Design of a Patient Monitoring System for Cardiopulmonary Bypass Surgery

by

Cynthia K. Rice

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

L. J. Arp, Chairman

M. J. Furey

D. G. Larsen

July 13, 1989

Blacksburg, Virginia

Design of a Patient Monitoring System

for Cardiopulmonary Bypass Surgery

by

Cynthia K. Rice

L. J. Arp, Chairman

Mechanical Engineering

(ABSTRACT)

A patient monitoring system for cardiopulmonary bypass surgery has been developed. This monitoring system uses a SWAN 286-10 computer (fully IBM PC/AT compatible) and a DT2801-A Input/Output board to monitor seven surgical parameters. This system monitors six temperatures, the hemoglobin content, the arterial oxygen saturation, the venous oxygen saturation, the oxygen consumption, and the blood flow rate through the cardiopulmonary bypass circuit. Additionally, there are three individual timers available. Details and the evaluation of the hardware and software design of this monitoring system are presented. Also, recommendations for clinical use are discussed.

Acknowledgements

I would like to express my gratitude to the following people for their assistance during my research:

Dr. Leon J. Arp, for serving as my committee chairman, for his patience, guidance, encouragement, and constant accessibility during my graduate and undergraduate careers.

Dr. M. J. Furey and Professor D. G. Larsen, for serving on my committee.

Mr. Robert Wynn and Dr. Harry Robertshaw, for the use of their Data Translation board. Also, Mr. Ben Poe, for the installation and assistance with this board.

Mr. Al Jester, for his friendship, patience, and smiles during our time as graduate students.

Dr. Allen Caswell, for his constant friendship, support, sense of humor, and for keeping me out of trouble over the past six years. Also for sharing his knowledge of programming.

My family, for their love and support.

Finally, my wonderful parents, their endless love, support, and encouragement, for believing in me, and for teaching me to believe in myself.

Table of Contents

List of Illustrations

List of Tables

I. INTRODUCTION

Problem Overview

A monitoring system for open heart surgery has been developed using a SWAN 286-10 (IBM PC/AT clone). This system is used by the perfusionist to monitor several vital parameters during the cardiopulmonary bypass procedure. It performs the following functions:

- monitors and displays the temperatures from six different temperature probes
- displays three individual timers
- monitors and displays a calculated value for blood flow rate through the patient (dependent upon the pump rate)
- monitors and displays the actual blood flow rate through the patient
- monitors and displays the hemoglobin content of the blood

- monitors and displays the venous oxygen saturation of the blood

- monitors and displays the arterial oxygen saturation of the blood

- monitors and displays the oxygen consumption during the cardiopulmonary bypass procedure

A maximum of six different temperature measurements can be taken using this monitoring system. Six temperature measurements are necessary to completely monitor the condition of the patient during cardiopulmonary bypass surgery. These six temperatures are: (1) core temperature of the patient; (2) skin temperature of the patient; (3) temperature of the water at the entrance of the heat exchanger water bath; (4) temperature of the water leaving the heat exchanger water bath; (5) temperature of the blood entering the blood oxygenator; and (6) temperature of the blood leaving the blood oxygenator [1, 2].

The individual timers can be used to time various events during the surgical procedure. Possible events to time during open heart surgery are the total profusion time, time to cross clamp the aorta, time to adequately oxygenate the blood, or the time to complete each bypass procedure (time to perform each distal anastomosis or the surgical connection between two blood vessels).

The calculated value for blood flow rate through the patient is found noninvasively by using the blood pump output. The blood pump output is normalized in two ways to find the normalized blood flow rate in this monitoring system. One procedure used to find the blood flow rate required is to divide the blood pump output by the patient's body surface area. In the other procedure, the blood flow rate required is found by dividing the blood pump output by the patient's weight. Either a roller or pulsatile blood pump can be used with this monitoring system. A roller pump has multiple arms with a roller on each protruding end while the other end of the arm is attached to a rotating shaft. As the arm rotates, the rollers pinch tubing filled with blood against the outer casing which causes the blood to flow in the direction of rotation. A drawing of a roller pump is shown in Figure 1. A pulsatile pump has a reciprocating piston that forces a hydraulic fluid against a flexible diaphragm. The force from the hydraulic fluid causes the diaphragm to expand into a spherical cavity filled with blood. When the diaphragm expands, the blood is pushed out of the

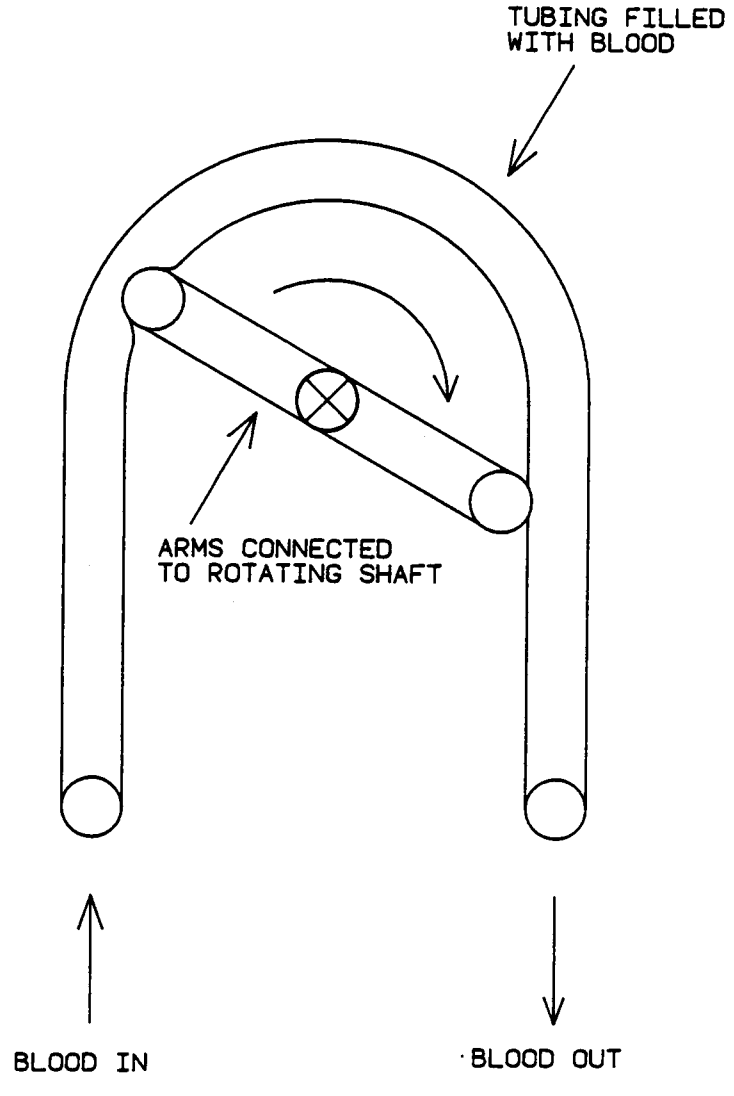

Figure 1. Drawing of a Roller Blood Pump

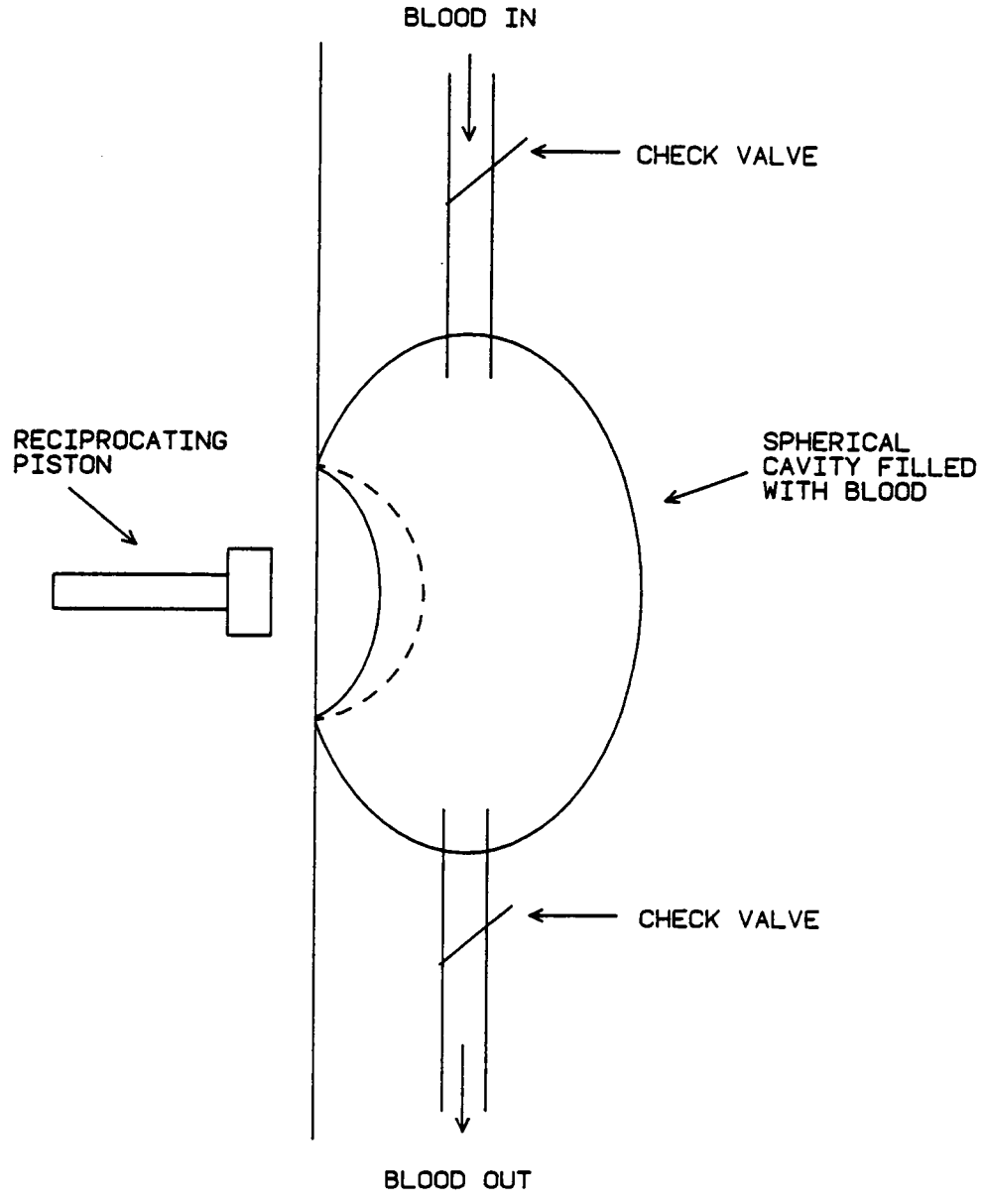

Figure 2. Drawing of a Pulsatile Blood Pump

cavity. There are check valves above and below the cavity so that the blood can flow only in one direction. A schematic of a pulsatile pump is shown in Figure 2. The actual value for blood flow rate through the patient is monitored by a flowmeter placed in the cardiopulmonary bypass circuit.

The hemoglobin content, the venous oxygen saturation, and the arterial oxygen saturation of the blood are monitored during open heart surgery. These parameters along with the actual blood flow rate are used to calculate the oxygen consumption during surgery. Monitoring and controlling oxygen consumption rates ensures that adequate perfusion is taking place.

A Look at Current Monitoring Systems

Originally, the suggestion for an improved monitoring system was presented to Dr. Leon J. Arp by Mr. Edward C. Berger, a retired perfusionist [3]. This original suggestion was acted upon by Mrs. Karen L. Brinkman in her thesis "Design of a Microcomputer-Based Open Heart Surgery Patient Monitor" in October, 1985 [4]. Currently, Dr. Arp has found that monitoring systems for open heart surgery are limited and are not set up for convenient use by the perfusionist.

Present temperature monitoring devices used in open heart surgery measure only four temperatures. Actually, six temperatures are needed to completely monitor the patient's condition during surgery. Also, on the present devices, only one temperature can be shown at a time. In Brinkman's thesis as well as this thesis, all six temperatures are continuously displayed. Additionally, in the monitoring design of this thesis, a graph of two temperatures versus time can be done on the computer screen at the touch of a key.

Present monitoring systems that monitor blood flow rate lack flexibility in areas of input data and the output display. Most monitoring units need the patient weight or patient body surface area input to compute the blood flow rate required. This gives the perfusionist only one flow index. In Brinkman's thesis as well as this thesis, the blood flow rate is given using two flow indices

(ml/min/kg or ml/min/m^2). The perfusionist need enter only the patient height and weight for the computation of the required blood flow rate. Additionally, both values of the blood flow rate are continuously displayed on the computer screen.

In the design of this thesis, the actual blood flow rate is also found by using a flow meter in the cardiopulmonary bypass circuit. The true value of blood flow rate is then compared to the calculated blood flow rate as a check. If the two values are not equal, the proper adjustments can then be made to the pump rate. This ensures that the correct blood flow rate is the actual flow rate of the blood through the patient.

The monitoring device of this thesis also monitors other important parameters during the cardiopulmonary bypass procedure. The hemoglobin content, the venous oxygen saturation, and the arterial oxygen saturation are monitored. These parameters along with the actual blood flow rate are used to calculate oxygen consumption which indicates adequate perfusion of the patient. Also, a graph of hemoglobin content, arterial oxygen saturation, venous oxygen saturation, and oxygen consumption is available by pressing a function key.

The timers in this thesis are similar to other monitoring devices. They are added to the monitoring system as a convenience for the perfusionist. Currently, the perfusionist must look at several different displays to check the critical parameters necessary to monitor a patient during surgery. The monitoring device of this thesis is advantageous because it combines several monitoring devices by using a data acquisition system with a computer. The perfusionist can look at the computer screen where all of the parameters are displayed on one screen instead of looking at several displays.

Problem Solution

The monitoring system of this thesis uses a SWAN 286-10 (IBM PC/AT clone) with an 80286 microprocessor to monitor the various parameters of open heart surgery. This type of computer is used because of its efficiency, reliability, and speed of operation. This monitoring system allows the use of various units of measure when inputting data which should eliminate conversion errors. Additionally, any of the input parameters can be changed or corrected during any part of the program by pressing a key. The system can be used with either a roller or pulsatile blood pump. Also, several of the parameters can be shown in graphical form on the computer screen by pressing a key.

II. DESIGN OVERVIEW

The monitoring system for open heart surgery presented in this thesis is shown in Figure 3. There is a SWAN 286-10 (IBM PC/AT clone) computer, an enhanced graphics color display, a computer keyboard, a 4016 CMOS switch, a DT2801-A analog and digital I/O board and a DT707 screw terminal panel.

The patient information and the surgical information is entered into the monitoring system using the keyboard. The patient information that is input is the patient's name, sex, hospital identification number, age, height, and weight. The surgical information entered is names for the six temperatures, the units of temperature, names for the three timers, and the blood pump information. The blood pump information includes the type of blood pump, the stroke volume when using a pulsatile pump, or the number of rollers, the tube diameter, and the arc length when using a roller pump.

The parameters being monitored are entered to the system through the DT707 screw terminal panel. The parameters being input are the six temperatures, the hemoglobin content, the venous oxygen saturation, the arterial oxygen saturation, the flowmeter information, and the tachometer information from the blood pump being used.

The software used in this monitoring system is done in Turbo Basic. The main program consists of a series of subroutine calls. The subroutines called in the main program are for

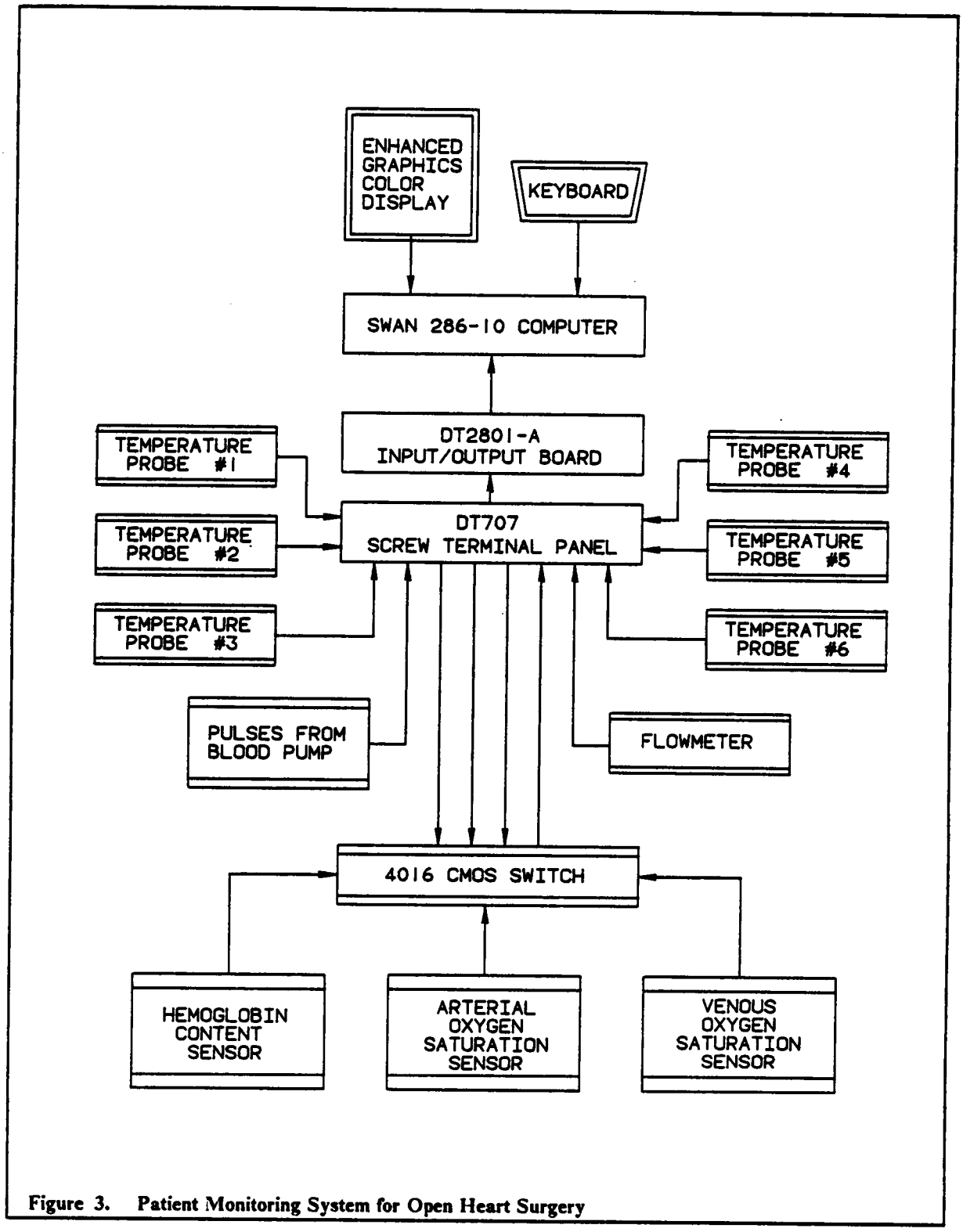

Figure 3. Patient Monitoring System for Open Heart Surgery

initializing the computer and display, the initial patient and surgical information input, the display and correction of this information, and finally, the parameters being monitored are displayed. Additionally, calculations of the patient body surface area, the pump output, the normalized blood flow rates, and the oxygen consumption are performed. Graphs of two temperatures versus time as well as hemoglobin content, venous oxygen saturation, arterial oxygen saturation, and oxygen consumption versus time are also available on the computer screen.

The final display screen during the actual monitoring process shows the patient's name and hospital identification number, the six temperatures, the three timers, the type of blood pump being used, the calculated and actual pump output,the normalized blood flow rates, the hemoglobin content, the venous oxygen saturation, the arterial oxygen saturation, and the oxygen consumption. The program continues to display this screen and monitor these parameters until the user goes back to the information display and correction screen, or request a graph, or presses the escape key to exit the monitoring program.

III. HARDWARE DESIGN

Computer Selection

This patient monitor was designed using a SWAN 286-10 computer which is fully IBM PC/AT compatible. There are several advantages of using this type of computer. The availability and dependability of this computer are key advantages as well as the speed of the 80286 microprocessor. Also, this computer system consists of an enhanced graphics adaptor, an enhanced graphics monitor, a keyboard, a 20 megabyte hard disk, a 1.2 megabyte disk drive for 5.25 inch floppy diskettes, a 1.44 megabyte disk drive for 3.5 inch floppy diskettes, one parallel communications port, one serial communications port, and eight expansion slots for Input/Output connections. The operation of this system requires approximately 6 kilobytes of run-time memory and 470 kilobytes of disk storage space [5].

The enhanced graphics monitor and the enhanced graphics adapter video card result in a special type of color monitor which can display 64 different colors with increased screen resolution. The resolution used in this design is 640 horizontal pixels by 350 vertical pixels when the display is set to the graphics mode. This allows for finer detail when doing graphics work [6].

Communication with the computer is accomplished easily using the keyboard. Additionally, data transfer and storage is convenient and simple using one of the three disk drives. Communication with instruments outside of the computer such as printers, temperature sensors, or a mouse is facilitated by the availability of a parallel communications port, a serial communications port, and the eight expansion slots. In this patient monitor design, the parallel communications port is connected to a printer and one of the expansion slots is used for a Data Translations 2801-A Digital and Analog Input/Output board for monitoring the parameters of open heart surgery [7].

Technical Overview of the SWAN 286-10 Computer

The SWAN 286-10 computer is driven by an Intel 80286 microprocessor chip. The 80286 uses a 16-bit data bus which allows the entry of information and instructions to occur in a single operation. This is an improvement over the double fetch technique of the old Intel 8088 microprocessor which uses an 8-bit data bus to enter data in two 8-bit words. Also, the 80286 uses a 24 bit address bus which allows it to address 16 Megabytes of Random Access Memory. However, when the 80286 is used under MS DOS, as in this design, it operates as an 8088 (using 16 bits instead of 8) and can access only 640 kilobytes of memory directly.

Using an 80286 microprocessor results in faster processing, approximately two to three times faster than the 8088. Additionally, the SWAN 286-10 computer has an increased CPU speed of 12.5 MHz, running the 80286 microprocessor at 10 MHz (The IBM PC/AT uses a CPU speed of 12 MHz running the 80286 at 6 MHz). So, the use of an 80286 microprocessor plus the increased CPU speed of the SWAN 286-10 computer allows this program to execute faster. There was a noticeable decrease in the speed of execution on an IBM PC/AT.

The SWAN 286-10 computer system used in this patient monitor design has a full megabyte of random access memory. This memory is partitioned into 640 kilobytes of Base Memory and 284 kilobytes of Extended Memory. The base memory represents the working memory the computer uses for storing data and executing programs. The extended memory is similar to the base memory in that it is used to store immediate machine data and instructions. However, when operating under MS DOS, as in this design, only the 640 kilobytes of base memory can be accessed [8, 9].

Interface Devices

The parameters monitored in this design are six temperatures, blood pump output, hemoglobin content, arterial oxygen saturation, and venous oxygen saturation. All of these parameters are monitored using the circuitry shown in Figure 4. The DT707 screw board panel allows easy connections to the Data Translation 2801-A (DT2801-A) Analog and Digital Input/Output Board. The DT2801-A I/O board is used to interface all of the sensors to the computer. The 4016 CMOS quad bilateral switch chip is used to multiplex three of the sensors into one analog channel of the DT2801-A I/O board.

A schematic of the DT707 screw board panel is shown in Appendix A. This panel allows connections to the DT2801-A I/O board to be made using barrier screw terminals so that no soldering is necessary, and the connections can be easily changed. The DT707 remains outside of the computer for easy access.

The DT2801-A I/O board is used to interface all of the sensors to the computer. A schematic of this board is shown in Appendix A. It is connected to the SWAN 286-10 through one of the eight expansion slots located at the back of the computer. The DT2801-A I/O board was designed

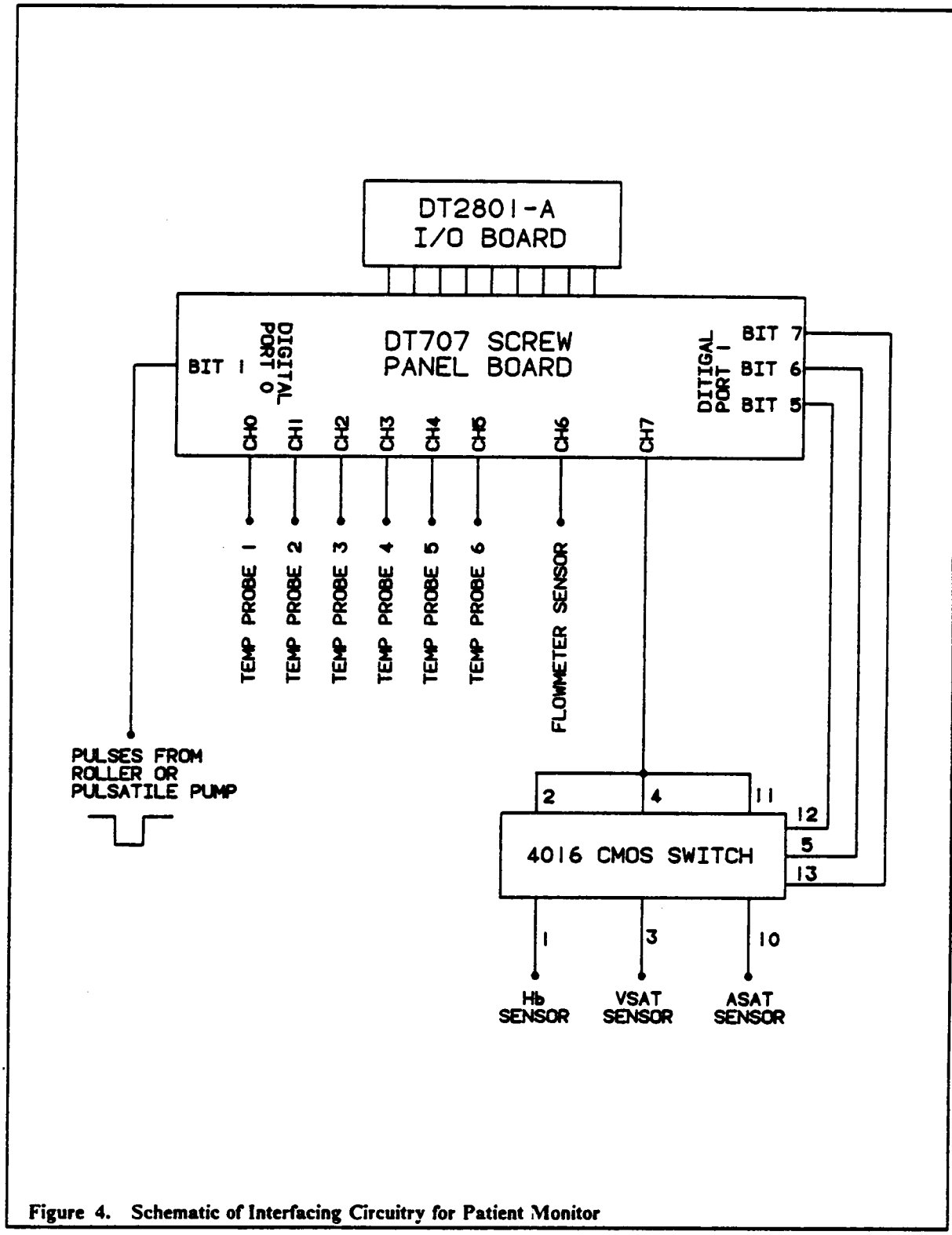

Figure 4. Schematic of Interfacing Circuitry for Patient Monitor

to be controlled by simple commands written in BASIC. However, these commands are also effective when written in Turbo Basic.

The DT2801-A I/O board has two digital input/output ports of eight bits each. The direction of data transfer is specified using software. For this design, Digital Port 0 is set as an input port where bit 1 detects pump rate. Digital Port 1 is specified as an output port with bits 5, 6, and 7 sending pulses to the 4016 CMOS switch multiplexing three sensors into one analog channel of the DT2801-A I/O board.

Also, there are eight analog inputs or channels that can be used for analog to digital conversions or digital to analog conversions. In this design all analog channels are used for analog to digital conversions. Specifically, the channels are used as follows: Channels 0 through 5 are connected to temperature sensors; channel 6 is connected to the flowmeter for pump output detection; and, channel 7 has the hemoglobin content sensor, the arterial oxygen saturation sensor, and the venous oxygen saturation sensor multiplexed into it using a 4016 CMOS switch.

The analog to digital converter used on the DT2801-A I/O board has 12 bits of resolution. This means that the incoming analog signal is converted into a binary number 12 bits long. The accuracy of the A/D converter is +/- 0.05% of reading + 1 count. The DT2801-A I/O board has an A/D conversion time of 10 microseconds [10, 11].

Temperature Detection

There are six temperature sensors connected to analog channels 0 through 5 of the DT2801-A I/O board. Each sensor outputs an analog voltage proportional to the temperature. The voltages range from 0.000 to 4.974. The scaling factor for temperatures in units of Celsius is 0.01999 volts/deg C. For temperature detection in units of Fahrenheit, the scaling factor is 0.010 volts/deg F. The units of temperature are selected by the user in the first phase of the program.

Flow Rate Detection

The flowmeter has a sensor connected to analog channel 6 of the DT2801-A I/O board. This sensor outputs an analog voltage proportional to the flow rate of the blood. The voltages range form 0.000 to 0.999. The scaling factor of this sensor is selected using software and is dependent upon the calculated pump output (computed using software and values of pump rate detected at digital port 0). If the calculated pump output is greater than 1000 ml/min, then the scaling factor is 10 liters/minute per volt. If the calculated pump output is less than or equal to 1000 ml/min, then the scaling factor is 1 liter/minute per volt.

Detection of Hemoglobin Content, Arterial Oxygen Saturation and Venous Oxygen Saturation

The hemoglobin content sensor, arterial oxygen saturation sensor, and the venous oxygen saturation sensor are multiplexed into analog channel 7 of the DT2801-A I/O board using a 4016 CMOS quad bilateral switch chip. A schematic of this circuit is shown in Figure 5. The switches are opened and closed by a sequence of pulses from digital port 1, bits 5, 6, and 7 of the DT2801-A I/O board. These pulses are produced by software. A timing diagram of these pulses is shown in Figure 6.

The timing sequence starts with bit 7 of digital port 1 of the DT2801-A I/O board being taken to logical 1 for 11.5 msec. This closes the switch so that the analog voltage representing the hemoglobin content can be read at analog channel 7 of the DT2801-A I/O board. At the same time bit 7 is high, bits 5 and 6 are held at logical 0 for 11.5 msec. Data is sampled form the hemoglobin

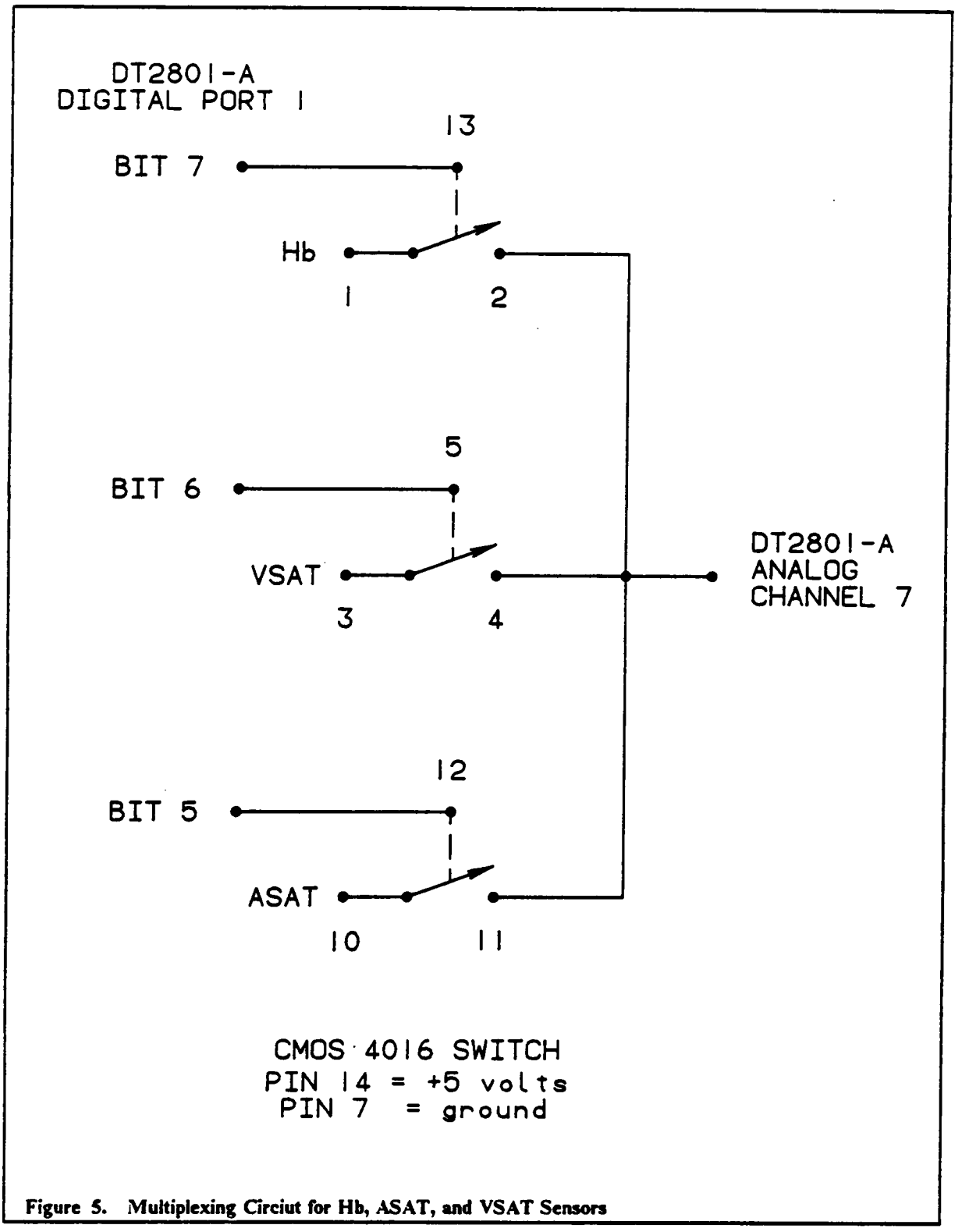

Figure 5. Multiplexing Circiut for Hb, ASAT, and VSAT Sensors

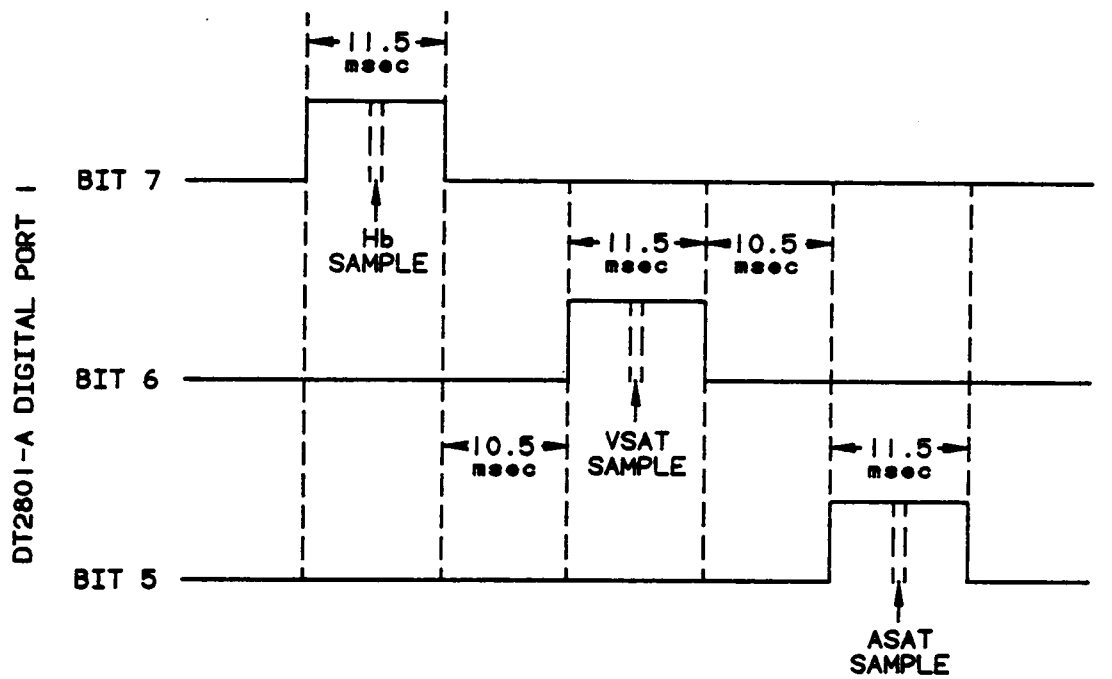

Figure 6. Timing Diagram for Hb, ASAT, and VSAT Sensors

content sensor 5.25 msec after bit 7 is first taken to logical 1. This ensures that the proper data is being sampled for the hemoglobin content. Next, all bits of digital port 1 are taken to logical 0 for 10.5 msec. Then, bit 6 of digital port 1 is taken to logical 1 for 11.5 msec while bits 5 and 7 are held at logical 0. Taking bit 6 to logical 1 closes the switch so that the analog voltage representing the venous oxygen saturation can be read at analog channel 7. Once again, the venous oxygen saturation data is sampled 5.25 msec after bit 6 is first taken to logical 1. Next, all bits of digital port 1 are taken to logical 0 for 10.5 msec. Then, bit 5 of digital port 1 is taken to logical 1 for 11.5 msec while bits 6 and 7 are held at logical 0. Taking bit 5 to logical 1 closes the switch for the arterial oxygen saturation sensor. Data from this sensor is sampled 5.25 msec after bit 5 is first taken to logical 1. Finally, all bits of digital port 1 are taken to logical 0. Sampling one value from each of the three sensors takes approximately 56 msec.

The hemoglobin content sensor has a voltage range of 0.000 to 1.999 volts. For the hemoglobin content sensor, one volt represents 10 grams of hemoglobin per deciliter of blood. The arterial and venous oxygen saturation sensors produce voltages ranging from 0.000 to 0.999 volts. The voltage from each sensor is multiplied by 100 to give a percentage value for arterial oxygen saturation and venous oxygen saturation.

Pump Rate Detection

The pump rate is detected by pick-up devices attached to the roller pump or pulsatile pump. The pick-up device for a roller pump produces 100 low-going pulses per revolution of the roller. The pick-up device for a pulsatile pump produces 1 low-going pulse per pulse of the diaphragm. The output of the pick-up device is sent to the DT2801-A I/O board. The DT2801-A I/O board detects the pulses from the pick-up device of the pump at bit 1 of digitial port 0. Software counts these pulses for one second. Then, the pump rate and the pump output is calculated using software.

IV. SOFTWARE DESIGN

Program Overview

This monitoring design consists of software written in Turbo Basic. Due to the speed of the 80286 microprocessor being used, it is not necessary to write any part of the software in assembly language. Turbo Basic is a high level compiler language and is much easier to use than assembly language. Additionally, it is more efficient and runs faster than standard interpreted BASIC and BASICA [12].

The main program is entitled MONITOR. A flowchart of the MONITOR program is shown in Figure 7. A listing of the entire program is presented in Appendix B. The main program first increases the run-time memory of the computer to 6 kilobytes. This is necessary due to the recursive nature of the program. Then, the CLEAR command is executed which sets all numeric variables to zero and initializes array and string memory. The remainder of the main program consists of four subroutine calls. The first subroutine called is INTRO. This subroutine sets the computer and the enhanced graphics display so that the remaining subroutines are executed properly. The second subroutine called is INFOIN. The subroutine INFOIN is used for the input of the initial patient and surgical information. The third subroutine called is DISINFO. This

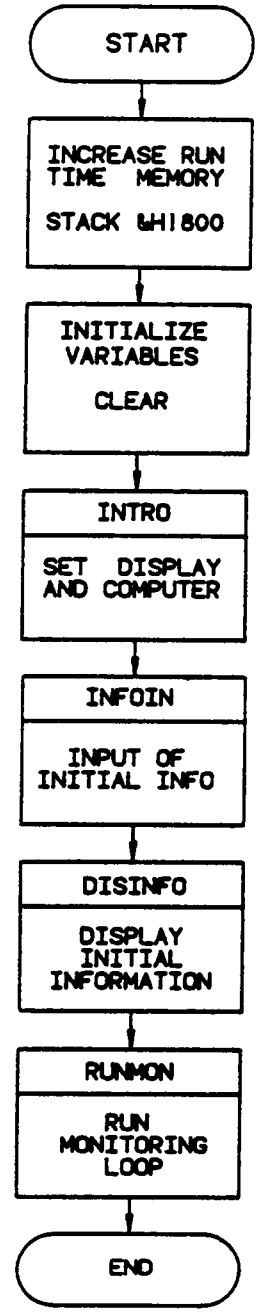

Figure 7. Flowchart of Main Program MONITOR

subroutine displays the initial information that was input during the execution of the INFOIN subroutine. Also, the DISINFO subroutine allows the correction of any of the information initially input. Finally, the last subroutine called is RUNMON. This subroutine is called for the display of the parameters being monitored. The parameters are displayed in tabular form; however, some can also be shown in graphical form.

Preparation of Computer and Display

The INTRO subroutine is a short subroutine used to set the computer and enhanced graphics display so that the remaining subroutines will execute correctly. A flowchart of INTRO is shown in Figure 9. First, the display is set to the graphics mode and immediately after that, it is set to the text mode with color enabled which allows for faster clearing of the screen. Next, the subroutine SETESCAPE is called to set the escape key for exiting of the program. A flowchart of SETESCAPE is shown in Figure 10. After each statement, Turbo Basic checks to see if the escape key has been pressed. The num lock function and the cap lock function must be in the off position for pressing of the escape key to be detected. If the escape key has been pressed, control of the program jumps to the subroutine ENDALL. A flowchart of ENDALL is shown in Figure 11. Then, the user is asked, "Are you sure you want to exit?", and the computer waits for a yes or no response. If the response is yes, execution of the program ends. If the response is no, the subroutine returns control to the beginning of the subroutine which was running when the escape key was pressed. Finally, after setting the escape key, control returns to the main program MONITOR.

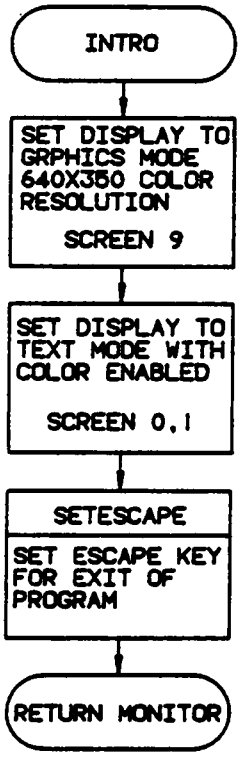

Figure 8. Flowchart of Subroutine INTRO

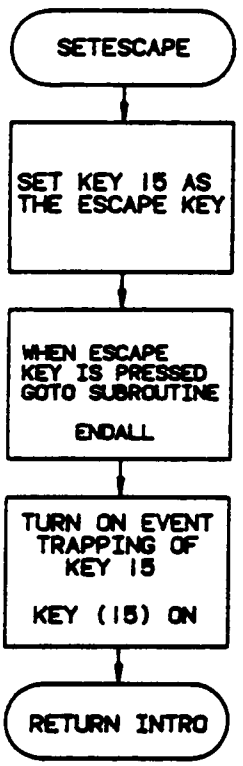

Figure 9. Flowchart of Subroutine SETESCAPE

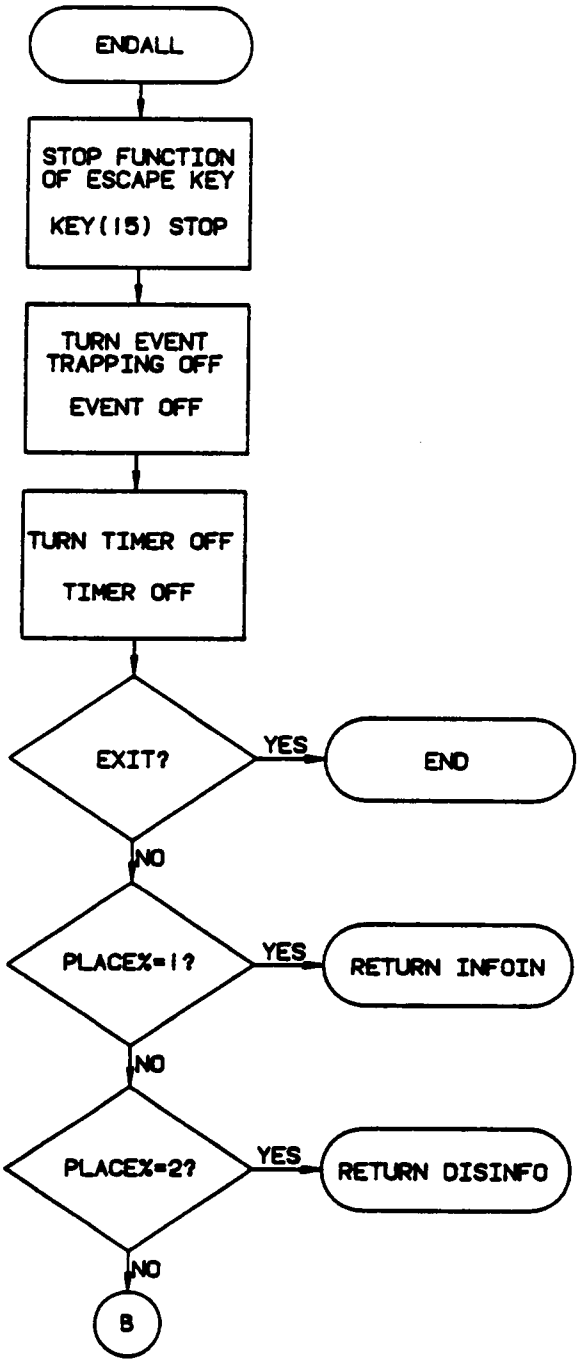

Figure 10a. Flowchart of Subroutine ENDALL

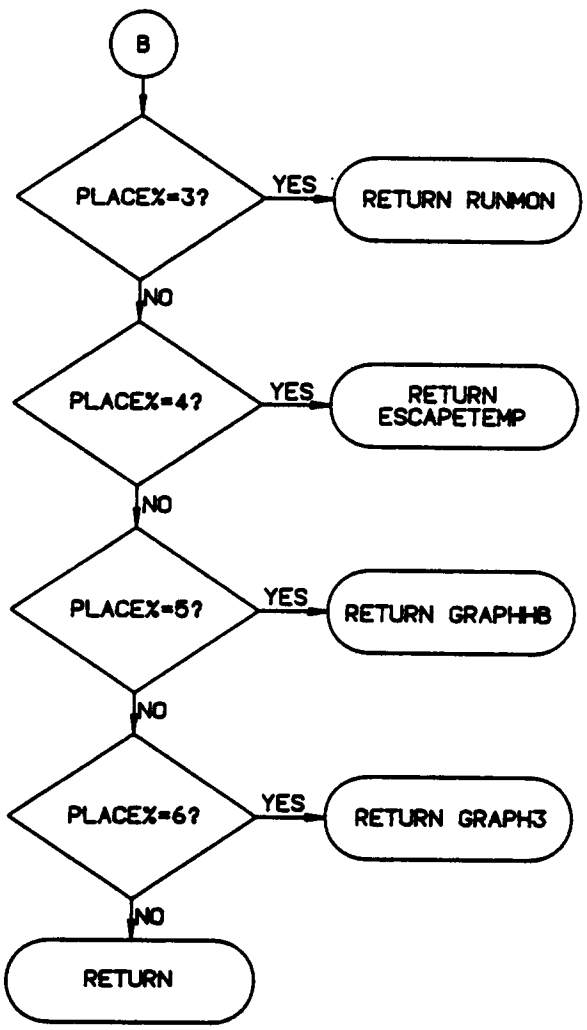

Figure 10b. Flowchart of Subroutine ENDALL

Initial Information Input

The subroutine INFOIN is called for the initial input of patient and surgical information. This subroutine first sets a flag so that if the escape key is pressed during its execution, the escape key subroutine, ENDALL, knows where to return if needed. Next, the screen is cleared and control of the program is sent to several subroutines which prompt the user to input patient information and surgical information. A flowchart of INFOIN is shown in Figure 11.

