TECHNICAL REPORT

NO. T 95-#11

A PROTOTYPE COMPUTER PROGRAM THAT INTEGRATES PREDICTIVE MODELS AND MEDICAL HANDBOOKS FOR ALTITUDE, COLD EXPOSURE, AND HEAT STRESS

by

Matthew J. Reardon, MAJ, MC

May 1995

U.S. Army Research Institute of Environmental Medicine
Natick, Massachusetts 01760-5007

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
May 1995

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
A Prototype Computer Program that Integrates Predictive Models and Medical Handbooks for Altitude, Cold Exposure, and Heat Stress

5. FUNDING NUMBERS

6. AUTHOR(S)
Matthew J. Reardon, MAJ, MC

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Institute of Environmental Medicine
Kansas Street
Natick, MA 01760-5007

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Medical Research and Materiel Command
Fort Detrick
Frederick, MD 21702-5007

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

First, a brief overview is provided of the current state of USARIEM algorithms for predicting the physiologic strain and weather related casualty rates associated with exposure to hot or cold environments. Next, a prototype software program is described that demonstrates the technical feasibility of consolidating numeric models for altitude, cold, and heat strain with operationally oriented handbooks for environmental physiology and medicine. This software program is a unified and dynamic environmental medicine and physiology computer-based tool of potential use for military medical personnel in predeployment analysis, planning, and training. The program's altitude module utilizes a simplified model of hypoxic strain. The altitude model does not yet include closed-loop feedback physiologic mechanisms or altitude illness prediction. These are areas requiring further altitude modeling efforts. The cold strain module's data structures and input-output interfaces were designed for USARIEM's lumped-parameter cold digit model. The heat strain module employs the well-established USARIEM Heat Strain Algorithm with an expanded interface that provides improved flexibility with regard to data entry and display modes. On-line medical handbooks provide narrative, tables, and reference lists. These could be enhanced with interactive tutorials for use in medical training situations.

14. SUBJECT TERMS
Biomedical modeling, biomedical simulation, environmental physiology and medicine, heat stress, frostbite, altitude illnesses, military medicine, medical training

15. NUMBER OF PAGES
131

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	NONE

ADA294006

DISCLAIMER

DTIC AVAILABILITY NOTICE

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

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

The author would like to thank Dr. Kent B. Pandolf (Director, Environmental Medicine and Physiology Directorate, USARIEM) and Dr. Richard R. Gonzalez (Chief, Biophysics and Biomedical Modeling Division, USARIEM) for their advice and suggestions. Editorial guidance and recommendations were obtained from Shari Hallas, USARIEM's technical editor. Mrs. Karen Pederson and Mrs. Edna Safran provided invaluable secretarial and administrative support.

EXECUTIVE SUMMARY

This technical report describes a software prototype that demonstrates a concept and a method for integrating, into a unitary and comprehensive medical advisory and training software package, predictive models of soldier responses to altitude, heat, and cold stresses and on-line medical handbooks. The title of this software prototype is FLDMED representing the concept of a field medicine decision, reference, and training aid. In this version of FLDMED, the altitude module employs a University of Vermont hypoxia model, the heat strain module utilizes the USARIEM Heat Strain Model, and the cold module has an interface designed to accommodate the USARIEM lumped-parameter cold digit model. The front-end FLDMED module and the associated altitude, cold, and heat submodules utilize comon formats for popup menus, graphics, and various input-output features. Output data can be viewed from within the program in alternative formats such as charts, data grids, or detailed data listings. Input and output data can be saved as text files annotated with time-date stamps and user notes. Alternatively, the data may be saved in delimited fields for subsequent importation into spreadsheets for more complex analysis. Each functionally distinct environmental stress module permits the simultaneous definition, calculation, and analysis of four scenarios. This characteristic facilitates parameter sensitivity analysis. Operationally oriented military medical manuals for altitude, heat, and cold were also incorporated into their respective modules. This is the first USARIEM biomedical modeling product that has included document support as integral on-line components. The FLDMED concept of a comprehensive and dynamic environmental medicine and physiology software package can be further developed and customized for a variety of different military medical support applications. For example, a program such as this could be modified to provide commanders and unit surgeons with a computer-based aid for systematic analysis of a broad range of environmental and opc tional preventive medicine issues. Such a decision aid would directly suppo.. a commander's efforts to prevent environmentally related illness and injury across a wide range of potential deployment scenarios involving combinations of heat, cold, and altitude stresses. To enhance the utility for medical training, other versions could incorporate tutorial submodules. This type of integrated and comprehensive environmental and operational military medical software product could also be customized for computer-based training at locations such as the Army Medical Department's Center and School or the Uniformed Services Health Sciences Center.

INTRODUCTION

Army field medical officers and other medical personnel do not currently have an operationally oriented computer-based medical reference resource or decision aid to assist them in planning medical support for preventing environmental stress casualties and performance degradations among soldiers and units during deployments. The development of such a software tool would allow medical planners at brigade and division levels to predict and analyze soldier responses to exposures to the broad range of adverse environmental scenarios inherent in the alternative mission plans that staffs present to combat and combat support commanders (see Reardon, 1990 for an example of brigade level medical operations and health care issues in a hot desert environment).

The title of the computer program presented in this report is FLDMED which represents the notion of field medicine. This program is a concept demonstration protctype that ties together previously developed models and documents. The intent of this prototype was to demonstrate the technical feasibility of developing a comprehensive software tool to assist field medical personnel in identifying the important determinants of environmental stresses for a broad range of alternative deployment or training scenarios. Such planning and assessment software would facilitate the development of comprehensive, efficient, and operationally focused preventive medicine plans and countermeasures. This type of software-based analysis and advisory tool is expected to enable a field medical officer to distinguish and prioritize the wide spectrum of environmental threats lurking within the given range of mission options developed by staffs to present to commanders.

A program such as FLDMED would enhance the capabilities of field surgeons and staff medicine officers to make comprehensive evaluations of the environmental threats, yet provide focused environmental preventive medicine advice supporting command and staff deployment plans . It would also increase the likelihood that field medical officers will systematically consider all relevant stresses in a timely manner, thereby leading to effective, comprehensive, and efficient approaches to minimizing adverse effects of harsh environments.

The FLDMED program draws upon previously developed predictive models of the physiologic and medical consequences of the stresses of hypoxic, cold, and hot environments. In order to orient the reader to the sources from which the FLDMED modules were drawn, as well as the scope of biomedical modeling and simulation products developed at USARIEM, concise reviews of some of these environmental physiology and medicine models are provided in the following section. Figure 1 is a schematic that depicts the relationship of FLDMED to extant USARIEM models and applications. To skip the following background section and continue with the description of FLDMED, the reader may turn to page 19.

BACKGROUND: OVERVIEW OF USARIEM MODELS AND THEIR APPLICATIONS

The development of a well-validated heat strain algorithm has been the principal focus of USARIEM's biomedical modeling efforts over the past two decades. This was primarily driven by requirements to analyze and provide solutions for health and performance concerns related to heat stress incurred by soldiers conducting military operations in such diverse locations as: the hot humid jungles of Vietnam in the 1960s and early 1970s, the semitropics of Grenada in the early 1980s, the hot desert of the Middle East (Desert Shield/Storm) in the early 1990s, other hot climactic regions such as Somalia, Cuba, and Haiti in the mid 1990s, and the military training areas of the South and Southwest in the United States.

The principal uses of the USARIEM Heat Strain Algorithm at USARIEM have been for the derivation of activity modification and water ingestion guidance for inclusion into military training and operational field manuals and thermal health hazard analyses for soldiers interacting with developmental military systems undergoing testing in thermally stressful environments. As will be discussed in following sections, the USARIEM Heat Strain Algorithm has also been incorporated into a variety of soldier-oriented environmental stress monitors, a tactical meteorologic system, and a number of different wargaming simulation software systems.

Figure 1 is a schematic that depicts physiologic response models for harsh environments and the derivative applications and products developed at USARIEM. The USARIEM Heat Strain Model, as well as other USARIEM models for predicting the adverse consequences of exposure to harsh environments are described in more detail in the following subsections.

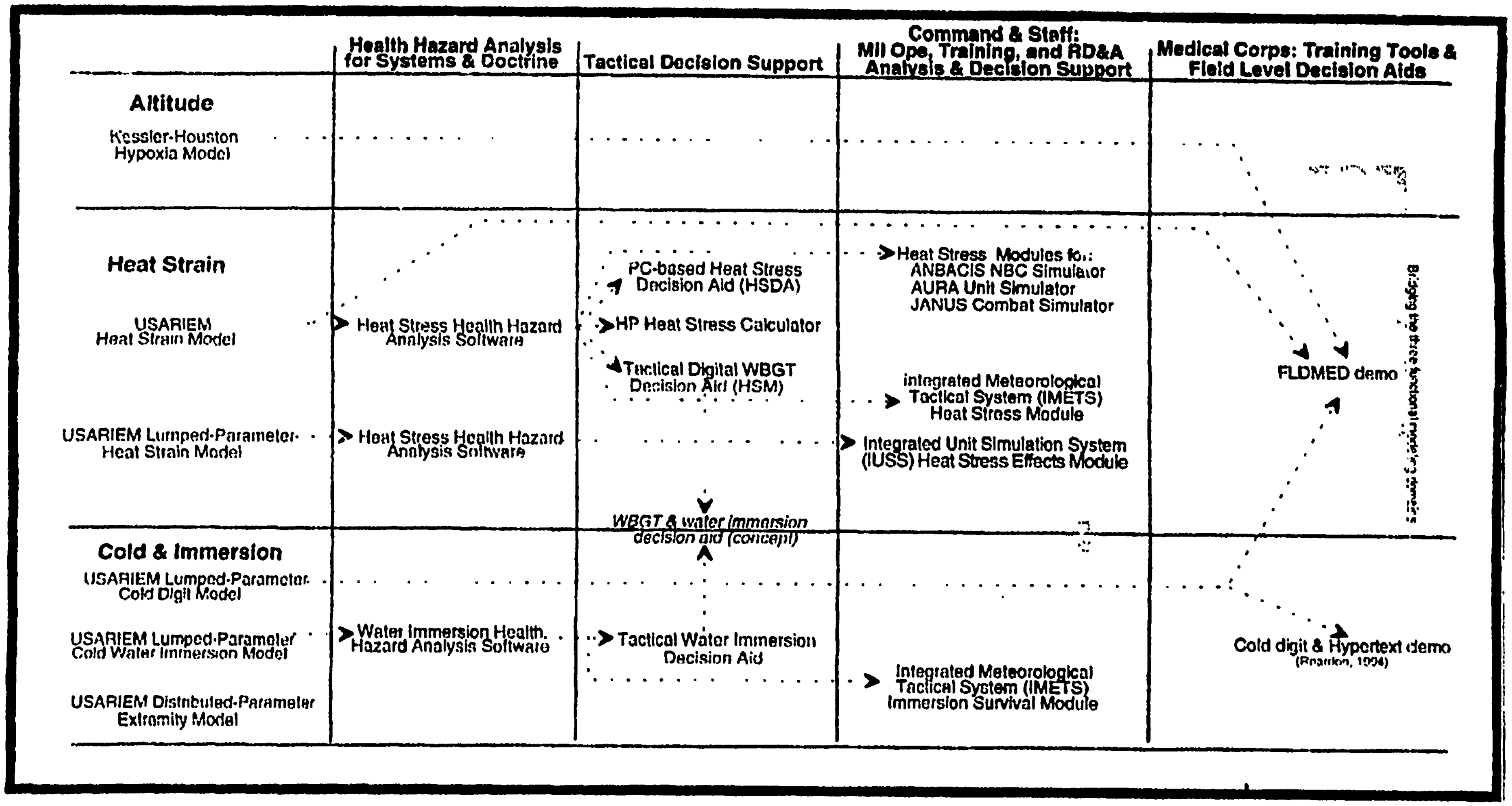

Health Hazard Analysis for Systems & Doctrine
Tactical Decision Support
Command & Staff: Mil Ops, Training, and RD&A Analysis & Decision Support
Medical Corps: Training Tools & Field Level Decision Aids
Altitude
Kessler-Houston Hypoxia Model
Heat Strain
USARIEM Heat Strain Model
USARIEM Lumped-Parameter Heat Strain Model
Heat Stress Health Hazard Analysis Software
Heat Stress Health Hazard Analysis Software
PC-based Heat Stress Decision Aid (HSDA)
HP Heat Stress Calculator
Tactical Digital WBGT Decision Aid (HSM)
Heat Stress Modules for: ANBACIS NBC Simulator AURA Unit Simulator JANUS Combat Simulator
Integrated Meteorological Tactical System (IMETS) Heat Stress Module
Integrated Unit Simulation System (IUSS) Heat Stress Effects Module
FLDMED demo
Bridging the three functional modeling domains
WBGT & water immersion decision aid (concept)
Cold & Immersion
USARIEM Lumped-Parameter Cold Digit Model
USARIEM Lumped-Parameter Cold Water Immersion Model
USARIEM Distributed-Parameter Extremity Model
Water Immersion Health Hazard Analysis Software
Tactical Water Immersion Decision Aid
Integrated Meteorological Tactical System (IMETS) Immersion Survival Module
Cold digit & Hypertext demo (Reardon, 1994)

Figure 1: A topology of USARIEM environmental stress-strain models and applications.

ADA294006

USARIEM HEAT STRAIN MODELS

The algorithm for what has become known as the USARIEM Heat Strain Model was constructed by interlinking a series of related predictive physiologic response formulas, initially published in 1972 and 1973 by Givoni and Goldman, into a unified algorithm (Berlin, Stroschein, and Goldman, 1975). USARIEM scientists, Goldman and Givoni, validated the equations discussed in their papers. Subsequent publications (Pandolf, et al.,1986; McNally, et al., 1990) presented validation results that demonstrated the generally excellent performance of the USARIEM Heat Strain Algorithm for accurately predicting the physiologic strain associated with a wide variety of heat stress scenarios.

Since the initial development of the USARIEM Heat Strain Model, slightly different software implementations have been developed for use in specific applications. Initial software implementations were programmed in FORTRAN. These predictive heat strain programs typically limited output to tabular data and character-based graphs for core temperature and heart rate as functions of time. Inputs to the programs could specify a wide variety of possible hot weather military scenarios. Since much of this initial software development occurred prior to the widespread availability of desktop or portable computers, these programs were not widely distributed. They were primarily utilized within USARIEM as simulation and prediction tools for health hazard analysis of thermal stress associated with research studies of soldier interactions with new equipment or systems and NBC protective gear in hot weather environments. Updated versions of the USARIEM Heat Strain Model continue to be used for this purpose as illustrated in the following excerpt from a recent USARIEM research study protocol:

> Individual exposure will be terminated if the rectal temperature rises more than 0.6°C per 5 minutes of exposure or rises above 39.2°C or if a volunteer exceeds 80% of her maximum predicted heart rate defined as 220 - age (in years) for a period of five minutes.
>
> The USARIEM Heat Strain Model predicts that unacclimated, sedentary volunteers wearing approximately MOPP 1 and exposed to the two environmental scenarios as proposed in this study qualify for Category 1. "No Limit" designation. Consequently medical coverage will consist of the long-range, on-call option (Endrusick and Gonzalez, 1994).

An innovative adaptation of the USARIEM Heat Strain Algorithm was proposed in the mid-1980s in order to directly extend the benefits of its predictive capabilities to members of small military units (e.g., Special Forces and Rangers) deployed or training in hot weather environments. This concept resulted in a design that embedded the USARIEM Heat Strain Algorithm into a customized Hewlett Packard (HP) hand-held calculator (Pandolf et al., 1986). The ensuing product, a hand-held, lightweight, heat strain calculator, was a technical success. The soldier-oriented heat strain calculator was capable of displaying recommendations for work-rest cycles, hourly water consumption, and maximum single-shot work times. Such readily available quantitative advisory information was meant to supplant the typically inaccurate subjective estimations of safe work-rest cycles and water requirements, as well as obviate the undesirable practice of relying on the appearance of overt sensations of excessive heat stress, definite thirst, and symptoms of impending heat injury as indications that heat stress and dehydration safety limits were being exceeded.

Developmental prototypes of the heat strain calculator successfully completed technical and operational testing. Although this device was not fielded, the concept and technical feasibility were verified Further efforts, therefore, were directed toward improving and refining the design. Design reevaluations identified that adding reduced-size meteorologic sensors to give the heat strain calculator a capability for capturing site-specific environmental conditions would be a practical and significant enhancement. Since the heat strain calculator did not have integrated weather sensors, users either had to obtain ambient temperature, humidity, wind speed, and radiant heat load from units (at possibly distant locations) monitoring the wet-bulb and globe temperatures (WBGT) or entered estimated values by selecting, from a scrolling menu, the applicable weather category (e.g., hot-dry, hot-wet, temperate, or jungle [each of these categories were associated with a set of default environmental parameter values in a look-up table that was incorporated into the heat stress calculator software]).

A new microprocessor-based environmental advisory devise was designed that incorporated capabilities for real-time local WBGT monitoring. With this developmental effort the heat strain calculator was successfully superseded by a more capable or "intelligent" heat strain monitor (HSM) with onboard WBGT sensors (see Figure 2 for a photograph of the HSM). Design, development, and test plans for this second-generation hand-held heat strain monitor were successfully implemented (Matthew, et al., 1993). HSM prototypes were fabricated by the Southwest Research Institute (SwRI, San Antonio, Tex.) under contract to USARIEM. As specified in the design, the prototype devices incorporated a stowable set of miniature weather sensors. The sensor suite measures and stores the local dry and black globe

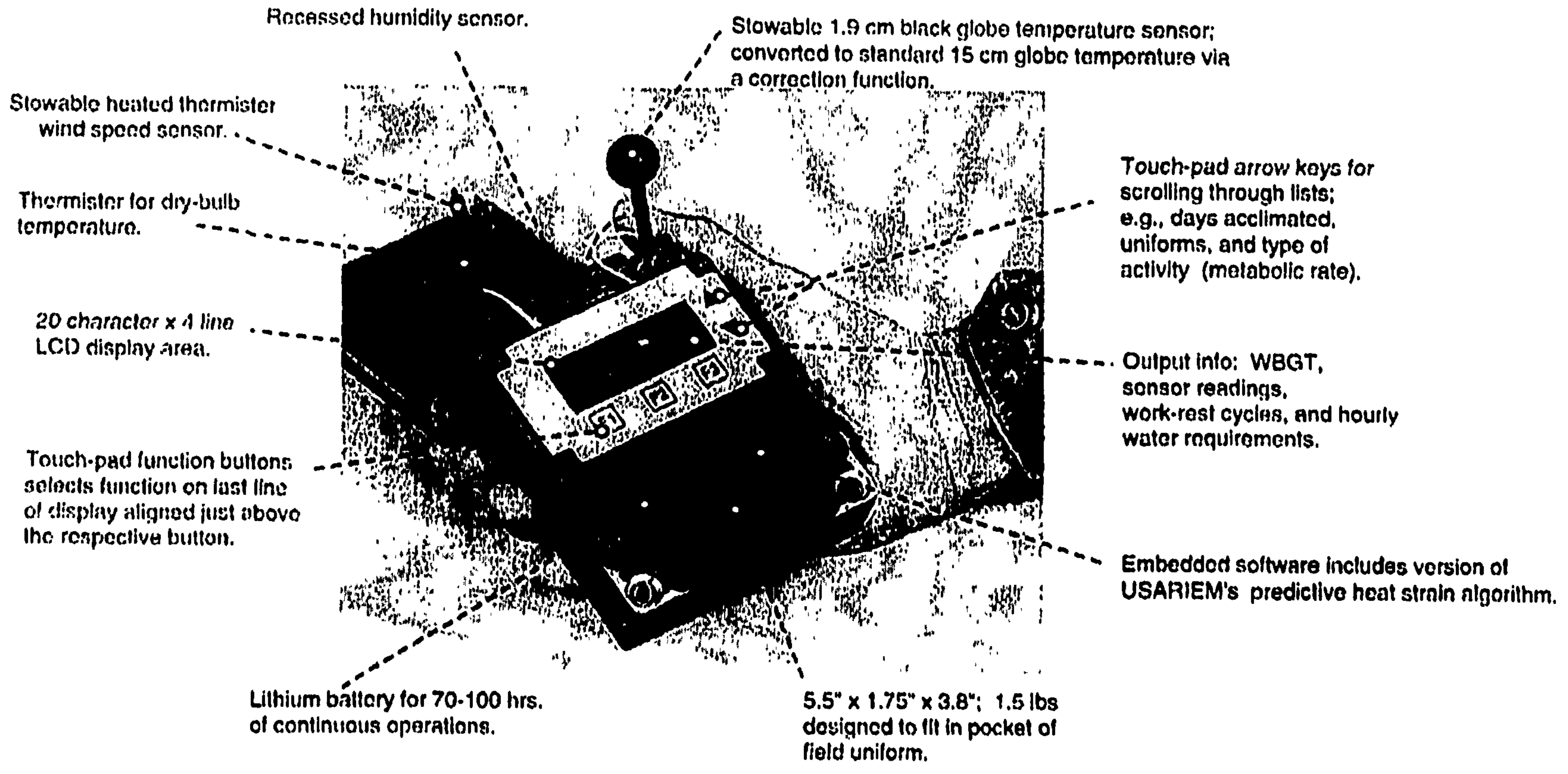

Recessed humidity sensor.
Stowable 1.9 cm black globe temperature sensor; converted to standard 15 cm globe temperature via a correction function.
Stowable heated thermister wind speed sensor.
Thermister for dry-bulb temperature.
Touch-pad arrow keys for scrolling through lists; e.g., days acclimated, uniforms, and type of activity (metabolic rate).
20 character x 4 line LCD display area.
Output info: WBGT, sensor readings, work-rest cycles, and hourly water requirements.
Touch-pad function buttons selects function on last line of display aligned just above the respective button.
Embedded software includes version of USARIEM's predictive heat strain algorithm.
Lithium battery for 70-100 hrs. of continuous operations.
5.5" x 1.75" x 3.8"; 1.5 lbs designed to fit in pocket of field uniform.

Figure 2: USARIEM Digital WBGT Monitor and Heat Stress Advisory Device
(Matthew et al., 1993)

ADA294006

temperatures, wind speed, and humidity. The WBGT itself is calculated from this real-time sensor data by internal subroutine in the embedded software.

The miniature weather sensor suite for the HSM digital heat strain monitoring and advisory device directly measures dry bulb and black globe temperatures, humidity, and wind speed. The raw data from these sensors are captured and provide the weather related inputs for the embedded algorithms that calculate and display the corresponding WBGT. These sensor data are also processed by the USARIEM Heat Strain Algorithm residing as an executable module in an onboard read-only memory microchip (ROM).

Soldier, uniform, activity, and other scenario-specific user inputs required by the embedded USARIEM Heat Strain Algorithm are entered by navigating through a shallow, easily traversed, hierarchy of input menus presented to the user on the liquid crystal display (LCD) area on the face of the de[illegible] Output data are also displayed on the LCD screen. The HSM calculates and disp[illegible] variety of tactically useful outputs such as maximum recommended work-rest cycles, one-shot maximum work time, and the corresponding hourly potable water intake requirements.

Operational field testing in desert Army training areas and at an Australian oil-refining facility demonstrated that HSM prototypes were rugged and reliable in hot environments in a representative variety of operational military and civilian settings (Gonzalez, 1995).

Technical testing of the prototype digital heat stress monitors in the carefully controlled conditions of the USARIEM environmental simulation chambers further validated the HSM concept and technical performance (Matthew, et al., 1993). That testing, however, did identify a few discrepancies in several regions of the validation envelope which were targeted for subsequent correction. For example, the variance of wind speed sensor measurements from calibrated chamber wind speed values was somewhat elevated at higher wind speeds. The advisory outputs for work-rest cycle times and water ingestion recommendations in some cases deviated from those provided by a validated PC-based version of the USARIEM Heat Strain Model. This was attributed to a possible software bug in the HSM software. The USARIEM project manager directed the contractor to investigate and corrected these software performance discrepancies. Additionally, a few hardware modifications were suggested for improving the HSM's mechanical reliability and ease of use.

Technical testing, therefore, served its purpose by leading to a number of specific recommendations for improving the accuracy of the HSM sensors and embedded software. The contract with SwRI was extended to implement these recommendations.

The HSM was primarily intended for use by small combat and combat support units. It has also been considered, by some, as a promising candidate for replacing the analog Wechsler WBGT thermometer systems currently deployed by field preventive medicine personnel and medical units. Despite largely successful testing, however, the digital HSM currently remains classified as a concept demonstration and validation prototype item. It is anticipated, however, that some type of modified version of the HSM will be fielded in the future.

Figure 3 depicts a progression of military thermal stress monitoring devices. The technology of the 1960s and 1970s is represented by the analog Wechsler glass WBGT thermometer kit (essentially identical to the Stortz Wetbulb/Globe Temperature Kit, PSG Industries, NSN 6665-00-159-2218). This kit incorporates a plastic sliderule for the user to determine the WBGT from the component dry, wet, and black bulb thermometer readings obviating the need for explicit calculations. The sliderule also provides color-coded thermal stress threat categories for modulating work or training intensity. The USARIEM HSM, in the center of Figure 3, is representative of the thermal stress monitoring technology of the early to mid-1990s. The pictures on the right-hand side of Figure 3 depict examples of the many possible year 2000+ technology alternatives for expanding and incorporating the functionality of USARIEM's microprocessor-based environmental monitoring devices into advanced individual soldier systems such as the 21st Century Land Warrior ensemble (Gourley, 1995).

The USARIEM Heat Strain Algorithm has also been embedded in the Integrated Meteorologic System (IMETS). IMETS is a complex workstation-based weather intelligence analysis and decision-support product that integrates with the Army's Tactical Command and Control System (Harris, 1994 and US Air Force, Air Weather Service, 1993). Initial IMETS prototypes were built in 1993. Testing and evaluation of IMETS prototypes apparently were successful, and the first operational versions of were scheduled for fielding in 1995 (Figure 4 is a schematic of the IMETS equipment set). The USARIEM heat strain component for IMETS, however, is currently in the prototyping and testing stages. Therefore, this capability will be inserted into IMETS as a software enhancement at a latter date.

The USARIEM heat strain and heat casualty prediction module for IMETS will, for example, allow units to generate color-coded contours of heat casualty risk as terrain-map overlays. Such overlays could be in the form of isoprobability contour

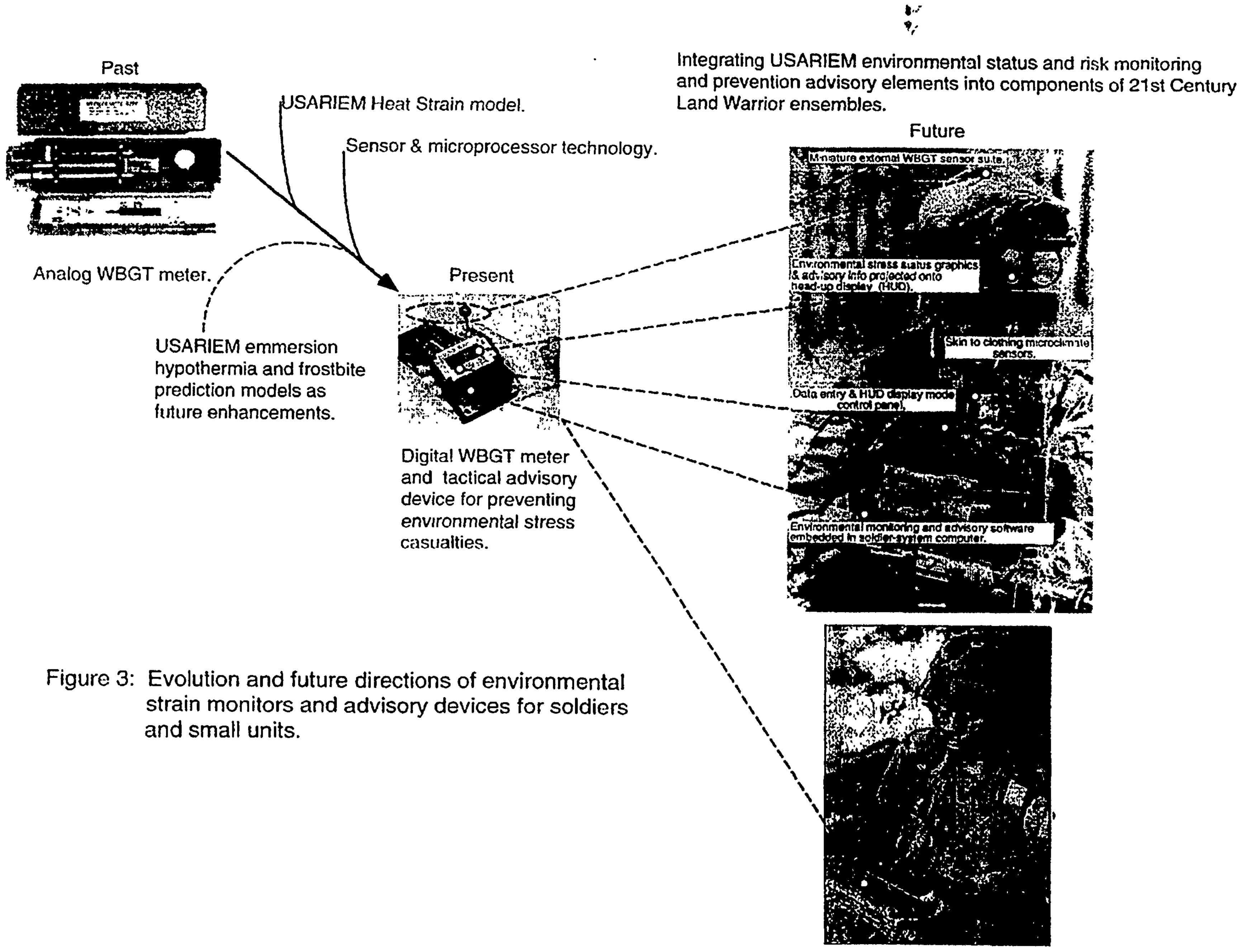

Past
USARIEM *Heat Strain* model.
Sensor & microprocessor technology.
Integrating USARIEM environmental status and risk monitoring and prevention advisory elements into components of 21st Century *Land Warrior* ensembles.
Future
Analog WBGT meter.
Present
USARIEM emmersion hypothermia and frostbite prediction models as future enhancements.
Digital WBGT meter and tactical advisory device for preventing environmental stress casualties.
Miniature external WBGT sensor suite.
Environmental stress status graphics & advisory info projected onto head-up display (HUD).
Skin to clothing microclimate sensors.
Data entry & HUD display mode control panel.
Environmental monitoring and advisory software embedded in soldier-system computer.

Figure 3: Evolution and future directions of environmental strain monitors and advisory devices for soldiers and small units.

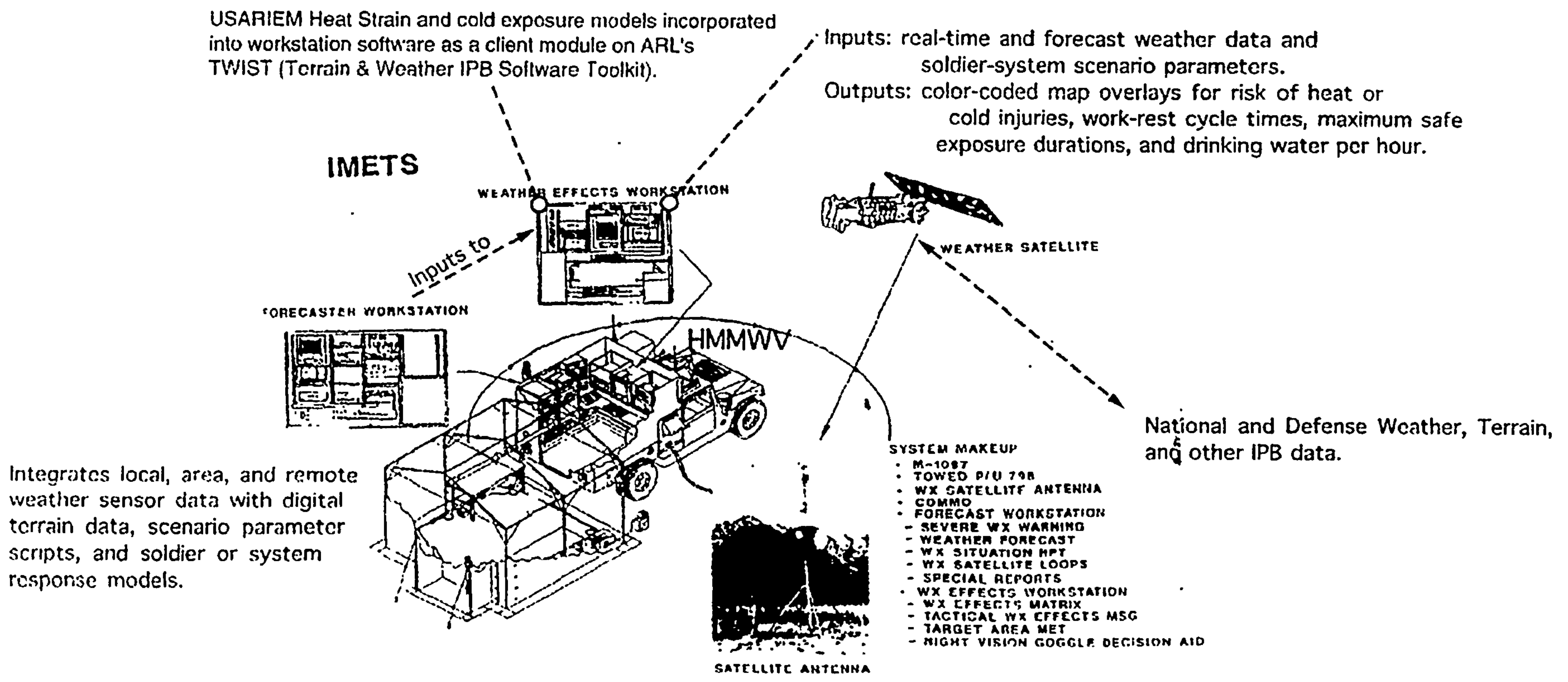

USARIEM Heat Strain and cold exposure models incorporated into workstation software as a client module on ARL's TWIST (Terrain & Weather IPB Software Toolkit).
Inputs: real-time and forecast weather data and soldier-system scenario parameters.
Outputs: color-coded map overlays for risk of heat or cold injuries, work-rest cycle times, maximum safe exposure durations, and drinking water per hour.
IMETS
WEATHER EFFECTS WORKSTATION
Inputs to
FORECASTER WORKSTATION
HMMWV
WEATHER SATELLITE
National and Defense Weather, Terrain, and other IPB data.
Integrates local, area, and remote weather sensor data with digital terrain data, scenario parameter scripts, and soldier or system response models.
SYSTEM MAKEUP
· M-1097
· TOWED P/U 798
· WX SATELLITE ANTENNA
· COMMO
· FORECAST WORKSTATION
- SEVERE WX WARNING
- WEATHER FORECAST
- WX SITUATION RPT
- WX SATELLITE LOOPS
- SPECIAL REPORTS
· WX EFFECTS WORKSTATION
- WX EFFECTS MATRIX
- TACTICAL WX EFFECTS MSG
- TARGET AREA MET
- NIGHT VISION GOGGLE DECISION AID
SATELLITE ANTENNA

Figure 4: The Integrated Meteorological System (IMETS)

maps for heat casualties or other formats depicting different levels of predicted heat-related performance degradation, optimal work-rest times, water intake requirements, etc. This USARIEM contribution to IMETS will assist military planners in determining the degree of adverse interactions between weather, terrain, and soldiers along alternative routes for various types of military missions.

The USARIEM Heat Strain Model has also been incorporated into other military simulation, training, and analysis and decision aid products as a supporting biomedical component. The Science Applications International Corporation (SAIC), under contract to USARIEM, implemented the USARIEM Heat Strain Algorithm as a personal computer, or workstation-based, tactical decision aid (SAIC, 1993). The resulting program was entitled "Heat Stress Decision Aid" (HSDA). This program was designed primarily for use by combat unit staff officers and NCO leaders to reduce the likelihood of dehydration and heat stress casualties among their troops. SAIC software developers used the FLDMED heat stress module interface described in this report as a design aid in developing the HSDA interface (SAIC, 1993, Acknowledgments section).

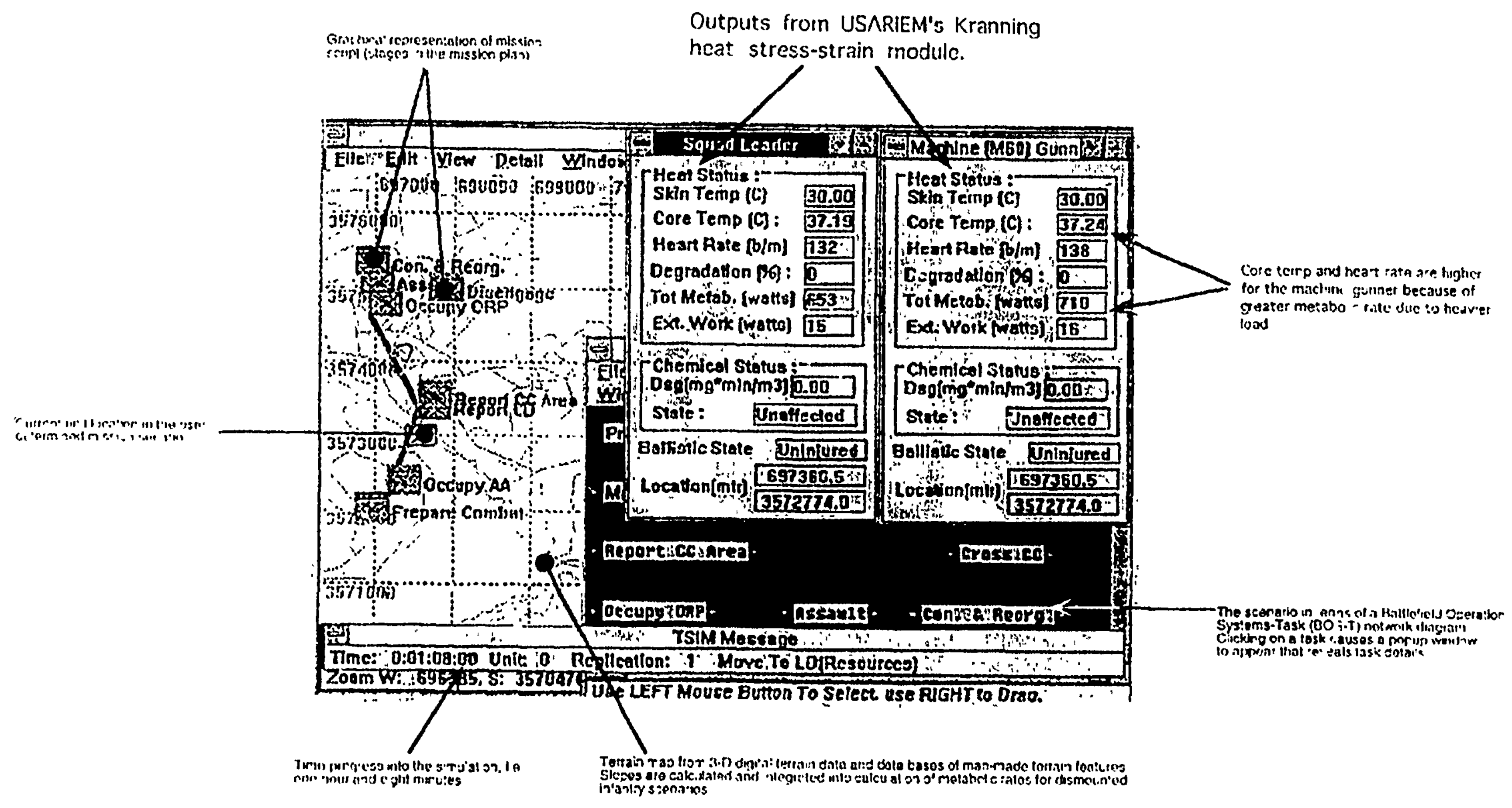
The USARIEM Heat Strain Algorithm has also been included in combat simulation software systems designed for command and staff planning and training support. The USARIEM predictive Heat Strain Algorithm, for example, was incorporated into the Army Unit Resiliency Analysis model (McNally, Stark, and Ellzy, 1990). Additionally, in collaboration with the U.S. Army's School of Chemical Defense at Fort McClellan, SAIC adapted the USARIEM Heat Strain Algorithm for use as a component in the Automated Nuclear Biologic And Chemical Information System (ANBACIS), an NBC effects simulator. The predictive heat stress module embedded in ANBACIS utilized numeric subroutines programmed in Ada that were developed initially as part of the HSDA software. The programming language Ada (ANSI, 1983; Saib, 1985; Naiditch, 1995) is currently the DoD's mandated programming language for C^3I systems. The use of Ada as the baseline programming language also facilitated porting the HSDA's numeric software modules from the PC environment to ANBACIS's Unix-based workstation environment with minimal changes to the core elements of the software code.

In addition to the USARIEM Heat Strain Model, whose equations were derived principally by use of various curve-fitting techniques, USARIEM has developed a more analytically based multicompartment, predictive heat strain algorithm. This multicompartment physiologic strain model has two dominant components. The first component is composed of a system of coupled passive heat flow equations. The second component provides thermoregulatory feedback control. The system of passive heat flow equations mathematically describes the dynamics of open-loop heat flow within and across tissue and vascular compartments as derived from the basic principles of biophysical heat transfer. The equations and relationships comprising the control component implement either nonlinear or piece-wise linear physiologically based thermoregulatory and cardiovascular feedback mechanisms (Kranning II, 1991).

This multicompartment closed-loop thermoregulatory algorithm simulates the body as a single cylindrical segment containing six concentric isotropic compartments (five tissue layers and a central vascular or core compartment). The passive heat transfer component of this model can be represented in terse matrix notation as a system of coupled nonhomogeneous first order differential equations (Appendix C). The active thermoregulatory control components of the model as well as the specific coefficient values for the various equations largely distinguishes this model from previous multi-node thermal strain models (e.g. Stolwijk and Hardy, 1977). The model has been successfully validated against data subsets from the expanding USARIEM's database of soldier-oriented heat stress study results (GEO-Centers, 1992). This analytically derived thermoregulatory model has been implemented at USARIEM as a BASIC software program entitled SCENARIO.

The SCENARIO lumped-parameter heat strain model as been used within USARIEM for thermal stress health hazard analyses. It also has been implemented in C++ and inserted as a thermal response module into the Integrated Unit Simulation System (IUSS). This was accomplished by Simulation Technologies, Inc. (Dayton, OH) under a software development contract (1993). Figure 5 illustrates a representative computer screen display from the IUSS simulation program which provides an example of the type of outputs provided by the embedded USARIEM SCENARIO heat strain model.

