from the donor.

One milliliter of each sample was then diluted with 1 mL of water. These dilutions were made in triplicate for each of the seven samples, and then placed in a gamma counter for a read-out on the radioactivity of each sample.

Assays

After 14 or 21 days, the assays performed on day zero were repeated.
Results

On the P50 assay I ran on fresh blood, the oxygen-carrying capacity, 25 mm Hg, was lower than that of average fresh blood, 27 mm Hg (15). This confirms our average P50 at zero day of 26.5 mm Hg. For the P50 assays, the temperature was kept at 37°C and the buffer at pH 7.4.

The results from the micro-Drabkins assay had a total average of 0.0696 g of hemoglobin per dL of supernatant. The average osmolarity was 304 mOsm/kg supernatant. The total average hematocrit on day zero, the day of thawing, was 67.9. On day zero the total average morphology index was 100. After 21 days, the total average morphology index dropped to 82.95. The total average blood pH at zero day was 7.028. For the hemoglobin assay at zero day, the total average was 22.9298 Hb/dL.

After freezing, the level of sodium (Na+) in the supernatant returned to 97.4% of its pre-frozen level and potassium (K+) in the supernatant returned to 49.9% of its pre-frozen level. After 14 days, the Na+ concentration decreased 30.6%, from 170.9 mM Na+ /L supernatant before freezing to 118.6 mM Na+ /L supernatant at 14 day post-thaw. The K+ level increased 133.1%, from 3.54 mM K+ /L supernatant before freezing to 47.2 mM K+ /L supernatant at 14 day post-thaw. After 21 days, the average concentration of Na+ in the supernatant decreased 27.9%, from 172.7 mM Na+ /L supernatant before freezing to 124.6 mM Na+ /L supernatant at 21 day post-thaw. After 21 days, the average concentration of K+ in the supernatant increased 144.6%, from 3.5 mM K+ /L supernatant before freezing to 8.3 mM K+ /L supernatant at 21 day post-thaw.
Discussion

The results of the different assays indicate the efficacy of the additive solution in preservation of the RBCs. Some of the assays are also used to ensure that the additive solution meets the regulations set up by the Food and Drug Administration (FDA).

On the single P50 assay I ran on fresh blood, the results were insignificant. But, if that blood had been frozen later that day and eventually thawed, the results from the assay I performed would have been used to compare the oxygen-carrying capacity of the thawed RBCs to their oxygen-carrying capacity before freezing.

The P50 data at zero day, however, indicated that the freezing and thawing process had no impact on the oxygen-carrying capacity of the RBCs, because the results were the same as those for normal fresh blood.

The results of the hemoglobin assay on RBCs would typically be used as a correction factor for the ATP and 2,3-DPG assays; however, because these assays are not documented in this paper, the results of the hemoglobin assay have not been used.

The purpose of the micro-Drabkins assay is to determine the amount of lysis that occurs in the RBCs. If the RBCs are ruptured, hemoglobin is released. To ensure that the majority of the RBCs are in good condition, the FDA maintains that the hemoglobin present in the supernatant is less than 1 g %. The average amount of hemoglobin per dL of supernatant in our study was 0.0690 g/dL, much less than FDA requirements.

The average osmolality of 304 mOsm/kg supernatant indicates the osmotic pressure of a molal concentration of an ion in the solvent. Normal osmolality for
fresh plasma is 320 mOsm/kg (Personal communication, G. Moore, Ph.D., 4 Aug. 92). Thus, this data indicates relatively normal osmotic pressure in the plasma before the freezing procedure took place.

By rating the RBCs via the morphology index, we are able to determine the condition of the RBCs. A high morphology index indicates that only a small percentage of the RBCs are crenating. On the day of thawing, the morphology index was 100; all of the RBCs were in the healthy smooth disc form. Thus, although the freezing, thawing, and washing of the RBCs are traumatic processes, they had no impact on the degradation of the RBCs. The total average morphology index for 21 days was 82.95, indicating that post-thaw storage does allow for the crenation of the RBCs. However, the normal life span for a RBC is 120 days (15), and thus over the course of 21 days, a certain percentage of the RBCs would be crenating due to their normal aging process. Because the morphology index is still high at 21 days, we can infer that the additive solution is effectively minimizing deterioration of the RBC membranes.

Normal blood has a pH between 7.0 and 7.2. At zero day, the normal pH was maintained. Previous studies have shown that when a normal pH level is maintained, the RBCs retain more of their oxygen-carrying capabilities. Thus, the freezing and thawing processes did not hinder the oxygen-carrying capability of the RBCs. This is reinforced by the P50 data at zero day, in which the normal oxygen-carrying capacity of the RBCs was maintained.