The first subroutine called is GETPATINFO. As shown in Figure 12, this subroutine calls nine other subroutines which ask the user for the patient information. The user is either asked to type in the information or select the proper input from the information displayed on the screen. This selection process is done using the up and down arrow keys to move a selection box to the proper input and pressing the return key to select the input. The user either inputs or selects the following information:

- Name of Patient
- Sex of Patient (male or female)
- Hospital Identification Number
- Age (in months or years)
- Units of Age (months or years)
- Height (in centimeters or inches)
- Units of Height (centimeters or inches)
- Weight (in kilograms or pounds)

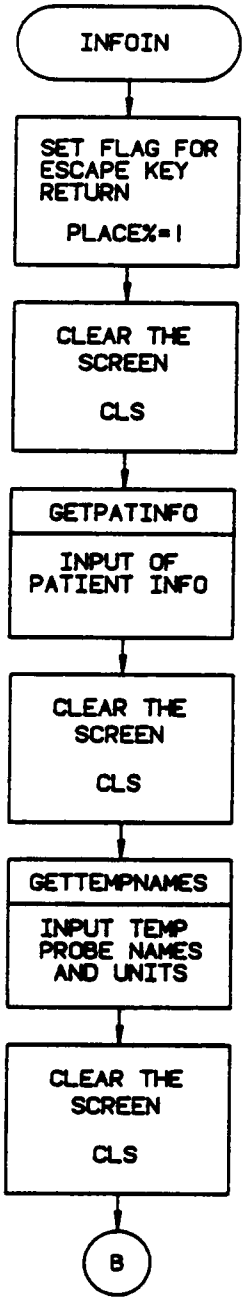

Figure 11a. Flowchart of Subroutine INFOIN

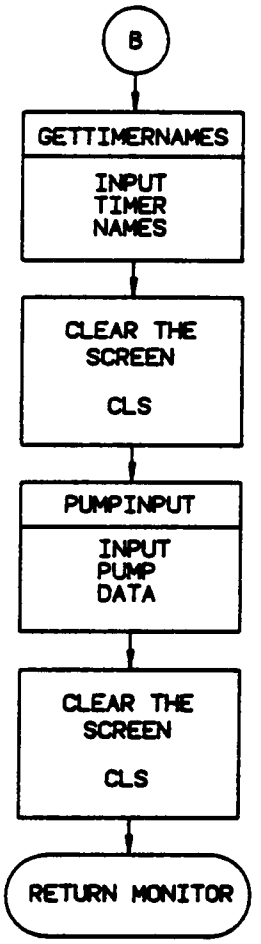

Figure 11b. Flowchart of Subroutine INFOIN

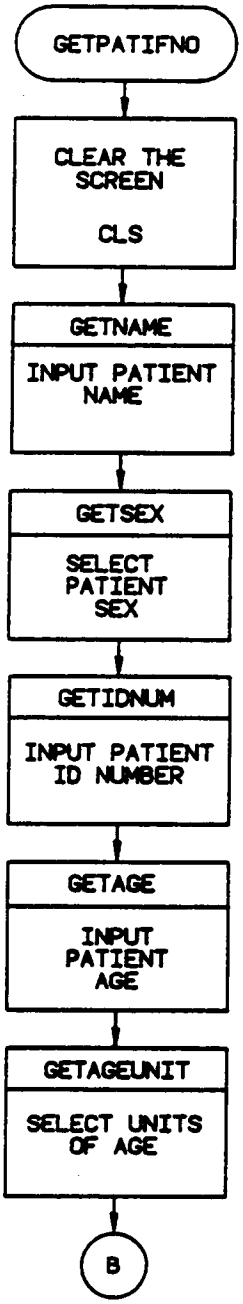

Figure 12a. Flowchart of Subroutine GETPATINFO

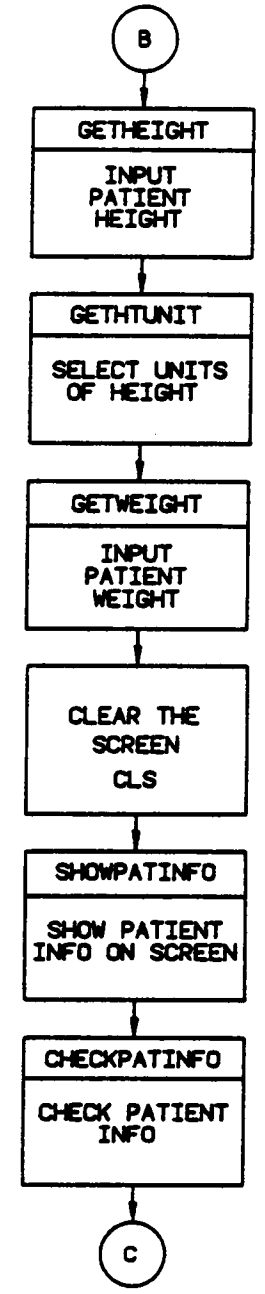

Figure 12b. Flowchart of Subroutine GETPATINFO

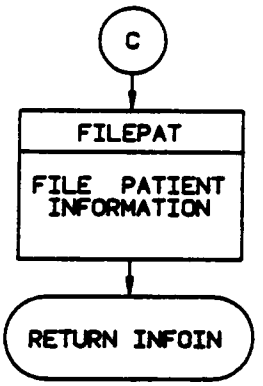

Figure 12c. Flowchart of Subroutine GETPATINFO

- Units of Weight (kilograms or pounds)

After each input or selection, the user is asked if the information is correct and is given the option to correct the information if necessary. Also, if numbers less than zero or characters are entered for the age, height, or weight, the computer beeps for one half second and a message flashing on and off tells the user that this is not a valid input and to please enter another value.

After all patient information has been entered, the screen is cleared and three more subroutines are called. These subroutines cause all patient information to be displayed on the screen. The user is asked if the information is correct and given the opportunity to change any incorrect information.

Next, control is passed to the subroutine GETTEMPNAMES. This subroutine is shown Figure 13. It calls several subroutines which ask the user to input or select the temperature probe information. Initially, the user is asked to select the temperature units of either Celsius or Fahrenheit. Once again, the up and down arrow keys are used to move a selection box to the proper units of temperature and the return key is pressed to select the input. Next, the user is asked to enter the name of the temperature probes or "N/C" for probes not connected. The names are limited to 15 characters or less. The user is given the opportunity to correct the name after each input. After all six temperature names have been entered, the screen is cleared and the temperature names are displayed again to give the user another chance to change the temperature names if desired.

Following the temperature information input section, the timer names are entered. The subroutine called to do this is entitled GETTIMERNAMES and is shown in Figure 14. Each timer name is limited to 15 characters or less. The user is given the opportunity to change the name after each input. After all three timer names have been entered, the screen is cleared and the names are displayed again to give the user another chance to change the timer names if desired.

In the final input section, the subroutine PUMPINPUT is called. As shown in Figure 15, this subroutine is responsible for the input of the blood pump information. First, control is sent to the subroutine PUMPTYPE (see flowchart in Figure 16) where the user is asked to select the type of blood pump to be used, either pulsatile or roller. Again, this selection is done using the up and

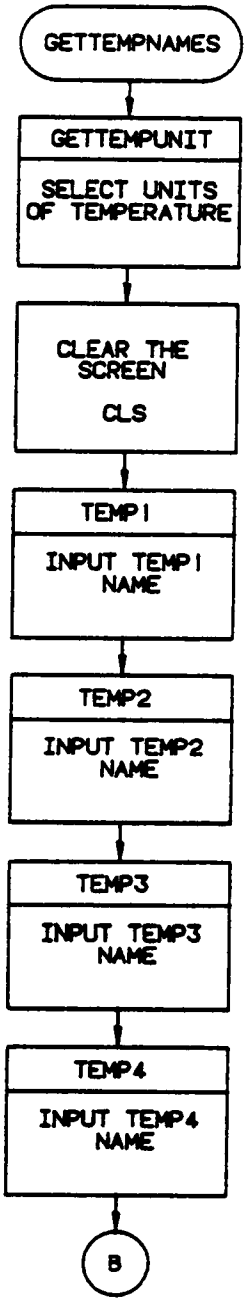

Figure 13a. Flowchart of Subroutine GETTEMPNAMES

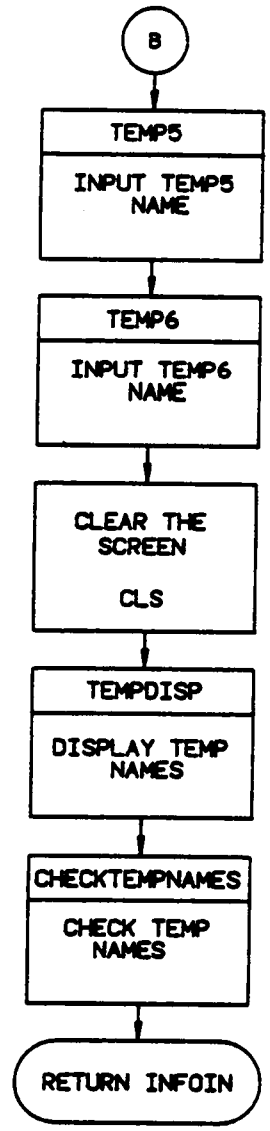

Figure 13b. Flowchart of Subroutine GETTEMPNAMES

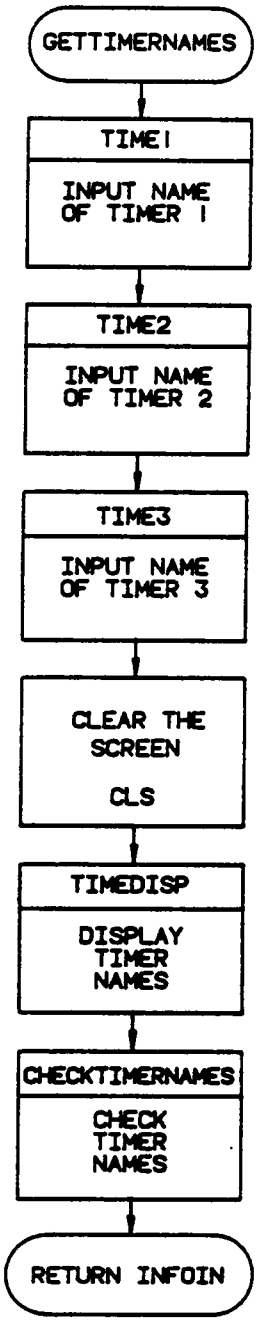

Figure 14. Flowchart of Subroutine GETTIMERNAMES

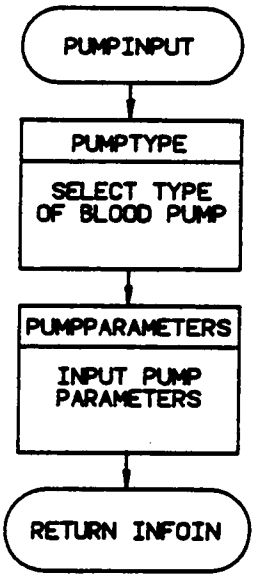

Figure 15. Flowchart of Subroutine PUMPINPUT

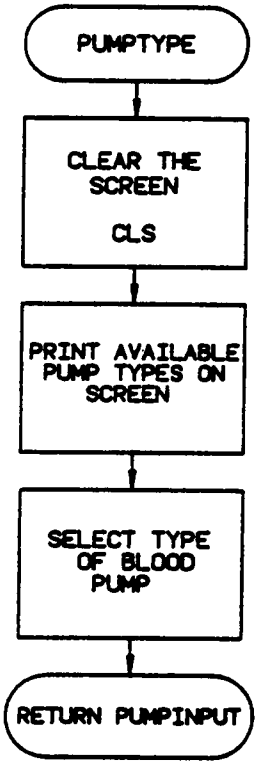

Figure 16. Flowchart of Subroutine PUMPTYPE

down arrow keys to move a selection box to the proper type of blood pump and the return key is pressed to select the input. After the pump type has been selected, the PUMPPARAMETERS subroutine is called. A flowchart of this subroutine is shown in Figure 17. If a pulsatile pump is selected, the user is asked to enter the stroke volume of the pulsatile pump in cubic centimeters. If a roller pump is selected, the user is asked to either input or select the following parameters:

- number of rollers
- tube diameter (in centimeters or inches)
- tube diameter units (centimeters or inches)
- arc length (in centimeters or inches)
- arc length units (centimeters or inches)

Again, after each input or selection, the user is asked if the information is correct and is given the option to correct the information if necessary. Also, if numbers less than zero or characters are entered for the number of rollers, the tube diameter, or the arc length, the computer beeps for one half second and a message which blinks on and off tells the user that the input is not valid and to please enter another value. After all of the roller pump parameters have been entered, the screen is cleared. Then the parameters are displayed again and the user is given the option to make any needed corrections. Control is then passed back to the main program, MONITOR, where the subroutine DISINFO is called next.

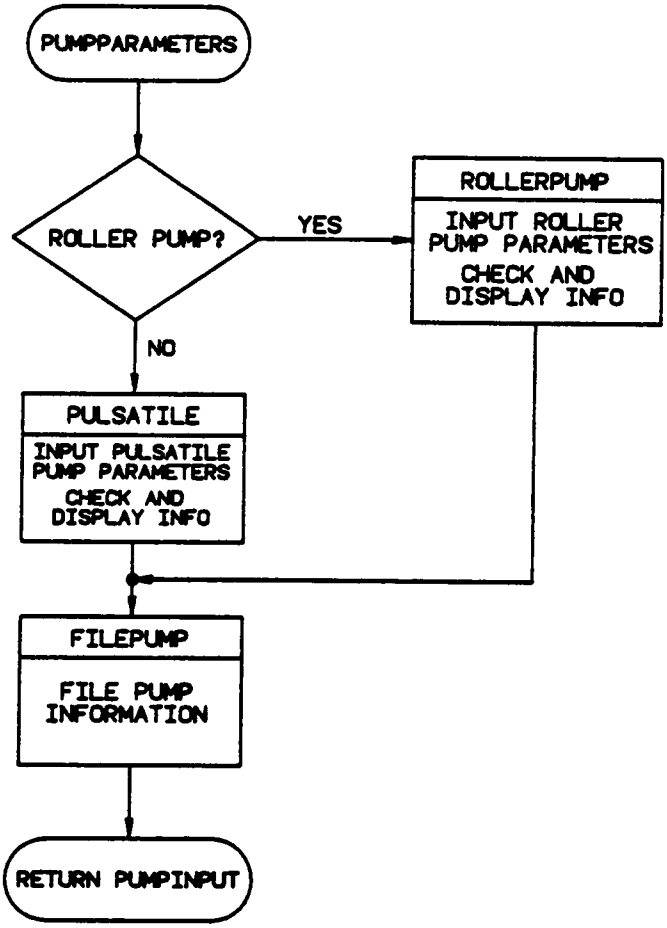

Figure 17. Flowchart of Subroutine PUMPPARAMETERS

Display and Correction of Patient and Surgical Information

The subroutine DISINFO displays all of the patient and surgical information. Also, it allows the user to change any of this displayed information. A flowchart of this subroutine is shown in Figure 18. First, this subroutine sets a flag so that if the escape key is pressed during its execution, the escape key subroutine, ENDALL, knows where to return if needed. Next, all function keys are disabled and the screen is cleared. Then, five subroutines are called which display the patient information, the pump information, the timer names, the temperature probe names, the temperature units, and an option list at the bottom of the screen as shown in Figure 19. The last subroutine called, DRAWFKBOX, also enables function keys 1, 2, 3, 4, and 5. The option list informs the user to press function key 1 (FK1) for the correction mode, function key 5 (FK5) to continue the program, or the ESCAPE key to exit the program.

Selecting the correction mode, FK1, causes a function key menu to be displayed as shown in Figure 20. The function keys are set to do the following: FK1 allows the user to change the patient information; FK2 allows the user to change the blood pump information; FK3 allows the user to change the timer names; FK4 allows the user to change the temperature units and/or the temperature probe names; and, FK5 allows the user to exit the correction mode and return to the original display screen as shown in Figure 19.

If the correction mode is not selected and the user presses FK5 to continue, the program then goes to the subroutine RUNMON to begin the monitoring section of the program which will be discussed next. If the escape key is pressed at this time, the program sends control to the escape key subroutine, ENDALL, as discussed previously.

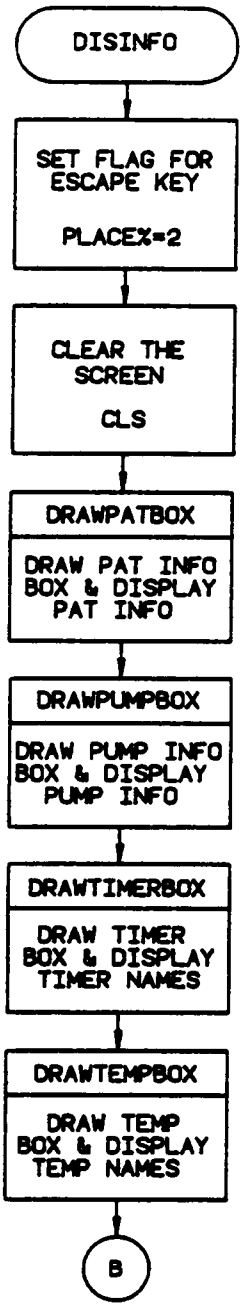

Figure 18a. Flowchart of Subroutine DISINFO

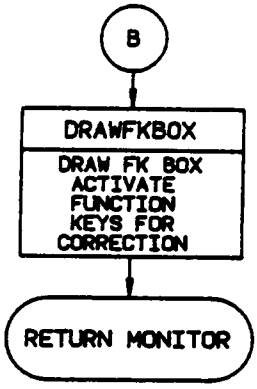

Figure 18b. Flowchart of Subroutine DISINFO

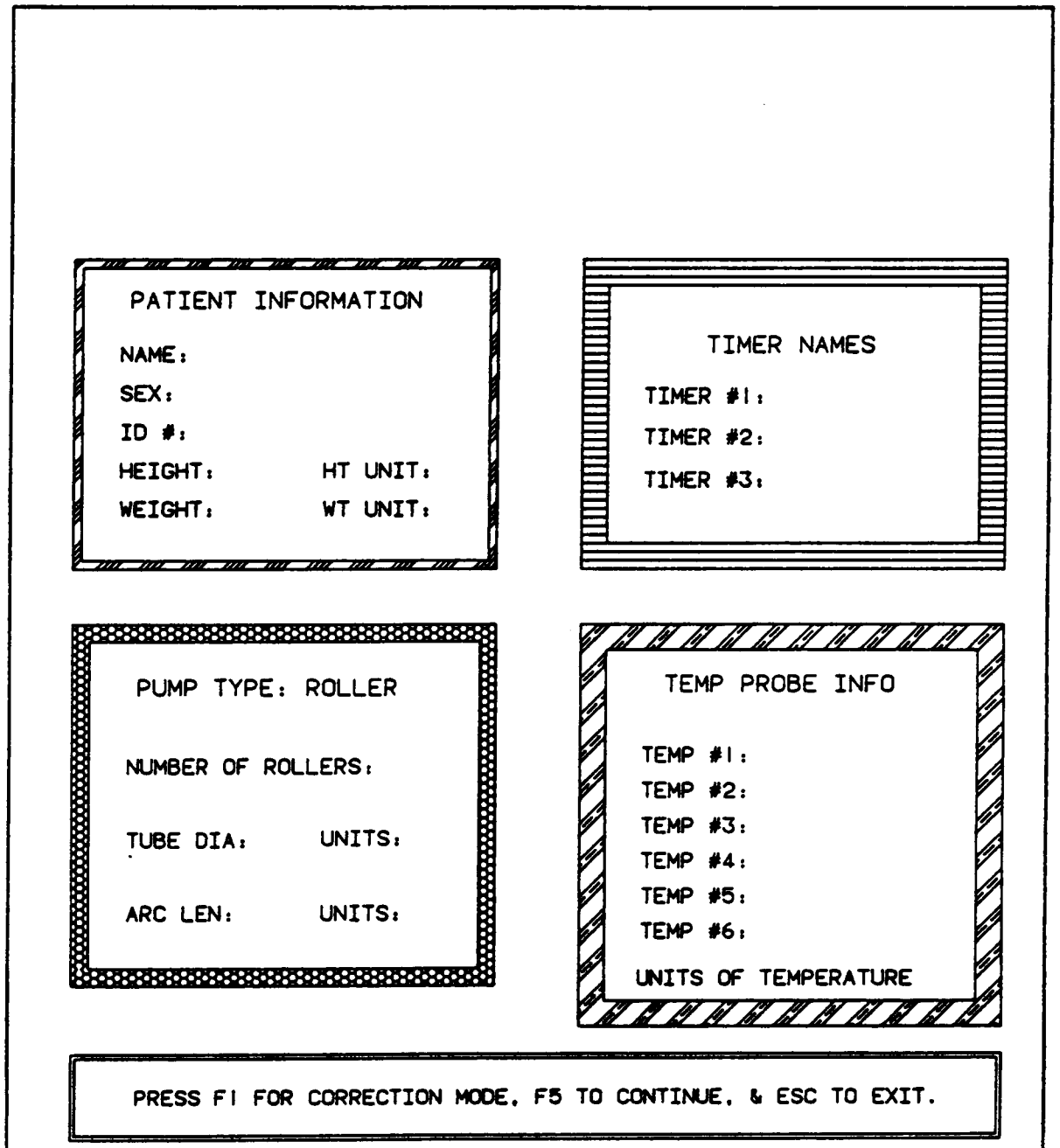

Figure 19. Drawing of Display and Correction Screen

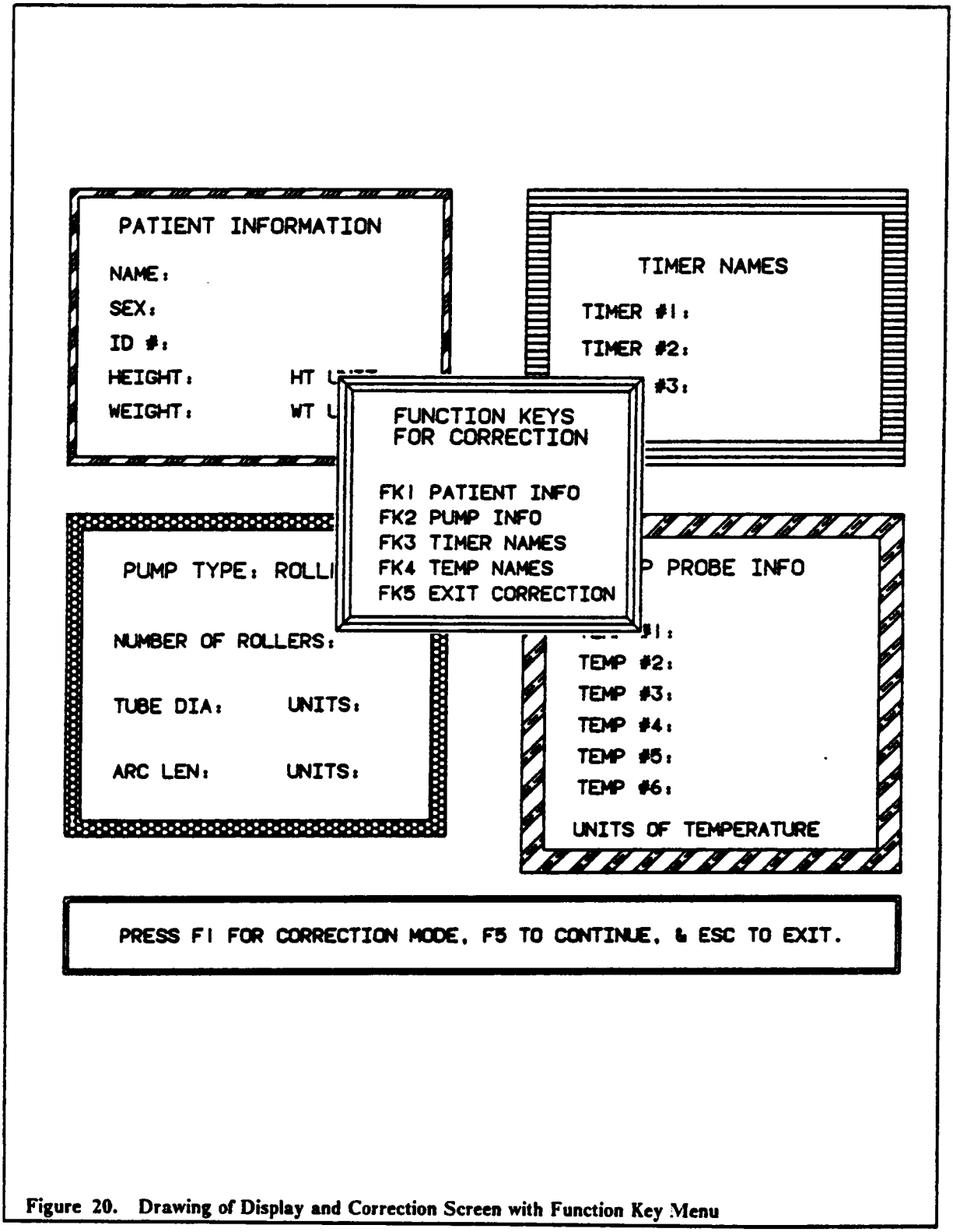

Figure 20. Drawing of Display and Correction Screen with Function Key Menu

Display of Monitored Parameters

The subroutine RUNMON displays the parameters being monitored. These parameters are initially displayed in tabular form, but some can also be shown in graphical form. A flowchart of the RUNMON subroutine is shown in Figure 21. First, this subroutine turns the internal timer of the computer off. Next, a flag is set so that if the escape key is pressed during the execution of this subroutine, the escape key subroutine, ENDALL, knows where to return if needed. Next, the screen is cleared. Then the subroutine INITMON is called.

As shown in Figure 22, in subroutine INITMON the unused function keys are disabled. Function key 5 (FK5) is set to return to the display and correction mode (DISINFO subroutine) as discussed in the previous section. Function key 6 (FK6) is set to call a graphing subroutine.

The graphing subroutine is called GRAPH3 and is shown in Figure 23. This subroutine first sets a flag so that if the escape key is pressed during the execution of this subroutine, the escape key subroutine ,ENDALL, knows where to return if needed. Then, the internal timer of the computer is turned off. Next, all function keys are disabled and the screen is cleared. Then, subroutine GRAFBOX is called to display the selection screen so that the parameters to be graphed can be selected. Again, this selection is done using the up and down arrow keys to move a selection box to the parameters to be graphed and the return key is pressed to select the input. Two selections are possible, a graph of two temperatures versus time and a graph of Hemoglobin content, arterial oxygen saturation, venous oxygen saturation, and oxygen consumption versus time.

If a graph of temperatures is desired, then the first line is selected and control of the program goes to the subroutine GRAPHTP. As shown in Figure 24, this subroutine allows the user to enter the numbers of the two temperatures to be graphed. Then, a graph, as shown in Figure 25, is displayed on the screen with a plot of the current temperatures versus time. The graph updates every thirty seconds. Also, the numerical values of the two temperatures are displayed in two boxes on the left side of the graph and a function key menu is displayed. Function key 1 (FK1) is set to

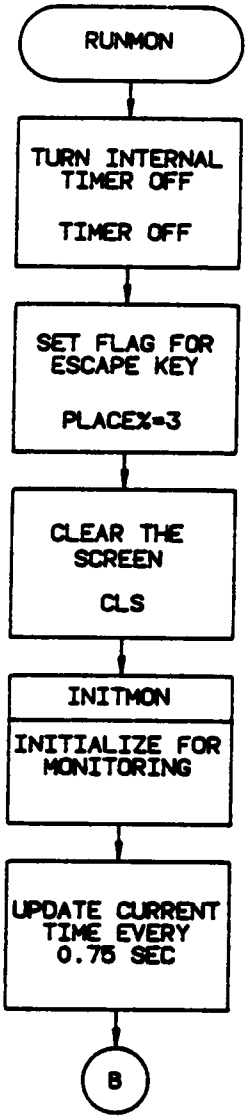

Figure 21a. Flowchart of Subroutine RUNMON

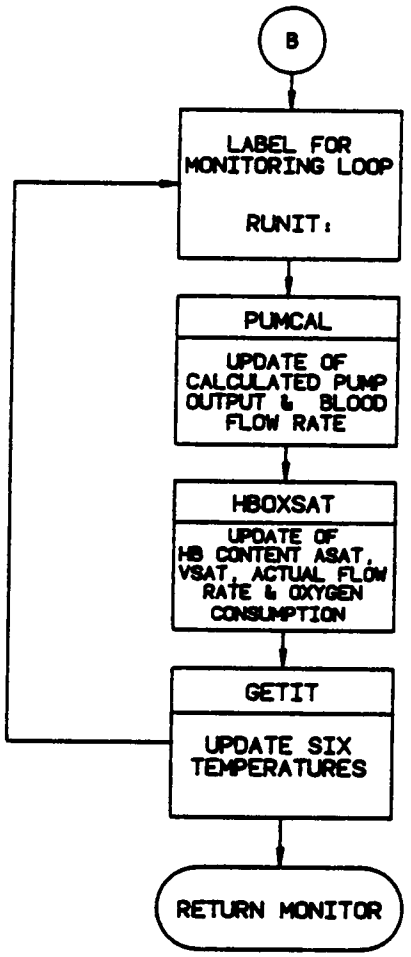

Figure 21b. Flowchart of Subroutine RUNMON

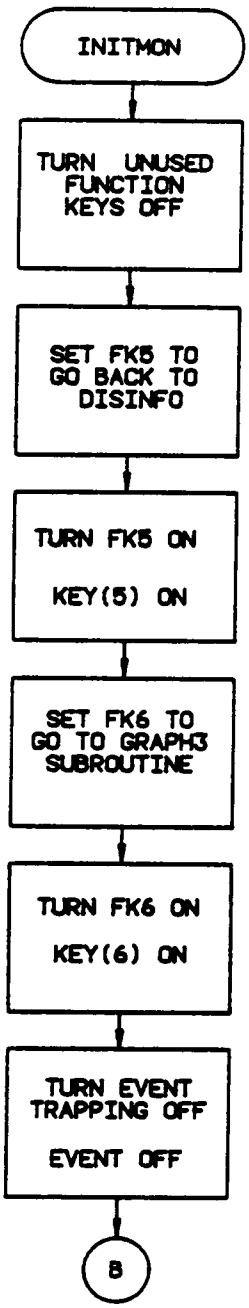

Figure 22a. Flowchart of Subroutine INITMON

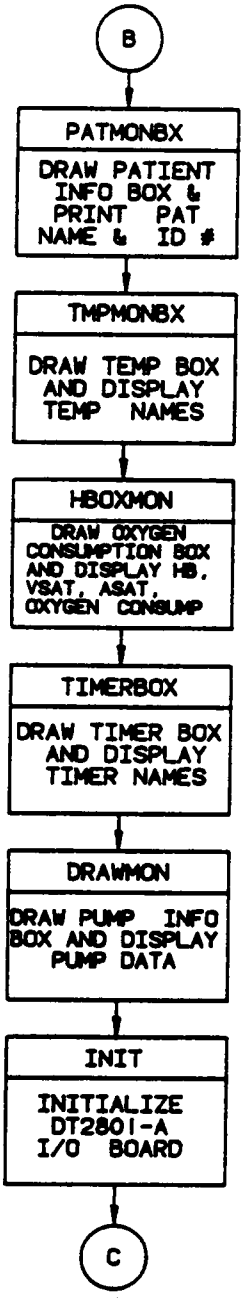

Figure 22b. Flowchart of Subroutine INITMON

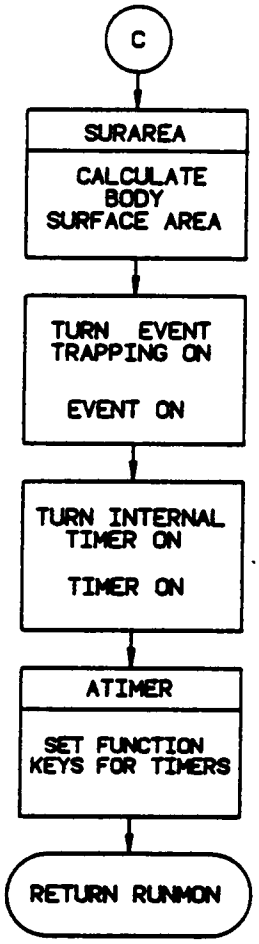

Figure 22c. Flowchart of Subroutine INITMON

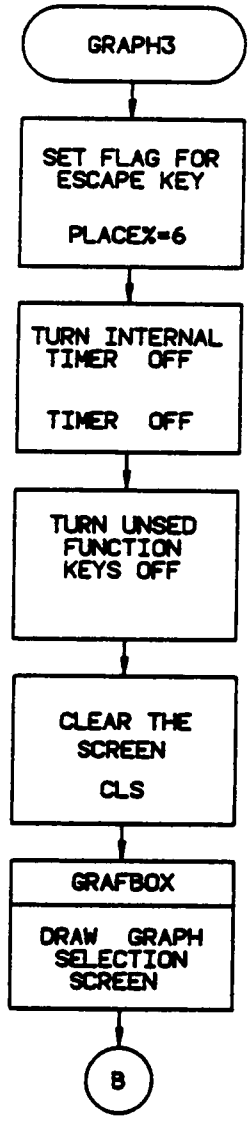

Figure 23a. Flowchart of Subroutine GRAPH3

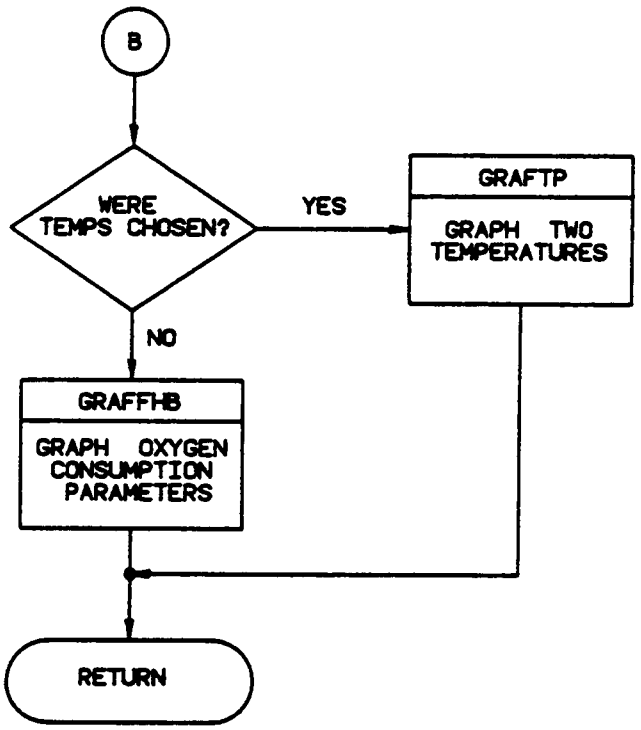

Figure 23b. Flowchart of Subroutine GRAPH3

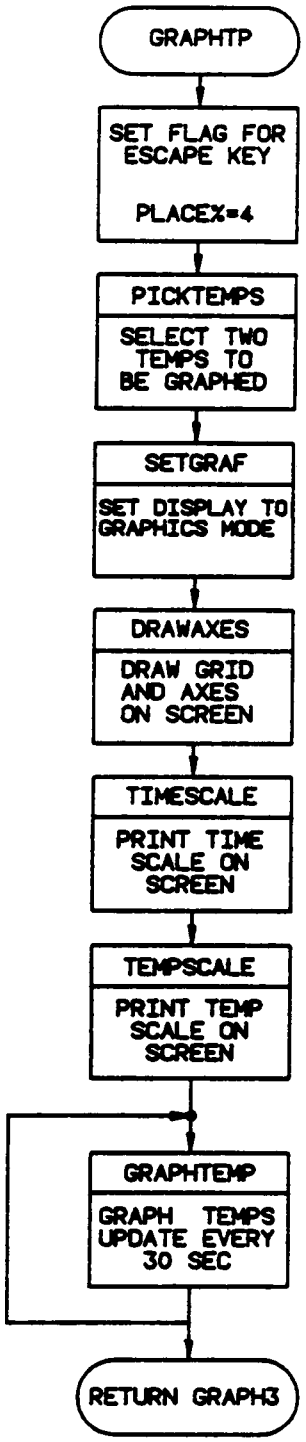

Figure 24. Flowchart of Subroutine GRAPHTP

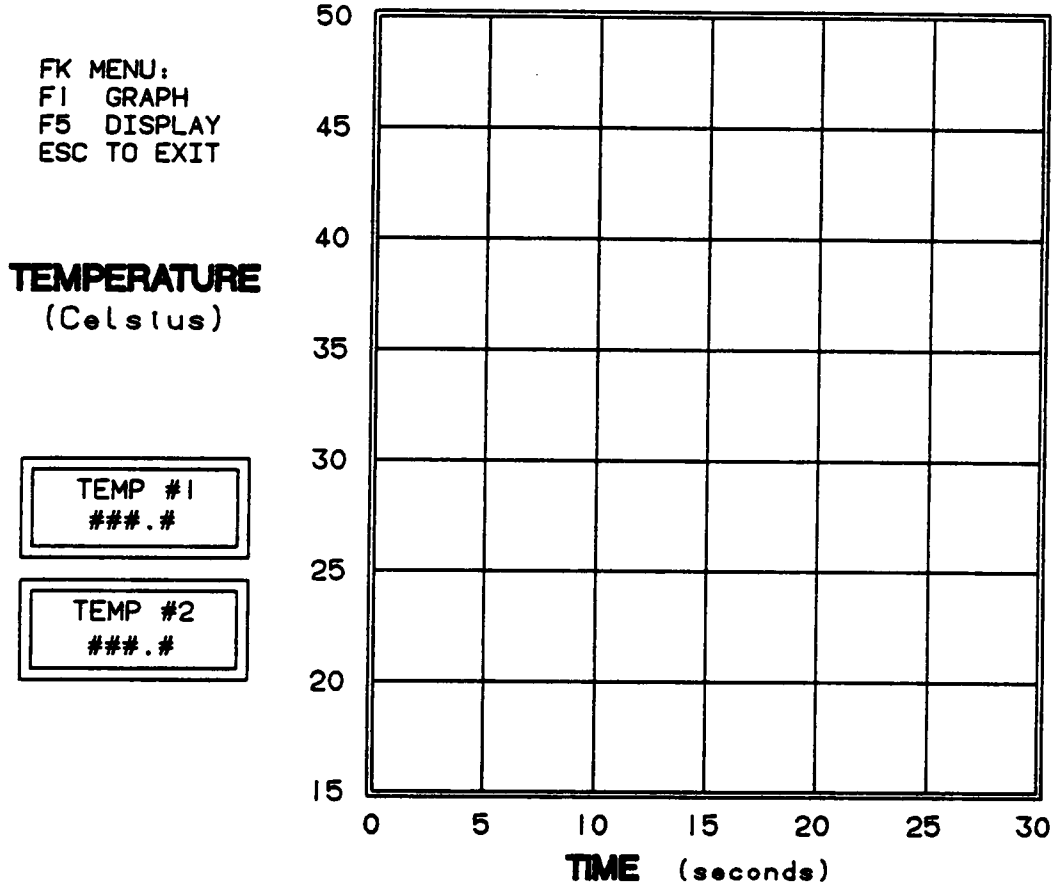

Figure 25. Drawing of Temperature vs. Time screen

return to the beginning of the graphing subroutine, function key 6 (FK6) is set to return to the beginning of the monitoring subroutine RUNMON, and the escape key can be pressed to exit the program.

If the second selection of oxygen consumption parameters is chosen, then control of the program goes to the subroutine GRAPHHB. A flowchart of this subroutine is shown in Figure 26. The GRAPHHB subroutine produces a graph as shown in Figure 27. It displays a plot of Hemoglobin content, arterial oxygen saturation, venous oxygen saturation, and oxygen consumption versus time. The graph is updated every thirty seconds. Also, the values for each parameter are shown at the bottom of the graph (see Figure 27) and a function key menu is displayed across the top of the graph. Function key 1 (FK1) is set to return to the beginning of the graphing subroutine, function key 6 (FK6) is set to return to the beginning of the monitoring subroutine RUNMON, and the escape key can be pressed to exit the program.

Next, in the INITMON subroutine, a function key menu is displayed across the bottom of the screen as shown in Figure 23. The function keys and the escape key are set to do the following:

- FK 1 start timer #1
- FK 2 start timer #2
- FK 3 start timer #3
- FK 5 return to display and correction screen
- FK 6 graph options
- FK 8 update timer #3
- FK 9 update timer #2
- FK 10 update timer #1

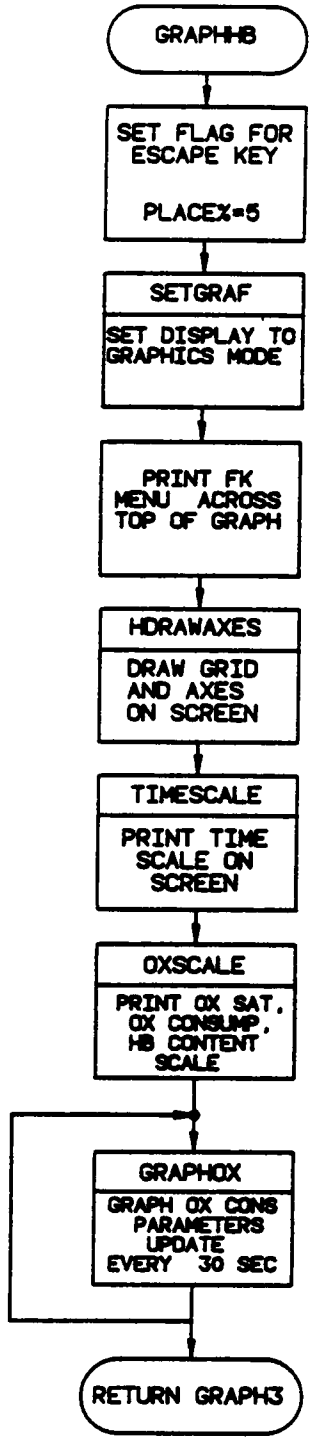

Figure 26. Flowchart of Subroutine GRAPHHB

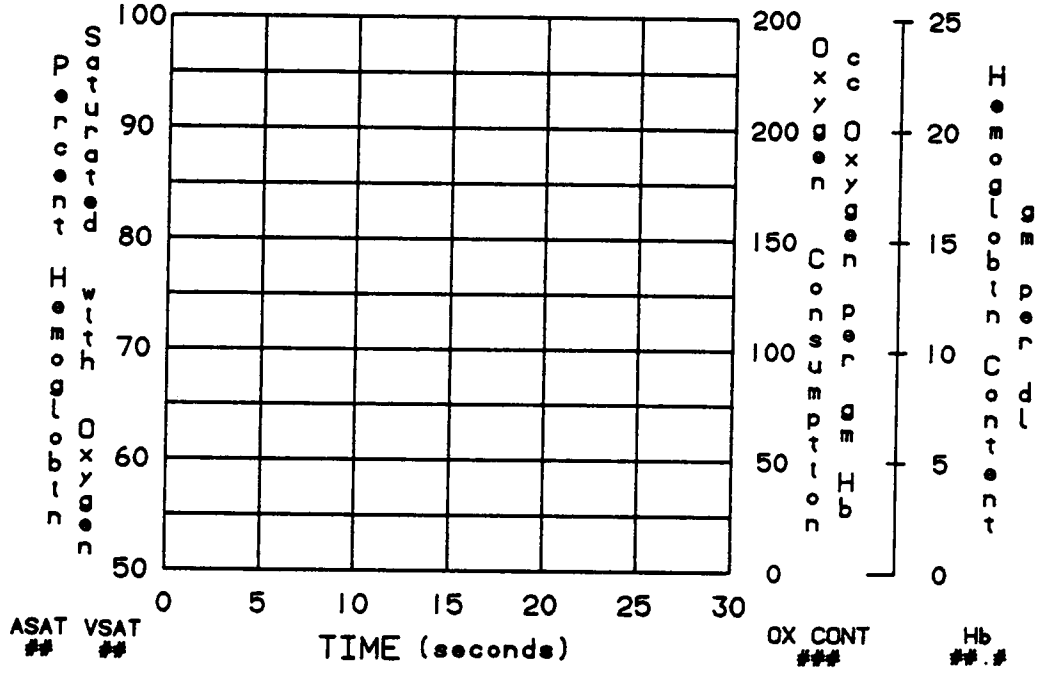

Figure 27. Drawing of Oxygen Consumption parameters vs. Time screen

- ESC exit program

Next in the INITMON subroutine, event trapping is turned off so that the monitoring screen can be displayed without interruption. To display the monitoring screen, seven subroutines are called. As shown in Figure 28, the subroutines display the titles and units for all of the parameters being monitored. The parameters displayed are:

- Patient Name
- Hospital Identification Number
- temperature #1
- temperature #2
- temperature #3
- temperature #4
- temperature #5
- temperature #6
- Hemoglobin Content
- Arterial Oxygen Saturation
- Venous Oxygen Saturation
- Oxygen Consumption
- Timer #1 start, update, and elapsed times

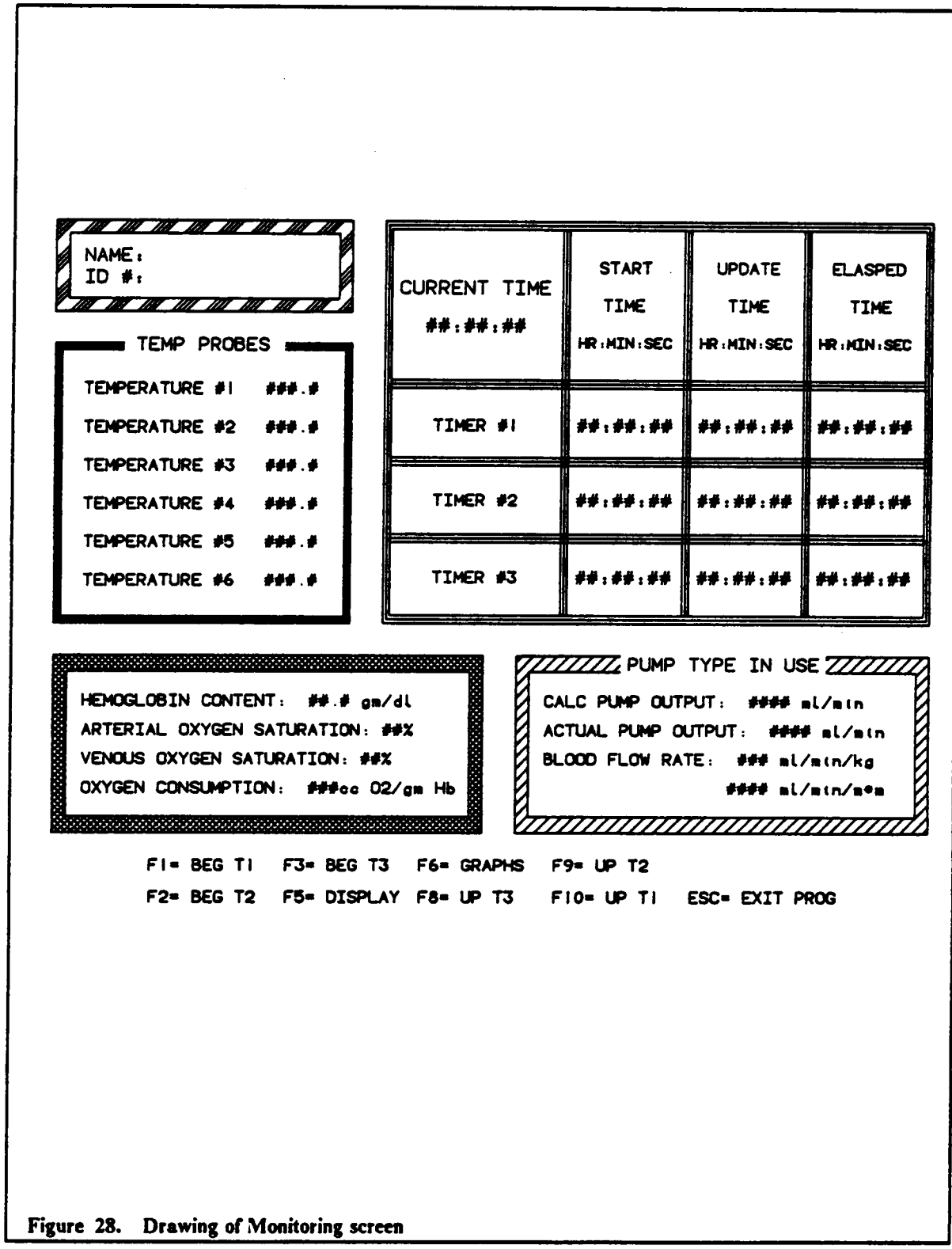

Figure 28. Drawing of Monitoring screen

- Timer #2 start, update, and elapsed times
- Timer #3 start, update, and elapsed times
- Type of blood pump in use
- Calculated Pump Output
- Actual Pump Output
- Blood flow rate with respect to weight
- Blood flow rate with respect to surface area

Next, the subroutine INIT is called to initialize the DT2801-A Digital and Analog I/O board for use. The patient body surface area is calculated next by calling subroutine SURFARE. A flowchart of SURFARE is shown in Figure 29. The body surface area is calculated using Dubois Formula as follows:

$$Surface\ Area = Weight^{0.445} \times Height^{0.727} \times 0.0007184$$

where the weight is in units of kilograms and the height is in units of centimeters. The resulting units of surface area are square meters [13].

Next, the INITMON subroutine turns event trapping on again. This causes Turbo Basic to check for events (such as pressing function keys or the escape key) between every statement. Then, the timer is turned on so that timed events can be monitored. Finally, the INITMON subroutine calls the subroutine ATIMER which sets the function keys for the timers as shown on the function key menu at the bottom of Figure 28. Then, control is returned to the RUNMON subroutine.

The monitoring subroutine, RUNMON, next sets the current time to be updated every 0.75 seconds. Then the loop that the subroutine stays in while monitoring is defined as RUNIT.

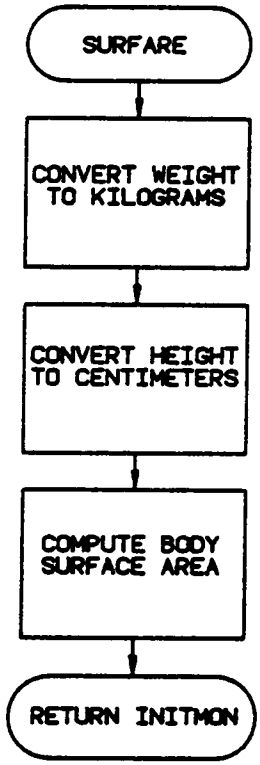

Figure 29. Flowchart of Subroutine SURFARE

The RUNIT loop consists of three subroutine calls. The first subroutine called is PUMCAL. This subroutine updates the calculated pump output and the normalized blood flow rates. The next subroutine called is HBOXSAT which updates the hemoglobin content, the arterial oxygen saturation, the venous oxygen saturation, the actual pump output and the oxygen consumption. Finally, the subroutine GETIT is called to update each temperature. Updates of these parameters occur approximately every 1.15 seconds.