The SCENARIO thermoregulatory model can generate dynamic temperature profiles for each of the six simulated tissue compartments. In contrast, the USARIEM Heat Strain Model, equivalent in many respects to a one-segment, one-compartment morphologic abstraction of the human body, generates a single, time-dependent, predictive, core temperature profile. Another advantage of USARIEM's SCENARIO model over the USARIEM Heat Strain Model is that SCENARIO has greater capability for simulating cardiovascular and bloodflow responses to heat stress. Although both the USARIEM Heat Strain Model and SCENARIO can simulate changes in heart rate, SCENARIO can also simulate changes in cardiac output, and intercompartmental bloodflow redistribution as functions of time. The cardiovascular responses for both models, however, have not yet been as extensively validated as the core temperature prediction capabilities.


Outputs from USARIEM's Kranning heat stress-strain module.
Graphical representation of mission script (stages in the mission plan)
Current unit location in the system and m simulation
Core temp and heart rate are higher for the machine gunner because of greater metabolic rate due to heavier load
The scenario in terms of a Battlefield Operation Systems-Task (BOS-T) network diagram. Clicking on a task causes a popup window to appear that reveals task details
Time progress into the simulation, i.e. one hour and eight minutes
Terrain map from 3-D digital terrain data and data bases of man-made terrain features. Slopes are calculated and integrated into calculation of metabolic rates for dismounted infantry scenarios
File Edit View Detail Window
Con. & Reorg.
Assault
Disengage
Occupy ORP
Report CC Area
Report LD
Occupy AA
Prepare Combat
Report:CC:Area
Cross:LD
Occupy:ORP
Assault
Con:C&:Reorg:
TSIM Message

Squad Leader

Heat Status :
Skin Temp (C) 30.00
Core Temp (C) : 37.19
Heart Rate (b/m) 132
Degradation (%) : 0
Tot Metab. (watts) 653
Ext. Work (watts) 16

Chemical Status :
Dsg(mg*min/m3) 0.00
State : Unaffected

Ballistic State Uninjured
Location(mtr) 697360.5
3572774.0

Machine (M60) Gunn

Heat Status :
Skin Temp (C) 30.00
Core Temp (C) : 37.24
Heart Rate (b/m) 138
Degradation (%) : 0
Tot Metab. (watts) 710
Ext. Work (watts) 16

Chemical Status :
Dsg(mg*min/m3) 0.00
State : Unaffected

Ballistic State Uninjured
Location(mtr) 697360.5
3572774.0

Time: 0:01:08:00 Unit: 0 Replication: 1 Move To LD(Resources)
Zoom W: 696385, S: 3570470
Use LEFT Mouse Button To Select, use RIGHT to Drag.

Figure 5: Outputs from USARIEM's 6-node heat strain model in the Integrated Unit Simulation System (IUSS) interface.

USARIEM COLD EXPOSURE MODELS

Recently (15 Feb. 95), four U.S. Army Ranger candidates perished from complications of generalized hypothermia secondary to prolonged immersion. This occurred during the swamp phase of the US Army Ranger training cycle conducted within the Eglin Air Force Base area in Florida. Although accident investigations are still in progress, unofficial initial reports (e.g., Walker, 1995 a and b) indicated that the hypothermia fatalities occurred after about six continuous hours of partial to complete immersion. Investigators' estimates of air and water temperatures have been 65°F and 52°F, respectively.

In order to provide a predictive immersion tolerance decision aid to field and training units for the purpose of preventing incidents similar to that recently incurred by the Rangers, USARIEM leaders have oriented cold water immersion modeling efforts to support the development and prototyping of a pocket-sized water temperature monitoring and tolerance time advisory device. This device is likely to have similar physical and interface characteristics as the digital heat stress monitor (HSM) described above. A possible expedient design option is the addition of a water temperature probe and insertion of the USARIEM cold immersion model into the HSM. Such a device would then be given a more inclusive product label reflecting its expanded utility for a broader range of environmental stress scenarios.

For this effort, USARIEM has been able to build upon an immersion hypothermia model previously developed at USARIEM by Tikuisis, Gonzalez, and Pandolf (1987 and 1988). This mathematically oriented model (USARIEM Water Immersion Model) simulates the effects of cold immersion with a multisegment. six-compartment (five tissue compartments and one central blood compartment) morphologic abstraction of the human body. It is a lumped parameter model with the passive heat distribution component representable as a series of nonhomogeneous, coupled, first order differential equations. As with the multicompartment heat strain model, these relationships can be tersely summarized with the use of matrix and vector notation.

The performance of the USARIEM lumped-parameter Water Immersion Model has been validated for water temperatures of 20°C (68°F) and 28°C (82.4°F). Further efforts are being directed toward extending the range of this model's validated performance envelope for cool to cold water temperatures. An additional enhancement is planned for this immersion model that will include extent of immersion as an independent variable.

A cold water immersion tolerance prediction capability is also being planned for inclusion into IMETS. The insertion of a predictive immersion hypothermia model in IMETS would enable the system to generate risk contour-map overlays for immersion hypothermia. Such contour ma,, overlays could be useful for identifying terrain-dependent emersion-time safety limits for dismounted training or combat operations. This type of analysis and decision tool would also be of importance in planning the distribution and dispatching of search and rescue assets for aviators downed over bodies of cool to cold water. Maximal permissible response times would be obtained from IMETS and the search and rescue assets deployed so that these physiologically based response time limits would not be exceeded.

In addition to a cold water immersion model for predicting hypothermia, USARIEM cold extremity mathematical models have been developed that predict responses of digits and extremities to prolonged cold exposure. The cold exposed extremity or digit models are of two types. One type is lumped parameter (Shitzer, et al., 1990 and 1993), which assumes isotropic and homogeneous tissue properties within each compartment and near instantaneous and simultaneous temperature changes at all points within a compartment (i.e., no intracompartmental temperature gradients). The other type of mathematical model utilizes a distributed parameter representation (Shitzer, et al., 1994).

A distributed parameter system can model the additional complexities associated with variations of tissue properties within compartments and the existence of int'?^ompartmental temperature gradients. In this type of model, thermal energy is not assumed to distribute uniformly and instantaneously within any compartment. Implementations of these two types of cold exposure models are currently in different stages of parameter identification, verification, and validation. The rate of progress in this regard indicates that the lumped-parameter model will become operationally available before the substantially more complicated distributed-parameter cold extremity model.

The USARIEM cold extremity models will provide the ability to predict risks of cold injuries, such as frostbite, to the extremities. These predictive cold extremity models could also be used to further expand the utility of IMETS for cold weather operations. This will enable IMETS, for example, to generate contour-map overlays depicting isoprobability contours for risk of frostbite as functions of real-time weather, forecasted weather, terrain, and soldier-associated parameters such as type of gloves and uniform, and estimates of various physiologic parameters such as extent of dehydration. Similarly, automatically generated contour-map overlays could also delineate predictions for spatially varying maximum outdoor cold exposure limits.

These type of informational products would provide physiologically based guidance for exposure-rewarming cycle times for sentry duty or for the soldiers of units manning defensive perimeters at specific locations in cold weather environments.

USARIEM ALTITUDE EXPOSURE MODELS

A USARIEM model for predicting the physiologic responses, performance decrements, and medical illnesses associated with exposures to acute and chronic altitude (or hypoxia) stress is currently in the concept definition and early design phase. USARIEM research physicians and physiologists in the Altitude Physiology and Medicine Division have generated a considerable body of literature (e.g., see the compendium of papers in Houston, et al., 1991) to support the eventual development of a comprehensive predictive altitude response model. An interim expedient approach is the initial creation of relatively simple predictive correlational models for altitude illness with subsequent development of more sophisticated and robust causal or servomechanistic models.

Causal altitude response models would characteristically be predicated on mathematical descriptions of the relevant physiologic and biophysical principals pertaining to altitude-associated stress-strain relationships. A closed-loop altitude model would also require delineation of the appropriate physiological feedback control mechanisms.

A rational or analytic altitude exposure model is preferred over a totally empiric or correlational derivation. This is because a rational model draws not only upon the data sets used to specify parameter values, but more importantly, on the robust, validated, and well accepted body of knowledge consisting of the fundamental and generally applicable physiologic, biophysical, and control principals implicit in the model's constitutive equations. During the design and development of a rational model, the relevant principals, constraints, and simplifying assumptions are usually explicitly identified. The analytic framework can facilitate an a priori prediction of the extent of the model's generalizability.

Another potentially major advantage of the rational model is that it may be possible to derive an analytic solution if the model is not excessively complex. An analytic solution can be used, for example, to check the accuracy of numerical implementations. Despite the inherent elegance and power of a purely analytically derived mathematical model, correlational analysis of empiric or experimental data, however, is still an important aspect of rational model building, because it bridges gaps in the scientific knowledge of the extraordinarily complex cause and effect mechanisms linking environmental and physiologic variables with performance or medical effects.

This concludes the general overview and discussion of the history, nature, and applications of the various USARIEM stress-strain predictive models for soldier responses to cold, hot, and high altitude environments. The next section resumes the discussion of the FLDMED prototype that integrates these model into a unified military medical deployment staff support software tool for analysis, decision making, and training.

METHODS

The concept and general design scheme for FLDMED, the software prototype developed for this report, was delineated in a previously published technical note (Reardon, 1993). FLDMED is an implementation that demonstrates many, but by no means all, of the features elaborated in that design document. A subset of the functionality defined in the design document was selected for developing FLDMED. A high-level priority for this implementation was the demonstration of a gateway interface from which the user could navigate between interconnected altitude, cold, and heat strain prediction modules.

Sketches and diagrammatic storyboarding were utilized for exploring interface and menu design alternatives. This design approach was utilized to identify different options for efficient and visually effective appearances for the program component interfaces. The desired program module functionalities were then translated into hierarchical menu structures. This was done in parallel with the interface design. A character-based screen drawing utility was used to create the static visual components of the module interfaces. These predefined screen backgrounds were stored as individual 4,000-byte files. This technique allowed screen files to be efficiently loaded into screen display arrays at the appropriate locations in the program. Various data display functions overwrite the background screen with input and output data at the appropriate locations.

The inputs necessary to create functional altitude, cold, and heat modules were determined by the specific model selected for each module. The requisite input and output data elements for each of the models were identified. Input, output, and miscellaneous data elements for each of the three modules were then organized into data structure hierarchies (Appendix F). The primary advantages of such data structures were simplification of function interface specifications and the facilitation of enblock movement of large packets of related data to and from files.

The software code for the program FLDMED was implemented with the "C" computer language. The user interfaces were developed with the assistance of functions from a library of C interface and utility functions from Star Guidance Consulting, Inc. (Waterbury, Conn.). This library of C interface functions was bundled in an interface development software package entitled 'The Window Boss and Data Clerk" (version 5.17). The FLDMED program also utilized functions from a C function library from Young Software Engineering (Mill Valley, Calif.) entitled "Software Tools for C."

A large data model was selected as one of many different types of compiler options. This enabled the simultaneous accommodation, in computer memory, of the rather large input-output data structures and large code modules. The program's source code, however, was segmented into groups of functionally related computer files as part of the effort to implement modularity. Program segmentation into separate source files also made possible the use of run-time overlaying of executable-code. To do this, code overlay was specified as a compiler option.

Overlaying is a technique that allows software to automatically swap compiled code segments in and out of standard RAM memory from disc storage. This occurs as particular code segments are required, rather than having the entire program continuously loaded in RAM memory. Code segment overlays often permit larger data structures to be maintained in active memory than otherwise would be possible. An alternative method for accommodating large data structures is storing active data sets in disk files and reloading the data back into the program as needed. This technique, however, typically results in more frequent read-writes to the computer storage media. This may result in slower program execution due to repetitive and relatively slow file input-output operations.

Algorithms for the FLDMED numerical functions in the altitude, cold, and heat modules were obtained from internal and external sources. FLDMED's heat strain module utilizes the USARIEM heat stress algorithm. This model was discussed in a preceding section of this report. The equations for the core temperature portion of this algorithm are provided in Appendix B in the format of a MathCAD document. Not included are a related series of heart rate equations described in the 1973 references by Givoni and Goldman. USARIEM does not currently have an altitude physiology algorithm. For demonstration purposes, therefore, an altitude physiology algorithm from a master's degree thesis at the University of Vermont was utilized (Kessler, 1980). The equations in Kessler's thesis were rewritten and tested as a MathCAD (MathSoft, Inc., Cambridge, Mass.) document. This listing of equations is included in Appendix D. Kessler's altitude or hypoxia model was primarily motivated by a paper authored by his thesis advisor and renowned altitude physiologist, Charles Houston (1947). For this reason it will henceforth be referred to as the Kessler-Houston altitude model.

The author was diverted to other taskings before the lumped-parameter cold digit numeric algorithm could be implemented and inserted into the cold exposure module. However, the data structures and user interface designed into the cold stress module's interface are specifically intended to accommodate the USARIEM lumped-parameter cold digit model (Shitzer, et al., 1993). A recent technical report (Reardon, 1994) illustrates a software implementation of this cold digit model that utilizes a Windows (Microsoft Corp., Redmond, WA) graphical user interface. That software implementation of the USARIEM lumped-parameter cold digit model included an on-line hypertext cold weather military medicine reference (Burr, 1993). It was written in the Visual Basic programming language. The numeric algorithm in that program, however, can be easily converted into the equivalent C functions for subsequent inclusion into the presently incomplete cold exposure module. -

RESULTS

On start-up, the main, or gateway, component of the FLDMED computer program displays a camouflaged background screen. The altitude, cold, and heat modules are accessed from the main horizontal menu located at the top of the program's introductory screen. This user interface screen, therefore, can be conceptualized as the program's hub or gateway to the component altitude, cold, and heat stress modules. The altitude, cold, and heat stress modules in FLDMED were designed as mutually independent components. Note, however, that since altitude and cold occur simultaneously, in many circumstances their effects will actually be interrelated or correlated (e.g., see Chang, et al., 1989). The available physiologic models did not incorporate these types of complex interactions and, therefore, could not be included into this version of FLDMED.

The FLDMED.exe program is a single executable 463,036 byte file. Header and software code files were compiled using compiler options that permitted overlaying, or swapping, of related code segments. This allows the appropriate blocks of program instructions to be swapped between rapidly accessible working memory and higher capacity but lower access speed storage media when the user switches between the altitude, cold, and heat modules. Three header files-one each for altitude, cold, and heat-contain definitions of data structures and function declarations (Appendix F). The body, or contents, of the functions declared in the header files are elaborated in separate source files (Appendix G and H). Ten code source files defined the functions for the altitude module, three source files implemented the cold module functions, and eighteen source files implemented the heat module functions.

In addition to the executable file, FLDMED utilizes 175 separate small files (totaling 700,000 bytes) for displaying the contents of the on-line environmental medicine and physiology handbooks, help section text, and reference lists. To run properly, the composite program requires the FLDMED.exe and the 175 *.alt, *.cld, and *.scr text files. The total disk space required to transport and fully utilize the FLDMED program is 1,163,036 bytes (or approximately 1.12 megabytes).

Computational and output formatting and display functions for some of the cold module menu items have yet to be implemented. The reader is directed to the references describing the USARIEM lumped-parameter (Shitzer, et. al., 1993) a distributed-parameter cold digit model (Shitzer, et. al., 1994), and a whole body cold immersion model (Tikuisis, Gonzalez, and Pandolf,1988a and 1988b) for possible alternative implementations of the cold module's computational core. The data input interface, however, for the current version of FLDMED's cold module is designed to accommodate the lumped parameter cold digit model. Functions implementing the lumped-parameter cold digit model can therefore be easily inserted as the cold module's computational core in a future version of FLDMED.

FLDMED's functional heirarchy are summarized as menu trees in Figures 1-4. These figures depict the relationship of user menu elements for the main module and altitude, cold, and heat submodules. Figure 6 illustrates the front end, or initial menu, that appears after starting the main executable program file. From this menu, the user may select the altitude, cold, or heat stress submodule. Alternatively, the user may exit the program, thereby returning to the operating system.

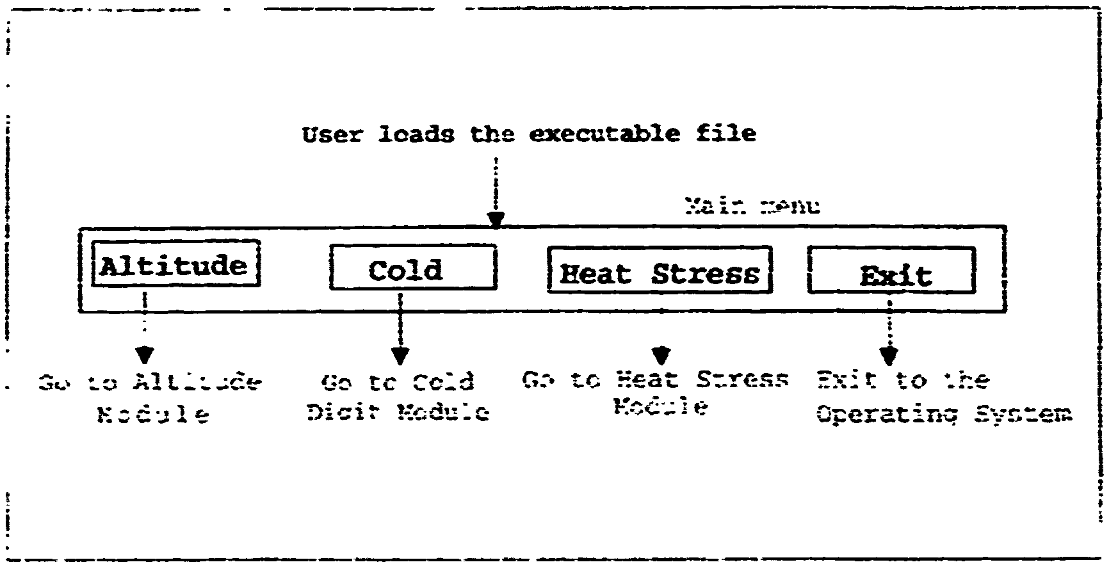


Figure 6: Main menu tree.

ALTITUDE MODULE

If the altitude module is selected from the menu in FLDMED's main menu, the introductory screen is cleared, and the altitude module's background screen is displayed along with the primary altitude bar menu located across the top of the screen. The options listed in the altitude module's pull-down top-bar submenus are illustrated in Figure 7. The user has direct control over input, output, and on-line reference for altitude medicine and physiology, as well as various other options. Upon initial entry into the altitude module, the program automatically loads a default profile. The principal feature of the input option is the capability for stepwise determination by the user of senario-specific ascent-descent profiles. An ascent-descent profile is dispayed as a graph as well as a tabulation of of movements to and from the specified altitudes

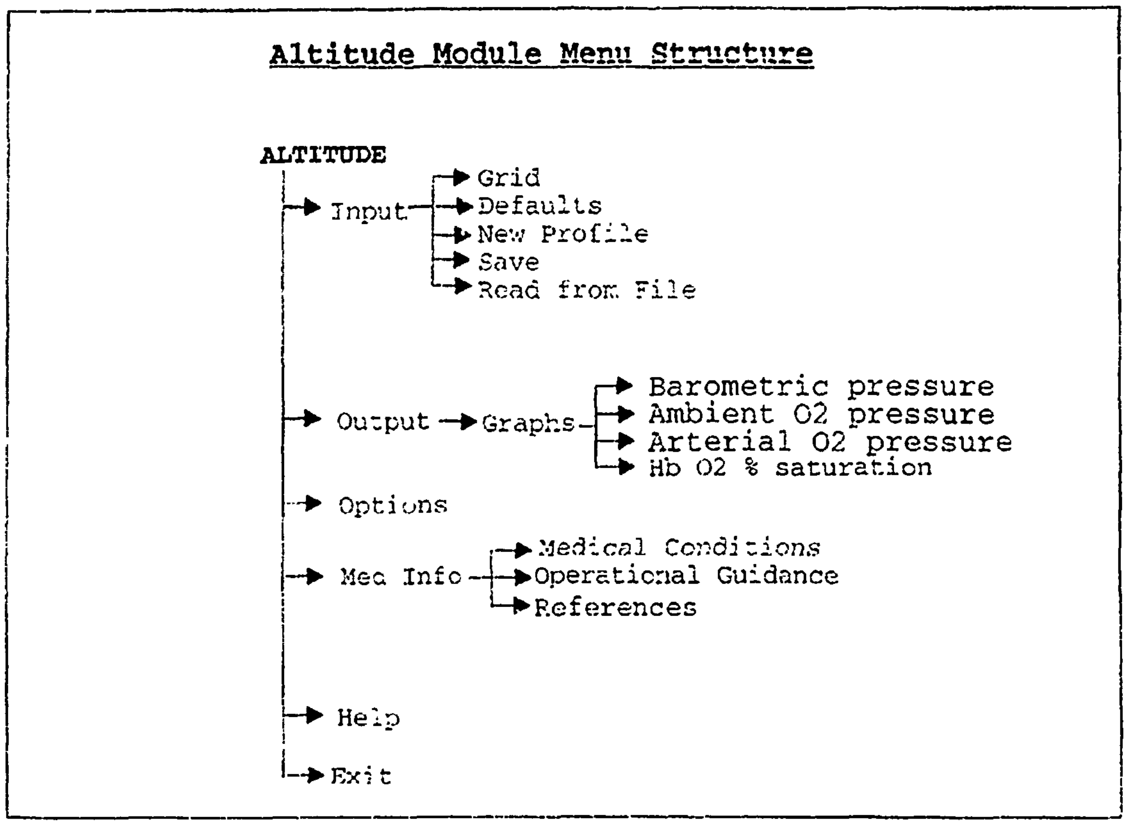


Figure 7: Altitude module menu tree.

In addition to the ascent-descent profile, other inputs for the altitude module are entered via an input grid for four sets of physiologic parameters (see the User's Guide in Appendix A for additional details). The physiologic input parameters are collated into two groups: hematologic and respiratory. Hematologic parameters include those for hemoglobin, hematocrit, bicarbonate, and plasma pH. Respiratory system parameters include those for respiratory ratio, inspired oxygen concentration, alveolar carbon dioxide pressure, and arteriolar-alveolar oxygen diffusion gradient.

The altitude module output currently is limited to graphic displays of barometric pressure, ambient oxygen (O_2) pressure, arterial O_2 pressure, as well as percent hemoglobin (Hb) O_2 saturation. Future enhancements could incorporate capabilities for generating detailed and summary listings of physiologic responses for each segment of the ascent-descent profile.

From the altitude module's top-bar menu the user may select the Med Info option. This provides access to on-line reviews of numerous topics in altitude medicine and physiology (Cymerman, 1994). Topics include high altitude pulmonary edema (HAPE), high altitude cerebral edema (HACE), possible altitude related complications associated with sickle cell trait, and potential thrombotic complications associated with altitude as reported in the medical literature. A synopsis of the cardiovascular, hematologic, and pulmonary adaptations to high altitude are also reviewed. Operational medical considerations for military deployment to high altitude regions are provided in a separate section. Reference lists from the USARIEM technical base, as well as from the civilian literature, are also provided.

COLD MODULE

The menu structure for the cold strain module is depicted in Figure 6. The first level menu provides interactive functionality and structural organization consistent with the altitude and heat stress module menus. A horizontal menu is located across the top portion of the cold module's main screen. Selecting any of the main options from the top-bar menu generates submenus of additional selectable options. The primary menu categories for the cold exposure module are inputs, output display options, medical on-line reference, help or user instructions, and an exit option that takes the user directly back to the main module.

Menu items bounded by the gray areas in Figure 8 indicate those cold module menu items that have not yet been implemented. The cold model input interface was designed to accommodate the USARIEM lumped-parameter cold digit model (Shitzer, 1993). An example of a Windows implementation of this lumped-parameter cold digit model can be found in Reardon,1994. The user interface structure for FLDMED's cold module was implemented in a manner consistent with the altitude and heat stress modules. For example, the physiologic inputs for the cold module can be entered into a datagrid either by row or column. Morphologic properties of the simulated digits can be collated into a separate input widow. A drop-down selection list containing different types of military gloves is functionally anal⁻gous to the heat module's uniform selection screen. Likewise output functions will be written to permit the generation of data grids, graphs, and data listings similar to those in the heat strain module. As indicated, many of the functions required to complete the cold module can be derived, with appropriate modifications, from the corresponding functions used in the heat stress module. The software development scheme did take advantage of software component reusability despite the fact that an object oriented approach was not utilized and objects have been recently touted as the primary mechanism for exploiting code reusability.

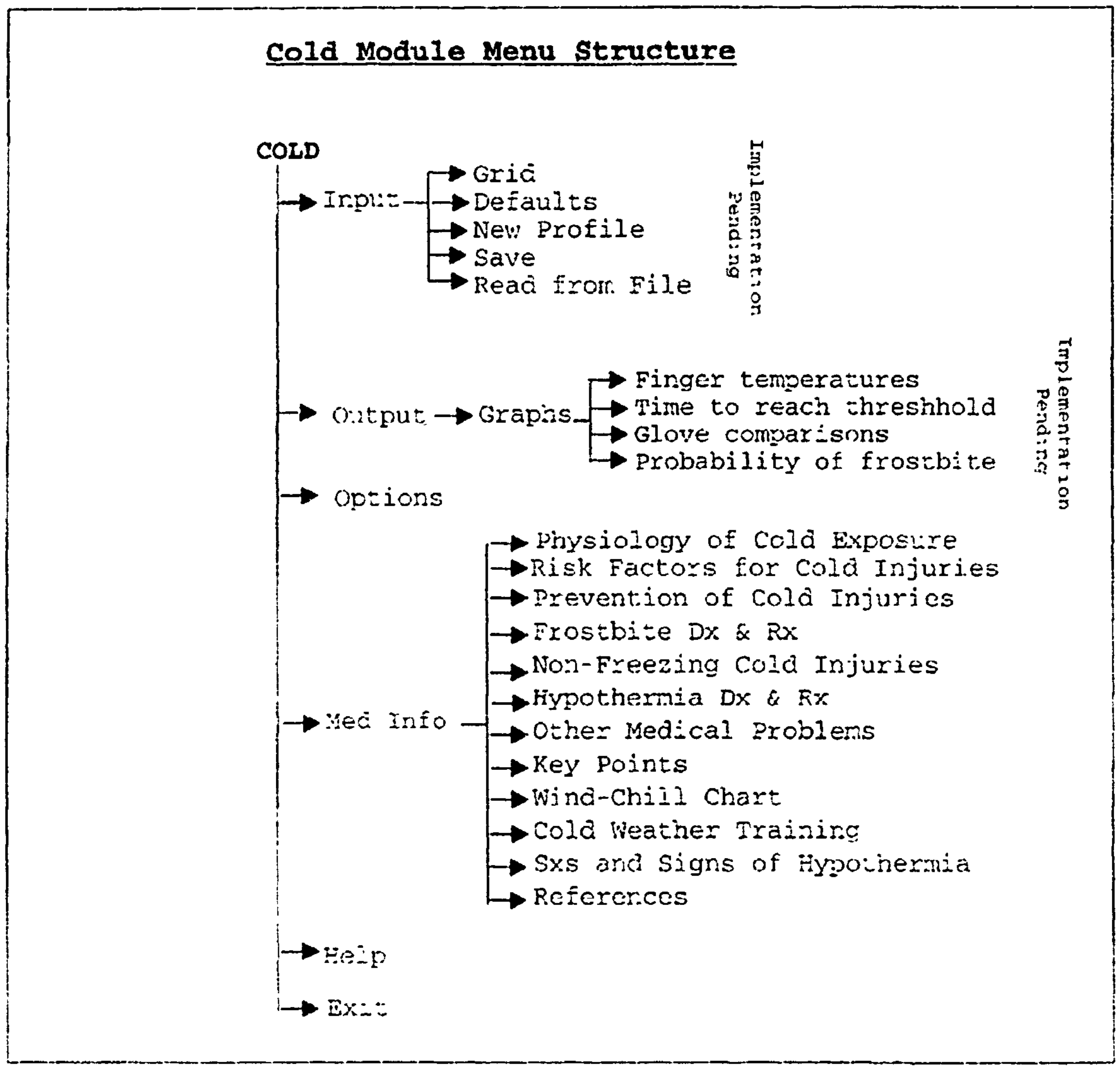


Figure 8: Cold module menu tree.

HEAT STRAIN MODULE

The menu structure for FLDMED's heat strain module is depicted in Figure 9. The first level menu in the heat strain module is, as with the other modules, a horizontal bar across the top of the screen. Selecting any of the main options in the top-bar menu generates submenus. The principal interaction categories for the menus are inputs, outputs, options for graphics display, medical on-line reference, help or instructions, and an exit option that takes the user back to the main module.

There are multiple data input options for the heat strain module. Data may be entered directly into data cells in the data grid. This can be done by row or column. Additionally, data may be entered as functionally related input parameter subsets. The multiple and redundant avenues for data entry provide flexibility and convenience to the user for the scenario building and editing process.

Popup windows are provided for inputting soldier-specific data such as height and weight, clothing parameters such as clo and permeability, specific soldier activities or, alternatively, an input window for entering marching related parameter for the automatic calculation of the metabolic rate (Pandolf, Givoni, and Goldman, 1977; Soule, Pandolf, and Goldman, 1978). Daily sodium intake is calculated based on user inputs for the type and frequency of ration consumption. Other input forms allow specification of solar conditions and associated radiant heat load (Breckenridge and Goldman, 1972), and terrain types that maps to the appropriate terrain coefficient. Terrain coefficients modify marching-related metabolic rates (Soule and Goldman, 1972).

The heat strain module provides for data persistency via numerous file handling capabilities. Previously constructed scenario files may be imported from a subwindow generated from an option in the Input menu. The user may save the most current data sets at any time by specifying a file name and adding, as a file header, an identifying comment for each set. A time and date stamp is also automatically appended in the data file's first line. When exiting the heat strain module, the complete scenario data set is automatically saved and, on reentering the module, the data is automatically reloaded so that work can be resumed without the need to reenter data. Alternatively, the user may revert to the default scenario data set by selecting the default input data option from the Input menu. One should note that if the data file saved from the most recent session is not in the same directory as the program, the default scenario set instead will be loaded by a call to a default data function within the program.

Output data for the heat strain module may also be saved in several ways. The user may use menu options to send output data to a local printer. If printing cannot be accomplished, the output data is stored in a text file, thereby making the data available for printing using some other mechanism after exiting the FLDMED program. In such a case, the program displays a message indicating that printing was unsuccessful, as well as the default name of the file where the output data was written in lieu of the printer.

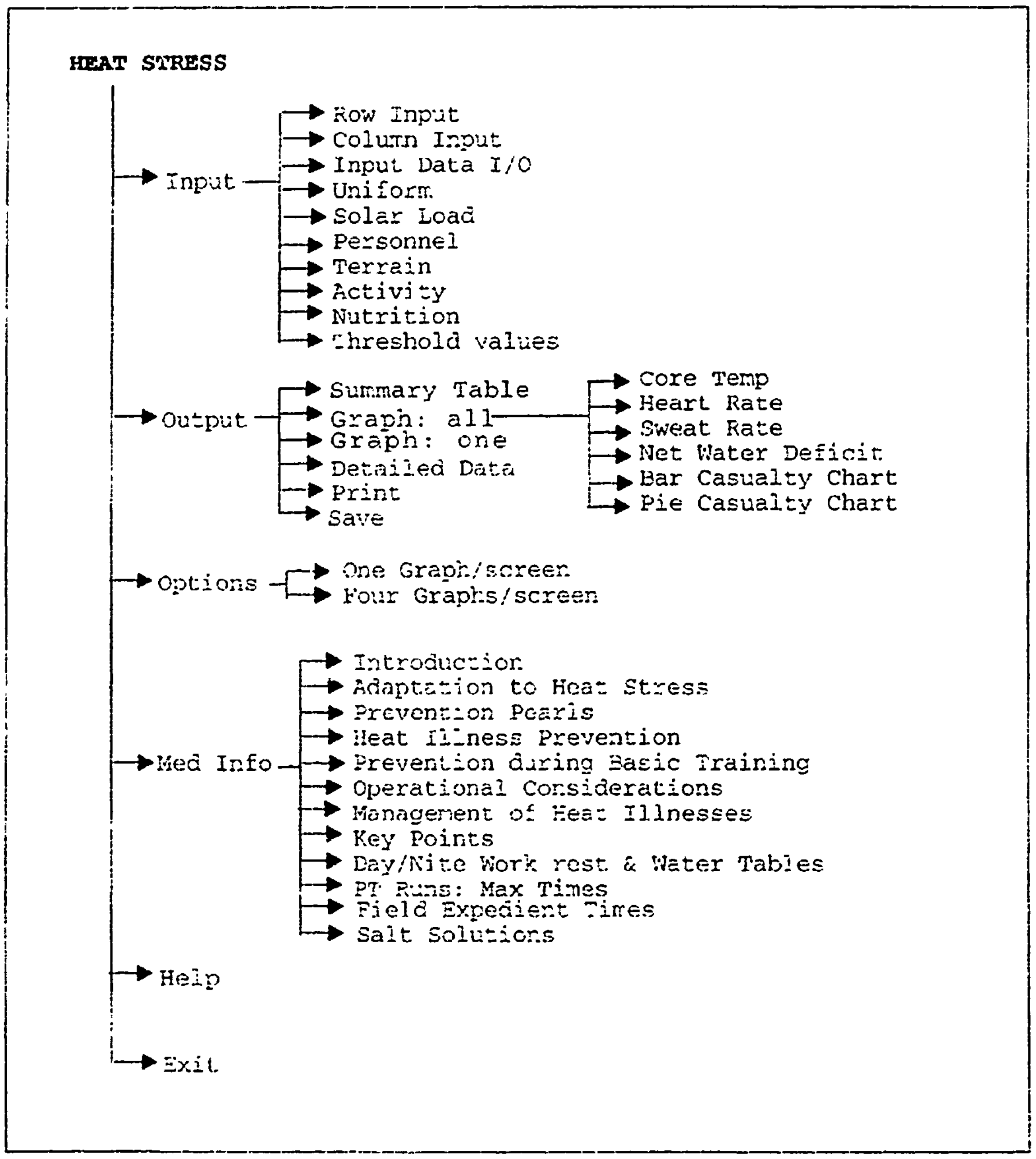


Figure 9: Heat strain module menu tree.

DISCUSSION

This project demonstrated the feasibility of consolidating numeric models for altitude, cold, and heat strain along with supporting environmental and operational military medical references The software program described in this report (FLDMED) provides an example of a unified and dynamic environmental medicine and physiology computer-based tool of potential use to military medical personnel for predeployment analysis, planning, and training.

A numerically based physiologic environmental strain model was selected for each of the three modules in the FLDMED program. The numeric algorithm for the altitude module (Kessler, 1980) is problematic in the sense that physiologic parameters are satic and output variables are altitude but not time dependent. For example, altitude acclimatization effects are not modeled. Additionally, this particular model does not include physiologic feedback control mechanisms or any ability to predict altitude illness rates. An alternative to the functionally limited Kessler-Houston altitude model, however, was not available. Looking towards the future, however, USARIEM's Altitude Physiology and Medicine Division has developed a broad base of research results in altitude physiology and medicine. Such data will be useful for creating a dynamic model capable of predicting a broad range of physiologic, performance, and medical responses to altitude exposure. This will eventually include the ability to simulate, or predict, acclimatization effects and time-dependent incidence rates of altitude illness as well as the extent of performance degradations (e.g., Kobrik, 1983).

FLDMED's heat strain module utilized the USARIEM Heat Strain Algorithm. An expanded interface was demonstrated that provides improved flexibility with regard to data entry and display modes (see the FLDMED User's Guidein Appendix A). This facilitates creating and analyzing scenarios spanning a wide range of environmental, soldier, metabolic, clothing, and other heat casualty risk factors. The extensive input modes were complemented with enhanced output capabilities. The heat strain module generates output in a variety of different tabular and graphical formats. The program can read and write its data structures to files and has features for printing detailed and summary output reports. The software also includes on-line references (Burr, 1991) for review of the physiology of heat stress and acclimatization changes, operationally oriented guidance for preventing heat stress casualties in deployment and training situations, and a review of the medical management of heat illnesses.

USARIEM scientists have done considerable research to evaluate the effectiveness of alternative microclimate cooling methods and devices for mitigating heat stress in soldiers operating military vehicles or conducting operations in NBC uniforms in hot weather (Pandolf, Gonzalez, and Sawka,1995). Such data are being used to expand the capabilities of the USARIEM Heat Strian Model is by including a capacity for predicting microclimate cooling effects (Gonzalez and Strochein, 1995). This involves specifying a heat extraction rate for the specific microclimate cooling device and incorporating it into the appropriate location(s) in the Heat Strain Algorithm.

The one-compartment, six-node, Kranning thermoregulatory model has been used in some circumstances in lieu of the USARIEM Heat Strain Model. The six-node model provides temperature predictions not only for the usual core temperature but also for four additional tissue compartments and a central blood compartment. Also, the six-node model provides capabilities for predicting and displaying the hemodynamic responses of heat stress exposure. To display these additional output capabilities, a more complex input-output interface than that used in the USARIEM Heat Strain Model is required.

The cold stress module's data structures and input-output interfaces were designed to accommodate USARIEM's lumped-parameter cold digit model. Another recent report illustrated how this model can be implemented (Reardon, 1994). Although the lumped-parameter algorithm is fairly straight forward to implement, Shitzer et al. (1994) developed a significantly more capable, but also more complicated, distributed-parameter cold extremity model. In that model, the passive component can be expressed mathematically as a multidimensional, partial differential equation. This distributed-parameter thermoregulatory model is nonhomogenous in structure and also has nonhomogenous boundary conditions. The nonhomogeneity of the mathematical representation makes for a complex and lengthy solution (spatially distributed temperatures as a function of time). Although considerable additional work will be required to finalize and validate this USARIEM distributed-parameter cold extremity model, it is expected to be available in the future as a higher resolution alternative to the lumped-parameter cold digit model. High resolution tissue temperature prediction capabilities could be utilized, for example, to generate synthetic thermograms. Conversely, the data from actual extremity thermograms could be utilized for parameter identification and validation of model-generated surface temperature dynamics.

As previously stated, the objective of the FLDMED software program was to demonstrate the feasibility ot integrating altitude, cold exposure, and heat strain models as well as various on-line environmental and operational medicine handbooks. This objective was accomplished. It must be admitted, however, that although the FLDMED program was not a trivial undertaking, collating models is often less tasking and resource intensive than constructing models de novo and taking them thorough the complex and arduous stages of verification and validation. For example, it took a generation of highly capable and experienced USARIEM scientists nearly two decades to develop, verify, and validate the USARIEM Heat Strain Model, and still, this model continues to require resources for updating and refinements so that it remains current with respect to new soldier uniforms and items of protective equipment, advances in microclimate cooling devices, and additional data from new heat stress studies.

The development of biomedical models that can survive the detailed scrutiny of those in the scientific and operational community with diverse backgrounds in medicine, physiology, military operations, research, biostatistics, mathematics, biophysics, computer science, or combinations of these high-level skills, essentially requires a dedicated model development and maintenance infrastructure. USARIEM has that infrastructure. It has (1) the facilities to obtain the basic data bases of biophysical properties of clothing, uniforms, and individual protective items; (2) tropic and arctic chambers and an immersion pool; (3) the specialized monitoring equipment and scientific expertise to accurately measure physiologic parameters, time constants, metabolic rates, and other variables for scenarios involving different types of soldier activities such as marching across different grades and terrain with a range of loads; (4) large altitude chambers for investigating the complex physiologic effects of hypobaric hypoxia and new pre-treatment strategies for the prevention of altitude illness.

USARIEM additionally has an extensive computerized data base of research study results (GeoCenters, 1992) and the statistical support required to professionally analyze the data for modeling purposes. Visiting professors periodically augment the USARIEM biomedical modeling staff, typically bringing with them analytic expertise and innovation in areas related to modeling complex physiologic processes.

It may be expensive and require considerable time and resources to develop, verify, and validate an environmental medicine model. Such efforts, however, result in models that can withstand the test of time and demonstrate robustness in a diverse array of initially unforeseen implementations and applications.

CONCLUSIONS

The objective of creating a software program to demonstrate the feasibility of merging altitude, cold, and heat strain models along with on-line medical handbooks was successful achieved. This is the first time that this has been accomplished at USARIEM. However, this prototype can only be considered as a proof of concept. It was an independent in-house effort. It remains for the more ambitious and talented to take this concept and rudimentary implementation and completely redo it, employing a user-centric approach. A structured development process would typically include a formal needs solicitation and assessment; software design, development, and maintenance process; state-of-the art software technologies; and fully validated models. Software standards (e.g., see Appendix E) can be used for guidance. Formal project management and control mechanisms should be implemented so that the product development effort occurs in managable phases with intervening review, analysis, and decision milestones. Additionally, an efficient process should be established for generating the supporting documentation required for in-house quality control and configuration management and to maintain compliance with external acquisition and accounting regulations.

It is commonly recognized that the prospective users should be involved in defining operational requirements and performance specifications for an incipient operational software product. It should be user driven from its inception. Final product design and development should only begin either in response to spontaneous direct requests for a product such as FLDMED from the military medical community external to USARIEM or secondary to an active and successful effort to market the concept after eliciting awareness of such a product's potential utility. In the latter case, the desired and operationally useful features and functionality of the environmental and operational medicine software-based decision and training aid should be actively solicited from potential users This can be done with job and workplace analysis, interviews, questionnaires, prototype demos, and other techniques. The potential users can thereby define and distinguish between the necessary and nice-to-have input and output data elements and functions, interface structures, responsivity, color schemes and other details. The program specifications and design should also be based largely on what the users' responsibilities and tasks are and how they anticipate that such a software tool can be of assistance to them. The software must satisfy the users', not the developer's, concepts of what the product should and should not be in terms of function and form (for a somewhat contrarian opinion, see Martin, 1995).

Storyboarding and rapid prototyping can be of use in helping the potential users formulate or articulate their ideas and concepts, as well as to demonstrate what will and will not be practical or possible. It is important that the potential users be assisted in clearly identifying all the services that they want the proposed software product to provide when operating under both routine and stressful conditions. The proposed product is not likely to be successful or used if it does not provide the high priority user requested services and functionality. Some examples of important generic services include the following: saving time; automating or reducing the time for completion of routine, boring, or difficult administrative tasks; imparting new or difficult-to-grasp knowledge in an efficient manner; automating quantitative analysis where this is required and too difficult or too complex to do mentally using simple calculations; assisting and assuring that all important elements are considered in operational or training plans; or providing a compact, portable reference source with a high-speed and easy-to-use search mechanism.