The levels of sodium (Na+) and potassium (K+) in the blood help the body regulate water retention, and are necessary to maintain life (16). Relatively high concentrations of Na+, approximately 140 μM/mL, and low concentrations of K+,
4.5 μM/mL, are found in extracellular fluids, although inside the cells, Na⁺ is low, 10-20 μM/mL, and K⁺ is high, 90-100 μM/mL (17). During the freezing of RBCs, K⁺ leaves the cell and Na⁺ enters it. After thawing, however, the concentrations of Na⁺ and K⁺ in the RBCs should return to normal, unless the cell membranes have been damaged (16). Thus the 2.6% difference between pre-frozen and thawed concentrations of Na⁺ in the supernatant and the 50% difference between pre-frozen and thawed concentrations of K⁺ in the supernatant indicate some damage occurred in the RBC membranes during freezing, thawing, and washing. After 14 days, Na⁺ concentrations decreased another 46.3 mM Na⁺/L supernatant (from concentration on day of thawing). Similar results were obtained after 21 days; thus, at this point a plateau was reached, and the additive solution maintained the concentrations of Na⁺ and K⁺ in the supernatant.

From the clinical trials, when donors were infused with their own blood after 14 or 21 days of post-thaw storage, the average in vivo survival measurements were above the FDA required level of 75% for both 14 and 21 days. (Details will be published at a later date.)

Throughout this study, RBCs remained safe and viable. Lysis and degradation of the RBCs were kept at a minimum, and the oxygen-carrying capacity of the RBCs maintained. Thus, except for disrupting the concentrations of Na⁺ and K⁺, these assays determined that the freezing and thawing procedures had minimal impact on the RBCs.
Conclusion

The RBCs from this study are still being analyzed. Eventually, the compiled data will be sent to the FDA for approval. This research is, however, discovering new methods and additive solutions that will eventually extend the post-thaw shelf life of RBCs.
Acknowledgments

In appreciation for their time, patience, and encouragement, I would like to thank my mentors, Dr. and Mrs. Moore. During this summer, they have shown me that science is a living field that is constantly evolving, and that being a part of that evolution is both challenging and rewarding.

Mrs. Siefert has been an enormous help with the writing of this paper, in particular by helping me understand the format of a scientific paper.

For his explanations on how to conduct many of the assays, time, and encouragement, I would like to thank Angulo Zegna.

Many thanks to Conrad Wheeler, Ph.D. for helping me locate a computer to work on, and his assistance with the graphs and formatting of this paper.

The first three weeks of this apprenticeship were spent in Quality Assurance with LT Cima. During that time, I learned about the "behind the scenes" process that allows research to be conducted here at LAIR. LT Cima's time and encouragement were greatly appreciated, along with his understanding, which allowed me to relocate midway in this program to allow for more hands-on lab work.

I would also like to thank the entire staff here at LAIR for making me feel welcome and for answering my many questions. Your time and kindness have been appreciated.

Human Use Statement

Human Subjects participated in these studies after giving their free and informed voluntary consent. Investigators adhered to AR 70-25 and USAMRDC Reg 50-25 on the use of volunteers in research.
References


### Appendix I

**Osmolarity before Freezing**

<table>
<thead>
<tr>
<th>Patient</th>
<th>mOsm/kg supernatant before freezing</th>
<th>Average (mOsm/kg supernatant)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample 1</td>
<td>sample 2</td>
</tr>
<tr>
<td>340</td>
<td>301</td>
<td>305</td>
</tr>
<tr>
<td>341</td>
<td>311</td>
<td>311</td>
</tr>
<tr>
<td>342</td>
<td>311</td>
<td>312</td>
</tr>
<tr>
<td>343</td>
<td>307</td>
<td>303</td>
</tr>
<tr>
<td>344</td>
<td>300</td>
<td>303</td>
</tr>
<tr>
<td>345</td>
<td>312</td>
<td>310</td>
</tr>
<tr>
<td>346</td>
<td>299</td>
<td>303</td>
</tr>
<tr>
<td>347</td>
<td>300</td>
<td>304</td>
</tr>
<tr>
<td>358</td>
<td>298</td>
<td>300</td>
</tr>
<tr>
<td>350</td>
<td>316*</td>
<td>301</td>
</tr>
<tr>
<td>351</td>
<td>302</td>
<td>300</td>
</tr>
</tbody>
</table>