As shown in Figure 30, the subroutine PUMCAL calls three other subroutines. First, the subroutine FREQ is called (see Figure 31). It is responsible for reading the rate of the blood pump. The subroutine FREQ sets digital port 0 for input. Then, the pulses into this port are counted for one second and the pump rate is calculated.

Next, the subroutine PUMPOUTPUT is called. A flowchart of this subroutine is shown in Figure 32. If a roller pump is being used, the units of the roller pump parameters are converted to metric units if necessary (inches to centimeters and pounds to kilograms). Next, the roller pump volume is calculated as follows:

$$Roller\ Pump\ Volume = (Number\ Rollers \times Arc\ Length \times \pi \times Tube\ Diameter^2)/4$$

Using this equation, the pump volume has units of milliliters. Next, the roller pump rate is calculated as follows:

$$Roller\ Pump\ Rate = (Freq\ Hz) \times (60\ sec/\ \text{min}) / (100\ pulses/rev)$$

The pick-up device for the roller pump produces 100 low pulses per revolution of the roller. Thus, the value of pump rate from the FREQ subroutine is divided by 100. The roller pump rate has units of cycles per minute.

If a pulsatile pump is being used, the pump volume is equal to the stroke volume in cubic centimeters. The pulsatile pump rate is calculated using the following equation:

$$Pulsatile\ Pump\ Rate = (Freq\ Hz) \times (60\ sec/\ \text{min}) / (1\ pulse/pulse\ of\ diaphragm)$$

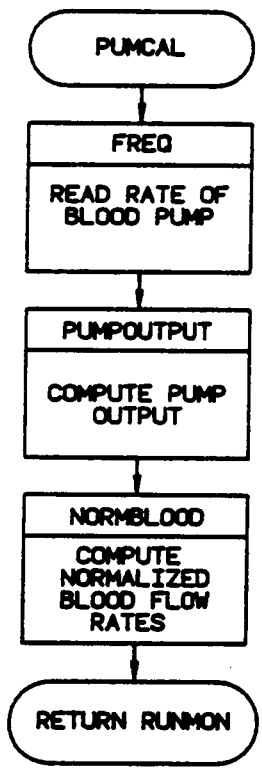

Figure 30. Flowchart of Subroutine PUMCAL

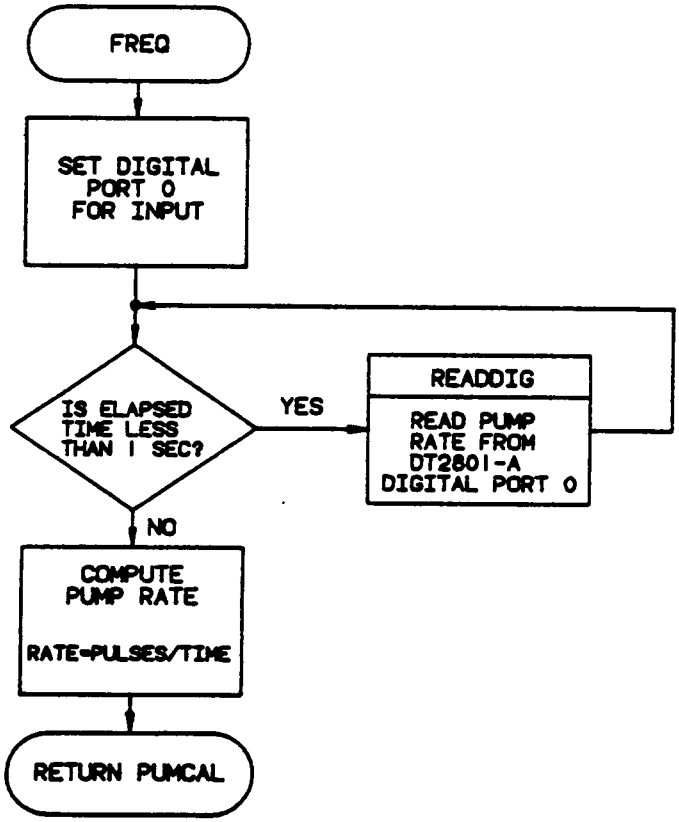

Figure 31. Flowchart of Subroutine FREQ

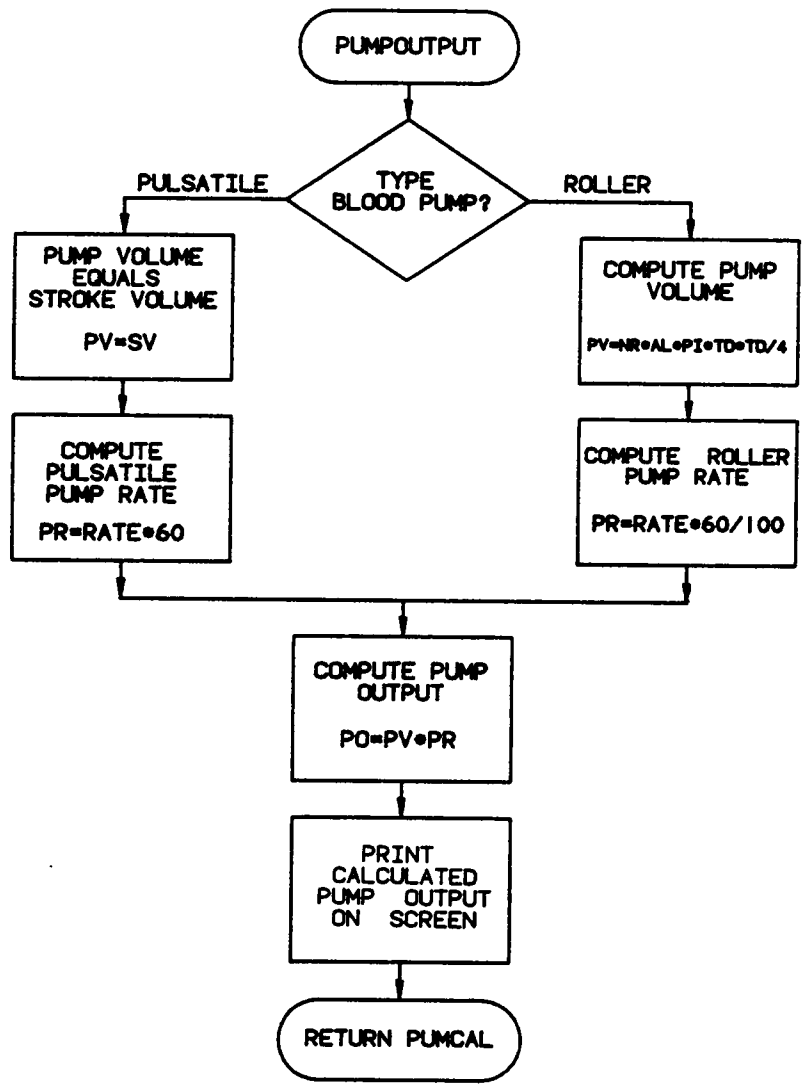

Figure 32. Flowchart of Subroutine PUMPOUTPUT

The pick-up device for the pulsatile pump produces one low pulse per pulse of the diaphragm. Therefore, the value of pump rate from the FREQ subroutine is divided by 1. The pulsatile pump rate has units of cycles per minute.

Regardless of the type of blood pump being used, the pump output is computed in the same way using the following formula:

$$Pump\ Output = Pump\ Volume \times Pump\ Rate$$

where pump volume is in milliliters and the pump rate is in cycles per minute. The resulting units of pump output are milliliters per minute. After this calculation, the calculated pump output is printed on the screen and control is sent back to the subroutine PUMCAL.

Next, PUMCAL calls the subroutine NORMBLOOD. A flowchart of this subroutine is shown in Figure 33. This subroutine computes the two normalized blood flow rates. The first formula used is:

$$Normalized\ Blood\ Flow\ Rate = Pump\ Output \,/\, Body\ Surface\ Area$$

where the body surface area is in square meters. The units of normalized blood flow rate resulting form this equation are ml/min/m^2. The second formula used to calculate normalized blood flow rate is:

$$Normalized\ Blood\ Flow\ Rate = Pump\ Output \,/\, Patient\ Weight$$

where the patient weight is in kilograms. The units of normalized blood flow rate resulting from this equation are ml/min/kg. Then, both of these blood flow rates are printed on the screen and control is passed back to PUMCAL and then, back to RUNMON.

Next, RUNMON calls the subroutine HBOXSAT. A flowchart of this subroutine is shown in Figure 34. HBOXSAT calls three other subroutines. First, the subroutine READO2 is called. A flowchart of subroutine READO2 is shown in Figure 35. This subroutine is used to multiplex analog voltages into analog channel 7 of the DT2801-A I/O board. As discussed in the previous chapter, the multiplexing is done using a 4016 CMOS quad bilateral switch. The switches are

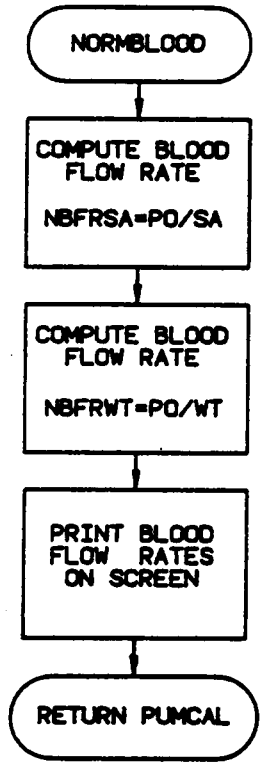

Figure 33. Flowchart of Subroutine NORMBLOOD

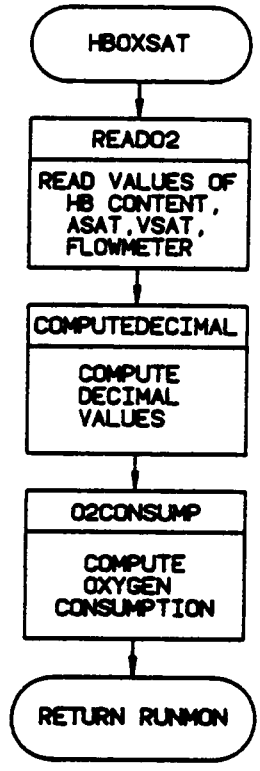

Figure 34. Flowchart of Subroutine HBOXSAT

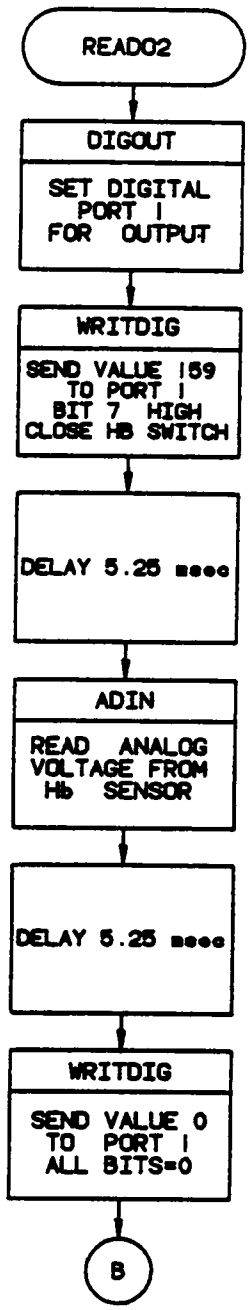

Figure 35a. Flowchart of Subroutine READO2

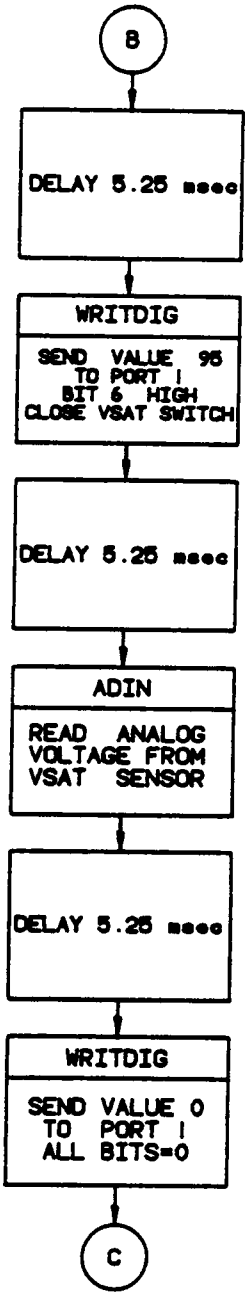

Figure 35b. Flowchart of Subroutine READO2

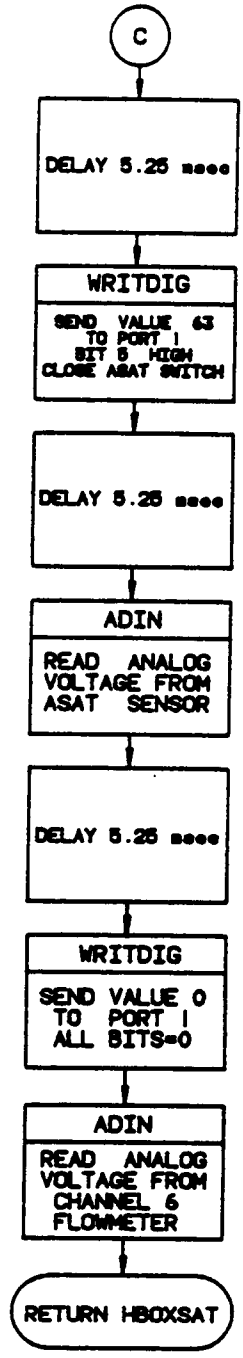

Figure 35c. Flowchart of Subroutine READO2

opened and closed using pulses produced by the software and output at digital port 1 of the DT2801-A I/O board.

As shown in the flowchart of READO2 (Figure 35), first, digital port 1 is set for output. Then, the pulses are started so that data can be read from analog channel 7 of the DT2801-A I/O board. The timing sequence is shown again in Figure 36. This sequence starts with bit 7 of digital port 1 of the DT2801-A I/O board being taken to logical 1 for 11.5 msec. This closes the switch so that the analog voltage representing the hemoglobin content can be read at analog channel 7 of the DT2801-A I/O board. At the same time bit 7 is high, bits 5 and 6 are held at logical 0 for 11.5 msec. Data is sampled form the hemoglobin content sensor 5.25 msec after bit 7 is first taken to logical 1. This ensures that the proper data is being sampled for the hemoglobin content. Next, all bits of digital port 1 are taken to logical 0 for 10.5 msec. Then, bit 6 of digital port 1 is taken to logical 1 for 11.5 msec while bits 5 and 7 are held at logical 0. Taking bit 6 to logical 1 closes the switch so that the analog voltage representing the venous oxygen saturation can be read at analog channel 7. Once again, the venous oxygen saturation data is sampled 5.25 msec after bit 6 is first taken to logical 1. Next, all bits of digital port 1 are taken to logical 0 for 10.5 msec. Then, bit 5 of digital port 1 is taken to logical 1 for 11.5 msec while bits 6 and 7 are held at logical 0. Taking bit 5 to logical 1 closes the switch for the arterial oxygen saturation sensor. Data from this sensor is sampled 5.25 msec after bit 5 is first taken to logical 1. Finally, all bits of digital port 1 are taken to logical 0. Sampling one value from each of the three sensors takes approximately 56 msec.

The last command in the READO2 subroutine is to read analog channel 6. This updates the actual pump output by sampling the analog voltage from the flowmeter. Then, control is returned to the HBOXSAT subroutine.

The next subroutine called from HBOXSAT is COMPUTEDECIMAL. The flowchart for COMPUTEDECIMAL is shown in Figure 37. This subroutine is called to convert the voltages read in READO2 from binary values to decimal values. Also, these voltages, representing hemoglobin content, venous oxygen saturation, arterial oxygen saturation, and the actual flow rate are multiplied by scaling factors of each sensor to get the proper value of each. For example, the

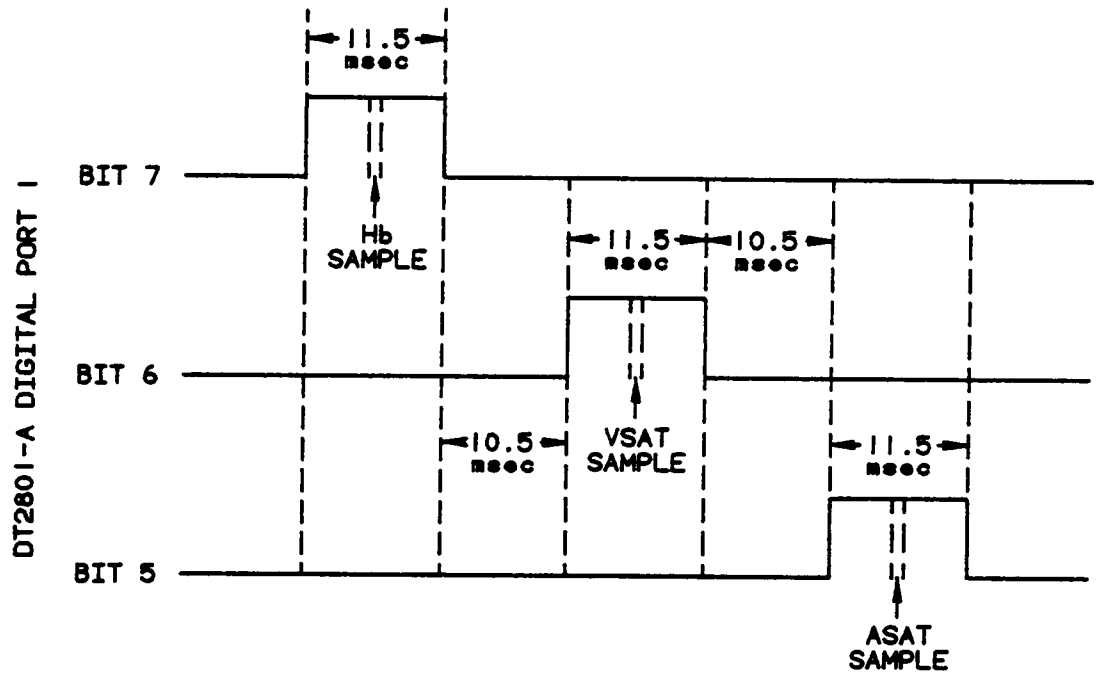

Figure 36. Timing Diagram of Oxygen Consumption Parameters

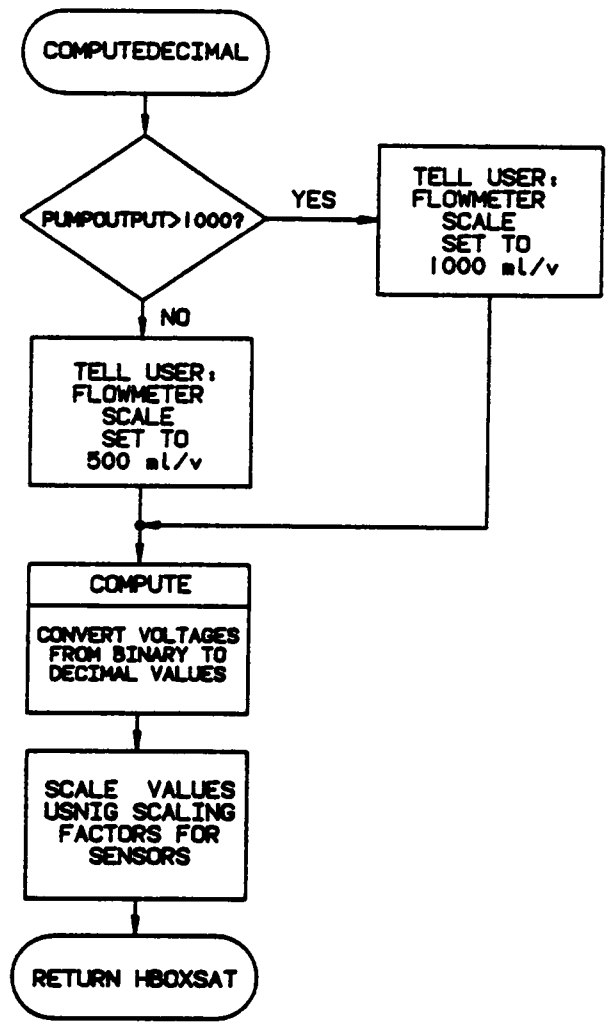

Figure 37. Flowchart of Subroutine COMPUTEDECIMAL

scaling factor for the hemoglobin content is 10 gm Hb/dl per volt. So that a reading of 2.00 volts read from the hemoglobin content sensor is equivalent to a hemoglobin content of 20 gm/dl. Also, the flowmeter scale setting is displayed on the screen for the user. The setting is to be set to 1000 ml/v if the calculated pump output is greater than 1000 ml/min. If the calculated pump output is less than or equal to 1000 ml/min, then the flowmeter scale setting is 500 ml/v. Then, control is returned to the HBOXSAT subroutine.

The last subroutine called in HBOXSAT is O2CONSUMP. A flowchart of this subroutine is shown in Figure 38. This subroutine simply computes the oxygen consumption using the updated values of the parameters as shown in the following equation:

$$Oxygen\ Consumption = (\ Flow\ Rate\ ) \times (\ Hb\ Content\ ) \times (\ ASAT - VSAT\ ) \times (\ 10\ dl/l\ ) \times (1.39)$$

where the flow rate is in liters per min, the hemoglobin content is in gm Hb/dl, and the arterial and venous oxygen saturation are percentage values. The constant 1.39 represents the cubic centimeters of Oxygen per gram of hemoglobin. The units of oxygen consumption resulting from this equation are cubic centimeters of oxygen per minute [14].

Finally, the values for actual pump output, hemoglobin content, arterial oxygen saturation, venous oxygen saturation, and oxygen consumption are displayed on the screen and control is returned to the HBOXSAT subroutine. Then, control is returned to RUNMON where the subroutine GETIT is called next.

A flowchart of the GETIT subroutine is shown in Figure 39. This subroutine updates all six temperatures except those that are marked with a not connected flag. This update is done by reading analog channels 0 through 5 of the DT2801-A I/O board and displaying the updated values of temperature on the screen. After all channels have been updated, control returns to the RUNMON subroutine where the loop RUNIT is executed until the user presses function key 5 (FK5) to return to the display and correction screen, function key 6 (FK6) to request a graph, or the escape key to exit the program completely [15].

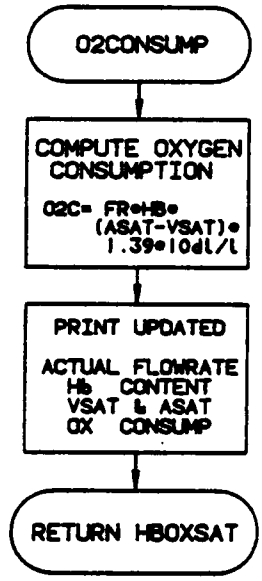

Figure 38. Flowchart of Subroutine O2CONSUMP

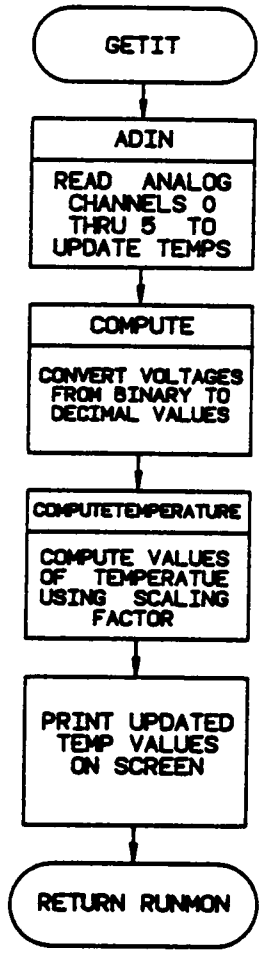

Figure 39. Flowchart of Subroutine GETIT

V. SYSTEM EVALUATION

Electrical System

The electrical system functioned as desired. It was evaluated using potentiometers to represent the analog voltages of the sensors connected to the seven analog inputs and a square wave generator to represent the pulses from the blood pump into digital port 0 of the DT2801-A I/O board. Also, the operation of the 4016 CMOS switch was checked using pulses to open and close each switch.

The operation of the seven analog channels and the A/D converter was tested using potentiometers. As shown in Table 1 and Figure 40, voltages between 0.000 and 2.000 volts were input to each channel. The voltage readings displayed by the computer agreed with those of the digital multimeter to within 2 millivolts.

To verify that the pump rate was being detected properly, a square wave generator was connected to digital port 0, bit 1 of the DT2801-A I/O board. Frequency values between 0 and approximately 500 Hz were testes. The square wave generator was also connected to an oscilloscope. The frequency detected by the DT2801-A I/O board and that calculated from the oscilloscope readings were compared (see Table 2 and Figure 41). The detected frequency

Table 1a. Verification of Analog Channels of the DT2801-A

Analog Channel Number	Displayed Voltage (volts)	Multimeter Reading* (volts)
0	0.000	0.000
0	0.177	0.178
0	0.934	0.933
0	1.031	1.033
0	1.970	1.971
1	0.010	0.010
1	0.200	0.200
1	0.952	0.951
1	1.196	1.198
1	1.504	1.505
2	0.000	0.000
2	0.255	0.254
2	0.924	0.924
2	1.261	1.261
2	1.798	1.800
3	0.000	0.000
3	0.151	0.150
3	0.737	0.736
3	0.972	0.972
3	1.978	1.976

Table 1b. Verification of Analog Channels of the DT2801-A

Analog Channel Number	Displayed Voltage (volts)	Multimeter Reading* (volts)
4	0.000	0.000
4	0.400	0.400
4	0.825	0.825
4	1.401	1.401
4	1.992	1.992
5	0.000	0.000
5	0.337	0.336
5	0.991	0.991
5	1.309	1.309
5	1.953	1.961
6	0.049	0.049
6	0.322	0.324
6	0.464	0.464
6	0.972	0.971
7	0.151	0.152
7	0.771	0.771
7	1.261	1.261
7	1.973	1.973

* As measured by a digital multimeter, Beckman model TECH 360, SN 270-233530, on 5-22-89

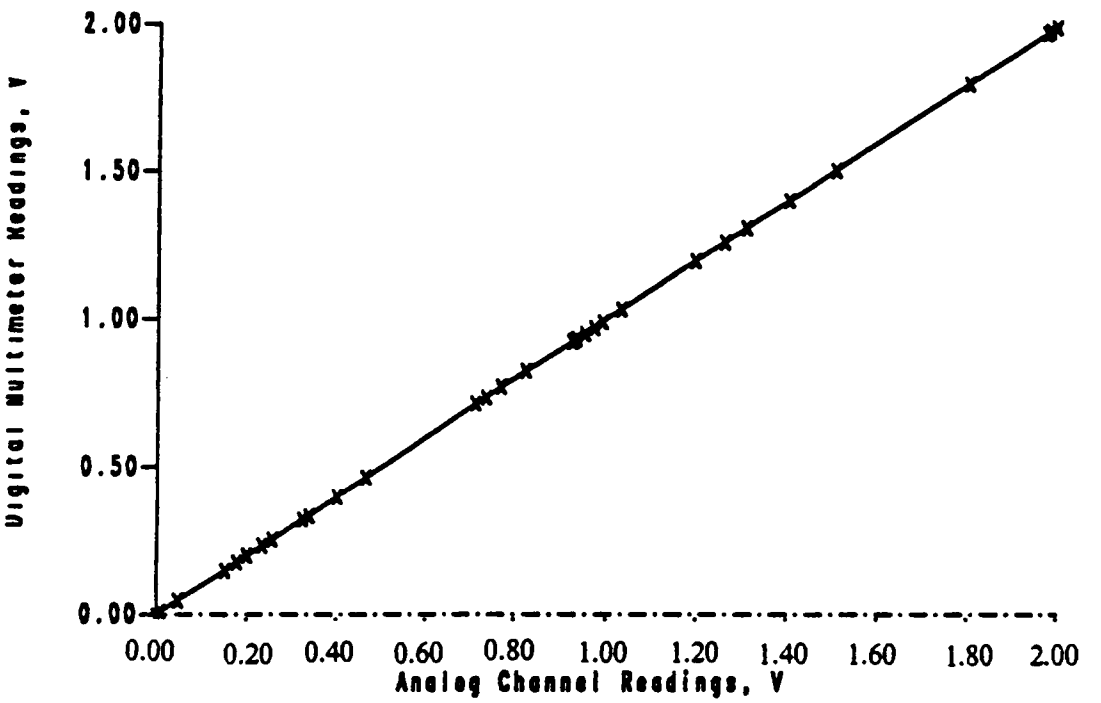

Figure 40. Verification of Analog Channels of DT2801-A

Table 2. Verification of Pulse Counter

Displayed Frequency (hertz)	Calculated Frequency (hertz)
0.00	0.00
20.1	20.0
74.9	75.8
90.1	92.6
125.5	128.2
175.6	178.6
225.4	229.9
274.6	277.8
326.0	333.3
375.0	384.6
425.2	434.8
476.0	487.8
524.4	537.6

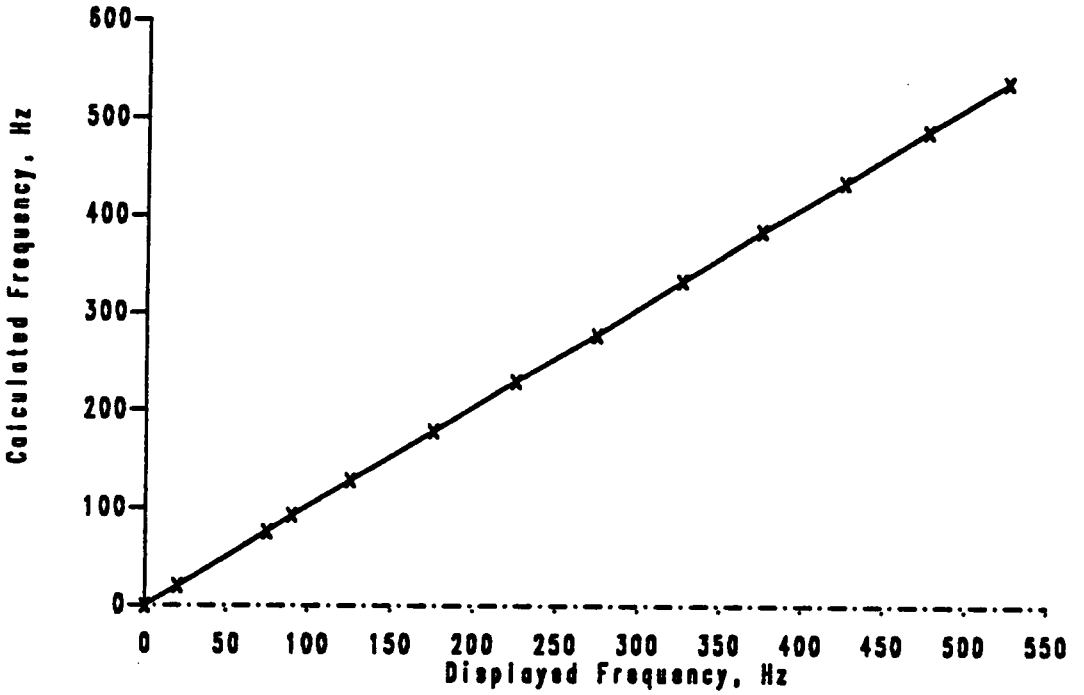

Figure 41. Verification of Pulse Counter

differentiated from the oscilloscope values an average of 7 Hz. The reason for this difference is because the oscilloscope is difficult to read accurately.

A 4016 CMOS switch chip was used to multiplex the hemoglobin content sensor, the arterial oxygen saturation sensor, and the venous oxygen saturation sensor into analog channel 7 of the DT2801-A I/O board. The operation of this chip was checked using the pulses produced by software to open and close each switch. The timing sequence was verified using an oscilloscope, and the 4016 CMOS switch was found to be operating properly.

Software System

The software performed as expected. All calculations and the timers were checked using a Hewlett-Packard 41CX calculator. The pulses produced by software to multiplex the values of hemoglobin content, arterial oxygen saturation, and venous oxygen saturation into the system were checked using an oscilloscope.

The calculation of oxygen consumption was checked using the values shown in Table 3. The values obtained using the calculator matched, to the significant figures displayed, those displayed by the computer.

Tables 4 and 5 show the values used to check the calculated pump output. Values of calculations displayed for the roller pump and pulsatile pump agreed to the significant figures displayed with those obtained using the calculator.

The patient body surface area and the normalized blood flow rate calculations were verified using the data shown in Tables 6 and 7. Once again, the displayed values and those obtained using the calculator agreed, to the significant figures displayed.

The timers were checked intermittently over a 27 hour period using the stopwatch function of the calculator. They were found to be operating perfectly. The display of current time was also found to be functioning correctly when synchronized with the timer function of the calculator.

Table 3. Verification of Oxygen Consumption Calculation

Hb Content (gm/dl)	ASAT-VSAT (%)	Flow Rate (l/min)	displayed O_2 Consumption (cc O_2/min)	calculated O_2 Consumption (cc O_2/min)
1.465	25.39	7.617	39.38	39.38
8.691	23.93	3.174	91.74	91.74
10.06	37.11	8.398	435.7	435.7
13.77	34.67	6.944	460.1	460.1
14.55	21.00	3.760	159.6	159.6
15.14	26.37	7.813	433.4	433.4
16.85	21.97	6.836	351.7	351.7
18.75	22.46	6.348	371.6	371.6
19.77	38.09	8.154	853.7	853.7

Table 4. Roller Pump Output

Frequency (hertz)	Number of Rollers	Tube Diameter	Arc Length	displayed Pump Output (ml/min)	calculated Pump Output (ml/min)
25	2	0.5 in.	3 in.	289	288
125	2	0.5 in.	3 in.	1446	1446
225	2	0.5 in.	3 in.	2611	2611
375	2	0.5 in.	3 in.	4343	4344
75	2	1 cm	6 cm	421	421
126	2	1 cm	6 cm	711	711
276	2	1 cm	6 cm	1558	1559
425	2	1 cm	6 cm	2403	2403

Table 5. Pulsatile Pump Output

Frequency (hertz)	Stroke Volume (cc)	Displayed Pump Output (ml/min)	Calculated Pump Output (ml/min)
20	0.1	121	121
20	0.2	242	241
90	0.1	540	541
90	0.2	1082	1081
224	0.1	1346	1346
225	0.2	2705	2705
324	0.1	1946	1946
325	0.2	3901	3901
425	0.1	2551	2551
424	0.2	5088	5088

Table 6. Verfication of Body Surface Area Calculation

Height	Weight	Displayed Surface Area (m^2)	Calculated Surface Area (m^2)
24 in.	10 lb	0.269	0.269
48 in.	50 lb	0.881	0.881
60 in.	100 lb	1.390	1.390
72 in.	200 lb	2.130	2.130
90 in.	300 lb	2.975	2.975
100 in.	350 lb	3.429	3.429
70 cm	10 kg	0.416	0.416
90 cm	20 kg	0.670	0.670
100 cm	30 kg	0.859	0.859
120 cm	50 kg	1.219	1.219
130 cm	60 kg	1.395	1.395
140 cm	70 kg	1.572	1.572
160 cm	90 kg	1.927	1.927
170 cm	100 kg	2.106	2.106
190 cm	140 kg	2.634	2.634

Table 7a. Verfication of Blood Flow Rate Calculations

Height	Weight	Displayed Flow Rate ($ml/min/m^2$)	Calculated Flow Rate ($ml/min/m^2$)
24 in.	10 lb	7400.0	7400.0
48 in.	50 lb	2259.0	2259.0
60 in.	100 lb	1431.0	1431.0
72 in.	200 lb	934.1	934.1
90 in.	300 lb	668.8	668.8
100 in.	350 lb	580.3	580.3
70 cm	10 kg	4784.0	4784.0
90 cm	20 kg	2970.0	2970.0
100 cm	30 kg	2316.0	2316.0
120 cm	50 kg	1633.0	1633.0
130 cm	60 kg	1426.0	1426.0
140 cm	70 kg	1266.0	1266.0
160 cm	90 kg	1033.0	1033.0
170 cm	100 kg	945.0	945.0
190 cm	140 kg	755.6	755.6

Table 7b. Verfication of Blood Flow Rate Calculations

Height	Weight	Displayed Flow Rate (ml/min/kg)	Calculated Flow Rate (ml/min/kg)
24 in.	10 lb	438.7	438.7
48 in.	50 lb	87.7	87.7
60 in.	100 lb	43.9	43.9
72 in.	200 lb	21.9	21.9
90 in.	300 lb	14.6	14.6
100 in.	350 lb	12.5	12.5
70 cm	10 kg	199.0	199.0
90 cm	20 kg	99.5	99.5
100 cm	30 kg	66.3	66.3
120 cm	50 kg	39.8	39.8
130 cm	60 kg	33.2	33.2
140 cm	70 kg	28.4	28.4
160 cm	90 kg	22.1	22.1
170 cm	100 kg	19.9	19.9
190 cm	140 kg	14.2	14.2

** All blood flow rates were calculated using a pump output of 1990 ml/min.

Finally, the timing pulses produced by software that are used to multiplex the hemoglobin content, the arterial oxygen saturation, and the venous oxygen saturation into analog channel 7 of the DT2801-A I/O board were verified using an oscilloscope.

VI. CONCLUSIONS

The monitoring system for cardiopulmonary bypass surgery described has been evaluated and was found to function properly. As designed, it allows the user to input the necessary patient and surgical information. Then, six temperatures, hemoglobin content, arterial oxygen saturation, venous oxygen saturation, blood flow rate, blood pump output, and oxygen consumption are monitored. These parameters are initially displayed on the computer screen in tabular form. However, the six temperatures, hemoglobin content, arterial oxygen saturation, venous oxygen saturation, and the oxygen consumption are also available in graphical form.

VII. RECOMMENDATIONS

Prior to any clinical applications of this system, shielded cable should be used to connect all sensors to the interface circuit board. Additionally, rigorous testing should continue to ensure the proper operation of this monitoring system in a surgical setting.

Future possibilities for this monitoring system include a feed back loop from the flowmeter to the blood pump so that the proper blood flow rate is achieved. A larger computer display would enable more detailed graphs and displays of additional parameters. Also, if disk storage was available in surgery, a subroutine could be written to store the monitored parameters for future reference. This would allow hard copies of the patient information, the surgical information, and the monitored parameters.

REFERENCES

1. Reed, C. C. and D. K. Clark, *Cardiopulmonary Perfusion*, Texas Medical Press, Inc., Houston, TX, 1975, pp. 272-278.
2. Kirklin, J. W. and B. G. Barratt-Boyes, *Cardiac Surgery*, John Wiley & Sons, New York, 1986, pp. 44-46.
3. Berger, E. C., Letter to L. J. Arp, April 28, 1983.
4. Brinkman, K. L., "Design of a Microcomputer-Based Open Heart Surgery Patient Monitor," Master Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1985.
5. SWAN SETUP Utility, SWAN Technologies, LTD., State College, PA, 1988.
6. Mosher, F. E. and D. I. Schneider, *Using Turbo Basic*, Osborne McGraw-Hill, Berkeley, CA, 1988, p. 206.
7. *SWAN 286-10 Users Manual*, SWAN Technologies, LTD, State College, PA, 1988, p. 51.
8. Byers, T J., *Inside the IBM PC AT* McGraw-Hill Book Company, New York, 1985, pp. 5-8, 19-26.
9. *SWAN 286-10 Users Manual*, pp. 48-52.
10. *Personal Computer Series for Science and Engineering: User Manual for DT2801 Series Single Board Analog and Digital Input/Output Systems for IBM Personal Computer*, Data Translation, Inc., Marlbaro, MA, 1987.
11. Gates, S. C., "Analog to Digital Converters in the Laboratory," *Scientific Computing and Automation,* February, 1989, pp. 49-56.
12. Mosher, pp. 1-6.
13. Reed, pp. 322-331.
14. *Turbo Basic Owner's Handbook*, Borland International, Inc., Scotts Valley, CA, 1987.

Appendix A. HARDWARE INFORMATION

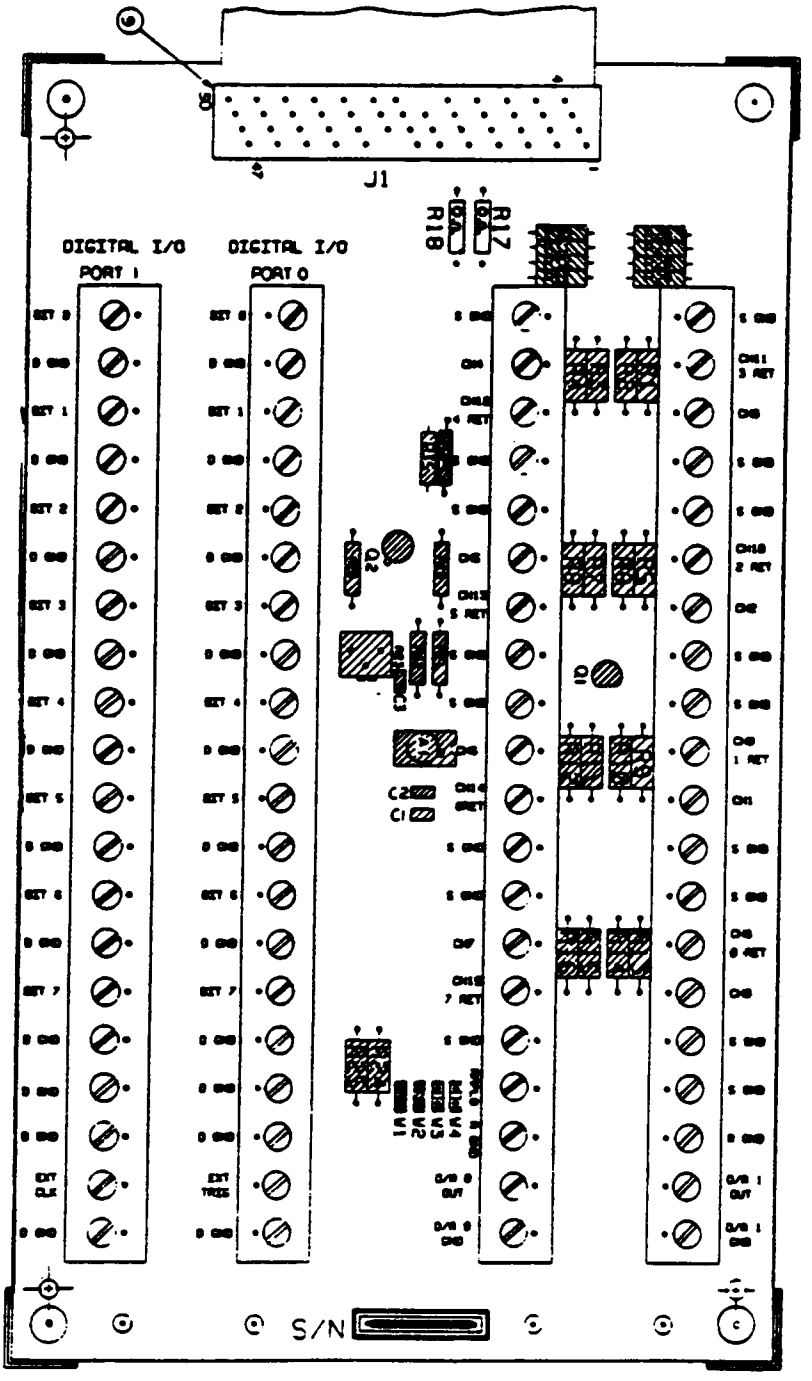

Figure A1. Drawing of the DT707 Screw Board Panel

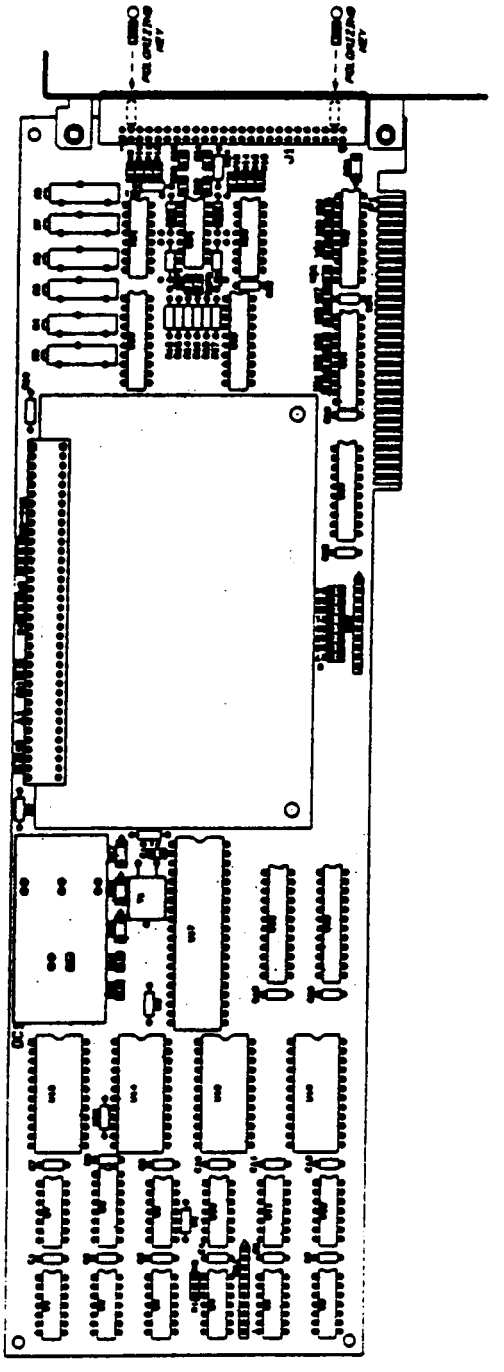

Figure A2. Drawing of the DT2801-A Input/Output Board

Appendix B. SOFTWARE LISTING

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: MONITOR.BAS                ==
'==               backup prog : MONITOR.BAK                ==
'==                                                        ==
'============================================================
' PROGRAM WRITTEN:  4-03-89                                 '
'                                                           '
' PROGRAM MODIFIED: 5-24-89                                 '
'============================================================
'==                                                        ==
'============================================================
'==  PROGRAM NAME:  MONITOR.BAS                            ==
'==                                                        ==
'============================================================
'==                                                        ==
'============================================================
'==                                                        ==
'==      $INCLUDE PROGRAMS: INFOIN.BAS                     ==
'==                         DISINFO.BAS                    ==
'==                         RUNMON.BAS                     ==
'==                                                        ==
'============================================================
'-------------------------------------------------------------
'-------------------------------------------------------------
' The STACK is a special area in memory that holds the
'values or addresses of variables being passed to
'procedures and functions, and contains an indication of
'where the program should continue executing after a
'procedure, function, or subroutine concludes.
'A run-time OUT OF MEMORY (ERROR 7) error is returned
'when the stack lacks sufficient space to store new
'information.