The user-identified services specified for a software product should also be analyzed for implicit secondary or supporting functional requirements. These can then be ranked according to user-oriented priorities. The requested software functionality in the highest priority levels are those that should be implemented before those having lower priority ratings (unless not technically feasible). If necessary, due to resource constraints, the lower priority functions can be implemented as enhancements in latter versions. After the hierarchy of user requirements has been determined, software performance specifications can be delineated, after which the high- and low-level design processes can proceed.

In conclusion, several specific recommendations are offered based on observations made during the development of the FLDMED prototype software:

- A repository of environmental physiology and medicine models implemented in a common computer programming language should be established by the biomedical modeling division. The selected computer language(s) should be stable into the foreseeable future, in so far as this can be predicted. These program modules should be implemented so that they are portable across the most prevalent operating systems for small and large computer systems. These biomedical simulation modules would provide numerically-based building blocks for the rapid development of new modeling applications. Additionally, libraries of interface functions or objects, designed to accomodate the input-output capabilities and limitations of the different models, could be developed. This would provide interface building blocks for rapid prototyping and testing of new application concepts. A software configuration management plan and data base should also be implemented to track, control, and synchronize the modification and documentation process for the different modules.

- Another recommendation is to proceed with the exploratory design and prototype development of a predictive altitude response model. Sufficient data exist to support the development of a prototype altitude model for predicting incidence rates of altitude illnesses and estimating performance decrements as functions of high altitude exposure. Initial versions of this type of model might only be useful for in-house concept development and defining gaps in the modeling data base where future experimentation is required. A preliminary nonoperational prototype altitude model could therefore serve to put the current body of knowledge of altitude physiology and medicine into a coherent framework and facilitate the identification of specific areas that need further research so that an adequate altitude model can be completed at some latter point in time. A validated operational altitude response model would obviously have many potential military applications. For example, such an algorithm could be used to provide unit commanders recommendations for optimal time-dependent ascent-descent profiles for minimizing incidence rates of altitude illnesses while simultaneously meeting the timeline of the deployment contingency.

- Exploring the development of complex interaction models for exposure to harsh environments may be useful and can be an additional method of enhancing currently available models. An interaction model might include the interaction of altitude associated hypoxia with simultaneous cold exposure. For example, military operations in high altitudes may be associated with increased risk of frostbite due not only to the typically cold, windy conditions of mountainous terrain, but also to the decreased vigilance caused by the depressed alertness and cognitive impairment from altitude associated hypoxia or incapacitation from the various types of altitude related medical conditions.

- USARIEM also has data bases to support inclusion of nutritional factors and injury predictions into current models for some applications. Present models could be expanded, therefore, to provide advice with regard to modulating dietary components for maximizing soldier performance in different types of environments and areas of operation (e.g., Thomas, et al., 1993). Military environmental medicine software tools such as FLDMED could also be enhanced by including predictive capabilities for estimating incidence rates for foot blisters and skeleto-muscular injuries for scenarios involving activities such as marching, climbing, lifting, or load carriage (e.g., Knapick, et al., 1992; Reynolds, et al., 1990; and Jones, 1983).

REFERENCES

American National Standards Institute (ANSI). American National Standard Reference for the Ada Programming Language, ANSI/MIL-STD-1815A-1983. American National Standards Institute, NY, NY, 1983.

Berlin, H.M., Stroschein, L.A. and Goldman, R.F. A computer program to predict energy cost, rectal temperature, and heart rate response to work, clothing, and environment. Aberdeen Proving Grounds, MD: Edgewood Arsenal Special Publication, ED-SP-75011, 1975.

Breckenridge, J.R. and Goldman, R.F. Human solar heat load. ASHRAE Transactions, 78:110-119, 1972.

Burr, R.E. Heat illness: a handbook for medical officers. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Note TN91-3, 1991.

Burr, R.E. Medical aspects of cold weather operations: a handbook for medical officers. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Note TN93-4, 1993.

Chang, S.K.W., Santee, W.R., Devine, J.A. and Gonzalez, R.R. The effect of hypobaric pressure on convective heat transfer. Natick, MA: U.S. Army Research Institute of Environmental Medicine. Technical Report T12-89, 1989.

Cymerman, A. and Rock, P.B. Medical problems in high mountain environments: a handbook for medical officers. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Note TN94-2, 1994.

DeCarlo, R.A. Linear Systems: A State Variable Approach with Numerical Implementation. Prentice Hall, Englewood Cliffs, NJ, 1989.

Doyle, C.D. (project editor). Computer Dictionary (Second Edition). Microsoft Press, Redmond, WA, 1994.

Endrusick, T.L. and Gonzalez, R.R. Physiological and psychological evaluation of women wearing protective clothing with varying moisture vapor transmission rates. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Research Protocol BBMD94006-AP034-H026, 1994.

Fanger, P.O. Thermal Comfort. Danish Technical Press, Copenhagen, 28-29, 1970.

GEO-Centers. P^2NBC^2 soldier performance database and modeling. Final Technical Report, Volume 1. Prepared by GEO-Centers, Inc., Newton Centre, MA, for U.S. Army Natick RD&E Center, Dec. 1992.

Givoni, B. and Goldman, R.F. Predicting rectal temperature response to work, environment, and clothing. Journal of Applied Physiology, 32(6): 812-822, 1972.

Givoni, B. and Goldman, R.F. Predicting effects of heat acclimation on heart rate and rectal temperature. Journal of Applied Physiology. 35(6): 875-879, 1973a.

Givoni, B. and Goldman, R.F. Predicting heart rate response to work, environment, and clothing. Journal of Applied Physiology, 34(2): 201-204, 1973b.

Gonzalez, R.R. Personal communications. Natick, MA: U.S. Army Research Institute of Environmental Medicine, 1995.

Gonzalez, R.R. and Cena K. Evaluation of vaporpermeation through garments during exercise. Journal of Applied Physiology, 58(3): 928-935, 1985.

Gonzalez, R.R., Levell, C.A., Stroschein, L.A., and Davio, D.J. Biophysics and heat strain modeling characteristics of advanced battledress overgarment prototypes (ABDO). Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T94-12, 1994.

Gonzalez, R.R., Levell, C.A., Strochein, L.A., Gonzalez, J.A. and Pandolf, K.B. Copper manikin and heat strain model evaluations of chemical protective ensembles for The Technical Cooperative Program (TTCP). Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T94-4, 1993.

Gonzalez, R.R. and Stroschein, L.A. Unclassified communications regarding work in progress. Natick, MA: U.S. Army Research Institute of Environmental Medicine, 1995.

Gourley, S. Infantry equipment update: next century soldier. Jane's Defence Weekly, 53(12): 31, 1995.

Harris. J.E. Fact sheet: integrated meteorological system (IMETS) AN/TMQ-40. White Sands Missile Range, NM: Army Research Lab. Battlefield Environment Directorate, 15 Oct. 1994.

Houston, C.S. The adaptations which produce acclimatization to oxygen lack. The Journal of Aviation Medicine, 18(3): 237-243, 1947.

Houston, C.. Cymerman, A. and Sutton, J.R. (Eds.) Operation Everest II: Biomedical Studies During a Simulated Ascent of Mt. Everest. Natick, MA: U.S. Army Research Institute of Environmental Medicine, December, 1991.

Institute of Electrical and Electronics Engineers. IEEE Standards Collection: Software Engineering. The Institute of Electrical and Electronics Engineers. NY, NY, 1994.

Jones, B.H. Overuse injuries of the lower extremities associated with marching, jogging and running. Military Medicine, 148: 783-787, 1983.

Kessler, G.C. A computer model of hypoxic man. Master of Science Thesis, The University of Vermont, May 1980.

Knapik, J., Reynolds, K., Staab, J., Vogel, J.A. and Jones, B. Injuries associated with strenuous road marching. Military Medicine, 157: 64-67, 1992.

Kobrick, J. Effects of hypoxia on the luminance threshold for target detection. Aviation, Space and Environmental Medicine, 54: 112-115, 1983.

Kobrick, J.L. and Johnson, R.F. Effects of hot and cold environments on military performance. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T7-92, 1992.

Kranning II, K.K. A computer simulation for predicting the time course of thermal and cardiovascular responses to various combinations of heat stress, clothing, and exercise. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T13-91, 1991.

Martin, J. Ignore your customer. Fortune, 131(8): 121-126, 1995.

Matthew, W.T., Stroschein, L.A., Blanchard, L.A. and Gonzalez, J. Technical testing of a prototype heat stress monitor: software verification and laboratory evaluations of sensor performance. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T9-93, 1993.

McNally, R.E., Stark, M.M. and Ellzy, D.T. Verification and usage of the Goldman-Givoni model: Predicting core temperature and casualty generation in thermally stressful environments. Science Applications International Corporation, Joppa, Maryland, 1990.

Naiditch, D. Rendezvous with ADA 95. John Wiley & Sons, Inc., NYC, NY, 1995

Pandolf, K.B., Givoni, B. and Goldman, R.F. Predicting energy expenditure with loads while standing or walking very slowly. Journal of Applied Physiology supplement: Respiratory, Environmental, and Exercise Physiology, 43(4): 577-581, 1977.

Pandolf, K.B., Stroschein, L.A., Drolet, L.L., Gonzalez, R.R. and Sawka, M.N. Prediction modeling of physiological responses and human performance in the heat. Computers in Biology and Medicine, 16: 319-329, 1986.

Pandolf, K.B., Gonzalez, J.A. and Sawka, M.N. An updated review: microclimate cooling of protective overgarments in the heat. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T95-7, 1995.

Reardon, M.J. Heat stress illness in a mechanized infantry brigade during simulated combat at Fort Irwin. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T94-14, 1990.

Reardon, M.J. Design concepts for an integrated environmental medicine workstation for prediction, simulation, and training. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Note TN94-1, 1993.

Reardon, M.J. Hypertext and multimedia for functional enhancement of USARIEM medical handbooks and biomedical simulation software. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T95-3, 1994.

Reynolds, K.L., Haszuba, J., Mello, R. and Patton, J. Prolonged treadmill load carriage: acute injuries and changes in foot anthropometry. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T11/90, 1990.

Saib, S. ADA: An Introduction. Holt, Rhinehart, and Winston, Inc., NYC, NY, 1985.

SAIC. P^2NBC^2 heat strain decision aid users guide, Version 2.1. Science Applications International Corporation, Joppa, Maryland, 1993.

Shapiro, Y., Pandolf, K.B., and Goldman, R.F. Predicting sweat loss response to exercise, environment and clothing. European Journal of Applied Physiology, 48: 83-96, 1982.

Shitzer, A., Stroschein, L.A., Gonzalez, A.A. and Pandolf, K.B. Lumped parameter finger tip model exhibiting cold induced vasodilatation. In: Advances in Bioheat and Mass Transfer Microscale Analysis of Thermal Injury Processes Instrumentation Modeling: Clinical Applications. R.B. Roemer, (Ed.). The American Society of Mechanical Engineers, 1993.

Shitzer, A., Stroschein, L.A., Santee, W.R., Gonzalez, R.R., and Pandolf, K.B. Quantification of lower bounds for endurance times in thermally insulated fingers and toes exposed to cold stress. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T18-90, 1990.

Shitzer, A., Stroschein, L.A., Vital, P., Gonzalez, R.R., and Pandolf, K.B. Numerical model of the thermal behavior of an extremity in a cold environment including counter-current heat exchange between blood vessels. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T94-10, 1994.

Simulation Technologies, Inc. Integrated unit simulation system (IUSS) software user's manual. Simulation Technologies, Inc., Dayton. OH, February, 1993.

Solomond, J.P. Software specifications and standards. Army RD&A, Jan-Feb., 1995.

Soule, R.G. and Goldman, R.F. Terrain coefficients for energy cost prediction. Journal of Applied Physiology, 32:706-708, 1972.

Soule, R.G., Pandolf, K.B. and Goldman, R.F. Energy expenditure of heavy load carriage. Ergonomics, 21:455-462, 1978.

Stolwijk, J.A. and Hardy, J.D. Control of body temperature. In: Handbook of Physiology, Reactions to Environmental Agents. American Physiological Society, Bethesda, MD: sect. 9, chap. 4, 45-67, 1977.

Strang, G. Linear Algebra and Its Applications. Harcourt Brace Javanovich, Inc.. San Diego, CA, 1988.

Stroschein, L.A. Personal communications. Natick, MA: U.S. Army Research Institute of Environmental Medicine, 1994.

Thomas, C.D., Baker-Fulco, C.J., Jones, T.E., et al. Nutritional guidance for military field operations in temperate and extreme environments. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Note TN93-8, 1993.

Tikuisis, P., Gonzalez, R.R. and Pandolf, K.B. Human thermoregulatory model for whole body immersion in water at 20 and 28°C. Natick, MA: U.S. Army Research Institute of Environmental Medicine, Technical Report T23-87, 1987.

Tikuisis, P., Gonzalez, R. R. and Pandolf, K. B. Prediction of human thermoregulatory responses and endurance time in water at 20 and 24°C. Aviation, Space, and Environmental Medicine, 59:742-8, 1988a.

Tikuisis, P., Gonzalez, R.R. and Pandolf, K.B. Thermoregulatory model for immersion of humans in cold water. Journal of Applied Physiology, 64(2): 719-727, 1988b.

U.S. Air Force, Air Weather Service. Integrated Meteorological System (IMETS). For Your Information. FYI #20, 1993.

Walker, P.V. Chain of errors link to Ranger tragedy. Army Times, 55(37): 3 and 18-19, 1995a.

Walker, P.V. Only a ranger understands: seekers of the tab face extraordinary stress, hardship. Army Times, 55(32): 14-16, 1995b.

APPENDIX A:

FLDMED USER'S GUIDE

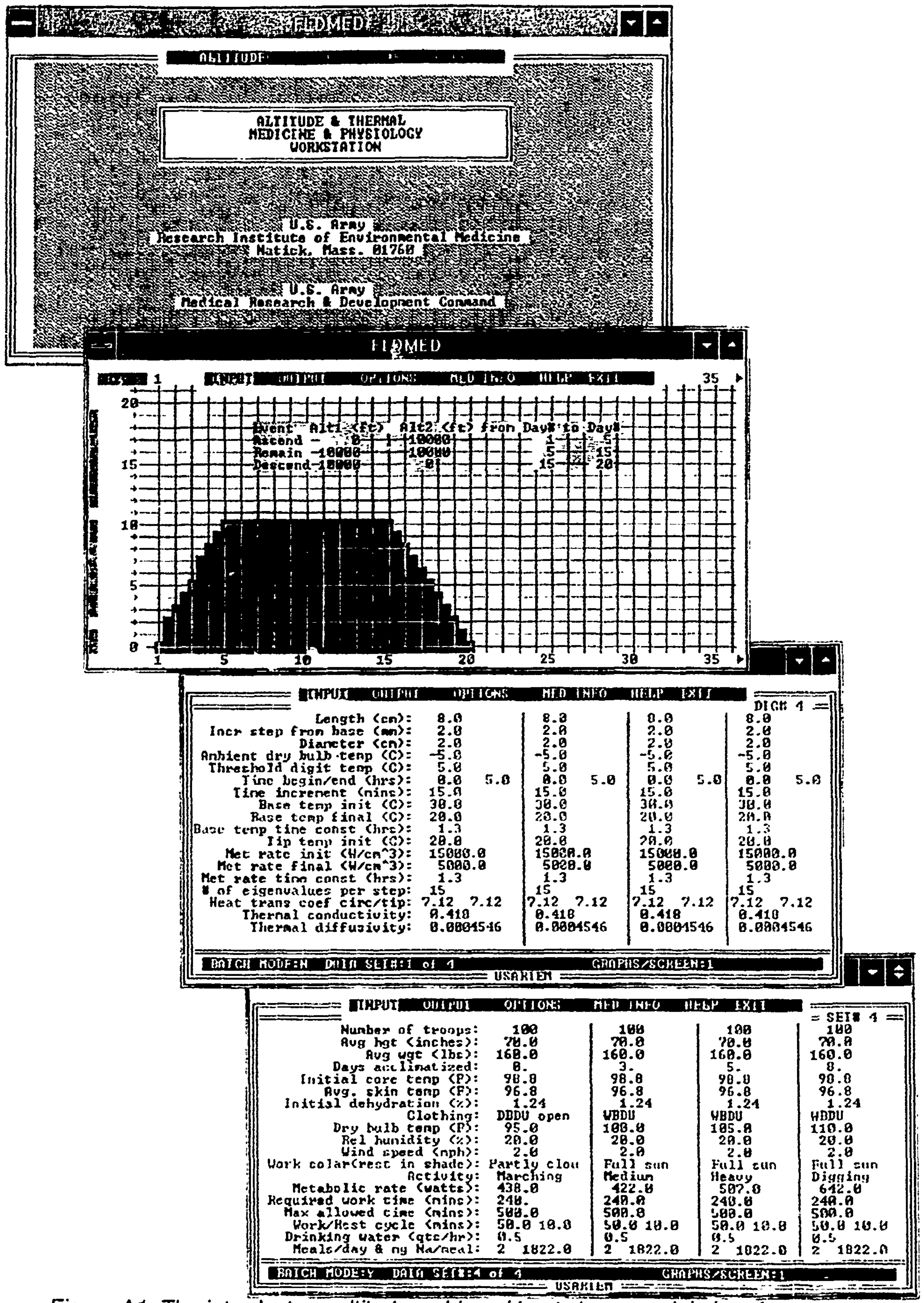
INTRODUCTION

The FLDMED program requires DOS (version 4.0 or latter) or an alternative operating systems that can run or emulate DOS. For example, FLDMED runs well in a DOS window within Microsoft Windows 3.1. The FLDMED program can be activated by running the fldmed.exe file. This can be done within DOS by typing "fldmed" (case insensitive) at the DOS prompt. Within the Windows operating system, double clicking on the fldmed.exe file name in the system file manager window will also activate the program in full screen mode. Simulaneously hitting the ALT and TAB keys will display the program in a reduced DOS window.

The FLDMED application is composed of four high-level components, an introductory module and one submodule each for altitude physiology, heat stress-strain, and cold stress. The introductory screen as well as the initial screen for each of the three submodules are depicted in Figure A1. The introductory screen has a camouflage pattern to emphasize the military nature of this application. The top-bar selection menu serves as the the gateway to the three environmental medicine and physiology submodules and establishes a hub for navigating between them.

The pictures of the FLDMED application in this user manual are screen captures of the actual FLDMED program run as an application in a DOS window within the Microsoft Windows 3.1 operating system. Although the figures in this user manual show FLDMED screens in greyscale, in actuality, FLDMED uses colorized text extensively for highlighting important items and provides an efficient mechanism for visually grouping and segregating functionally related material or options as well as directing the user's attention to important messages.

After initiating the program by typing "fldmed" at the DOS command line prompt, the camouflaged introductory user interface screen appears (top of Figure A1). Any key will then cause the main selection top-bar menu to appear. This top-bar menu provides the following choices: Altitude, Heat, Cold, and Exit. The user scrolls to the desired choice and hits the Return key to activate it. The top-bar menu format, as well as the overall interface layout scheme (look and feel) and use of color schemes, were features designed to be uniform across the submodules.



```
ALTITUDE & THERMAL
MEDICINE & PHYSIOLOGY
WORKSTATION

U.S. Army
Research Institute of Environmental Medicine
Natick, Mass. 01760

U.S. Army
Medical Research & Development Command
```

```
Event   Alt1 (ft)  Alt2 (ft) from Day# to Day#
Ascend       0       10000         1       5
Remain   10000       10000         5      15
Descend  10000           0        15      20
```

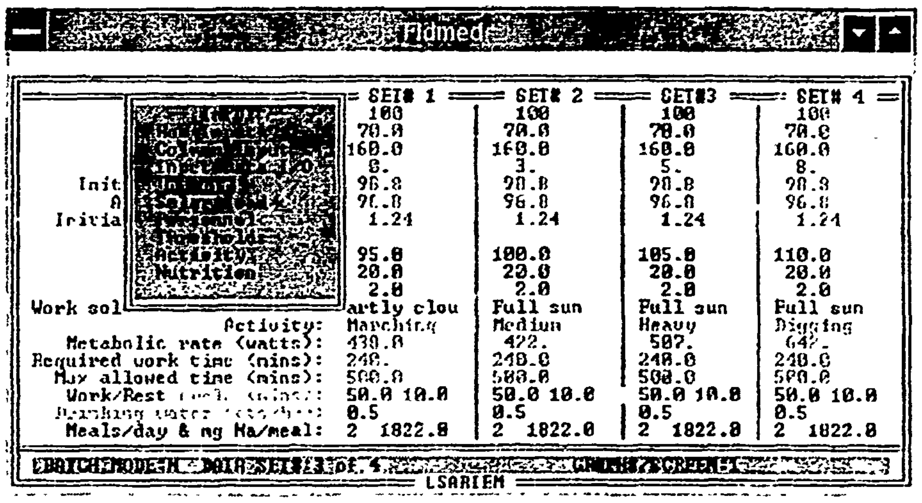
```
                                                              DIG# 4
           Length (cm):  8.0        | 8.0        | 8.0        | 8.0
  Incr step from base (mm):  2.0    | 2.0        | 2.0        | 2.0
           Diameter (cm):  2.0      | 2.0        | 2.0        | 2.0
 Ambient dry bulb temp (C):  -5.0   | -5.0       | -5.0       | -5.0
 Threshold digit temp (C):  5.0     | 5.0        | 5.0        | 5.0
    Time begin/end (hrs):  0.0  5.0 | 0.0   5.0  | 0.0   5.0  | 0.0   5.0
    Time increment (mins):  15.0    | 15.0       | 15.0       | 15.0
       Base temp init (C):  30.0    | 30.0       | 30.0       | 30.0
      Base temp final (C):  20.0    | 20.0       | 20.0       | 20.0
Base temp time const (hrs):  1.3    | 1.3        | 1.3        | 1.3
        Tip temp init (C):  20.0    | 20.0       | 20.0       | 20.0
   Met rate init (W/cm^3):  15000.0 | 15000.0    | 15000.0    | 15000.0
  Met rate final (W/cm^3):  5000.0  | 5000.0     | 5000.0     | 5000.0
 Met rate time const (hrs):  1.3    | 1.3        | 1.3        | 1.3
 # of eigenvalues per step:  15     | 15         | 15         | 15
  Heat trans coef circ/tip: 7.12  7.12 | 7.12  7.12 | 7.12  7.12 | 7.12  7.12
     Thermal conductivity:  0.418   | 0.418      | 0.418      | 0.410
      Thermal diffusivity:  0.0004546 | 0.0004546 | 0.0004546 | 0.0004546

 BATCH MODE:N  DATA SET#:1 of 4                    GRAPHS/SCREEN:1
                               USARIEM
```

```
                                                              = SET# 4 =
         Number of troops:   100       | 100        | 100        | 100
       Avg hgt (inches):   70.0        | 70.0       | 70.0       | 70.0
          Avg wgt (lbs):  160.0        | 160.0      | 160.0      | 160.0
        Days acclimatized:   0.        | 3.         | 5.         | 8.
    Initial core temp (F):   98.8      | 98.8       | 98.0       | 98.8
      Avg. skin temp (F):   96.8       | 96.8       | 96.8       | 96.8
  Initial dehydration (%):    1.24     | 1.24       | 1.24       | 1.24
                Clothing:  DBDU open   | WBDU       | WBDU       | WBDU
       Dry bulb temp (F):   95.0       | 100.0      | 105.0      | 110.0
        Rel humidity (%):   20.0       | 20.0       | 20.0       | 20.0
        Wind speed (mph):    2.0       | 2.0        | 2.0        | 2.0
Work solar(rest in shade): Partly clou | Full sun   | Full sun   | Full sun
                Activity:  Marching    | Medium     | Heavy      | Digging
   Metabolic rate (watts):  438.0      | 422.0      | 507.0      | 642.0
 Required work time (mins):  240.      | 240.0      | 240.0      | 240.0
   Max allowed time (mins):  500.0     | 500.0      | 500.0      | 500.0
  Work/Rest cycle (mins):  50.0 10.0   | 50.0 10.0  | 50.0 10.0  | 50.0 10.0
  Drinking water (qts/hr):  0.5        | 0.5        | 0.5        | 0.5
  Meals/day & mg Na/meal:  2  1822.0   | 2  1822.0  | 2  1822.0  | 2  1822.0

 BATCH MODE:Y  DATA SET#:4 of 4                    GRAPHS/SCREEN:1
                               USARIEM
```

Figure A1: The introductory, altitude, cold, and heat stress module interface screens.

HEAT STRAIN MODULE

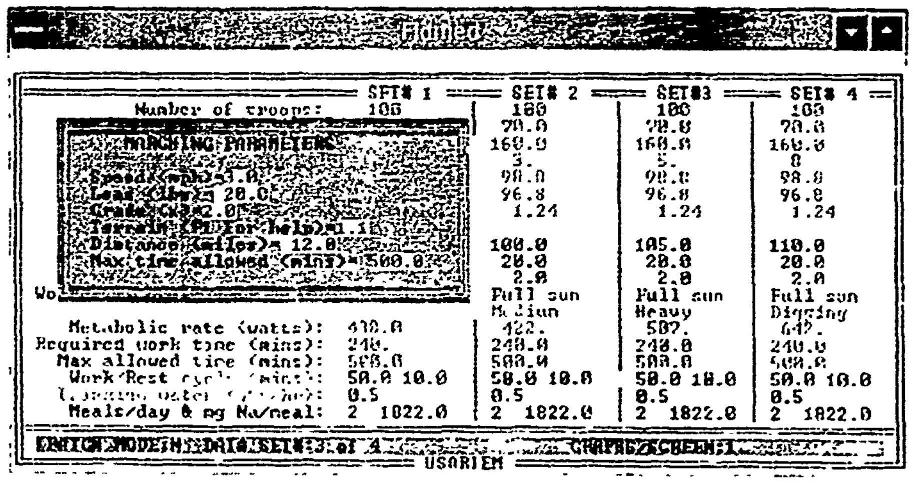
The heat strain module can be accessed by selecting the HEAT option from the top-bar menu in FLDMED's main screen. The heat strain module also has its own top-bar menu with selections for inputs, outputs, options, medical information, help and exiting. If the user selects INPUT from the heat module's top-bar menu, a popup submenu of alternative input mode options are presented as depicted in Figure A2 below.



```
                  = SET# 1 ===== SET# 2 ===== SET#3 ====== SET# 4 =
                     100       |  100       |  100       |  100
                     70.0      |  70.0      |  70.0      |  70.0
                    160.0      | 160.0      | 160.0      | 160.0
                     8.        |  3.        |  5.        |  8.
   Init              96.8      |  98.8      |  98.8      |  98.8
      A              96.8      |  96.8      |  96.0      |  96.8
 Initia               1.24     |   1.24     |   1.24     |   1.24

                     95.0      | 100.0      | 105.0      | 110.0
                     20.0      |  20.0      |  20.0      |  20.0
                      2.0      |   2.0      |   2.0      |   2.0
Work sol            artly clou | Full sun   | Full sun   | Full sun
              Activity:  Marching   | Medium     | Heavy      | Digging
  Metabolic rate (watts):  430.0      |  422.        |  507.        |  642.
Required work time (mins):  240.       | 240.0      | 240.0      | 240.0
  Max allowed time (mins):  500.0      | 500.0      | 500.0      | 500.0
  Work/Rest cycle (mins):  50.0 10.0  | 50.0 10.0  | 50.0 10.0  | 50.0 10.0
Drinking water [illegible]:  0.5        | 0.5        | 0.5        | 0.5
  Meals/day & mg Na/meal:  2  1822.0  | 2  1822.0  | 2  1822.0  | 2  1822.0
BATCH MODE:N  DATA SET#:3 of 4        GRAPHS/SCREEN:1
                          LSARIEM
```

Figure A2: Heat strain input options.

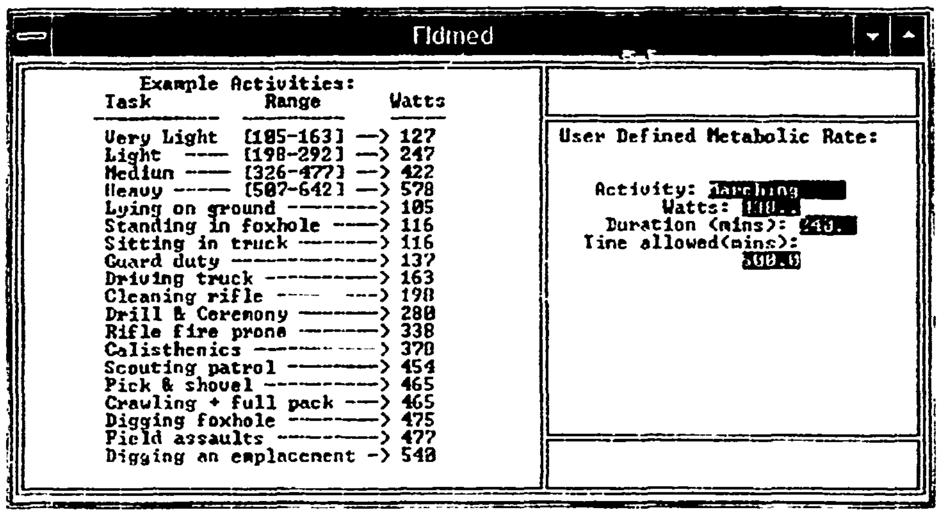
Several methods are provided in lower-level submenus for selectively changing data in the input grid from default or previous session values. Input data items can be entered or modified by rows or columns (data sets). Related groups of data elements can also be changed by selecting from among input functional groupings as enumerated in the Input menu selection list. For example, selecting Activity generates a submenu from which the user can select Marching, Activity List, or User Defined activity. If the Marching input option is selected, the input window shown in Figure A3 is displayed.



```
                  ====== SET# 1 ===== SET# 2 ===== SET#3 ====== SET# 4 =
        Number of troops:   100      |  100       |  100       |  100
                                     |  70.0      |  70.0      |  70.0
     MARCHING PARAMETERS             | 160.0      | 160.0      | 160.0
                                     |  3.        |  5.        |  8
   Speed (mph)= 3.0                  |  98.0      |  98.8      |  98.8
   Load (lbs)= 20.0                  |  96.8      |  96.8      |  96.8
   Grade (%)= 2.0                    |   1.24     |   1.24     |   1.24
   Terrain (F1 for help)= 1.1        |            |            |
   Distance (miles)= 12.0            | 100.0      | 105.0      | 110.0
   Max time allowed (mins)= 500.0    |  20.0      |  20.0      |  20.0
                                     |   2.0      |   2.0      |   2.0
Wo                                   | Full sun   | Full sun   | Full sun
                                     | Medium     | Heavy      | Digging
  Metabolic rate (watts):  430.0     |  422.      |  507.      |  642.
Required work time (mins):  240.     | 240.0      | 240.0      | 240.0
  Max allowed time (mins):  500.0    | 500.0      | 500.0      | 500.0
  Work/Rest cycle (mins):  50.0 10.0 | 50.0 10.0  | 50.0 10.0  | 50.0 10.0
Drinking water [illegible]:  0.5     | 0.5        | 0.5        | 0.5
  Meals/day & mg Na/meal:  2  1822.0 | 2  1822.0  | 2  1822.0  | 2  1822.0
BATCH MODE:N  DATA SET#:3 of 4        GRAPHS/SCREEN:1
                          USARIEM
```

Figure A3: Inputs for marching parameters.

The marching data entry window allows the user to input the data required to calculate the associated metabolic rates (e.g., see Pandolf, et al., 1977). On exiting the Marching data input window, the metabolic rate is automatically calculated and entered into the appropriate input grid cells. Alternatively, the user may enter activity descriptors and metabolic rates directly into the grid cells or the user may select the Activity list option as shown in Figure A4. The metabolic rate table in the user-defined activity entry screen provides a convenient reference for entering the metabolic rates associated with common soldier tasks or activities. Data entered in the activity subform are automatically entered into their proper locations in the input datagrid on exiting to the main heat stress-strain screen.

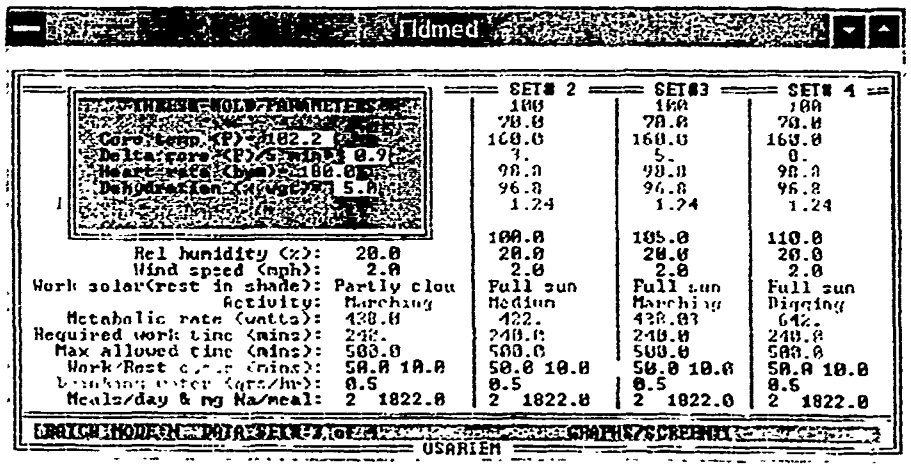


```
   Example Activities:
Task              Range        Watts

Very Light   [105-163] --> 127
Light  ----  [198-292] --> 247
Medium ----  [326-477] --> 422
Heavy  ----  [507-642] --> 578
Lying on ground --------> 105
Standing in foxhole ----> 116
Sitting in truck -------> 116
Guard duty -------------> 137
Driving truck ----------> 163
Cleaning rifle ----  ---> 190
Drill & Ceremony -------> 280
Rifle fire prone -------> 338
Calisthenics -----------> 370
Scouting patrol --------> 454
Pick & shovel ----------> 465
Crawling + full pack ---> 465
Digging foxhole --------> 475
Field assaults ---------> 477
Digging an emplacement -> 540
```

```
User Defined Metabolic Rate:

      Activity: Marching
         Watts: 430.
  Duration (mins): 240.
 Time allowed(mins):
              500.0
```

Figure A4: Activity list and metabolic rate input window.

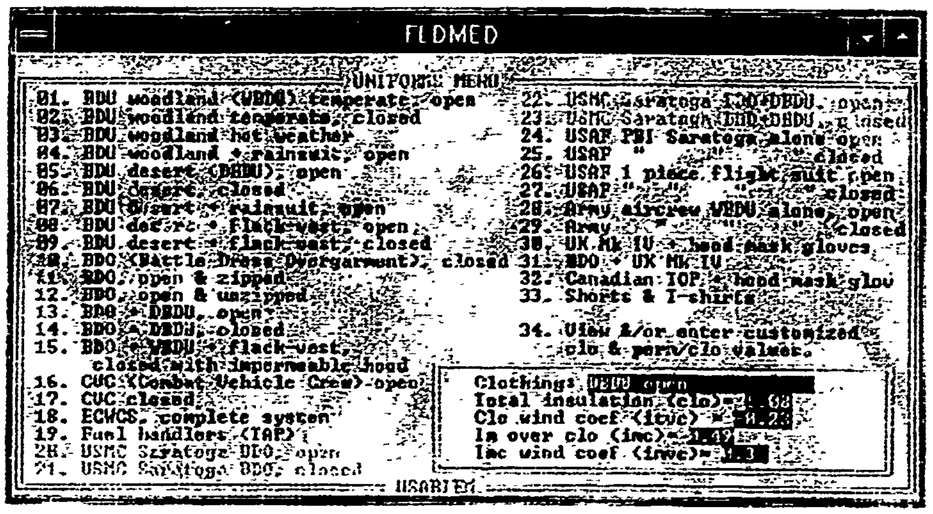
The heat strain module's input popup menu contains an option for specifying thresholds (Figure A5). These thresholds define the values of core temperature, rate of temperature rise, heart rate, and level of dehydration above which output listings for core temperature, heart rate and dehydration will be prefixed with special characters for easy identification.



```
THRESHOLD PARAMETERS
Core temp. (F)=102.2
Delta core (F)/5 min= 0.9
Heart rate (bpm)=180.0
Dehydration (%(wgt)= 5.0

                                        SET# 2        SET#3         SET# 4
                                        100           100           100
                                        70.0          70.0          70.0
                                        160.0         160.0         160.0
                                        3.            5.            0.
                                        98.0          98.0          98.0
                                        96.8          96.8          96.8
                                        1.24          1.24          1.24

                                        100.0         105.0         110.0
            Rel humidity (%):  20.0     20.0          20.0          20.0
            Wind speed (mph):   2.0      2.0           2.0           2.0
Work solar(rest in shade): Partly clou  Full sun      Full sun      Full sun
                   Activity: Marching   Medium        Marching      Digging
    Metabolic rate (watts): 430.0       422.          430.03        642.
 Required work time (mins): 240.        240.0         240.0         240.0
   Max allowed time (mins): 500.0       500.0         500.0         500.0
    Work/Rest cycle (mins): 50.0 10.0   50.0 10.0     50.0 10.0     50.0 10.0
     Drinking water (qts/hr): 0.5       0.5           0.5           0.5
   Meals/day & mg Na/meal: 2  1822.0    2  1822.0     2  1822.0     2  1822.0
```

Figure A5: Inputing thresholds for tagging output data.

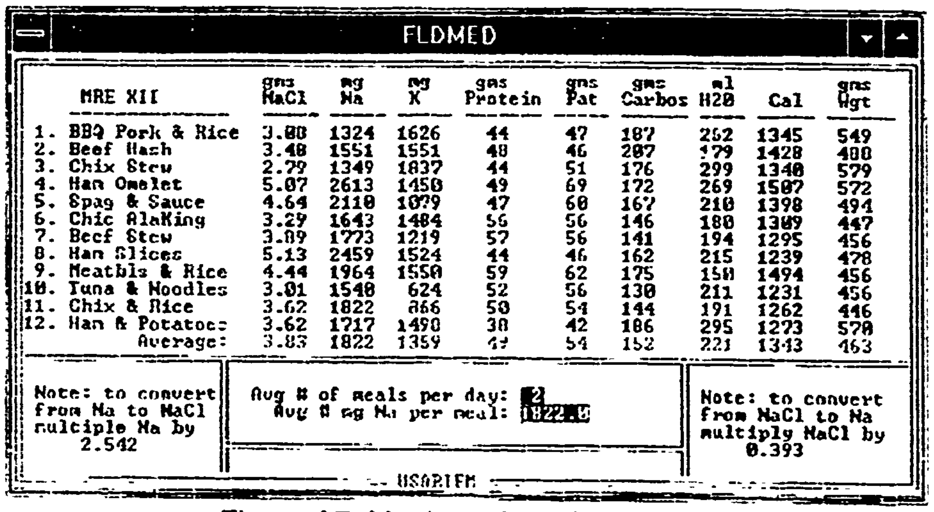
The type of uniform and solar load cannot be entered directly into the grid cells, but must be entered via subsidiary popup data entry screens or windows (Figure A6). Solar condition is entered one set at a time by selecting from Clear, Partly Cloudy, Cloudy, or Indoors from the solar condition menu (Breckenri[illegible], 1972). Uniforms are selected from the Uniform option in the Input listing (Gonzalez, et al., 1993 and 1994).



```
FLDMED
                              UNIFORMS MENU
01. BDU woodland (WBDU) temperate, open     22. USMC Saratoga IOO+DBDU, open
02. BDU woodland temperate, closed          23. USMC Saratoga IOO+DBDU, closed
03. BDU woodland hot weather                24. USAF PBI Saratoga alone open
04. BDU woodland + rainsuit, open           25. USAF  "                 closed
05. BDU desert (DBDU), open                 26. USAF 1 piece flight suit open
06. BDU desert, closed                      27. USAF  "        "       closed
07. BDU desert + rainsuit, open             28. Army aircrew WBDU alone, open
08. BDU desert + flack vest, open           29. Army   "       "        closed
09. BDU desert + flack vest, closed         30. UK Mk IV + hood mask gloves
10. BDO (Battle Dress Overgarment), closed  31. BDO + UK Mk IV
11. BDO, open & zipped                      32. Canadian IOP + hood mask glov
12. BDO, open & unzipped                    33. Shorts & T-shirts
13. BDO + DBDU, open
14. BDO + DBDU, closed                      34. View &/or enter customized
15. BDO + WBDU + flack vest,                    clo & perm/clo values.
     closed with impermeable hood
16. CVC (Combat Vehicle Crew) open       Clothing: DBDU open
17. CVC closed                           Total insulation (clo)=[illegible].08
18. ECWCS, complete system               Clo wind coef (icvc) = -0.28
19. Fuel handlers (TAP)                  Im over clo (imc)= [illegible]
20. USMC Saratoga BDO, open              Imc wind coef (imvc)= [illegible]
21. USMC Saratoga BDO, closed
                                  USARIEM
```

Figure A6 Uniform selection.

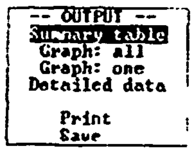
Nutritional factors are an additional input option within the heat module's Input menu. Figure A7 below is the nutrition input screen. It allows the user to specify the number of field rations consumed per day by the soldiers as well as the grams of sodium per ration. From this information the heat module calculates daily salt intake. As part of the output, this salt intake is compared to the predicted amount of salt lost during sweating. A time-dependent and cumulative body sodium, or salt, balance can therefore be calculated and displayed graphically or in the output grid.