*Discarded when average calculated.

| Total Osmolarity Average before freezing (mOsm/kg supernatant) | 304 |
Appendix II

P50 at Zero Day

Patient: 196
Date: 3 Aug. 92
Temp.: 37.1°C
pH: 7.4
P50: 25.0 mm Hg
P50 at Zero Day

Patient: 268
Date: 3 Aug. 92
Temp.: 37.1°C
pH: 7.4
P50: 25.0 mm Hg
Patient: 335
Date: 3 Aug. 92
Temp.: 37.1°C
pH: 7.4
P50: 26.0 mm Hg
3 Aug 92
Temp.: 37.1°C
pH 7.4
P50 26.0 mm Hg

P50 at Zero D2 O2

% Oxyhemoglobin (% HbO2)

0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 15

Oxygen Partial Pressure (mm Hg)
P50 at Zero Day

Patient: 357
Date: 3 Aug. 92
Temp.: 37.1°C
pH: 7.4
P50: 26.0 mm Hg


PSO on Fresh Blood before Freezing

Patient: 500
Date: 29 July 92
Temp.: 37.1°C
pH: 7.4
P50: 25.0 mm Hg
Appendix III

Hematocrits at Zero Day

<table>
<thead>
<tr>
<th>Patient</th>
<th>Hematocrits at Zero Day</th>
<th>Average Hematocrit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample 1</td>
<td>sample 2</td>
</tr>
<tr>
<td>196</td>
<td>71.0</td>
<td>71.0</td>
</tr>
<tr>
<td>268</td>
<td>68.0</td>
<td>68.0</td>
</tr>
<tr>
<td>335</td>
<td>66.0</td>
<td>65.0</td>
</tr>
<tr>
<td>352</td>
<td>66.5</td>
<td>67.5</td>
</tr>
<tr>
<td>357</td>
<td>68.0</td>
<td>68.0</td>
</tr>
</tbody>
</table>

Total Average Hematocrit 67.9
Appendix IV

Morphology at Zero Day

<table>
<thead>
<tr>
<th>Total Average Morphology Index at Zero Day</th>
<th>100</th>
</tr>
</thead>
</table>

21 Days after Thawing

<table>
<thead>
<tr>
<th>Patient</th>
<th>Smooth Discs</th>
<th>Crenated Discs</th>
<th>Crenated Discoids &amp; Spheroids</th>
<th>Crenated &amp; Smooth Spheres</th>
<th>Morphology Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>148</td>
<td>45</td>
<td>41</td>
<td>18</td>
<td>78.28</td>
</tr>
<tr>
<td>209</td>
<td>112</td>
<td>39</td>
<td>16</td>
<td>3</td>
<td>86.15</td>
</tr>
<tr>
<td>217</td>
<td>306</td>
<td>130</td>
<td>28</td>
<td>10</td>
<td>86.84</td>
</tr>
<tr>
<td>224</td>
<td>141</td>
<td>51</td>
<td>17</td>
<td>2</td>
<td>87.42</td>
</tr>
<tr>
<td>248</td>
<td>146</td>
<td>44</td>
<td>15</td>
<td>9</td>
<td>86.11</td>
</tr>
<tr>
<td>267</td>
<td>135</td>
<td>113</td>
<td>64</td>
<td>18</td>
<td>73.36</td>
</tr>
<tr>
<td>340</td>
<td>98</td>
<td>33</td>
<td>17</td>
<td>5</td>
<td>84.08</td>
</tr>
<tr>
<td>341</td>
<td>124</td>
<td>66</td>
<td>22</td>
<td>2</td>
<td>84.20</td>
</tr>
<tr>
<td>343</td>
<td>124</td>
<td>39</td>
<td>24</td>
<td>16</td>
<td>80.07</td>
</tr>
</tbody>
</table>