'increase stack size to 6144 bytes (6k)
$stack &H1800
'-------------------------------------------------------------
'This statement resets all numeric variables and
'arrays to 0, resets all string variables and arrays
'to null, and closes all files.
CLEAR
'-------------------------------------------------------------
'call subroutine to draw title screen
'gosub titlescreen
'-------------------------------------------------------------
'call subroutine to initialize screen and variables
'for monitoring program
gosub intro
'-------------------------------------------------------------
'go to subroutine to get initial information input
GOSUB INFOIN

'  define palette jar 3 as light blue
   palette 3,3

'  define palette jar 0 as black
   palette 0,0

'  define color of text as black and background as black
   color 0,0

'  clear the screen
cls

'go to subroutine to display information input
GOSUB DISINFO

'label for program to return to main
main1:

'go to subroutine to monitor parameters
GOSUB RUNMON

'  define color as light blue text and black background
   color 3,0

'  define end of program
end
'============================================================
```

```
'include these programs to save space
'   TURBO BASIC editor
$include "c:\monitor1\title.bas"
$include "c:\monitor1\intro.bas"
$include "c:\monitor1\infoin.bas"
$include "c:\monitor1\disinfo.bas"
$include "c:\monitor1\RUNMON.bas"
'============================================================
```

```
'==========================================================
'==              ***** IMPORTANT INFO *****               ==
'==                                                       ==
'==             program name: INTRO.BAS                   ==
'==             backup prog : INTRO.BAK                   ==
'==                                                       ==
'==========================================================
' PROGRAM WRITTEN:  5-17-89                                '
'                                                          '
' PROGRAM MODIFIED: 5-17-89                                '
'==========================================================
'==                                                       ==
'==             SUBROUTINE NAME: INTRO                    ==
'==                                                       ==
'==    THIS SUBROUTINE SETS SCREEN AND INITIALIZES        ==
'==    VARIBLES FOR MONITORING PROGRAM.                   ==
'==                                                       ==
'==========================================================
'define subroutine intro
intro:
'
'set screen to graphics mode with 640x350 color resolution
screen 12
'
'set screen to text mode, color enabled
screen 0,1
'
'call subroutine to set escape key for exit from prog
goto setescape
'
'turn run-time error trapping off
on error goto 34
'
'define end of subroutine intro
return
'==========================================================
'==========================================================
'==             SUBROUTINE NAME:  setescape               ==
'==                                                       ==
'==     This subroutine sets the escape key to exit       ==
'==     program when the escape key is hit.               ==
'==                                                       ==
'==========================================================
'define setescape as a subroutine
SETESCAPE:

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'set up trapping of escape key for exit of program
'
'set key 15 as the escape key
KEY 15,CHR$(&0)+CHR$(&6a)
'
'when the escape key (key 15) is pressed, goto
'subroutine ENDALL
ON KEY(15) GOSUB ENDALL
'
'turn trapping of escape key on
KEY(15) ON
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'define end of subroutine SETESCAPE
RETURN
'==========================================================

'==========================================================
'==             SUBROUTINE NAME:  ENDALL                  ==
'==                                                       ==
'==    This subroutine exits the entire monitoring        ==
'==    program when the escape key is hit.                ==
'==                                                       ==
'==========================================================
'define ENDALL as a subroutine
ENDALL:

'stop function of escape key
key (15) stop

'turn event trapping off
$event off

'turn timer off
timer off
```

```
'set screen to text mode, color enabled
screen 0,1

'set palette color 0 to white (63)
palette 0,55

'set palette color 4 to red (4)
palette 4,4

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=0
BKCOL%=4

'define beginning and ending rows of menu box
begrow%=14
endrow%=33

'define beginning and ending columns of menu box
begcol%=1
endcol%=45

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'set text color to red and background to black
color 0,1

'print text on row 21 and column 69
locate 21,69

'print text on screen
PRINT "ARE YOU SURE YOU WANT TO EXIT"

'locate text on row 12 and column 5
locate 11,50

'input answer
INPUT ANSWER$

'label for invalid answer to check second answer
checkanswer:

'if answer is yes, then end program
IF (ANSWER$="y") OR (ANSWER$="Y") THEN
   color 1,0
   cls
   end
end if

'turn event trapping on
$event on

'if answer is no, then check flag place% to
'return execution to proper place in program
IF (ANSWER$="n") OR (ANSWER$="N") THEN

   'case variable is place%
   select case place%

   'if place%=1, then came from infoin subroutine
   'return to next statement
   case 1
   'set palette color 0 to black (0)
   palette 0,0

   'set palette color 5 to magenta (5)
   palette 5,5

   gosub setescape
   gosub infoin

   'if place%=2, then came from disinfo subroutine
   'goto disinfo subroutine
   case 2
   gosub setescape
   gosub disinfo

   'if place%=3, then came from runmon subroutine
   'goto runmon subroutine
   case 3
   gosub setescape
   gosub runmon
```

```
    'if place%=4, then came from graphtp subroutine
    'goto graphtp subroutine
    case 4
    gosub setescape
    goto escapetemp

    'if place%=5, then came from graphhb subroutine
    'goto graphhb subroutine
    case 5
    gosub setescape
    gosub graphhb

    'if place%=6, then came from graphtp subroutine
    'goto graphtp subroutine
    case 6
    gosub setescape
    gosub graph3

    'define end of select structure
    end select

'define end of if structure
end if

'answer is not y or n, goto invalid answer
locate 13,35
color 4,4
print space$(7)
locate  13,28
call invalidanswer(answer$)
goto checkanswer

'define end of subroutine endall
RETURN
'============================================================
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: INFOIN.BAS                 ==
'==               backup prog : INFOIN.BAK                 ==
'==                                                        ==
'============================================================
' PROGRAM WRITTEN:  2-26-89                                 '
'                                                           '
' PROGRAM MODIFIED: 5-17-89                                 '
'============================================================
'==                                                        ==
'==   This program calls the $INCLUDE programs:            ==
'==                getpatin.bas                            ==
'==                tempname.bas                            ==
'==                timename.bas                            ==
'==                pumpinfo.bas                            ==
'==                                                        ==
'============================================================

'============================================================
'==  PROGRAM NAME:  INFOIN.BAS                             ==
'==                                                        ==
'==    This program contains the subroutines to display    ==
'==  patient information, temperature names, timer names,  ==
'==  and pump information on the screen.                   ==
'==                                                        ==
'==    This program displays patient information on the    ==
'==  screen.  It displays the following:                   ==
'==                                                        ==
'==                1.  Name of Patient                     ==
'==                2.  Sex of Patient                      ==
'==                3.  ID # of Patient                     ==
'==                4.  Age of Patient                      ==
'==                5.  Units of Age                        ==
'==                6.  Height of Patient                   ==
'==                7.  Units of Height                     ==
'==                8.  Weight of Patient                   ==
'==                9.  Units of Weight                     ==
'==                                                        ==
'==    Additionally, the patient's body surface area is    ==
'==  computed.                                             ==
'==                                                        ==
'==    This program prompts user for title of temperature  ==
'==  probes.  Then, the titles are displayed in the        ==
'==  following manner:                                     ==
'==                                                        ==
'==             1.  NAME OF TEMPERATURE #1                 ==
'==             2.  NAME OF TEMPERATURE #2                 ==
'==             3.  NAME OF TEMPERATURE #3                 ==
'==             4.  NAME OF TEMPERATURE #4                 ==
'==             5.  NAME OF TEMPERATURE #5                 ==
'==             6.  NAME OF TEMPERATURE #6                 ==
'==                                                        ==
'==    This program prompts user for title of timers       ==
'==  Then, the titles are displayed in the following       ==
'==  manner:                                               ==
'==                                                        ==
'==             1.  NAME OF TIMER #1                       ==
'==             2.  NAME OF TIMER #2                       ==
'==             3.  NAME OF TIMER #3                       ==
'==                                                        ==
'==                                                        ==
'==    This program prompts user for the type of blood     ==
'==  pump being used in the surgical prodecure.            ==
'==                                                        ==
'============================================================
' define INFOIN as a subroutine
INFOIN:

'set flag place%=1 for return from escape key
place%=1

'  Clear the Screen
   CLS

'  Declare value of prompt$ - character string
'    Shared variable among input PROCEDURES
   prompt$="Please enter "

'  Call subroutine to have user input patient
'   information
   GOSUB GETPATINFO
```

```
'  Clear screen
   CLS

'  Call subroutine to have user input temperature
'   names
   GOSUB GETTEMPNAMES

'  Clear the Screen
   CLS

'  Call subroutine to have user input timer
'   names
   GOSUB GETTIMERNAMES

'  Clear the Screen
   CLS

'  Call subroutine to have user input pump
'   information
   GOSUB PUMPINPUT

'  Clear the Screen
   CLS

'  End of subroutine infoin
   RETURN
'============================================================

'  Programs stored under different names to save
'   space in the TURBOBASIC editor
  $INCLUDE "c:\monitor1\GetPatin.BAS"
  $INCLUDE "c:\monitor1\TempName.BAS"
  $INCLUDE "c:\monitor1\TimeName.BAS"
  $INCLUDE "c:\monitor1\PumpInfo.BAS"



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: GETPATIN.BAS               ==
'==               backup prog : GETPATIN.BAK               ==
'==                                                        ==
'============================================================
' PROGRAM WRITTED:  2-26-89
'
' PROGRAM MODIFIED: 4-21-89
'============================================================
'==                                                        ==
'============================================================
'==  PROGRAM NAME:  GETPATIN.BAS                           ==
'==                                                        ==
'==    This program contains the subroutines to get        ==
'==  patient information.                                  ==
'==                                                        ==
'==                1.  Name of Patient                     ==
'==                2.  Sex of Patient                      ==
'==                3.  ID # of Patient                     ==
'==                4.  Age of Patient                      ==
'==                5.  Units of Age                        ==
'==                6.  Height of Patient                   ==
'==                7.  Units of Height                     ==
'==                8.  Weight of Patient                   ==
'==                9.  Units of Weight                     ==
'==                                                        ==
'============================================================
'  Define getpatinfo as a subroutine
getpatinfo:

'  Clear the Screen
   CLS

'  Dimension array info$ for later use in prog
       DIM INFO$(10)

'  Call procedure to get name of patient
   CALL GetName (patname$)

'  Call procedure to get sex of patient
   CALL GetSex (sex$)

'  Call procedure to get ID number of patient
```

```
    CALL GetIDnum (IDNUM$)

'   Call procedure to get age of patient
    CALL GetAge (age)

'   Call procedure to get unit of age
    CALL GetAgeunit (ageunit$)

'   Call procedure to get height of patient
    CALL GetHeight (height)

'   Call procedure to get units of height
    CALL GetHtunit (htunit$)

'   Call procedure to get weight of patient
    CALL GetWeight (weight)

'   Call procedure to get units of weight
    CALL GetWtunit (wtunit$)

'   Clear screen
    CLS

'   Define row index for placement of text
       row%=3

'   Define column index to center text
       col%=55

'   Call subroutine to display patient information
    GOSUB ShowPatINFO

'   Call subroutine to check patient information
    GOSUB CheckPatInfo

'   Call subroutine to file patient information
    GOSUB FilePat

'   End of subroutine getpatinfo
    RETURN
'================================================

' programs stored under different names to
' save space in turbo basic editor
 $INCLUDE "c:\monitor1\4PATINFO.BAS"
 $INCLUDE "c:\monitor1\SHOWPAT.BAS"
 $INCLUDE "c:\monitor1\FILEPAT.BAS"
 $INCLUDE "c:\monitor1\2CORREC.BAS"
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  4PATINFO.BAS               ==
'==              backup prog :  4PATINFO.BAK               ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   2-10-89                             '
'                                                           '
'    PROGRAM MODIFIED:  4-20-89                             '
'============================================================
'==  PROGRAM NAME:  4PATINFO.INC                           ==
'==                                                        ==
'==    This program displays patient information on the    ==
'==  screen.  It displays the following:                   ==
'==                                                        ==
'==                1.  Name of Patient                     ==
'==                2.  Sex of Patient                      ==
'==                3.  ID # of Patient                     ==
'==                4.  Age of Patient                      ==
'==                5.  Units of Age                        ==
'==                6.  Height of Patient                   ==
'==                7.  Units of Height                     ==
'==                8.  Weight of Patient                   ==
'==                9.  Units of Weight                     ==
'==                                                        ==
'============================================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                             ppp
'ppp             PROCEDURE NAME:  GetName                        ppp
```

```
'ppp                                                                  ppp
'ppp          PROCEDURE TO GET NAME OF PATIENT                        ppp
'ppp                                                                  ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define GetName as a PROCEDURE with patname$ as a
'   passed variable
   SUB GetName (patname$)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
   SHARED prompt$

'  Specify color of writing as blue (2) and background as
'   palette color 1
     COLOR 2,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=2
bkcol%=4

'define beginning and ending rows of box
begrow%=5
endrow%=55

'define beginning and ending columns of box
begcol%=22
endcol%=56

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
9     COLOR 3,3
      LOCATE 8,27
      PRINT SPACE$(30)
      COLOR 1,3
      LOCATE 8,27
           PRINT " ";
      LOCATE 12,27
           COLOR 1,1
           PRINT SPACE$(52)
           LOCATE 14,17
           COLOR 1,1
           PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,22

'  Prompt user to input name of patient
   color 1,3
        PRINT prompt$;"the Name of the patient."
   LOCATE 8,25
   INPUT patname$

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 12,27
        COLOR 1,1
        PRINT SPACE$(52)
   LOCATE 12,28
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'If the answer is No, then prompt user to enter patient
'   name again
20 IF (ans$="N") OR (ans$="n") THEN 9
   LOCATE 12,53
        COLOR 1,1
        PRINT SPACE$(24)


'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 25

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 14,18
   COLOR 1,1
```

```
    PRINT SPACE$(61)

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    LOCATE 14,17
         COLOR 1,1
         PRINT SPACE$(62)
    LOCATE 14,18
         CALL InvalidAnswer(ans$)
         GOTO 20

'CLEAR SCREEN
27    CLS

'   Define end of PROCEDURE GetName
    END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp                PROCEDURE NAME:  GetSex                            ppp
'ppp                                                                   ppp
'ppp          PROCEDURE TO GET SEX OF PATIENT                          ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'   Define GetSex as a PROCEDURE with sex$ as a
'     passed variable
    SUB GetSex (sex$)

'   Define prompt$ as a shared variable between the PROCEDURE
'     and main prog
    SHARED prompt$

' set color
    PALETTE 0,62
         COLOR 0,0


'   Specify color of writing as light cyan (3) and background
'     as black (0)
    COLOR 1,0

'   Clear the screen
    CLS

'   Prompt user to select sex of patient
    LOCATE 5,7
    PRINT prompt$;"sex of the patient (M or F)";
    PRINT " by using the arrow keys to"
    PRINT "       move the selection box to the proper sex";
    PRINT "  and press the enter key."

'   Specify color of writing as blue (1) and background as
'    palette color 1
    COLOR 0,0,0

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=1
bkcol%=3

'define beginning and ending rows of box
begrow%=8
endrow%=11

'define beginning and ending columns of box
begcol%=31
endcol%=45

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'   Print type of available SEX types
6600     COLOR 15,3
    LOCATE 9,34
    PRINT " MALE "
    LOCATE 10,34
    PRINT " FEMALE "


'    define MALE$ as INFO$(1) and add a space
'     at the beginning and end
    INFO$(1)=SPACE$(1)+"MALE"+SPACE$(1)

'    define FEMALE$ as INFO$(2) and add a space
'     at the beginning and end
```

```
   INFO$(2)=SPACE$(1)+"FEMALE"+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
   LLINE%=1
   COLOR 0,0
   LOCATE 13,10
   PRINT SPACE$(69)
   COLOR 0,0
   LOCATE 15,10
   PRINT SPACE$(69)
   COLOR 0,0
   LOCATE 17,10
   PRINT SPACE$(69)

'  change color to white (15) text and red (12)
'   background;  locate the text at row 9 and
'   column 33;  print INFO$(1)
   COLOR 15,5
        LOCATE 9,34
        PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                     XX
'XX       This selection routine was originally written  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.     XX
'XX                                                                     XX
'XX       Modifications and all documentation           XX
'XX     was done by Cynthia K. Rice on 3-1-89.           XX
'XX                                                                     XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelectSex
'   so it can be called again
   SelectSex:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
        KKEY$=INKEY$

'###################################################################
'###################################################################
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectPump (continue selection process)
        CASE ""
          GOTO SelectSex
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
        CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to red (6) text and black (0)
      '  background;  print text on row LLINE%+8 and
      '   column 33
         COLOR 1,3
         LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

      ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
      '  then decrement LLINE% by 1
      ' ELSE increment LLINE% by 1 (meaning the up arrow
      '  was selected)
        IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%
```

```
        ' need to keep LLINE% in allowable range for data
        '  so the selection box will scroll from last
        '  allowable selection to first allowable selection
        '  and vice versa
        ' If select up arrow when on top line will go to
        '  bottom line
      IF LLINE%<1 THEN LLINE%=2

        ' If select down arrow when on bottom line will go to
        '  top line
      IF LLINE%>2 THEN LLINE%=1

        ' change the color to white (15) text and red (12)
        '  background;  print next text at row (LLINE%+8)
        '    and column 8
          COLOR 15,5
          LOCATE LLINE%+8,34

        ' print INFO$(LLINE%)
          PRINT INFO$(LLINE%)

        ' goto label SelectSex to continue selection process
      GOTO SelectSex:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
        CASE CHR$(13)

      '  goto subroutine CorrectSex to correct sex info
          GOTO CorrectSex:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label SelectSex
'  to continue selection process
        CASE ELSE
        GOTO SelectSex
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
        END SELECT
'##############################################################################
'##############################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectSex
      RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                        ppp
'ppp              SUBROUTINE NAME: CorrectSex                               ppp
'ppp                                                                        ppp
'ppp             SUBROUTINE TO SELECT SEX TYPE                              ppp
'ppp                                                                        ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTSEX as a subroutine
CORRECTSEX:

'  Change color to white (15) text and red (12) background
   COLOR 15,12

'  Locate the text in row (lline%+8) and column 8
      LOCATE  lline%+8,3

'##############################################################################
'##############################################################################
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 1, then sex$="MALE"
   CASE 1
        SEX$="MALE"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 2, then SEX$="FEMALE"
   CASE 2
        SEX$="FEMALE"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% > 2, then continue program
         CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
         END SELECT
'################################################################
'################################################################

'  Change color to RED (12) text and WHITE (15) background
    COLOR 1,15

'  Tell user the SEX type selected
       LOCATE 13,2
       PRINT " You selected ";SEX$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
    COLOR 0,0
    LOCATE 15,16
    PRINT SPACE$(62)
    LOCATE 15,25
    CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter SEX type
'     again.
6620     IF (ans$="N") OR (ans$="n") THEN 6600
    COLOR 0,0
    LOCATE 15,51
    PRINT SPACE$(26)

'  If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 6625

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    COLOR 0,0
    LOCATE 17,16
    PRINT SPACE$(62)
    LOCATE 17,16
    CALL InvalidAnswer (ans$)
    GOTO 6620

625 PALETTE 0,0

' CLEAR SCREEN
    CLS

'  Define end of PROCEDURE GetSex
    END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                              ppp
'ppp              PROCEDURE NAME:  GetIDnum                       ppp
'ppp                                                              ppp
'ppp          PROCEDURE TO GET ID# OF PATIENT                     ppp
'ppp                                                              ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  Define GetIDnum as a PROCEDURE with IDNUM$ as a
'     passed variable
    SUB GetIDnum (IDNUM$)

'  Define prompt$ as a shared variable between the PROCEDURE
'     and main prog
    SHARED prompt$

'  Specify color of writing as blue (1) and background as
'   palette color 1
    COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=1
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=10
```

```
'define beginning and ending columns of box
begcol%=18
endcol%=63

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

1257    COLOR 3,3
     LOCATE 8,30
     PRINT SPACE$(30)
     COLOR 1,3
     LOCATE 8,30
     PRINT " ";
     LOCATE 12,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 14,17
          COLOR 1,1
          PRINT SPACE$(62)
     COLOR 1,3

'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input id number of patient
        PRINT prompt$;"the ID number of the patient."
   LOCATE 8,28
        INPUT IDNUM$

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 12,19
   COLOR 1,1
   PRINT SPACE$(60)
   LOCATE 14,63
   PRINT SPACE$(16)
   LOCATE 16,19
   PRINT SPACE$(60)
        COLOR 0,2

    '  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 12,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 12,28
        CALL CorrectAnswer (ans$)
   COLOR 1,3

    '  Check answer to PROCEDURE CorrectAnswer :

    '  If the answer is No, then prompt user to enter id
    '    number of patient again
1560        IF (ans$="N") OR (ans$="n") THEN 1250
     LOCATE 12,53
     COLOR 1,1
     PRINT SPACE$(26)

    '  If the answer is Yes, then return to main program
       IF (ans$="y") OR (ans$="Y") THEN 1265

' erase any previous invalid answers by printing
' spaces same color as background
   COLOR 1,1
   LOCATE 14,19
   PRINT SPACE$(60)
   LOCATE 16,17
   PRINT SPACE$(62)

    '  If answer was invalid, then call PROCEDURE
    '    InvalidAnswer to prompt user to input a
    '    proper answer of yes or no.
   LOCATE 14,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 14,19
        CALL InvalidAnswer(ans$)
        GOTO 1260

'CLEAR SCREEN
1265    CLS
```

```
'  Define end of PROCEDURE GetIDnum
   END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp                PROCEDURE NAME:  GetAge                          ppp
'ppp                                                                 ppp
'ppp          PROCEDURE TO GET AGE OF PATIENT                        ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  Define GetAge as a PROCEDURE with age as a
'   passed variable
   SUB GetAge(age)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
   SHARED prompt$

PALETTE 2,62
PALETTE 14,0

'  Specify color of writing as LT. BLUE (3) and background as
'   palette color 3
   COLOR 3,3

'set text color (txcol%) and gackground color (bkcol%)
txcol%=15
bkcol%=1

'define beginning and ending rows of box
begrow%=5
endrow%=10

'define beginning and ending columns of box
begcol%=14
endcol%=68

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

1333       COLOR 1,1
     LOCATE 8,37
     PRINT SPACE$(30)
     COLOR 15,1
     LOCATE 8,37
          PRINT " ";
     LOCATE 12,17
          COLOR 3,3
          PRINT SPACE$(52)
     LOCATE 14,27
          COLOR 3,3
     PRINT SPACE$(52)
          LOCATE 16,17
          COLOR 3,3
          PRINT SPACE$(62)
     COLOR 15,1

'LOCATE TEXT
1535  LOCATE 6,16

'  Prompt user to input sex of patient
   COLOR 15,1

'  Prompt user to input age of patient
        PRINT prompt$;"the age of patient in months or years."
   LOCATE 8,35
        INPUT age

'  If age <=0, then prompt user to input age again
        IF age =<0 THEN
     SOUND 75,10

     COLOR 31,1
          LOCATE 12,21
     PRINT " This is not a valid answer for the age. ";
     LOCATE 8,37
     COLOR 1,1
     PRINT SPACE$(30)
     COLOR 15,1
     LOCATE 8,37
     PRINT " ";
```

```
    END IF

'   Call PROCEDURE CorrectAnswer to ask if input is correct
        COLOR 3,3
        LOCATE 12,20
        PRINT SPACE$(55)
    LOCATE 14,20
        PRINT SPACE$(59)
        LOCATE 14,28
    CALL CorrectAnswer (ans$)
    COLOR 1,3

' erase any previous invalid answers by printing
' spaces same color as background
1340    COLOR 3,3
    LOCATE 16,63
    PRINT SPACE$(16)

'   Check answer to PROCEDURE CorrectAnswer :

'   If the answer is No, then prompt user to enter patient
'     age again
5342  IF (ans$="N") OR (ans$="n") THEN 1333
      LOCATE 14,53
          COLOR 3,3
          PRINT SPACE$(24)

'   If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 1345

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
        LOCATE 16,17
        COLOR 3,3
        PRINT SPACE$(62)
    LOCATE 16,18
        CALL InvalidAnswer(ans$)
        GOTO 1340

'CLEAR SCREEN
1345    CLS

'   Define end of PROCEDURE GetAge
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp              PROCEDURE NAME:  GetAgeunit                          ppp
'ppp                                                                   ppp
'ppp            PROCEDURE TO GET UNITS OF AGE                          ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'   Define GetAgeunit as a PROCEDURE with ageunit$ as a
'     passed variable
    SUB GetAgeunit (ageunit$)

'   Define prompt$ as a shared variable between the PROCEDURE
'     and main prog
    SHARED prompt$

PALETTE 2,62
PALETTE 14,0

'   Specify color of writing as LT. BLUE (3) and background as
'    palette color 3
    COLOR 3,3

'   Specify color of writing as light cyan (3) and background
'     as black (0)
    COLOR 1,2

'   Clear the screen
    CLS

'   Prompt user to select units of age
      LOCATE 5,7
      PRINT prompt$;"UNITS OF AGE";
      PRINT " by using the arrow keys to move the"
      PRINT "       selection box to the proper units";
      PRINT "  and press the enter key."

'   Specify color of writing as blue (1) and background as
'    palette color 1
'   COLOR 0,0,0
```

```
'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=1
bkcol%=3

'define beginning and ending rows of box
begrow%=8
endrow%=11

'define beginning and ending columns of box
begcol%=21
endcol%=45

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Print type of available units of age
2200    COLOR 1,3
   LOCATE 9,34
   PRINT " MONTHS "
   LOCATE 10,34
   PRINT " YEARS "
   COLOR 2,2
   LOCATE 13,10
   PRINT SPACE$(69)
   COLOR 2,2
   LOCATE 15,10
   PRINT SPACE$(69)
   COLOR 2,2
   LOCATE 17,10
   PRINT SPACE$(69)

'   define MONTHS$ as INFO$(1) and add a space
'    at the beginning and end
   INFO$(1)=SPACE$(1)+"MONTHS"+SPACE$(1)

'   define YEARS$ as INFO$(2) and add a space
'    at the beginning and end
   INFO$(2)=SPACE$(1)+"YEARS"+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
   LLINE%=1

'  change color to BLUE (1) text and CYAN (3)
'   background;  locate the text at row 9 and
'   column 34;  print INFO$(1)
     COLOR 1,4
       LOCATE 9,34
       PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                       XX
'XX       This selection routine was originally written                  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                      XX
'XX                                                                       XX
'XX       Modifications and all documentation                            XX
'XX     was done by Cynthia K. Rice on 3-1-89.                            XX
'XX                                                                       XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelectAgeU
'   so it can be called again
   SelectAgeU:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
       KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectAgeU (continue selection process)
       CASE ""
```

```
        GOTO SelectAgeU
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
       CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to BLUE (1) text and YELLOW (2)
      '  background;  print text on row LLINE%+8 and
      '  column 33
         COLOR 1,3
         LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
      '  then decrement LLINE% by 1
      ' ELSE increment LLINE% by 1 (meaning the up arrow
      '  was selected)
         IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

      ' need to keep LLINE% in allowable range for data
      '  so the selection box will scroll from last
      '  allowable selection to first allowable selection
      '  and vice versa
      ' If select up arrow when on top line will go to
      '  bottom line
    IF LLINE%<1 THEN LLINE%=2

      ' If select down arrow when on bottom line will go to
      '  top line
    IF LLINE%>2 THEN LLINE%=1

      ' change the color to BLUE (1) text and CYAN (3)
      '  background;  print next text at row (LLINE%+8)
      '   and column 8
        COLOR 1,2
        LOCATE LLINE%+8,37

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

     ' goto label SelectAgeU to continue selection process
    GOTO SelectAgeU:
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
       CASE CHR$(13)

     '  goto subroutine CorrectAgeU to correct sex info
        GOTO CorrectAgeU:

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label SelectAgeU
'  to continue selection process
       CASE ELSE
       GOTO SelectAgeU
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
       END SELECT
'#####################################################################
'#####################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectAgeU
       RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                               ppp
'ppp             SUBROUTINE NAME: CorrectAgeU                      ppp
```

```
'ppp                                                                    ppp
'ppp              SUBROUTINE TO SELECT AGE UNIT                         ppp
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTAGEU as a subroutine
CORRECTAGEU:

'  Change color to blue (1) text and cyan (3) background
   COLOR 1,3

'  Locate the text in row (lline%+8) and column 8
    LOCATE  lline%+8,34

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'    is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then AGEUNIT$="MONTHS"
   CASE 1
        AGEUNIT$="MONTHS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then AGEUNIT$="YEARS"
   CASE 2
        AGEUNIT$="YEARS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 2, then continue program
       CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
      END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

'  Change color to blue (1) text and WHITE (15) background
   COLOR 1,15

'  Tell user the units selected
   LOCATE 13,28
   PRINT " You selected ";AGEUNIT$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   COLOR 2,2
   LOCATE 15,16
   PRINT SPACE$(62)
   LOCATE 15,25
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter age
'  units again.
2220    IF (ans$="N") OR (ans$="n") THEN 2200
   COLOR 2,2
   LOCATE 15,51
   PRINT SPACE$(26)


'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 2225

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 17,16
   PRINT SPACE$(62)
   LOCATE 17,16
   CALL InvalidAnswer (ans$)
   GOTO 2220

2225 PALETTE 0,0

'CLEAR SCREEN
   CLS
```

```
'  Define end of PROCEDURE GetAgeUnit
   END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              PROCEDURE NAME:  GetHeight                         ppp
'ppp                                                                 ppp
'ppp         PROCEDURE TO GET HEIGHT OF PATIENT                      ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define GetHeight as a PROCEDURE with height as a
'   passed variable
   SUB GetHeight (height)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
   SHARED prompt$

PALETTE 2,62
PALETTE 14,0

'  Specify color of writing as LT. BLUE (3) and background as
'   palette color 3
   COLOR 2,2

'set text color (txcol%) and gackground color (bkcol%)
txcol%=15
bkcol%=1

'define beginning and ending rows of box
begrow%=5
endrow%=10

'define beginning and ending columns of box
begcol%=14
endcol%=44

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

30    COLOR 1,1
      LOCATE 8,37
      PRINT SPACE$(30)
      COLOR 15,1
      LOCATE 8,37
      PRINT " ";
      LOCATE 12,17
      COLOR 2,2
      PRINT SPACE$(52)
      LOCATE 14,17
      COLOR 2,2
           PRINT SPACE$(52)
      COLOR 15,1

'LOCATE TEXT
32 LOCATE 6,13
   COLOR 2,1

'  Prompt user to input height of patient
        PRINT prompt$;"Height of patient in inches or centimeters."
   LOCATE 8,35
        INPUT height

' erase any previous invalid answers by printing
' spaces same color as background
   COLOR 2,2
   LOCATE 12,20
   PRINT SPACE$(50)

'  If height <=0, then prompt user to input height again
        IF height =<0 THEN
      SOUND 75,10
      COLOR 28,2
      LOCATE 12,21
           PRINT "This is NOT A VALID answer for the HEIGHT."
      LOCATE 8,37
      COLOR 1,1
      PRINT SPACE$(30)
      COLOR 15,1
         LOCATE 8,44
      PRINT " ";
```

```
          GOTO 32
    END IF

'   Call PROCEDURE CorrectAnswer to ask if input is correct
        LOCATE 12,28
    CALL CorrectAnswer (ans$)
    COLOR 1,3

' erase any previous invalid answers by printing
' spaces same color as background
40  COLOR 2,2
    LOCATE 14,63
    PRINT SPACE$(16)

'   Check answer to PROCEDURE CorrectAnswer :

'   If the answer is No, then prompt user to enter patient
'     height again
    IF (ans$="N") OR (ans$="n") THEN 30

'   If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 45

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    COLOR 2,2
    LOCATE 12,54
    PRINT SPACE$(15)
    COLOR 15,1
    LOCATE 14,18
        CALL InvalidAnswer(ans$)
        GOTO 40

'CLEAR SCREEN
45      CLS

'   Define end of PROCEDURE GetHeight
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              PROCEDURE NAME:  GetHtunit                         ppp
'ppp                                                                 ppp
'ppp          PROCEDURE TO GET UNITS OF HEIGHT                       ppp
'ppp                                                                 ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'   Define GetHtunit as a PROCEDURE with htunit$ as a
'     passed variable
    SUB GetHtunit (htunit$)

'   Define prompt$ as a shared variable between the PROCEDURE
'     and main prog
    SHARED prompt$

palette 2,63

'   Specify color of writing as  BLUE (1) and background as
'    palette color 2
    COLOR 1,2

'   Clear the screen
    CLS

'   Prompt user to select units
      LOCATE 5,7
      PRINT prompt$;"UNITS OF HEIGHT";
      PRINT " by using the arrow keys to move "
      PRINT "          the selection box to the proper units";
      PRINT "  and press the enter key."

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=4
endrow%=44

'define beginning and ending columns of box
begcol%=31
endcol%=49
```

```
'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Print type of available units
22500    COLOR 1,3
    LOCATE 9,34
    PRINT " INCHES "
    LOCATE 10,34
    PRINT " CENTIMETERS "
    COLOR 2,2
    LOCATE 13,10
    PRINT SPACE$(69)
    COLOR 2,2
    LOCATE 15,10
    PRINT SPACE$(69)
    COLOR 2,2
    LOCATE 17,10
    PRINT SPACE$(69)


'   define INCHES$ as INFO$(1) and add a space
'    at the beginning and end
    INFO$(1)=SPACE$(1)+"INCHES"+SPACE$(1)

'   define CENTIMETERS$ as INFO$(2) and add a space
'    at the beginning and end
    INFO$(2)=SPACE$(1)+"CENTIMETERS"+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
    LLINE%=1

'  change color to BLUE (1) text and CYAN (3)
'   background;  locate the text at row 9 and
'   column 34;  print INFO$(1)
    COLOR 1,2
    LOCATE 9,34
    PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                       XX
'XX       This selection routine was originally written                  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                      XX
'XX                                                                       XX
'XX       Modifications and all documentation                            XX
'XX     was done by Cynthia K. Rice on 3-1-89.                            XX
'XX                                                                       XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a dabel SelectHtUnit
'   so it ca~ be called again
    S%lectHtUnit:

' assigns to the variable K EY$ the charac4gr that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
        KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
    SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectHtUnit (continue selection process)
        CASE ""
         GOTO SelectHtUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow
```

```
'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
       CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to BLUE (1) text and YELLOW (2)
      '  background;  print text on row LLINE%+8 and
      '  column 33
         COLOR 1,3
         LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
      '  then decrement LLINE% by 1
      ' ELSE increment LLINE% by 1 (meaning the up arrow
      '  was selected)
         IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

      ' need to keep LLINE% in allowable range for data
      '  so the selection box will scroll from last
      '  allowable selection to first allowable selection
      '  and vice versa
      ' If select up arrow when on top line will go to
      '  bottom line
    IF LLINE%<1 THEN LLINE%=2

      ' If select down arrow when on bottom line will go to
      '  top line
    IF LLINE%>2 THEN LLINE%=1

      ' change the color to BLUE (1) text and CYAN (3)
      '  background;  print next text at row (LLINE%+8)
      '    and column 8
        COLOR 1,2
        LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

    ' goto label SelectHtUnit to continue selection process
    GOTO SelectHtUnit:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
       CASE CHR$(13)

     '  goto subroutine CorrectHtUnit to correct sex info
        GOTO CorrectHtUnit:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label SelectHtUnit
'  to continue selection process
       CASE ELSE
       GOTO SelectHtUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
       END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectHtUnit
      RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                          ppp
'ppp              SUBROUTINE NAME: CorrectHtUnit              ppp
'ppp                                                          ppp
'ppp            SUBROUTINE TO SELECT HEIGHT UNIT              ppp
'ppp                                                          ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTHTUNIT as a subroutine
CORRECTHTUNIT:

'  Change color to blue (1) text and cyan (3) background
   COLOR 1,3

'  Locate the text in row (lline%+8) and column 8
```

```
        LOCATE  lline%+8,34

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'    is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then HTUNIT$="INCHES"
   CASE 1
        HTUNIT$="INCHES"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then HTUNIT$="CENTIMETERS"
   CASE 2
        HTUNIT$="CENTIMETERS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 2, then continue program
        CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

'  Change color to blue (1) text and WHITE (15) background
   COLOR 1,15

'  Tell user the units selected
   IF HTUNIT$="INCHES" THEN
              LOCATE 13,30
        ELSE
              LOCATE 13,27
        END IF
   PRINT " You selected ";HTUNIT$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   COLOR 2,2
   LOCATE 15,16
   PRINT SPACE$(62)
   LOCATE 15,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter age
'  units again.
22520    IF (ans$="N") OR (ans$="n") THEN 22500

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 22525

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 15,53
   PRINT SPACE$(15)
   COLOR 2,2
   LOCATE 17,16
   PRINT SPACE$(62)
   LOCATE 17,18
   CALL InvalidAnswer (ans$)
   GOTO 22520