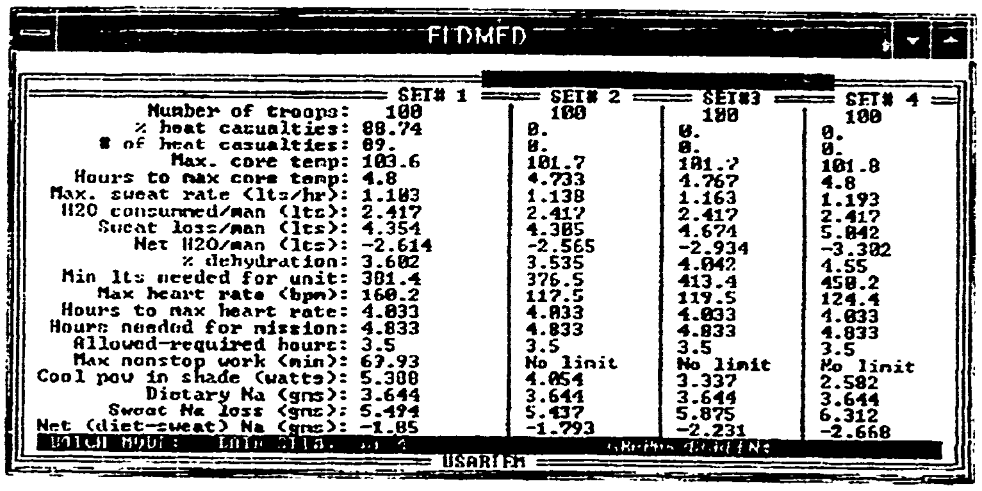
```
FLDMED
                          gms    mg    mg     gms    gms   gms    ml           gms
    MRE XII               NaCl   Na    K    Protein  Fat  Carbos  H20   Cal    Wgt
    ------------------    ----  ----  ----  -------  ---  ------  ---  ----    ---
 1. BBQ Pork & Rice       3.00  1324  1626    44      47   187    262  1345    549
 2. Beef Hash             3.48  1551  1551    48      46   207    179  1428    400
 3. Chix Stew             2.79  1349  1837    44      51   176    299  1340    579
 4. Ham Omelet            5.07  2613  1450    49      69   172    269  1507    572
 5. Spag & Sauce          4.64  2110  1079    47      60   167    210  1398    494
 6. Chic AlaKing          3.29  1643  1484    56      56   146    180  1309    447
 7. Beef Stew             3.09  1773  1219    57      56   141    194  1295    456
 8. Ham Slices            5.13  2459  1524    44      46   162    215  1239    478
 9. Meatbls & Rice        4.44  1964  1550    59      62   175    150  1494    456
10. Tuna & Noodles        3.01  1540   624    52      56   130    211  1231    456
11. Chix & Rice           3.62  1822   866    50      54   144    191  1262    446
12. Ham & Potatoes        3.62  1717  1490    38      42   186    295  1273    570
              Average:    3.83  1822  1359    49      54   152    221  1343    463

Note: to convert     Avg # of meals per day: 2          Note: to convert
from Na to NaCl       Avg # mg Na per meal: 1822.0      from NaCl to Na
multiple Na by                                           multiply NaCl by
    2.542                                                    0.393
                                  USARIEM
```

Figure A7: Meals and salt intake.

After input parameters are specified in the heat module input data grid, the user may select from a variety of different output options. The output options in the heat strain module are accessed from Output on the top-bar horizontal menu. The pick list for output format options appears in the following popup menu:



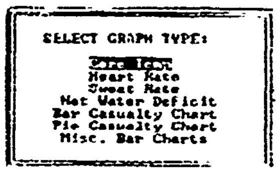
If the Output Grid option is selected, Figure A8 appears in the display window. This presents a tabular summary of the outputs for each of the input data sets. Pressing any key then returns the user to the screen with the input data grid.



```
                                      FLDMED
                          SET# 1      SET# 2      SET#3       SET# 4
        Number of troops:  100         100         100         100
        % heat casualties: 88.74       0.          0.          0.
      # of heat casualties: 89.        0.          0.          0.
            Max. core temp: 103.6      101.7       101.7       101.8
   Hours to max core temp: 4.8         4.733       4.767       4.8
 Max. sweat rate (lts/hr): 1.103       1.138       1.163       1.193
 H2O consumed/man (lts): 2.417         2.417       2.417       2.417
     Sweat loss/man (lts): 4.354       4.385       4.674       5.042
        Net H2O/man (lts): -2.614      -2.565      -2.934      -3.302
            % dehydration: 3.602       3.535       4.042       4.55
  Min lts needed for unit: 301.4       376.5       413.4       458.2
     Max heart rate (bpm): 160.2       117.5       119.5       124.4
   Hours to max heart rate: 4.833      4.833       4.833       4.833
 Hours needed for mission: 4.833       4.833       4.833       4.833
    Allowed-required hours: 3.5        3.5         3.5         3.5
    Max nonstop work (min): 67.93      No limit    No limit    No limit
Cool pow in shade (watts): 5.308       4.054       3.337       2.582
        Dietary Na (gms): 3.644        3.644       3.644       3.644
      Sweat Na loss (gms): 5.494       5.437       5.875       6.312
Net (diet-sweat) Na (gms): -1.85       -1.793      -2.231      -2.668
                                   USARIEM
```

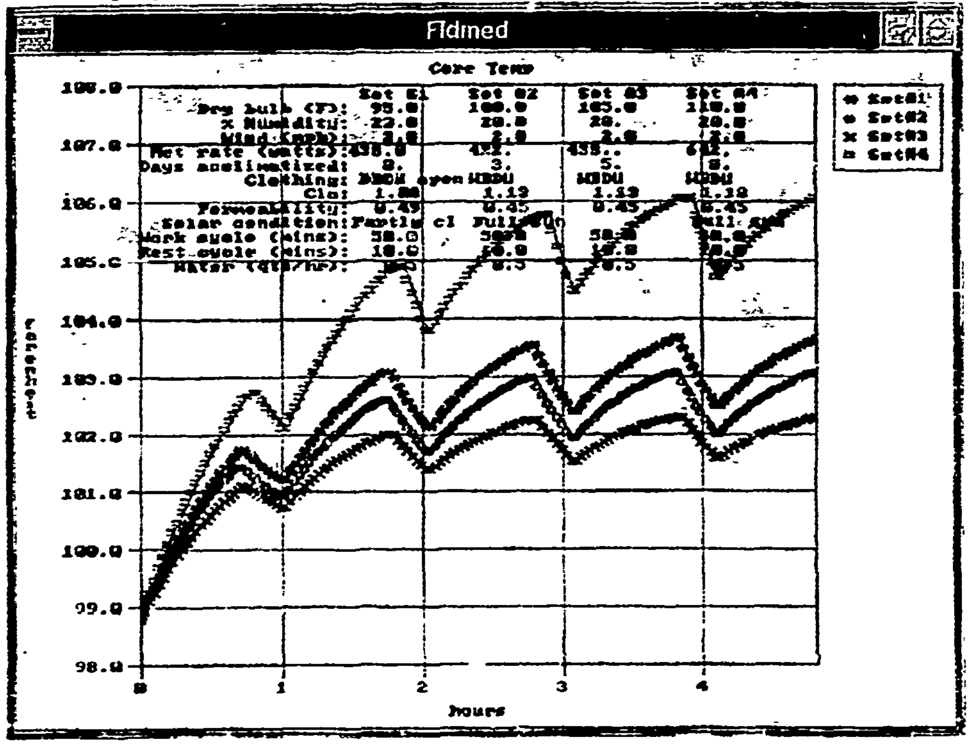
Figure A8: Heat strain output data grid.

The graphical display options are listed in the graphics selection menu shown below.



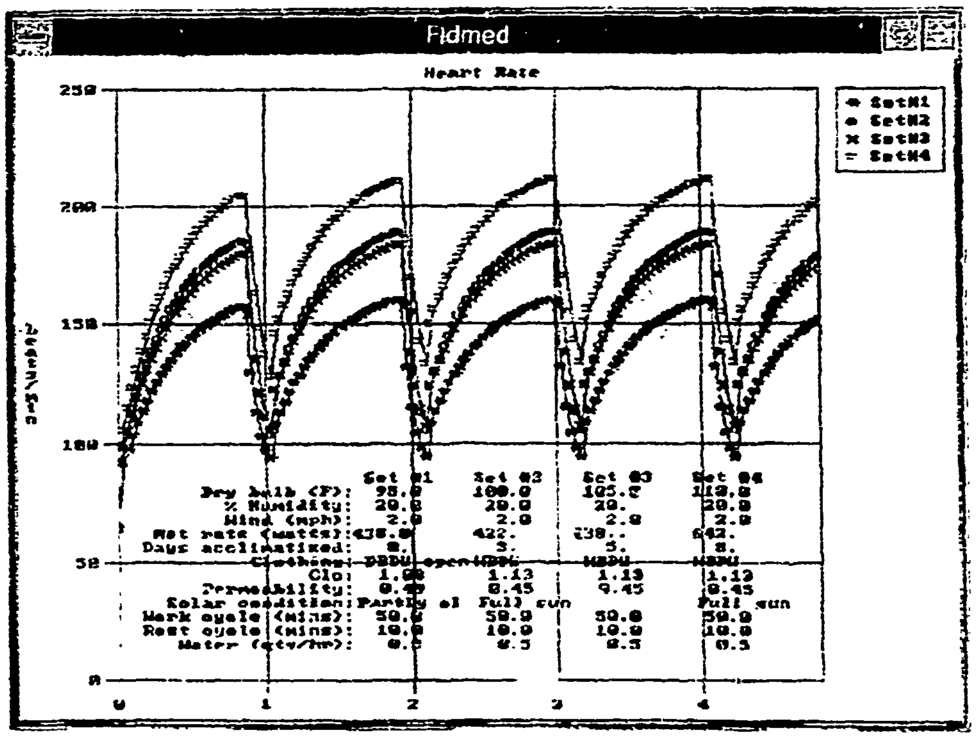
Multiple output data sets can be graphed simultaneously. Alternatively, each data set can be graphed individually. The user may select either one or four graphs per screen. Graphs can be printed directly by pressing the letters p or P, if the DOS Graphics command is installed prior to running FLDMED, and the printer is in a Hewlett-Packard (HP) printer mode.

Figures A9 and A10 illustrate core temperature and heart rate profiles for four scenarios with the same work-rest cycles. Tabular scenario input data are provided in the background to facilitate comparing results. The data sets and corresponding graphic trage~tories are given the same colors to enhance rapid visual coordination of tabular and graphic data belonging to the same sets.



	Set #1	Set #2	Set #3	Set #4
Dry bulb (F):	95.0	100.0	105.0	110.0
% Humidity:	22.0	20.0	20.	20.0
Wind (mph):	2.0	2.0	2.0	2.0
Met rate (watts):	438.0	422.	438..	642.
Days acclimatized:	0.	3.	5.	8.
Clothing:	BDU open	MBDU	MBDU	MBDU
Clo:	1.00	1.13	1.13	1.13
Permeability:	0.49	0.45	0.45	0.45
Solar condition:	Partly cl	Full sun		Full sun
Work cycle (mins):	50.0	50.0	50.0	50.0
Rest cycle (mins):	10.0	10.0	10.0	10.0
Water (qts/hr):	0.5	0.5	0.5	0.5

Figure A9: Core temperature graph.



	Set #1	Set #2	Set #3	Set #4
Dry bulb (F):	95.0	100.0	105.0	110.0
% Humidity:	20.0	20.0	20.	20.0
Wind (mph):	2.0	2.0	2.0	2.0
Met rate (watts):	438.0	422.	438..	642.
Days acclimatized:	0.	3.	5.	8.
Clothing:	BDU open	MBDU	MBDU	MBDU
Clo:	1.00	1.13	1.13	1.13
Permeability:	0.49	0.45	0.45	0.45
Solar condition:	Partly cl	Full sun		Full sun
Work cycle (mins):	50.0	50.0	50.0	50.0
Rest cycle (mins):	10.0	10.0	10.0	10.0
Water (qts/hr):	0.5	0.5	0.5	0.5

Figure A10: Heart rate profiles.

From the Options choice in the heat module' s top-bar menu, the user may select to have four graphs plotted in one graphics screen. Figure A11 reveals a typical series of graphs drawn after selecting this option. This type of output format is useful for visually comparing principal differences in the physiologic responses to the different scenarios. In the multiple graphs per view, the tabular summaries of the input data are not provided.

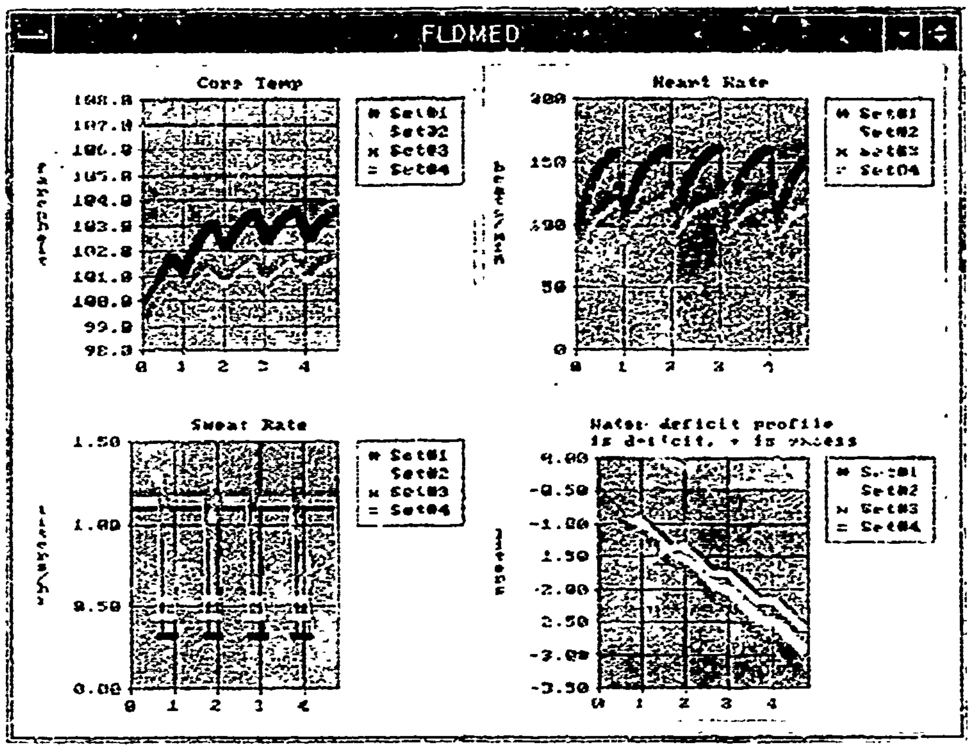


Figure A11: Multiple graphs per view.

Some of the outputs for the heat strain module are also summarized in bar charts (Figures A12 and A13). A pie chart option is also available for depicting percentage of casualties when each data-set represents different parts of a composite senario. Line graphs are useful for presenting time dependent, or dynamic quantities, whereas bar and pie charts are more useful for comparing relative magnitudes of point values, final cumulative effects, or results such as casualties.

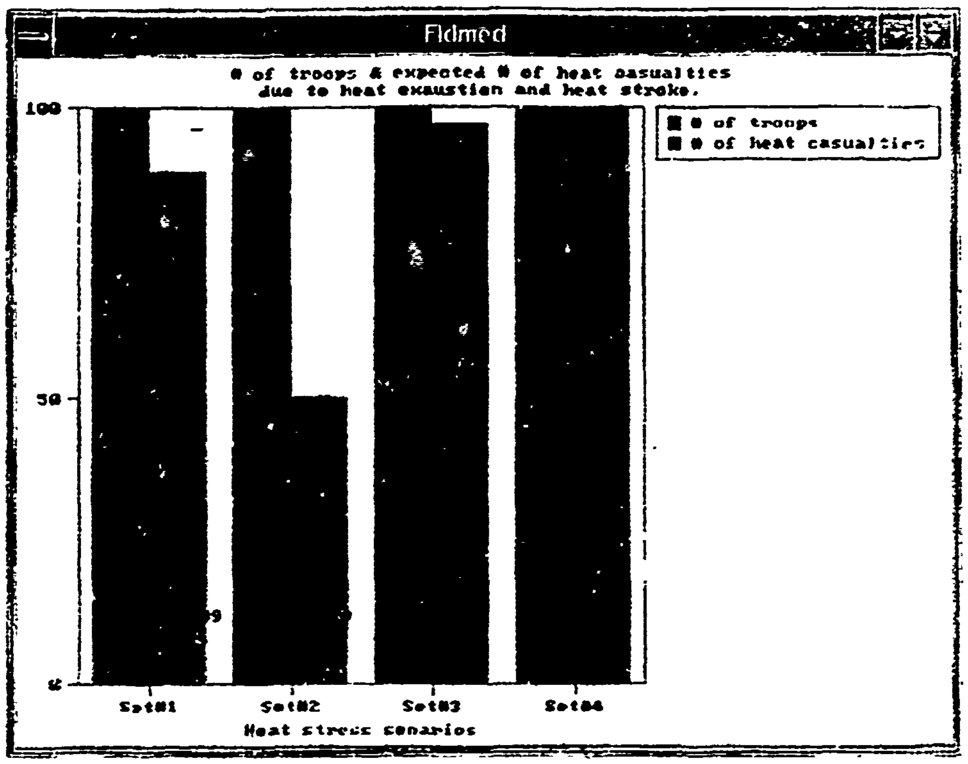


Figure A12: Heat casualty bar chart.

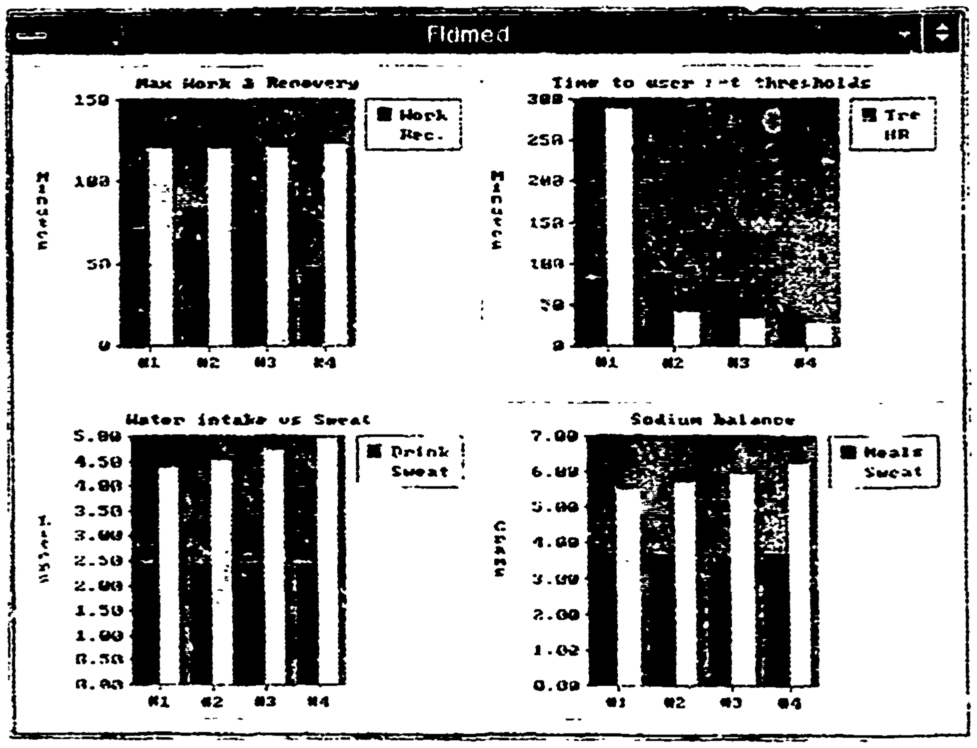
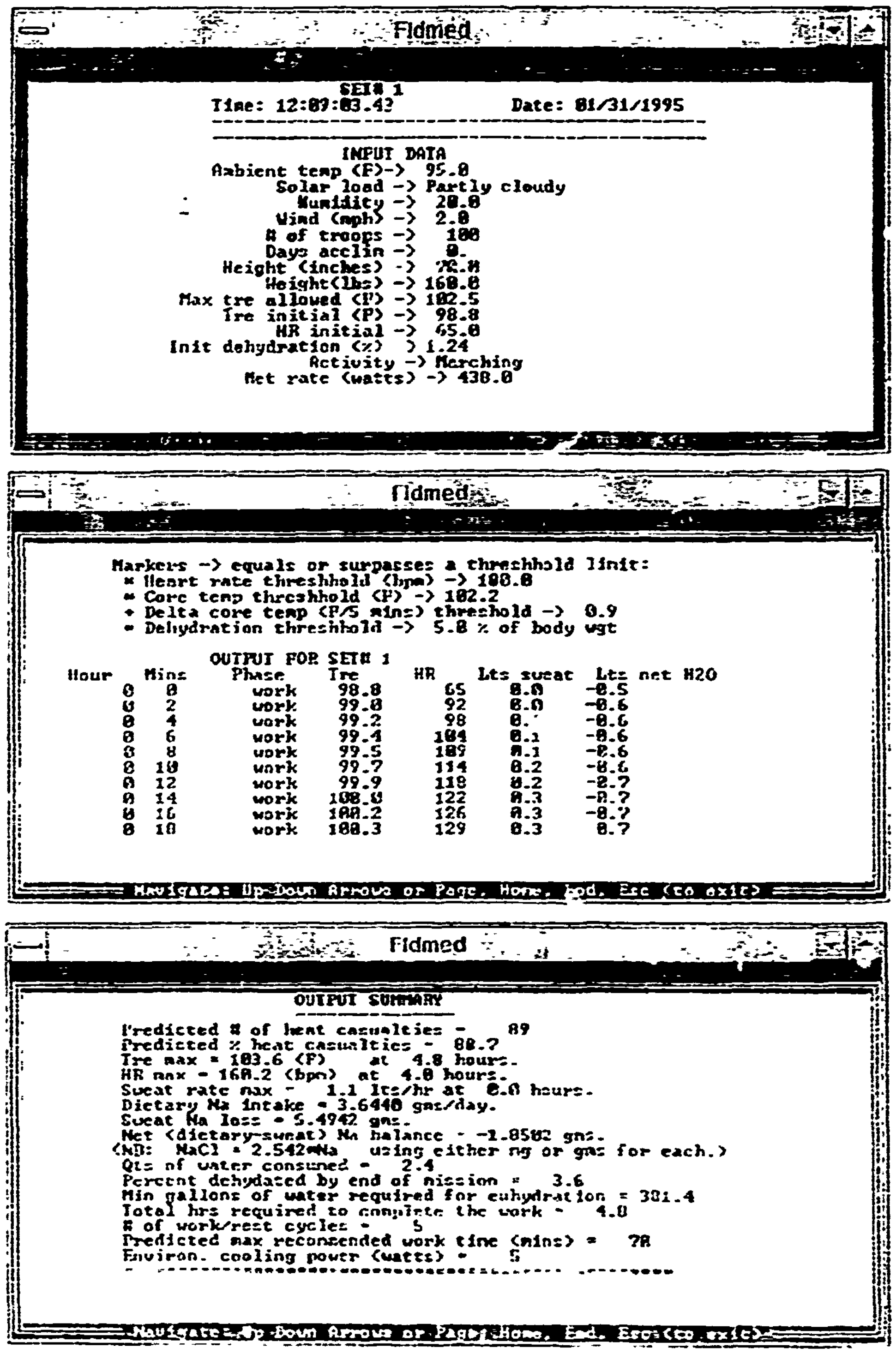


Figure A13: Miscellaneous bar charts

From the heat strain module's Output menu, the user may select the option to view a detailed listing of output data presented in text format (Figure A1`` The detailed output data listings activates a scrolling window of inputs, time dependent outputs, and output summary. The time dependent outputs are automatically tagged when values exceed those specified in the thresholds section of the inputs as previously described.



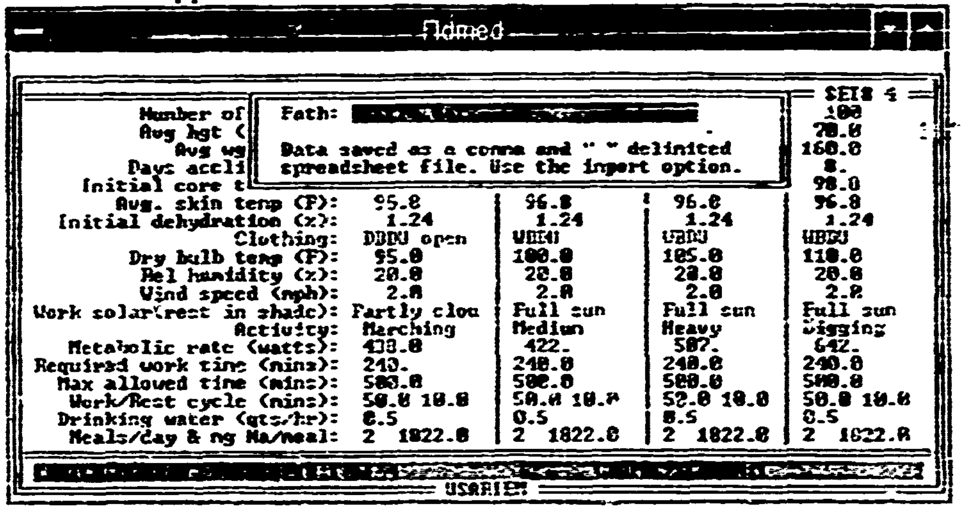
```
                    SET# 1
     Time: 12:09:03.43            Date: 01/31/1995
     ---------------------------------------------
     ---------------------------------------------
                  INPUT DATA
     Ambient temp (F)->  95.0
           Solar load -> Partly cloudy
             Humidity ->  20.0
           Wind (mph) ->  2.0
          # of troops ->   100
          Days acclim ->   0.
      Height (inches) ->  72.0
          Weight(lbs) -> 160.0
  Max tre allowed (F) -> 102.5
      Tre initial (F) ->  98.8
           HR initial ->  65.0
 Init dehydration (%)  ) 1.24
             Activity -> Marching
     Met rate (watts) -> 430.0
```

```
     Markers -> equals or surpasses a threshhold limit:
      * Heart rate threshhold (bpm) -> 100.0
      * Core temp threshhold (F) -> 102.2
      + Delta core temp (F/5 mins) threshold ->  0.9
      * Dehydration threshhold ->  5.0 % of body wgt

              OUTPUT FOR SET# 1
 Hour    Mins      Phase     Tre      HR    Lts sweat  Lts net H20
        0   0       work     98.8      65     0.0     -0.5
        0   2       work     99.0      92     0.0     -0.6
        0   4       work     99.2      98     0.'     -0.6
        0   6       work     99.4     104     0.1     -0.6
        0   8       work     99.5     109     0.1     -0.6
        0  10       work     99.7     114     0.2     -0.6
        0  12       work     99.9     118     0.2     -0.7
        0  14       work    100.0     122     0.3     -0.7
        0  16       work    100.2     126     0.3     -0.7
        0  18       work    100.3     129     0.3      0.7

 ==== Navigate: Up-Down Arrows or Page, Home, End, Esc (to exit) ====
```

```
                    OUTPUT SUMMARY
                    --------------
      Predicted # of heat casualties -    89
      Predicted % heat casualties -  88.7
      Tre max = 103.6 (F)    at  4.0 hours.
      HR max - 160.2 (bpm)  at  4.0 hours.
      Sweat rate max -   1.1 lts/hr at  0.0 hours.
      Dietary Na intake = 3.6440 gms/day.
      Sweat Na loss = 5.4942 gms.
      Net (dietary-sweat) Na balance - -1.8502 gms.
     (NB:  NaCl = 2.542*Na   using either mg or gms for each.)
      Qts of water consumed =   2.4
      Percent dehydated by end of mission =   3.6
      Min gallons of water required for euhydration = 301.4
      Total hrs required to complete the work -   4.0
      # of work/rest cycles =    5
      Predicted max reconmended work time (mins) =   78
      Environ. cooling power (watts) =    5

 ==== Navigate: Up-Down Arrows or Page, Home, End, Esc (to exit) ====
```

Figure A14: Scrolling window to view detailed output data for one input set .

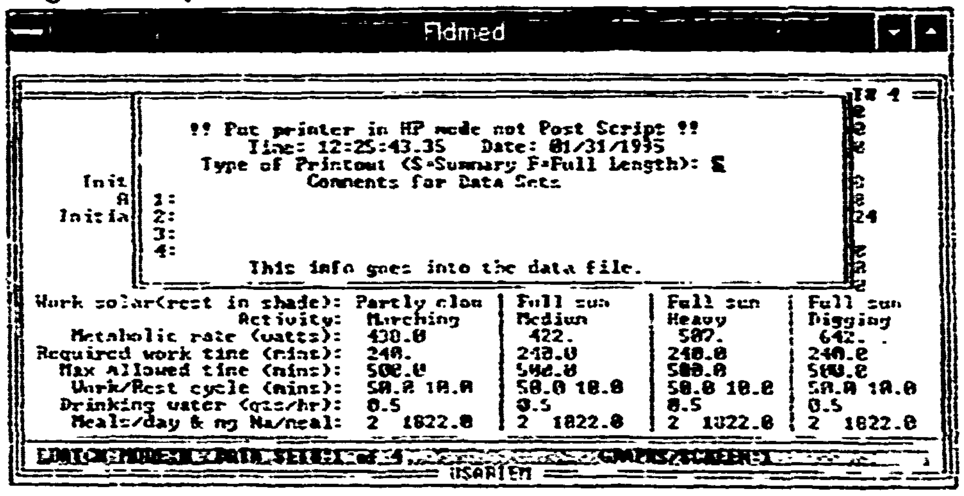
Data can also be saved in a speadsheet compatibie format. This is accomplished via serving delimit the separate data elements (Figure A15). To view the saved data file from within a speadsheet program, use the speadsheet's file import option and set the delimiter to the double quotation mark. The data will then parse automatically into the individual speadsheet cells. Further data analysis or data-set comparisons can then be carried out from within the speadsheet. If FLDMED is run within Windows this can be done without closing the FLDMED application.



```
                                                                        SET# 4
         Number of   Path: [illegible]                                   100
         Avg hgt (                                                       70.0
           Avg wg    Data saved as a comma and " " delimited            160.0
        Days accli   spreadsheet file. Use the import option.             8.
   Initial core t                                                        98.0
       Avg. skin temp (F):   95.8        | 96.8        | 96.8        | 96.8
  Initial dehydration (%):    1.24       |  1.24       |  1.24       |  1.24
                Clothing:   DBDU open    | UBDU        | UBDU        | UBDU
       Dry bulb temp (F):   95.0         | 100.0       | 105.0       | 110.0
        Rel humidity (%):   20.0         | 20.0        | 20.0        | 20.0
        Wind speed (mph):    2.0         |  2.0        |  2.0        |  2.0
Work solar(rest in shade):  Partly clou  | Full sun    | Full sun    | Full sun
                Activity:   Marching     | Medium      | Heavy       | Digging
  Metabolic rate (watts):   430.0        | 422.        | 507.        | 642.
Required work time (mins):  240.         | 240.0       | 240.0       | 240.0
  Max allowed time (mins):  500.0        | 500.0       | 500.0       | 500.0
   Work/Rest cycle (mins):  50.0 10.0    | 50.0 10.0   | 50.0 10.0   | 50.0 10.0
  Drinking water (qts/hr):  0.5          | 0.5         | 0.5         | 0.5
   Meals/day & mg Na/meal:  2  1822.0    | 2  1822.0   | 2  1822.0   | 2  1822.0
[illegible]
                                 USARIEM
```

Figure A15: Saving input & output data sets in a speadsheet compatible format.

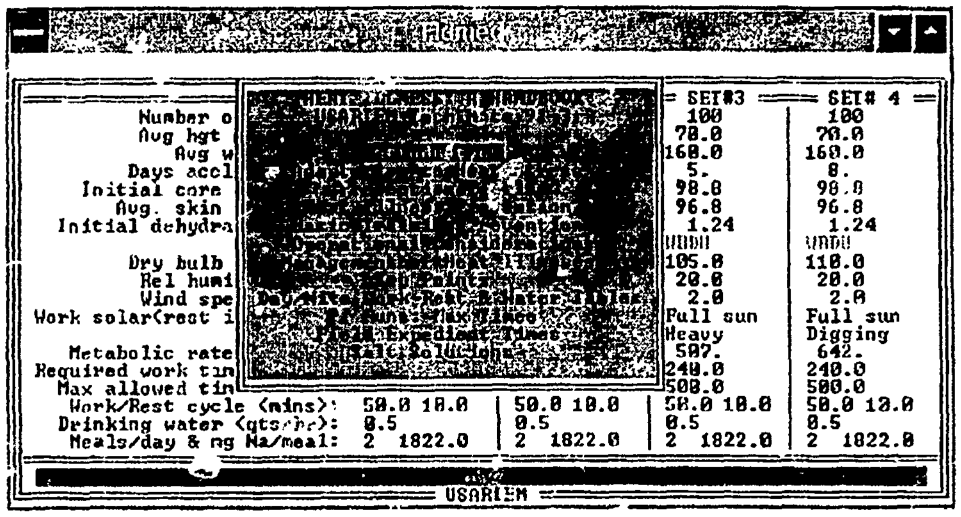
The FLDMED program also is capable of printing input-output data. The user may select either a short data summary or a longer detailed listing. Each set can be annotated with a short note (Figure A16). The program currently only supports printing in the Hewlett-Packard (HP) printer mode. If a compatible print driver is not accessible. an error message appears to notify the user that, although printing from within FLDMED was not possible, the data was written to a named text file. The user may latter print this text file from outside the program using another print driver.



```
                                                                          T# 4
                !! Put printer in HP mode not Post Script !!
                  Time: 12:25:43.35   Date: 01/31/1995
             Type of Printout (S=Summary F=Full Length): S
                        Comments for Data Sets
      Init
         A   1:
    Initia   2:                                                           24
             3:
             4:
                   This info goes into the data file.
Work solar(rest in shade):  Partly clou  | Full sun    | Full sun    | Full sun
                Activity:   Marching     | Medium      | Heavy       | Digging
  Metabolic rate (watts):   430.0        | 422.        | 507.        | 642.
Required work time (mins):  240.         | 240.0       | 240.0       | 240.0
  Max allowed time (mins):  500.0        | 500.0       | 500.0       | 500.0
   Work/Rest cycle (mins):  50.0 10.0    | 50.0 10.0   | 50.0 10.0   | 50.0 10.0
  Drinking water (qts/hr):  0.5          | 0.5         | 0.5         | 0.5
   Meals/day & mg Na/meal:  2  1822.0    | 2  1822.0   | 2  1822.0   | 2  1822.0
EDIT MODE DATA SET#1 of 4                          GRAPHS/SCREEN 1
                                 USARIEM
```

Figure A16: Requesting a printout of annotated input-output data

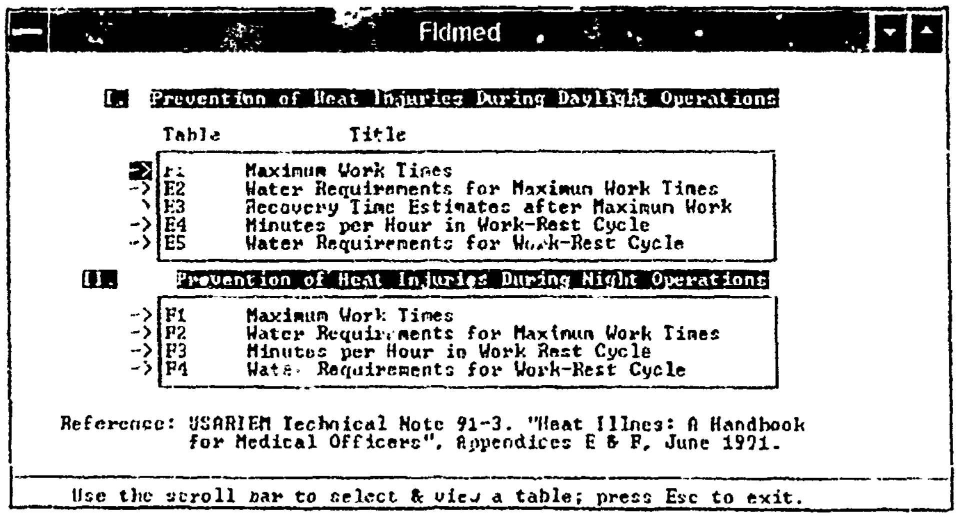
The MED INFO option in the heat strain module's top-bar menu activates a popup menu of topic choices per Figure A17 below. The MED INFO section represents an on-line version of the USARIEM Technical Note "Heat Illness: A Handbook for Medical Officers" (Burr, 1992). This handbook was reformatted somewhat to facilitate its implementation as an on-line computer-based reference. Additionally, color highlighting is used throughout the text to draw the reader's attention to key words, phrases, and concepts.



```
                                [illegible]                  = SET#3 ====== SET# 4 =
          Number o                                             100          100
          Avg hgt                                              70.0         70.0
             Avg w                                            160.0        160.0
          Days accl                                            5.           8.
       Initial core                                            98.8         98.0
          Avg. skin                                            96.8         96.8
   Initial dehydra                                             1.24         1.24
                                                              UDDU         UDDU
          Dry bulb                                            105.0        110.0
          Rel humi                                             20.0         20.0
          Wind spe                                             2.0          2.0
  Work solar(rest i                                           Full sun     Full sun
                                                              Heavy        Digging
      Metabolic rate                                           507.         642.
  Required work tin                                           240.0        240.0
    Max allowed tin                                           500.0        500.0
    Work/Rest cycle (mins):  50.0 10.0   | 50.0 10.0   | 50.0 10.0   | 50.0 10.0
   Drinking water (qts/hr):  0.5         | 0.5         | 0.5         | 0.5
    Meals/day & mg Na/meal:  2  1822.0   | 2  1822.0   | 2  1822.0   | 2  1822.0
                                  USARIEM
```

Figure A17: On-line medical heat strain handbook.

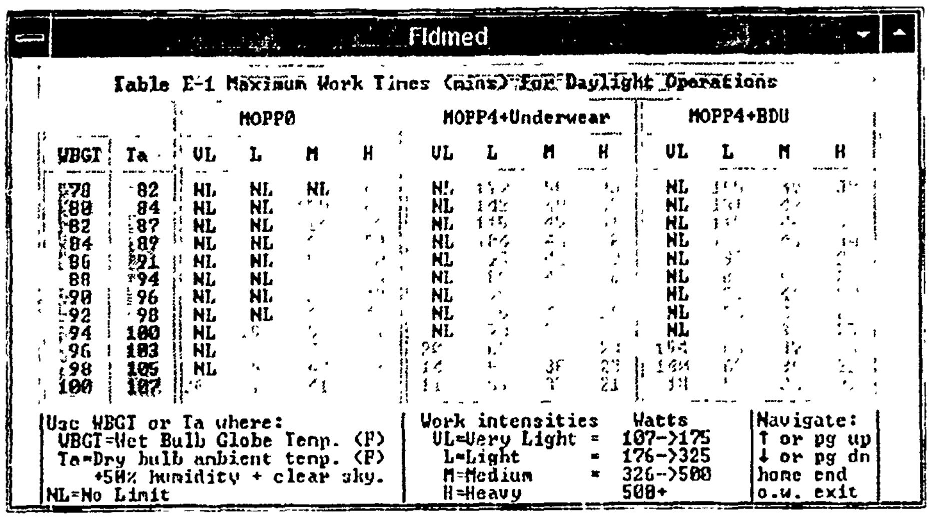
As an example, if the user scrolls down the MED INFO menu to the line for Work-Rest and Water Tables and then hits the Return kev, Figure A18 below appears. This scrolling pick list of tables allows the user to rapidly jump directly to specific work-rest, maximum work time, or water requirement tables. As indicated, separate tables are provided for both day-time and night-time military operations.



```
                                Fldmed

   I.   Prevention of Heat Injuries During Daylight Operations

        Table          Title
     -> E1      Maximum Work Times
     -> E2      Water Requirements for Maximum Work Times
      > E3      Recovery Time Estimates after Maximum Work
     -> E4      Minutes per Hour in Work-Rest Cycle
     -> E5      Water Requirements for Work-Rest Cycle

  II.       Prevention of Heat Injuries During Night Operations

     -> F1      Maximum Work Times
     -> F2      Water Requirements for Maximum Work Times
     -> F3      Minutes per Hour in Work Rest Cycle
     -> F4      Wate  Requirements for Work-Rest Cycle

 Reference: USARIEM Technical Note 91-3. "Heat Illnes: A Handbook
            for Medical Officers", Appendices E & F, June 1991.

   Use the scroll bar to select & view a table; press Esc to exit.
```

Figure A18: Work-rest and water table selection list.

Figure A19 is an example of a table that provided recommendations for single-shot maximum work times during daylight operations (Burr, 1991).



Fldmed

Table E-1 Maximum Work Times (mins) for Daylight Operations

		MOPP0				MOPP4+Underwear				MOPP4+BDU			
WBGT	Ta	UL	L	M	H	UL	L	M	H	UL	L	M	H
78	82	NL	NL	NL	[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]
80	84	NL	NL	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]	[illegible]	
82	87	NL	NL	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]
84	89	NL	NL	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]
86	91	NL	NL	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]		[illegible]
88	94	NL	NL		[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]
90	96	NL	NL	[illegible]	[illegible]	NL	[illegible]			NL	[illegible]	[illegible]	[illegible]
92	98	NL	NL	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]	[illegible]	[illegible]
94	100	NL	[illegible]	[illegible]	[illegible]	NL	[illegible]			NL	[illegible]	[illegible]	[illegible]
96	103	NL		[illegible]		[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
98	105	NL	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
100	107	[illegible]	[illegible]	[illegible]		[illegible]	[illegible]	[illegible]	21	[illegible]	[illegible]	[illegible]	[illegible]

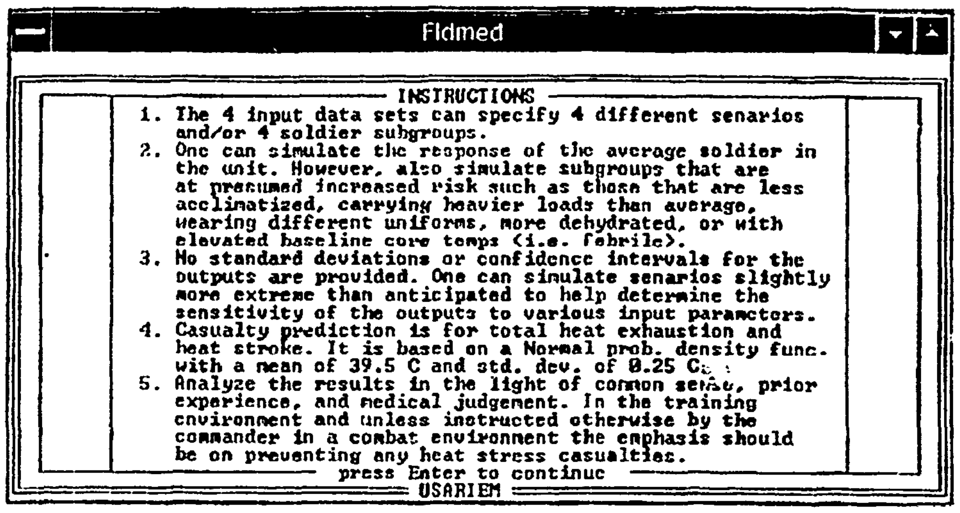
Use WBGT or Ta where:
WBGT=Wet Bulb Globe Temp. (F)
Ta=Dry bulb ambient temp. (F)
+50% humidity + clear sky.
NL=No Limit

Work intensities		Watts
UL=Very Light	=	107->175
L=Light	=	176->325
M=Medium	=	326->500
H=Heavy		500+

Navigate:
↑ or pg up
↓ or pg dn
home end
o.w. exit

Figure A19: Maximum-work time table in the medical handbook.