Total Average Morphology Index | 82.95 |
**Appendix V**

**Micro-Drabkins Assay at Zero Day**

<table>
<thead>
<tr>
<th>Patient</th>
<th>Hemoglobin in Supernatant (mg/dL)</th>
<th>Average (mg/dL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sample 1</td>
<td>sample 2</td>
</tr>
<tr>
<td>204</td>
<td>51.6</td>
<td>50.0</td>
</tr>
<tr>
<td>209</td>
<td>58.3</td>
<td>60.6</td>
</tr>
<tr>
<td>217</td>
<td>67.2</td>
<td>73.8</td>
</tr>
<tr>
<td>224</td>
<td>89.9</td>
<td>94.6</td>
</tr>
<tr>
<td>248</td>
<td>101.0</td>
<td>100.2</td>
</tr>
<tr>
<td>249</td>
<td>50.6</td>
<td>48.6</td>
</tr>
<tr>
<td>267</td>
<td>74.0</td>
<td>76.0</td>
</tr>
<tr>
<td>334</td>
<td>63.5</td>
<td>66.0</td>
</tr>
<tr>
<td>340</td>
<td>79.3</td>
<td>79.3</td>
</tr>
<tr>
<td>341</td>
<td>74.2</td>
<td>77.6</td>
</tr>
<tr>
<td>343</td>
<td>82.9</td>
<td>77.1</td>
</tr>
<tr>
<td>344</td>
<td>38.2</td>
<td>43.0</td>
</tr>
</tbody>
</table>

Total Average Hemoglobin in Supernatant (mg/dL) | 69.6
### Appendix VI

**Sodiums and Potassiums**

<table>
<thead>
<tr>
<th>Patient</th>
<th>mM Na⁺/L Supernatant before freezing</th>
<th>0 day</th>
<th>14 day</th>
<th>mM K⁺/L Supernatant before freezing</th>
<th>0 day</th>
<th>14 day</th>
</tr>
</thead>
<tbody>
<tr>
<td>197</td>
<td>171</td>
<td>165</td>
<td>129</td>
<td>3.61</td>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>199</td>
<td>178</td>
<td>154</td>
<td>140</td>
<td>3.74</td>
<td>1</td>
<td>22</td>
</tr>
<tr>
<td>201</td>
<td>174</td>
<td>160</td>
<td>135</td>
<td>3.60</td>
<td>1</td>
<td>38</td>
</tr>
<tr>
<td>203</td>
<td>175</td>
<td>185</td>
<td>140</td>
<td>3.68</td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>212</td>
<td>171</td>
<td>152</td>
<td>135</td>
<td>3.71</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>213</td>
<td>160</td>
<td>152</td>
<td>139</td>
<td>3.22</td>
<td>1</td>
<td>48</td>
</tr>
<tr>
<td>214</td>
<td>172</td>
<td>172</td>
<td>133</td>
<td>3.47</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>219</td>
<td>172</td>
<td>166</td>
<td>136</td>
<td>3.51</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>220</td>
<td>169</td>
<td>159</td>
<td>135</td>
<td>3.88</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>223</td>
<td>170</td>
<td>167</td>
<td>128</td>
<td>3.81</td>
<td>1</td>
<td>54</td>
</tr>
<tr>
<td>231</td>
<td>170</td>
<td>158</td>
<td>135</td>
<td>3.79</td>
<td>1</td>
<td>39</td>
</tr>
<tr>
<td>232</td>
<td>175</td>
<td>160</td>
<td>131</td>
<td>3.49</td>
<td>3</td>
<td>44</td>
</tr>
<tr>
<td>243</td>
<td>173</td>
<td>167</td>
<td>129</td>
<td>3.45</td>
<td>2</td>
<td>53</td>
</tr>
<tr>
<td>244</td>
<td>163</td>
<td>164</td>
<td>121</td>
<td>3.10</td>
<td>1</td>
<td>61</td>
</tr>
<tr>
<td>245</td>
<td>172</td>
<td>188</td>
<td>129</td>
<td>3.30</td>
<td>2</td>
<td>62</td>
</tr>
<tr>
<td>246</td>
<td>172</td>
<td>159</td>
<td>126</td>
<td>3.46</td>
<td>3</td>
<td>49</td>
</tr>
<tr>
<td>247</td>
<td>170</td>
<td>167</td>
<td>124</td>
<td>3.43</td>
<td>4</td>
<td>57</td>
</tr>
<tr>
<td>253</td>
<td>168</td>
<td>170</td>
<td>130</td>
<td>3.09</td>
<td>2</td>
<td>47</td>
</tr>
<tr>
<td>270</td>
<td>169</td>
<td>156</td>
<td>129</td>
<td>3.71</td>
<td>2</td>
<td>44</td>
</tr>
<tr>
<td>332</td>
<td>173</td>
<td>175</td>
<td>117</td>
<td>3.63</td>
<td>3</td>
<td>58</td>
</tr>
</tbody>
</table>