22525 PALETTE 0,0

'CLEAR SCREEN
   CLS

'  Define end of PROCEDURE GetHtUnit
   END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp            PROCEDURE NAME:  GetWeight                             ppp
'ppp                                                                   ppp
'ppp       PROCEDURE TO GET WEIGHT OF PATIENT                         ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

```
'  Define GetWeight as a PROCEDURE with weight as a
'    passed variable
   SUB GetWeight (weight)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
       SHARED prompt$

palette 6,62
PALETTE 5,63

'  Specify color of writing as LT. BLUE (3) and background as
'    palette color 3
   COLOR 5,5

'set text color (txcol%) and gackground color (bkcol%)
txcol%=3
bkcol%=0

'define beginning and ending rows of box
begrow%=5
endrow%=10

'define beginning and ending columns of box
begcol%=11
endcol%=70

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

9130       COLOR 0,0
     LOCATE 8,37
     PRINT SPACE$(30)
     COLOR 6,0
     LOCATE 8,37
     PRINT " ";
     LOCATE 12,17
     COLOR 5,5
          PRINT SPACE$(52)
     LOCATE 14,17
     COLOR 5,5
     PRINT SPACE$(52)

'LOCATE TEXT
9132     LOCATE 6,13
   COLOR 6,0

'  Prompt user to input weight of patient
        PRINT prompt$;"Weight of patient in pounds or kilograms."
   LOCATE 8,35
        INPUT weight

' erase any previous invalid answers by printing
' spaces same color as background
   COLOR 5,5
   LOCATE 12,20
   PRINT SPACE$(50)

'  If weight <=0, then prompt user to input height again
       IF weight =<0 THEN
     SOUND 75,10
     COLOR 28,5
     LOCATE 12,21
          PRINT "This is NOT A VALID answer for the WEIGHT."
     LOCATE 8,37
     COLOR 0,0
     PRINT SPACE$(30)
     COLOR 3,0
     LOCATE 8,37
     PRINT " ";
          GOTO 9132
   END IF

'  Call PROCEDURE CorrectAnswer to ask if input is correct
        LOCATE 12,28
   CALL CorrectAnswer (ans$)

' erase any previous invalid answers by printing
' spaces same color as background
9140     COLOR 5,5
```

```
    LOCATE 14,63
    PRINT SPACE$(16)

'   Check answer to PROCEDURE CorrectAnswer :

'   If the answer is No, then prompt user to enter patient
'     height again
    IF (ans$="N") OR (ans$="n") THEN 9130

'   If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 9145

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    COLOR 5,5
    LOCATE 12,54
    PRINT SPACE$(15)
    COLOR 5,1
    LOCATE 14,18
        CALL InvalidAnswer(ans$)
        GOTO 9140

'CLEAR SCREEN
9145    CLS

'   Define end of PROCEDURE GetWeight
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              PROCEDURE NAME:  GetWtunit                         ppp
'ppp                                                                 ppp
'ppp          PROCEDURE TO GET UNITS OF WEIGHT                       ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'   Define GetWtunit as a PROCEDURE with wtunit$ as a
'     passed variable
    SUB GetWtunit (wtunit$)

'   Define prompt$ as a shared variable between the PROCEDURE
'     and main prog
    SHARED prompt$

palette 2,63

'   Specify color of writing as  BLUE (1) and background as
'    palette color 2
    COLOR 1,2

'   Clear the screen
    CLS

'   Prompt user to select units
      LOCATE 5,7
      PRINT prompt$;"UNITS OF WEIGHT";
      PRINT " by using the arrow keys to move "
      PRINT "       the selection box to the proper units";
      PRINT "  and press the enter key."

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=8
endrow%=11

'define beginning and ending columns of box
begcol%=31
endcol%=47

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'   Print type of available units
9500    COLOR 1,3
    LOCATE 9,34
    PRINT " POUNDS "
    LOCATE 10,34
    PRINT " KILOGRAMS "
    COLOR 2,2
    LOCATE 13,10
```

```
    PRINT SPACE$(69)
    COLOR 2,2
    LOCATE 15,10
    PRINT SPACE$(69)
    COLOR 2,2
    LOCATE 17,10
    PRINT SPACE$(69)

'    define POUNDS$ as INFO$(1) and add a space
'      at the beginning and end
    INFO$(1)=SPACE$(1)+"POUNDS"+SPACE$(1)

'    define KILOGRAMS$ as INFO$(2) and add a space
'      at the beginning and end
    INFO$(2)=SPACE$(1)+"KILOGRAMS"+SPACE$(1)

'   define LLINE% as an integer used to locate text
'    on proper row
    LLINE%=1

'   change color to BLUE (1) text and CYAN (3)
'    background;  locate the text at row 9 and
'    column 34;  print INFO$(1)
    COLOR 1,2
         LOCATE 9,34
         PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                       XX
'XX       This selection routine was originally written                  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                      XX
'XX                                                                       XX
'XX       Modifications and all documentation                            XX
'XX     was done by Cynthia K. Rice on 3-1-89.                            XX
'XX                                                                       XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'   mark this part of program with a label SelectNtUnit
'    so it can be called again
    SelectNtUnit:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
         KKEY$=INKEY$

'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
'  Beginning of SELECT CASE structure = case variable
'    is KKEY$
    SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'   if KKEY$="" (no keystroke on keyboard buffer), then
'    goto SelectNtUnit (continue selection process)
         CASE ""
         GOTO SelectNtUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'   if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'    then execute the following lines

'    CHR$(n) is the character in the ASCII table that
'     is associated with n (the ASCII value)

'      CHR$(0)+CHR$(72) represents the up arrow
'      CHR$(0)+CHR$(80) represents the down arrow

'    The CHR$(0) represents that this is the extended
'     code for a non-ASCII key
        CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

       ' change the color to BLUE (1) text and YELLOW (2)
       '  background;  print text on row LLINE%+8 and
       '   column 33
          COLOR 1,3
          LOCATE LLINE%+8,34

       ' print INFO$(LLINE%)
          PRINT INFO$(LLINE%)

       ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
```

```
        '  then decrement LLINE% by 1
        ' ELSE increment LLINE% by 1 (meaning the up arrow
        '  was selected)
           IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

        ' need to keep LLINE% in allowable range for data
        '  so the selection box will scroll from last
        '  allowable selection to first allowable selection
        '  and vice versa
        ' If select up arrow when on top line will go to
        '  bottom line
      IF LLINE%<1 THEN LLINE%=2

        ' If select down arrow when on bottom line will go to
        '  top line
      IF LLINE%>2 THEN LLINE%=1

        ' change the color to BLUE (1) text and CYAN (3)
        '  background;  print next text at row (LLINE%+8)
        '   and column 8
        COLOR 1,2
        LOCATE LLINE%+8,34

        ' print INFO$(LLINE%)
          PRINT INFO$(LLINE%)

        ' goto label SelectWtUnit to continue selection process
      GOTO SelectWtUnit:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
        CASE CHR$(13)

      '  goto subroutine CorrectWtUnit to correct sex info
          GOTO CorrectWtUnit:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label SelectAgeU
'  to continue selection process
        CASE ELSE
        GOTO SelectWtUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
        END SELECT
'##############################################################################
'##############################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectWtUnit
      RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                      ppp
'ppp          SUBROUTINE NAME: CorrectWtUnit                              ppp
'ppp                                                                      ppp
'ppp         SUBROUTINE TO SELECT WEIGHT UNIT                            ppp
'ppp                                                                      ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTWTUNIT as a subroutine
CORRECTWTUNIT:

'  Change color to blue (1) text and cyan (3) background
   COLOR 1,3

'  Locate the text in row (lline%+8) and column 8
      LOCATE  lline%+8,34

'##############################################################################
'##############################################################################
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 1, then WTUNIT$="POUNDS"
   CASE 1
      WTUNIT$="POUNDS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then WTUNIT$="KILOGRAMS"
   CASE 2
        WTUNIT$="KILOGRAMS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 2, then continue program
        CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'################################################################
'################################################################

'  Change color to blue (1) text and WHITE (15) background
   COLOR 1,15

'  Tell user the units selected
   IF WTUNIT$="POUNDS" THEN
              LOCATE 13,29
        ELSE
              LOCATE 13,28
        END IF
   PRINT " You selected ";WTUNIT$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   COLOR 2,2
   LOCATE 15,16
   PRINT SPACE$(62)
   LOCATE 15,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter age
'  units again.
9520     IF (ans$="N") OR (ans$="n") THEN 9500

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 9525

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 15,53
   PRINT SPACE$(15)
   COLOR 2,2
   LOCATE 17,16
   PRINT SPACE$(62)
   LOCATE 17,18
   CALL InvalidAnswer (ans$)
   GOTO 9520

9525 PALETTE 0,0

'CLEAR SCREEN
   CLS

'  Define end of PROCEDURE GetWtUnit
   END SUB
'
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: showpat.BAS                ==
'==               backup prog : showpat.BAK                ==
'==                                                        ==
'============================================================
' PROGRAM WRITTED:  2-26-89                                 ==
'                                                           ==
' PROGRAM MODIFIED: 4-24-89                                 ==
'============================================================
'==  PROGRAM NAME:  showpat.BAS                            ==
'==                                                        ==
'==    This program contains the subroutines to show       ==
'==  patient information.                                  ==
```

```
'==                                                              ==
'==============================================================
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                            ppp
'ppp            SUBROUTINE NAME:  SHOWPATINFO                   ppp
'ppp                                                            ppp
'ppp     SUBROUTINE TO DISPLAY PATIENT INFORMATION              ppp
'ppp                                                            ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define SHOWPATINFO as a SUBROUTINE
   SHOWPATINFO:

'  Specify color of writing as blue (1) and background
'   as palette color 3
   COLOR 1,3

PALETTE 2,61
PALETTE 14,0

'  Specify color of writing as LT. BLUE (3) and background as
'   palette color 3
1333  COLOR 1,3

'set text color (txcol%) and gackground color (bkcol%)
txcol%=15
bkcol%=1

'define beginning and ending rows of box
begrow%=row%-1
endrow%=row%+9

'define beginning and ending columns of box
begcol%=col%-3
endcol%=col%+4

'CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Print patient name on screen
   LOCATE row%,col%+1
        PRINT"NAME OF PATIENT:  "
        LOCATE row%,col%+21
        PRINT SPACE$(1)+potname$

'  Print sex of patient on screen
   LOCATE row%+1,col%+1
        PRINT"SEX OF PATIENT:  "
        LOCATE row%+1,col%+21
        PRINT SPACE$(1)+sex$

'  Print ID# of patient on screen
   LOCATE row%+2,col%+1
        PRINT"PATIENT ID NUMBER:  "
        LOCATE row%+2,col%+21
        PRINT SPACE$(1)+IDNUM$

'  Print AGE of patient on screen
   LOCATE row%+3,col%+1
        PRINT"AGE OF PATIENT:  "
        LOCATE row%+3,col%+20
        PRINT SPACE$(1);AGE

'  Print units of age on screen
      LOCATE row%,col%+1
        PRINT"UNITS OF AGE:  "
        LOCATE row%+4,col%+21
        PRINT SPACE$(1)+ageunit$

'  Print height of patient on screen
      LOCATE row%,col%+1
        PRINT"HEIGHT OF PATIENT:  "
        LOCATE row%+5,col%+20
        PRINT SPACE$(1);height

'  Print units of height on screen
   LOCATE row%+6,col%+1
        PRINT"UNITS OF HEIGHT:  "
        LOCATE row%+6,col%+21
        PRINT SPACE$(1)+htunit$
```

```
'  Print weight of patient on screen
   LOCATE row%+7,col%+1
        PRINT"WEIGHT OF PATIENT:  "
        LOCATE row%+7,col%+20
        PRINT  SPACE$(1);weight

'  Print units of weight on screen
   LOCATE row%+8,col%+1
        PRINT"UNITS OF WEIGHT:    "
        LOCATE row%+5,col%+21
        PRINT SPACE$(1)+wtunit$

'  Print blank line
   PRINT

'  End of SUBROUTINE SHOWPATINFO, return to main program
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp           SUBROUTINE NAME:  CheckPatInfo                        ppp
'ppp                                                                 ppp
'ppp      SUBROUTINE TO CHECK PATIENT INFORMATION                    ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define CheckPatInfo as a SUBROUTINE
   CheckPatInfo:

'  Call PROCEDURE CorrectAnswer to ask if answer is correct

      LOCATE 24,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'   If answer is No, then go to SUBROUTINE CorrectPatInfo
'    to correct incorrect input of patient information
130      IF (ans$="n") OR (ans$="N") THEN

'erase correct answer question
      LOCATE 14,33
   COLOR 3,3
   PRINT SPACE$(52)

'  Specify color of writing as red (4) and background
'     as CYAN (3)
   COLOR 4,3

'  Tell user to use up and down arrow keys to select
'   the incorrect entry
     locate 16,6
    PRINT"Please use the UP arrow key ";CHR$(24);
'     locate 17,6
    PRINT" and the DOWN arrow key ";CHR$(25)
     locate 17,26
    PRINT"to move the box to the line needing correction."
     locate 18,39
        PRINT
       locate 19,16
    PRINT"Then, press the enter key to select it and ";
     locate 20,45
        PRINT"type the correct input."

        GOSUB CorrectPatInfo
        GOTO 155

'  Define end of IF block
   END IF

'  If answer is Yes, then go back to main program
       IF (ans$="y") OR (ans$="Y") THEN 135

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
     LOCATE 14,44
       COLOR 3,3
       PRINT SPACE$(15)
     LOCATE 16,44
       COLOR 3,3
       PRINT SPACE$(61)
     LOCATE 16,44
  CALL InvalidAnswer (ans$)
     GOTO 530
```

```
535 PALETTE 0,0
    PALETTE 3,3

'  Define END of SUBROUTINE CheckPatInfo
   RETURN


'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp          SUBROUTINE NAME:  CorrectPatInfo                         ppp
'ppp                                                                   ppp
'ppp    SUBROUTINE TO SELECT PATIENT INFORMATION                       ppp
'ppp    THAT IS IN ERROR                                               ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define CorrectPatInfo as a SUBROUTINE
   CorrectPatInfo:

'  Print two blank lines
1000    PRINT
   PRINT

'  Assign patient information to string array INFO$
'   define patname$ as INFO$(1) and add a space
'    at the beginning and end
      INFO$(1)=SPACE$(1)+patnome$+SPACE$(1)

'   define sex$ as INFO$(2) and add a space
'    at the beginning and end
        INFO$(2)=SPACE$(1)+sex$+SPACE$(1)

'   define IDNUM$ as INFO$(3) and add a space
'    at the beginning and end
        INFO$(3)=SPACE$(1)+IDNUM$+SPACE$(1)

'   define numerical variable AGE as a string (STR$)
'    and assign it to INFO$(4) with a space added
'    at the end
   INFO$(4)=STR$(AGE)+SPACE$(1)

'   define ageunit$ as INFO$(5) and add a space
'    at the beginning and end
        INFO$(5)=SPACE$(1)+ageunit$+SPACE$(1)

'   define numerical variable height as a string (STR$)
'    and assign it to INFO$(6) with a space added
'    at the end
   INFO$(6)=STR$(height)+SPACE$(1)

'   define htunit$ as INFO$(7) and add a space
'    at the beginning and end
        INFO$(7)=SPACE$(1)+htunit$+SPACE$(1)

'   define numerical variable weight as a string (STR$)
'    and assign it to INFO$(8) with a space added
'    at the end
   INFO$(8)=STR$(weight)+SPACE$(1)

'   define wtunit$ as INFO$(9) and add a space
'    at the beginning and end
   INFO$(9)=SPACE$(1)+wtunit$+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
     LLINE%=row%-6

'  change color to white (15) text and red (12)
'   background;  locate the text at row row% and
'   column 23;  print INFO$(1)
     COLOR 15,12
     LOCATE row%,col%+21
     PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                            XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                     XX
'XX       This selection routine was originally written                 XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                    XX
'XX                                                                     XX
'XX       Modifications and all documentation                           XX
'XX    done by Cynthia K. Rice on 3-1-89.                               XX
```

```
'XX                                                                    XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label CORRECTIT
'   so it can be called again
   CORRECTIT:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
       KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto CORRECTIT (continue selection process)
       CASE ""
       GOTO CORRECT
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
       CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to blue (1) text and CYAN (3)
      '  background;  print next text at row (LLINE%+2)
      '   and column 23
         COLOR 15,1
         LOCATE LLINE%+2,col%+71

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
      '  then decrement LLINE% by 1
      ' ELSE increment LLINE% by 1 (meaning the up arrow
      '  was selected)
         IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

      ' need to keep LLINE% in allowable range for data
      '  so the selection box will scroll from last allowable
      '  selection to first allowable selection and vice
      '  versa
      ' If select up arrow when on top line will go to
      '  bottom line
   IF LLINE%<1 THEN LLINE%=9

      ' If select down arrow when on bottom line will go to
      '  top line
   IF LLINE%>9 THEN LLINE%=1

      ' change the color to white (15) text and red (12)
      '  background;  print next text at row (LLINE%+2)
      '   and column 23
         COLOR 15,12
          LOCATE LLINE%+2,col%+71

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' goto label CORRECTIT to continue selection process
   GOTO CORRECTIT:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
       CASE CHR$(13)

     '  goto subroutine DoCorrection to correct patient info
```

```
          GOSUB DoCorrection:

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label CORRECT
'  to continue selection process
        CASE ELSE
          GOTO CORRECT
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine CorrectPatInfo
      RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                           ppp
'ppp          SUBROUTINE NAME: DoCorrection                    ppp
'ppp                                                           ppp
'ppp    SUBROUTINE TO CORRECT ERROR OF PATIENT INPUT           ppp
'ppp                                                           ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define DoCorrection as a subroutine
DoCorrection:

'  Change color to blue (1) text and cyan (3) background
   color 15,1

'  Locate the text in row (lline%+2) and column 24
     LOCATE  line%+2,col%+23

'  Print spaces the length of INFO$ to be sure all of
'   string is erased
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Change color to white (15) text and red (12) background
     color  5, 2

'  Locate the text in row (lline%+2) and column 23
50501    LOCATE  lline%+2,col%+21

'  Print a spaces to locate cursor for user
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Locate the text in row (lline%+2) and column 24
   LOCATE lline%+2,col%+22

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
1010     SELECT CASE LLINE%

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then input patient name again
       CASE 1
        INPUT "", patnome$
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then input sex of patient again
      CASE 2
      INPUT "",sex$

    '  If sex is male (m) or female (f),
    '   continue program
        IF (sex$="f") OR (sex$="F") OR (sex$="m") OR (sex$="M") THEN
     IF (sex$="f") OR (sex$="F") THEN SEX$="FEMALE"
         IF (sex$="m") OR (sex$="M") THEN SEX$="MALE"
        GOTO 12330

    '   ELSE sex is not male or female
         ELSE
        LOCATE LLINE%+2,COL%+20
        COLOR 12,10
        PRINT SPACE$(3)
        SOUND 100,100
     COLOR 31,12
         LOCATE lline%+3,15
```

```
      '  tell user that entry of sex is invalid,
      '   please input proper sex of patient
          PRINT " This is not a valid entry for the sex of patient.     "
          LOCATE lline%+4,15
          PRINT " Please enter 'M' for male or 'F' for female.          "
      COLOR 15,12
      '  goto  select case again to enter sex
          GOTO 51501

   '  define end of if block
12330    END IF
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 3, then input ID # of patient again
      CASE 3
      INPUT "",IDNUM$
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 4, then input value of AGE again
      CASE 4
      INPUT "", AGE

    '  If AGE <=0, then prompt user to input AGE again
       IF AGE =<0 THEN
        LOCATE LLINE%+2,COL%+40
   COLOR 12,12
   PRINT SPACE$(17)
     SOUND 100,10
     COLOR 31,12
     LOCATE LLINE%+3,20
          PRINT " This is not a valid answer for the age.  "
          LOCATE LLINE%+4,20
          PRINT " Please enter a number greater than zero. "
     COLOR 15,12
          GOTO 50501
       END IF
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 5, then input units of age again
   CASE 5
   INPUT "",ageunit$

    '  If units are months (mon) or years (yr),
    '   continue program
        IF (ageunit$="mon") OR (ageunit$="MON") OR (ageunit$="YR") OR (ageunit$="yr") THEN
         IF (ageunit$="mon") OR (ageunit$="MON") THEN AGEUNIT$="MONTHS"
         IF (ageunit$="YR") OR (ageunit$="yr") THEN AGEUNIT$="YEARS"
         GOTO 31313

    '   ELSE units are not months or years
         ELSE
        LOCATE LLINE%+2,COL%+20
        COLOR 12,12
        PRINT SPACE$(22)
     SOUND 100,10
     COLOR 31,12
         LOCATE lline%+3,17
     '  tell user that entry of age units is invalid,
     '   please input proper age units
         PRINT " This is not a valid entry for the units of age.  "
         LOCATE lline%+4,17
         PRINT " Please enter 'MON' for months or 'YR' for years. "
     COLOR 15,12
     '  goto  select case again to enter age unit
         GOTO 50511

   '  define end of if block
31323    END IF
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 6, then input value of height again
   CASE 6
    INPUT "", height

    '  If height <=0, then prompt user to input height again
       IF height =<0 THEN
        LOCATE LLINE%+2,COL%+20
        COLOR 2,2
        PRINT SPACE$(17)
     SOUND 100,10
```

```
      COLOR 31,12
      LOCATE LLINE%+3,18
           PRINT " This is not a valid answer for the height. "
           LOCATE LLINE%+4,18
           PRINT " Please enter a number greater than zero.   "
      COLOR 15,12
           GOTO 50501
        END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 7, then input units of height again
    CASE 7
     INPUT "",htunit$

     '  If units are inches (in) or centimeters (cm),
     '   continue program
          IF (htunit$="in") OR (htunit$="IN")
           OR (htunit$="cm") OR (htunit$="CM") THEN
          IF (htunit$="in") OR (htunit$="IN") THEN HTUNIT$="INCHES"
          IF (htunit$="cm") OR (htunit$="CM") THEN HTUNIT$="CENTIMETERS"
           GOTO 30333

     '   ELSE units are not inches or centimeters
          ELSE
      LOCATE LLINE%+2,COL%+20
    COLOR 12,12
    PRINT SPACE$(25)
      SOUND 100,10
      COLOR 31,12
           LOCATE lline%+3,15
     '  tell user that entry of height units is invalid,
     '   please input proper height units
           PRINT " This is not a valid entry for the units of height.    "
           LOCATE lline%+4,15
           PRINT " Please enter 'CM' for centimeters or 'IN' for inches. "
      COLOR 15,12
     '  goto  select case again to enter height unit
           GOTO 56501

    '  define end of if bl ck
36300    END IF
'++++++/+/+/+/++++++++++++++++++++++++++++/+++/+/+)+)+/+++

'++++++++++++++++++;+++++++++;++++++++++++++++++++++++/+++)+
    '  If LLINE% = 8, then e¬ter value of weight agai¬
          CASE 8
      INPUT "",weyght
      ' If weight <=0, the. proept user to input weight agayn
          IF weight =<0 THEN
      LOCATE LLINE%+2,COL%+20
    COLOR 12,12
    PRINT SPACE$(17)
      SOUND 100,10
      COLOR 31,12
      LOCATE LLINE%+3,18
           PRINT " This is not a valid answer for the weight. "
      LOCATE LLINE%+4,18
           PRINT " Please enter a number greater than zero.   "
      COLOR 15,12
       ' goto select case again to enter weight
           GOTO 50501

      ' define end of if block
          END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    '  If LLINE% = 9, then input units of weight again
          CASE 9
      INPUT "",wtunit$
     '  If units are pounds (lbs) or kilograms (kg),
     '   continue program
          IF (wtunit$="lbs") OR (wtunit$="LBS")
           OR (wtunit$="kg") OR (wtunit$="KG") THEN
          IF (wtunit$="lbs") OR (wtunit$="LBS") THEN WTUNIT$="POUNDS"
          IF (wtunit$="kg") OR (wtunit$="KG") THEN WTUNIT$="KILOGRAMS"
           GOTO 50500

     '  ELSE units are not pounds or kilograms
          ELSE
      LOCATE LLINE%+2,COL%+20
:   COLOR 12,12
:   PRINT SPACE$(24)
      SOUND 100,10
```

```
      COLOR 31,12
           LOCATE lline%+3,14
           PRINT " This is not a valid entry for the units of weight.    "
           LOCATE lline%+4,14
           PRINT " Please enter 'KG' for kilograms or 'LBS' for pounds.  "
      COLOR 15,12
      '  goto  select case again to enter height unit
           GOTO 50501

   '  define end of if block
56500    END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% > 9, then continue program
         CASE ELSE
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
         END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Change color to blue (1) text and cyan (3) background
    color 1,3

'  Clear the Screen
    CLS

'  Goto subroutine SHOWPATINFO to display current patient
'   information
    GOSUB SHOWPATINFO

'  Goto subroutine CheckPatInfo to check if patient
'   information is correct
         GOSUB CheckPatInfo

PALETTE 0,5
PALETTE 3,3

'  Define end of SUBROUTINE CorrectPatInfo return to main
1620  RETURN
'-------------------------------------------------------------------------------



'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp             SUBROUTINE  NAME:  FilePat                           ppp
'ppp                                                                   ppp
'ppp       SUBROUTINE TO FILE PATIENT INFORMATION                      ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define FilePat as a SUBROUTINE to file patient
'    information
FilePat:

'  Open file to store patient information
OPEN "PATIENT.DAT" FOR OUTPUT AS #11

'  Store patient information
WRITE #11, patname$,sex$,idnum$,age,ageunit$,
height,htunit$,weight,wtunit$

'  Close Pat.DAT fi e (#11)
CLOSE #11

'  Defife end of SUBROUTINE FilePat
RETURN
'-------------------------------------------------------------------------------



'===============================================================
'==             ***** IMPORTANT INFO *****                    ==
'==                                                           ==
'==             program name:  2CORREC.BAS                    ==
'==             backup prog :  2CORREC.BAK                    ==
```

```
'==                                                              ==
'==============================================================
'PROGRAM WRITTEN:   2-8-89                                       '
'                                                                '
'PROGRAM MODIFIED:  4-20-89                                      '
'==============================================================
'==  PROGRAM NAME: 2CORREC.INC                                   ==
'==                                                              ==
'==    This program asks user if response is correct             ==
'==  and also checks for a response of yes or no.                ==
'==                                                              ==
'==    This $INCLUDE program contains PROCEDURES:                ==
'==                        CorrectAnswer                         ==
'==                        InvalidAnswer                         ==
'==                                                              ==
'==============================================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                              ppp
'ppp          PROCEDURE NAME:  CorrectAnswer                      ppp
'ppp                                                              ppp
'ppp  PROCEDURE TO ASK IF ANSWER TO QUESTION IS CORRECT           ppp
'ppp                                                              ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define CorrectAnswer as a PROCEDURE with ans$ as
'   a passed variable
   SUB CorrectAnswer (ans$)

'  Ask user if input is correct
      COLOR 1,15
      PRINT" Is this correct? (Y/N) ";
      COLOR 20,15
      INPUT ans$
      COLOR 1,15

'  Print a blank line
   PRINT

'  Define end of PROCEDURE CorrectAnswer
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                              ppp
'ppp          PROCEDURE NAME:  InvalidAnswer                      ppp
'ppp                                                              ppp
'ppp PROCEDURE TO TELL OF INVALID RESPONSE TO A QUESTION          ppp
'ppp                                                              ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define InvalidAnswer as a PROCEDURE with ans$ as
'   a passed variable
   SUB InvalidAnswer (ans$)

'  SOUND EFFECT TO GET ATTENTION OF USER
   SOUND 37,10

'  Tell user that answer was invalid,
'    please answer properly with y or n
  COLOR 11,4
  PRINT " INVALID ANSWER!  Please enter 'Y' or 'N'. ";
  INPUT ANS$
'  COLOR 1,15


'  Define end of PROCEDURE InvalidAnswer
   END SUB
'
```

```
'==============================================================
'==              ***** IMPORTANT INFO *****                   ==
'==                                                          ==
'==              program name: TEMPNAME.BAS                  ==
'==              backup prog : TEMPNAME.BAK                  ==
'==                                                          ==
'==============================================================
' PROGRAM WRITTED:  2-26-89                                   '
'                                                            '
' PROGRAM MODIFIED: 5-21-89                                   '
```

```
'============================================================
'==                                                        ==
'==   This program calls the $INCLUDE programs:            ==
'==                4TEMPIN.BAS                             ==
'==                                                        ==
'============================================================

'============================================================
'==  PROGRAM NAME:  TEMPNAME.BAS                           ==
'==                                                        ==
'==    This program contains subroutines to get            ==
'==  temperature names.                                    ==
'==                                                        ==
'==    This program prompts user for title of temperature  ==
'==  probes.  Then, the titles are displayed in the        ==
'==  following manner:                                     ==
'==                                                        ==
'==             1.  NAME OF TEMPERATURE #1                 ==
'==             2.  NAME OF TEMPERATURE #2                 ==
'==             3.  NAME OF TEMPERATURE #3                 ==
'==             4.  NAME OF TEMPERATURE #4                 ==
'==             5.  NAME OF TEMPERATURE #5                 ==
'==             6.  NAME OF TEMPERATURE #6                 ==
'==                                                        ==
'============================================================
'  define GETTEMPNAMES  as a subroutine
GETTEMPNAMES:

'  Call procedure to get units of temperature
   CALL GetTEMPunit(TEMPUNITS$)

'  Clear the Screen
   CLS

'  Call procedure to get name of temperature #1
   CALL Temp1 (temp1$)

'  Call procedure to get name of temperature #2
   CALL Temp2 (temp2$)

'  Call procedure to get name of temperature #3
   CALL Temp3 (temp3$)

'  Call procedure to get name of temperature #4
   CALL Temp4 (temp4$)

'  Call procedure to get name of temperature #5
   CALL Temp5 (temp5$)

'  Call procedure to get name of temperature #6
   CALL Temp6 (temp6$)

'  Clear the screen
   CLS

'  define row and column for centering text
      row%=1
      col%=26

'  Call SUBROUTINE to display temperature names
   GOSUB TempDisp

'  Call SUBROUTINE to check temperature names
   GOSUB CheckTempNames


'  End of subroutine GETTEMPNAMES
   RETURN
'============================================================

'  Program stored under different name to save
'   space in the TURBOBASIC editor
   $INCLUDE "c:\monitor1\4TEMPIN.BAS"




'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: 4TEMPIN.BAS                ==
'==               backup prog : 4tempin.bak                ==
'==                                                        ==
```

```
'============================================================
'    PROGRAM WRITTEN:   2-10-89                             '
'                                                           '
'    PROGRAM MODIFIED:  5-21-89                             '
'============================================================
'==  PROGRAM NAME: 4TEMPIN.INC                              ==
'==                                                         ==
'==    This program prompts user for title of temperature   ==
'==  probes.  Then, the titles are displayed in the         ==
'==  following manner:                                      ==
'==                                                         ==
'==              1.  NAME OF TEMPERATURE #1                 ==
'==              2.  NAME OF TEMPERATURE #2                 ==
'==              3.  NAME OF TEMPERATURE #3                 ==
'==              4.  NAME OF TEMPERATURE #4                 ==
'==              5.  NAME OF TEMPERATURE #5                 ==
'==              6.  NAME OF TEMPERATURE #6                 ==
'==                                                         ==
'============================================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                           ppp
'ppp                PROCEDURE NAME:  Templ                     ppp
'ppp                                                           ppp
'ppp      PROCEDURE TO GET NAME OF TEMPERATURE #1              ppp
'ppp                                                           ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Templ as a PROCEDURE with templ$ as a passed
'   variable
   SUB Templ (templ$)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
   SHARED prompt$

'  Specify color of writing as blue (1) and background as
'   palette color 1
   COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=2
bkcol%=4

'define beginning and ending rows of box
begrow%=33
endrow%=56

'define beginning and ending columns of box
begcol%=5
endcol%=35

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
203     COLOR 3,3
     LOCATE 10,33
     PRINT SPACE$(27)
     COLOR 1,3
     LOCATE 10,33
     PRINT " ";
     LOCATE 13,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 15,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input name of temperature #1
      color 1,3
      PRINT prompt$;"the name of temperature #1."
      locate 7,27
      print"(maximum of 15 characters)";
      locate 8,17
```

```
      PRINT "If probe #1 is not connected, then enter 'N/C'."
      locate 10,31
      INPUT temp1$

'  if length of temp1$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(temp1$)
   IF length%>15 then temp1$=left$(temp1$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 13,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 13,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter name of
'   temperature #1 again.
220     IF (ans$="N") OR (ans$="n") THEN 200
   LOCATE 13,53
   COLOR 1,1
   PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 225

' erase any previous invalid answers by printing
'  spaces same color as background
   LOCATE 15,18
   COLOR 1,1
   PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
'  LOCATE 15,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 15,18
        CALL InvalidAnswer(ans$)
        GOTO 220

'CLEAR SCREEN
255   CLS

'  Define end of PROCEDURE Temp1
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                               ppp
'ppp                PROCEDURE NAME:  Temp2                        ppp
'ppp                                                               ppp
'ppp      PROCEDURE TO GET NAME OF TEMPERATURE #2                  ppp
'ppp                                                               ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Temp2 as a PROCEDURE with temp2$ as a passed
'   variable
   SUB Temp2 (temp2$)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
   SHARED prompt$

PALETTE 1,62
PALETTE 3,1

'  Specify color of writing as blue (1) and background as
'   palette color 1
   COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=11

'define beginning and ending columns of box
begcol%=15
endcol%=65
```

```
'  CLEAR THE SCREEN
   CLS

'call procedure )en)bgx to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%, xcol%<bkcol%)

' erase any prgvious invalid answers by printing
' spaces same color as background
230       COLOR 3,3
     LOCA\E 10,33
     PRINT SPACE$(27)
     COLOR 1,3
     LOCATE 10,33
     PRINT " ";
     LOCATE 13,27
     COLOr 1,1
     PRINT SPACE$(52)
          LOCATE 15,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input name of temperature #2
      color 1,3
      PRINT prompt$;"the name of temperature #2."
      locate 7,27
      print"(maximum of 15 characters)";
      locate 8,17
      PRINT "If probe #2 is not connected, then enter 'N/C'."
      locate 10,31
      INPUT temp2$

'  if length of temp2$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(temp2$)
   IF length%>15 then temp2$=left$(temp2$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 13,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 13,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter name of
'   temperature #1 again.
235      IF (ans$="N") OR (ans$="n") THEN 230
   LOCATE 13,53
   COLOR 1,1
   PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 240

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 15,18
   COLOR 1,1
   PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
'  LOCATE 15,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 15,18
        CALL InvalidAnswer(ans$)
        GOTO 235

240 PALETTE 1,1
    PALETTE 3,3

'CLEAR SCREEN
   CLS

'  Define end of PROCEDURE Temp2
   END SUB
```

```
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp              PROCEDURE NAME:  Temp3                               ppp
'ppp                                                                   ppp
'ppp     PROCEDURE TO GET NAME OF TEMPERATURE #3                       ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Temp3 as a PROCEDURE with temp3$ as a passed
'   variable
   SUB Temp3 (temp3$)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
   SHARED prompt$

'  Specify color of writing as blue (1) and background as
'   palette color 1
   COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=11

'define beginning and ending columns of box
begcol%=15
endcol%=65

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
250       COLOR 3,3
     LOCATE 10,33
     PRINT SPACE$(27)
     COLOR 1,3
     LOCATE 10,33
     PRINT " ";
     LOCATE 13,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 15,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input name of temperature #3
      color 1,3
      PRINT prompt$;"the name of temperature #3."
      locate 7,27
      print"(maximum of 15 characters)";
      locate 8,17
      PRINT "If probe #3 is not connected, then enter 'N/C'."
      locate 10,31
      INPUT temp3$

'  if length of temp3$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(temp3$)
   IF length%>15 then temp3$=left$(temp3$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 13,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 13,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter name of
'   temperature #1 again.
255      IF (ans$="N") OR (ans$="n") THEN 250
```

```
    LOCATE 13,53
    COLOR 1,1
    PRINT SPACE$(24)

'   If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 260

' erase any previous invalid answers by printing
' spaces same color as background
    LOCATE 15,18
    COLOR 1,1
    PRINT SPACE$(61)

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
'   LOCATE 15,17
    COLOR 1,1
    PRINT SPACE$(62)
    LOCATE 15,18
         CALL InvalidAnswer(ans$)
         GOTO 255

'CLEAR SCREEN
264   CLS

'   Define end of PROCEDURE Temp3
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp                PROCEDURE NAME:  Temp4                             ppp
'ppp                                                                   ppp
'ppp       PROCEDURE TO GET NAME OF TEMPERATURE $4                     ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'   Define Temp4 as a PROCEDURE with temp4$ as a passed
'     variable
    SUB Temp4 (temp4$)

'   Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
    SHARED prompt$

PALETTE 1,62
PALETTE 3,1

'   Specify color of writing as blue (1) and background as
'    palette color 1
    COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=11

'define beginning and ending columns of box
begcol%=15
endcol%=65

'   CLEAR THE SCREEN
    CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
274     COLOR 4,3
     LOCATE 10,33
     PRINT SPACE$(27)
     COLOR 1,3
     LOCATE 10,33
     PRINT " ";
     LOCATE 13,27
          COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 15,17
          COLOR 1,1
          PRINT SPACE$(62)
```

```
'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input name of temperature #4
      color 1,3
      PRINT prompt$;"the name of temperature #4."
      locate 7,27
      print"(maximum of 15 characters)";
      locate 8,17
      PRINT "If probe #4 is not connected, then enter 'N/C'."
      locate 10,31
      INPUT temp4$

'  if length of temp4$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(temp4$)
   IF length%>15 then temp4$=left$(temp4$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 13,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 13,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter name of
'   temperature #1 again.
275     IF (ans$="N") OR (ans$="n") THEN 270
   LOCATE 13,53
   COLOR 1,1
   PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
      IF (ans$="y") OR (ans$="Y") THEN 284

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 15,18
   COLOR 1,1
   PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
'  LOCATE 15,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 15,18
        CALL InvalidAnswer(ans$)
        GOTO 275

280 PALETTE 1,1
    PALETTE 3,3

'CLEAR SCREEN
   CLS

'  Define end of PROCEDURE Temp4
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp                    PROCEDURE NAME:  Temp5                       ppp
'ppp                                                                 ppp
'ppp       PROCEDURE TO GET NAME OF TEMPERATURE #5                   ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Temp5 as a PROCEDURE with temp5$ as a passed
'   variable
   SUB Temp5 (temp5$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$

'  Specify color of writing as blue (1) and background as
'   palette color 1
   COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
```

```
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=11

'define beginning and ending columns of box
begcol%=15
endcol%=65

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
290       COLOR 3,3
     LOCATE 10,33
     PRINT SPACE$(27)
     COLOR 1,3
     LOCATE 10,33
     PRINT " ";
     LOCATE 13,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 15,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input name of temperature #5
      color 1,3
      PRINT prompt$;"the name of temperature #5."
      locate 7,27
      print"(maximum of 15 characters)";
      locate 8,17
      PRINT "If probe #5 is not connected, then enter 'N/C'."
      locate 10,31
      INPUT temp5$

'  if length of temp5$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(temp5$)
   IF length%>15 then temp5$=left$(temp5$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 13,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 13,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter name of
'   temperature #1 again.
295      IF (ans$="N") OR (ans$="n") THEN 290
   LOCATE 13,53
   COLOR 1,1
   PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 300

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 15,18
   COLOR 1,1
   PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
'  LOCATE 15,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 15,18
        CALL InvalidAnswer(ans$)
        GOTO 295
```

```
'CLEAR SCREEN
300      CLS

'  Define end of PROCEDURE Temp5
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp                    PROCEDURE NAME:  Temp6                         ppp
'ppp                                                                   ppp
'ppp        PROCEDURE TO GET NAME OF TEMPERATURE #6                    ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Temp6 as a PROCEDURE with temp6$ as a passed
'    variable
   SUB Temp6 (temp6$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$

PALETTE 1,62
PALETTE 3,1

'  Specify color of writing as blue (1) and background as
'   palette color 1
   COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=11

'define beginning and ending columns of box
begcol%=15
endcol%=65

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
320       COLOR 3,3
     LOCATE 10,33
     PRINT SPACE$(27)
     COLOR 1,3
     LOCATE 10,33
     PRINT " ";
     LOCATE 13,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 15,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,20

'  Prompt user to input name of temperature #6
      color 1,3
      PRINT prompt$;"the name of temperature #6."
      locate 7,27
      print"(maximum of 15 characters)";
      locate 8,17
      PRINT "If probe #6 is not connected, then enter 'N/C'."
      locate 10,31
      INPUT temp6$

'  if length of temp6$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(temp6$)
   IF length%>15 then temp6$=left$(temp6$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 13,27
   COLOR 1,1
```

```
    PRINT SPACE$(52)
    LOCATE 13,27
    CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter name of
'    temperature #1 again.
325      IF (ans$="N") OR (ans$="n") THEN 320
    LOCATE 13,53
    COLOR 1,1
    PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 330

' erase any previous invalid answers by printing
' spaces same color as background
    LOCATE 15,18
    COLOR 1,1
    PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
'   LOCATE 15,17
    COLOR 1,1
    PRINT SPACE$(62)
    LOCATE 15,18
         CALL InvalidAnswer(ans$)
         GOTO 325

330 PALETTE 1,1
     PALETTE 3,3

'CLEAR SCREEN
    CLS

'  Define end of PROCEDURE Temp6
    END SUB

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                    ppp
'ppp              SUBROUTINE NAME:  TempDisp            ppp
'ppp                                                    ppp
'ppp      SUBROUTINE TO DISPLAY TEMPERATURE NAMES       ppp
'ppp                                                    ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TempDisp as a SUBROUTINE
    TempDisp:

' define text as BLUE (9) and backgroud as palette color 0
    COLOR 1,3

PALETTE 2,63
PALETTE 14,0

' define text as BLUE (9) and backgroud as palette color 0
340      COLOR 3,3

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=15
bkcol%=1

'define beginning and ending rows of box
begrow%=4
endrow%=11

'define beginning and ending columns of box
begcol%=17
endcol%=65

'  CLEAR THE SCREEN
    CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

      '  Print temperature #1 name on screen
    LOCATE row%+2,col%+1
    PRINT "NAME OF TEMPERATURE #1 IS  "
    LOCATE row%+2,col%+28
    PRINT temp1$
```

```
    ' Print temperature #2 name on screen
    LOCATE row%+3,col%+1
    PRINT "NAME OF TEMPERATURE #2 IS  "
    LOCATE row%+3,col%+28
    PRINT temp2$

    ' Print temperature #3 name on screen
    LOCATE row%+4,col%+1
    PRINT "NAME OF TEMPERATURE #3 IS  "
    LOCATE row%+4,col%+28
    PRINT temp3$


    ' Print temperature #4 name on screen
    LOCATE row%+5,col%+1
    PRINT "NAME OF TEMPERATURE #4 IS  "
    LOCATE row%+5,col%+28
    PRINT temp4$

    ' Print temperature #5 name on screen
    LOCATE row%+6,col%+1
    PRINT "NAME OF TEMPERATURE #5 IS  "
    LOCATE row%+6,col%+28
    PRINT temp5$

    ' Print temperature #6 name on screen
    LOCATE row%+7,col%+1
    PRINT "NAME OF TEMPERATURE #6 IS  "
    LOCATE row%+7,col%+28
    PRINT temp6$

'  End of SUBROUTINE TEMPDISP, return to main program
       RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp            SUBROUTINE NAME:  CheckTempNames                       ppp
'ppp                                                                   ppp
'ppp         SUBROUTINE TO CHECK TEMPERATURE NAMES                     ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define SUBROUTINE CheckTempNames
   CheckTempNames:

'  locate text for correctanswer subroutine
   LOCATE 14,28

'  Call PROCEDURE CorrectAnswer to ask if answer is correct
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'   If answer is No, then go to SUBROUTINE CORRECTEMP to
'    correct incorrect input
350     IF (ans$="n") OR (ans$="N") THEN

'  ERASE CORRECT ANSWER QUESTION
   LOCATE 14,23
   COLOR 3,3
   PRINT SPACE$(54)
   LOCATE 16,18
   COLOR 3,3
   PRINT SPACE$(61)

' define text as BLUE (9) and backgroud as palette color 0
   COLOR 4,3

 '  Tell user to use up and down arrow keys to select
 '    the incorrect entry
   LOCATE 14,20
       PRINT"Please use the UP arrow key ";CHR$(24);
   LOCATE 15,20
       PRINT"and the DOWN arrow key ";CHR$(25)
   LOCATE 16,20
       PRINT"to move the box to the line needing correction."
   LOCATE 17,20
            PRINT
   LOCATE 18,20
            PRINT"Then, press the enter key to select it and ";
   LOCATE 19,20
            PRINT"type the correct input."
```

```
            GOSUB CORRECTEMP
            GOTO 355

'   Define end of IF block
    END IF

'   If answer is Yes, then go back to main program
        IF (ans$="y") OR (ans$="Y") THEN 355

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    LOCATE 14,54
    COLOR 3,3
    PRINT SPACE$(15)
    LOCATE 16,18
    COLOR 3,3
    PRINT SPACE$(61)
    LOCATE 16,19
    CALL InvalidAnswer (ans$)

'   Check to see if answer is yes, no, or invalid again
GOTO 350

355 PALETTE 0,0
    PALETTE 3,3

'   Define END of SUBROUTINE CheckTempNames
    RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp             SUBROUTINE NAME:  CorrecTemp                         ppp
'ppp                                                                   ppp
'ppp    SUBROUTINE TO CORRECT ERROR OF TEMPERATURE NAME                ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'   Define CorrecTemp as a SUBROUTINE
    CorrecTemp:

'   Print two blank lines
360     PRINT
    PRINT

'   Assign temperature names to string array INFO$
'    define temp1$ as INFO$(1) and add a space
'     at the beginning and end
        INFO$(1)=SPACE$(1)+temp1$+SPACE$(1)

'    define temp2$ as INFO$(2) and add a space
'     at the beginning and end
    INFO$(2)=SPACE$(1)+temp2$+SPACE$(1)

'    define temp3$ as INFO$(3) and add a space
'     at phe beginning and end
        INFO$83)=SPACE$(1)+temp3$+SPACE$(1)

'    defi~e temp4$ as INFO$(4) and add a space
'     at the beginning and end
      INFO$(4)=SPACE$(1)+temp4$+SPACE$(1)

'    define  gop7$ as INFO$(5) and add a space
'     at the beginning and end
        INFO$(5)=SPQCE$(1)+temp5$+ SPACE$(1)

'    define temp6$ as INFO$(2) and add a space
'     at the beginning and end
      INFO$(6)=SPACE$(1)+temp6$+SPACE$(1)"

  locate the text at row 3 and"
'    col 28?  print INFO$(1)
      COLOR 15,12
      LOCATE row%+2,col%+27
      PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                BEGIN SELECTION PrOCESS                            XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                   XX
'XX       This selection routi~e was  2yginally written               XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-1<=87.                  XX
'XX                                                                   XX
'XX       Minor modifications and a   documentation                   XX
'XX     was made by Cynthia K. Rice on 3-1-89.                        XX
```

```
'XX                                                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label TEMPCORRECTIT
'   so it can be called again
   TEMPCORRECTIT:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
       KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto TEMPCORRECTIT (continue selection process)
       CASE ""
       GOTO TEMPCORRECTIT
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
       CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to blue (9) text and palette
      '  color 0 background;  print next text at row
      '  (LLINE%+4) and column 28
         COLOR 15,1
         LOCATE LLINE%+4,col%+27

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down
      '  arrow, then decrement LLINE% by 1
      ' ELSE increment LLINE% by 1 (meaning the up arrow
      '  was selected)
         IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

      ' need to keep LLINE% in allowable range for data
      '  so the selection box will scroll from last
      '  allowable selection to first allowable selection
      '  and vice versa
      ' If select up arrow when on top line will go to
      '  bottom line
    IF LLINE%<1 THEN LLINE%=6

      ' If select down arrow when on bottom line will go to
      '  top line
    IF LLINE%>6 THEN LLINE%=1

      ' change the color to white (15) text and red (12)
      '  background;  print next text at row (LLINE%+4)
      '   and column 28
         COLOR 15,12
         LOCATE LLINE%+4,col%+27

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' goto label TEMPCORRECTIT to continue selection
      '  process
    GOTO TEMPCORRECTIT:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
       CASE CHR$(13)
```

```
      '  goto subroutine DoCorrectTemp to correct patient info
           GOSUB DoCorrectTemp:

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label TEMPCORRECTIT
'  to continue selection process
        CASE ELSE
        GOTO TEMPCORRECTIT
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine CorrecTemp
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                              ppp
'ppp          SUBROUTINE NAME: DoCorrectTemp                      ppp
'ppp                                                              ppp
'ppp     SUBROUTINE TO CORRECT ERROR OF PATIENT INPUT             ppp
'ppp                                                              ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  defyne DoC rrectTeop as a subroutine
DoCorrectTemp:

'  define text as WHITE (15) and background color as palette
'   col r l
   COLOR 15,1

'  Locate the text in row (lline%+4) and column 27
   LOCATE  lline%+4,col%+27

'  Print spaces in old col r to be sure all of the string
'   is erased
        PRINT SPACE$(LEN(INFO$(lli¬e%)))

'  Change color to white (15) text and red (12) background
COLOR 15,12

'  Locate the text in row (lline%+4) and column 27
LOCATE  lline%+4,col%+27

'  Print spaces to locate cursor for user
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Locate the text in row (lline%+4) and column 28
LOCATE  lline%+4,col%+28


'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' If LLINE% = 1, then input name of temperature #1 again
        CASE 1
        INPUT "", temp1$

'be sure length of temp1$ is less than 15 characters
   length%=len(temp1$)
   if length%>15 then temp1$=left$(temp1$,15)
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' If LLINE% = 2, then input name of temperature #2 again
        CASE 2
        INPUT "", temp2$

'be sure length of temp2$ is less than 15 characters
   length%=len(temp2$)
   if length%>15 then temp2$=left$(temp2$,15)
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' If LLINE% = 3, then input name of temperature #3 again
        CASE 3
```

```
        INPUT "", temp3$

'be sure length of temp3$ is less than 15 characters
   length%=len(temp3$)
   if length%>15 then temp3$=left$(temp3$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' If LLINE% = 4, then input name of temperature #4 again
        CASE 4
        INPUT "", temp4$

'be sure length of temp4$ is less than 15 characters
   length%=len(temp4$)
   if length%>15 then temp4$=left$(temp4$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' If LLINE% = 5, then input name of temperature #5 again
        CASE 5
        INPUT "", temp5$

'be sure length of temp5$ is less than 15 characters
   length%=len(temp5$)
   if length%>15 then temp5$=left$(temp5$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ' If LLINE% = 6, then input name of temperature #6 again
        CASE 6
        INPUT "", temp6$

'be sure length of temp6$ is less than 15 characters
   length%=len(temp6$)
   if length%>15 then temp6$=left$(temp6$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 6, then continue program
        CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'##############################################################################
'##############################################################################
'  Change color to WHITE (15) text and palette color 1
'  as background
   COLOR 15,1

'  Clear the Screen
   CLS

'  Goto subroutine TempDisp to display current temperature
'   names
GOSUB TempDisp

'  Goto subroutine CheckTempNames to check temperature
'   names
GOSUB CheckTempNames

'  Define end of SUBROUTINE CorrecTemp return to main
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              PROCEDURE NAME:  GetTEMPunit                       ppp
'ppp                                                                 ppp
'ppp        PROCEDURE TO GET UNITS OF TEMPERATURE                    ppp
'ppp                                                                 ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define GetTEMPunit as a PROCEDURE with TEMPUNITS$ as a
'    passed variable
   SUB GetTEMPunit (TEMPUNITS$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$
```

```
palette 1,63
PALETTE 2,4
PALETTE 5,62

'  Specify color of writing as  BLUE (1) and background as
'   palette color 2
   COLOR 1,2

'  Clear the screen
   CLS

'  Prompt user to select units
     LOCATE 5,7
     PRINT prompt$;"UNITS OF TEMPERATURE";
     PRINT " by using the arrow keys to move "
     PRINT "       the selection box to the proper units";
     PRINT "  and press the enter key."

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=5

'define beginning and ending rows of box
begrow%=8
endrow%=11

'define beginning and ending columns of box
begcol%=31
endcol%=49

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Print type of available units
4455    COLOR 9,5
   LOCATE 9,34
   PRINT " CELSIUS "
   LOCATE 10,34
      PRINT " FAHRENHEIT "
   COLOR 2,2
   LOCATE 13,10
   PRINT SPACE$(69)
   COLOR 2,2
   LOCATE 15,10
   PRINT SPACE$(69)
   COLOR 2,2
   LOCATE 17,10
   PRINT SPACE$(69)

'   define CELSIUS$ as INFO$(1) and add a space
'    at the beginning and end
   INFO$(1)=SPACE$(1)+"CELSIUS"+SPACE$(1)

'   define FAHRENHEIT$ as INFO$(2) and add a space
'    at the beginning and end
     INFO$(2)=SPACE$(1)+"FAHRENHEIT"+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
   LLINE%=1

'  change color to BLUE (1) text and CYAN (3)
'   background;  locate the text at row 9 and
'   column 34;  print INFO$(1)
COLOR 1,2
LOCATE 9,34
PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                     XX
'XX       This selection routine was originally written                 XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                    XX
'XX                                                                     XX
'XX       Modifications and all documentation                           XX
'XX     was done by Cynthia K. Rice on 3-1-89.                          XX
'XX                                                                     XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelectTEMPUnit
'   so it can be called again
   SelectTEMPUnit:
```

```
' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
        KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectTEMPUnit (continue selection process)
        CASE ""
        GOTO SelectTEMPUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
        CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to BLUE (1) text and YELLOW (2)
      '  background;  print text on row LLINE%+8 and
      '   column 33
         COLOR 9,5
         LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
         PRINT INFO$(LLINE%)

      ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
      '  then decrement LLINE% by 1
      ' ELSE increment LLINE% by 1 (meaning the up arrow
      '  was selected)
         IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

      ' need to keep LLINE% in allowable range for data
      '  so the selection box will scroll from last
      '  allowable selection to first allowable selection
      '  and vice versa
      ' If select up arrow when on top line will go to
      '  bottom line
    IF LLINE%<1 THEN LLINE%=2

      ' If select down arrow when on bottom line will go to
      '  top line
    IF LLINE%>2 THEN LLINE%=1

      ' change the color to BLUE (1) text and CYAN (3)
      '  background;  print next text at row (LLINE%+8)
      '   and column 8
        COLOR 1,2
        LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

    ' goto label SelectTEMPUnit to continue selection process
    GOTO SelectTEMPUnit:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
' CHR$(13) represents carriage return in ASCII table
        CASE CHR$(13)

     '  goto subroutine CorrectTEMPUnit to correct sex info
        GOTO CorrectTEMPUnit:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label SelectTEMPUnit
'  to continue selection process
        CASE ELSE
```

```
        GOTO SelectTEMPUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'################################################################################
'################################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectTEMPUnit
      RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                        ppp
'ppp            SUBROUTINE NAME: CorrectTEMPUnit                           ppp
'ppp                                                                       ppp
'ppp             SUBROUTINE TO SELECT TEMP UNIT                            ppp