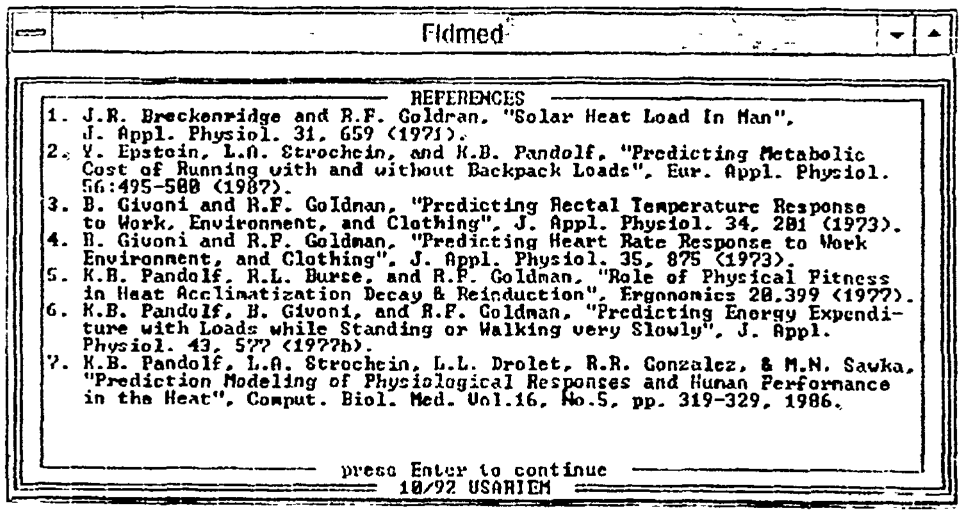
The Help option in the heat strain module's top-bar menu provides the user instructions and caveats as shown in Figure A20.



```
                         INSTRUCTIONS
1. The 4 input data sets can specify 4 different senarios
   and/or 4 soldier subgroups.
2. One can simulate the response of the average soldier in
   the unit. However, also simulate subgroups that are
   at presumed increased risk such as those that are less
   acclimatized, carrying heavier loads than average,
   wearing different uniforms, more dehydrated, or with
   elevated baseline core temps (i.e. febrile).
3. No standard deviations or confidence intervals for the
   outputs are provided. One can simulate senarios slightly
   more extreme than anticipated to help determine the
   sensitivity of the outputs to various input parameters.
4. Casualty prediction is for total heat exhaustion and
   heat stroke. It is based on a Normal prob. density func.
   with a mean of 39.5 C and std. dev. of 0.25 C.
5. Analyze the results in the light of common sense, prior
   experience, and medical judgement. In the training
   environment and unless instructed otherwise by the
   commander in a combat environment the emphasis should
   be on preventing any heat stress casualties.
                     press Enter to continue
                            USARIEM
```

Figure 20: Heat strain module help and information screen.

From this screen, if the user hits the Return key, a list of references for the USARIEM heat strain model is provided (Figure A21).



```
                         REFERENCES
1. J.R. Breckenridge and R.F. Goldman, "Solar Heat Load In Man",
   J. Appl. Physiol. 31, 659 (1971).
2. Y. Epstein, L.A. Strochein, and K.B. Pandolf, "Predicting Metabolic
   Cost of Running with and without Backpack Loads", Eur. Appl. Physiol.
   56:495-500 (1987).
3. B. Givoni and R.F. Goldman, "Predicting Rectal Temperature Response
   to Work, Environment, and Clothing", J. Appl. Physiol. 34, 201 (1973).
4. B. Givoni and R.F. Goldman, "Predicting Heart Rate Response to Work
   Environment, and Clothing", J. Appl. Physiol. 35, 875 (1973).
5. K.B. Pandolf, R.L. Burse, and R.F. Goldman, "Role of Physical Fitness
   in Heat Acclimatization Decay & Reinduction", Ergonomics 20,399 (1977).
6. K.B. Pandolf, B. Givoni, and R.F. Goldman, "Predicting Energy Expendi-
   ture with Loads while Standing or Walking very Slowly", J. Appl.
   Physiol. 43, 577 (1977b).
7. K.B. Pandolf, L.A. Strochein, L.L. Drolet, R.R. Gonzalez, & M.N. Sawka,
   "Prediction Modeling of Physiological Responses and Human Performance
   in the Heat", Comput. Biol. Med. Vol.16, No.5, pp. 319-329, 1986.

                     press Enter to continue
                         10/92 USARIEM
```

Figure A21: On-line refernce list for the heat strain module.

Figure A21 shows the introductory screen for the Altitude module. Depicted is a grid with a default ascent-decent profile in overlaid tabular descriptive and graphical formats. The vertical axis is altitude in thousands of feet. The units for the horizontal axis are number of days. The user may select the Input options from the top-bar menu. A pop-up input selection menu then appears.

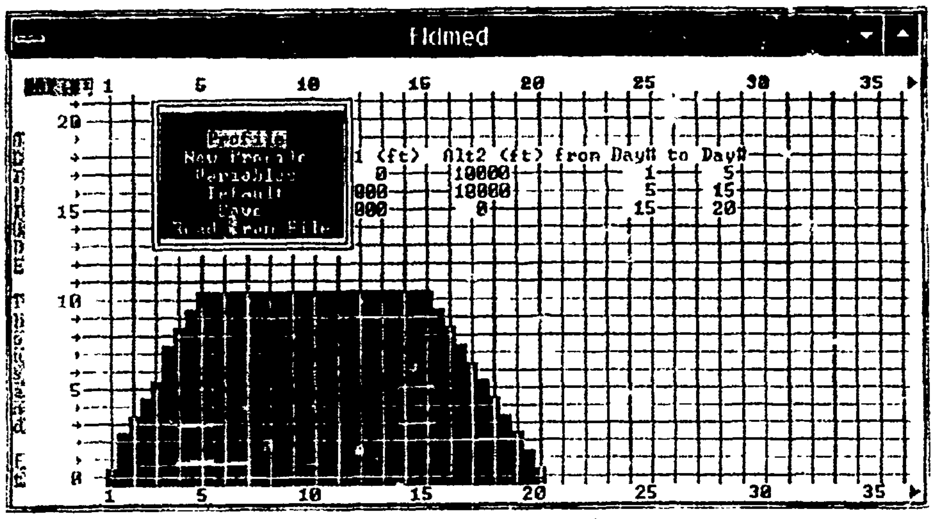
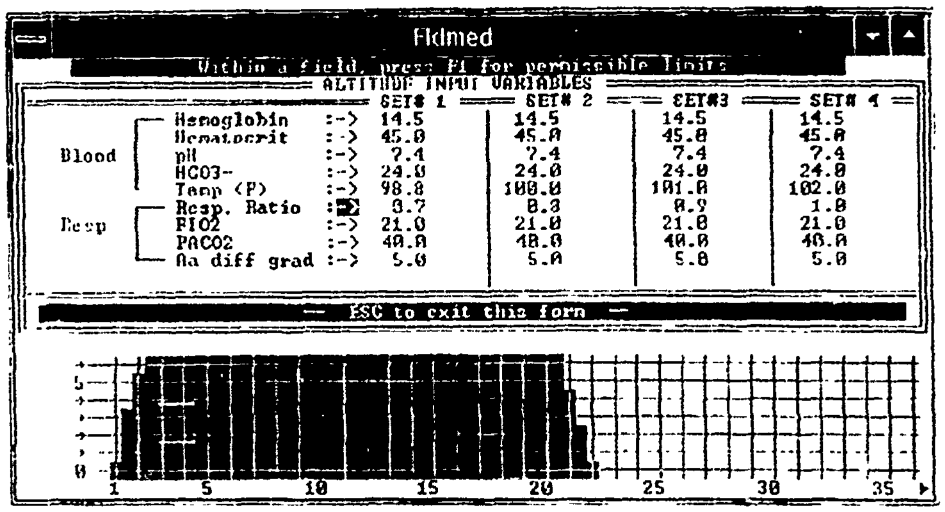


Figure A21: Altitude default profile and input mode selection pop-up menu.

To specify the physiologic parameters for any or all of four senarios or sets the user should scroll within the Inputs menu to the Variables choice and then hit Return. This causes an input grid for the physiologic variables to appear as shown in Figure A22. Within this input data grid parameters can be changed within predefined constraints. Values outside of the valid data ranges are not accepted. The F1 key can be pressed to activate a small pop-up dialog box that states a variable's limits. In Figure A22 the user has entered a different value for the respiratory ratio in each of the four data sets.



```
                         ALTITUDE INPUT VARIABLES
                               SET# 1      SET# 2      SET#3      SET# 4
         ┌── Hemoglobin    :->  14.5        14.5        14.5       14.5
         │   Hematocrit    :->  45.0        45.0        45.0       45.0
  Blood  │   pH            :->   7.4         7.4         7.4        7.4
         │   HCO3-         :->  24.0        24.0        24.0       24.0
         └   Temp (F)      :->  98.8       100.0       101.0      102.0
         ┌── Resp. Ratio   :->   0.7         0.3         0.9        1.0
  Resp   │   FIO2          :->  21.0        21.0        21.0       21.0
         │   PACO2         :->  40.0        40.0        40.0       40.0
         └── Aa diff grad  :->   5.0         5.0         5.0        5.0
```

Figure A22: Input grid for physiologic parameters.

The altitude module's default ascent-decent profile can be cleared by selecting the Clear Profile option from the Input option. The Create New Profile selection also clears the current profile and generates a window for creating a new ascent-descent profile. Figures A23-A25 illustrates the data input windows and movement options popup for constructing a custom ascent-decent profile.

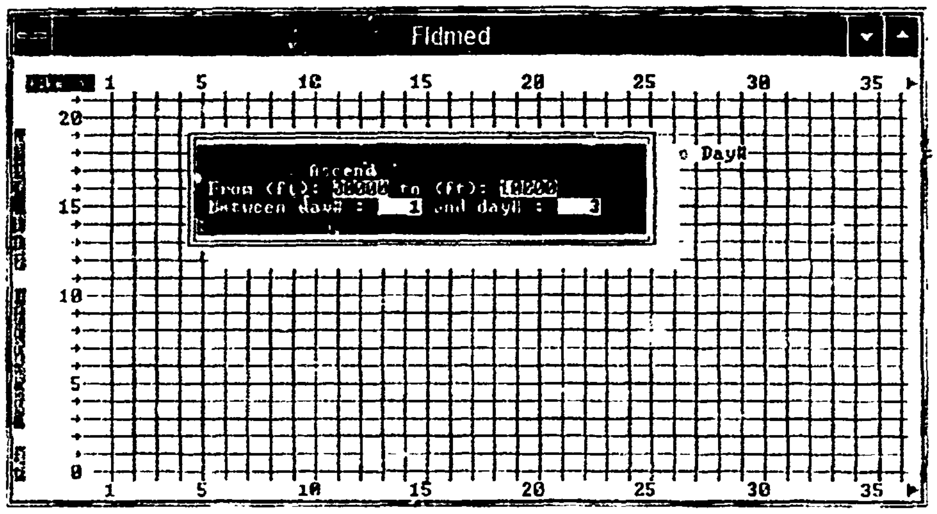

Ascend
From (ft): [illegible] to (ft): [illegible]
Between day# : 1 and day# : 3

Figure A23: Old ascent-decent profile cleared, starting new profile.

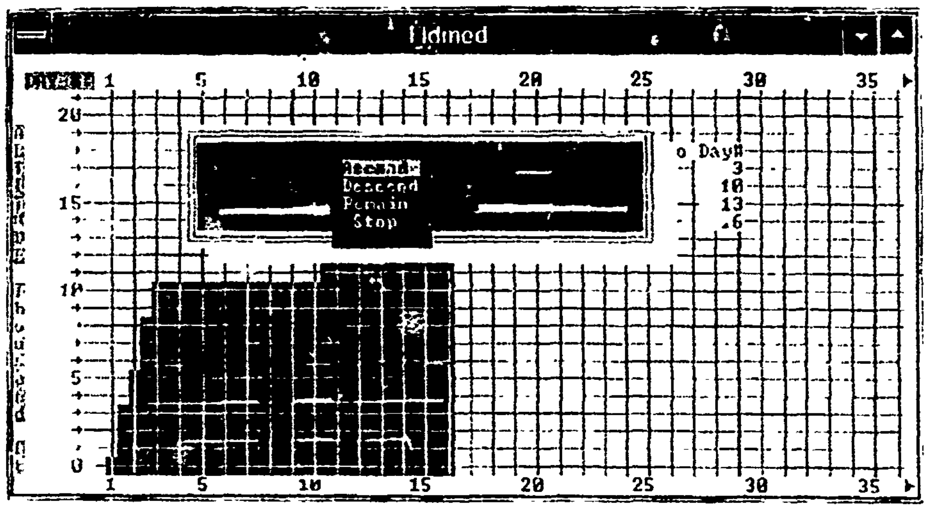


Figure A24: Selecting an ascent-decent movement.

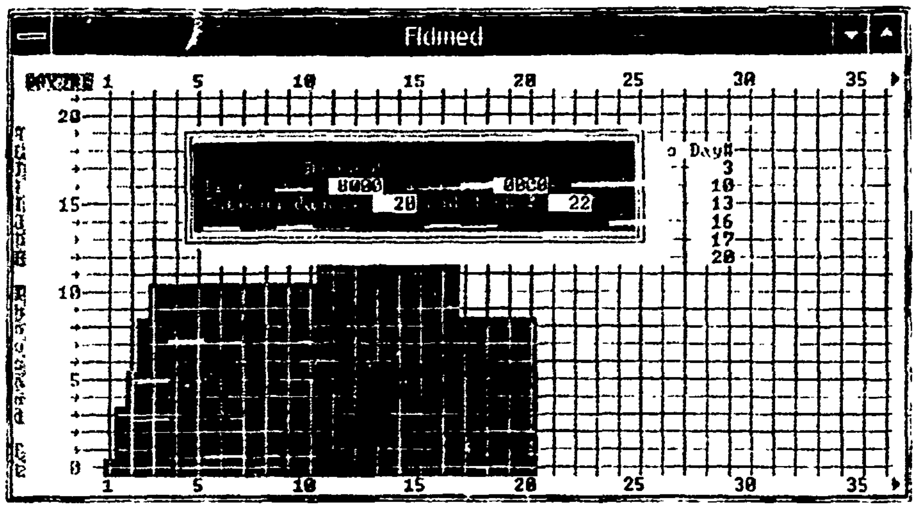
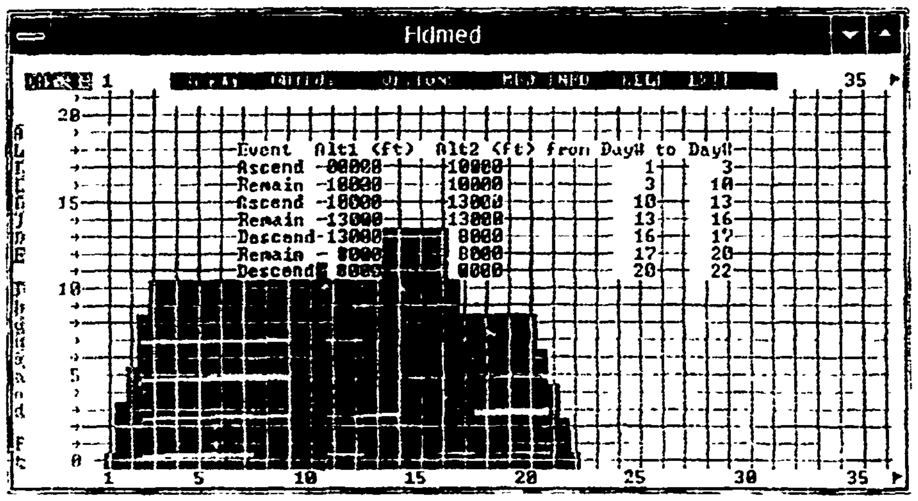


Figure A25: Entering data for an ascent-decent movement phase.



```
Event  Alt1 (ft)  Alt2 (ft) from Day# to Day#
Ascend  00000     10000          1       3
Remain  10000     10000          3      10
Ascend  10000     13000         10      13
Remain  13000     13000         13      16
Descend 13000      8000         16      17
Remain   8000      8000         17      20
Descend  8000      0000         20      22
```

Figure A26: New ascent-decent profile with return to main altitude screen.

After the physiologic parameters and ascent-decent profile (Figure A26) have been entered into the altitude module, the user may select the Output option from the top-bar menu to generate a set of composite line graphs (Figure A27). The graphs plot ambient pressure, ambient oxygen concentration, alveolar oxygen concetration, and arterial oxygen concentration as functions of the values of time, altitude, and physiologic variables.

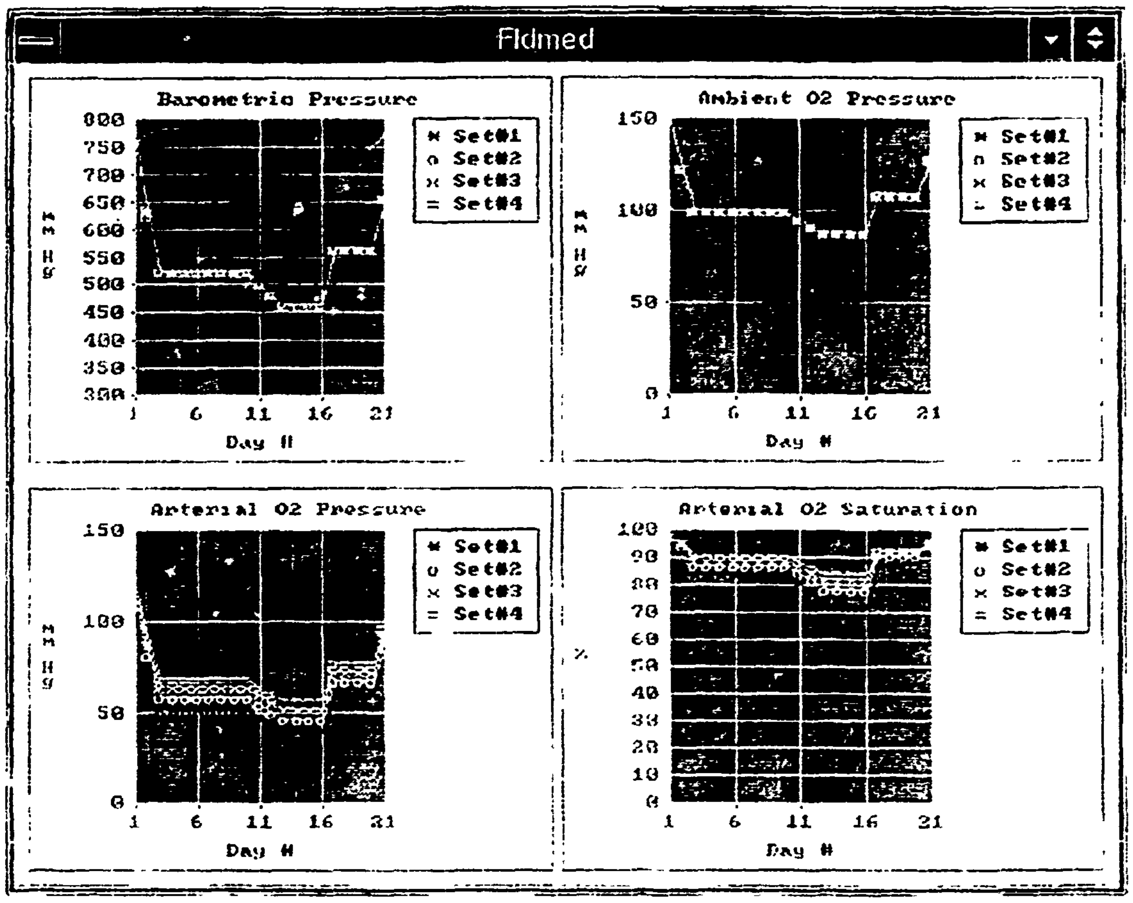


Figure A27: Graphical output for the altitude module.

As was the case for the heat and cold modules, in the altitude module, the user may select MED INFO from the altitude module's top-bar menu. This option activates a popup menu that lists topics in military medicine related to deployments to locations at high altitude (Cymeman and Rock, 1994). Figure A28 below illustrates the appearance of the medical reference topic selection mode. To select a topic, the up or down arrow is utilized to scroll through the options. The Enter key activates the selected choice. Alternatively, typing the first letter of a topic will send the cursor directly to that line.

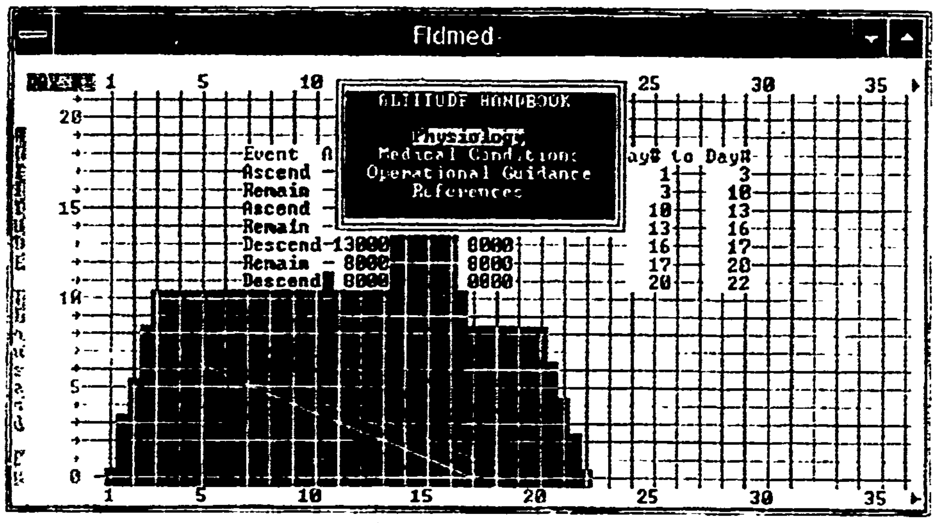
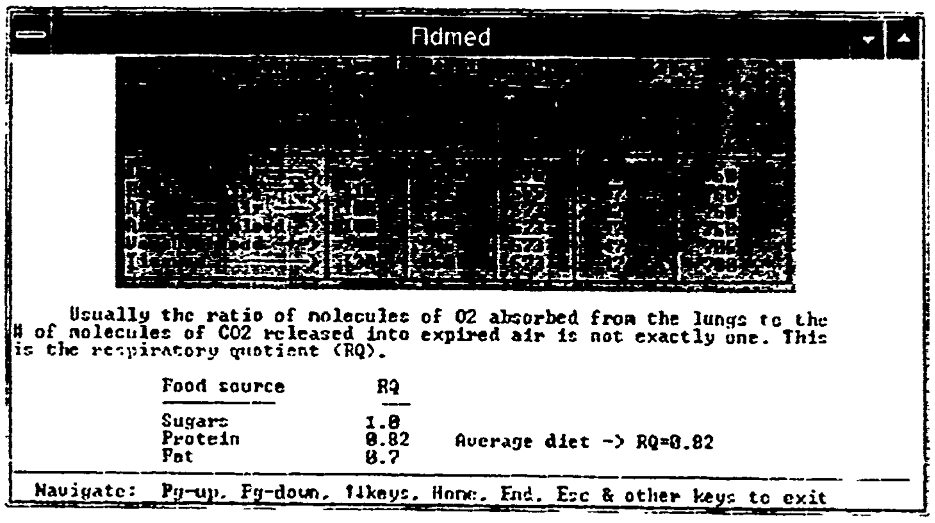


Figure A28: Altitude's medical reference topic menu.

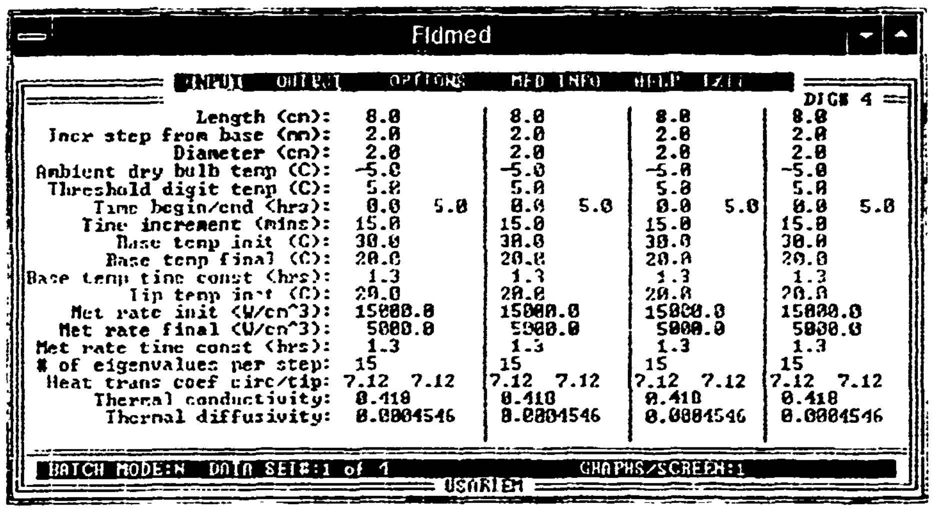
Figure A29 below is a typical display from the altitude module's medical information reference section. In this example, the user can review a table of reference values for the constituents of air in the environment as well as for various internal locations. Additionally the respiratory ratios are provided for the three major macronutrient food groups.



```
    Usually the ratio of molecules of O2 absorbed from the lungs to the
# of molecules of CO2 released into expired air is not exactly one. This
is the respiratory quotient (RQ).

             Food source        RQ
             ___________        __
             Sugars             1.0
             Protein            0.82    Average diet -> RQ=0.82
             Fat                0.7

  Navigate:  Pg-up. Pg-down. ↑↓keys. Home. End. Esc & other keys to exit
```

Figure A29: Altitude physiology content example.

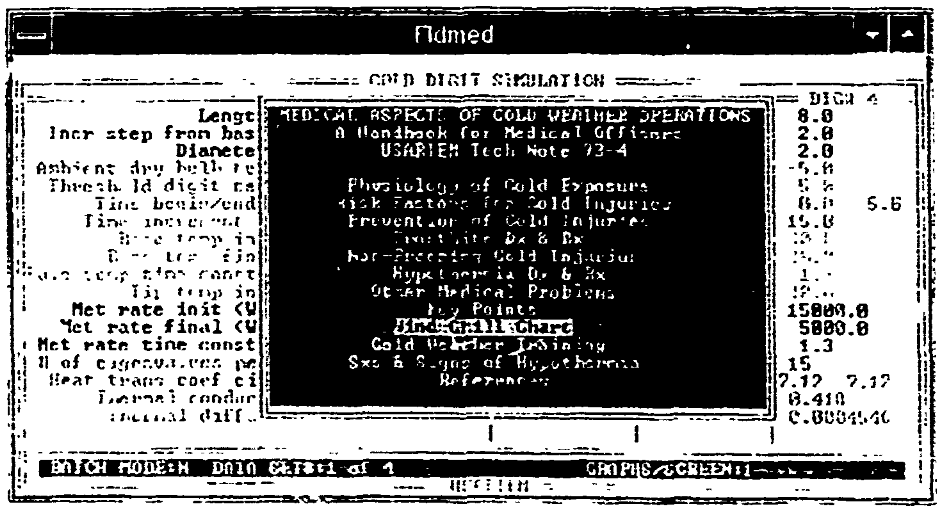
FLDMED's cold module is reached from the top-bar menu of the main program's introductory screen. The dynamic numerical portion of the cold module has not been implemented; however, the data structure heirarchy for thisd module was specifically desigred to accommodate the insertion of the predictive USARIEM lumped parameter cold digit model (mathematics of the model described by Shitzer, et al., 1992; an expanded software implementation derived from that reference is described in Reardon, 1994). The input interface designed to accept the necessary inputs for the cold digit model is shown in Figure A30. This input interface utilizes the same format as the input screen for the heat stress module.



```
                                   Fldmed
      INPUT   OUTPUT   OPTIONS   MED INFO   HELP   EXIT
                                                                  DIG# 4
                  Length (cm):  8.0          | 8.0          | 8.0          | 8.0
     Incr step from base (mm):  2.0          | 2.0          | 2.0          | 2.0
                Diameter (cm):  2.0          | 2.0          | 2.0          | 2.0
  Ambient dry bulb temp (C):   -5.0          | -5.0         | -5.0         | -5.0
     Threshold digit temp (C):  5.0          | 5.0          | 5.0          | 5.0
         Time begin/end (hrs):  0.0    5.0   | 0.0    5.0   | 0.0    5.0   | 0.0    5.0
      Time increment (mins):    15.0         | 15.0         | 15.0         | 15.0
           Base temp init (C):  30.0         | 30.0         | 30.0         | 30.0
          Base temp final (C):  20.0         | 20.0         | 20.0         | 20.0
Base temp time const (hrs):     1.3          | 1.3          | 1.3          | 1.3
             Tip temp init (C): 20.0         | 20.0         | 20.0         | 20.0
      Met rate init (W/cm^3):   15000.0      | 15000.0      | 15000.0      | 15000.0
     Met rate final (W/cm^3):   5000.0       | 5000.0       | 5000.0       | 5000.0
  Met rate time const (hrs):    1.3          | 1.3          | 1.3          | 1.3
  # of eigenvalues per step:    15           | 15           | 15           | 15
   Heat trans coef circ/tip:  7.12  7.12     | 7.12  7.12   | 7.12  7.12   | 7.12  7.12
       Thermal conductivity:    0.418        | 0.418        | 0.418        | 0.418
        Thermal diffusivity:    0.0004546    | 0.0004546    | 0.0004546    | 0.0004546

 BATCH MODE:N  DATA SET#:1 of 1                     GRAPHS/SCREEN:1
                               USARIEM
```

Figure A30: Cold module input screen.

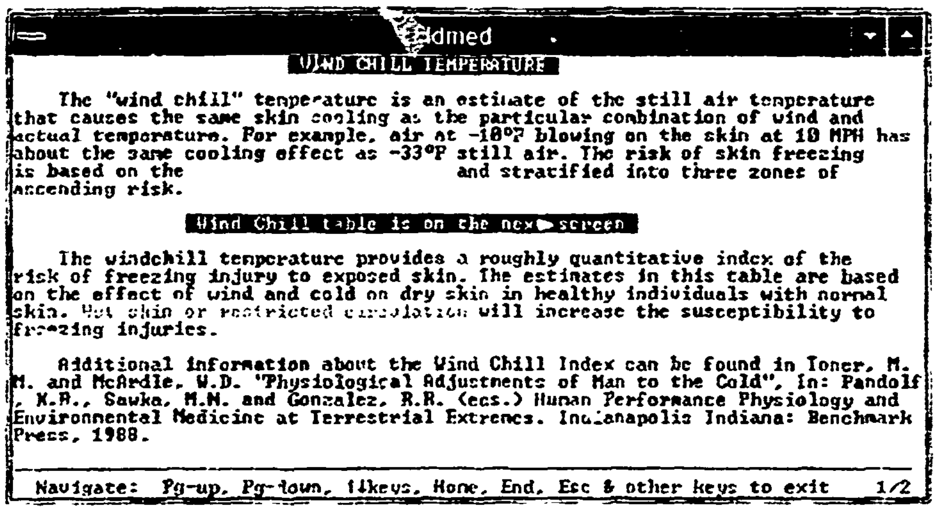
Although the cold module's output section has not yet been completed, the MEDINFO top-bar menu choice is active and is depicted below. The popup menu contents for the online cold weather medical reference is shown in Figure A31 (Burr, 1993). In this example, the user has scrolled down to the Wind Chill Chart topic line. Hitting the Return key causes the corresponding section of the altitude medical reference to appear in subsequent screens.



```
                                   Fldmed
                       COLD DIGIT SIMULATION
                                                                  DIG# 4
              Lengt  MEDICAL ASPECTS OF COLD WEATHER OPERATIONS   8.0
  Incr step from bas    A Handbook for Medical Officers           2.0
            Diamete        USARIEM Tech Note 93-4                 2.0
Ambient dry bulb te                                               -5.0
 Threshold digit te       Physiology of Cold Exposure             5.0
     Time begin/end       Risk Factors for Cold Injuries          0.0    5.0
     Time increment       Prevention of Cold Injuries             15.0
       Base temp in           Frostbite Dx & Rx                   30.0
       Base temp fin     Non-Freezing Cold Injuries               20.0
Base temp time const          Hypothermia Dx & Rx                 1.3
        Tip temp in        Other Medical Problems                 20.0
     Met rate init (W           Key Points                        15000.0
    Met rate final (W         Wind Chill Chart                    5000.0
Met rate time const       Cold Weather Training                   1.3
# of eigenvalues pe     Sxs & Signs of Hypothermia                15
 Heat trans coef ci             References                        7.12  7.12
   Thermal conduc                                                 0.418
    Thermal diff                                                  0.0004546

 BATCH MODE:N  DATA SET#:1 of 1                     GRAPHS/SCREEN:1
                               USARIEM
```

Figure A31: Cold module's medical reference topic menu.

Figure A32 illustrates the contents of the first information screen for the windchill section of the altitude medicine reference. It provides background descriptive information about the effective use of the windchill index.



Aldmed

WIND CHILL TEMPERATURE

The "wind chill" temperature is an estimate of the still air temperature that causes the same skin cooling as the particular combination of wind and actual temperature. For example, air at -10°F blowing on the skin at 10 MPH has about the same cooling effect as -33°F still air. The risk of skin freezing is based on the   and stratified into three zones of ascending risk.

Wind Chill table is on the next screen

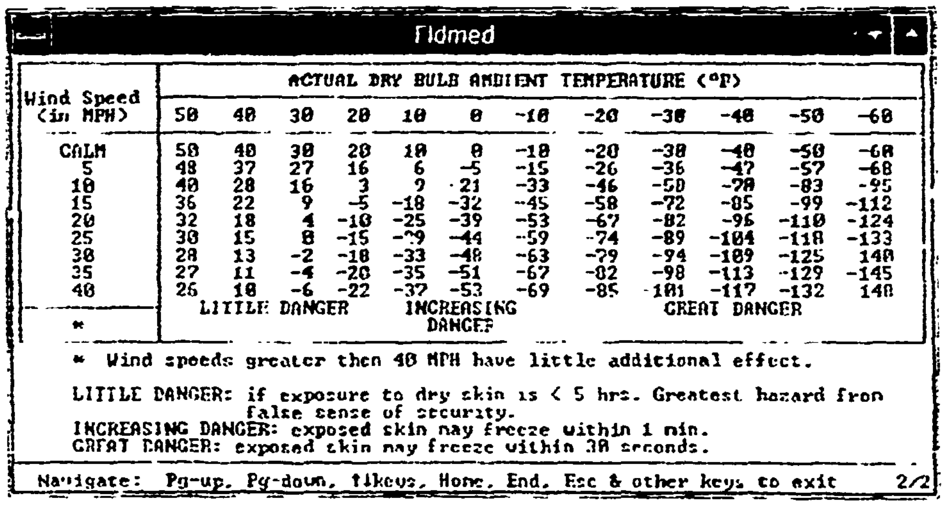
The windchill temperature provides a roughly quantitative index of the risk of freezing injury to exposed skin. The estimates in this table are based on the effect of wind and cold on dry skin in healthy individuals with normal skin. Wet skin or restricted circulation will increase the susceptibility to freezing injuries.

Additional information about the Wind Chill Index can be found in Toner, M. M. and McArdle, W.D. "Physiological Adjustments of Man to the Cold", In: Pandolf, K.B., Sawka, M.N. and Gonzalez, R.R. (eds.) Human Performance Physiology and Environmental Medicine at Terrestrial Extremes. Indianapolis Indiana: Benchmark Press, 1988.

Navigate: Pg-up, Pg-down, ↑↓keys, Home, End, Esc & other keys to exit 1/2

Figure A32: Wind chill section of cold module medical reference.

Now, if the user presses the down arrow key or Page Down key, the wind chill chart will appear (Figure A33). Such charts and tables provide on-line advisory references as functions of operationally relevant parameters within predefined useful ranges. These expand the utility of the overall program and preclude the need for the user or program to carry out cumbersome and time-consuming calculations for commonly accessed data such as the windchill index.



Aldmed

Wind Speed (in MPH)	ACTUAL DRY BULB AMBIENT TEMPERATURE (°F)											
	50	40	30	20	10	0	-10	-20	-30	-40	-50	-60
CALM	50	40	30	20	10	0	-10	-20	-30	-40	-50	-60
5	48	37	27	16	6	-5	-15	-26	-36	-47	-57	-68
10	40	28	16	3	9	-21	-33	-46	-50	-70	-83	-95
15	36	22	9	-5	-18	-32	-45	-58	-72	-85	-99	-112
20	32	18	4	-10	-25	-39	-53	-67	-82	-96	-110	-124
25	30	15	8	-15	-29	-44	-59	-74	-89	-104	-118	-133
30	28	13	-2	-18	-33	-48	-63	-79	-94	-109	-125	140
35	27	11	-4	-20	-35	-51	-67	-82	-98	-113	-129	-145
40	26	10	-6	-22	-37	-53	-69	-85	-101	-117	-132	140
*	LITTLE DANGER				INCREASING DANGER				GREAT DANGER			

* Wind speeds greater then 40 MPH have little additional effect.

LITTLE DANGER: if exposure to dry skin is < 5 hrs. Greatest hazard from false sense of security.
INCREASING DANGER: exposed skin may freeze within 1 min.
GREAT DANGER: exposed skin may freeze within 30 seconds.

Navigate: Pg-up, Pg-down, ↑↓keys, Home, End, Esc & other keys to exit 2/2

Figure A33: The wind chill chart.

APPENDIX B: USARIEM Heat Strain Model in MathCAD

(adapted from Stroschein, 1994)

Input Variables (initial values):

Soldier:	Environment:	Metabolic rate:	Clothing:	Tre limits:
$Ht := 172$	$Ta := 35.0$	$Watts_{work} := 350$	$Itc := 2.09$	$tre\_max_{max\_work} := 39.0$
$Wgt := 70$	$Rh := 50$	$Watts_{rec} := 100$	$Itvc := -0.15$	$tre\_max_{work\_rest} := 38.5$
$DIH := 99.0$	$Wsp := 0.6$	$Watts_{ext} := 0.0$	$Imc := 0.16$	$Tre\_limit_{wk} := 39.0$
$Treo := 37.0$	$Tglb := 0.0$		$Imvc := 0.20$	
$Tsk := 36.5$	$SlrF := 0.0$			
	$CldF := 4.0$			

Vapor Pressure Calculations

$$Torr\_h2o_{sat}(T) := 10^{8.1076 - \frac{1750.286}{T + 235.0}}$$

Function declaration for determining water vapor pressure of air saturated with water at a specified skin temperature

$$Torr\_h2o_{skin} := Torr\_h2o_{sat}(Tsk)$$

$$Torr\_h2o_{ambient} := \frac{Rh}{100} \cdot \left(Torr\_h2o_{sat}(Ta)\right)$$

$Torr\_h2o_{skin} = 45.801$ $\quad$ $Torr\_h2o_{ambient} = 21.088$

$$Wind_{eff}(M) := Wsp + 0.004 \cdot (M - 105.0)$$

$$It(V) := Itc \cdot V^{Itvc}$$

$$Evap_{cloth}(V) := Imc \cdot V^{Imvc}$$

$$Solar_{cloth\_efficiency}(V) := \frac{0.41}{Itc} \cdot V^{-(0.43 - Itvc)}$$

Function declarations for:

1. determining effective wind speed across the body given actual wind speed and metabolic rate.
2. determining insulation (clo) corrected for a specified effective wind speed.
3. determining evaporative capacity as a function of wind speed for a given clothing water vapor permeability index (imc).
4. determining efficiency with which clothing transmits solar load though clothing to the skin as a function of effective wind velocity.

$Wind_{eff}\left(Watts_{work}\right) = 1.58$ $\quad$ $Wind_{eff}\left(Watts_{rec}\right) = 0.6$

$$Vel_{wk} := Wind_{eff}\left(Watts_{work}\right)$$

$$Vel_{rec} := Wind_{eff}\left(Watts_{rec}\right)$$

$It_{work} := It(Vel_{wk})$ Obtaining specific values for variables from previously defined functions.

$It_{rec} := It(Vel_{rec})$ Subscripts wk = during work period
rec = during rest or recovery period

$Im_{work} := Evap_{cloth}(Vel_{wk})$

$Im_{rec} := Evap_{cloth}(Vel_{rec})$

$Vel_{wk} = 1.58$ $Vel_{rec} = 0.6$ Vel = total effective wind velocity across the body.

$It_{work} = 1.951$ $It_{rec} = 2.256$ It = total effective insulation.

$Im_{work} = 0.175$ $Im_{rec} = 0.144$ Im = total effective permeability.

$U_{wk} := Solar_{cloth\_efficiency}(Vel_{wk})$ U = efficiency of transfer of solar load to the skin.

$U_{rec} := Solar_{cloth\_efficiency}(Vel_{rec})$

$U_{wk} = 0.173$

$U_{rec} = 0.226$

$Bsa(cm, kg) := 0.007184 \cdot cm^{0.725} \cdot kg^{0.425}$ Bsa = body surface area in m2.

$Bsa := Bsa(Ht, Wgt)$

$Bsa = 1.825$

The following block contains additional function declarations.

$Hrc(Clo) := 6.45 \cdot Bsa \cdot \frac{(Ta - Tsk)}{Clo}$ Hrc = radiant and convective heat gain.

$Ereq(Hrc, Met, Solar) := Hrc + Met - Watts_{ext} + Solar \cdot SlrF \cdot CldF$ Ereq = heat that needs to be dissipated to prevent core temp increase.