**Total Averages**

<table>
<thead>
<tr>
<th>mM Na⁺/L Supernatant before freezing</th>
<th>0 day</th>
<th>14 day</th>
<th>mM K⁺/L Supernatant before freezing</th>
<th>0 day</th>
<th>14 day</th>
</tr>
</thead>
<tbody>
<tr>
<td>170.9</td>
<td>164.8</td>
<td>118.6</td>
<td>3.54</td>
<td>1.65</td>
<td>47.15</td>
</tr>
</tbody>
</table>
(Sodiums and Potassiums con’d)

<table>
<thead>
<tr>
<th>Patient</th>
<th>mM Na⁺/L Supernatant</th>
<th>mM K⁺/L Supernatant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>before freezing</td>
<td>0 day</td>
</tr>
<tr>
<td>204</td>
<td>175</td>
<td>174</td>
</tr>
<tr>
<td>209</td>
<td>173</td>
<td>175</td>
</tr>
<tr>
<td>217</td>
<td>168</td>
<td>178</td>
</tr>
<tr>
<td>218</td>
<td>177</td>
<td>174</td>
</tr>
<tr>
<td>224</td>
<td>165</td>
<td>165</td>
</tr>
<tr>
<td>248</td>
<td>161</td>
<td>164</td>
</tr>
<tr>
<td>249</td>
<td>171</td>
<td>172</td>
</tr>
<tr>
<td>267</td>
<td>166</td>
<td>180</td>
</tr>
<tr>
<td>334</td>
<td>175</td>
<td>175</td>
</tr>
<tr>
<td>340</td>
<td>183</td>
<td>165</td>
</tr>
<tr>
<td>341</td>
<td>174</td>
<td>163</td>
</tr>
<tr>
<td>342</td>
<td>177</td>
<td>174</td>
</tr>
<tr>
<td>343</td>
<td>186</td>
<td>166</td>
</tr>
<tr>
<td>344</td>
<td>172</td>
<td>165</td>
</tr>
<tr>
<td>345</td>
<td>173</td>
<td>175</td>
</tr>
<tr>
<td>346</td>
<td>171</td>
<td>175</td>
</tr>
<tr>
<td>347</td>
<td>173</td>
<td>164</td>
</tr>
<tr>
<td>348</td>
<td>171</td>
<td>165</td>
</tr>
<tr>
<td>350</td>
<td>171</td>
<td>164</td>
</tr>
<tr>
<td>351</td>
<td>172</td>
<td>165</td>
</tr>
</tbody>
</table>

Total Averages

<table>
<thead>
<tr>
<th>mM Na⁺/L Supernatant</th>
<th>mM K⁺/L Supernatant</th>
</tr>
</thead>
<tbody>
<tr>
<td>before freezing</td>
<td>0 day</td>
</tr>
<tr>
<td>172.7</td>
<td>169.9</td>
</tr>
</tbody>
</table>

Total Compiled Averages

<table>
<thead>
<tr>
<th>mM Na⁺/L Supernatant</th>
<th>mM K⁺/L Supernatant</th>
</tr>
</thead>
<tbody>
<tr>
<td>before freezing</td>
<td>0 day</td>
</tr>
<tr>
<td>171.8</td>
<td>167.4</td>
</tr>
</tbody>
</table>
Appendix VII

Hemoglobin Assay at Zero Day

<table>
<thead>
<tr>
<th>Patient</th>
<th>Hb/dL at Zero Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>196</td>
<td>24.872</td>
</tr>
<tr>
<td>268</td>
<td>22.7782</td>
</tr>
<tr>
<td>335</td>
<td>22.2814</td>
</tr>
<tr>
<td>352</td>
<td>22.8356</td>
</tr>
<tr>
<td>357</td>
<td>22.1667</td>
</tr>
</tbody>
</table>

Total Average Hb/dL at Zero Day 22.9298
Appendix VIII

Blood pH at Zero Day

<table>
<thead>
<tr>
<th>Patient</th>
<th>Blood pH</th>
</tr>
</thead>
<tbody>
<tr>
<td>196</td>
<td>7.033</td>
</tr>
<tr>
<td>268</td>
<td>7.053</td>
</tr>
<tr>
<td>335</td>
<td>7.021</td>
</tr>
<tr>
<td>352</td>
<td>6.978</td>
</tr>
<tr>
<td>357</td>
<td>7.053</td>
</tr>
</tbody>
</table>

Total Average Blood pH at Zero Days: 7.028
Figure 1

A  Smooth Discs
B  Crenated Discs
C  Crenated Discoids and Crenated Spheroids
D  Crenated Spheres and Smooth Spheres
E  Total number RBCs