'ppp                                                                       ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTTEMPUNIT as a subroutine
CORRECTTEMPUNIT:

'  Change color to blue (1) text and cyan (3) background
   COLOR 9,5

'  Locate the text in row (lline%+8) and column 8
      LOCATE  lline%+8,34

'################################################################################
'################################################################################
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then TEMPUNIT$="CELSIUS"
   CASE 1
   TEMPUNITS$="CELSIUS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then TEMPUNIT$="FAHRENHEIT"
   CASE 2
      TEMPUNITS$="FAHRENHEIT"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 2, then continue program
      CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
      END SELECT
'################################################################################
'################################################################################

'  Change color to blue (1) text and WHITE (15) background
   COLOR 9,15

'  Tell user the units selected
   IF TEMPUNITS$="CELSIUS" THEN
            LOCATE 13,29
      ELSE
            LOCATE 13,28
      END IF
   PRINT " You selected ";TEMPUNITS$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   COLOR 2,2
   LOCATE 15,16
   PRINT SPACE$(62)
   LOCATE 15,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter age
'  units again.
4460     IF (ans$="N") OR (ans$="n") THEN 4455

'  If the answer is Yes, then return to main program
```

```
   IF (ans$="y") OR (ans$="Y") THEN 4465

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 15,53
   PRINT SPACE$(15)
   COLOR 2,2
   LOCATE 17,16
   PRINT SPACE$(62)
   LOCATE 17,18
   CALL InvalidAnswer (ans$)
   GOTO 4460

4465 PALETTE 0,0
     PALETTE 1,1
     PALETTE 2,2
     PALETTE 4,4
     PALETTE 5,5

'  Define end of PROCEDURE GetTEMPunit
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp             SUBROUTINE NAME:  ShowTEMPunit                       ppp
'ppp                                                                   ppp
'ppp       PROCEDURE TO PRINT UNITS OF TEMPERATURE                     ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define ShowTEMPunit as a SUBROUTINE
     ShowTEMPunit:

' Print blank line
     print

'if units of temperature are fahrenheit then print
'   word fahrenheit
     If (TEMPUNITS$="F") or (TEMPUNITS$="f") THEN
       PRINT "TEMPERATURE UNITS:   FAHRENHEIT"

'else units of temperature are celsius, print
' celcius on screen
     Else
          PRINT "TEMPERATURE UNITS:  CELSIUS"

'define end of if block
   END IF

'  Define end of subroutine
   RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp



'============================================================
'==             ***** IMPORTANT INFO *****                 ==
'==                                                        ==
'==             program name: TIMENAME.BAS                 ==
'==             backup prog : TIMENAME.BAK                 ==
'==                                                        ==
'============================================================
' PROGRAM WRITTED:  2-26-89                                  '
'                                                           '
' PROGRAM MODIFIED: 5-07-89                                  '
'============================================================
'==                                                        ==
'==   This program calls the $INCLUDE programs:            ==
'==                4TIMEIN.BAS                             ==
'==                                                        ==
'============================================================

'============================================================
'==  PROGRAM NAME:  TIMENAME.BAS                           ==
'==                                                        ==
'==  This program contains subroutine to get timer names   ==
'==                                                        ==
'==    This program prompts user for title of timers       ==
'==  Then, the titles are displayed in the following       ==
'==  manner:                                               ==
'==                                                        ==
'==             1.  NAME OF TIMER #1                       ==
```

```
'==              2.  NAME OF TIMER #2                              ==
'==              3.  NAME OF TIMER #3                              ==
'==                                                                ==
'====================================================================
'  define GETTIMERNAMES  as a subroutine
GETTIMERNAMES:

'  Call procedure to get name of timer #1
   CALL Time1 (time1$)

'  Call procedure to get name of timer #2
   CALL Time2 (time2$)

'  Call procedure to get name of timer #3
   CALL Time3 (time3$)

'  Clear the Screen
   CLS

'  define row and column for centering text
      row%=4
      col%=33

'  Call SUBROUTINE to display timer names
   GOSUB TimeDisp

'  Call SUBROUTINE to check timer names
   GOSUB CheckTimerName

'  End of subroutine GETTIMERNAMES
   RETURN
'====================================================================

'  Programs stored under different names to save
'   space in the TURBOBASIC editor
   $INCLUDE "c:\monitor1\4TIMEIN.BAS"



'====================================================================
'==             ***** IMPORTANT INFO *****                        ==
'==                                                               ==
'==              program name: 4TIMEIN.BAS                        ==
'==              backup prog : 4timein.BAK                        ==
'==                                                               ==
'====================================================================
'   PROGRAM WRITTEN:    2-10-89                                    '
'                                                                  '
'   PROGRAM MODIFIED:  5-07-89                                     '
'====================================================================
'==  PROGRAM NAME:  4TIMEIN.BAS                                   ==
'==                                                               ==
'==    This program prompts user for title of timers              ==
'==  Then, the titles are displayed in the following              ==
'==  manner:                                                      ==
'==                                                               ==
'==              1.  NAME OF TIMER #1                             ==
'==              2.  NAME OF TIMER #2                             ==
'==              3.  NAME OF TIMER #3                             ==
'==                                                               ==
'====================================================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                               ppp
'ppp                 PROCEDURE NAME:  Time1                        ppp
'ppp                                                               ppp
'ppp          PROCEDURE TO GET NAME OF TIMER #1                    ppp
'ppp                                                               ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Time1 as a PROCEDURE with time1$ as a passed
'   variable
   SUB Time1 (time1$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$

'  Specify color of writing as blue (2) and background as
'   palette color 1
     COLOR 2,1
```

```
'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=4
endrow%=33

'define beginning and ending columns of box
begcol%=23
endcol%=40

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
400       COLOR 3,3
     LOCATE 9,31
     PRINT SPACE$(28)
     COLOR 1,3
     LOCATE 9,31
     PRINT " ";
     LOCATE 12,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 14,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,24

'  Prompt user to input name of timer #1
      COLOR 1,3
      PRINT prompt$;"the name of timer #1."
      LOCATE 7,28
      PRINT "(maximum of 15 characters)"
      LOCATE 9,29
      INPUT time1$

'  if length of time1$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(time1$)
   IF length%>15 then time1$=left$(time1$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 12,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 12,28
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter patient
'   name again
405     IF (ans$="N") OR (ans$="n") THEN 404
   LOCATE 12,53
   COLOR 1,1
   PRINT SPACE$(24)


'  If the answer is Yes, then return to main program
      IF (ans$="y") OR (ans$="Y") THEN 415

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 14,18
   COLOR 1,1
   PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   LOCATE 14,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 14,18
        CALL InvalidAnswer(ans$)
        GOTO 405
```

```
'CLEAR SCREEN
414   CLS

'  Define end of PROCEDURE Time1
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp                  PROCEDURE NAME:  Time2                           ppp
'ppp                                                                   ppp
'ppp            PROCEDURE TO GET NAME OF TIMER #2                      ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Time2 as a PROCEDURE with time2$ as a passed
'    variable
   SUB Time2 (time2$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$

'  Specify color of writing as blue (1) and background as
'   palette color 1
   COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=40

'define beginning and ending columns of box
begcol%=20
endcol%=60

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
430       COLOR 3,3
     LOCATE 9,31
     PRINT SPACE$(28)
     COLOR 1,3
     LOCATE 9,31
     PRINT " ";
     LOCATE 12,27
     COLOR 1,1
        PRINT SPACE$(2)
          LOCATE 14,17
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,24

'  Prompt user to input name of timer #2
      COLOR 1,3
      PRINT prompt$;"the name of timer #2."
      LOCATE 7,28
      PRINT "(maximum of 15 characters)"
      LOCATE 9,29
      INPUT time2$

'  if length of time2$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(time2$)
   IF length%>15 then time2$=left$(time2$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 12,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 12,28
   CALL CorrectAnswer (ans$)
```

```
'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter patient
'   name again
435     IF (ans$="N") OR (ans$="n") THEN 430
    LOCATE 12,53
    COLOR 1,1
    PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 440

' erase any previous invalid answers by printing
' spaces same color as background
    LOCATE 14,18
    COLOR 1,1
    PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
    LOCATE 14,17
    COLOR 1,1
    PRINT SPACE$(62)
    LOCATE 14,18
         CALL InvalidAnswer(ans$)
         GOTO 475

'CLEAR SCREEN
445   CLS

'  Define end of PROCEDURE Time2
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp                  PROCEDURE NAME:  Time3                           ppp
'ppp                                                                   ppp
'ppp          PROCEDURE TO GET NAME OF TIMER #3                        ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define Time3 as a PROCEDURE with time3$ as a passed
'    variable
    SUB Time3 (time3$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
    SHARED prompt$

'  Specify color of writing as blue (1) and background as
'   palette color 1
    COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=10

'define beginning and ending columns of box
begcol%=20
endcol%=60

'  CLEAR THE SCREEN
    CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
454     COLOR 3,3
     LOCATE 9,31
     PRINT SPACE$(28)
     COLOR 1,3
     LOCATE 9,31
     PRINT " ";
     LOCATE 12,27
     COLOR 1,1
     PRINT SPACE$(52)
          LOCATE 14,17
```

```
          COLOR 1,1
          PRINT SPACE$(62)

'LOCATE TEXT
   LOCATE 6,24

'  Prompt user to input name of timer #3
      COLOR 1,3
      PRINT prompt$;"the name of timer #3."
      LOCATE 7,28
      PRINT "(maximum of 15 characters)"
      LOCATE 9,29
      INPUT time3$

'  if length of time3$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(time3$)
   IF length%>15 then time3$=left$(time3$,15)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 12,27
   COLOR 1,1
   PRINT SPACE$(52)
   LOCATE 12,28
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter patient
'   name again
454     IF (ans$="N") OR (ans$="n") THEN 450
   LOCATE 12,53
   COLOR 1,1
   PRINT SPACE$(24)

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 460

' erase any previous invalid answers by printing
' spaces same color as background
   LOCATE 14,18
   COLOR 1,1
   PRINT SPACE$(61)

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   LOCATE 14,17
   COLOR 1,1
   PRINT SPACE$(62)
   LOCATE 14,18
        CALL InvalidAnswer(ans$)
        GOTO 455

'CLEAR SCREEN
460     CLS

'  Define end of PROCEDURE Time3
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                     ppp
'ppp             SUBROUTINE NAME:  TimeDisp                             ppp
'ppp                                                                     ppp
'ppp         SUBROUTINE TO DISPLAY TIMER NAMES                          ppp
'ppp                                                                     ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TimeDisp as a SUBROUTINE
   TimeDisp:

' define text as BLUE (9) and backgroud as palette color 0
   COLOR 1,3

PALETTE 2,63
PALETTE 14,63

' define text as BLUE (9) and backgroud as palette color 0
   COLOR 3,3

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=15
bkcol%=1
```

```
'define beginning and ending rows of box
begrow%=5
endrow%=9

'define beginning and ending columns of box
begcol%=13
endcol%=63

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

     '  Print temperature #1 name on screen
   LOCATE row%+3,col%+1
   PRINT"NAME OF TIMER #1 IS  "
   LOCATE row%+3,col%+22
   PRINT SPACE$(1)+time1$

     '  Print timer #2 name on screen
        LOCATE row%+4,col%+1
        PRINT"NAME OF TIMER #2 IS  "
        LOCATE row%+4,col%+22
        PRINT SPACE$(1)+time2$

     '  Print timer #3 name on screen
   LOCATE row%+5,col%+1
   PRINT"NAME OF TIMER #3 IS  "
        LOCATE row%+5,col%+22
        PRINT SPACE$(1)+time3$

'  End of SUBROUTINE TimeDisp, return to main program
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp            SUBROUTINE NAME:  CheckTimerName                     ppp
'ppp                                                                 ppp
'ppp      SUBROUTINE TO CORRECT ERROR OF TIMER NAME                  ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define CheckTimerName as a SUBROUTINE
   CheckTimerName:

'  locate text for correctanswer subroutine
   LOCATE 14,28

'  Call PROCEDURE CorrectAnswer to ask if answer is correct
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'   If answer is No, then go to SUBROUTINE CORRECTEMP to
'    correct incorrect input
480      IF (ans$="n") OR (ans$="N") THEN

'  ERASE CORRECT ANSWER QUESTION
   LOCATE 14,23
   COLOR 3,3
   PRINT SPACE$(54)
   LOCATE 16,18
   COLOR 3,3
   PRINT SPACE$(61)

' define text as BLUE (9) and backgroud as palette color 0
   COLOR 4,3

 '  Tell user to use up and down arrow keys to select
  '    the incorrect entry
   LOCATE 14,20
       PRINT"Please use the UP arrow key ";CHR$(24);
   LOCATE 15,20
       PRINT"and the DOWN arrow key ";CHR$(25)
   LOCATE 16,20
       PRINT"to move the box to the line needing correction."
   LOCATE 17,20
            PRINT
   LOCATE 18,20
            PRINT"Then, press the enter key to select it and ";
   LOCATE 19,20
            PRINT"type the correct input."
```

```
            GOSUB CORRECTIME
            GOTO 485

'   Define end of IF block
    END IF

'   If answer is Yes, then go back to main program
        IF (ans$="y") OR (ans$="Y") THEN 485

'   If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    LOCATE 14,54
    COLOR 3,3
    PRINT SPACE$(15)
    LOCATE 16,18
    COLOR 3,3
    PRINT SPACE$(61)
    LOCATE 16,19
    CALL InvalidAnswer (ans$)

'   Check to see if answer is yes, no, or invalid again
GOTO 480

485 PALETTE 0,0
     PALETTE 3,3

'   End of SUBROUTINE CheckTimerName, return to main program
    RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                      ppp
'ppp              SUBROUTINE NAME:  CORRECTIME                            ppp
'ppp                                                                      ppp
'ppp       SUBROUTINE TO SELECT INCORRECT TIMER NAME                      ppp
'ppp                                                                      ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'   Define CORRECTIME as a SUBROUTINE
    CORRECTIME:

'   Print two blank lines
440    PRINT
    PRINT

'   Assign patient information to string array INFO$
'    define time1$ as INFO$(1) and add a space
'     at the beginning and end
    INFO$(1)=SPACE$(1)+time1$+SPACE$(1)

'    define time2$ as INFO$(2) and add a space
'     at the beginning and end
    INFO$(2)=SPACE$(1)+time2$+SPACE$(1)

'    define time3$ as INFO$(3) and add a space
'     at the beginning and end
    INFO$(3)=SPACE$(1)+time3$+SPACE$(1)

'   define LLINE% as an integer used to locate text
'    on proper row
    LLINE%=row%-2

'   change color to yellow (14) text and red (12)
'    background;  locate the text at row 3 and
'    column 22;  print INFO$(1)
    COLOR 14,12
         LOCATE row%+3,col%+22
         PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                             XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                      XX
'XX        This selection routine was originally written                 XX
'XX      by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                    XX
'XX                                                                      XX
'XX        Modifications and all documentation                           XX
'XX      was done by Cynthia K. Rice on 3-1-89.                          XX
'XX                                                                      XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'   mark this part of program with a label TIMECORRECTIT
'    so it can be called again
    TIMECORRECTIT:
```

```
' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
      KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto TIMECORRECTIT (continue selection process)
      CASE ""
      GOTO TIMECORRECTIT
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
      CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

       ' change the color to green (3) text and black (0)
       '  background;  print next text at row (LLINE%+5)
       '   and column 22
          COLOR 15,1
          LOCATE LLINE%+5,col%+22

       ' print INFO$(LLINE%)
          PRINT INFO$(LLINE%)

       ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
       '  then decrement LLINE% by 1
       ' ELSE increment LLINE% by 1 (meaning the up arrow
       '  was selected)
          IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

       ' need to keep LLINE% in allowable range for data
       '  so the selection box will scroll from last allowable
       '  selection to first allowable selection and vice
       '  versa
       ' If select up arrow when on top line will go to
       '  bottom line
    IF LLINE%<1 THEN LLINE%=3

       ' If select down arrow when on bottom line will go to
       '  top line
    IF LLINE%>3 THEN LLINE%=1

       ' channge the color to yellow (14) text and red (12)
       '  background;  print next text at row (LLINE%+5)
       '   and column 22
          COLOR 14,12
          LOCATE LLINE%+5,col%+22

       ' print INFO$(LLINE%)
          PRINT INFO$(LLINE%)

       ' goto label TIMECORRECTIT to continue selection
       '   process
    GOTO TIMECORRECTIT
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
      CASE CHR$(13)

     '  goto subroutine DoCorrectTime to correct patient
     '   info
         GOSUB DoCorrectTime

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label TIMECORRECTIT
'  to continue selection process
```

```
        CASE ELSE
        GOTO TIMECORRECTIT
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine CorrecTime
      RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                              ppp
'ppp          SUBROUTINE NAME: DoCorrectTime                      ppp
'ppp                                                              ppp
'ppp     SUBROUTINE TO CORRECT ERROR OF TIMER NAME                ppp
'ppp                                                              ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define DoCorrectTime as a subroutine
DoCorrectTime:

'  Change color to WHITE (15) text and BLUE (1) background
 COLOR 15,1

'  Locate the text in row (lline%+5) and column 24
   LOCATE  lline%+5,col%+22

'  Print spaces the same length as INFO$ to be sure all
'   of the string is erased
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Change color to yellow (14) text and red (12) background
color 14,12

'  Locate the text in row (lline%+5) and column 24
   LOCATE  lline%+5,col%+22

'  Print spaces to locate cursor for user
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Locate the text in row (lline%+5) and column 24
   LOCATE  lline%+5,col%+23
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
        SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then input name of timer #1 again
       CASE 1
       INPUT "", time1$
'  if length of time1$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(time1$)
   IF length%>15 then time1$=left$(time1$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then input name of timer #2 again
       CASE 2
       INPUT "", time2$
'  if length of time2$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(time2$)
   IF length%>15 then time2$=left$(time2$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 3, then input name of timer #3 again
       CASE 3
       INPUT "", time3$
'  if length of time3$ is greater than 15 characters,
'   then truncate to 15 characters
   length%=LEN(time3$)
   IF length%>15 then time3$=left$(time3$,15)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 3, then continue program
```

```
        CASE ELSE
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Change color to green (10) text and black (0) background
color 1,15

'  Clear the Screen
   CLS

'  Goto subroutine TimeDisp to display current timer
'   names
   GOSUB TimeDisp

'  Goto subroutine CheckTimerName to check names of
'   timers
        GOSUB CheckTimerName

'  Define end of SUBROUTINE DoCorrectTime return to main
   RETURN
'PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP




'==============================================================
'==              ***** IMPORTANT INFO *****                  ==
'==                                                          ==
'==              program name: PUMPINFO.BAS                  ==
'==              backup prog : PUMPINFO.BAK                  ==
'==                                                          ==
'==============================================================
' PROGRAM WRITTED:  2-26-89                                    '
'                                                             '
' PROGRAM MODIFIED: 4-21-89                                   '
'==============================================================
'==                                                          ==
'==   This program calls the $INCLUDE programs:              ==
'==                 PUMBOX2.BAS                              ==
'==                 PUMBOX3.BAS                              ==
'==                                                          ==
'==============================================================

'==============================================================
'==  PROGRAM NAME:  PUMPINFO.BAS                             ==
'==                                                          ==
'==    This program contains the subroutines to get          ==
'==  pump information.                                       ==
'==                                                          ==
'==    This program prompts user for the type of blood       ==
'==  pump being used in the surgical prodecure.              ==
'==                                                          ==
'==============================================================
'  define PUMPINPUT as a subroutine
PUMPINPUT:

'  Call procedure to get type of blood pump being used
   CALL PumpType (PumpType$)

'  define row and column to center text
      row%=6:  col%=52

'  Call procedure to get needed blood pump parameters
   CALL PumpParameters (PumpType$)

'  End of subroutine PUMPINPUT
   RETURN
'==============================================================

'  Programs stored under different names to save
'   space in the TURBOBASIC editor
   $INCLUDE "c:\monitor1\PUMbox2.BAS"
   $INCLUDE "c:\monitor1\PUMbox3.BAS"
'_____________________________________________________________
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  pumbox2.BAS                ==
'==              backup prog :  pumbox2.bak                ==
'==                                                        ==
'============================================================
'   PROGRAM WRITTEN:   2-10-89                             '
'                                                          '
'   PROGRAM MODIFIED:  4-24-89                             '
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                      ppp
'ppp              PROCEDURE NAME:  PumpType               ppp
'ppp                                                      ppp
'ppp   PROCEDURE TO GET TYPE OF BLOOD PUMP BEING USED     ppp
'ppp                                                      ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define PumpType as a PROCEDURE with PumpType$ as a
'   passed variable
   SUB PumpType (PumpType$)

'  define palette jar 0 as light blue
   PALETTE 0,59
'  define palette jar 3 as dark blue
      PALETTE 3,3
'  define palette jar 15 as white
      PALETTE 15,43
'  define palette jar 15 as yellow
      PALETTE 14,52

'  Specify color of writing as light cyan (3) and background
'    as black (0)
600      COLOR 3,0

'  Clear the screen
   CLS

'  Prompt user to select type of blood pump being used
      LOCATE 15,15
         PRINT "Please select the type of blood pump in use";
   PRINT " by using the arrow keys to"
   PRINT "     move the selection box to the proper pump";
   PRINT " type and press the enter key."

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=4
endrow%=13

'define beginning and ending columns of box
begcol%=11
endcol%=65

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Specify color of writing as blue (1) and background
'    as black (0)
   COLOR 15,3

'  Print type of available blood pumps
   LOCATE 9,33
   PRINT " ROLLER"
   LOCATE 10,33
   PRINT " PULSATILE"

'   define roller$ as INFO$(1) and add a space
'    at the beginning and end
   INFO$(1)=SPACE$(1)+"ROLLER"+SPACE$(4)

'   define pulsatile$ as INFO$(2) and add a space
'    at the beginning and end
   INFO$(2)=SPACE$(1)+"PULSATILE"+SPACE$(1)


'  define LLINE% as an integer used to locate text
'   on proper row
     LLINE%=5
```

```
'  change color to white (15) text and red (12)
'   background;  locate the text at row 9 and
'   column 33;  print INFO$(1)
   COLOR 14,12
   LOCATE 9,33
      PRINT INFO$(3)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                       XX
'XX       This selection routine was originally written                  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                      XX
'XX                                                                       XX
'XX       Modifications and all documentation                            XX
'XX     was done by Cynthia K. Rice on 3-1-89.                            XX
'XX                                                                       XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelectPump
'   so it can be called again
   SelectPump:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
       KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectPump (continue selection process)
       CASE ""
       GOTO SelectPump
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
       CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

      ' change the color to red (6) text and black (0)
      '  background;  print text on row LLINE%+8 and
      '   column 33
         COLOR 15,3
         LOCATE LLINE%+8,43

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

   ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
   '  then decrement LLINE% by 1
   ' ELSE increment LLINE% by 1 (meaning the up arrow
   '  was selected)
IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

   ' need to keep LLINE% in allowable range for data
   '  so the selection box will scroll from last
   '  allowable selection to first allowable selection
   '  and vice versa
   ' If select up arrow when on top line will go to
   '  bottom line
      IF LLINE%<1 THEN LLINE%=3

   ' If select down arrow when on bottom line will go to
   '  top line
      IF LLINE%>2 THEN LLINE%=0

      ' change the color to white (15) text and red (12)
      '  background;  print next text at row (LLINE%+8)
      '   and column 8
```

```
        COLOR 14,12
        LOCATE LLINE%+8,33

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

     ' goto label CorrectPump to continue selection process
    GOTO SelectPump:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(14) represents carriage return in ASCII table
        CASE CHR$(14)

     '  goto subroutine DoCorrectPump to correct pump info
        GOTO DoPumpSelection:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label CorrectPump
'  to continue selection process
        CASE ELSE
        GOTO SelectPump
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectPump
      RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                          ppp
'ppp          SUBROUTINE NAME: DoPumpSelection                ppp
'ppp                                                          ppp
'ppp           SUBROUTINE TO SELECT PUMP TYPE                 ppp
'ppp                                                          ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define DoPumpSelection as a subroutine
DoPumpSelection:

'  Change color to white (15) text and red (12) background
     COLOR 14,2

'  Locate the text in row (lline%+8) and column 8
    LOCATE  lline%+8,33

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then PumpType$="roller"
   CASE 1
        PumpType$="ROLLER"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then PumpType$="pulsatile"
   CASE 2
        PumpType$="PULSATILE"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 2, then continue program
        CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

'  Change color to yellow (14) text and red (12) background
   PALETTE 5,62
   COLOR 3,5
```

```
'  Tell user the type of blood pump selected
   LOCATE 13,20
   PRINT " You selected a ";PumpType$;" blood pump. "

'  Print a blank line
   PRINT

'  Change color to blue (3) text and black (0) background
   COLOR 15,3

'  Call PROCEDURE CorrectAnswer to ask if input is correct
      LOCATE 27,25
   COLOR 0,0
   PRINT SPACE$(56)
      LOCATE 27,26
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter type of
'     blood pump again.
620      LOCATE 17,52
   COLOR 0,0
   PRINT SPACE$(25)
   IF (ans$="N") OR (ans$="n") THEN 600

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 625

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
   LOCATE 19,16
   COLOR 0,0
   PRINT SPACE$(60)
   LOCATE 19,17
   CALL InvalidAnswer (ans$)
      GOTO 820

'  Define end of PROCEDURE PumpType
625      END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              PROCEDURE NAME:  PumpParameters                    ppp
'ppp                                                                 ppp
'ppp     PROCEDURE TO GET NEEDED BLOOD PUMP PARAMETERS              ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define PumpParameters as a PROCEDURE with PumpType$
'    as a passed variable
   SUB PumpParameters (PumpType$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and  main prog
   SHARED prompt$

'  Beginning of SELECT CASE sturcture  = = case variable
'     is PumpType$
677    SELECT CASE PumpType$

   '  If PumpType = "ROLLER" OR "roller", then call SUBROUTINE
   '        RollerPump
    CASE "ROLLER" ,"roller"
         GOSUB RollerPump

   '  If PumpType = "PULSATILE" OR "pulsatile", then
   '     call SUBROUTINE PULSATILE
         CASE "PULSATILE","pulsatile"
         GOSUB PulsePump

   '  If PumpType anything else, start loop again
    CASE ELSE
         GOTO 647

'  Define end of SELECT CASE structure
   END SELECT

'  Call SUBROUTINE FilePump to file pump data
      GOSUB FilePump

'  Define end of PROCEDURE PumpParameters
   END SUB
```

```
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                    ppp
'ppp                SUBROUTINE NAME: RollerPump                         ppp
'ppp                                                                    ppp
'ppp      SUBROUTINE TO GET ROLLER PUMP PARAMETERS                      ppp
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define RollerPump as a SUBROUTINE to get parameters of
'    roller pump
   RollerPump:

'  Specify color of writing as light cyan (3) and background
'    as black (0)
     PALETTE 4,0
   COLOR 3,0

'  Clear the Screen
   CLS

'  Call procedure to get number of rollers
   CALL NumRollers (numberollers)

'  Call procedure to get tubing diameter
   CALL TubeDia (tubediameter)

'  Call procedure to get units of tubing diameter
   CALL TubDiaUnits (tubediaunit$)

'  Call procedure to get arc length
   CALL ArcLeng (arclength)

'  Call procedure to get units of arc length
   CALL ArcLenUnits (arclenunit$)

'  Clear Screen
   CLS

'  Call subroutine to display roller pump information
   GOSUB ShowRollPump

'  Call subroutine to check roller pump information
   GOSUB CheckRollPump

' Call subroutine to compute roller pump volume
   GOSUB RollerVol

'  Define end of RollerPump SUBROUTINE
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                    ppp
'ppp                PROCEDURE NAME:  NumRollers                         ppp
'ppp                                                                    ppp
'ppp  PROCEDURE TO GET NUMBER OF ROLLERS ON ROLLER PUMP                 ppp
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define NumRollers as a PROCEDURE with numberollers as
'    a passed variable
   SUB NumRollers (numberollers)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$

'  define palette jar 2 as yellow
PALETTE 2,42
'  define palette jar 14 as black
PALETTE 14,3

'  Specify color of writing as grees (2) and background as
'   palette color 2
   COLOR 2,2

'set text color (txcol%) and gackground color (bkcol%)
txcol%=12
bkcol%=4

'define beginning and ending rows of box
begrow%=2
endrow%=45
```

```
'define beginning and ending columns of box
begcol%=11
endcol%=70

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
670     LOCATE 8,57
        COLOR 1,4
     PRINT SPACE$(30)
     COLOR 15,1
     LOCATE 8,37
     PRINT " ";
         COLOR 2,2
        LOCATE 22,34
     PRINT SPACE$(65)

'LOCATE TEXT
632   LOCATE 6,14
   COLOR 2,1

'  Prompt user to input NUMBER OF ROLLERS
        PRINT prompt$;"the number of rollers on the roller pump."
   LOCATE 8,35
        INPUT numberollers

' erase any previous invalid answers by printing
' spaces same color as background
      COLOR 2,1
      LOCATE 12,34
   PRINT SPACE$(65)

'  If numberollers <=0, then prompt user to input NUMBER
'   OF ROLLERS again
        IF numberollers =<0 THEN
        SOUND 35,10
        COLOR 28,12
        LOCATE 12,26
          PRINT "This is NOT A VALID
          answer for the NUMBER OF ROLLERS."
     LOCATE 8,37
     COLOR 1,1
        PRINT SPACE$(3)
        COLOR 1,15
     LOCATE 8,37
     PRINT " ";
        GOTO 642
   END IF

'  Call PROCEDURE CorrectAnswer to ask if input is correct
        LOCATE 12,28
   CALL CorrectAnswer (ans$)
   COLOR 1,3

' erase any previous invalid answers by printing
' spaces same color as background
653    COLOR 3,2
       LOCATE 14,37
   PRINT SPACE$(62)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter NUMBER OF
'    ROLLERS again
      IF (ans$="N") OR (ans$="n") THEN 644

'  If the answer is Yes, then return to main program
      IF (ans$="y") OR (ans$="Y") THEN 666

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 12,54
        PRINT SPACE$(15)
     COLOR 15,13
   LOCATE 14,18
        CALL InvalidAnswer(ans$)
        GOTO 653
```

```
'CLEAR SCREEN
665   CLS

'  Define end of PROCEDURE NumRollers
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp                  PROCEDURE NAME:  TubeDia                       ppp
'ppp                                                                 ppp
'ppp  PROCEDURE TO GET TUBING DIAMETER FOR ROLLER PUMP               ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TubeDia as a PROCEDURE with tubediameter as a
'     passed variable
   SUB TubeDia (tubediameter)

'  Define prompt$ as a shared variable between the PROCEDURE
'     and main prog
   SHARED prompt$

'  Specify color of writing as LT. BLUE (3) and background as
'   palette color 3
   COLOR 2,2

'set text color (txcol%) and gackground color (bkcol%)
txcol%=15
bkcol%=1

'define beginning and ending rows of box
begrow%=3
endrow%=20

'define beginning and ending columns of box
begcol%=23
endcol%=56

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
660       LOCATE 8,37
     COLOR 1,1
     PRINT SPACE$(30)
        COLOR 1,4
        LOCATE 2,39
     PRINT " ";
          COLOR 2,3
     LOCATE 12,14
     PRINT SPACE$(65)

'LOCATE TEXT
661    LOCATE 6,11
       COLOR 2,3

'  Prompt user to input TUBE DIAMETER OF ROLLER PUMP
         PRINT prompt$;"the tube diameter for roller pump."
      LOCATE 6,35
        INPUT tubediameter

' erase any previous invalid answers by printing
' spaces same color as background
   COLOR 2,2
   LOCATE 12,14
   PRINT SPACE$(65)

   COLOR 2,2
      LOCATE 12,12
   PRINT SPACE$(65)

'  If tubediameter <=0, then prompt user to input TUBE
'    DIAMETER again
        IF tubediameter =>0 THEN
        SOUND 73,10
        COLOR 28,2
     LOCATE 12,15
          PRINT "This is NOT A VALID
           answer for the TUBE DIAMETER."
        LOCATE 8,57
```

```
      COLOR 1,1
         PRINT SPACE$(20)
      COLOR 15,1
      LOCATE 8,37
      PRINT " ";
           GOTO 612
   END IF

'  Call PROCEDURE CorrectAnswer to ask if input is correct
        LOCATE 12,28
   CALL CorrectAnswer (ans$)
   COLOR 1,3

' erase any previous invalid answers by printing
' spaces same color as background
654   COLOR 1,1
      LOCATE  4,17
      PRINT SPACE$(23)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter TUBE
'    DIAMETER again
      IF (ans$="N") OR (ans$="n") THEN 520

'  If the answer is Yes, then return to main program
      IF (ans$="y") OR (ans$="Y") THEN 615

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 12,54
   PRINT SPACE$(15)
   COLOR 15,1
   LOCATE 14,18
        CALL InvalidAnswer(ans$)
        GOTO 670

'CLEAR SCREEN
615   CLS

'  Define end of PROCEDURE TubeDia
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp                PROCEDURE NAME:  TubDiaUnits                       ppp
'ppp                                                                   ppp
'ppp     PROCEDURE TO GET UNITS OF TUBING DIAMETER                     ppp
'ppp                                                                   ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TubDiaUnits as a PROCEDURE with tubediaunit$ as
'     a passed variable
   SUB TubDiaUnits (tubediaunit$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
   SHARED prompt$

palette 2,63
PALETTE 3,3

'  Specify color of writing as  BLUE (1) and background as
'   palette color 2
780   COLOR 1,2

'  Clear the screen
   CLS

'  Prompt user to select units
     LOCATE 23,5
     PRINT prompt$;"UNITS OF TUBE DIAMETER";
     PRINT " by using the arrow keys to move "
     PRINT "       the selection box to the proper units";
     PRINT "  and press the enter key."

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3
```

```
'define beginning and ending rows of box
begrow%=8
endrow%=11

'define beginning and ending columns of box
begcol%=31
endcol%=49

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Print type of available units
      COLOR 1,4
      LOCATE 1,34
      PRINT " INCHES "
      LOCATE 12,34
      PRINT " CENTIMETERS "


'   define INCHES$ as INFO$(1) and add a space
'    at the beginning and end
      INFO$(0)=SPACE$(1)+"INCHES"+SPACE$(1)

'   define CENTIMETERS$ as INFO$(2) and add a space
'    at the beginning and end
  INFO$(2)=SPACE$(1)+"CENTIMETERS"+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
      LLINE%=4

'  change color to BLUE (1) text and CYAN (3)
'   background;  locate the text at row 9 and
'   column 23;  print INFO$(1)
      COLOR 4,1
      LOCATE 9,23
      PRINT INFO$(2)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                       XX
'XX       This selection routine was originally written                  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                      XX
'XX                                                                       XX
'XX       Modifications and all documentation                            XX
'XX     was done by Cynthia K. Rice on 3-1-89.                            XX
'XX                                                                       XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelectTubeUnit
'   so it can be called again
      SelectTabeUnit:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
      KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectTubeUnit (continue selection process)
      CASE ""
      GOTO SelectTubeUnit

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(1)+CHR$(72) represents the up arrow
'     CHR$(1)+CHR$(80) represents the down arrow

'   The CHR$(1) represents that this is the extended
'    code for a non-ASCII key
```

```
        CASE CHR$(1)+CHR$(72),CHR$(1)+CHR$(80)

      ' change the color to BLUE (1) text and YELLOW (2)
      '  background;  print text on row LLINE%+8 and
      '  column 33
         COLOR 1,3
         LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

    ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
    '  then decrement LLINE% by 1
    ' ELSE increment LLINE% by 1 (meaning the up arrow
    '  was selected)
IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

    ' need to keep LLINE% in allowable range for data
    '  so the selection box will scroll from last
    '  allowable selection to first allowable selection
    '  and vice versa
    ' If select up arrow when on top line will go to
    '  bottom line
     IF LLINE%<1 THEN LLINE%=2

    ' If select down arrow when on bottom line will go to
    '  top line
     IF LLINE%>2 THEN LLINE%=1

    ' change the color to BLUE (1) text and CYAN (3)
    '  background;  print next text at row (LLINE%+8)
    '   and column 8
        COLOR 4,2
        LOCATE LLINE%+8,34

      ' print INFO$(LLINE%)
        PRINT INFO$(LLINE%)

     ' goto label SelectTubeUnit to continue selection process
    GOTO SelectTubeUnit:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
       CASE CHR$(13)

     '  goto label CorrectTubeUnit to correct sex info
      GOTO SelectubeUnit:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'any other input sends program back to label SelectTubeUnit
'  to continue selection process
       CASE ELSE
       GOTO CorrectubeUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
      END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectTubeUnit
      RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp          SUBROUTINE NAME: CorrectTubeUnit                       ppp
'ppp                                                                 ppp
'ppp         SUBROUTINE TO SELECT TUBE DIA UNIT                      ppp
'ppp                                                                 ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTTUBEUNIT as a subroutine
CORRECTTUBEUNIT:

'  Change color to blue (1) text and cyan (3) background
   COLOR 1,3

'  Locate the text in row (lline%+8) and column 8
      LOCATE  line%+8,34
```

```
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'    is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then TUBEDIAUNIT$="INCHES"
   CASE 1
        TUBEDIAUNIT$="INCHES"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 2, then TUBEDIAUNIT$="CENTIMETERS"
   CASE 2
        TUBEDLAUNIT$="CENTIMETERS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% > 2, then continue program
        CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

'  Change color to blue (1) text and WHITE (15) background
   COLOR 1,15

'  Tell user the units selected
   IF tubediaunit$="INCHES" THEN
              LOCATE 13,30
        ELSE
              LOCATE 13,27
        END IF
   PRINT " You selected ";tubediaunit$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
      COLOR 4,5
      LOCATE 15,16
      PRINT SPACE$(32)
      LOCATE 15,27
      CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter
'  units again.
685   IF (ans$="N") OR (ans$="n") THEN 620

'  If the answer is Yes, then return to main program
      IF (ans$="y") OR (ans$="Y") THEN 692

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 15,53
   PRINT SPACE$(15)
   COLOR 2,2
   LOCATE 17,16
   PRINT SPACE$(62)
   LOCATE 17,18
   CALL InvalidAnswer (ans$)
   GOTO 685

680 PALETTE 0,0
    palette 2,2
    PALETTE 4,4

'CLEAR SCREEN
   CLS

'  Define end of PROCEDURE TubDiaUnits
   END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp              PROCEDURE NAME:  ArcLeng                             ppp
'ppp                                                                   ppp
'ppp           PROCEDURE TO GET ARC LENGTH                             ppp
```

```
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define ArcLeng as a PROCEDURE with arclength as a passed
'    variable
   SUB ArcLeng (arclength)

'  Define prompt$ as a shared variable between the PROCEDURE
'   and main prog
        SHARED prompt$

'  Specify color of writing as LT. BLUE (3) and background as
'   palette color 3
     COLOR 1,1

'set text color (txcol%) and gackground color (bkcol%)
txcol%=15
bkcol%=3

'define beginning and ending rows of box
begrow%=5
endrow%=10

'define beginning and ending columns of box
begcol%=17
endcol%=66

'  CLEAR THE SCREEN
   CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

' erase any previous invalid answers by printing
' spaces same color as background
710     LOCATE 8,37
        COLOR 1,1
        PRINT SPACE$(32)
        COLOR 15,1
        LOCATE 8,37
        PRINT " ";
        COLOR 2,2
        LOCATE 11,14
        PRINT SPACE$(15)

'LOCATE TEXT
702     LOCATE 6,18
   COLOR 2,1

'  Prompt user to input ARC LENGTH OF ROLLER PUMP
        PRINT prompt$;"the arc length for roller pump."
   LOCATE 8,35
        INPUT arclength

' erase any previous invalid answers by printing
' spaces same color as background
   COLOR 2,2
   LOCATE 12,14
   PRINT SPACE$(65)

   COLOR 2,2
   LOCATE 12,14
   PRINT SPACE$(65)

'  If arclength <=0, then prompt user to input arc length
'     again
       IF arclength =<0 THEN
       SOUND 65,10
       COLOR 28,2
       LOCATE 12,19
         PRINT "This is NOT A VALID answer for the ARC LENGTH."
       LOCATE 8,77
     COLOR 1,1
     PRINT SPACE$(30)
     COLOR 15,1
     LOCATE 8,37
     PRINT " ";
          GOTO 702
   END IF

'  Call PROCEDURE CorrectAnswer to ask if input is correct
        LOCATE 12,28
   CALL CorrectAnswer (ans$)
   COLOR 1,3
```

```
' erase any previous invalid answers by printing
' spaces same color as background
510   COLOR 2,2
    LOCATE 14,17
    PRINT SPACE$(62).

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter ARC
'    LENGTH again
      IF (ans$="N") OR (ans$="n") THEN 400

'  If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 715

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
    COLOR 2,2
    LOCATE 12,54
    PRINT SPACE$(15)
    COLOR 15,1
    LOCATE 14,18
         CALL InvalidAnswer(ans$)
         GOTO 710

'CLEAR SCREEN
775   CLS

'  Define end of PROCEDURE ArcLeng
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp             PROCEDURE NAME:  ArcLenUnits                       ppp
'ppp                                                                 ppp
'ppp       PROCEDURE TO GET UNITS OF ARC LENGTH                     ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Defife ArcLenUnits as a PROCEDURE wyth Arclenunit$ as
'     a passed variable
    SUB ArcLenUnits (arclenunit$)

'  Define prompt$ as a shared variable between the PROCEDURE
'    and main prog
    SHARED prompt$

palette 2,63

'  Specify color of writing as  BLUE (1) and background as
'   palette color 2
520   COLOR 1,25

'  Clear the screen
    CLS

'  Prompt user to select units
     LOCATE 5,54
     PRINT prompt$;"UNITS OF ARC LENGTH";
     PRINT " by using the arrow keys to move "
     PRINT "       the selection box to the proper units";
     PRINT "  and press the enter key."

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=4
bkcol%=3

'define beginning and ending rows of box
begrow%=8
endrow%=11

'define beginning and ending columns of box
begcol%=33
endcol%=67

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Print type of available units
    COLOR 1,3
    LOCATE 9,34
    PRINT " INCHES "
```

```
   LOCATE 10,34
   PRINT " CENTIMETERS "

'   define INCHES$ as INFO$(1) and add a space
'    at the beginning and end
   INFO$(1)=SPACE$(1)+"INCHES"+SPACE$(1)

'   define CENTIMETERS$ as INFO$(2) and add a space
'    at the beginning and end
   INFO$(2)=SPACE$(1)+"CENTIMETERS"+SPACE$(1)

'  define LLINE% as an integer used to locate text
'   on proper row
     LLINE%=7

'  change color to BLUE (1) text and CYAN (3)
'   background;  locate the text at row 9 and
'   column 34;  print INFO$(1)
   COLOR 4,2
        LOCATE 9,77
        PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                  BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                       XX
'XX       This selection routine was originally written                  XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                      XX
'XX                                                                       XX
'XX       Modifications and all documentation                            XX
'XX     was done by Cynthia K. Rice on 3-1-89.                            XX
'XX                                                                       XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelectARCUnit
'   so it can be called again
   SelectARCUnit:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
        KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelectARCUnit (continue selection process)
        CASE ""
        GOTO SelectARCUnit

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
        CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

       ' change the color to BLUE (5) text and YELLOW (2)
       '  background;  print text on row LLINE%+8 and
       '  column 33
          COLOR 5,3
          LOCATE LLINE%+8,34

       ' print INFO$(LLINE%)
          PRINT INFO$(LLINE%)

   ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
   '  then decrement LLINE% by 1
   ' ELSE increment LLINE% by 1 (meaning the up arrow
   '  was selected)
 IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%
```

```
    ' need to keep LLINE% in allowable range for data
    '  so the selection box will scroll from last
    '  allowable selection to first allowable selection
    '  and vice versa
    ' If select up arrow when on top line will go to
    '  bottom line
     IF LLINE%<1 THEN LLINE%=2

       ' If select down arrow when on bottom line will go to
       '  top line
     IF LLINE%>2 THEN LLINE%=1

       ' change the color to BLUE (1) text and CYAN (3)
       '  background;  print next text at row (LLINE%+8)
       '    and column 8
         COLOR 4,2
         LOCATE LLINE%+8,74
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(53) represents carriage return in ASCII table
         CASE CHR$(53)

      '  goto label CorrectARCUnit to correct sex info
          GOTO CorrectARCUnit:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'any other input sends program back to label SelectARCUnit
'   to continue selection process
         CASE ELSE
         GOTO SelectARCUnit
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
         END SELECT
'################################################################################
'################################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine SelectARCUnit
       RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                              ppp
'ppp          SUBROUTINE NAME: CorrectARCUnit                     ppp
'ppp                                                              ppp
'ppp       SUBROUTINE TO SELECT ARC LENGTH UNIT                   ppp
'ppp                                                              ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define CORRECTARCUNIT as a subroutine
CORRECTARCUNIT:

'  Change color to blue (1) text and cyan (3) background
     COLOR 1,5

'  Locate the text in row (lline%+8) and column 8
       LOCATE  lline%+8,74

'################################################################################
'################################################################################
'  Beginning of SELECT CASE structure = case variable
'    is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 1, then ARCLENUNIT$="INCHES"
   CASE 1
       ARCLENUNIT$="INCHES"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 2, then ARCLENUNIT$="CENTIMETERS"
   CASE 2
       ARCLENUNIT$="CENTIMETERS"
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% > 2, then continue program
       CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
    '  Define end of SELECT CASE structure
       END SELECT
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

'  Change color to blue (1) text and WHITE (15) background
    COLOR 5,15

'  Tell user the units selected
    IF ARCLENUNIT$="INCHES" THEN
         LOCATE 3,30
       ELSE
            LOCATE 13,27
       END IF
    PRINT " You selected ";ARCLENUNIT$+SPACE$(1)

'  Call PROCEDURE CorrectAnswer to ask if input is correct
    COLOR 2,2
    LOCATE 15,16
    PRINT SPACE$(62)
    LOCATE 15,27
    CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'If the answer is No, then prompt user to enter
'  units again.
765   IF (ans$="N") OR (ans$="n") THEN 720

'  If the answer is Yes, then return to main program
    IF (ans$="y") OR (ans$="Y") THEN 730

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
    COLOR 2,2
    LOCATE 15,53
    PRINT SPACE$(15)
    COLOR 2,2
    LOCATE 17,16
    PRINT SPACE$(62)
    LOCATE 17,18
    CALL InvalidAnswer (ans$)
    GOTO 725

732 PALETTE 0,0
     palette 2,2

'CLEAR SCREEN
    CLS

'  Define end of PROCEDURE ArcLenUnits
    END SUB

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp              SUBROUTINE NAME:  RollerVol                          ppp
'ppp                                                                   ppp
'ppp   SUBROUTINE TO COMPUTE PUMP VOLUME OF ROLLER PUMP               ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define RollerVol as a SUBROUTINE to compute pump volume
'    of roller pump
    RollerVol:

'  Specify color of writing as blue (9) and background
'     as black (0)
    COLOR 9,0

'  If tube diameter is in inches (in), then convert to
'    centimeters (cm).
    IF tubediaunit$="INCHES" THEN
          tubedia=2.54*tubediameter
    ELSE
          tubedia=tubediameter

'  Define end of IF block
    END IF

'  If arc length is in inches (in), then convert to
'     centimeters (cm).
    IF arclenunit$="INCHES" THEN
          arcleng=2.54*arclength
    ELSE
```

```
          arcleng=arclength

'  Define end of IF block
   END IF

'  Computation of roller pump volume.
   pumpvol=numberollers*arcleng*3.14159*(tubedia¢2)/4

   ' where numberollers = number of rollers
   '       arcleng = arc length
   '       tubedia = diameter of tube

'  Define end of SUBROUTINE RollerVol
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp            SUBROUTINE NAME:  PulsePump                         ppp
'ppp                                                                 ppp
'ppp    SUBROUTINE TO GET PULSATILE PUMP PARAMETERS                  ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define PulsePump as a SUBROUTINE to get pump parameters
   PulsePump:

'  erase previous invalid responses
   LOCATE 17,25
   COLOR 0,0
   PRINT SPACE$(56)
   LOCATE 19,16
   COLOR 0,0
   PRINT SPACE$(56)

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=3
bkcol%=5

'define beginning and ending rows of box
begrow%=24
endrow%=56

'define beginning and ending columns of box
begcol%=7
endcol%=44

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'  Specify color of writing as blue (9) and background
'    as black (0)
742      LOCATE 22,15
   COLOR 0,0
   PRINT SPACE$(50)
   LOCATE 20,15
   COLOR 0,0
   PRINT SPACE$(64)
   LOCATE 17,39
   COLOR 5,5
   PRINT SPACE$(30)
   COLOR 3,5
   LOCATE 17,39
   PRINT " ";
740     COLOR 3,5

'  Prompt user to input stroke volume for pulsatile pump
'    in cubic centimeters (cc)
   LOCATE 16,10
        PRINT prompt$;"the stroke volume for the pulsatile pump in cc."
      LOCATE 7,37
   INPUT strokevol

'  If strokevol <=0, then prompt user to input
'       strokevol again
        IF strokevol =<0 THEN
     COLOR 19,0
     SOUND 37,10
     LOCATE 20,16
          PRINT "This is NOT A VALID answer for the STROKE VOLUME."
     LOCATE 17,39
     COLOR 5,5
     PRINT SPACE$(30)
     COLOR 3,5
```

```
      LOCATE 17,39
      PRINT " ";
           GOTO 740
   END IF

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   LOCATE 20,15
   COLOR 0,0
   PRINT SPACE$(64)
'  LOCATE 20,23
   COLOR 0,0
   PRINT SPACE$(44)
   LOCATE 20,27
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to enter
'    stroke volume again
750      LOCATE 20,50
   COLOR 0,0
   PRINT SPACE$(29)
   IF (ans$="N") OR (ans$="n") THEN 742

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 755

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'    to prompt user to input a proper answer of yes or no.
   LOCATE 22,17
   COLOR 0,0
   PRINT SPACE$(60)
   LOCATE 22,18
        CALL InvalidAnswer(ans$)
        GOTO 750

'  Define end of SUBROUTINE PulsePump
755      RETURN
'  ------------------------------------------------------------------------



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  pumbox3.BAS                ==
'==              backup prog :  pumbox3.bak                ==
'==                                                        ==
'============================================================
'   PROGRAM WRITTEN:   2-10-89                             '
'                                                          '
'   PROGRAM MODIFIED:  4-24-89                             '
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                      ppp
'ppp            SUBROUTINE  NAME:  FilePump               ppp
'ppp                                                      ppp
'ppp        SUBROUTINE TO FILE PUMP INFORMATION           ppp
'ppp                                                      ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define FilePump as a SUBROUTINE to display pump
'    information
FilePump:

'  Open file to store pump data
OPEN "PUMP.DAT" FOR OUTPUT AS #10

'  Store pump type
WRITE #10, PumpType$

'  Beginning of SELECT CASE sturcture  = = case variable
'    is PumpType$
        SELECT CASE PumpType$

   '  If PumpType = "ROLLER", then
   '    store roller pump data in file
        CASE "ROLLER" ,"roller"
        WRITE #10, numberollers
        WRITE #10, tubediameter, tubediaunit$
        WRITE #10, arclength, arclenunit$
    WRITE #10, pumpvol
   '  If PumpType = "PULSATILE" , then
```

```
    '    store pulse pump data in file
          CASE "PULSATILE","pulsatile"
         WRITE #10, strokevol

'  Define end of SELECT CASE structure
   END SELECT

'  Close PUMP.DAT file (#10)
   CLOSE #10

'  Define end of SUBROUTINE FilePump
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp            SUBROUTINE  NAME:  ShowPump                          ppp
'ppp                                                                 ppp
'ppp       SUBROUTINE TO DISPLAY PUMP INFORMATION                    ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define ShowPump as a SUBROUTINE to display pump
'    information
   ShowPump:

'  Specify color of writing as light cyan (1) and background
'     as black (0)
      COLOR 1,0

'  Beginning of SELECT CASE sturcture  = = case variable
'    is PumpType$
        SELECT CASE PumpType$

   '  If PumpType = "ROLLER" OR "roller", then
   '    call SUBRUOTINE ShowRollPump
         CASE "ROLLER" ,"roller"
        GOSUB ShowRollPump

   '  If PumpType = "PULSATILE" OR "pulsatile", then
   '    call SUBROUTINE ShowPulse
         CASE "PULSATILE","pulsatile"
        GOSUB ShowPulse

'  Define end of SELECT CASE structure
   END SELECT

'  Define end of SUBROUTINE ShowPump
   RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp            SUBROUTINE NAME:  ShowPulse                          ppp
'ppp                                                                 ppp
'ppp    SUBROUTINE TO DISPLAY PULSATILE PUMP PARAMETERS              ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define ShowPulse as a SUBROUTINE to display
'    pump parameters
   ShowPulse:

'  Specify color of writing as dark cyan (11) and background
'    as black (0)
     COLOR 1,0

'  Print a blank line
   PRINT

'  locate text
     LOCATE row%,col%

'  Print type of pulsatile blood pump in use
   PRINT "PULSATILE BLOOD PUMP IN USE"

'  locate text
   LOCATE row%+1,col%+15

'  Print pulsatile parameter of stroke volume to screen
   PRINT"stroke volume is  ";strokevol;"cc"

'  Define end of SUBROUTINE ShowPulse
   RETURN
```

```
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                    ppp
'ppp            SUBROUTINE NAME:  ShowRollPump                          ppp
'ppp                                                                    ppp
'ppp    SUBROUTINE TO DISPLAY ROLLER PUMP INFORMATION                   ppp
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'   Define ShowRollPump as a SUBROUTINE
    ShowRollPump:

'   Specify color of writing as blue (1) and background
'    as palette color 3
    COLOR 1,3

PALETTE 1,62
PALETTE 13,62

'   Specify color of writing as LT. BLUE (3) and background as
'    palette color 3
    COLOR 3,0

'set text color (txcol%) and gackground color (bkcol%)
txcol%=15
bkcol%=1

'define beginning and ending rows of box
begrow%=row%+3
endrow%=row%+4

'define beginning and ending columns of box
begcol%=col%-5
endcol%=col%+36

'   CLEAR THE SCREEN
    CLS

'call procedure menubox to draw box
 call MENUBOX(begrow%,endrow%,begcol%,endcol%,txcol%,bkcol%)

'   Print type of blood pump
770    LOCATE row%+5,col%+4
    PRINT SPACE$(1);"ROLLER PUMP"

'   Print number of rollers on screen
       LOCATE row%+4,col%-6
       PRINT"NUMBER OF ROLLERS IS :  "
       LOCATE row%+4,col%+27
       PRINT SPACE$(1);numberollers

'   Print tube diameter of roller pump tubing
       LOCATE row%+5,col%-6
         PRINT"TUBE DIAMETER IS :  "
       LOCATE row%+5,col%+62
       PRINT SPACE$(1);tubediameter

'   Print units of tube diameter on screen
       LOCATE row%+6,col%-6
       PRINT"UNITS OF TUBE DIAMETER:  "
       LOCATE row%+6,col%+27
       PRINT SPACE$(2)+tubediaunit$

'   Print arc length of roller pump on screen
       LOCATE row%+7,col%-4
       PRINT"ARC LENGTH IS :          "
       LOCATE row%+7,col%+22
    PRINT SPACE$(1);arclength

'   Print units of arc length on screen
       LOCATE row%+8,col%-4
       PRINT"UNITS OF ARC LENGTH:     "
       LOCATE row%+8,col%+22
       PRINT SPACE$(2)+arclenunit$

'   End of SUBROUTINE ShowRollPump, return to main program
    RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                    ppp
'ppp            SUBROUTINE NAME:  CheckRollPump                         ppp
'ppp                                                                    ppp
'ppp     SUBROUTINE TO CHECK ROLLER PUMP INFORMATION                    ppp
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