$Emax(Cevap) := 14.21 \cdot Cevap \cdot Bsa \cdot (Torr\_h2o_{skin} - Torr\_h2o_{ambient})$ Emax = Maximum evaporative heat absorbing capacity of the ambient air

$Hrc_{work} := Hrc(It_{work})$

$Hrc_{rec} := Hrc(It_{rec})$

$Ereq_{work} := Ereq(Hrc_{work}, Watts_{work}, U_{wk})$

$Ereq_{rec} := Ereq(Hrc_{rec}, Watts_{rec}, U_{rec})$

$Emax_{work} := Emax(Im_{work})$

$Emax_{rec} := Emax(Im_{rec})$

$Watts_{work} = 350$

$Hrc_{work} = -9.049$

$Hrc_{rec} = -7.826$

$Ereq_{work} = 340.951$

$Ereq_{rec} = 97.174$

$Emax_{work} = 112.374$

$Emax_{rec} = 92.59$

Function declaration for Tref, which is the asymptotic predicted core temp.

$$Tref(M, U, Hrc, Ereq, Emax) := 36.75 + 0.004 \cdot M + 0.0025 \cdot U \cdot CldF \cdot SkF + 0.0011 \cdot Hrc + 0.8 \cdot \exp(0.0047 \cdot (Ereq - Emax))$$

$Tref_{wk} := Tref(Watts_{work}, U_{wk}, Hrc_{work}, Ereq_{work}, Emax_{work})$

$Tref_{rec} := Tref(Watts_{rec}, U_{rec}, Hrc_{rec}, Ereq_{rec}, Emax_{rec})$

Calculation of specific values for Tref for work and recovery phases of the work-rest cycle.

$Tref_{wk} = 40.482$

$Tref_{rec} = 37.979$

Dtref is a function that adjusts the predicted asymptotic core temp for acclimatization measured as days in heat (DiH). Note that the benefit of heat acclimatization can be negated by low Emax, which represents low evaporative heat acceptance capacity of the environment (e.g. as occurs in hot humid conditions).

$$DTref(Tref, Emax) := (0.5 + 1.2 \cdot (1.0 - \exp(0.5 \cdot (37.15 - Tref))) \cdot (1.0 - \exp(-0.005 \cdot Emax))) \cdot (\exp(-0.3 \cdot DiH))$$

$TrefA_{wk} := DTref(Tref_{wk}, Emax_{work})$ $\quad$ $TrefA_{rec} := DTref(Tref_{rec}, Emax_{rec})$

$TrefA_{wk} := if(Emax_{work} > 0, TrefA_{wk}, 0)$ $\quad$ $TrefA_{rec} := if(Emax_{rec} > 0, TrefA_{rec}, 0)$

$TrefA_{wk} = 1.16 \cdot 10^{-13}$ $\quad$ $TrefA_{rec} = 8.221 \cdot 10^{-14}$

The following block calculates recommended water intakes during work and recovery. Note that max water absorption is ~1.5 lts/hr therefore, sustained sweating at rates >1.5 lts/hr will lead to progressive dehydration irrespective of amounts of water intake

$$Water(Ereq, Emax) := if(Emax < 0, 2000, 27.9 \cdot Bsa \cdot Ereq \cdot Emax^{-0.455})$$

$$Sweat(Ereq, Emax) := if(Water(Ereq, Emax) > 2000, 2000, (if(Water(Ereq, Emax) \le 150, 150, Water(Ereq, Emax))))$$

$Water_{wk} := 1.0567 \cdot 10^{-3} \cdot Sweat(Ereq_{work}, Emax_{work})$

$Water_{rec} := 1.0567 \cdot 10^{-3} \cdot Sweat(Ereq_{rec}, Emax_{rec})$

$Water_{wk} = 2.113$

$Water_{rec} = 0.666$

$Delay_{wk} := \dfrac{3480}{Watts_{work}}$ Delay = time for temperature effects of starting work cycle to become apparent.

$Delay_{wk} = 9.943$

$$TreoC_{wk} := Treo + \left(Tref_{rec} + TrefA_{rec} - Treo\right)\cdot 0.1^{\left(0.4^{\frac{Delay_{wk} - 30.0}{60}}\right)}$$

Treo = initial core temp.
Trefrec = core temp at end of recovery stage.
TrefA = core temp correction for acclimatization.

note this is a double exponential term

$TreoC_{wk} = 37.043$

$$k_{wk} := \frac{1 - 3\cdot\exp\left[-0.3\cdot\left(TreoC_{wk} - Tref_{wk}\right)\right]}{120}$$

k = time constant for progression of core temp from initial to asymptotic.

$k_{wk} = 0.017$

$$CP := 0.015\cdot\left(Emax_{rec} - Ereq_{rec}\right)$$

$$Delay_{rec} := if(CP<0, 15, 15\cdot\exp(-0.5\cdot CP))$$

$$k_{rec} := \frac{1 - \exp(-1.5\cdot|CP|)}{40}$$

CP = cooling power (i.e., how much thermal energy can be lost via sweat evaporation given the ability of ambient air to absorb additional water vapor).

$CP = -0.0$[illegible]

$Delay_{rec} = 15$

$k_{rec} = 0.002$

$$\Delta Tre_{wk} := Tref_{wk} + TrefA_{wk} - TreoC_{wk}$$

$$\Delta Tre_{rec} := Tref_{rec} + TrefA_{rec} - TreoC_{rec}$$

$\Delta Tre_{wk} = 3.44$

$$Max\_work\_mins := Delay_{wk} - \frac{\ln\left(\frac{\Delta Tre_{wk} + TreoC_{wk} - Tre\_limit_{wk}}{\Delta Tre_{wk}}\right)}{k_{wk}}$$

(Eqn description for maximum single shot work time.

$$C := \frac{\Delta Tre_{wk} + TreoC_{wk} - Tre\_limit_{wk}}{\Delta Tre_{wk}}$$

$$Max\_work\_mins := Delay_{wk} - \left(\frac{\ln(if(\Delta Tre_{wk}>0, if(C>0, C, 0), 0))}{k_{wk}}\right)$$

implementation of the max work time eqn (if 1st term is true, then 2nd term, otherwise do the 3rd term).

$$Max\_work\_mins := if(Max\_work\_mins>300, 300, Max\_work\_mins)$$

$Max\_work\_mins = 58.758$

Time Series Results: Rest phase

$\Delta t = 5$ $\quad t_{final} = 300$ $\quad t_{rest} = 20$ $\quad t_{work} = 50$ $\quad t_{rec} = 30$

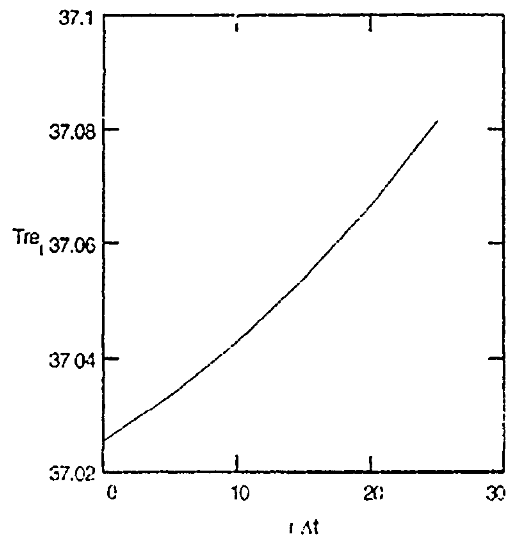
$tswitch_{rest} := t_{rest} + Delay_{wk}$

$tswitch_{rest} = 23.943$

$t := 0, \Delta t .. tswitch_{rest}$ $\qquad i = 0 .. \frac{tswitch_{rest}}{\Delta t}$

$\Delta Tre_{rec} := Tref_{rec} - TrefA_{rec} \cdot 90$

$$Tre_i = Treo + \Delta Tre_{rec} \cdot 0.1 \left(0.4 \frac{i \cdot \Delta t - 30}{60} \right)$$



t	Tre_i
0	37.026
5	37.034
10	37.043
15	37.054
20	37.067
25	37.082

Time Series Results: Work phase

$tswitch_{work} := t_{work} + Delay_{rec}$

$tswitch_{rest} = 29.943$ $\quad$ $tswitch_{work} = 65$

$t := tswitch_{rest}, tswitch_{rest} + \Delta t .. tswitch_{rest} + tswitch_{work}$

$$i_{last} := \frac{tswitch_{work}}{\Delta t}$$

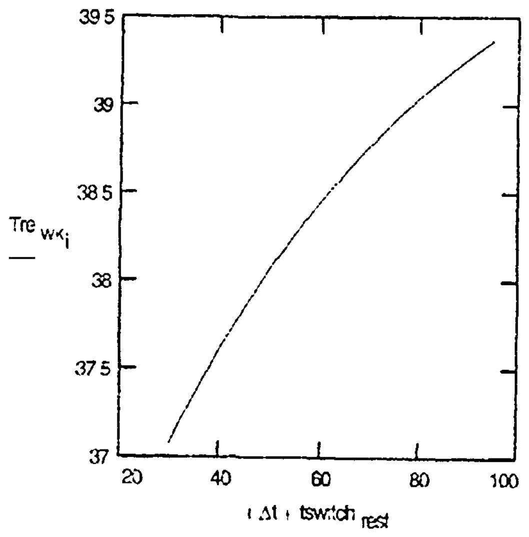
$i := 0 .. i_{last}$

$TreoC_{wk} := Tre_{last(Tre)}$

$\Delta Tre_{wk} := Tref_{wk} + TrefA_{wk} - TreoC_{wk}$

$\Delta Tre_{wk} = 3.401$

$$Tre_{wk_i} := TreoC_{wk} + \Delta Tre_{wk} \cdot \left(1 - \exp\left(-k_{wk} \cdot i \cdot \Delta t\right)\right)$$



t	Tre_{wk_i}
29.943	37.082
34.943	37.362
39.943	37.62
44.943	37.857
49.943	38.073
54.943	38.272
59.943	38.455
64.943	38.622
69.943	38.776
74.943	38.917
79.943	39.046
84.943	39.165
89.943	39.274
94.943	39.374

Time Series Results: Recovery Phase

$tt_i := t$

$t_{rec} = 30 \qquad Delay_{wk} = 9.943$

$tswitch_{rec} := t_{rec} + Delay_{wk}$

$tswitch_{rec} = 39.943 \qquad tswitch_{work} := tt_{i_{last}}$

$t := tt_{i_{last}}, tt_{i_{last}} + \Delta t .. tt_{i_{last}} + tswitch_{rec}$

$TreoC_{rec} := Tre_{wk_{last(Tre_{wk})}}$

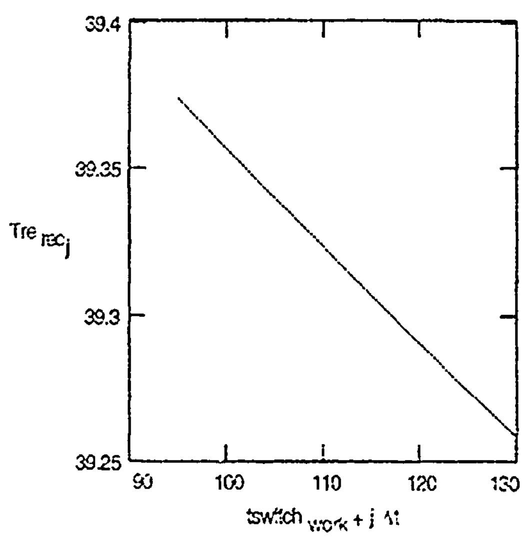
$\Delta Tre_{rec} := Tref_{rec} + TrefA_{rec} - TreoC_{rec}$

$\Delta Tre_{rec} = -1.395$

$i_{last} := \frac{tswitch_{rec}}{\Delta t} \qquad i_{last} = 7.989$

$j := 0 .. i_{last} \qquad Tre$

$Tre_{rec_j} := TreoC_{rec} + \Delta Tre_{rec} \cdot \left[1 - \exp\left[-k_{rec} \cdot (j \cdot \Delta t)\right]\right]$



t	Tre_{rec_j}
94.943	39.374
99.943	39.357
104.943	39.34
109.943	39.323
114.943	39.307
119.943	39.291
124.943	39.275
129.943	39.259

Respiratory Heat Loss as Percent of Total Required Heat Loss

(Fanger, 1970; Gonzalez and Cena, 1985)

Using the inputs from the previous section: Function definitions:

$E\_resp(M,P) \equiv 0.0023 \cdot M \cdot (44.0 - P)$	Evaporative resp heat loss.
$C\_resp(M,T) \equiv 0.0014 \cdot M \cdot (34.0 - T)$	Convective resp heat gain/loss. n.b.: 34C = 93.2 F

Work phase:

$$E\_resp_{work} := E\_resp\left(Watts_{work}, Torr\_h2o_{ambient}\right)$$

$$C\_resp_{work} := C\_resp\left(Watts_{work}, Ta\right)$$

$$net\_E\_resp_{work} := E\_resp_{work} + C\_resp_{work}$$

$$E\_resp_{work} = 18.444$$

$$C\_resp_{work} = -0.49$$

$$net\_E\_resp_{work} = 17.954$$

Recovery (rest) phase:

$$E\_resp_{rec} := E\_resp\left(Watts_{rec}, Torr\_h2o_{ambient}\right)$$

$$C\_resp_{rec} := C\_resp\left(Watts_{rec}, Ta\right)$$

$$net\_E\_resp_{rec} := E\_resp_{rec} + C\_resp_{rec}$$

$E\_resp_{rec} = 5.533$	This specific scenario is permitting evaporative respiratory heat loss.
$C\_resp_{rec} = -0.147$	The negative sign here indicates convective heat gain due to inhaling the relatively high temperature ambient air.
$net\_E\_resp_{rec} = 5.386$	The net result for this environmental, soldier, activity senario is a net ability to dissipate body heat via the respiratory tract at a rate of slightly more than 5 Watts per meter sq. of body surface area.

$\dfrac{Bsa \cdot net\_E\_resp_{work}}{Ereq_{work}} \cdot 100 = 9.611$	The percentage of accumulating body heat load during the work phase that can be dissipated via respiration
$\dfrac{Bsa \cdot net\_E\_resp_{rec}}{Ereq_{rec}} \cdot 100 = 10.117$	The percentage of accumulating body heat load during the recovery phase that can be dissipated via respiration.

$Watts_{work} = 350$

$Hrc_{work} = -9.049$

$Hrc_{rec} = -7.826$

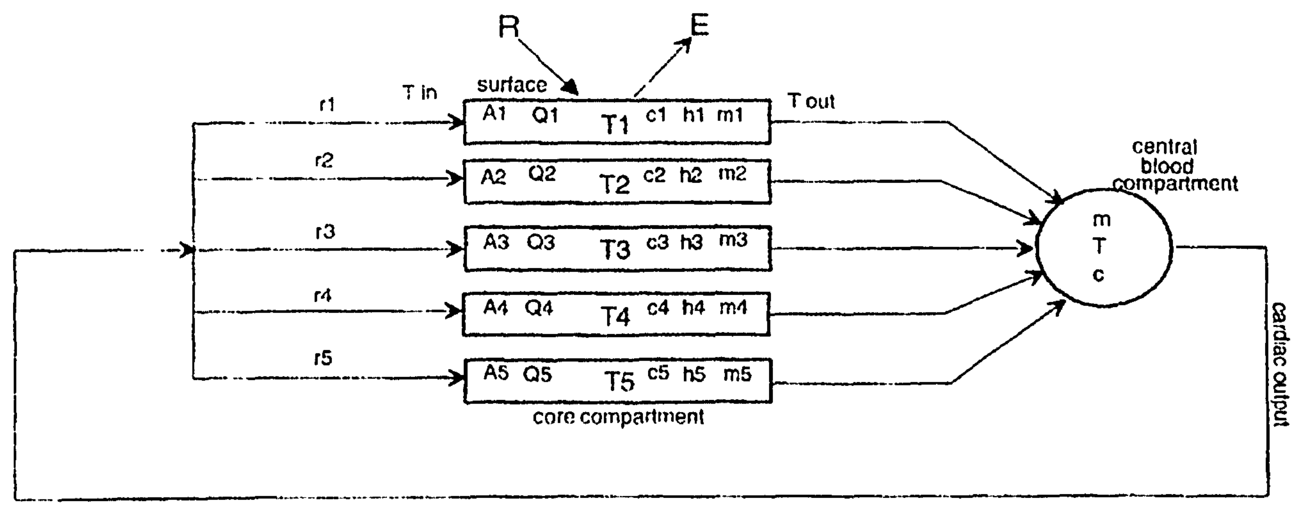
$Ereq_{work} = 340.951$

$Ereq_{rec} = 97.174$

$Emax_{work} = 112.374$

$Emax_{rec} = 92.59$

APPENDIX C: General Outline of a Lumped-Parameter Thermoregulatory Model



compartment #i
Ai = surface area
ci = specific heat in Joules/(kg*K)
hi = surface heat transfer coefficient in Joules/m^2*K)
mi= mass
Qi = metabolic heat production
ri = fraction (or ratio) of cardiac output
Ti = temperature
arrows = bloodflow

Schematic of the passive components of a lumped-parameter thermoregulatory model. The active component is the system of ancillary equations that modify the parameters of the passive components as functions of time and state.

Assumptions, rules, constraints, and relationships:

Tissue properties are spatially homogeneous, therefore, the response of all points within a tissue compartment are identical and can be modeled by one representative point. Hence the term: "lumped parameter".

Compartments representing the human body can be depicted as concentric cylinders, but the lumped-parameter model allows for other artistic representations as long as values for surface area and volume are specified rather than automatically determined via some boundary detection and measurement algorithm, such as might be available within a CAD software program.

Parameters in the system equations may be time-varying functions of inputs, state variables, and outputs. Parameters that are functions of compartmental temperatures (state variables) embody the mechanism for thermoregulatory feedback control.

Only the outer surface of compartment #1 interacts with the environment. h1 is the convective heat transfer coefficient from the surface of the most superficial compartment to the environment in $\frac{\text{Watts}}{m^2 K^\circ}$.

Conductive heat exchange occurs between the core and successively superficial layers or compartments. The conductive heat transfer coefficients for intercompartmental heat flow is expressed in terms of thermal conductivity (k) and distances from compartment centroids $h_{i,j} = \frac{k_{ij}}{d_{ij}}$. The calculation of intercompartmental conductive heat transfer coefficients is somewhat complicated, as explained in the details of the technical report by Kranning II (1991).

It is assumed that the diffusion of heat, between a compartment and the blood flowing through it, is sufficiently fast, relative to the blood's transit time through the compartment, that: T_blood_out[compartment i] = T[compartment i].

Mass blood flow rate through each compartment is: $r_i \cdot \rho_b \cdot co$ where r_i is the fraction of the cardiac output (co) going to compartment #i and $\Sigma r_i = 1$, but where some r_i may =0. ρ_b is the mass density of blood.

Cardiac output (co) can be expressed as the product of heart rate and stroke volume, $co(t,T,M) = hr(t,T,M) \cdot sv(t,T,M)$, each of which can be expressed as (nonlinear or piece-wise linear) functions of time and other variables (e.g., compartment temperatures and metabolic rate).

Stroke volume can be expressed as a function of the left ventricular end diastolic volume and ejection fraction: $sv(t) = r_{e,rt}(t) \cdot LVEDV(t)$.

Age adjusted maximum predicted hr and sv determines the upper limit for the co.

The total metabolic rate (MR) associated with a selected activity is partitioned into thermogenic effects for each compartment, Q_i. Q_i for nonmuscle compartments will be fractions of the basal metabolic rate. Q_i for the muscle compartment is: $Q_i = MR_i - External\_work_i$.

The compartmental lumped-parameter heat transfer equations are:

$$m_1 \cdot c_1 \cdot \frac{dT_1}{dt} = r_1 \cdot co \cdot \rho_b \cdot c_b \cdot (T_b - T_1) + A_2 \cdot h_2 \cdot (T_2 - T_1) - A_1 \cdot h_1 \cdot (T_1 - T_o) + Q_1 + R - E$$

$m_i \cdot c_i \cdot \frac{dT_i}{dt} = r_i \cdot co \cdot \rho_b \cdot c_b \cdot (T_b - T_i) + A_{i+1} \cdot h_{i+1} \cdot (T_{i+1} - T_i) - A_i \cdot h_i \cdot (T_i - T_{i-1}) + Q_i$ for $i = 2 \ldots N-2$ where N = # of compartments.

$m_c \cdot c_c \cdot \frac{dT_c}{dt} = r_c \cdot co \cdot \rho_b \cdot c_b \cdot (T_b - T_c) - A_c \cdot h_c \cdot (T_c - T_{N-2}) + Q_c$ where subscript c represents the core, or N-1, compartment.

$m_b \cdot c_b \cdot \frac{dT_b}{dt} = co \cdot \rho_b \cdot c_b \cdot (r_1 \cdot T_1 + r_2 \cdot T_2 + \ldots + r_c \cdot T_c - T_b)$ where subscript b represents the central blood, or Nth, compartment.

For a six compartment system (N=6) these equations can be expressed in matrix and vector format:

$$T \begin{bmatrix} -\left(\frac{r_1 c_0 \rho_b c_b + A_1 h_1 + A_2 h_2}{m_1 c_1}\right) & \frac{A_2 h_2}{m_1 c_1} & 0 & 0 & 0 & \frac{r_1 c_0 \rho_b c_b}{m_1 c_1} \\ \frac{A_2 h_2}{m_2 c_2} & -\left(\frac{r_2 c_0 \rho_b c_b + A_2 h_2 + A_3 h_3}{m_2 c_2}\right) & \frac{A_3 h_3}{m_2 c_2} & 0 & 0 & \frac{r_2 c_0 \rho_b c_b}{m_2 c_2} \\ 0 & \frac{A_3 h_3}{m_3 c_3} & -\left(\frac{r_3 c_0 \rho_b c_b + A_3 h_3 + A_4 h_4}{m_3 c_3}\right) & \frac{A_4 h_4}{m_3 c_3} & 0 & \frac{r_3 c_0 \rho_b c_b}{m_3 c_3} \\ 0 & 0 & \frac{A_4 h_4}{m_4 c_4} & -\left(\frac{r_4 c_0 \rho_b c_b + A_4 h_4 + A_5 h_5}{m_4 c_4}\right) & \frac{A_5 h_5}{m_4 c_4} & \frac{r_4 c_0 \rho_b c_b}{m_4 c_4} \\ 0 & 0 & 0 & \frac{A_5 h_5}{m_5 c_5} & -\left(\frac{r_5 c_0 \rho_b c_b + A_5 h_5 + A_6 h_6}{m_5 c_5}\right) & \frac{r_5 c_0 \rho_b c_b}{m_5 c_5} \\ \frac{r_1 c_0 \rho_b c_b}{m_6 c_6} & \frac{r_2 c_0 \rho_b c_b}{m_6 c_6} & \frac{r_3 c_0 \rho_b c_b}{m_6 c_6} & \frac{r_4 c_0 \rho_b c_b}{m_6 c_6} & \frac{r_5 c_0 \rho_b c_b}{m_6 c_6} & \frac{1 c_0 \rho_b c_b}{m_6 c_6} \end{bmatrix} \cdot T + \begin{bmatrix} \frac{(A_1 h_1 T_0) + Q_1 + R - E}{m_1 c_1} \\ \frac{Q_2}{m_2 c_2} \\ \frac{Q_3}{m3 c_3} \\ \frac{Q_4}{m_4 c_4} \\ \frac{Q_5}{m_5 c_5} \\ 0 \end{bmatrix}$$

This is of the form $\dot{T} = A \cdot T + B$. The solution for T when A is time invariant is: $T(t) = e^{At} \cdot T_{init} + \int e^{A(t-\tau)} \cdot B \cdot d\tau$ where T_{init} is the vector of initial compartment temperatures (see DeCarlo, 1989). To implement this solution, A should first be diagonalized (if it has N independent eigenvectors) using the linear transformation: $\Lambda = S^{-1} \cdot A \cdot S$ where S is a matrix whose columns are the eigenvectors of A (see Strang, 1988) . Then, Λ is a matrix with all zeros except for the eigenvalues of A on the diagonal in the same order as the matching eigenvectors in S. This solution approach is fine in theory however. as previously stated, due to implementation of thermoregulatory feedback, the elements of A are time-varying functions of the compartment temperatures. Hence, the time invariant solution technique can only be applied for short integration intervals where the elements of A are approximately constant. In reality, the solution is obtained by using a discrete time form of the equation. For example, if a first order Euler numerical approximation is used, the numerical solution takes the iterative matrix form:

$$T_{n+1} = T_n + \Delta t \cdot (A_n \cdot T_n + B_n) \Rightarrow T_{n+1} = (I + \Delta t \cdot A_n) \cdot T_n + \Delta t \cdot B_n \Rightarrow T_{n+1} = \hat{A}_n \cdot T_n + \hat{B}_n .$$

For the Euler approximation of the time-dependent compartment temperatures, the truncation error will be of the order Δt^2. Hence, truncation error can be contained by using appropriately small Δt . Higher order, predictor-corrector, or Runge-Kutta numerical methods may decrease truncation error at the expense of increased computational burden. If an analytic solution cannot be determined, higher-order numeric approximations of the true solution may be used to estimate the accuracy and stability of the less complex and faster Euler approximation.

APPENDIX D: Hypoxia Model in MathCAD

(Kessler, 1980)

$i = 0..15$

Inputs --> $Alt_{m_i} = 250 \cdot i$

$R = 0.8$ Respiratory quotient or resp. exchange ratio: CO2 production/O2 uptake

$FIO2 = 0.21$

$PACO2 = 40$ Since diffusion of CO2 through the alveolar membrane is very rapid, the PaCO2 and PACO2 do not significantly differ and are therefore used interchangeably.

$PaCO2 = 40$

$T_{core} = 37$ Body core temp in centigrade

$pH = 7.4$ Blood pH

$HCO_3 = 24$ Plasma bicarb level

$Hgb = 15$ Hemoglobin gms/100cc

$Hct = 45$ Hematocrit

For this altitude range the barometric pressure is

$$P_{mmHg} = \overrightarrow{\left[\left(1 - \frac{Alt_m}{44364.236}\right)^{5.256} \cdot 760\right]}$$

Arrow indicates vectorization (i.e., the operations in the eqn are performed on each element in the Alt vector)

Inspired O2

$$PIO2 = FIO2 \cdot \left(P_{mmHg} - 47\right)$$

$DO2_{Aa} = 5$ The alveolar pulmonary venous pO2 diffusion gradient, normally < 10 mmHG

The alveolar partial pressure of O2 is:

$$PAO2 = PIO2 - PACO2 \cdot \left(1 - \frac{FIO2 \cdot (1 - R)}{R}\right)$$

The arterial O2 partial pressure:

$$PaO2_i = \text{if}\left(PAO2_i > DO2_{Aa}, PAO2_i - DO2_{Aa}, 0.0\right)$$

Calculate an effective PaO2 to enable the use of standard Hb dissociation curve when core temp, blood pH, and PaCO2 are not at standard levels.

$$PaO2_{effective} = PaO2 \cdot 10^{0.024\cdot(37 - T_{core}) + 0.4\cdot(pH - 7.4) + 0.06\cdot(\log(40) - \log(PaCO2))}$$

$$u := \overrightarrow{\left(0.00925 \cdot PaO2_{effective} - 0.00028 \cdot PaO2_{effective}^{2} + 0.0000306 \cdot PaO2_{effective}^{3}\right)}$$

$$O2sat_i = \text{if}\left[PaO2_i \geq 10, \frac{u_i}{1 - u_i}, 0.003683 \cdot PaO2_{effective_i} - 0.000584 \cdot \left(PaO2_{effective_i}\right)^{2}\right]$$

$$O2_{Hgb} = 1.39 \cdot Hgb \cdot O2sat$$

$O2_{Hg}$ cc O2/100cc blood

To determine dissolved O2.

$$Sol_{coef} = 0.0059519 - 0.0001266 \cdot T_{core} - 0.0000013 \cdot T_{core}^{2}$$

$$Sol_{coef} = 0.003$$

$$O2_{dissolved} = Sol_{coef} \cdot PaO2$$

$O2_{dissolved}$ cc O2/100cc blood

Total O2 carried:

$O2_{capacity} = O2_{Hgb} + O2_{dissolved}$ cc O2/100cc blood

About 23% of blood CO_2 combines with Hgb to form carbaminohemoglobin ($HbCO_2$)

- 7% is in the dissolved form
- 70% is in the form of bicarbonate ions (HCO_3^-)

$pK_1 := 6.1$

$CO2_{blood} := 0.03 \cdot PaCO2$ $\qquad$ $CO2_{blood} = 1.2$

$pH := pK_1 + \log\left(\frac{HCO_3}{CO2_{blood}}\right)$ $\qquad$ $pH = 7.401$

$pK := 6.086 + 0.042 \cdot (7.4 - pH) + (38 - T_{core}) \cdot (0.0047 + 0.0014 \cdot (7.4 - pH))$

$pK = 6.091$

$CO2_{solubility} := 0.0307 + 0.00057 \cdot (37 - T_{core}) + 0.00002 \cdot (37 - T_{core})^2$

$CO2_{solubility} = 0.031$

$CO2_{blood} := CO2_{solubility} \cdot PaCO2 \cdot \left[1 + 10^{(pH - pK)}\right]$

$CO2_{blood} = 26.322$ $\quad$ cc CO2/100cc blood

$FOB := 0.590 - 0.2913 \cdot (7.4 - pH) - 0.0844 \cdot (7.4 - pH)^2$

$FOB = 0.59$

$FRB := 0.664 + 0.2275 \cdot (7.4 - pH) - 0.0938 \cdot (7.4 - pH)^2$

$FRB = 0.664$

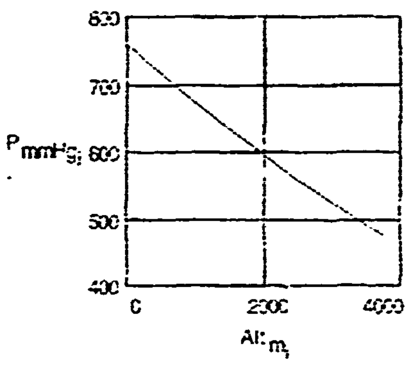
The cell (rbc) to plasma ratio (CPRAT) for dissolved CO2 is given by

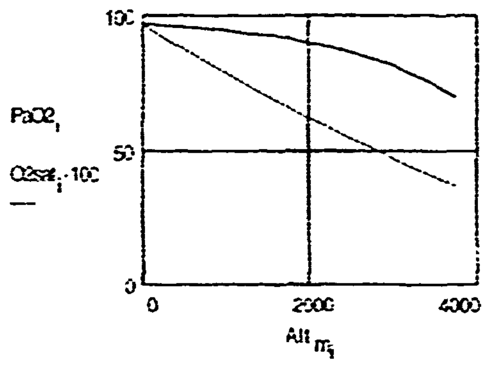
$CPRAT := FOB + (FRB - FOB) \cdot (1 - O2sat)$

$CO2_{rbc} := CPRAT \cdot CO2_{blood}$ $\quad$ cc CO2/100cc blood

$CO2_{plasma} := \frac{Hct}{100} \cdot CO2_{rbc} + \left(1 - \frac{Hct}{100}\right) \cdot CO2_{blood}$ $\quad$ cc CO2/100cc blood meters

Alt_{m_i}	P_{mmHg_i}	$PIO2_i$	$PaO2_i$	$O2sat_i \cdot 100$	$O2_{capacity_i}$	$O2_{dissolved_i}$	$CO2_{rbc_i}$	$CO2_{plasma_i}$
0	760	149.73	96.83	96.904	20.5	0.295	18.287	22.706
250	737.758	145.059	92.159	96.452	20.391	0.281	18.406	22.76
500	716.047	140.5	87.6	95.923	20.267	0.267	18.545	22.822
750	694.855	136.05	83.15	95.304	20.124	0.253	18.708	22.896
1000	674.174	131.707	78.807	94.578	19.96	0.24	18.899	22.962
1250	653.995	127.469	74.569	93.724	19.769	0.227	19.124	23.083
1500	634.308	123.335	70.435	92.718	19.546	0.215	19.388	23.202
1750	615.103	119.302	66.402	91.531	19.237	0.202	19.701	23.343
2000	596.371	115.368	62.468	90.128	18.982	0.19	20.07	23.509
2250	578.105	111.532	58.632	88.469	18.625	0.179	20.507	23.705
2500	560.294	107.792	54.892	86.509	18.204	0.167	21.023	23.937
2750	542.93	104.145	51.245	84.195	17.711	0.156	21.632	24.211
3000	526.004	100.591	47.691	81.472	17.132	0.145	22.349	24.534
3250	509.508	97.127	44.227	78.279	16.456	0.135	23.189	24.912
3500	493.434	93.751	40.851	74.563	15.671	0.124	24.167	25.352
3750	477.773	90.462	37.562	70.275	14.767	0.114	25.296	25.66





$i = 0..20$

$PIO2_i = 10 \cdot i$

The alveolar partial pressure of O2 is:

$$PAO2 - PIO2 \quad PACO2 \cdot \frac{1 - FIO2 \cdot (1 \quad R)}{R}$$

The arterial O2 pp:

$$PaO2 \cdot if\left(PAO2_i > DO2_{Aa}, PAO2_i - DO2_{Aa}, 0.0\right)$$

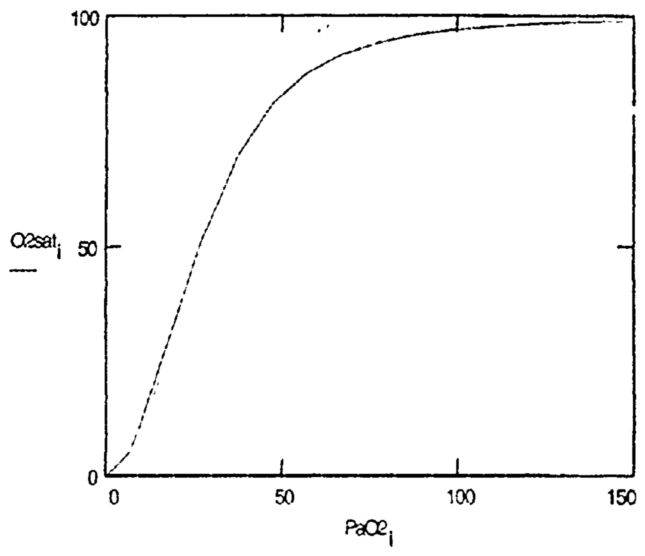
$$PaO2_{effective} = PaO2 \cdot 10^{0.024 \cdot (37 - T_{core}) + 0.4 \cdot (pH - 7.4) - 0.06 \cdot (\log(40) - \log(PaCO2))}$$

$$u = \overrightarrow{\left(0.00925 \cdot PaO2_{effective} + 0.00028 \cdot PaO2_{effective}^{2} + 0.0000306 \cdot PaO2_{effective}^{3}\right)}$$

$$O2sat_i = \mathrm{if}\left(PaO2_i \geq 10, \frac{u_i}{1 - u_i}, 0.003683 \cdot PaO2_{effective_i} - 0.000584 \cdot \left(PaO2_{effective_i}\right)^2\right) \cdot 100$$

$PIO2_i$	$PaO2_i$	$O2sat_i$	$PaO2_{effective_i}$	$\frac{u_i}{1 - u_i}$
0	0	0	0	0
10	0	0	0	0
20	0	0	0	0
30	0	0	0	0
40	0	0	0	0
50	0	0	0	0
60	7.1	5.567	7.107	0.063
70	17.1	26.254	17.116	0.283
80	27.1	51.637	27.126	0.516
90	37.1	69.666	37.135	0.697
100	47.1	81.006	47.145	0.81
110	57.1	87.74	57.154	0.877
120	67.1	91.773	67.164	0.918
130	77.1	94.266	77.173	0.943
140	87.1	95.87	87.183	0.959
150	97.1	96.936	97.192	0.969
160	107.1	97.669	107.202	0.977
170	117.1	98.169	117.211	0.982
180	127.1	98.566	127.221	0.986
190	137.1	98.846	137.23	0.988
200	147.1	99.058	147.24	0.991

The O2 saturation curve as given by the preceeding equations:



APPENDIX E

ANNOTATED LIST OF IEEE SOFTWARE DEVELOPMENT STANDARDS

Software developers should be familiar with general contents and scope of the specific project relevant software standards (Solomond, 1995). As defined by the Microsoft Press Computer Dictionary (Doyle, 1994), computer standards are:

> ... a set of detailed technical guidelines used as a means of establishing uniformity in an area of hardware or software development. Computer standards have traditionally developed in either of two ways. The first, a highly informal process, occurs when a product or philosophy is developed by a single company and, through success and imitation, becomes so widely used that deviation from the norm causes compatibility problems or limits marketability. ... The second type of standard setting is a far more formal process in which specifications are drafted by a cooperative group or committee after an intensive study of existing methods, approaches, and technological trends and developments. The proposed standards are later ratified or approved by a recognized organization and are adopted over time by consensus as products based on the standards become increasingly prevalent in the market.

Well-known and highly respected professional organizations that promulgate software standards include the American National Standards Institute (ANSI), the International Standards Organization (ISO), and the Institute of Electrical and Electronics Engineers (IEEE). These software development standards are independent of specific applications and programming languages. They provide general guidelines, recommendations, and suggestions for methodologies, documentation, processes, organization, resourcing, and task decomposition for all phases of the software life cycle.

The majority of IEEE software standards are not lengthy and, contrary to what might be expected, not difficult to read. This is due to the generality of the standards, and their logical organization, straightforward definition of terms, and liberal use of diagrams, tables, and examples to enhance understanding of the text. These standards are not ponderous and can be quickly reviewed. They often include checklists to expedite rapid assessment of the various components of the control and quality effort comprising software design and documentation.

Below is an annotated list of many of the current IEEE software development, maintenance, and testing standards (The Institute of Electrical and Electronics Engineers, Inc., 1994).

- IEEE Std 610.121990
  (Revision and redesignation of IEEE Std 729-1983)
  Standard Glossary of Software Engineering Terminology (ANSI)

- IEEE Std 730-1989
  (Revision of IEEE Std 730-1984 and redesignation of IEEE Std 730.1-1989
  Standard for Software Quality Assurance Plans (ANSI)
  This standard provides guidance for the contents of a software quality assurance plan for the development and maintenance of critical software. Critical software is software used in such manners or locations in systems that errors or failures may lead to injury, damage, or significant financial or social losses.

- IEEE Std 828-1990
  Standard for Software Configuration Management Plans (ANSI)
  This standard provides guidance for es.ablishing the contents of a software configuration management p' in. Configuration management involves the identification, control, and documentation of the critical elements of a software project. Critical software project elements include design documents, performance and technical specifications, data structures, source code, and user manuals. Configuration management controls and documents the most current status and history of a software project. This facilitates the coordination of diverse software development resources in order to efficiently achieve customer-driven specifications, identify and control costs, assist with troubleshooting, and develop training and user manuals. An important configuration management control technique employs change requests. With this method, development team members must obtain approval of change requests from the project or team leader prior to deviating from established design, implementation, or documentation guidance. If change requests are approved, a configuration management auditing process ensures that all secondary changes due to the primary change are made. For example, a change in software outputs requires that this change be reflected in the design and user documents.

- IEEE Std 829-1983 (Reaff 1991)
  Standard for Software Test Documentation (ANSI)
  This standard provides guidance for developing the contents of a test plan, test-design specification, test-case specification, test-procedure specifications, test-item transmittal report, test log, test incident report, and test summary report. Appendices provide an example of each of these test control documents.

- IEEE Std 830-1993
  Guide to Software Requirements Specifications (SRS) (ANSI)
  This standard provides guidance for developing a document that specifies the functional and performance requirements as well as the attributes, design constraints, interfaces, and responses of a proposed software product or program. Interface functionality and

performance requirements should be specified for interfaces with other systems, the user, hardware components, other software, and communication ports and protocols. Software performance requirements should be assessed for the following properties: completeness, consistency, importance ranking, probability of change, verifiability, modifiability, and tracibility. Eight alternative methods of organizing the requirements in the SRS are described. The SRS is developed with the customer and is tracked in the configuration management process. Rapid prototyping may facilitate delineation of requirements, feasibility, and risk assessment. The SRS should only specify targets for performance and functionality; not the development processes that are elaborated in separate design documents.

- IEEE Std 982.1-1988

  Standard Dictionary of Measures to Produce Reliable Software (ANSI)

  Defines thirty-nine software reliability metrics primarily in terms of functions of the number of detected errors, faults, and failures; completeness and consistency; complexity; and risks, benefits, and costs. These measures are categorized in terms of the product itself or the product development and maintenance process. Reliability measures are defined with respect to the type of application (what it is useful for), primitives (what data elements are required), and implementation (how to process and present the data).

- IEEE Std 982.2-1988

  Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software (ANSI)

  This standard provides a constructive approach to embedding reliability into software products at every step of the software product life cycle. Measures of reliability are functionally classified into product and process measures. The product life-cyle stages at which these measures of reliability are most relevant is also delineated. Recommendations are provided for the administrative, monitoring, and reliability measurement structures, procedures and documentation. The dynamics of faults and errors are discussed. Eleven figures are provided that graphically depict software reliability models, strategies, interrelationships of causative factors for errors and faults, and process control schemes. Additionally, there are tables for measure classification, complexity assessment, reliability management control, and risk-benefit and cost evaluation.

- IEEE Std 990-1987 (Reaff 1992)

  Recommended Practice for Ada* as a Program Design Language (ANSI)

  This standard provides high-level recommendations for programming design languages (PDL) that use Ada programming language constructs. Ada-oriented PDLs are used to develop data and object structures, data value constraints, permissible operations on objects, and algorithm specifications and designs primarily, but not exclusively, to support Ada implementations.

- IEEE Std 1002-1987 (Reaff 1992)

  Standard Taxonomy for Software Engineering Standards (ANSI)

  This standard provides basic and comprehensive ordered classifications of standards according to functional and lifecycle aspects of the software development process. The basic types of standards are process, product, professional, and notational. These taxonomies provide application independent frameworks for assessing, categorizing, and comparing the content and scope of different software standards

- IEEE Std 1008-1987 (Reaff 1993)

  Standard for Software Unit Testing (ANSI)

  This standard provides a recommended process for requirements-based testing of software modules. The process is defined in terms of phases, activities, and tasks. General and specific test plans should identify the input-output data elements that need to be measured and the tasks that the software tester needs to perform on the test data.

- IEEE Std 1012-1986 (Reaff 1992)

  Standard for Software Verification and Validation Plans (ANSI)

  This standard provides recommendations for the contents of a software verification and validation (V&V) plan. V&V plans should include the V&V organizational structures, schedules, resource requirements, partitioning of responsibilities. and techniques and tools. Several figures and tables illustrate how V&V tasks relate to the different phases of the software life-cycle model.

- IEEE Std 1016-1987 (Reaff 1993)

  Recommended Practice for Software Design Descriptions (ANSI)

  This standard provides recommended contents for documenting the design of software modules and their data and related processes. Contents should explain and illustrate how project specifications were decomposed into modules of data structures and functions, explanations of the functions and their argument lists (interfaces), and how modules are interrelated. A detailed design section for each module defines data elements and the algorithms utilized in the different functions.

- IEEE Std 1016.1-1993,
  Guide to Software Design Descriptions
  This standard provides alternative software module design representation methods based on the selected design method. A design method may be preferentially oriented toward data structures, function, objects, programming design language, or real-time capabilities. Each design method should specify a module's identification, type, purpose, data, functions, subordinates, dependencies, interface, and resources. Module descriptions should be scaleable from high-level, low-resolution views to low-level, high-resolution views. Design representations include structure charts, data flow diagrams, data dictionaries, data flow context diagrams, process specifications, pseudo code, state transition diagrams, transform specifications, screen layouts, and requirements traceability matrix. Terms are defined.

- IEEE Std 1028-1988,
  Standards for Software Reviews and Audits
  This standard provides guidelines for conducting project management, technical, and software reviews, inspections and audits. Reviews and inspections evaluate the software development plan, design documents, and source code to detect errors, inconsistencies, and incompleteness. Audits determine compliance with suggested changes. Responsibilities of the review and audit team members and the processes that they should follow are delineated.

- IEEE Std 1042-1987,
  Guide to Software Configuration Management
  This ninety page standard suggests the management structure, processes, and tools for controlling the design, implementation, testing, and maintenance of software products. A configuration plan must first be established. Its complexity will be determined by the nature and extent of the proposed software. Large projects require a highly structured approach, while configuration management for small or proof of concept projects may be primarily informal. Appendices provided sample configuration management plans for these two types of projects.

- IEEE Std 1044-1993,
  Standard for Classification of Software Anomalies
  This standard applies to the detection, investigation, correction, and administrative activities associated with errors or inconsistencies in specifications, design documents, source code, software performance or other aspects of a software program. Anomalies can be prioritized within the following categories of potential adverse effects: severity, customer value, mission safety, project schedule, project cost, project risk, quality and reliability, and societal. An example for an anomaly reporting data base system is presented in the appendix.

- IEEE Std 1045-1992,
  Standard for Software Productivity Metrics (ANSI)
  This standard defines measures of productivity and consistent ways to describe or categorize software projects. Productivity is the ratio of output to input. Output often is comprised of lines of code or text Input typical has units of person-hours. The standard provides guidelines for how to count lines of code or documentation. Input and output may be classified as deliverable, nondeliverable, or supporting. Software projects may be characterized according to the qualifications and experience of the development team members, complexity, criticality, innovativeness, design and developmental concurrency, and extent of user participation. These characteristics can facilitate the objective comparison of different software projects.

- IEEE Std 1058.1-1987 (Reaff 1993),
  Standard for Software Project Management Plans (ANSI)
  This standard identifies the organization and contents for a project management plan. The following should be provided: an introduction, project overview deliverables, project management structure, responsibilities. process model with milestones, methods, tools and techniques, software documentation. schedule, budget, references, and definitions.

- IEEE Std 1059-1993
  Guide for Software Verification and Validation Plans (ANSI)
  This standard provides recommendations for the structure and contents of a software verification and validation plan (SVVP). The intent of a SVVP is to provide the guidelines for a quality assurance process that detects actual or potential software errors. ensures that the performance specifications are achieved and that the product is safe

when used for critical applications. The SVVP also defines the management structure. allocation of roles and responsibiliti s, and resources required for V&V. It also indicates how software changes will retroactively impact on the V&V effort. The components of a sample SVVP are provided in the appendices.

- IEEE Std 1061-1992

  Standard for a Software Quality Metrics Methodology (ANSI)

  This standard provides a methodology and discusses issues to consider when developing measurement instruments for software quality. The relevant quality related factors and subfactors are determined and prioritized. They are implemented and results analyzed. That the quality metrics actually measure what they purport to measure and are effective in improving software quality must be validated to ensure their relevance and effective use.

- IEEE Std 10621993

  Recommended Practice for Software Acquisition (ANSI)

  This standard first reviews the phases (planning. contracting, product implementation, product acceptance. and follow-c-) and milestones in the software acquisition cycle. A nine-step flexible algorithm is then presented for acquiring quality software. An organizational strategy is suggested for controlling the software acquisition process. Specific activities include assigning roles and responsibilities, delineating software requirements. surveying suppliers and available products, soliciting RFPs, and negotiating contracts. The selected supplier is subsequently monitored for product quality and adherence to specifications. A software acceptance plan is implemented and user satisfaction evaluated. Various checklists are provided in the appendices for use in the different phases of the software acquisition process.

- IEEE Std 1063-1987 (Reaff 1993)

  Standard for Software User Documentation (ANSI)

  This standard recommends a format and general content for user manuals. It requires analysis of the type of software and its interface properties and functionalities. The relevant characteristics of the main document user groups must be explicitly identified and utilized in designing the documents. The usage mode (instructional versus informational) also has considerable impact on content, format. and organization. A list of thecomponents for a user manual is also provided.

- IEEE Std 1074-1991

  Standard for Developing Software Life Cycle Processes (ANSI)

  This standard provides guidance for a comprehensive software development process. A software life-cycle model (SLCM) is selected that is appropriate and efficient for the particular software to be developed. A hierarchy of necessary activities are defined for various process categories (e.g., project management, development, verification and validation, operations and support, documentation, installation, training, maintenance, retirement, and configuration management). Requisite inputs, standards, taskings and resources, outputs, and performance metrics are assigned to each activity. The selected SLCM serves as the framework for organizing, coordinating, tracking, and controlling all these activities.

- IEEE Std 1219-1992

  Standard for Software Maintenance (ANSI)

  This standard provides a logical structure for post-development software changes. Problems or upgrade suggestions are identified, classified, and prioritized. Modification requests (MR) are submitted and organized into groups of related MRs. The MRs are analyzed with respect to feasibility and risk, implementation strategies, and testing. Each of the major steps for software maintenance is described in terms of inputs, processes, controls, and outputs.

- IEEE Std 1228-1994

  Standard for Software Safety Plans

  This standard provides process, control, and documentation recommendations for assuring that software does not pose a safety risk. The safety issue must be integrated into the software development effort from its inception. Safety requirements must be defined in the context of the entire system and operational environment(s). Safety critical equations, algorithms, data constraints, functionalities, and timing must be identified. The software is designed and implemented to satisfy the safety requirements. These are documented, as are the test results.

*Ada is a Registered trademark of the U.S. Government (Ada Joint Program Office)

APPENDIX F: FLDMED DATA STRUCTURES

```
/**************************************************************
          Data definitions and function declarations
           to support the heat strain module
**************************************************************/
enum phases_enum
  {
   rest=0,
   work=1,
   recovery=2,
  };

enum activity_enum{ marching =1,
                  task_list=2,
                  user_defined=3};

enum forms_enum {allinput_form,
                clothing_form,
                environ_form,
                pers_form,
                activity_form,
                marching_form,
                data_set_form,
                nutrition_form,
                threshholds_form};

enum graph_type_enum {tre_graph,hr_graph,sweatrate_graph,totsweat_graph,
                  casualty_bar_chart,casualty_pie_chart,misc};

enum option_choice_enum { one_graph_per_scr=1,
                        four_graphs_per_scr=2,
                        exit_options=3 };

struct nutrition_struct
  {
   int num_mres;
   float sodium_mg;
   float food_h2o;
   float cals;
   float wgt_gms;
  };

struct nutritionbuff_struct
  {
   char num_mres[3];
   char sodium_mg[7];
   char food_h2o[7];
   char cals[7];
   char wgt_gms[7];
  };
```

```
struct input_struct
  {
   int num_troops;
   float hgt;
   float wgt;
   float days_accl;
   enum activity_enum activity_type;
   int activity_list_num;
   float met_rate;
   float march_mph;
   float grade;
   float load;
   float miles;
   float tot_work_mins;
   float tot_allowed_mins;
   float work_mins;
   float rest_mins;
   float qts_water_per_hr;
   float temp;
   int solar;
   float tre_init;
   float avg_skin_temp;
   float tre_max_allowed;
   float hr_init;
   float init_dehydration; /* percent */
   float rh;
   float wind_mph;
   float percent_max_solar_load;
   float terrain_coef;
   int clothing_choice;
   float clo_itc;
   float clo_coef_itvc;
   float evap_perm_imc;
   float evap_perm_coef_imvc;
   float output_interval_mins;
   float init_rest_dur;
   float last_rec_dur;
/** float altitude;
   float mg_pyrido_q6hrs; **/
   struct nutrition_struct nutrition;
  };


struct inputbuff_struct
  {
   char num_troops[10];
   char hgt[7];
   char wgt[7];
   char days_accl[7];
   char activity[14];
   char met_rate[7];
/** char mg_pyrido_q6hrs[7]; **/
   char march_mph[5];
   char grade[5];
```

```
     char miles[7];
     char load[7];
     char tot_work_mins[7];
     char tot_allowed_mins[7];
     char work_mins[7];
     char rest_mins[7];
     char qts_water_per_hr[5];
     char temp[7];
     char solar[15];
     char tre_max_allowed[7];
     char tre_init[7];
     char avg_skin_temp[7];
     char hr_init[7];
     char init_dehydration[5];
     char rh[7];
     char wind_mph[7];
     char percent_max_solar_load[7];
     char terrain_coef[5];
     char clothing[35];
     char clo_itc[7];
     char clo_coef_itvc[7];
     char evap_perm_imc[7];
     char evap_perm_coef_imvc[7];
     struct nutritionbuff_struct nutritionbuff;
    };

struct threshhold_struct
    {
     float tre;
     float del_tre;
     float hr;
     float dehydration;
     char tre_buff[7];
     char del_tre_buff[5];
     char hr_buff[7];
     char dehydration_buff[5];
    };

struct clothing_struct
    {
     int menu_num;
     char name[35];
     float clo_itc;
     float clo_coef_itvc;
     float evap_perm_imc;
     float evap_perm_coef_imvc;
    };

struct activity_struct
    {
     int menu_num;
     char name[35];
     float watts;
    };
```

```
struct phasedata_struct
      {
       enum phases_enum phase;
       float phase_dur;
      } :

struct options_struct
      {
       int batch;
       int current_run_num;
       int output_file_units:
       int num_graphs_per_scr;
       enum graph_type_enum graph_type:
      };

struct output_struct
      {
      int  num_points;
      enum phases_enum phase[300];
      float t[300];
      float tre[300];
      float tre_max;
      float tre_max_at_hr;
      float hr[300];
      float hr_max;
      float hr_max_at_hr;
      float sweatrate[300];
      float sweatrate_max;
      float sweatrate_max_at_hr;
      float cumulative_sweat[300]:
      float net_its_water_loss[300];
      float percent_dehydrated:
      float sweat_Na_mg;
      float percent_heat_casualties:
      float num_heat_casualties;
      float qts_water_drank;
      float min_tot_drinking_water_required;
      float tot_hrs_required;
      float max_work_mins:
      float max_rec_mins:
      float num_cycles:
      float environ_cooling_pow;
      }:

struct io_data_struct
  {
      struct input_struct input[4];
      struct inputbuff_struct inputbuff[4]:
      struct threshhold_struct threshhold:
      struct options_struct options;
      int    calculated[4];  /* 0=not since last entry. 1=up to date */
      struct output_struct output[4];
  }:
```

```
/*****************************************************************
   Data definitions for the altitude module
*****************************************************************/

#include <vid.h>
#define lgt_blue  FG_IIFG_B
#define lgt_green FG_IIFG_G
#define yellow   FG_IIFG_RIFG_G
#define lgt_red  FG_IIFG_R
#define red      FG_R
#define white    FG_IIFG_RIFG_GIFG_B
#define magenta  FG_RIFG_B


#define yes 1
#define no  0

#define MAX_EVENTS 10
#define MAX_OUTPUT_ELEMENTS 30
#define NUM_SETS 4
#define NUM_GRAPHS 4 /* Tot_P, PIO2, PaO2, O2Sat see Alt_graf.c */

struct event_struct
   {
    int type_index;
    int day1;
    int day2;
    int alt1;
    int alt2;
    char typebuff[9];
    char day1buff[5];
    char day2buff[5];
    char alt1buff[7];
    char alt2buff[7];
   };

struct input_profile_struct
  {
   struct event_struct event[MAX_EVENTS];
   int num_events;
   int current_event;
  };

struct input_parameter_struct
 {
  float hgb;
  float hct;
  float pH;
  float HCO3;
  float tre;
  float resp_ratio;
  float FIO2;
  float PACO2;
  float Aa_diffusion_grad;
  char hgb_buff[7];
  char hct_buff[7];
  char pH_buff[7];
  char HCO3_buff[7];
  char tre_buff[7];
  char resp_ratio_buff[7];
  char FIO2_buff[7];
  char PACO2_buff[7];
  char Aa_diffusion_grad_buff[7];
  int calculated;
 };

enum output_type_enum
 {
  Pmbar=1,
  PmmHg=2
 };

struct input_struct
 {
  struct input_profile_struct profile;
  struct input_parameter_struct parameters[4];
 };

struct output_struct
 {
  char *diffmsg;  /* one liner wrt which inputs are different across sets */
  enum output_type_enum output_type;
  int num_vals;
  float x_val[NUM_SETS][MAX_OUTPUT_ELEMENTS];
  float y_val[NUM_GRAPHS][NUM_SETS][MAX_OUTPUT_ELEMENTS];
 };

/***************************************************************************
```

```
/**********************************************************************
        Data definitions for the cold digit module
**********************************************************************/

struct input_struct
  {
   float length_cm;
   float z_incr_mm;
   float dia_cm;
   float Tdb_C;
   float T_thresh_C;
   float time_init_hrs;
   float time_stop_hrs;
   float mins_incr;
   float T_base_init_C;
   float T_base_final_C;
   float T_base_time_const_hrs;
   float T_tip_init_C;
   float met_rate_init;    /* Watts/m^3 */
   float met_rate_final;   /* Watts/m^3 */
   float met_rate_time_const_hrs;
   int num_eigvals;
   float h_circ;  /* heat transfer coef at cylinder circumferential surface
                        Watts/(m^2*C)    */
   float h_tip;
   float k;      /* avg thermal conductivity of digit tissue W/(m*C)  */
   float LL;
   float area;
   float Bi;     /* Biot modulus  */
   float tau;
   float rho;
   float d;
  };


struct inputbuff_struct
  {
   char length_cm[5];
   char z_incr_mm[5];
   char circ_cm[5];
   char dia_cm[5];
   char Tdb_C[6];
   char T_thresh_C[5];
   char time_init_hrs[6];
   char time_stop_hrs[6];
   char mins_incr[5];
   char T_base_init_C[5];
   char T_base_final_C[5];
   char T_base_time_const_hrs[5];
   char T_tip_init_C[5];
   char met_rate_init[8];    /* Watts/m^3 */
   char met_rate_final[8];  /* Watts/m^3 */
   char met_rate_time_const_hrs[5];
```

```
 int num_eigvals[3];
 char h_circ[5];  /* heat transfer coef at cylinder circumferential surface
                        Watts/(m^2*C)    */
 char h_tip[5];
 char k[6];      /* avg thermal conductivity of digit tissue W/(m*C)  */
};

  struct options_struct
        {
         int batch;
         int current_run_num;
         int output_file_units;
         int num_graphs_per_scr;
         enum graph_type_enum graph_type;
        };

 struct io_data_struct
   {
        struct input_struct input[4];
        struct inputbuff_struct inputbuff[4];
        struct options_struct options;
        int    calculated[4];   /* 0=not since last entry, 1=up to date */
 /** struct output_struct output[4];  **/
   };
```

Default Input Values

I. Altitude Module:

Hemoglobin (gms/100cc): 14.5
Hematocrit (%): 45.0
Blood pH: 7.4
HCO_3^-: 24.0
Blood temp (F): 98.8
Respiratory ratio: 0.8
FIO2 (%): 21.0
PACO2: 40.0
Aa diffusion gradient (mmHg): 5.0

Default ascent profile:

Event	Alt1 (ft)	Alt2(ft)	from Day1	to Day2
Ascend	0	10,000	1	5
Remain	10,000	10,000	5	15
Descend	10,000	0	15	20

II. Heat Stress Module:

Number of troops: 100
Average height (inches): 70.0
Average weight (lbs): 160.0
Days acclimatized: 0.0
Initial core temp (F): 98.9
Average skin temperature (F): 96.8
Initial dehydration (%): 1.24
Clothing: BDU
  Total insulation (clo): 1.25
  Clo air velocity correction coefficient: -0.20
  Evaporative permeability: 0.38
  Evap perm air vel correction coefficient: 0.38
Dry bulb temp (F): 95.0
Relative humidity (%): 20.0
Wind speed (mph): 2.0
Solar exposure during work: Full sun
Solar exposure during rest: Shade
Metabolic rate (watts): 247.0
 Activity description: Light
 If 'Marching' is the selected activity, met rate is calculated using the following defaults:
  Rate of march (mph): 3.0
  Distance (miles): 12.0
  Grade (%): 2.0

Load (lbs): 20.0
Total amount of work required (mins): 240.0
Max allowed time to complete the work (mins): 500.0
Work-rest cycle   Work (mins): 50.0
                   Rest (mins): 10.0
Drinking water (qts/hr): 0.0
Meals per day: 3
Sodium per meal (mg): 1822.0
Water content per meal (cc): 221.0
Calories per meal: 1343
Weight per meal (gms): 463.0
Thresholds:
  Core temperature (°F): 102.0
  Rate of change of core temp (°F/5 mins): 0.5
  Heart rate (beats per min): 180
  Dehydration (% of body wgt): 3.0

III. Cold Digit Module:

Digit length (cm): 8.0;
Axial calculation increment (mm): 2.0
Digit circumference (cm): 2.0
Ambient temperature (C): -5.0
Theshold digit temperature (C): 5.0
Initial time (hrs): 0.0
Final time (hrs): 5.0
Time increment (mins): 15.0
Initial digit base temperature (C): 30.0
Final digit base temperature (C): 20.0
Base temperature time constant (hrs): 1.3
Initial digit tip temperature (C): 20.0
Initial digit tissue metabolic rate (W/m^3): 15,000.0
Final digit tissue metabolic rate (W/m^3): 5,000.0
Metabolic rate time constant (hrs): 1.3
Number of eigenvalues to use in the calculations: 15
Heat transfer coefficient for the circumference surface (W/m^2/hr): 7.12
Heat transfer coefficent for the digit tip (W/m^2/hr): 7.12
Thermal conductivity (W/m/C): 0.418

APPENDIX G: FLDMED FUNCTION TREE DIAGRAM

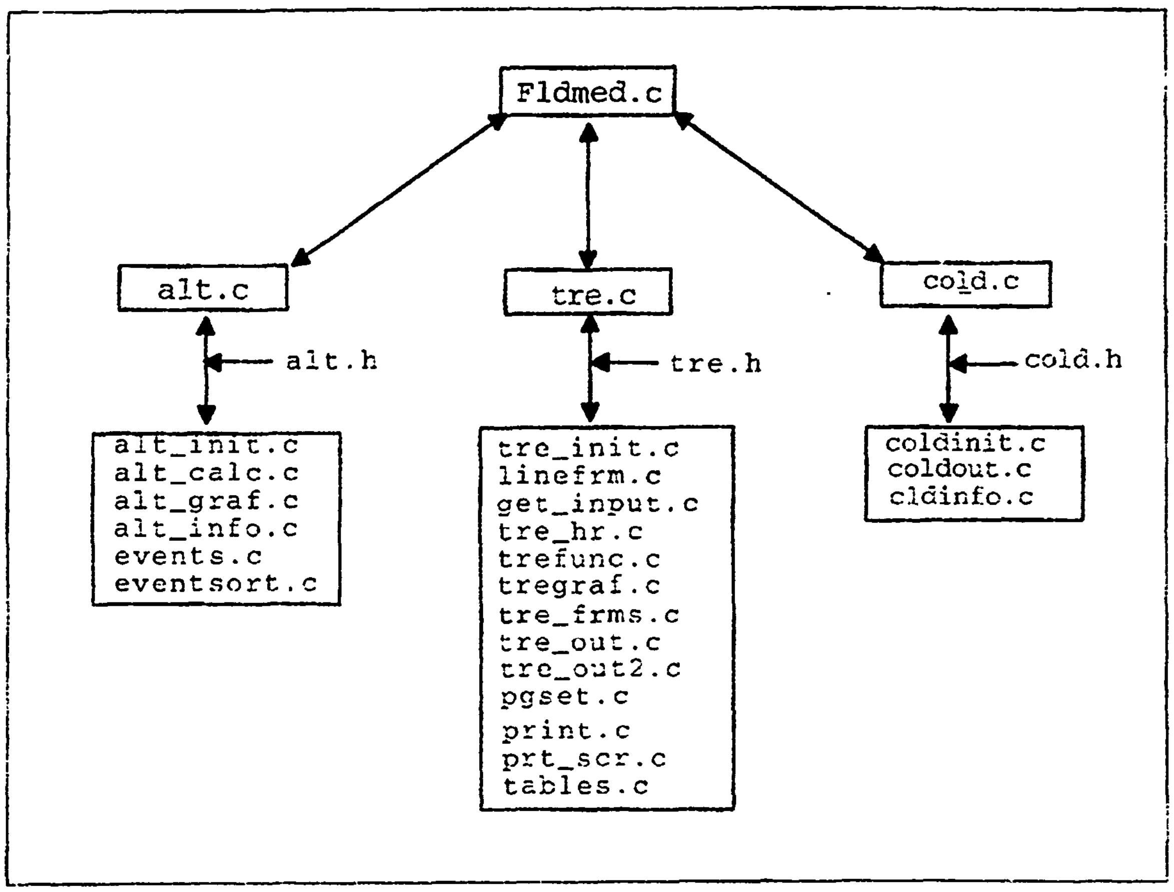


Figure E1: Tree diagram of FLDMED functions.

FUNCTIONS IN THE FLDMED SOURCE FILES

FLDMED.C

main()

-- the FLDMED's main function that loads first and initiates the program. It generates a top bar menu that allows the user to go to the altitude, cold exposure, or heat strain modules.

ALT.C

int alt()

-- the top-level altitude procedure. It manages most of the menus and routes requests involving numerical processing and file handling to subordinate functions.

ALT_CALC.C

int calc_output(struct input_struct *p, struct output_struct *output)

-- calculates the barometric, alveolar, and arterial O2 partial pressures and saturation.

ALT_FRMS.C

void alt_input_by_line(struct input_struct *alt_input, int input_line)

-- accepts the information associated with stages in the ascent-descent profile.

ALT_GRAF.C

int graph_output(struct output_struct *output)

-- produces presentation quality graphs of barometric, O2,and arteriolar O2 pressures and arteriolar O2 saturation.

ALT_INFO.C

int show_alt_info(int topic)

-- displays the selected med info topic text screens.

ALT_TBL.C

void init_alt_input(struct input_struct *input)

-- initializes the altitude profile and physiologic parameters.

void write_current_alt_input(struct input_parameter_struct *input,int column)

-- writes the physiologic inputs to the screen in column format.

BLANK.C

void blank_inputbuff(struct inputbuff_struct *inputbuff)

-- blanks the input data leaving the background screen.

CHKFILES.C

int checkfiles()

-- checks to determine that all necessary files are in the current directory. If not, alerts the user by listing the missing files on the screen and also writes that list to a text file for print outs.

EVENTS.C

int get_alt_profile(struct input_profile_struct *p)

-- generates a pop-up data input window for constructing a custom ascent-descent profile.

int draw_profile(struct input_profile_struct *profile)
-- updates the current ascent-descent profile after a new profile segment is entered.

EVENTSORT.C

int sort_events(struct input_profile_struct *p)
-- sorts the ascent-decent profile segments (events) to avoid altitude or time discontinuities.

int compare_event_dates(struct event_struct *event1, struct event_struct *event2)
-- supports the sort_events(...) fun on.

GET_INPT.C

int inputdataio(struct io_data_struct *io_data)
-- generates the heat strain module's input menus.

int importinputs(struct io_data_struct *io_data)
-- imports a preconstructed set of inputs for the heat strain module.

int saveinputs(struct io_data_struct *io_data)
-- saves the current inputs to a text input data file.

int getinitfile(struct io_data_struct *io_data)
-- imports the initialization input data file.

int savetoinitfile(struct io_data_struct *io_data)
-- on exiting the heat strain module saves the current input data as the new initialization file.

LINEFRM.C

input_by_line(struct io_data_struct *io_data, int input_line)
-- allows inputs by line in the heat stress input data grid.

LIST.C

int list_output(struct output_struct *output)
-- writes the heat strain output summary table into the screen output grid cells.

PGSETS.C

void writeinputs_on_chart(chartenv *env, struct io_data_struct *io_data, enum graph_type_enum graph_type)
-- displays the heat strain input data sets onto unused areas of the output graphs.

PRINT.C

int data_to_printer(struct io_data_struct *io_data)
-- sends summary or detailed output data listings to the local printer.

PRT_SCR.C

int print_screen()
-- sends the selected graphics mode chart to the printer.

TABLES.C

int show_heatstress_info(int topic)
-- manages the display of the heat strain module's med info text files.

TRE.C

int tre()
-- the main function and top level popup menu manager for the heat stress module.

TREFUNC.C

float sweat_rate_func(float e_req, float e_max)
-- returns the sweat rate based on differences between the evaporative heat loss required to maintain normal temperature and the maximum evaporative heat capacity of the ambient air.

float t_ref_func(float met,float w_ext,float e_req,float e_max,float h_rc)
-- returns the predicted steady state core temperature.

float tre_delay_func(enum phases_enum phase,float met_rate, float CP_eff)
-- returns the lag time for effects of changes in metabolic rate when switching from resting to working..

int find_max(float *array, int num_points)
-- a utility function that finds maximum value in output data sets.

```
float casualty_prob_func(float tre_max, float integration_stepsize,
                         float tre_mean_casualty,float tre_std_casualty)
```
-- returns the percent or likelihood of heat stress casualties based on the calculated maximum core temperature.

```
float hr_index(float met,float e_req,float e_max,float temp_db,
               float t_avg_skin,float bsa,float clo)
```
-- returns an intermediate index value for subsequent calculations of heart rate.

float hr_equilib(float i_hr)
-- returns the predicted steady state heart rate

float del_hr_accl_func(float hr_f, float hro, float e_max)
-- steady state heart rate adjustment for heat acclimatization.

float bsa_func(float hgt_cm, float kg)
-- returns body surface area (bsa) based on th DuBois formula.
float met_rate(float wgt,float load,float grade,float terrain_coef, float vel_walk)
-- returns the predicted metabolic rate for marching.

float external_work(float wgt,float load,float vel_walk,float grade)
-- returns the energy dissipated as external work when marching.

float eff_air_vel_func(float nominal_air_vel, float met_rate)
-- returns the effective wind velocity which includes ambient wind plus the relative wind associated with the activities at the given metabolic rate.

float clo_func(float itc, float itvc, float eff_air_vel)
-- returns clothing insulation (clo) value corrected for wind and motion effects.

float evap_impedance_func(float imc, float imvc, float eff_air_vel)
-- returns clothing moisture impedance value corrected for wind and motion effects.

float rad_conv_heat(float temp_db,float temp_sk,float clo,float bsa)
-- returns amount of radiant and convective heat transfer from skin and through the clothing.

float e_max_func(float evap_impedance,float bsa,float vp_sk, float vp_air,float rel_hum)
-- returns maximum possible heat loss via sweat evaporation.

float e_req_func(float met,float w_ext,float h_rc)
-- returns the amount of evaporative heat loss necessary to prevent heat gain.

float solar_func(float eff_air_vel, float itc, float itvc, int index)
-- returns amount of heat gained from solar sources.

float eff_env_cool_power(float emax,float ereq)
-- returns the effective cooling power of the ambient conditions.

float vp_air_sat(float temp_db)
-- returns the predicted saturated vapor pressure of the ambient air.

float max_work_time(float t_delay, float k, float tre_init, float tref, float tre_max_allowed)
-- returns predicted maximum work time based on a specified core temperature limit.

TREGRAF.C

void graph_data(struct io_data_struct *io_data)
-- graphs the outputs for each input data set separately.

void graph_all(struct io_data_struct *io_data)
-- graph the outputs for all four input data sets as four output series per graph.

TRE_FRMS.C

int get_win_forms(WINDOWPTR wn_ptr, enum forms_enum form, struct input_struct *input, struct inputbuff_struct *inputbuff)
-- generates input data entry forms for the heat stress module.

TRE_HR.C

int tre_hr(struct io_data_struct *io_data)

-- the core numerical function for the heat stress unit. Calculates core temperature, heart rate, sweating rate, cummulative sweat and fluid losses, and casualty predictions. Loads the I/O data structure with the simulated time and corresponding results.

TRE_INIT.C

init_input(struct input_struct *input, struct inputbuff_struct *inputbuff)

- initializes the default heat strain module input data structures if an initialization data file cannot be found or if the user requests default values from the input menu.

copy_inputbuff(struct inputbuff_struct *inputbuff, int *current_data_set)

-- copies heat stress input data sets. This function is not currently utilized.

TRE_OUT.C

write_current_input(struct inputbuff_struct *inputbuff,int column)

-- writes the current heat stress inputs onto the screen input data grid.

write_output_table(struct io_data_struct *io_data)

-- writes the heat stress output summary data to the output screen output data grid.

TRE_OUT2.C

int data_to_file(struct io_data_struct *io_data, char *output_path)

-- writes the heat strain output data structures to a text file.

COLD.C

int cold_digit()

- the main cold digit function that generates the cold module interface.

COLDINIT.C

init_input(struct input_struct *input. struct inputbuff_struct *inputbuff)

- initializes the data structures for the cold digit module.

COLDOUT.C

write_current_input(struct inputbuff_struct *inputbuff.int column)

- display the current inputs on the screen.

CLD_INFO.C

int show_cold_info(int topic)

- displays selected topic from military cold medicine handbook.

APPENDIX H

FLDMED C SOURCE CODE

```
/*************** source file:  FLDMED.C ***********************/

#include <windows.h>
#include <vid.h>
#include <process.h>

/** #include "fldmed.h"  **/
/*** #include "key.h"     **/

int tre();
int alt();

char bak_grnd_screen[4000];


 main()
  {

/****
 char *vidbuff_ptr;
 char *output_vidbuff_ptr;
****/
 static WINDOWPTR event_wn. input_wn;
 static WIFORM    event_frm. save_frm;

 /* static */ struct pmenu main_menu=
 {
  0,
  FALSE,
  0.
  0,
  3.
  { 0,4.  "ALTITUDE".                  1.
    0,16. "COLD".                     2,
    0,24. "HEAT ".                    3.
    0,32. "EXIT".                     4,
    99,99,"".99
  }
 };


enum main menu enum {     altitude     =1.
                          cold         =2.
                          heat         =3.
                          quit         =4} main  menu  choice;

v_cls(BLACK<<4IWHITE);
v_border(BLACK);
```

```
  wn_init();
  wn_dmode(FLASH);

/** system("print"); **/ /** load the DOS print driver **/
/** system(" "); **/  /** need a return to the DOS response **/

  if(checkfiles()==0) exit(0); /* At least one of the scr files is missing */

  VidReadWindow(bak_grnd_screen,"army.scr");
  VidPutWindow(bak_grnd_screen,0,0,24,79,0,0);
  getch();

 do
  {
   VidReadWindow(bak_grnd_screen,"army.scr");
   VidPutWindow(bak_grnd_screen,0,0,24,79,0,0);

   main_menu_choice=wn_popup(0,0,18,41,1,(BLACK|YELLOW)<<4|(GREEN|BOLD),
                            BLACK<<4|BLACK,&main_menu,TRUE);

   switch(main_menu_choice)
    {
     case altitude:
          alt();
          break;
     case cold:
          cold_digit();
          break;
     case heat:
          tre();
          break;
     case quit:
          break;
    }
   } while(main_menu_choice != quit);

  v_cls(BLACK<<4|WHITE);
  v_locate(0,0,0);
  printf("*************************************************************\n\n");
  printf("USARIEM Environmental Medicine for the Field Medical Officer \n\n");
  printf("                V: MJR 10/93  \n\n");
  printf("*************************************************************\n\n");

 } /* end main module */
```

ALTITUDE Module: Source Code

```
/*************** source file: ALT.C ***********************/

#include <windows.h>
#include "alt.h"
/*** #include "key.h" **/

extern char bak_cmd_screen[4000];


int alt()
 {

  char *vidbuff_ptr; /* used with VidSave & Restore Block functions */
  char *output_vidbuff_ptr;

  WINDOWPTR event_wn, input_wn;
  WIFORM    event_frm, save_frm;

  int data_row, data_set choice,i, topic;
  static struct input_struct inputdata;
  enum output_type enum output_type;

  static struct output_struct outputdata;

   struct pmenu topbar_menu=
        {
         0,
         FALSE,
         0,
         0,
         5,
         { 0,1, "INPUT",        1,
           0,9, "OUTPUT",        2,
           0,19,"OPTIONS",        3,
           0,30,"MED INFO",       4,
           0,41,"HELP",        5,
           0,47,"EXIT",         6,
           99,99,"",99
         }
     };

enum topbar_menu_enum { input     =1,
                        output      =2,
                        runoptions  =3,
                        info        =4,
                        help        =5,
                        exit_topbar =6 } topbar_menu_choice;
```

```
struct pmenu input_menu=
        {
         0,
         FALSE,
         0,
         0,
         5,
         { 1,4, "Profile",        1,
           2,2, "New Profile",    2,
           3,3, "Variables",      3,
           4,4, "Default",        4,
           5,5, "Save",           5,
           6,1, "Read from File", 6,
           99,99,"",99
         }
    };

    enum input_menu_enum {    view_profile      =1,
                              create_new_profile =2,
                              input_parameters   =3,
                               default_input    =4,
                               save_input       =5,
                               read_file_input  =6 } input_menu_choice;

     struct pmenu med_info_menu=
          {
          0,
          FALSE,
          0,
          1,
          4,
          {  0,3,"ALTITUDE HANDBOOK",      0,
             2,6,"Physiology",             1,
             3,3,"Medical Conditions",     2,
             4,2,"Operational Guidance",   3,
             5,6,"References",             4,
            99,99,"",99
           }
          };

   struct pmenu data_row_menu=
          {
           0,
           FALSE,
           0,
           0,
           8,
          { 0,0, "->",             0,
             1,0, "->",            1,
             2,0, "->",            2,
             3,0, "->",            3,
             4,0, "->",            4,
             5,0, "->",            5,
             6,0, "->",            6,
              7,0, "->",            7,
              8,0, "->",            8,
              99,99,"",99
             }
            };

    struct pmenu data_set_menu=
            {
            0,
            FALSE,
            0,
            0,
            3,
            { 0,3, "SET# 1",            300,
              0,15,"SET# 2",            301,
              0,28,"SET# 3",            302,
              0,40,"SET# 4",            303,
              99,99,"",99
             }
            };

    enum output_choice_enum { view=320,
                              print=321,
                              save=322
                            } output_choice;

    enum output_type_choice_enum { data=0,
                                   graph_all_sets=1,
                                   graph_one_set=2,
                                 } output_type_choice;

 int input_pos[]={29,41,54,66};


 VidReadWindow(bak_grnd_screen,"grid.alt");
 VidPutWindow(bak_grnd_screen,0,0,24,79,0,0);

  if(getinitaltfile(&inputdata) ==99) init_alt_input(&inputdata);
  draw_profile(&inputdata.profile);

do
 {
   topbar_menu_choice=wn_popup(0,0,13.55,1,WHITE<<4IBLUE,
                               BLACK<<4IBLACK,&topbar_menu,TRUE);

   switch(topbar_menu_choice)
    {
     case input:
        input_menu_choice=wn_popup(0,2,12,16,7,WHITE<<4IBLUE,
                                    BLACK<<4IRED,&input_menu,TRUE);
        switch(input_menu_choice)
         {
          case view_profile:
            get_alt_profile(&inputdata.profile);
            break;
          case create_new_profile:
            VidPutWindow(bak_grnd_screen,0,0,24,79,0,0);
            init_input_profile(&inputdata.profile);
            break;
          case input_parameters:
            VidPush();
            VidReadWindow(bak_grnd_screen,"altinput.scr");
            VidPutWindow(bak_grnd_screen,0,0,16,79,0,0);
            VidPutWindow(bak_grnd_screen,17,0,24,2,17,0);
            for(i=0;i<=3;i++)
                 write_current_alt_input(&inputdata.parameters[i],input_pos[i]+2);
            VidReadWindow(bak_grnd_screen,"scraps.scr");
             VidPutWindow(bak_grnd_screen,22,5,22,72,0,5);
             do
               {
                data_row=wn_popup(1000,3,28,2,9,BLACK<<4|RED,
                                   BLACK<<4|RED,&data_row_menu,TRUE);
                alt_input_by_line( &inputdata, data_row);
                for(i=0;i<=3;i++)
                  {
                   inputdata.parameters[i].calculated=no;
                   write_current_alt_input(&inputdata.parameters[i],input_pos[i]+2);
                  } /** end for **/
               } while(data_row !=99);
              VidReadWindow(bak_grnd_screen,"grid.alt");
              VidPop(1);
              break;
          case default_input:
            init_alt_input(&inputdata);
            break;
          case save_input:
            break;
          case read_file_input:
            break;
          }
         break;
case runoptions:
         break;
case output:
    VidPush();
    outputdata.output_type=PmmHg;
    calc_output(&inputdata,&outputdata);
    graph_output(&outputdata);
    VidPop(1);
    break;
case info:
   do
    {
    topic=wn_popup(0,1,28,24,7,WHITE<<4|BLUE,
              BLACK<<4|RED,&med_info_menu,TRUE);
    if(topic==99) break;
```

```
     show_alt_info(topic);
     } while(topic != 99); /* loop until quit info menu */
     break;
   case help:
     break;
  } /* end switch for topbar_menu_choice */
 } while(topbar_menu_choice!=exit_topbar);

 savetoinitaltfile(&inputdata);
 flushall();
 return 0;

} /* end main altitude module */



/******************** source file:  ALT_CALC.C *****************/

#include "alt.h"
#include <math.h>
#include <conio.h>
#include <stdio.h>

#define deg_C(F) ( (5.0/9.0)*(F-32.0) )  /* F to C conversion macro */

#define Tot_P 0
#define PIO2  1
#define PaO2  2
#define O2Sat 3


int calc_output(struct input_struct *p, struct output_struct *output)
 {
  int i, j, k, set, return_val;
  float alt, m, term1, u, PaO2 eft, PAO2_x;

  return_val=0;


 for(set=0;set<NUM_SETS;set++)      /* NUM_SETS defined in alt.h */
  {
  k=0;
  switch(output->output_type)
   {
    case PmmHg:
    for(i=0;(i <= p->profile.num_events) && (k < MAX_OUTPUT_ELEMENTS);i++)
     {

m=(p->profile.event[i].alt2-p->profile.event[i].alt1)/(p->profile.event[i].day2-p->profile.event[i].day1);
          for(j=0;(j < (p->profile.event[i].day2-p->profile.event[i].day1))
                                         &&(k < MAX_OUTPUT_ELEMENTS);j++)
          {
           output->x_val[set][k]=(float)(p->profile event[i].day1+j);
```

```
            alt=p->profile.event[i].alt1+j*m;
            alt=alt*0.3048;  /* converting from feet to meters */
            term1=(44331.514-alt)/11880.516;
            output->y_val[Tot_P][set][k]=0.75*pow(term1,5.25588);
              /* 0.75 converts from mbar to mmHg */
             output->y_val[PIO2][set][k]=(0.01*p->parameters[set].FIO2)*
                                          (output->y_val[Tot_P][set][k]-47.0);
            k++;
           } /* end for(j ... ) */
          } /* end for(i ... ) */
      output->num_vals=k;
      break;
    case Pmbar:
      break;
   }; /** end switch **/

   term1=(1.0-(0.01*p->parameters[set].FIO2*p->parameters[set].resp_ratio))
                                          /p->parameters[set].resp_ratio;
   for(j=0;j<output->num_vals;j++)
    {
     PAO2_x= output->y_val[PIO2][set][j] - p->parameters[set].PACO2*term1;
     output->y_val[PaO2][set][j]= PAO2_x - p->parameters[set].Aa_diffusion_grad;
    }

   term1=0 024*(37.0-deg_C(p->parameters[set].tre));
   term1+=0.4*(p->parameters[set].pH-7.4);
   term1+=0.06*(log10(40.0)-log10(p->parameters[set].PACO2));
   term1=pow(10.0,term1);
   for(j=0;j < output->num_vals;j++)
    {
     PaO2_eff=output->y_val[PaO2][set][j]*term1;
     u=0.00925*PaO2_eff+0.00028*pow(PaO2_eff,2.0)+0.0000306*pow(PaO2_eff,3.0);
     if(output->y_val[PaO2][set][j] >= 10.0)
      output->y_val[O2Sat][set][j]=100.0*(u/(1.0+u));
     else
      output->y_val[O2Sat][set][j]=100.0*(0.003683*PaO2_eff+0.000584*pow(PaO2_eff.2.0));
    } /** end for **/
  } /* end for(set ...) */
  return return_val;

 } /** end of function **/


/******************** source file: ALT_DIFF.C ****************************/

 #include "alt.h'
 #include <stdio.h>
 #include <string.h>

 char* inputdifferences(struct input_struct *input)
  {
   int i.set;
   char msg[70];
   char *inputstr[9]={" Hb "," Hct "," pH "," HCO3- "," Core temp ",
                     " Resp ratio "," % FIO2 "," PACO2 "," Aa diff grad "};
  int diff[9];

  for(i=0;i<9;i++) diff[i]=0;   /* initialize difference indicators
                                      0 = no difference
                                      1 = difference              */

  for(set=0;set<(NUM_SETS-1);set++)
   {
        if(input->parameters[set].hgb != input->parameters[set+1].hgb)  diff[0]=1;
        if(input->parameters[set].hct != input->parameters[set+1].hct)  diff[1]=1;
        if(input->parameters[set].pH  != input->parameters[set+1].pH )  diff[2]=1;
        if(input->parameters[set].HCO3 != input->parameters[set+1].HCO3) diff[3]=1;
        if(input->parameters[set].tre  != input->parameters[set+1].tre)  diff[4]=1;
        if(input->parameters[set].resp_ratio != input->parameters[set+1].resp_ratio) diff[5]=1;
        if(input->parameters[set].FIO2 != input->parameters[set+1].FIO2) diff[6]=1;
        if(input->parameters[set].PACO2 != input->parameters[set+1].PACO2) diff[7]=1;
        if(input->parameters[set].Aa_diffusion_grad !=
                        input->parameters[set+1].Aa diffusion_grad) diff[8]=1;
   }

    strcpy(msg,"N.B.: input differences wrt --> ");
    for(i=0;i<9;i++)
        {
         if(diff[i] == 1) strcpy(msg,strcat(msg,inputstr[i]));
        }

   return &msg[0];
  } /* end function inputdifferences */


/****************** source file: ALT_FRMS.C **********************/

#include <windows.h>
#include "alt.h"

void alt_input_by_line(struct input_struct *alt_input, int input_line)
 {
   WINDOWPTR wn_ptr;
   WIFORM frm_ptr;
    int r0;


    r0=3;
    switch(input_line)
    {
     case 0:
       wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B),
    frm_ptr=wn_frmopn(5);

    wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_red,' ',
              &alt_input->parameters[0].hgb,5,1,10.0,25.0,
              alt_input->parameters[0].hgb_buff,NSTR,
              "Hemoglobin: 0.0 - 25.0");
 wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_red,' ',
          &alt_input->parameters[1].hgb,5,1,10.0,25.0,
          alt_input->parameters[1].hgb_buff,NSTR,
          "Hemoglobin: 0.0 - 25.0");
 wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_red,' ',
          &alt_input->parameters[2].hgb,5,1,10.0,25.0,
          alt_input->parameters[2].hgb_buff,NSTR,
          "Hemoglobin: 0.0 - 25.0");
 wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_red,' ',
          &alt_input->parameters[3].hgb,5,1,10.0,25.0,
          alt_input->parameters[3].hgb buff,NSTR,
          "Hemoglobin: 0.0 - 25.0");

 wn_frmget(frm_ptr);
 wn_frmcls(frm_ptr);
 wn_close(wn_ptr);
 break;
case 1:
 wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG B);
 frm_ptr=wn_frmopn(5);

 wn_gfloat(SET,frm ptr,0,wn_ptr,0,1,NSTR,lgt_red,' ',
          &alt_input->parameters[0].hct,5,1,10.0,90.0,
          alt_input->parameters[0].hct_buff,NSTR,
          "Hematocrit: 0.0 - 90.0");
 wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_red,' ',
          &alt_input->parameters[1].hct,5,1,10.0,90.0,
          alt_input->parameters[1].hct buff,NSTR,
          "Hematocrit: 0.0 - 90.0");
 wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_red,' ',
          &alt input->parameters[2].hct,5,1,10.0,90.0,
          alt_input->parameters[2].hct_buff,NSTR,
          "Hematocrit: 0.0 - 90.0");
 wn_gfloat(SET,frm_ptr,3,wn ptr,0,38,NSTR,lgt_red,' ',
          &alt input->parameters[3].hct,5,1,10.0,90.0,
          alt_input->parameters[3].hct_buff,NSTR,
          "Hematocrit: 0.0 - 90.0");

 wn_frmget(frm ptr);
 wn_frmcls(frm_ptr);
 wn_close(wn_ptr);
 break;
case 2:
 wn_ptr=wn open(1000,r0+input_line,30,45,1,FG_R,FG_B);
 frm_ptr=wn frmopn(5);

 wn gfloat(SET,frm ptr,0,wn_ptr,0,1,NSTR,lgt_red,' ',
          &alt input->parameters[0].pH,5,1,4.0,8.0,
          alt input->parameters[0].pH_buff,NSTR,
          "blood pH: 4.0 - 8.0");
 wn gfloat(SET,frm_ptr,1,wn ptr,0,13,NSTR,lgt_red,' ',
          &alt input->parameters[1].pH,5,1,4.0,8.0,
          alt_input->parameters[1].pH_buff,NSTR,
          "blood pH: 4.0 - 8.0");
```

```
    wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_red,' ',
             &alt_input->parameters[2].pH,5,1,4.0,8.0,
             alt_input->parameters[2].pH_buff,NSTR,
             "blood pH: 4.0 - 8.0");
    wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_red,' ',
             &alt_input->parameters[3].pH,5,1,4.0,8.0,
             alt_input->parameters[3].pH_buff,NSTR,
             "blood pH: 4.0 - 8.0");

    wn_frmget(frm_ptr);
    wn_frmcls(frm_ptr);
    wn_close(wn_ptr);
    break;

  case 3:
    wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B);
    frm_ptr=wn_frmopn(5);

    wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_red,' ',
             &alt_input->parameters[0].HCO3,5,1,0.0,50.0,
             alt_input->parameters[0].HCO3_buff,NSTR,
             "Blood bicarb: 0.0 - 50.0");
    wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_red,' ',
             &alt_input->parameters[1].HCO3,5,1,0.0,50.0,
             alt_input->parameters[1].HCO3_buff,NSTR,
             "Blood bicarb: 0.0 - 50.0");
    wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_red,' ',
             &alt input->parameters[2].HCO3,5,1,0.0,50.0,
             alt_input->parameters[2].HCO3_buff,NSTR,
             "Blood bicarb: 0.0 - 50.0");
    wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_red,' ',
             &alt_input->parameters[3].HCO3,5,1,0.0,50.0,
             alt_input->parameters[3].HCO3_buff,NSTR,
             "Blood bicarb: 0.0 - 50.0");

    wn_frmget(frm_ptr);
    wn_frmcls(frm_ptr);
    wn_close(wn_ptr);
    break;
  case 4:
    wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B);
    frm_ptr=wn_frmopn(5);

    wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_red,' ',
             &alt_input->parameters[0].tre,5,1,90.0,110.0,
             alt_input->parameters[0].tre_buff,NSTR,
             "Blood temp(F): 90.0 - 110.0");
    wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_red,' ',
             &alt input->parameters[1].tre,5,1,90.0,110.0,
             alt_input->parameters[1].tre_buff,NSTR,
             "Blood temp (F): 90.0 - 110.0");
    wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_red,' ',
             &alt input->parameters[2].tre,5,1,90.0,110.0,
             alt input->parameters[2].tre_buff,NSTR,
            "Blood temp(F): 90.0 - 110.0");
  wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_red,' ',
            &alt_input->parameters[3].tre,5,1,90.0,110.0,
            alt_input->parameters[3].tre_buff,NSTR,
            "Blood temp(F): 90.0 - 110.0");

  wn_frmget(frm_ptr);
  wn_frmcls(frm_ptr);
  wn_close(wn_ptr);
  break;
 case 5:
  wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B);
  frm_ptr=wn_frmopn(5);

  wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_blue,' ',
            &alt_input->parameters[0].resp_ratio,5,1,0.0,2.0,
            alt_input->parameters[0].resp_ratio_buff,NSTR,
            "CO2/O2 ratio: 0.0 - 2.0");
  wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_blue,' ',
            &alt_input->parameters[1].resp_ratio,5,1,0.0,2.0,
            alt_input->parameters[1].resp_ratio_buff,NSTR,
            "CO2/O2 ratio: 0.0 - 2.0");
  wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_blue,' ',
            &alt_input->parameters[2].resp_ratio,5,1,0.0,2.0,
            alt_input->parameters[2].resp_ratio_buff,NSTR,
            "CO2/O2 ratio: 0.0 - 2.0");
  wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_blue,' ',
            &alt_input->parameters[3].resp_ratio,5,1,0.0,2.0,
            alt_input->parameters[3].resp_ratio_buff,NSTR,
            "CO2/O2 ratio: 0.0 - 2.0");

  wn_frmget(frm_ptr);
  wn_frmcls(frm_ptr);
  wn_close(wn_ptr);
  break;

 case 6:
  wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B);
  frm_ptr=wn_frmopn(5);

  wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_blue,' ',
            &alt_input->parameters[0].FIO2,5,1,0.0,100.0,
            alt_input->parameters[0].FIO2_buff,NSTR,
            "Inspilgt_red O2: 0.0 - 100%");
  wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_blue,' ',
            &alt_input->parameters[1].FIO2,5,1,0.0,100.0,
            alt_input->parameters[1].FIO2_buff,NSTR,
            "Inspilgt_red O2: 0.0 - 100%");
  wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_blue,' ',
            &alt_input->parameters[2].FIO2,5,1,0.0,100.0,
            alt_input->parameters[2].FIO2_buff,NSTR,
            "Inspilgt_red O2: 0.0 - 100%");
  wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_blue,' ',
            &alt_input->parameters[3].FIO2,5,1,0.0,100.0,
              alt_input->parameters[3].FIO2_buff,NSTR,
              "Inspilgt_red O2: 0.0 - 100%");

   wn_frmget(frm_ptr);
   wn_frmcls(frm_ptr);
   wn_close(wn_ptr);

   break;
 case 7:
   wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B);
   frm_ptr=wn_frmopn(5);

   wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_blue,' ',
              &alt_input->parameters[0].PACO2,5,1,0.0,100.0,
              alt_input->parameters[0].PACO2_buff,NSTR,
              "Inspilgt_red CO2: 0.0 - 100 mmHg");
   wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_blue,' ',
              &alt_input->parameters[1].PACO2,5,1,0.0,100.0,
              alt_input->parameters[1].PACO2_buff,NSTR,
              "Inspilgt_red O2: 0.0 - 100% mmHg");
   wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_blue,' ',
              &alt_input->parameters[2].PACO2,5,1,0.0,100.0,
              alt_input->parameters[2].PACO2_buff,NSTR,
              "Inspilgt_red O2: 0.0 - 100% mmHg");
   wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_blue,' ',
              &alt_input->parameters[3].PACO2,5,1,0.0,100.0,
              alt_input->parameters[3].PACO2_buff,NSTR,
              "Inspilgt_red O2: 0.0 - 100% mmHg");

   wn_frmget(frm_ptr);
   wn_frmcls(frm_ptr);
   wn_close(wn_ptr);
  break;
 case 8:
   wn_ptr=wn_open(1000,r0+input_line,30,45,1,FG_R,FG_B);
   frm_ptr=wn_frmopn(5);

   wn_gfloat(SET,frm_ptr,0,wn_ptr,0,1,NSTR,lgt_blue,' ',
              &alt_input->parameters[0].Aa_diffusion_grad,5,1,0.0,100.0,
              alt_input->parameters[0].Aa_diffusion_grad_buff,NSTR,
              "Nl Aa diff grad: 0.0 - 10.0 mmHg ");
   wn_gfloat(SET,frm_ptr,1,wn_ptr,0,13,NSTR,lgt_blue,' ',
              &alt_input->parameters[1].Aa_diffusion_grad,5,1,0.0,100.0,
              alt_input->parameters[1].Aa_diffusion_grad_buff,NSTR,
              "Nl Aa diff grad: 0.0 - 10.0 mmHg ");
   wn_gfloat(SET,frm_ptr,2,wn_ptr,0,26,NSTR,lgt_blue,' ',
              &alt_input->parameters[2].Aa_diffusion_grad,5,1,0.0,100.0,
              alt_input->parameters[2].Aa_diffusion_grad_buff,NSTR,
              "Nl Aa diff grad: 0.0 - 10.0 mmHg ").
   wn_gfloat(SET,frm_ptr,3,wn_ptr,0,38,NSTR,lgt_blue,' ',
              &alt_input->parameters[3].Aa_diffusion_grad,5,1,0.0,100.0,
              alt_input->parameters[3].Aa_diffusion_grad_buff,NSTR,
              "Nl Aa diff grad: 0.0 - 10.0 mmHg ");
```

```
      wn_frmget(frm_ptr);
      wn_frmcls(frm_ptr);
      wn_close(wn_ptr);
     break;

    } /** end switch(...) **/
  } /** end input_by_line **/


/********************** source file: ALT_GRAF.C *******************/

#include "alt.h"
 #include <stdio.h>
 #include <string.h>
 #include <conio.h>
 #include <graph.h>
 #include <pgchart.h>

 #define Tot_P 0
 #define PIO2  1
 #define PaO2  2
 #define O2Sat 3


 #define TRUE 1
 #define FALSE 0

int graph_output(struct output_struct *output)
  {
   int return_val,i,j,n,set,graph_num;
   float max,min,tic_interval;
   chartenv  env;
   float x[NUM_SETS][MAX_OUTPUT_ELEMENTS],
                     y[NUM_SETS][MAX_OUTPUT_ELEMENTS];

   char *labels[]={"Set#1","Set#2","Set#3", Set#4"};
   int mode=_VRES16COLOR;
   return_val=0;




   n=output->num_vals;
   for(set=0;set<NUM_SETS;set++)
          for(i=0;i<n && i<MAX_OUTPUT_ELEMENTS;i++)
                                x[set][i]=output->x_val[set][i];

/* set highest video mode available. */
   while(! _setvideomode(mode)) mode--;

   _pg_initchart();
   _pg_defaultchart(&env,_PG_SCATTERCHART,_PG_POINTANDLINE);
```

```
for(graph_num=0;graph_num<4;graph_num++)
 {
 switch(graph_num)
 {
 case Tot_P:
          for(set=0;set<NUM_SETS;set++)
            for(i=0;i<n && i<MAX_OUTPUT_ELEMENTS;i++)
                                  y[set][i]=output->y_val[Tot_P][set][i];
          strcpy(env.maintitle.title,"Barometric Pressure");
          strcpy(env.xaxis.axistitle.title,"Day #");
          strcpy(env.yaxis.axistitle.title,"mm Hg");

          env.chartwindow.x1=8;  /* pixels */
          env.chartwindow.y1=8;
          env.chartwindow.x2=311;
          env.chartwindow.y2=231;

          env.xaxis.autoscale=FALSE;
          env.yaxis.autoscale=FALSE;
          env.datawindow.background=8;
          env.chartwindow.background=0;
          env.chartwindow.bordercolor=0;
          env.xaxis.axistitle.titlecolor=5;
          env.yaxis.axistitle.titlecolor=5;
          env.xaxis.axiscolor=2;
          env.yaxis.axiscolor=2;
          env.xaxis.scalemin=1.0;
          env.xaxis.scalemax=x[0][n-1];
          env.xaxis.scalefactor=1.0;
          env.xaxis.ticinterval=5.0;
          env.xaxis.ticformat= _PG_DECFORMAT;
          env.xaxis.ticdecimals=0;

          env.yaxis.scalemin=300.0;
          env.yaxis.scalemax=800.0;
          env.yaxis.scalefactor=1.0;
          env.yaxis.ticinterval=50.0;
          env.yaxis.ticformat=_PG_DECFORMAT;
          env.yaxis.ticdecimals=0;

          env.yaxis.grid=TRUE;
          env.xaxis.grid=TRUE;

          env.maintitle.titlecolor=5;     /* 5=red */
          env.maintitle.justify=_PG_CENTER;

          env.subtitle.titlecolor=5;
          env.subtitle.justify=_PG_CENTER;
          break;
 case PIO2:
          for(set=0;set<NUM_SETS;set++)
               for(i=0;i<n && i<MAX_OUTPUT_ELEMENTS;i++)
                                  y[set][i]=output->y_val[PIO2][set][i];
          strcpy(env.maintitle.title,"Ambient O2 Pressure");
```

```
        env.chartwindow.x1=317;  /* pixels */
        env.chartwindow.y1=8;
        env.chartwindow.x2=631;
        env.chartwindow.y2=231;

        env.yaxis.autoscale=TRUE;
/**   env.yaxis.scalemin=00.0;
        env.yaxis.scalemax=150.0; **/
        env.yaxis.ticinterval=10.0;
        break;
case PaO2:
        max=0.0;
        min=100.0;
        for(set=0;set<NUM_SETS;set++)
          {
          for(i=0;i<n && i<MAX_OUTPUT_ELEMENTS;i++)
                {
                  y[set][i]=output->y_val[PaO2][set][i];
                  if(y[set][i]<min) min=y[set][i];
                  if(y[set][i]>max) max=y[set][i];
                }
          }
        strcpy(env.maintitle.title,"Arterial O2 Pressure");
        strcpy(env.yaxis.axistitle.title,"mmHg");

        env.yaxis.autoscale=FALSE;
        env.yaxis.scalemin=min-5.0;
        env.yaxis.scalemax=max+5.0;
        env.yaxis.ticinterval=5.0;

        env.chartwindow.x1=8;
        env.chartwindow.y1=247;
        env.chartwindow.x2=311;
        env.chartwindow.y2=471;

        break;
case O2Sat:
        max=0.0;
        min=100.0;

      for(set=0;set<NUM_SETS;set++)
          {
          for(i=0;i<n && i<MAX_OUTPUT_ELEMENTS;i++)
                {
                  y[set][i]=output->y_val[O2Sat][set][i];
                  if(y[set][i]<min) min=y[set][i];
                  if(y[set][i]>max) max=y[set][i];
                }
          }
        strcpy(env.maintitle.title,"Arterial O2 Saturation");
        strcpy(env.yaxis.axistitle.title,"%");
        env.chartwindow.x1=317;  /* pixels */
        env.chartwindow.y1=247;
        env.chartwindow.x2=631;
```

```
            env.chartwindow.y2=471;

            env.yaxis.autoscale=FALSE;
            env.yaxis.scalemin=min-5.0;
            env.yaxis.scalemax=max+5.0;
            env.yaxis.ticinterval=5.0;
            break;
    } /** end switch **/

   if( _pg_analyzescatterms(&env,&x[0][0].&y[0][0].NUM_SETS,n,MAX_OUTPUT_ELEMENTS,&labels))
     {
          _outtext("Error analyzing chart data ");
          fflush(stdin);
          getch();
          _setvideomode(_DEFAULTMODE);
          return 98;
     }

   if( _pg_chartscatterms( &env,&x[0][0],&y[0][0].NUM_SETS,n,MAX  OUTPUT_ELEMENTS,&labels))
     {
         _outtext("Error: can't draw chart!");
          fflush(stdin);
          getch();
         return  val=99;
     }
   } /** end for(graph_num=0;graph  num<4;graph  num++) **/

 /**   _pg_hlabelchart( &env. 5,5.&output->diffmsg.15); **/ /* 15=yellow */
   fflush(stdin);
   getch();
   fflush(stdin);

   _setvideomode(_DEFAULTMODE);

   return return_val;
 } /* end graph_output  */



/****************** source file: ALT_INFO.C *******************/
#include <key.h>
#include "alt.h"

extern char bak_grnd_screen[4000];


int show_alt_info(int topic)
 {
  int i, key, scan. index_num, start_index_num. stop_index  num;
/***  FILE *screenfiles;  ****/

                              /*    Index #    Topic                 */
  char *files[20]={"p1 alt". /*   0   Altitude physiology      */
```

```
                "p2.alt",  /*  1          "          */
                "p3.alt",  /*  2          "          */
                "p4.alt",  /*  3          "          */
                "p5.alt",  /*  4          "          */
                "m1.alt",  /*  5    Medical Conditions */
                "m2.alt",  /*  6          "          */
                "m3.alt" , /*  7          "          */
                "m4.alt" , /*  8          "          */
                "m5.alt",  /*  9          "          */
                "m6.alt",/*   10          "          */
                "m7.alt",/*   11          "          */
                "m8.alt", /*  12          "          */
                "medops1.alt", /* 13 Operational Guidance */
                "medops2.alt", /* 14      "
                "medops3.alt", /* 15      "          */
                "ref1.alt", /* 16    Reference Lists  */
                "ref2.alt", /* 17         "          */
                "ref3.alt", /* 18         "          */
                "ref4.alt" /* 19          "          */
                };

switch(topic)
 {
 case 1: /* Alt Physiology */
  start_index_num=0:
  stop_index_num=4;
  break;
 case 2: /* Medical Conditions at Alt */
  start_index_num=5;
  stop_index_num=12;
  break;
 case 3: /* Operational Quidance */
  start_index_num=13;
  stop_index_num=15;
  break;
 case 4: /* References */
  start_index_num=16;
  stop_index_num=19;
  break;
 } /* end switch */


index_num=start_index_num;
if( VidReadWindow(bak_grnd_screen,files[index_num]) != NOERROR)
     {
          printf("Unable to read %s screen file \n",files[index_num]);
          getch();
          return 0;
     }
VidPush();
VidPutWindow(bak_grnd_screen,0,0,24,79,0,0);
v_hidec();
```

```
do
   {
          KeyFlush();
          key=KeyGetC();
          scan=key>>8;

          switch(scan)
           {
            case 28:      /* enter     */
            case 80:      /* down arrow */
            case 81:      /* page down  */
             index_num++;
             break;
            case 72:      /* up arrow   */
            case 73:      /* page up    */
             index_num--;
             break;
            case 71:      /* home key   */
             index_num=start_index_num;
             break;
            case 79:
             index_num=stop_index num;
             break;
            default:
             VidPop(1);
             v_sctype(1,10,12); /* restore cursor */
             return 0;
             break;
           } /* end switch */
           if(index_num<start_index_num) index_num=stop_index_num;
           if(index_num>stop_index_num) index_num=start_index_num;

           if( VidReadWindow(bak_grnd_screen,files[index_num]) != NOERROR)
             {
                printf("Unable to read %s screen file \n",files[index_num]);
                getch();
                VidPop(1);
                return 0;
             }
           VidPutWindow(bak_grnd_screen,0,0,24,79,0,0);
   } while((key & 0x00ff) != 27);
   VidPop(1);
   v_sctype(1,10,12); /* restore cursor */
   return 0;

  } /* end show_alt_info(int topic) */
```

```
 **************** source file:  ALT_INIO.C *******************/

#include "alt.h"
#include <stdio.h>

int getinitaltfile(struct input_struct *alt_input)
   {
        int set_num;
        FILE *input_file;
        size_t inputsize;
        inputsize=sizeof(*alt_input);

        input_file=fopen("altlast.dat","rb");
        if(input_file==NULL) return 99;

         fread(alt_input,inputsize,1,input_file);

         fclose(input_file);
         return 1;
      } /*end this function */

 int savetoinitaltfile(struct input_struct *alt_input)
    {
        int set_num;
        FILE *out_file;
        size_t inputsize;
        inputsize=sizeof(*alt_input);

        out_file=fopen("altlast.dat","wb");
        if(out_file==NULL) return 99;

        fwrite(alt_input,inputsize,1,out_file);

        fclose(out_file);
        return 1;
   } /* end this function */


/**************** source file:  ALT_TBL.C *******************/

#include <string.h>
#include "alt.h"



void init_alt_input(struct input_struct *input)
 {
   int n;

   input->profile.num_events=2;
   input->profile.current_event=0;
   input->profile.event[0].type_index=0;
   input->profile.event[0].day1=1;
input->profile.event[0].day2=5;
input->profile.event[0].alt1=0;
input->profile.event[0].alt2=10000;
strcpy(input->profile.event[0].typebuff,"Ascend ");
strcpy(input->profile.event[0].day1buff," 1");
strcpy(input->profile.event[0].day2buff," 5");
strcpy(input->profile.event[0].alt1buff,"    0");
strcpy(input->profile.event[0].alt2buff,"10000");

input->profile.event[1].type_index=2;
input->profile.event[1].day1=5;
input->profile.event[1].day2=15;
input->profile.event[1].alt1=10000;
input->profile.event[1].alt2=10000;
strcpy(input->profile.event[1].typebuff,"Remain ");
strcpy(input->profile.event[1].day1buff," 5");
strcpy(input->profile.event[1].day2buff," 15");
strcpy(input->profile.event[1].alt1buff,"10000");
strcpy(input->profile.event[1].alt2buff,"10000");

input->profile.event[2].type_index=1;
input->profile.event[2].day1=15;
input->profile.event[2].day2=20;
input->profile.event[2].alt1=10000;
input->profile.event[2].alt2=0;
strcpy(input->profile.event[2].typebuff,"Descend");
strcpy(input->profile.event[2].day1buff," 15");
strcpy(input->profile.event[2].day2buff," 20");
strcpy(input->profile.event[2].alt1buff,"10000");
strcpy(input->profile.event[2].alt2buff,"    0");


for(n=0;n<=3;n++)
 {
  input->parameters[n].hgb=14.5;
  input->parameters[n].hct=45.0;
  input->parameters[n].pH=7.4;
  input->parameters[n].HCO3=24;
  input->parameters[n].tre=98.8;
  input->parameters[n].resp_ratio=0.8;
  input->parameters[n].FIO2=21.0;
  input->parameters[n].PACO2=40.0;
  strcpy(          input->parameters[n].hgb_buff," 14.5");
  strcpy(          input->parameters[n].hct_buff," 45.0");
  strcpy(           input->parameters[n].pH_buff,"  7.4");
  strcpy(         input->parameters[n].HCO3_buff," 24.0");
  strcpy(          input->parameters[n].tre_buff," 98.8");
  strcpy(    input->parameters[n].resp_ratio_buff,"  0.8");
  strcpy(         input->parameters[n].FIO2_buff," 21.0");
  strcpy(        input->parameters[n].PACO2_buff," 40.0");
  strcpy(input->parameters[n].Aa_diffusion_grad_buff,"  5.0");
  input->parameters[n].calculated=0;
 } /** end for **/
} /** end init_alt_input function **/
```

```
void  write_current_alt_input(struct input_parameter_struct *input,int column)
  {
        /*  n.b. colors are defined in alt.h */
        int r=3;  /*init row */

        VidPutS(input->hgb_buff,red,r,column);
        VidPutS(input->hct_buff,red,++r,column);
        VidPutS(input->pH_buff,red,++r,column);
        VidPutS(input->HCO3_buff,red,++r,column);

        VidPutS(input->tre_buff,red,++r,column);
        VidPutS(input->resp_ratio_buff,lgt_blue,++r,column);
        VidPutS(input->FIO2_buff,lgt_blue,++r,column);
        VidPutS(input->FACO2_buff,lgt_blue,++r,column);
        VidPutS(input->Aa_diffusion_grad_buff,lgt_blue,++r,column);
  } /*  end write_current_alt_input function  */



/************************* source file: ALT  INIT.C *********************/

#include <windows.h>
#include "alt.h"

int init_input_profile(struct input_profile_struct *p)
 {
  int i;

  p->num_events=0;
  p->current_event=0;

  for(i=0;i<MAX_EVENTS;i++)
   {
    p->event[i].type_index=0; /* 0=remain */
    p->event[i].day1=1;
    p->event[i].day2=3;
    p->event[i].alt1=0;
    p->event[i].alt2=10000;
    strcpy(p->event[i].typebuff,"Remain "):
    strcpy(p->event[i].day1buff,"  1");
    strcpy(p->event[i].day2buff,"  3");
    strcpy(p->event[i].alt1buff,'00000").
    strcpy(p->event[i].alt2buff,"10000");
   } /* end for      */
   return 1;
 }  /* end function  */
```

```
/************************ source file: EVENTS      .C *********************/

#include <windows.h>
#include "alt.h"

#define TABLE_HEADER_ROW  5
#define TABLE_HEADER_COL 20


int get_alt_profile(struct input_profile_struct *p )
{
 WINDOWPTR event_wn;
 WIFORM    event_frm;

 int i, current_event_index;
 int alt1,alt2, day1,day2, event_type;
 int r0,r1,r2,r,c;

 char typebuff[9], day1buff[5], alt1buff[7], day2buff[5], alt2buff[7];

 float m;

 int r_table=TABLE_HEADER_ROW;
 int c_table=TABLE_HEADER_COL;

  struct pmenu event_menu=
  {
   0,
   FALSE,
   0,
   0,
   3,
   { 0,0, "Ascend ',        0,
     1,0, "Descend",        1,
     2,0, "Remain ",        2,
     3,1, "Stop"  ,        3,
     99,99,"",99
   }
 };

 enum event_enum { ascend  = 0,
                   descend = 1,
                   remain  = 2,
                   stop    = 3} event_menu  choice;

 char eventstr[4][8]={"Ascend ","Descend","Remain ", Stop  "};


    r_table=TABLE_HEADER_ROW;
    c_table=TABLE_HEADER_COL;


 if(r_table==TABLE_HEADER_ROW)
    VidPutS("Event  Alt1 (ft)  Alt2 (ft) from Day# to Day#',
                FG_R,r_table,c_table);

current_event_index=0;
do
 {
  day1=p->event[current_event_index].day1;
  day2=p->event[current_event_index].day2;
  alt1=p->event[current_event_index].alt1;
  alt2=p->event[current_event_index].alt2;
  event_type=p->event[current_event_index].type_index;
  strcpy(day1buff,p->event[current_event_index].day1buff);
  strcpy(day2buff,p->event[current_event_index].day2buff);
  strcpy(alt1buff,p->event[current_event_index].alt1buff);
  strcpy(alt2buff,p->event[current_event_index].alt2buff);
  strcpy(typebuff,p->event[current_event_index].typebuff);


  event_wn=wn_open(500,4,15,40,5,WHITE<<4|RED,BLACK<<4|BLUE);

  event_menu_choice=wn_popup(0,5,28,7,4,WHITE<<4|BLUE,WHITE<<4|WHITE,
                              &event_menu,TRUE);
          if(event_menu_choice==99)
           {
            wn_close(event_wn);
            return 99;
           }
  strcpy(typebuff,eventstr[event_menu_choice]);
  strcpy(p->event[current_event_index].typebuff,typebuff);

  wn_dtext(XEQ,NFRM,NFLD,event_wn,1,10,&eventstr[event_menu_choice]);

  switch(event_menu_choice)
   {
    case ascend:
    case descend:
        event_frm=wn_frmopn(5);
        wn_gint(SET,event_frm,0,event_wn,2,1,"From (ft): ",BLACK<<4|RED,' ',
                &alt1,5,0,23000,alt1buff,NSTR,"Limits: 0 - 23,000 ft");
        wn_gint(SET,event_frm,1,event_wn,2,18,"to (ft): ",BLACK<<4|RED,' ',
                &alt2,5,0,23000,alt2buff,NSTR,"Limits: 0 - 23,000 ft");
        wn_gint(SET,event_frm,2,event_wn,3,1,"Between day# : ",BLACK<<4|RED,' ',
                &day1,4,1,35,day1buff,NSTR,"Limits: 1 - 35");
        wn_gint(SET,event_frm,3,event_wn,3,21,"and day# : ",BLACK<<4|RED,' ',
                &day2,4,day1,35,day2buff,NSTR,"Limits: 1 - 35");
        if(wn_frmget(event_frm)==ESC_CODE)
          {
           wn_frmcls(event_frm);
           wn_close(event_wn);
           return 99;
          }
        break;
    case remain:
        event_frm=wn_frmopn(4);
        wn_gint(SET,event_frm,0,event_wn,2,1 "At (ft): ",BLACK<<4|RED,' ',
                &alt1,5,0,23000,alt1buff,NSTR,"Limits: 0 - 23,000 ft");
        wn_gint(SET,event_frm,1,event_wn,3,1,"Between day# : ",BLACK<<4|RED,' ',
                &day1,4,1,35,day1buff,NSTR,"Limits: 1 - 35");
        wn_gint(SET,event_frm,2,event_wn,3,21,"and day# : ",BLACK<<4|RED,' ',
                &day2,4,day1,35,day2buff,NSTR,"Limits: 1 - 35");
        if(wn_frmget(event_frm)==ESC_CODE)
           {
            wn_frmcls(event_frm);
            wn_close(event_wn);
            break;
           }
        alt2=alt1;
        strcpy(alt2buff,alt1buff);
        break;
    case stop:
        wn_close(event_wn);
        break;
  };  /* end switch */

 if(event_menu_choice==stop) break;
 wn_frmcls(event_frm);
 wn_close(event_wn);

 r0=23;
 r1=(int)(23-(alt1/1000));
 r2=(int)(23-(alt2/1000));
 c=(int)(8+2*(day1-1));

 if( (day2-day1) == 0 )
  {
        VidPutS(typebuff,FG_I|FG_R|FG_G,++r_table,c_table);
        VidPutS(alt1buff,FG_I|FG_R|FG_G,r_table,c_table+8);
        VidPutS(alt2buff,FG_I|FG_R|FG_G,r_table,c_table+19);
        VidPutS(day1buff,FG_I|FG_R|FG_G,r_table,c_table+34);
        VidPutS(day2buff,FG_I|FG_R|FG_G,r_table,c_table+41);

        VidPutAttr(WHITE<<4|BLUE,r2,c,r0,c);
  }
 else
  {
        VidPutS(typebuff,FG_I|FG_R|FG_G,++r_table,c_table);
        VidPutS(alt1buff,FG_I|FG_R|FG_G,r_table,c_table+8);
        VidPutS(alt2buff,FG_I|FG_R|FG_G,r_table,c_table+19);
        VidPutS(day1buff,FG_I|FG_R|FG_G,r_table,c_table+34);
        VidPutS(day2buff,FG_I|FG_R|FG_G,r_table,c_table+41);



        if(alt1==alt2) m=0.0;
        else m=(float)(r2-r1)/(2*(day2-day1));

        for(i=0;i<=(2*(day2-day1));i++,c++)
         {
          if(m==0.0) r=r1;                  /* remain at current alt */
```

```
            else if(m < 0.0) r=(int)(r1+m*i);     /* ascend  */
            else if(m > 0.0) r=(int)(r1+m*i);     /* descend */
            VidPutAttr(WHITE<<4IBLUE,r,c,r0,c);
           }
         } /* end else */

     p->num_events++;
     p->current_event=current_event_index;
     p->event[current_event_index].day1=day1;
     p->event[current_event_index].day2=day2;
     p->event[current_event_index].alt1=alt1;
     p->event[current_event_index].alt2=alt2;
     p->event[current_event_index].type_index=event_type;
     strcpy(p->event[current_event_index].day1buff,day1buff);
     strcpy(p->event[current_event_index].day2buff,day2buff);
     strcpy(p->event[current_event_index].alt1buff,alt1buff);
     strcpy(p->event[current_event_index].alt2buff,alt2buff);
     strcpy(p->event[current_event_index].typebuff,typebuff);

     p->event[current_event_index+1].day1=day2;
     p->event[current_event_index+1].day2=day2;
     p->event[current_event_index+1].alt1=alt2;
     p->event[current_event_index+1].alt2=alt2;
     p->event[current_event_index+1].type_index=event_type;
     strcpy(p->event[current_event_index+1].day1buff,day2buff);
     strcpy(p->event[current_event_index+1].day2buff,day2buff);
     strcpy(p->event[current_event_index+1].alt1buff,alt2buff);
     strcpy(p->event[current_event_index+1].alt2buff,alt2buff);
     strcpy(p->event[current_event_index+1].typebuff,typebuff);

     current_event_index++;


    } while(event_menu_choice != stop);

    p->num_events--;
    return 0;

  } /* end function  get_input_profile()  */


int draw_profile(struct input_profile_struct *profile)
  {

   int i,j;
   int r0,r1,r2,r,c;
   float m;

   int r_table=TABLE_HEADER_ROW;
   int c_table=TABLE_HEADER_COL;

   r0=23;

   VidPutS("Event  Alt1 (ft)  Alt2 (ft) from Day# to Day#",FG_R,r_table,c_table);
  for(i=0;i <= profile->num_events;i++)
   {
    r1=(int)(23-(profile->event[i].alt1/1000));
    r2=(int)(23-(profile->event[i].alt2/1000));
    c=(int)(8+2*(profile->event[i].day1-1));

    if( (profile->event[i].day2-profile->event[i].day1) == 0 )
     {
        VidPutS(profile->event[i].typebuff,FG_IIFG_RIFG_G,++r_table,c_table);
        VidPutS(profile->event[i].alt1buff,FG_IIFG_RIFG_G,r_table,c_table+8);
        VidPutS(profile->event[i].alt2buff,FG_IIFG_RIFG_G,r_table,c_table+19);
        VidPutS(profile->event[i].day1buff,FG_IIFG_RIFG_G,r_table,c_table+34);
        VidPutS(profile->event[i].day2buff,FG_IIFG_RIFG_G,r_table,c_table+41);

        VidPutAttr(WHITE<<4IBLUE,r2,c,r0,c);
     }
    else
     {
        VidPutS(profile->event[i].typebuff,FG_IIFG_RIFG_G,++r_table,c_table);
        VidPutS(profile->event[i].alt1buff,FG_IIFG_RIFG_G,r_table,c_table+8);
        VidPutS(profile->event[i].alt2buff,FG_IIFG_RIFG_G,r_table,c_table+19);
        VidPutS(profile->event[i].day1buff,FG_IIFG_RIFG_G,r_table,c_table+34);
        VidPutS(profile->event[i].day2buff,FG_IIFG_RIFG_G,r_table,c_table+41);

        if(profile->event[i].alt1==profile->event[i].alt2) m=0.0;
        else m=(float)(r2-r1)/(2*(profile->event[i].day2-profile->event[i].day1));


        for(j=0;j<=(2*(profile->event[i].day2-profile->event[i].day1));j++,c++)
         {
          if(m==0.0) r=r1;                   /* remain at current alt */
          else if(m < 0.0) r=(int)(r1+m*j);    /* ascend  */
          else if(m > 0.0) r=(int)(r1+m*j);    /* descend */
          VidPutAttr(WHITE<<4IBLUE,r,c,r0,c);
         } /** end for **/
       } /** end else **/
   } /**  end for **/
   return 0;
 } /*** end function int draw_profile(...) ***/


/*********************** source file: EVNTSORT.C ********************/

#include "alt.h"
#include <search.h>


int sort_events(struct input_profile_struct *p)
 {
  qsort( (void *)p->event, (size_t)p->num_events,
           sizeof( struct event_struct ),
          compare_event_dates);
  }

int compare_event_dates( struct event_struct *event1,
                         struct event_struct *event2 )
{
  if( event1->day1 > event2->day1 )
        return 1;
  else if( event1->day1 < event2->day1 )
        return -1;
  else
        return 0;
}
```

/***

The lengthy source code for the functions comprising the cold and heat stress modules is not included but is available upon request from the author.

***/

DISTRIBUTION LIST

2 Copies to:

Defense Technical Information Center
ATTN: DTIC-SDAC
Alexandria, VA 22304-6145

Office of the Assistant Secretary of Defense (Hlth Affairs)
ATTN: Medical Readiness
Washington, D.C. 20301-1200

Commander
U.S. Army Medical Research and Materiel Command
ATTN: SGRD-OP
Fort Detrick
Frederick, MD 21702-5012

Commander
U.S. Army Medical Research and Materiel Command
ATTN: SGRD-PLE
Fort Detrick
Frederick, MD 21702-5012

Commander
U.S. Army Medical Research and Materiel Command
ATTN: SGRD-PLC
Fort Detrick
Frederick, MD 21702-5012

Commandant
Army Medical Department Center and School
ATTN: HSHA-FM, Bldg. 2840
Fort Sam Houston, TX 78236

1 Copy to:

Joint Chiefs of Staff
Medical Plans and Operations Division
Deputy Director for Medical Readiness
ATTN: RAD Smyth
The Pentagon, Washington, D.C. 20310

HQDA
Office of the Surgeon General
Preventive Medicine Consultant
ATTN: SGPS-PSP
5109 Leesburg Pike
Falls Church, VA 22041-3258

Commander
U.S. Army Environmental Hygiene Agency
Aberdeen Proving Ground, MD 21010-5422

Director, Biological Sciences Division
Office of Naval Research - Code 141
800 N. Quincy Street
Arlington, VA 22217

Commanding Officer
Naval Medical Research and Development Command
NMC/ Bldg. 1
Bethesda, MD 20889-5044

Commanding Officer
U.S. Navy Clothing & Textile Research Facility
ATTN: NCTRF-01
Natick, MA 01760-5000

Commanding Officer
Navy Environmental Health Center
2510 Walmer Avenue
Norfolk, VA 23513-2617

Commanding Officer
Naval Medical Research Institute
Bethesda, MD 20889

ADA294006

Commanding Officer
Naval Health Research Center
P.O. Box 85122
San Diego, CA 92138-9174

Commander
USAF Armstrong Medical Research Laboratory
Wright-Patterson Air Force Base, OH 45433

Commander
USAF Armstrong Medical Research Laboratory
ATTN: Technical Library
Brooks Air Force Base, TX 78235-5301

Commander
USAF School of Aerospace Medicine
Brooks Air Force Base. TX 78235-5000

Commander
U.S. Army Medical Research Institute of Chemical Defense
ATTN: SGRD-UVZ
Aberdeen Proving Ground, MD 21010-5425

Commander
U.S. Army Medical Materiel Development Activity
ATTN: SGRD-UMZ
Fort Detrick
Frederick, MD 21702-5009

Commander
U.S. Army Institute of Surgical Research
ATTN: SGRD-UMZ
Fort Sam Houston, TX 21702-6200

Commander
U.S. Army Medical Research Institute of Infectious Diseases
ATTN: SGRD-UIZ
Fort Detrick
Frederick, MD 21702-5011

Director
Walter Reed Army Institute of Research
ATTN: SGRD-UWZ-C (Director for Research Management)
Washington. D.C. 20307-5100

ADA294006

Commander
U.S \.my Natick Research, Development & Engineering Center
ATTN: SATNC-Z
Natick, MA 01760-5000

Commander
U.S. Army Natick Research, Development & Engineering Center
ATTN: SATNC-T
Natick, MA 01760-5000

Commander
U.S. Army Natick Research, Development & Engineering Center
ATTN: SATNC-MIL
Natick, MA 01760-5000

Director
U.S. Army Research Institute for the Behavioral Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Commander
U.S. Army Training and Doctrine Command
Office of the Surgeon
ATTN: ATMD
Fort Monroe, VA 23651-5000

HQDA
Assistant Secretary of the Army for
Research, Development and Acquisition
ATTN: SARD-TM
The Pentagon, Washington, D.C. 20310

HQDA
Office of the Surgeon General
ATTN: DASG-ZA
5109 Leesburg Pike
Falls Church, VA 22041-3258

HQDA
Office of the Surgeon General
ATTN: DASG-MS
5109 Leesburgh Pike
Falls Chaurch, VA 22041-3258

HQDA
Office of the Surgeon General
Assistant Surgeon General
ATTN: DASG-RDZ/Executive Assistant
Room 3E368, The Pentagon
Washington, D.C. 20310-2300

Dean, School of Medicine
Uniformed Services University of the Health Sciences
4301 Jones Bridge Road
Bethesda, MD 20814-4799

Commandant
U.S. Army Medical Department Center & School
Stimson Library
ATTN: Chief Librarian
Bldg. 2840, Room 106
Fort Sam Houston, TX 78234-6100

Commandant
U.S. Army Medical Department Center & School
ATTN: Director of Combat Development
Fort Sam Houston, TX 78234-6100

Commander
U.S. Army Aeromedical Research Laboratory
ATTN: SGRD-UAX-SI
Fort Rucker, AL 36362-5292

Director
U.S. Army Research Laboratory
Human Research & Engineering Directorate
ATTN: SLCHE-SS-TS
Aberdeen Proving Ground, MD 21005-5001

Technical Director
Defence and Civil Institute of Environmental Medicine
1133 Sheppard Avenue W.
P.O. Box 2000
Downsview, Ontario
CANADA M3M 3B9

Commander
U.S. Army Military History Institute
ATTN: Chief, Historical Reference Branch
Carlisle Barracks
Carlisle, Pennsylvania 17013-5008

Commander
U.S. Army Natick Research, Development and Engineering Center
ATTN: SATNC-TM
U.S. Marine Corps Representative
Natick, MA 01760-5004