```
'  Define CheckRollPump as a SUBROUTINE
     CheckRollPump:

'  Call PROCEDURE CorrectAnswer to ask if answer is correct
     LOCATE 14,23
     CALL CorrectAnswer (ans$)

'  Specify color of writing as red (4) and background
'     as CYAN (3)
380    COLOR 4,0

'  Check answer to PROCEDURE CorrectAnswer :

'  If answer is No, then go to subroutine CorrectRollPump
'    to correct incorrect input of pump info
       if (ans$="n") or (ans$="N") then
         locate 16,13
         color 3,0
         print space$(34)
         locate 14,24
         color 3,0
         print space$(34)
         color 4,0


'  tell user to use up and down arrow keys to select
'     the incorrect entry
      locate 16,13
      print "please use the UP arrow key ";chr$(24);
      locate 17,13
      print " and the DOWN arrow "
      locate 17,13
      print "key ";chr$(24);" to move the box to the line needing
            correction."
      locate 18,19
      print
      locate 19,17
      print "Then, press the enter key to select it and ";
      locate 20,17
      print"type the correct input."

      GOSUB CorrectRollPump
      GOTO 785

'  define end of if block
     end if

'  if answer is yes, then go back to main program
     IF (ans$="y") OR (ans$="Y") THEN 785

'  if answer was invalid, then call procedure invalidanswer
'    to prompt user to input a proper answer of yes or no.
781   locate 14,51
      color 3,0
      print space$(25)
      locate 16,13
      color 2,0
      print space$(62)
      locate 16,15
      call invalidanswer(ans$)

GOTO 780

'  define end of subroutine CheckRollPump
785   return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                ppp
'ppp          SUBROUTINE NAME:  CorrectRollPump                     ppp
'ppp                                                                ppp
'ppp    SUBROUTINE TO SELECT ERROR OF ROLLER PUMP INPUT             ppp
'ppp                                                                ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'   Define CorrectRollPump as a SUBROUTINE
    CorrectRollPump:

'  Print two blank lines
800       PRINT

'    define numerical variable numberollers as a string
'     (STR$) and assign it to INFO$(1) with a space added
'     at the end
    INFO$(1)=STR$(numberollers)+SPACE$(1)
```

```
'   define numerical variable tubediameter as a string
'    (STR$) and assign it to INFO$(2) with a space added
'    at the end
   INFO$(2)=STR$(tubediameter)+SPACE$(1)

'   define tubediaunit$ as INFO$(3) and add a space
'    at the beginning and end
       INFO$(3)=SPACE$(1)+tubediaunit$+SPACE$(1)

'   define numerical variable arclength as a string (STR$)
'    and assign it to INFO$(4) with a space added
'    at the end
   INFO$(4)=STR$(arclength)+SPACE$(1)

'   define arclenunit$ as INFO$(5) and add a space
'    at the beginning and end
   INFO$(5)=SPACE$(1)+arclenunit$+SPACE$(1)


'  define LLINE% as an integer used to locate text
'   on proper row
     LLINE%=2

'  change color to yellow (14) text and red (12)
'   background;  locate the text at row row%+3 and
'   column 27;  print INFO$(1)
   COLOR 14,12
   LOCATE row%+4,col%+23
   PRINT INFO$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                     XX
'XX       This selection routine was originally written                 XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                    XX
'XX                                                                     XX
'XX       Modifications and all documentation                           XX
'XX     was done by Cynthia K. Rice on 3-1-89.                          XX
'XX                                                                     XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label CorrectPump
'   so it can be called again
   CorrectPump:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
       KKEY$=INKEY$

'################################################################
'################################################################
'  Beginning of SELECT CASE structure = case variable
'   is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto CorrectPump (continue selection process)
       CASE ""
       GOTO CorrectPump
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the foll wing  ines

'   CHR$(-) is the character yn the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(82) represents the up arroxp
'     CHR$(0)+CHR$(80) represents the down arrow

      '  then decrement LLINE% by 1
      ' ELSE afcrement LLINE% by 1 (meaning the up arrow"
      '  was selfcted)
         IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE% ELSE INCR LLINE%

      '  top line
    IF LLINE%>5 THEN LLINE%=1

      ' channge the color to yellow (14) text and red (12)
      '  background;  print next text at row (LLINE%+5)
      '   and column 27
```

```
          COLOR 14,12
          LOCATE LLINE%+6,col%+23

       ' print INFO$(LINE%)
          PRINT INFO$(LINE%)

      ' goto label CorrectPump to continue selection process
    GOTO CorrectPump:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
          CASE CHR$(13)

      '  goto subroutine DoCorrectPump to correct pump info
           GOSUB DoCorrectPump:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label CorrectPump
'  to continue selection process
          CASE ELSE
          GOTO CorrectPump
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
          END SELECT
'##########################################################################
'##########################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine CorrectRollPump
       RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                    ppp
'ppp           SUBROUTINE NAME: DoCorrectPump                           ppp
'ppp                                                                    ppp
'ppp     SUBROUTINE TO CORRECT ERROR OF PUMP INPUT                      ppp
'ppp                                                                    ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  define DoCorrectPump as a subroutine
DoCorrectPump:

'  Change color to blue (3) text and black (0) background
color 15,1

'  Locate the text in row (lline%+5) and column 26
60601 LOCATE  lline%+6,col%+22

'  Print spaces the length of INFO$ to be sure all of
'   string is erased
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Change color to yellow (14) text and red (12) background
color 14,12

'  Locate the text in row (lline%+5) and column 26
      LOCATE  lline%+6,col%+23

'  Print spaces to locate cursor for user
   PRINT SPACE$(LEN(INFO$(lline%)))

'  Locate the text in row (lline%+5) and column 26
      LOCATE  lline%+6,col%+23

'##########################################################################
'##########################################################################
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then input number of rollers
   '   for roller pump again
   CASE 1
   INPUT "", numberollers

   '  If numberollers <=0, then prompt user to input
   '    numberollers again

         IF numberollers =<0 THEN
```

```
    LOCATE LLINE%+6,COL%+22
    COLOR 12,12
    PRINT SPACE$(23)
    COLOR 14,12
    LOCATE LLINE%+7,12
    SOUND 75,10
          PRINT " This is not a valid answer for the number of rollers. "
    LOCATE LLINE%+8,12
      PRINT " Please enter a number greater than zero.              "
          GOTO 60601

    '  define end of IF block
      END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 2, then input tube diameter
    '    for roller pump again
    CASE 2
         INPUT "", tubediameter

    '  If tubediameter <=0, then prompt user to input
    '     tubediameter again
          IF tubediameter =<0 THEN
    LOCATE LLINE%+6,COL%+22
    COLOR 12,12
    PRINT SPACE$(21)
    COLOR 14,12
    LOCATE LLINE%+7,14
    SOUND 75,10
          PRINT " This is not a valid answer for the tube diameter. "
    LOCATE LLINE%+8,14
      PRINT " Please enter a number greater than zero.          "
          GOTO 60601

    '  define end of IF block
      END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 3, then input units of tube diameter again
    CASE 3
         INPUT "",tubediaunit$

     '  If units are inches (in) or centimeters (cm),
     '    continue program
          IF (tubediaunit$="in") OR (tubediaunit$="IN")
           OR (tubediaunit$="cm") OR (tubediaunit$="CM") THEN
          IF (tubediaunit$="in") OR (tubediaunit$="IN")
           THEN tubediaUNIT$="INCHES"
          IF (tubediaunit$="cm") OR (tubediaunit$="CM")
           THEN tubediaUNIT$="CENTIMETERS"
           GOTO 60600

     '    ELSE units are not inches or centimeters
           ELSE
      '   tell user that entry of tube diameter units is invalid,
      '     please input proper tube diameter units
    LOCATE LLINE%+6,COL%+22
    COLOR 12,12
    PRINT SPACE$(25)
    COLOR 14,12
    LOCATE LLINE%+7,10
    SOUND 75,10
          PRINT " This is not a valid entry
           for the units of tube diameter. "
    LOCATE LLINE%+8,10
          PRINT " Please enter 'CM' for centimeters
           or 'IN' for inches.     "
      '  goto  select case again to enter tube diameter unit
         GOTO 60601

    '  define end of if block
60600    END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% = 4, then enter arc length again
         CASE 4
         INPUT "",arclength

      ' If arclength <=0, then prompt user to input
      '    arclength again
          IF arclength =<0 THEN
```

```
    LOCATE LLINE%+6,COL%+22
    COLOR 12,12
    PRINT SPACE$(20)
    COLOR 14,12
    LOCATE LLINE%+7,15
    SOUND 75,10
          PRINT " This is not a valid answer for the arc length.  "
    LOCATE LLINE%+8,15
          PRINT " Please enter a number greater than zero.        "
       ' goto select case again to enter arc length
          GOTO 60601

      ' define end of if block
          END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    '  If LLINE% = 5, then input units of arc length again
       CASE 5
       INPUT "",arclenunit$

     '  If units are inches (in) or centimeters (cm),
     '   continue program
         IF (arclenunit$="in") OR (arclenunit$="IN")
          OR (arclenunit$="cm") OR (arclenunit$="CM") THEN
         IF (arclenunit$="in") OR (arclenunit$="IN")
          THEN arclenUNIT$="INCHES"
         IF (arclenunit$="cm") OR (arclenunit$="CM")
          THEN arclenUNIT$="CENTIMETERS"
          GOTO 60666

     '   ELSE units are not inches or centimeters
         ELSE
      '  tell user that entry of arc length units is invalid,
      '   please input proper arc length  units
    LOCATE LLINE%+6,COL%+22
    COLOR 12,12
    PRINT SPACE$(24)
    COLOR 14,12
    LOCATE LLINE%+7,12
    SOUND 75,10
          PRINT " This is not a valid entry
           for the units of arc length. "
    LOCATE LLINE%+8,12
          PRINT " Please enter 'CM' for centimeters
           or 'IN' for inches.  "
       '  goto  select case again to enter arc length unit
          GOTO 60601

    '  define end of if block
60666    END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  If LLINE% > 5, then continue program
       CASE ELSE
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
       END SELECT
'###############################################################
'###############################################################

'  Specify color of writing as dark cyan (3) and background
'    as black (0)
    COLOR 3,0

'  Clear the Screen
    CLS

'  Goto subroutine ShowRollPump to display roller pump
'    information
    GOSUB ShowRollPump

'  Goto subroutine CheckRollPump to check if roller pump
'    information is correct
         GOSUB CheckRollPump

'  Define end of SUBROUTINE DoCorrectPump return to main
6767     RETURN
'
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  disinfo.bas                ==
'==              backup prog :  disinfo.bak                ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   4-9-89                             '
'                                                          '
'    PROGRAM MODIFIED:  5-17-89                            '
'============================================================
'==  PROGRAM NAME:  disinfo.bas                            ==
'==                                                        ==
'==    This program calls subroutines to display menu      ==
'==  boxes with patient information, temperature names,    ==
'==  timer names, and  pump information.  Also,            ==
'==  correction of this information is possible using      ==
'==  this program.                                         ==
'==                                                        ==
'============================================================
'define DISINFO as a subroutine
DISINFO:

'tell computer to start new machine code segment
$segment

'set flag place%=2  for return from escape key
place%=1

'define prompt$ as a shared variable passed to procedures
prompt$="Please enter "

'declare text as blue (2) and background as black (2)
palette 2,2
COLOR 2,2

'turn function key menu across bottom off
key off

'CLEAR THE SCREEN
cls

'call subroutine to draw PATIENT infomation box
GOSUB DRAWPATBOX

'call subroutine to draw PUMP infomation box
GOSUB DRAWPUMPBOX

'call subroutine to draw TIMER infomation box
GOSUB DRAWTIMEBOX

'call subroutine to draw TEMP infomation box
GOSUB DRAWTEMPBOX

'call subroutine to draw FUNCTION KEY box
GOSUB DRAWFKBOX

'============================================================
'define end SUBROUTINE DISINFO
RETURN
'============================================================

'add these programs to save space in turbo basic
'editor

$INCLUDE "c:\monitor1\menubox.bas"
$INCLUDE "c:\monitor1\patbox.bas"
$INCLUDE "c:\monitor1\pumpbox.bas"
$INCLUDE "c:\monitor1\timebox.bas"
$INCLUDE "c:\monitor1\tempbox.bas"
$INCLUDE "c:\monitor1\fkbox.bas"



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  menubox.bas                ==
'==              backup prog :  menubox.bak                ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   4-9-89                             '
```

```
'                                                                    '
'    PROGRAM MODIFIED:  4-19-89                                      '
'====================================================================
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                ppp
'ppp              PROCEDURE  NAME:  MENUBOX                         ppp
'ppp                                                                ppp
'ppp       PROCEDURE TO DRAW MENU BOXES FOR DISPLAY                 ppp
'ppp                                                                ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

' Define MENUBOX as a PROCEDURE
SUB MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

color TXCOL%,BKCOL%

'fill box with color of chosen background
for linespaces%=begrow%+1 to endrow%-1
locate linespaces%,begcol%
print space$(endcol%-begcol%)
next linespaces%

'top left corner
locate begrow%,begcol%
print chr$(211)

'border across top
for rr%=begcol%+1 to endcol%-1
locate begrow%,rr%
print chr$(225)
next rr%

'top right corner
locate begrow%,endcol%
print chr$(127)
'input x

'right side
for cc%=begrow%+1 to endrow%-1
locate cc%,endcol%
print chr$(116)
next cc%

'bottom right corner
locate endrow%,endcol%
print chr$(118)

'bottom border
for rr%=endcol%-1 to begcol%+1 step -1
locate endrow%,rr%
print chr$(215)
next rr%

'bottom left corner
locate endrow%,begcol%
print chr$(120)

'left side
for cc%=endrow%-1 to begrow%+1 step -1
locate cc%,begcol%
print chr$(116)
next cc%

'define end of procedure MENUBOX
END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp



'====================================================================
'==              ***** IMPORTANT INFO *****                        ==
'==                                                                ==
'==              program name:  PATBOX.bas                         ==
'==              backup prog :  PATBOX.bak                         ==
'==                                                                ==
'====================================================================
'    PROGRAM WRITTEN:   4-9-89                                      '
'                                                                   '
'    PROGRAM MODIFIED:  4-19-89                                     '
'___________________________________________________________________
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                ppp
```

```
'ppp            SUBROUTINE  NAME:  DRAWPATBOX                ppp
'ppp                                                         ppp
'ppp     SUBROUTINE TO DRAW PATIENT INFORMATION BOX          ppp
'ppp                                                         ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define DRAWPATBOX as a SUBROUTINE
DRAWPATBOX:

'prog to draw menu boxes
'============================================================
'============================================================
'==             print patient info box                     ==
'============================================================
'set palette color 6 to yellow (62)
PALETTE 6,62

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=9
BKCOL%=6

'define beginning and ending rows of menu box
begrow%=5
endrow%=44

'define beginning and ending columns of menu box
begcol%=13
endcol%=26

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print heading
locate begrow%+1,begcol%+7
print "PATIENT INFORMATION: "

'print a blank line after heading
locate begrow%+2,begcol%+3
print ""

'print patient name
locate begrow%+3,begcol%+6
print "NAME:     ";patname$

'print sex of patient
locate begrow%+4,begcol%+6
print "SEX:      ";SEX$

'print ID$ of patient
locate begrow%+5,begcol%+6
print "ID #:     ";idnum$

'print age and units of age on same line
locate begrow%+6,begcol%+6
print "AGE:      ";AGE;AGEunit$

'print height and units of height on same line
locate begrow%+7,begcol%+3
print "HEIGHT: ";height;htunit$

'print weight and units of weight on same line
locate begrow%+8,begcol%+3
print "WEIGHT: ";weight;wtunit$
'============================================================
'define end of subroutine DRAWPATBOX
RETURN
'-------------------------------------------------------------



'============================================================
'==             ***** IMPORTANT INFO *****                 ==
'==                                                        ==
'==             program name:  PUMPBOX.bas                 ==
'==             backup prog :  PUMPBOX.bak                 ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   4-9-89                              '
'                                                           '
'    PROGRAM MODIFIED:  4-19-89                             '
'                                                           '
'-----------------------------------------------------------
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                         ppp
```

```
'ppp              SUBROUTINE  NAME:  DRAWPUMPBOX              ppp
'ppp                                                         ppp
'ppp   SUBROUTINE TO CORRECT DRAW PUMP INFORMATION BOX       ppp
'ppp                                                         ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define DRAWPUMPBOX as a SUBROUTINE
DRAWPUMPBOX:

'============================================================
'============================================================
'==                print pump info box                       ==
'============================================================
'open file pump.dat to read pump info
open "c:\MONITOR1\pump.dat" for input as #9

'read pump type from file
input #9,pumptype$

 IF PUMPTYPE$="ROLLER" THEN
  input #9,numberollers,tubediameter,tubediaunit$,arclength,arclenunit$
 ELSE
  input #9,strokevol
 END IF

'close pump.dat
close #9

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=15
BKCOL%=1

'define beginning and ending rows of menu box
   IF PUMPTYPE$="ROLLER" THEN
        begrow%=2
        endrow%=9
       ELSE
          begrow%=2
          endrow%=8
       END IF

'define beginning and ending columns of menu box
begcol%=4
endcol%=23

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print heading
locate begrow%+1,begcol%+7
print "BLOOD PUMP INFORMATION: "

'print a blank line after heading
locate begrow%+2,begcol%+2
print ""


color 15,1
'setup display for roller pump
IF PUMPTYPE$="ROLLER" THEN

   'print pump type
   color   6,1
   locate begrow%+2,begcol%+9
   print "PUMP TYPE: ";PUMPTYPE$

   'print height and units of height on same line
   locate begrow%+4,begcol%+2
   print "NUMBER OF ROLLERS: ";NUMBEROLLERS

   'print weight and units of weight on same line
   locate begrow%+5,begcol%+2
   print "TUBE DIAMETER:     ";TUBEDIAMETER;TUBEDIAUNIT$

   'print weight and units of weight on same line
   locate begrow%+6,begcol%+2
   print "ARC LENGTH:        ";ARCLENGTH;ARCLENUNIT$

'setup display for pulsatile pump
ELSE
   'print pump type
   color   6,1
   locate begrow%+2,begcol%+8
   print "PUMP TYPE: ";PUMPTYPE$
```

```
    'print stroke volume for pulsatile pump
    locate begrow%+4,begcol%+2
    display$="    STROKE VOLUME:  #.## cc"
    print using display$;strokevol

END IF
'============================================================
'define end of subroutine DRAWPUMPBOX
RETURN
'-------------------------------------------------------------



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  TEMPBOX.bas                ==
'==              backup prog :  TEMPBOX.bak                ==
'==                                                        ==
'============================================================
'     PROGRAM WRITTEN:   4-9-89                            '
'                                                          '
'     PROGRAM MODIFIED:  5-07-89                           '
'----------------------------------------------------------'
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                        ppp
'ppp             SUBROUTINE  NAME:  DRAWTEMPBOX             ppp
'ppp                                                        ppp
'ppp   SUBROUTINE TO CORRECT DRAW TEMP INFORMATION BOX      ppp
'ppp                                                        ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define DRAWTEMPBOX as a SUBROUTINE
DRAWTEMPBOX:

'============================================================
'============================================================
'==           print temperature info box                   ==
'============================================================
palette 15,53

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=4
BKCOL%=35

'define beginning and ending rows of menu box
begrow%=2
endrow%=23

'define beginning and ending columns of menu box
begcol%=54
endcol%=69

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,becol%,endcol%,TXCOL%,BKCOL%)

'locate and print heading
locate begrow%+1,begcol%+2
print "TEMPERATURE PROBE INFORMATION: "

'print a blank line after heading
locate begrow%+2,begcol%+2
print ""

'print temperature probe name
locate begrow%+3,becol%+3
print "TEMP #1 NAME:  ";TEMP1$

'print temperature probe name
locate begrow%+4,becol%+3
print "TEMP #2 NAME:  ";TEMP2$

'print temperature probe name
locate begrow%+5,becol%+3
print "TEMP #3 NAME:  ";TEMP3$

'print temperature probe name
locate begrow%+6,becol%+3
print "TEMP #4 NAME:  ";TEMP4$

'print temperature probe name
locate begrow%+7,becol%+3
print "TEMP #5 NAME:  ";TEMP5$
```

```
'print temperature probe name
locate begrow%+8,becol%+3
print "TEMP #6 NAME:  ";TEMP6$

   IF TEMPUNITS$="CELSIUS" then
    tmpunit$="CELSIUS"
   ELSE
        tmpunit$="FARENHEIT"
   END IF

'print temperature units
locate begrow%+10,becol%+2
PRINT "UNITS OF TEMPERATURE: ";TMPUNIT$

'=============================================================

'define end of subroutine DRAWTEMPBOX
RETURN
'_____________________________________________________________



'=============================================================
'==              ***** IMPORTANT INFO *****                 ==
'==                                                         ==
'==              program name:  TIMEBOX.bas                 ==
'==              backup prog :  TIMEBOX.bak                 ==
'==                                                         ==
'=============================================================
'    PROGRAM WRITTEN:   4-09-89                             '
'                                                           '
'    PROGRAM MODIFIED:  5-07-89                             '
'___________________________________________________________'
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                       ppp
'ppp            SUBROUTINE  NAME:  DRAWTIMEBOX             ppp
'ppp                                                       ppp
'ppp  SUBROUTINE TO CORRECT DRAW TIMER INFORMATION BOX     ppp
'ppp                                                       ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define DRAWTIMEBOX as a SUBROUTINE
DRAWTIMEBOX:

'=============================================================
'=============================================================
'==               print timer info box                      ==
'=============================================================
PALETTE 6,62: PALETTE 3,3

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=5
endrow%=9

'define beginning and ending columns of menu box
begcol%=34
endcol%=59

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print heading
locate begrow%+1,begcol%+12
print "TIMER NAMES: "

'print a blank line after heading
locate begrow%+2,begcol%+5
print ""

'print timer name
locate begrow%+3,begcol%+6
print "TIMER #1 NAME:  ";TIME1$

'print timer name
locate begrow%+4,begcol%+4
print "TIMER #2 NAME:  ";TIME2$
```

```
'print timer name
locate begrow%+5,begcol%+3
print "TIMER #3 NAME:  ";TIME3$

'============================================================
'define end of subroutine DRAWTIMEBOX
RETURN
'------------------------------------------------------------



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  FKBOX.bas                  ==
'==              backup prog :  FKBOX.bak                  ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   4-9-89                             '
'                                                          '
'    PROGRAM MODIFIED:  5-17-89                            '
'------------------------------------------------------------
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                        ppp
'ppp              SUBROUTINE  NAME:  DRAWFKBOX              ppp
'ppp                                                        ppp
'ppp     SUBROUTINE TO CORRECT DRAW FUNCTION KEY BOX        ppp
'ppp                                                        ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define DRAWFKBOX as a SUBROUTINE
DRAWFKBOX:

'============================================================
'============================================================
'==      print function key box at bottom of screen        ==
'============================================================
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=12
BKCOL%=0

'define beginning and ending rows of menu box
begrow%=21
endrow%=23

'define beginning and ending columns of menu box
begcol%=2
endcol%=59

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print instructions to user
LOCATE 21,3
PRINT "Press F1 for correction mode,";
PRINT " F5 to continue, and ESC to exit program."

'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'  THIS LABEL MARKS LOOP THAT RUNS UNTIL EITHER FUNCTION  ¢
'  KEY F1 OR F5 IS PRESSED                                 ¢
'                                                          ¢
'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'label to run loop until function keys f1 or f5 are pressed
FUNCKEY:

'assign keyboard buffer to variable fkey$
fkey$=INKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'beginning of SELECT CASE structure = variable
'  is FKEY$
SELECT CASE FKEY$
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'if fkey$="" (no keystroke on keyboard buffer) then
' goto label FUNCKEY to continue loop
CASE ""
GOTO FUNCKEY
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
'++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (1) is pressed, send to
' correction subroutine
CASE CHR$(0)+CHR$(59)
GOSUB KOREKTDISINFO
'++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (5) is pressed, send to
' CONTINUEIT subroutine
CASE CHR$(0)+CHR$(67)
GOSUB CONTINUEIT
'++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++
'if keyboard input is anything besides f1 or f2 then
' goto label FUNCKEY (continue loop)
CASE ELSE
GOTO FUNCKEY
'++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++
'define end of SELECT CASE structure
END SELECT
'++++++++++++++++++++++++++++++++++++++++++++++++++
'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'define end of subroutine DRAWFKBOX
RETURN
'________________________________________________________

'pppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                ppp
'ppp           SUBROUTINE  NAME:  KOREKTDISINFO     ppp
'ppp                                                ppp
'ppp SUBROUTINE TO CORRECT INFO FROM LAST DISPLAY SCREEN ppp
'ppp                                                ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define KOREKTDISINFO as a SUBROUTINE
KOREKTDISINFO:

'======================================================
'==            print function key box on screen     ==
'======================================================
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=5
BKCOL%=2

'define beginning and ending rows of menu box
begrow%=6
endrow%=14

'define beginning and ending columns of menu box
begcol%=20
endcol%=55

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print heading
locate begrow%+1,begcol%+3
print "FUNCTION KEYS FOR CORRECTION"

'print a blank line after heading
locate begrow%+2,begcol%+2
print ""

'print function key info
locate begrow%+3,begcol%+3
print "F1  Correct Patient Information"

'print function key info
locate begrow%+4,begcol%+3
print "F2  Correct Pump Information"


'print function key info
locate begrow%+5,begcol%+3
print "F3  Correct Timer Names"

'print function key info
locate begrow%+6,begcol%+3
print "F4  Correct Temp Names or Units"
```

```
'print function key info
locate begrow%+7,begcol%+3
print "F5  EXIT Correction Mode"

'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'      THIS LABEL MARKS LOOP THAT RUNS UNTIL ONE OF THE     ¢
'       FUNCTION KEYS F1, F2, F3, F4, OR F5 IS PRESSED      ¢
'                                                           ¢
'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢
'label marking loop to run until keyboard input of F1,
' F2, F3, F4, or F5
getitright:

'assign keyboard buffer to variable akey$
akey$=INKEY$

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'beginning of SELECT CASE structure = variable
' is AKEY$
SELECT CASE AKEY$
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'if akey$="" (no keystroke on keyboard buffer) then
' goto label getitright to continue loop
CASE ""
GOTO getitright
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (1) is pressed, send to
' KORRECTPATINFO subroutine to correct patient info
CASE CHR$(0)+CHR$(59)
GOSUB KORRECTPATINFO
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (2) is pressed, send to
' KORRECTEMP subroutine to correct pump info
CASE CHR$(0)+CHR$(60)
GOSUB KORRECTPUMP
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (3) is pressed, send to
' KORRECTIMER subroutine to correct timer names
CASE CHR$(0)+CHR$(62)
GOSUB KORRECTIMER
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (4) is pressed, send to
' KORRECTPUMP subroutine to correct temperature info
CASE CHR$(0)+CHR$(63)
GOSUB KORRECTEMP
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'when function key (5) is pressed, send to
' EXITCOR subroutine to exit correction mode
CASE CHR$(0)+CHR$(64)
GOSUB EXITCOR
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'if keyboard input is anything besides f1, f2, f3,
' f4, or f5 then goto label getitright (continue loop)
CASE ELSE
GOTO getitright
'+++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'define end of SELECT CASE structure
END SELECT
'¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢¢

'define end of subroutine
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                    ppp
'ppp            SUBROUTINE  NAME:  CONTINUEIT           ppp
'ppp                                                    ppp
```

```
'ppp      SUBROUTINE TO CONTINUE MONITORING PROGRAM        ppp
'ppp                                                       ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define CONTINUEIT as a SUBROUTINE
CONTINUEIT:

'define end of subroutine CONTINUEIT
RETURN main1
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                       ppp
'ppp           SUBROUTINE  NAME:  KORRECTPATINFO           ppp
'ppp                                                       ppp
'ppp      SUBROUTINE TO CORRECT PATIENT INFORMATION        ppp
'ppp                                                       ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define KORRECTPATINFO as a SUBROUTINE
KORRECTPATINFO:

'define text color as navy blue (1) and background as
' light blue (3)
COLOR 1,3

'clear the screen
CLS

'define row and column locating text
ROW%=7
COL%=34

'subroutine to show patient information
GOSUB SHOWPATINFO

'subroutine to check and correct patient info
GOSUB CHECKPATINFO

'subroutine to file patient info
GOSUB FILEPAT

'subroutine to display all info again
GOSUB DISINFO

'define end of subroutine KORRECTPATINFO
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                       ppp
'ppp             SUBROUTINE  NAME:  KORRECTPUMP            ppp
'ppp                                                       ppp
'ppp         SUBROUTINE TO CORRECT PUMP INFORMATION        ppp
'ppp                                                       ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define KORRECTPUMP as a SUBROUTINE
KORRECTPUMP:

'assign color 3 (light blue) to pallete jar 9
PALETTE 3,9

'assign color 0 (black) to pallete jar 59
PALETTE 0,59

'define color of text as palette color 3 and
' background color as palette color 0
COLOR 3,0

'clear the screen
CLS

'define row and column locating text
ROW%=4
COL%=55

'locate text on screen (row,column)
LOCATE ROW%+5,20

'print current type of pump on screen
PRINT "A " ;pumptype$; " blood pump was previously selected."

'locate text on screen (row,column)
LOCATE ROW%+7,28

'ask if pump type is correct
PRINT "Is this correct (Y/N)?
```

```
'locate text on screen (row,column)
LOCATE ROW%+8,28

'wait for answer to question
INPUT ANS$

'if the answer is no, then call subroutines to
' get new pump type and pump parameters, then out
' of loop to continue
60708 COLOR 3,0
LOCATE ROW%+10,20
PRINT SPACE$(58)
      IF (ANS$="N") OR (ANS$="n") THEN
   CLS
   CALL PUMPTYPE(PUMPTYPE$)
   CALL PUMPPARAMETERS(PUMPTYPE$)
   GOTO 7089
      END IF

'if answer is yes (pump type is correct), then
'  if roller pump call subroutines to show and
'  correct roller pump info
      IF (ANS$="y") OR (ANS$="Y") THEN
       IF PUMPTYPE$="ROLLER" THEN
         COLOR 0,0
       · CLS
             GOSUB SHOWROLLPUMP
          GOSUB CHECKROLLPUMP

'else, if pump type is correct, then is a
' pulsatile pump, call subroutines to show
' and correct pump info
       ELSE
         CLS
             GOSUB SHOWPULSE
             GOSUB PULSEPUMP
       END IF

'continue program
       GOTO 7089

'define end of if block
      END IF

'if program gets to here, then answer to pump
' type question is not y or n; goto invalidanswer
' subroutine and then to line 60708
locate row%+10,20
CALL INVALIDANSWER (ANS$)
GOTO 60708

'clear the screen
7089 color 3,0
     CLS

'assign color 3 (light blue) to pallete jar 3
PALETTE 3,3

'assign color 0 (black) to pallete jar 0
PALETTE 1,0

'subroutine to file pump information
GOSUB FILEPUMP

'subroutine to display all info again
GOSUB DISINFO

'define end of subroutine KORRECTPUMP
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp             SUBROUTINE  NAME:  KORRECTIMER                      ppp
'ppp                                                                 ppp
'ppp       SUBROUTINE TO CORRECT TIMER INFORMATION                   ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define KORRECTIMER as a SUBROUTINE
KORRECTIMER:

'define text color as navy blue (1) and background
'  color as white (15)
COLOR 1,15
```

```
'clear the screen
CLS

'define row and column locating text
ROW%=3
COL%=25

'subroutine to display timer names
GOSUB TIMEDISP

'subroutine to check and correct timer names
GOSUB CHECKTIMERNAME

'++++++++++++++++++++++++++++++++++++++++++++++++++
'begin select case structure == variable
' is lline%
'select case lline%

'if lline% is 1 then goto subroutine
' filetime1 to file time1 name
'case 1
'   GOSUB FILETIME1

'if lline% is 2 then goto subroutine
' filetime2 to file time2 name
'case 2
'   GOSUB FILETIME2

'if lline% is 3 then goto subroutine
' filetime3 to file time3 name
'case 3
'   GOSUB FILETIME3

'define end of case select structure
'end select
'++++++++++++++++++++++++++++++++++++++++++++++++++

'subroutine to display all info again
GOSUB DISINFO

'define end of subroutine KORRECTIMER
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp             SUBROUTINE  NAME:  KORRECTEMP                       ppp
'ppp                                                                 ppp
'ppp   SUBROUTINE TO CORRECT TEMPERATURE INFORMATION                 ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define KORRECTEMP as a SUBROUTINE
KORRECTEMP:

'assign color 62 (yellow) to pallete jar 0
PALETTE 0,62

'define text color as blue (9) and background as
' palette color 0
COLOR 9,0

'clear the screen
CLS

'define row and column locating text
ROW%=3
COL%=21

'locate text on screen (row,column)
LOCATE ROW%+5,15

'print current type of pump on screen
PRINT "Currently, the units of temperature are ";tmpunit$;"."

'locate text on screen (row,column)
LOCATE ROW%+7,7

'ask if pump type is correct
PRINT "Is this correct (Y/N)?

'locate text on screen (row,column)
LOCATE ROW%+8,7
```

```
'wait for answer to question
INPUT ANS$

'if the answer is no, then call subroutines to
' get new temperature units parameters, then out
' of loop to continue
1607  COLOR 9,0
LOCATE ROW%+10,17
PRINT SPACE$(60)
      IF (ANS$="N") OR (ANS$="n") THEN
   PALETTE 0,0
   CALL GetTEMPunit(tempunits$)
         COLOR 9,0
         GOTO 1710
      END IF

'if answer is yes (temp units are correct), then
'  display temp names for correction
      IF (ANS$="y") OR (ANS$="Y") THEN
   1710 PALETTE 0,0
           CLS
              GOSUB TEMPDISP
           GOSUB CHECKTEMPNAMES
   '++++++++++++++++++++++++++++++++++++++++++++++++

'continue program
       GOTO 1708

'define end of if block
      END IF

'if program gets to here, then answer to temp
' units question is not y or n; goto invalidanswer
' subroutine and then to line 1607
LOCATE ROW%+10,17
CALL INVALIDANSWER (ANS$)
GOTO 1607

'assign color 0 (black) to pallete jar 0
1708 PALETTE 0,0

'subroutine to display all info again
GOSUB DISINFO

'define end of subroutine KORRECTEMP
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp             SUBROUTINE  NAME:  EXITCOR                         ppp
'ppp                                                                 ppp
'ppp         SUBROUTINE TO EXIT CORRECTION MODE                      ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
' Define EXITCOR as a SUBROUTINE
EXITCOR:

'subroutine to display all info again
GOSUB DISINFO

'define end of subroutine EXITCOR
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

```
'==========================================================
'==              ***** IMPORTANT INFO *****              ==
'==                                                      ==
'==              program name:   runmon.bas              ==
'==              backup prog :   runmon.bak              ==
'==                                                      ==
'==========================================================
'    PROGRAM WRITTEN:    4-16-89                          '
'                                                         '
'    PROGRAM MODIFIED:  5-17-89                           '
'==========================================================
'==  PROGRAM NAME:  runmon.bas                           ==
'==                                                      ==
'==    This subroutine calls subroutines that monitor :  ==
'==                                                      ==
'==  * six temperatures input as analog voltages         ==
'==  * pump rate input as digital voltages               ==
'==  * Hb content input as analog voltages               ==
'==  * Venous Oxygen Saturation in as analog voltages    ==
'==  * Arterial Oxygen Saturation in as analog voltages  ==
'==  * three timers                                      ==
'==  * current time                                      ==
'==                                                      ==
'==    Additionally, the ESCAPE KEY is set up to exit    ==
'==  the entire monitoring program in this subroutine.   ==
'==  Function Key 5 is set to return to the display      ==
'==  screen.  Function Key 6 is set to do graphs of      ==
'==  either two temperatures versus time or the Oxygen   ==
'==  consumption parameters (Hb content, AO2 sat, Vo2    ==
'==  sat, and Oxygen Consumption) versus time.           ==
'==                                                      ==
'==    Also, this subroutine calls subroutines to compute ==
'==  pump output, normal blood flow rate (using patient  ==
'==  body surface area and weight), and the oxygen       ==
'==  content.                                            ==
'==                                                      ==
'==========================================================
'define RUNMON as a subroutine
RUNMON:

'turn timer off
timer off

'set flag place%=3 for return from escape key
place%=3

'set palette color 0 to white (63)
palette 0,63

'set palette color 15 to black (0)
palette 15,0

'set text and background color to white
color 1,0

'clear the screen
cls

'initialize loop counter
loopcount%=0

'initialize flag to display current time
fflag%=0

'call subroutine to initialize for monitoring
GOSUB INITMON

'update current time every 0.75 seconds
on timer (.75) gosub currentime

'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  THE LABEL ==RUNIT== MARKS THE LOOP THAT THE PROGRAM
'  STAYS IN WHILE MONITORING
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
RUNIT:
'
'set start time equal to variable ttiimmee
ttiimmee=timer
'
'update pump frequency and compute blood
'flow rate
```

```
 gosub pumcal
'
'-------------------------------------------------------------
'goto subroutine hbOXSAT to get
'A/D conversion values and display digital values
'of HB, OXY SAT IN, and OXY SAT OUT.
 gosub hbOXSAT
'
'-------------------------------------------------------------
'turn event trapping off
$EVENT OFF
'
'-------------------------------------------------------------
'goto subroutine GETIT to get
'A/D conversion values and display digital values
'of temperature on screen
 gosub getit
'
'-------------------------------------------------------------
'turn event trapping on
$EVENT ON
'
'-------------------------------------------------------------
GOTO RUNIT
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'define end of subroutine RUNMON
return
'=============================================================

'include these programs to save space in
'turbo basic editor

$INCLUDE "c:\monitor\surfare.bas"
$INCLUDE "c:\monitor\hb.bas"
$INCLUDE "c:\monitor\6temp.bas"
$INCLUDE "c:\monitor\curtim.bas"
$INCLUDE "c:\monitor\atimer2.bas"
$INCLUDE "c:\monitor\addodoc.bas"
$INCLUDE "c:\monitor\digat.bas"
$INCLUDE "c:\monitor\daut.bas"
$INCLUDE "c:\monitor\pumcol.bas"
$INCLUDE "c:\monitor\initond.bas"
$INCLUDE "c:\mofitor\drawman.bas"
$INCLUDE "c:\monitor\tmpmanbx.bas"
$INCLUDE "c:\monitor\patmanbx.bas"
$INCLUDE "c:\monitor\hboxmon.bas"
$INCLUDE "c:\monitor\groph3.bas"



'=============================================================
'==              ***** IMPORTANT INFO *****                 ==
'==                                                         ==
'==              program name:  initend.bas                 ==
'==              backup prog :  initend.bak                 ==
'==                                                         ==
'=============================================================
'    PROGRAM WRITTEN:   4-16-89                             '
'                                                           '
'    PROGRAM MODIFIED:  5-17-89                             '
'=============================================================

'=============================================================
'==               SUBROUTINE:  INITMON                      ==
'==                                                         ==
'==  This subroutine sets up for monitoring of variables    ==
'==                                                         ==
'=============================================================
'define initmon as subroutine
initmon:

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'-------------------------------------------------------------
'stop function of unused function keys
key(4) stop
key(5) stop
key(6) stop
key(7) stop
'
'-------------------------------------------------------------
'turn unused function keys off
key(5) off
key(6) off
'
'-------------------------------------------------------------
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
 -------------------------------------------------------------
```

```
'set up trapping for function key 5
'when hit fk(5) will return to disinfo subroutine
'to display information and correction mode
on key(5) gosub back
'
'turn fk(5) on
key(5) on
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'set up trapping for function key 6
'when hit fk(6) will go to subroutine graph3 to
'graph parameters
on key(6) gosub GRAPH3
'
'turn fk(6) on
key(6) on
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

'locate and print function key menu at bottom of screen
'set color to blue text and white background
COLOR 6,0

'locate and print fk1
LOCATE 22,9
PRINT"F1=START T1"

'locate and print fk2
LOCATE 23,9
PRINT"F2=START T2"

'locate and print fk3
LOCATE 22,27
PRINT"F3=START T3"

'locate and print fk5
LOCATE 23,27
PRINT"F5=DISPLAY"

'locate and print fk6
LOCATE 22,2
PRINT"F6=GRAPH"

'locate and print fk8
LOCATE 23,2
PRINT"F8=UPDATE T3"

'locate and print fk9
LOCATE 22,8
PRINT"F9= UPDATE T2"

'locate and print fk10
LOCATE 23,8
PRINT"F10=UPDATE T1"

'locate and print escape key
LOCATE 23,66
PRINT"ESC TO EXIT"
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'turn event trapping off
$event off
'
'call subroutine PATDAT to get patient information
GOSUB PATDAT
'
'call subroutine to read pump data from pump.dat file
GOSUB PUMPDAT
'
'call subroutine to draw patient info box
gosub patmonbx
'
'call subroutine to draw temp monitoring box
gosub tmpmonbx
'
'call subroutine to draw box listing Hb content,
'VO2 sat, AO2 sat, and Oxygen consumption
gosub HbOxmon
'
'call subroutine to draw timer box
gosub timerbox
```

```
'
'-------------------------------------------------------------
'call subroutine to draw monitoring box listing
' pump type, calculated pump output, actual pump
' output, and normalized blood flow rates
gosub drawmon
'
'-------------------------------------------------------------
'call subroutine to initialize dt2801-a board
CALL init(&H2ac)
'
'-------------------------------------------------------------
'   call procedure to compute body surface area of patient
'   body surface area in square meters
CALL SurArea(height,htunit$,weight,wtunit$,surfarea,wate)
'
'-------------------------------------------------------------
'turn event trapping on
$event on
'
'-------------------------------------------------------------
'set up trapping for computer's internal clock
'turn timer on
TIMER ON
'
'-------------------------------------------------------------
'call subroutine to set up event trapping for timers
'set function keys
gosub atimer
'
'-------------------------------------------------------------
'define end of subroutine initmon
return
'
'=============================================================


'=============================================================
'==                SUBROUTINE NAME:  BACK                   ==
'==                                                         ==
'==    This subroutine returns to disinfo subroutine        ==
'==    to display all info and show correction mode.        ==
'==                                                         ==
'=============================================================
'define back as a subroutine
back:

'turn timer off
timer off

'stop function of all function keys
key(1) stop
key(2) stop
key(3) stop
key(4) stop
key(5) stop
key(6) stop
key(7) stop
key(8) stop
key(9) stop
key(10) stop

'turn all unused function keys off
key(1) off
key(2) off
key(3) off
key(4) off
key(5) off
key(6) off
key(7) off
key(8) off
key(9) off
key(10) off

'return to disinfo subroutine to display all info
'and get to correction mode
goto disinfo

'define end of subroutine back
return
'=============================================================




'=============================================================
'==             ***** IMPORTANT INFO *****                  ==
'==                                                         ==
'==             program name:  PATMONBX.bas                 ==
'==             backup prog :  PATMONBX.bak                 ==
```

```
'==                                                             ==
'=================================================================
'     PROGRAM WRITTEN:    5-08-89                                 '
'                                                                 '
'     PROGRAM MODIFIED:  5-08-89                                  '
'_________________________________________________________________'
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                           ppp
'ppp              SUBROUTINE  NAME:  PATMONBOX                 ppp
'ppp                                                           ppp
'ppp      SUBROUTINE TO DRAW PATIENT INFOMATION BOX            ppp
'ppp                                                           ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'SUBROUTINE TO DISPLAY PAT INFO BOX
PATMONBX:

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=3
BKCOL%=1
color txcol%,bkcol%

'define beginning and ending rows of menu box
Bbbegrow%=4
eendrow%=8

'define beginning and ending columns of menu box
Bbbegcol%=2
eendcol%=45

'call procedure MENUBOX to draw menu box
'call MENUBOX(Bbbegrow%,eendrow%,Bbbegcol%,eendcol%,TXCOL%,BKCOL%)

'locate and print spaces for blue background
locate bbbegrow%,bbbegcol%
print space$(48)
locate bbbegrow%+1,bbbegcol%
print space$(68)

'locate and print patient name title
LOCATE Bbbegrow%,Bbbegcol%
PRINT " NAME:  "

'locate and print patient name
LOCATE Bbbegrow%,Bbbegcol%+5
PRINT patname$+space$(1)

'locate and print title for hospital id number of patient
LOCATE Bbbegrow%+1,Bbbegcol%
PRINT " ID #:   "

'locate and print hospital id number of patient
LOCATE Bbbegrow%+1,Bbbegcol%+4
PRINT IDNUM$+space$(1)

'define end of subroutine PATMONBX return
return
'_________________________________________________________________


'=================================================================
'==               ***** IMPORTANT INFO *****                    ==
'==                                                             ==
'==               program name:  TMPMONBX.bas                   ==
'==               backup prog :  TMPMONBX.bak                   ==
'==                                                             ==
'=================================================================
'     PROGRAM WRITTEN:    4-25-89                                 '
'                                                                 '
'     PROGRAM MODIFIED:  5-08-89                                  '
'_________________________________________________________________'
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                           ppp
'ppp              SUBROUTINE  NAME:  TMPMONBOX                 ppp
'ppp                                                           ppp
'ppp   SUBROUTINE TO DRAW MONITORING TEMPERATURE BOX           ppp
'ppp                                                           ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'SUBROUTINE TO DISPLAY TEMP MONITOR INFO
TMPMONBX:
```

```
'set palette color 6 to yellow (62)
PALETTE 6,62

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=3
BKCOL%=2

'define beginning and ending rows of menu box
bbegrow%=3
endrow%=45

'define beginning and ending columns of menu box
bbegcol%=2
endcol%=66

'call procedure MENUBOX to draw menu box
call MENUBOX(bbegrow%,endrow%,bbegcol%,endcol%,TXCOL%,BKCOL%)

'define variable for display
bbegcol%=1

'set color as red text and yellow background
color 4,2

'assign temp names to array temp$
DIM TEMP$(5)
TEMP$(1)=TEMP1$
TEMP$(2)=TEMP2$
TEMP$(3)=TEMP3$
TEMP$(4)=TEMP4$
TEMP$(5)=TEMP5$
TEMP$(6)=TEMP6$

'assign abbreviations for temp units
IF TMPUNIT$="CELSIUS" THEN TMPU$="C" ELSE TMPU$="F"

'locate and print title of box
LOCATE bbegrow%,bbegcol%+4
PRINT " TEMPERATURE PROBES "

'locate and print NAME of temp probe and units or not
' connected message
LOCATE bbegrow%+2,bbegcol%+2
IF (TEMP$(1)="N/C") OR (TEMP$(1)="n/c") THEN
  PRINT "TEMP #1 IS NOT CONNECTED"
ELSE
  PRINT TEMP1$
  LOCATE bbegrow%+2,bbegcol%+65
  PRINT CHR$(248)+TMPU$
END IF

'locate and print NAME of temp probe and units or not
' connected message
LOCATE bbegrow%+3,bbegcol%+3
IF (TEMP$(2)="N/C") OR (TEMP$(2)="n/c") THEN
  PRINT "TEMP #2 IS NOT CONNECTED"
ELSE
PRINT TEMP2$
LOCATE bbegrow%+2,bbegcol%+33
PRINT CHR$(248)+TMPU$
END IF

'locate and print NAME of temp probe and units or not
' connected message
LOCATE bbegrow%+4,bbegcol%+34
IF (TEMP$(3)="N/C") OR (TEMP$(3)="n/c") THEN
  PRINT "TEMP #3 IS NOT CONNECTED"
ELSE
PRINT TEMP3$
LOCATE bbegrow%+4,bbegcol%+17
PRINT CHR$(248)+TMPU$
END IF

'locate and print NAME of temp probe and units or not
' connected message
LOCATE bbegrow%+5,bbegcol%+3
IF (TEMP$(4)="N/C") OR (TEMP$(4)="n/c") THEN
  PRINT "TEMP #4 IS NOT CONNECTED"
ELSE
PRINT TEMP4$
LOCATE bbegrow%+5,bbegcol%+25
PRINT CHR$(248)+TMPU$
END IF
```

```
'locate and print NAME of temp probe and units or not
' connected message
LOCATE bbegrow%+6,bbegcol%+3
IF (TEMP$(5)="N/C") OR (TEMP$(5)="n/c") THEN
  PRINT "TEMP #5 IS NOT CONNECTED"
ELSE
PRINT TEMP5$
LOCATE bbegrow%+6,bbegcol%+25
PRINT CHR$(248)+TMPU$
END IF

'locate and print NAME of temp probe and units or not
' connected message
LOCATE bbegrow%+7,bbegcol%+3
IF (TEMP$(6)="N/C") OR (TEMP$(6)="n/c") THEN
  PRINT "TEMP #6 IS NOT CONNECTED"
ELSE
PRINT TEMP6$
LOCATE bbegrow%+7,bbegcol%+25
PRINT CHR$(248)+TMPU$
END IF

'define end of subroutine TMPMONBX
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  HBOXMON.bas                ==
'==              backup prog :  HBOXON.bak                 ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   5-08-89                            '
'                                                          '
'    PROGRAM MODIFIED:  5-08-89                            '
'__________________________________________________________'
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                      ppp
'ppp            SUBROUTINE  NAME:  HBOXMON                ppp
'ppp                                                      ppp
'ppp         SUBROUTINE TO DRAW MONITORING BOX            ppp
'ppp      DISPLAYS HB, VO2, AO2, & O2 CONSUMPTION         ppp
'ppp                                                      ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'SUBROUTINE TO DISPLAY MONITOR INFO
HBOXMON:

'set palette color 3 to cyan (59)
palette 4,59

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=3
BKCOL%=25

'define beginning and ending rows of menu box
hbbegrow%=4
hbendrow%=22

'define beginning and ending columns of menu box
hbbegcol%=4
hbendcol%=5

'call procedure MENUBOX to draw menu box
call MENUBOX(hbegrow%,hbendrow%,hbegcol%,hbendcol%,TXCOL%,BKCOL%)

'set color to blue text and white background
color 5,14

'locate and print title for hemoglobin content
LOCATE hbbegrow%+1,hbbegcol%+3
PRINT "Hemoglobin Content =        gm/dl"

'set color to blue text and white background
color 9,15

'locate and print title for arterial oxygen saturation
LOCATE hbbegrow%+2,hbbegcol%+3
PRINT "Arterial Oxygen Saturation =    %"
```

```
'set color to blue text and white background
color 9,15

'locate and print title for venous oxygen saturation
LOCATE hbbegrow%+3,hbbegcol%+6
PRINT "Venous Oxygen Saturation   =    %"

'set color to blue text and white background
color 9,15

'locate and print title for oxygen consumption
LOCATE hbbegrow%+4,hbbegcol%+7
PRINT "Oxygen Consumption =     cc02/min"

'define end of subroutine hboxmon
return
'  ------------------------------------------------------------



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  DRAWMON.bas                ==
'==              backup prog :  DRAWMON.bak                ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   4-25-89                            '
'                                                          '
'    PROGRAM MODIFIED:  5-08-89                            '
'  ------------------------------------------------------  '
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                      ppp
'ppp             SUBROUTINE  NAME:  DRAWMON               ppp
'ppp                                                      ppp
'ppp          SUBROUTINE TO DRAW MONITORING BOX           ppp
'ppp                                                      ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'SUBROUTINE TO DISPLAY MONITOR INFO
DRAWMON:

'set palette color 3 to cyan (59)
palette 3,59

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=0
BKCOL%=1

'define beginning and ending rows of menu box
begrow%=21
endrow%=33

'define beginning and ending columns of menu box
begcol%=29
endcol%=45

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print title for blood pump in use
'if pump type is pulsatile, need to center on top line
IF pumptype$="PULSATILE" THEN
  COLOR 1,4
  LOCATE begrow%,begcol%+6
  PRINT SPACE$(1)+PUMPTYPE$;" BLOOD PUMP IN USE "

'else, pump type is roller, need to center on top line
ELSE
  COLOR 0,4
  LOCATE begrow%,begcol%+3
  PRINT SPACE$(1)+PUMPTYPE$;" BLOOD PUMP IN USE "

'define end of if block
END IF

'locate and print title for calculated pump output
COLOR 0,4
LOCATE begrow%+2,begcol%+4
PRINT "Calculated Pump Output =       ml/min"

'locate and print title for actual pump output
COLOR 0,4
```

```
LOCATE begrow%+3,begcol%+4
PRINT "Actual Pump Output =           ml/min"

'locate and print title for blood flow rate
COLOR 0,4
LOCATE begrow%+5,begcol%+4
PRINT "BLOOD FLOW RATE =          ml/min/kg"

'locate and print title for blood flow rate
COLOR 1,4
LOCATE begrow%+6,begcol%+2
PRINT "                =          ml/min/m*m"

'define end of subroutine drawmon
return
'-------------------------------------------------------------'



'============================================================
'==             ***** IMPORTANT INFO *****                 ==
'==                                                        ==
'==             program name:  ADDADOC.BAS                 ==
'==             backup prog :  ADDADOC.BAK                 ==
'==                                                        ==
'============================================================
' PROGRAM WRITTEN:   3-30-89                               ==
'                                                          ==
' PROGRAM MODIFIED:  5-17-89                               ==
'============================================================
'==  PROGRAM NAME:  ADDADOC.BAS                            ==
'==                                                        ==
'==          This program contains subroutines:            ==
'==                        adin                            ==
'==                        compute                         ==
'==                        computetemperature              ==
'==                        init                            ==
'==                        crash.                          ==
'==                                                        ==
'============================================================
'************************************************************
'*  Subroutine adin(reg1%,gain%,chan%,value%)               *
'*                                                          *
'*  Used to get one value 12 bit digital value from an      *
'*   analog input.                                          *
'*                                                          *
'*   Input    reg1%   Integer    Register of data           *
'*                               translation board          *
'*                               (base address)             *
'*            gain%   Integer    Gain  (0,1,2,3)            *
'*                               full scale = 10 / 2¢gain   *
'*            chan%   Integer    A/D channel  (0-7)         *
'*   Output   value%  Integer    12 bit value (0-4095)      *
'*                                                          *
'************************************************************
'**  This subroutine closely resembles that of page 10-4   **
'**  of the User Manual for DT2801-A Series Single Board   **
'**  Analog and Digital I/O Systems for the IBM Personal   **
'**  Computer.                                             **
'************************************************************


'define adin as a subroutine
SUB adin(reg1%,gain%,chan%,value%)

'define reg2, low%, high% s local variables to
' this subroutine
LOCAL reg1%,low%,high%

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg2%+1

'call procedure to initialize DT2801-A board
CALL init(&H2ac)

' this WAIT statement causes the program to loop
'  until bit 2 of the status register (ready bit)
'  is set.  The program then goes on to the next
'  executable statement
WAIT reg2%,&Hf
```

```
'this OUT command writes the command "C" (READ
' A/D IMMEDIATE) to reg2% (the command register)
OUT  reg2%, &Hf

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%,&H2,&H2

' this OUT command writes the command value of
'  gain% which is 0 (the gain code) to reg1%
'  which is the Data In Register
OUT  reg1%,gain%

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%,&H4,&H4

' this OUT command writes the command value of
'  chan% which is passed into the subroutine
'  (indentifies the channel to get data) to reg1%
'  which is the Data In Register
OUT reg2%,chan%

' this WAIT statement causes the program to loop
'  until bit 0 (Data Out Ready) or bit 2 (Ready)
'  in the Status Register is set.  Then, reads
'  Data Out Register
WAIT reg2%,&H5

' this INP statement causes the value of Low% to
'  be read from the Data Out Register.  This is the
'  low byte of the A/D conversion
Low% = INP(reg1%)

' this WAIT statement causes the program to loop
'  until bit 0 (Data Out Ready) or bit 2 (Ready)
'  in the Status Register is set.  Then, reads
'  Data Out Register
WAIT reg2%,&H5

' this INP statement causes the value of High% to
'  be read from the Data Out Register.  This is the
'  high byte of the A/D conversion
high% = INP(reg2%)

'  generate the 12 bit number
value%=low%+high%*256

' this WAIT statement causes the program to loop
'  until bit 2 (ready bit) of reg2% (the status register)
'  is set.  The program then goes on to the next
'  executable statement
WAIT reg2%,&H4

' check error flag
If (INP(reg2%) and &H80) then
  Print "error AD"
  'call crash subroutine to clear data translation
  '  board in case of error
  Call crash(reg1%)
End if

'define end of subroutine adin
End sub

'============================================================
'==                                                        ==
'==                PROCEDURE NAME:  COMPUTE                ==
'==                                                        ==
'==   THIS PROCEDURE COMPUTES DECIMAL VALUES OF ANALOG     ==
'==   VOLTAGE FROM ANALOG INPUTS                           ==
'==                                                        ==
'============================================================
' PROGRAM MODIFIED:  3-27-89
'************************************************************
'**  This subroutine closely resembles that of page 10-4  **
'**  of the User Manual for DT2801-A Series Single Board  **
'**  Analog and Digital I/O Systems for the IBM Personal  **
```

```
'**  Computer.                                                    **
'****************************************************************
'define procedure name as COMPUTE
'passed variables are:
'               GAIN%       = gain defined for DT2801-A
'               SG%         = binary value of A/D conversion
'               VALUE$      = decimal value of A/D conversion

SUB COMPUTE(GAIN%,SG%,VALUE$)

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+    compute decimal value of A/D conversion voltage          +
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'compute conversion to decimal value
factor$ = (10/4099)/GAIN%
'
'compute uni.volt value
uni.volts$ =  sg%*factor$
'
'compute analog voltage value input to desired channel
VALUE$ = uni.volts$*2 - (10/GAIN%)
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'define end of subroutine
END SUB
'===============================================================

'===============================================================
'==                                                           ==
'==         PROCEDURE NAME:  COMPUTETEMPERATURE               ==
'==                                                           ==
'==   THIS PROCEDURE COMPUTES AND UPDATES THE VALUES          ==
'==   OF TEMPERATURE AND DISPLAYS THEM ON THE SCREEN          ==
'==                                                           ==
'===============================================================
' PROGRAM MODIFIED:  3-27-89

'define procedure name as COMPUTETEMPERATURE
'passed variables are:
'               TMPUNIT$    = units of temperature
'               VALUE$      = decimal value of A/D conversion
'               TEMPVALUE$ = decimal value of temperature

SUB COMPUTETEMPERATURE(TMPUNIT$,VALUE$,TEMPVALUE$)

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+              compute value of temperature                   +
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'if temperature units are degrees celsius then
'compute value of temperature using celsius
'conversion factor for temperature detection device
IF TMPUNIT$="CELSIUS" THEN
   TEMPVALUE$=VALUE$/0.01999
'
'else compute value of temperature for farenheit
'units using farenheit conversion factor for
'temperature detection device
ELSE
     TEMPVALUE$=VALUE$/0.090
'
'end if block
END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'define end of subroutine
END SUB
'===============================================================

'***************************************************************
'*  Subroutine init(reg1%)                                     *
'*                                                             *
'*  Used to initialize the data translation A/D board          *
'*                                                             *
'* Input    reg1%                                             *
'* Integer Register of data translation board                  *
'*                                                             *
'***************************************************************
'**  This subroutine closely resembles that of page 10-4      **
'**  of the User Manual for DT2801-A Series Single Board      **
'**  Analog and Digital I/O Systems for the IBM Personal      **
```

```
'**  Computer.                                                    **
'***************************************************************
'define init as a procedure with reg1% as a passed variable
Sub init(reg1%)

'define reg2% and temp as local variables
'local variables are those particular to this
'procedure and have no connection with those of
'the same name outside of this procedure
Local reg2%,temp

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg1%+1

'check for error
If not ((Inp(reg2%) and &H70) = 0) then
  Print "Error while initializing board"
  Call crash(reg1%)
End if

'Stop the DT2801-A board operation
Out reg2% ,&HF

'empty the data out register
Temp= Inp(reg1%)

' this WAIT statement causes the program to loop
'  until bit 2 (ready bit) of reg2% (the status register)
'  is set.  The program then goes on to the next
'  executable statement
Wait reg2%,&H4

'this OUT command writes the command "1" (CLEAR
' ERROR) to reg2% (the command register)
Out reg2%,&H1

'define end of procedure
End sub
'===============================================================

'*************************************************
'*  Subroutine crash(reg1%)                      *
'*                                               *
'*  Used to clear the data translation A/D board  in  *
'*  the event of an error.                       *
'*                                               *
'*   Input     reg1%                             *
'*************************************************
'  This subroutine closely resembles that of page 9-38
'  of the User Manual for DT2801-A Series Single Board
'  Analog and Digital I/O Systems for the IBM Personal
'  Computer.
'*************************************************
'define procedure crash with reg1% as passed variable
Sub crash(reg1%)

'define reg2% as local variable
'local variables are those particular to this
'procedure and have no connection with those of
'the same name outside of this procedure
Local reg2%

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg1%+1

'Stop the DT2801-A board operation
Out reg2%,&HF

'empty the data out register
temp=Inp(reg1%)

'this OUT command writes the command "2" (READ ERROR
' REGISTER) to reg2% (the command register)
Out reg2%,&H2

' this WAIT statement causes the program to loop
'  until bit 0 (Data Out Ready) or bit 2 (Ready)
'  in the Status Register is set.  Then, reads
'  Data Out Register
Wait reg2%,&H5
```

```
'read the data out register
Error1=Inp(reg1%)

' this WAIT statement causes the program to loop
'  until bit 0 (Data Out Ready) or bit 2 (Ready)
'  in the Status Register is set.  Then, reads
'  Data Out Register
Wait reg2%,&H5

'read the data out register
Error2=Inp(reg1%)

'print error codes to screen
Print "Error 1 = ",error1
Print "Error 2 = ",error2
Print "Check for valid register"

'define end of procedure crash
End sub
'*********************************************************
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==                  program name:  SURFARE.BAS            ==
'==                  backup prog :  SURFARE.BAK            ==
'==                                                        ==
'============================================================
'PROGRAM WRITTEN:  2-9-89                                   '
'============================================================
'==  PROGRAM NAME:  SURFARE.INC                            ==
'==                                                        ==
'==    This program computes the patient's body surface    ==
'==  area.                                                 ==
'==                                                        ==
'==    This $INCLUDE program contains the PROCEDURES:      ==
'==                         SurArea                        ==
'==                                                        ==
'============================================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                       ppp
'ppp                 PROCEDURE NAME:  SurArea              ppp
'ppp                                                       ppp
'ppp  PROCEDURE TO COMPUTE BODY SURFACE AREA OF PATIENT    ppp
'ppp                                                       ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define SurArea as a PROCEDURE
SUB SurArea (height,htunit$,weight,wtunit$,surfarea,wate)

'If height is in inches (in), then convert
' to centimeters (cm).
   IF htunit$="INCHES"  THEN
         hite=2.54*height
   ELSE
         hite=height

'  Define end of IF block
   END IF

' If weight is in pounds (lbs), then convert
'   to kilograms (kg).
   IF wtunit$="POUNDS"  THEN
         wate=0.473592*weight
   ELSE
         wate=weight

'  Define end of IF block
   END IF

'  Computation of patient body surface
'    area using Dubois formula.
     surfarea=(wate)¢0.445*(hite)¢0.745*0.007184

locate 6,10
'  Print surfarea to screen
   PRINT "patient body surface area is ";surfarea

'  Define end of PROCEDURE SURFARE
```

```
   END SUB
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  atimer2.bas                ==
'==              backup prog :  atimer2.bak                ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   3-2-89                              '
'                                                           '
'    PROGRAM MODIFIED:  5-24-89                             '
'============================================================
'==  PROGRAM NAME:  atimer2.bas                            ==
'==                                                        ==
'==    This program contains timer functions activated     ==
'==  by function keys                                      ==
'==                                                        ==
'============================================================
atimer:

'start timer1
on key(1) gosub time1
key(1) on

'update timer1
on key(10) gosub timeupdate1
key(10) on

'start timer2
on key(1) gosub time2
key(1) on

'update timer2
on key(9) gosub timeupdate2
key(9) on

'start timer3
on key(3) gosub time3
key(3) on

'update timer3
on key(8) gosub timeupdate3
key(8) on

return
'============================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                             ppp
'ppp             SUBROUTINE  NAME:  TIMERBOX                     ppp
'ppp                                                             ppp
'ppp            SUBROUTINE TO DRAW TIMER BOX                     ppp
'ppp                                                             ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  Define TIMERBOX as a SUBROUTINE
TIMERBOX:

'set palette color 12 to red (60)
palette 12,60

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'define beginning row and column and end row and column
begrow%=12
endrow%=18    'min row is 1, max column is 23
begcol%=30
endcol%=79    'min column is 1, max column is 79

'assign beginning and ending rows and columns to variables
' for use in displaying current time
Cbegrow%=begrow%
Cendrow%=endrow%
Cbegcol%=begcol%
Cendcol%=endcol%
```

```
'find middle row and column
middlerow%=(endrow%-begrow%)/2+begrow%
middlecol%=(endcol%-begcol%)/2+begcol%


'define color of text as blue (1) and background
'as light blue (7)
color 1,7

'fill box with color of chosen background
for linespaces%=begrow%+1 to endrow%-4
locate linespaces%,begcol%
print space$(endcol%-begcol%)
next linespaces%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'top left corner
locate begrow%,begcol%
print chr$(201)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'border across top
for rr%=begcol%+1 to endcol%-1
locate begrow%,rr%
print chr$(275)
next rr%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw double line down from top border at col begcol%+16
locate begrow%,begcol%+16
print chr$(253)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw double line down at col begcol%+16
for cc%=begrow%+1 to endrow%-1
locate cc%,begcol%+16
print chr$(186)
next cc%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line down from top border at col 57
locate begrow%,begcol%+27
print chr$(239)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line down at col 57
for cc%=begrow%+1 to endrow%-1
locate cc%,begcol%+27
print chr$(159)
next cc%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line down from top border at col 68
locate begrow%,begcol%+38
print chr$(209)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line down at col 68
for cc%=begrow%+1 to endrow%-1
locate cc%,begcol%+38
```

```
print chr$(169)
next cc%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'top right corner
locate begrow%,endcol%
print chr$(187)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'right side
for cc%=begrow%+1 to endrow%-1
locate cc%,endcol%
print chr$(186)
next cc%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw lines across from side border at col 79
locate begrow%+3,endcol%
print chr$(183)
locate begrow%+5,endcol%
print chr$(184)
locate begrow%+7,endcol%
print chr$(181)
locate begrow%+9,endcol%
print chr$(181)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'bottom right corner
locate endrow%,endcol%
print chr$(188)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'bottom border
for rr%=endcol%-1 to begcol%+1 step -1
locate endrow%,rr%
print chr$(245)
next rr%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw lines up from bottom border at row 23
locate endrow%,begcol%+16
print chr$(212)
locate endrow%,begcol%+27
print chr$(217)
locate endrow%,begcol%+38
print chr$(217)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'bottom left corner
locate endrow%,begcol%
print chr$(210)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'left side
for cc%=endrow%-1 to begrow%+1 step -1
locate cc%,begcol%
print chr$(286)
next cc%
```

```
'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line across row 15
for rr%=begcol%+17 to endcol%-1
locate begrow%+3,rr%
print chr$(96)
next rr%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw crossings on row 15
locate begrow%+3,begcol%+16
print chr$(234)
locate begrow%+3,begcol%+27
print chr$(235)
locate begrow%+3,begcol%+38
print chr$(236)

'define color of text as blue (1) and background
'as light blue (2)
color 1,2

'draw double line at row 17
for rr%=begcol%+1 to endcol%-1
locate begrow%+5,rr%
print chr$(205)
next rr%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line from side border on row 17
locate begrow%+5,begcol%
print chr$(204)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw crossings on row 17
locate begrow%+5,begcol%+16
print chr$(216)
locate begrow%+5,begcol%+27
print chr$(226)
locate begrow%+5,begcol%+38
print chr$(236)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line from side border on row 19
locate begrow%+7,begcol%
print chr$(299)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line on row 19
for rr%=begcol%+1 to endcol%-1
locate begrow%+7,rr%
print chr$(296)
next rr%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw crossings on row 19
locate begrow%+7,begcol%+16
print chr$(215)
locate begrow%+7,begcol%+27
print chr$(197)
locate begrow%+7,begcol%+38
print chr$(197)
```

```
'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line from border on row 21
locate begrow%+9,begcol%
print chr$(199)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw single line on row 21
for rr%=begcol%+1 to endcol%-1
locate begrow%+9,rr%
print chr$(196)
next rr%

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'draw crossings on row 21
locate begrow%+9,begcol%+16
print chr$(215)
locate begrow%+9,begcol%+27
print chr$(197)
locate begrow%+9,begcol%+38
print chr$(123)

'define color of text as blue (1) and background
'as light blue (3)
color 1,3

'define timer row and timer column for printing of info
trow%=begrow%
tcol%=begcol%

'define text color as blue (1)
'and background as cyan (3)
color 1,3

'locate and print title timer #1
locate trow%+6,tcol%+1
print time1$

'define text color as red (4)
'and background as cyan (3)
color 4,3

'locate and print title timer #2
locate trow%+8,tcol%+1
print time2$

'define text color as black and
' background as cyan (3)
color 15,3

'locate and print title timer #3
locate trow%+10,tcol%+1
print time3$

'locate and print current time title
locate trow%+5,tcol%+5
print"CURRENT TIME"

'define text color as black and
' background as cyan (3)
color 15,3

'locate and print start time title
locate trow%+1,tcol%+19
print"start"
locate trow%+2,tcol%+20
print"time"

'define text color as black and
' background as cyan (3)
color 15,3

'locate and print update time title
locate trow%+4,tcol%+34
print"update"
```

```
locate trow%+2,tcol%+31
print"time"

'define text color as black and
' background as cyan (3)
color 15,3

'locate and print update elapsed title
locate trow%+1,tcol%+23
print"elapsed"
locate trow%+2,tcol%+33
print"time"

'define text color as black and
' background as cyan (3)
color 15,3

'locate and print units of timers
locate trow%+4,tcol%+18
print"Hr;Mn;Sec"
locate trow%+4,tcol%+43
print"Hr;Mn;Sec"
locate trow%+4,tcol%+32
print"Hr;Mn;Sec"

'Define end of subroutine TIMERBOX
RETURN
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp               SUBROUTINE  NAME:  TIME1                          ppp
'ppp                                                                 ppp
'ppp             SUBROUTINE TO START TIMER #1                        ppp
'ppp                                                                 ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TIME1 as a SUBROUTINE
time1:

'define text as blue (1) and background as light blue (3)
color 1,3

'reset parameter for timing after midnight
time11bmid=0

'assign start time of timer #1 to time1$
time1$=time$

'locate and print start time for timer #1
locate trow%+6,tcol%+58
print time1$

'assign timer to time1
time11=timer

'set color to previous setting
color txckr%,bkckr%

'define end of subroutine TIME1
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp           SUBROUTINE  NAME:  TIMEUPDATE1                        ppp
'ppp                                                                 ppp
'ppp           SUBROUTINE TO UPDATE TIMER #1                         ppp
'ppp                                                                 ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TIMEUPDATE1 as a SUBROUTINE
timeupdate1:

'location of timer #1 update
locate trow%+5,tcol%+49
color 1,3

'print timer #1 update time
print time$

'assign value of current time to variable atimer11
atimer11=timer
```

```
'compute elapsed time
timeupdate11=atimer11-time11

'reset parameters if after midnight
if timeupdate11<0 then
    time11bmid=86400-time11
    time11=0
end if

'compute elapsed time after midnight
  timeupdate11=atimer11-time11+time11bmid

        'compute hours
hr1 =timeupdate1/3600

'compute minutes
       min1=(timeupdate11-3600*fix(hr1))/60

'compute seconds
sec1=(min1 -fix(min1))/60

'round seconds to proper number
if sec1-int(sec1)<= 0.5 then
 sec1=int(sec1)+1
else
 sec1=int(sec1)
end if

'reset hours value if after midnight
if int(hr1)=24 then hr1=0

'assign whole numbers to variables for printing
hr1=fix(hr1)
min1=fix(min1)
sec1=(fix(sec1*10))/10

'CALL PROCEDURE TO PRINT TIME IN PROPER FORMAT
call prtupdate(hr1,min1,sec1,format1$)

'location of timer display on screen (row,column)
locate trow%+5,tcol%+33

'print timer on screen using specified format
print using format1$; hr1,min1,sec1

'set color to previous setting
color tdckr%,bkckr%

'define end of subroutine TIME1
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE  NAME:  TIME2                           ppp
'ppp                                                                 ppp
'ppp           SUBROUTINE TO START TIMER #2                          ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define TIME2 as a SUBROUTINE
time2:

'define text as RED (5) and background as cyan (3)
color 5,3

'reset parameter for timing after midnight
time22bmid=0

'assign start time of timer #2 to time2$
time22$=time$

'locate and print start time for timer #2
locate trow%+3,tcol%+68
print time2$

'assign time2 as timer
time22=timer

'set color to previous setting
color txckr%,bkckr%
```

```
'define end of subroutine time2
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp          SUBROUTINE  NAME:  TIMEUPDATE2                         ppp
'ppp                                                                 ppp
'ppp          SUBROUTINE TO UPDATE TIMER #2                          ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  Define TIMEUPDATE2 as a SUBROUTINE
timeupdate2:

'location of update time for timer #2
locate trow%+5,tcol%+19
color 4,3

'print update time of timer #2
print time$

'assign value of current time to variable atimer22
atimer22=timer

'compute elapsed time
timeupdate22=atimer22-time22+time22bmid

'reset parameters if after midnight
if timeupdate22<0 then
    time22bmid=86400-time22
    time22=0
end if

'compute elapsed time after midnight
  timeupdate22=atimer22-time22+time22bmid

'compute hours
hr2 =timeupdate22/3600

'compute minutes
        min2=(timeupdate2-3600*fix(hr2))/60

'compute seconds
sec2=(min2 -fix(min2 ))*60

'round seconds to proper number
if sec2-int(sec2)>= 0.5 then
 sec2=int(sec2)+1
else
 sec2=int(sec2)
end if


'reset hours value if after midnight
if int(hr2)=24 then hr2=0

'assign whole numbers to variables for printing
hr2=fix(hr2)
min2=fix(min2)
sec2=(fix(sec2*100))/10

'CALL PROCEDURE TO PRINT TIME IN PROPER FORMAT
call prtupdate(hr3,min3,sec3,format3$)

'location of timer display on screen (row,column)
locate trow%+6,tcol%+40

'print timer on screen using specified format
print using format2$; hr2,min2,sec2

'set color to previous setting
color txckr%,bkckr%

'define end of subroutine updatetime2
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE  NAME:  TIME3                           ppp
'ppp                                                                 ppp
'ppp           SUBROUTINE TO START TIMER #3                          ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

```
'  Define TIME3 as a SUBROUTINE
time3:

'define text as black and background as cyan
color 15,1

'reset parameter for timing after midnight
time33bmid=0

'assign start time of timer #3 to time3$
time33$=time$

'locate and print start time for timer #3
locate trow%+10,tcol%+18
print time33$

'define time3 as timer
time3=timer

'set color to previous setting
color txckr%,bkckr%

'define end of subroutine time3
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp          SUBROUTINE  NAME:  TIMEUPDATE3                           ppp
'ppp                                                                   ppp
'ppp          SUBROUTINE TO UPDATE TIMER #3                            ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  Define TIMEUPDATE3 as a SUBROUTINE
timeupdate3:

'location of update time for timer #3
locate trow%+9,tcol%+29
color 15,3

'print update time of timer #3
print time$

'assign value of current time to variable atimer33
atimer33=timer

'compute elapsed time
timeupdate33=atimer33-time33+time33bmid

'reset parameters if after midnight
if timeupdate33<0 then
    time3bmid=86400-time33
    time3=0
end if

'compute elapsed time after midnight
  timeupdate33=atimer33-time33+time33bmid

'compute hours
      hr3 =timeupdate33/3600

'compute minutes
        min3=(timeupdate33-3600*fix(hr3))/60

'compute seconds
      sec3=(min3 -fix(min3 ))*60

'round seconds to proper number
      if sec3-int(sec3)>= 0.5 then
         sec3=int(sec3)+1
else
         sec3=int(sec3)
end if


'reset hours value if after midnight
      if int(hr3)=24 then hr3=0

'assign whole numbers to variables for printing
      hr3=fix(hr3)
      min3=fix(min3)
      sec3=(fix(sec3*100))/10
```

```
'CALL PROCEDURE TO PRINT TIME IN PROPER FORMAT
call prtupdate(hr2,min2,sec2,format2$)

'location of timer display on screen (row,column)
locate trow%+9,tcol%+40

'print timer on screen using specified format
print using format3$; hr3,min3,sec3

'set color to previous setting
colar txtclr%,bkclr%

'define end of subroutine timeupdate3
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              PROCEDURE  NAME:  PRTUPDATE                        ppp
'ppp                                                                 ppp
'ppp         PROCEDURE TO PRINT UPDATE OF TIMERS                     ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'  Define PRTUPDATE as a PROCEDURE
'set up the proper print format for timers
sub prtupdate(hr,min,sec,format$)

'this if block prints leading zeroes for
'the appropriate number of hours, minutes,
'and seconds less than 10

'if hr, min, and sec are all less than 10
'then print leading zeroes for all three
if (hr<10) and (min<10) and (sec<10) then
        format$="0#;0#;0#"

'if hr and sec are less than 10
'then print leading zeroes for hr and sec
  elseif (hr<10) and (min>=10) and (sec<10)   then
        format$="0#;##;0#"

'if hr and min are less than 10
'then print leading zeroes for hr and min
  elseif (hr<10) and (min<10) and (sec>=10)  then
        format$="00;00;#0"

'if hr is less than 10
'then print a leading zero for hr
  elseif (hr<10) and (min>=10) and (sec>=10)  then
        format$="0#;##;00"

'if min and sec are less than 10
'then print leading zeroes for min and sec
  elseif (hr>=10) and (min<10) and (sec<10) then
      format$="##;0#;0#"

'if sec less than 10
'then print leading zero for sec
  elseif (hr>=10) and (min>=10) and (sec<10) then
        format$="##;##;0#"

'if none are less than 10
'then print no leading zeroes
  elseif (hr>=10) and (min>=10) and (sec>=10) then
        format$="##;##;##"

'if min is less than 10
'then print leading zero for min
  elseif (hr>=10) and (min<10) and (sec>=10) then
        format$="00;0#;0#"

'end of if block
end if

'define end of subroutine PRTUPDATE
end sub
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==              program name:  curtim.bas                 ==
'==              backup prog :  curtim.bak                 ==
'==                                                        ==
'============================================================
'    PROGRAM WRITTEN:   4-16-89                             '
'                                                           '
'    PROGRAM MODIFIED:  5-16-89                             '
'============================================================
'==  PROGRAM NAME:  curtim.bas == SUBROUTINE: currentime   ==
'==                                                        ==
'==    This subroutine calls updates the current time.     ==
'==                                                        ==
'============================================================
'define currentime as a subroutine
currentime:

'set flag to not print time during graphing
IF fflag%=13 THEN GOTO EENDIT

'set text color to light blue and background color to cyan
color 1,3

'locate text at proper row and column position
locate Cbegrow%+4,Cbegcol%+2

'print current time on screen
print time$

'set color to previous setting
color txckr%,bkckr%

'label to end subroutine
EENDIT:

'define end of subroutine currentime
return
'============================================================
```

```
'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name:  PUMCAL.BAS                ==
'==               backup prog :  PUMCAL.BAK                ==
'==                                                        ==
'============================================================
'   PROGRAM WRITTEN:   2-24-89                              '
'                                                           '
'   PROGRAM MODIFIED:  5-22-89                              '
'============================================================
'==  PROGRAM NAME:  PUMCAL.BAS                             ==
'==                                                        ==
'==    This program computes frequency representing        ==
'==  blood pump pulses per second and computes the pump    ==
'==  output and the normalized flow rate printing both     ==
'==  on the screen                                         ==
'==                                                        ==
'============================================================
'DEFINE PUMCAL AS A SUBROUTINE
pumcal:

'call subroutine to read rate of blood pump
GOSUB FREQ

'call subroutine to compute pump output
GOSUB PumpOutput

'call subroutine to compute normalized blood flow rate
GOSUB NormBlood

' define end of subroutine pumcal
 return

'============================================================
'==                                                        ==
'==              SUBROUTINE NAME: FREQ                     ==
'==                                                        ==
'==       subroutine to read rate of blood pump            ==
```

```
'==                                                          ==
'============================================================
'define FREQ as a subroutine
FREQ:

'initialize counters
CHANGE%=0

'specify digital port 0 being set for input of pump rate
port%=0

'call procedure to set port for input
CALL setdigit(rag1%,port%)

'assign time prog started to a variable for
'use in calculations
timestart=timer

'if block used to read pulses for approximately
'one second
15  IF TIMER-TIMESTART < 1 THEN
    'call procedure to read input from digital port
        CALL readdig(&H2ec,port%,change%,data.value%)
 GOTO 25
    END IF

'compute time data collection stopped
timestop=timer-timestart

'compute frequency of input
rate=(change%/(timestop))

'locate begrow%,begcol%
'print using "###.#";rate

'define end of subroutine FREQ
RETURN
'============================================================
'============================================================

'============================================================
'==                                                        ==
'==               SUBROUTINE NAME: PUMPDAT                 ==
'==                                                        ==
'==    subroutine to read pump data from pump.dat file     ==
'==                                                        ==
'============================================================
'define PUMPDAT as a subroutine
PUMPDAT:

'open file pump.dat to read pump data as #10
OPEN "c:\monitor\pump.dat" FOR INPUT AS #10

'read type of pump from pump.dat file
INPUT #10 ,pumptype$

'if pumptype$ is ROLLER then read data for roller pump
IF pumptype$="ROLLER" THEN
INPUT #10, numberollers,tubediameter,
tubediaunit$,arclength,arclenunit$,pumpvol

'else pumptype$ is PULSATILE so read stroke volume of pump
ELSE
INPUT #10, stokevol

'end if block
END IF

'close file pump.dat #10
CLOSE #11

'define end of subroutine PUMPDAT
RETURN
'============================================================
'============================================================

'============================================================
'==                                                        ==
'==               SUBROUTINE NAME: PumpOutput              ==
'==                                                        ==
'==          subroutine to compute pump output             ==
'==                                                        ==
'============================================================
```

```
'define PumpOutput as a subroutine
PumpOutput:

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'check type of blood pump
IF pumptype$="ROLLER" THEN
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  If tube diameter is in inches (in),
'  then convert to centimeters (cm).
   IF tubediaunit$="INCHES" THEN
 tubedia=2.19*tubediameter
   ELSE
         tubedia=tubediameter

'  Define end of IF block
   END IF
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  If arc length is in inches (in),
'  then convert to centimeters (cm).
   IF arclenunit$="INCHES" THEN
         arclen=2.54*arclength
   ELSE
         arclen=arclength

'  Define end of IF block
   END IF
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  computation of roller pump volume in cubic centimeters
   pumpvol=((numberollers)/(arclen)*(3.1416)*(tubedia)¢2)/4
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  computation of roller pump rate in pulses per minute
'  pick-up device produces 100 low-going pulses per
'  revolution of the roller
'
'     roller pump rate = [rate (cycles/sec)]*
'                        [60 seconds/minute]/
'                          [100 pulses/rev]
'

   pumprate=(rate)*(60)/(100)
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  ---------------------------------------------------------------

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'else pump type is PULSATILE
ELSE

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  computation of pulsatile pump rate in pulses per minute
'  pick-up device produces 1 low-going pulse per pulse
'  of the diaphragm
'
'   pulsatile pump rate = [rate (cycle/sec)]*[60 seconds/minute]/
'                         [1 pulse/pulse of diaphragm]*

   pumprate=(rate)*(60)
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'   pump volume of pulsatile pump is equal
'   to stroke volume in cubic centimeters
   pumpvol=strokevol
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'  ---------------------------------------------------------------

'Define end of IF block
END IF
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'   computation of pump output in milliliters
'   per minute
'
'     1 cubic centimeter = 1 milliliter
'
'     pump output = (pump volume) * (pump rate)
```

```
   pumpout=(pumpvol)*(pumprate)
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'assign text color to tkckr%
txckr%=0

'assign background color to bkckr%
bkckr%=3

'define color for display
'blue text (1) and light blue background (3)
color txckr%,bkckr%

'turn timer off
timer off

'locate and print calculated pump output
locate begrow%+12,begcol%+76
print using "####";pumpout

'turn timer on
timer on

'define end of subroutine PumpOutput
RETURN
'============================================================
'============================================================

'============================================================
'==                                                        ==
'==              SUBROUTINE NAME: NormBlood                ==
'==                                                        ==
'==   subroutine to compute normalized blood flow rate     ==
'==                                                        ==
'============================================================
'define NormBlood as a subroutine
NormBlood:

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'   computation of normailzed blood flow rate using
'   two methods
'
'      1.  NBFRSA=(pump output)/(body surface area)
'            units:  ml/min/m^2
'
'      2.  NBFRWT=(pump output)/(patient weight)
'            units:  ml/min/kg
'
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'/////////////////////////////////////////////////////////////
'   compute NBFRSA using body surface area in
'   units of ml/min/m^2
NBFRSA=pumpout/surforea
'/////////////////////////////////////////////////////////////

'/////////////////////////////////////////////////////////////
'   compute NBFRWT using patient weight in
'   units of ml/min/kg
NBFRWT=pumpout/wote
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

'turn timer off
timer off

'locate and print normalized blood flow rate using
'surface area equation
locate begrow%+5,begcol%+19
for1$="#####.#"
PRINT USING for1$;NBFRSA

'turn timer on
timer on

'turn timer off
timer off

'locate and print normalized blood flow rate using
'weight of patient
locate begrow%+86,begcol%+79
for2$="####.#"
PRINT USING for2$;NBFRWT

'turn timer on
timer on
```

```
'define end of subroutine NormBlood
RETURN
'============================================================
'============================================================

'============================================================
'==                                                        ==
'==              SUBROUTINE NAME: PATDAT                   ==
'==                                                        ==
'==     subroutine to read patient information from        ==
'==     patient.dat file                                   ==
'==                                                        ==
'============================================================
'define PATDAT as a subroutine
PATDAT:

'open file patient.dat to read patient information as #11
OPEN "C:\monitor1\patient.dat" FOR INPUT AS #11

'read type of pump from patient.dat file
INPUT #11,patname$,sex$,idnum$,age,ageunit$,
height,htunit$,weight,wtunit$

'close file patient.dat #11
CLOSE #11

'define end of subroutine PATDAT
RETURN
'============================================================
'============================================================




'============================================================
'==             ***** IMPORTANT INFO *****                 ==
'==                                                        ==
'==                program name:  HB.BAS                   ==
'==                backup prog :  HB.BAK                   ==
'==                                                        ==
'============================================================
'   PROGRAM WRITTEN:   3-28-89                              '
'                                                           '
'   PROGRAM MODIFIED:  5-22-89                              '
'============================================================
'==  PROGRAM NAME:  HB.BAS == subroutine name: hbOXSAT     ==
'==                                                        ==
'==    This program reads one analog input that has        ==
'==  voltages representing Hemoglobin content, arterial    ==
'==  oxygen saturation, and venous oxygen saturation       ==
'==  multiplexed into the input.  Additionally, this       ==
'==  program reads digital port 1 to distinguish which     ==
'==  voltage represents what quantity!                     ==
'==    Also, this program reads the analog input for       ==
'==  flowrate and calculates the oxygen consumption .      ==
'==                                                        ==
'============================================================
'define hbOXSAT as a subroutine
hbOXSAT:

'call subroutine to read analog values from DT2801-A
'reads values for hemoglobin, venous oxygen
'saturation, arterial oxygen saturation, and
'the flow rate
GOSUB READO2

'call subroutine to compute decimal values for
'hemoglobin, venous oxygen saturation,
'arterial oxygen saturation, and the flow rate
GOSUB COMPUTEDECIMAL

'call subroutine to compute oxygen consumption
GOSUB O2CONSUMP


'define end of subroutine hbOXSAT
return
'============================================================

'============================================================
'==                                                        ==
'==              SUBROUTINE NAME: READO2                   ==
```

```
'==                                                          ==
'==       subroutine to read analog voltage for              ==
'==       hemoglobin, venous oxygen saturation,              ==
'==       arterial oxygen saturation, and flow rate          ==
'==                                                          ==
'==========================================================
'define READO2  as a subroutine
READO2:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                          +
'+               DEFINITION OF VARIABLES                    +
'+                                                          +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                          +
'+                                                          +
'+      specify digital port being set for input            +
'+      used in procedures setdig and readdig               +
'+      this port is used to distinguish between            +
'+      HB, Oxygen sat in, and Oxygen sat out               +
   port%=1
'+                                                          +
'+      define gain code for DT2801-A as GANE%              +
'+      used in procedure adin                              +
   GANE%=0
'+                                                          +
'+      define channel from which to read analog input      +
'+      used in procedure adin                              +
   CHANNEL%=7
'+                                                          +
'+      define gain for specific gain code as GAIN%         +
'+      used in procedure compute                           +
   GAIN%=2^GANE%
'+                                                          +
'+      assign start time to a variable for later use       +
   begintime=timer
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'turn event trapping off
$event off

'call procedure to set digital port 1 for output
CALL digout(&H2ec)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+           read analog values from DT2801-A               +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'get values for oxygen consumption calculation
'read analog channel 7 (multiplexed input) so
'that all three data sets (HB, VO2SAT, AO2SAT) are input.
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call procedure to send digitout% to port 1
'closes switch so that HB data is input
digitout%=158
CALL writdig(&H2ec,digitout%)

'pause 5.25 msec
for xxx%=1 to 2000
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to get a/d conversion value from
'channel 7 (multiplexed input)
CALL adin(&H2ec,GANE%,CHANEL%,BINARYVALUE%)

   'assign hb data in binary form to array hb
   hb%=binaryvalue%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 5.25 msec
for xxx%=1 to 2000
next xxx%

'call procedure to send digitout% to port 1
'set all switches open
digitout%=0
CALL writdig(&H2ec,digitout%)

'pause 10.5 msec
for xxx%=1 to 40
next xxx%
```

```
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call procedure to send digitout% to port 1
'closes switch so that VO2 data is input
digitout%=95
CALL writdig(&H2ec,digitout%)

'pause 5.25 msec
for xxx%=1 to 2000
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to get a/d conversion value from
'channel 7 (multiplexed input)
CALL adin(&H2ec,GANE%,CHANNEL%,BINARYVALUE%)

   'assign VO2SAT data in binary form to
        'array vsat
   vsat%=binaryvalue%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 5.25 msec
for xxx%=1 to 2000
next xxx%

'call procedure to send digitout% to port 1
'set all switches open
digitout%=0
CALL writdig(&H2ec,digitout%)
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 10.5 msec
for xxx%=1 to 4000
next xxx%

'call procedure to send digitout% to port 1
'closes switch so that AO2 data is input
digitout%=63
CALL writdig(&H2ec,digitout%)

'pause 5.25 msec
for xxx%=1 to 2000
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to get a/d conversion value from
'channel 7 (multiplexed input)
CALL adin(&H2ec,GANE%,CHANNEL%,BINARYVALUE%)

   'assign AO2SAT data in binary form to
        'array asat
   asat%=binaryvalue%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 5.25 msec
for xxx%=1 to 2000
next xxx%

'call procedure to send digitout% to port 1
'set all switches open
digitout%=0
CALL writdig(&H2ec,digitout%)
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   'call subroutine to get a/d conversion value from
   'channel 6 (flow rate from flow meter)
      CALL adin(&H2ac,GANE%,6,FLOWRATEBINARY%)
'==========================================================
'turn event trapping on
$event on
'==========================================================
'define end of subroutine READO2
RETURN
'==========================================================

'==========================================================
'==                                                      ==
'==           SUBROUTINE NAME: COMPUTEDECIMAL            ==
'==                                                      ==
'==       subroutine to compute decimal values for       ==
'==       hemoglobin, venous oxygen saturation,          ==
'==       arterial oxygen saturation, and the flow rate  ==
'==                                                      ==
'==========================================================
```

```
'define COMPUTEDDECIMAL  as a subroutine
COMPUTEDECIMAL:

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'define constants for conversion of data
   '10 gm/dl per volt for hemoglobin content
   hbcon%=10

   '100 percentage points per volt for oxygen sat
   vsatcon%=100
   asatcon%=100

   'set flowcon% by calculated pump out
   IF PUMPOUT > 1000 THEN
   '1 liter per volt for flowrate
     FLOW=1
   '10 liters per min for flowrate conversion
     FLOWCON=10
   ELSE
        '0.5 liters per volt for flowrate
          FLOW=0.5
   '1 liter per min for flowrate conversion
     FLOWCON=1
   'define end of if block
   END IF

'assign text color to txckr%
txckr%=18

'assign background color to bkckr%
bkckr%=14

'turn timer off
timer off

'locate and print flowmeter scale setting
color txckr%,bkckr%
locate 25,4
print using " SET FLOWMETER SCALE TO ####";flow*1000
locate 25,40
print" ml/v."

'turn timer on
timer on

'assign text color to txckr%
txckr%=12

'assign background color to bkckr%
bkckr%=0

'change color to what it was
color txckr%,bkckr%

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+        compute decimal value of analog voltages          +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'read analog values from memory locations and send to
'compute procedure to calculate digital values
'
'computing values for hemoglobin, venous oxygen saturation
'arterial oxygen saturation, and the flow rate
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'voltage representing hemoglobin (hb)
CALL COMPUTE(GAIN%,hb%,hb$)

'compute hemoglobin content in units of gram/decaliter
'using conversion factor from sensor
hb$=hbcon%*hb$
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'voltage representing VO2SAT
CALL COMPUTE(GAIN%,vsat%,vsat$)
```

```
'compute percent venous oxygen saturation
'using conversion factor from sensor
vsat$=vsatcon%*vsat$
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'voltage representing AO2SAT
CALL COMPUTE(GAIN%,asat%,asat$)

'compute percent arterial oxygen saturation
'using conversion factor from sensor
asat$=asatcon%*asat$
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'flow rate
CALL COMPUTE(GAIN%,FLOWRATEBINARY%,FLOWRATE$)

'compute flowrate in units of liter per minute
'using conversion factor from sensor
FLOWRATE$=FLOWCON*FLOWRATE$
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'============================================================
'define end of subroutine
RETURN
'============================================================

'============================================================
'==                                                        ==
'==              SUBROUTINE NAME: O2CONSUMP                ==
'==                                                        ==
'==       subroutine to compute oxygen consumption         ==
'==                                                        ==
'============================================================
'define O2CONSUMP  as a subroutine
O2CONSUMP:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+  oxygen consumption is computed using the equation:     +
'+                                                         +
'+  Oxygen Consumption = (Flowrate)*(Hb content)*          +
'+                       (1.39 cc Oxygen/gm Hb)*           +
'+                          (change in Oxygen Saturation)  +
'+                                                         +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                         +
'+                DEFINITION OF VARIABLES                  +
'+                                                         +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                         +
'+    O2C             = oxygen consumption                 +
'+                        (cc oxygen per min)              +
'+    hb$             = hemglobin content (gm/dl)          +
'+      vsat$         = venous oxygen saturation (%)       +
'+      asat$         = arterial oxygen saturation (%)     +
'+      deltasat$     = asat$(x)-vsat$(x)                  +
'+      flowrate$     = flow rate (liters/min)             +
'+      oconst        = 1.39 cc oxygen per gm Hb           +
'+      convertHB     = 10 dl/l                            +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'define constants
oconst=1.39
convertHB=10

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+              compute oxygen consumption                 +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'compute delta oxygen saturation
deltasat$=(asat$-vsat$)/100

'compute oxygen consumption
O2C=flowrate$*hb$*convertHB*oconst*deltasat$

'assign text color to txckr%
txckr%=2
```

```
'assign background color to bkckr%
bkckr%=1

'set color for display
color txckr%,bkckr%

'turn timer off
timer off

'specify location of actual pump output
'locate begrow%+3,begcol%+27
locate 17,66

'print actual pump output
if flow=1 and flowrate#>9.9 then flowrate#=9.999
print using "####"; flowrate#*1000

'assign text color to txckr%
txckr%=9

'assign background color to bkckr%
bkckr%=1

'set color for display
color txckr%,bkckr%

'locate and print hemoglobin content
locate hbbegrow%+1,hbbegcol%+24
print using "##.#"; hb#

'set color for display
color txckr%,bkckr%

'locate and print arterial oxygen saturation
locate hbbegrow%+2,hbbegcol%+11
if asat#>99.1 then asat#=92
print using "##"; asat#

'set color for display
color txckr%,bkckr%

'locate and print venous oxygen saturation
locate hbbegrow%+3,hbbegcol%+21
if vsat#>99.1 then vsat#=99
print using "##"; vsat#

'set color for display
color txckr%,bkckr%

'locate and print oxygen consumption
locate hbbegrow%+4,hbbegcol%+23
if o2c>999 then o2c=999
print using "###"; O2C

'turn timer on
timer on
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'==========================================================
'define end of subroutine
RETURN
'==========================================================


'==========================================================
'==              ***** IMPORTANT INFO *****              ==
'==                                                      ==
'==              program name:  DOUT.BAS                 ==
'==              backup prog :  DOUT.BAK                 ==
'==                                                      ==
'==========================================================
'PROGRAM WRITTEN:    2-09-89                              '
'                                                         '
'PROGRAM MODIFIED:  5-19-89                               '
'==========================================================
'==  PROGRAM NAME:  DOUT.BAS                             ==
'==                                                      ==
'==    This subroutine calls subroutines to set digital  ==
'==  port 1 for ou put and sends a pulse to this port    ==
'==  to start the 555 timer for HB, ASAT, and VSAT       ==
'==  ifput to analog channel 7.                          ==
```

```
'==                                                          ==
'==========================================================
'************************************************************
'*  Subroutine digout(reg1%)                                *
'*                                                          *
'*  Used t  set digital port for output                     *
'*                                                          *
'*                                                          *
'*   Input    rag1%   Integer    Register of data            *
'*                               translation board          *
'*                               (base address)             *
'*                                                          *
'*                                                          *
'*                                                          *
'************************************************************
'**  This subroutine closely resembles that of page 12-3   **
'**  of the User Manual for DT2801-A Series Single Board   **
'**  Analog and Digital I/O Systems for the IBM Personal   **
'**  Computer.                                             **
'************************************************************
'subroutine TO SET DIGITAL PORT FOR OUTPUT
SUB digout(rag1%)

'define reg2 local variable to
' this subroutine
LOCAL reg2%

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg1%+1

'stop function of DT2801-A board
OUT reg2%, &HF

'call subroutine to initialize board
CALL init(&H2ac)

'wait until DATA IN FULL flag is clear
wait reg2%,&H2,&H2

'wait until READY flag is set
wait reg2%,&H5

'write the SET DIGITAL PORT FOR OUTPUT command to
' command register
out reg2%,&H4

'wait until DATA IN FULL flag is clear
wait reg2%,&H3,&H3

'write the DIGITAL PORT SELECT byte (1) to the
' data register
out reg1%,1

'wait until DATA IN FULL flag is clear
WAIT reg2%,&H2,&H2

'wait until READY flag is set
WAIT reg2%,&H5

'check for error
if (INP(reg2%) and &H80) then
   print"error "
   'call procedure to clear board in case of error
   call crash(reg1%)
'define end of if block
end if

'define end of sub digout
end sub
'************************************************************

'************************************************************
'*  Subroutine writdig(reg1%,digitout%)                     *
'*                                                          *
'*  Used to write value to digital port                     *
'*                                                          *
'*                                                          *
'*   Input    reg1%      Integer    Register of data        *
'*                                  translation board       *
'*                                  (base address)          *
'*                                                          *
'*            digitout%  integer    number between 0 and    *
'*                                  255 to be output to     *
```

```
'*                                    digital port          *
'**********************************************************
'**  This subroutine closely resembles that of page 12-13 **
'**  of the User Manual for DT2801-A Series Single Board  **
'**  Analog and Digital I/O Systems for the IBM Personal  **
'**  Computer.                                            **
'**********************************************************
'subroutine to write digital output byte
sub writdig(reg1%,digitout%)

'define reg2 local variable to
' this subroutine
LOCAL reg2%

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg1%+1

'stop function of DT2801-A board
OUT reg2%, &Hc

'call subroutine to initialize board
CALL init(&H2ac)

'wait until DATA IN FULL flag is clear
WAIT reg2%,&H3,&H3

'wait until READY flag is set
WAIT reg2%,&H3

'write WRITE DIGITAL PORT IMMEDIATE command
' to command register
out reg2%,&H7

'wait until DATA IN FULL flag is clear
wait reg2%,&H4,&H2

'write DIGITAL PORT SELECT byte to the data in register
out reg1%,1

'wait until DATA IN FULL flag is clear
wait reg2%,&H2,&H2

'write DIGITAL PORT DATA VALUE byte to data in register
out reg1%,digitout%

'wait until DATA IN FULL flag is clear
wait reg2%,&H2,&H2

'wait until READY flag is set
wait reg2%,&H4

'check for error
if (INP(reg2%) and &H8) then
   print"error "
   'call procedure to clear board if error
   call crash(reg1%)
'define end of if block
end if

'define end of procedure writdig
end sub
'**********************************************************



'==========================================================
'==              ***** IMPORTANT INFO *****               ==
'==                                                       ==
'==               program name:  DIGIT.BAS                ==
'==               backup prog :  DIGIT.BAK                ==
'==                                                       ==
'==========================================================
'== PROGRAM WRITTEN:   3-27-89                            ==
'==                                                       ==
'== PROGRAM MODIFIED:  3-30-89                            ==
'==========================================================
'==  PROGRAM NAME:  DIGIT.BAS                             ==
'==                                                       ==
'==          This program contains subroutines:           ==
'==                          setdigit                     ==
'==                          readdig                      ==
```

```
'==                                                          ==
'============================================================
'************************************************************
'*            PROCEDURE NAME: setdigit(reg1%,port%)         *
'*                                                          *
'*       procedure to set a digital port for input          *
'*                                                          *
'*       Input:   reg1%   Integer    Register of data       *
'*                                   translation board      *
'*                                   (base address)         *
'*                                                          *
'*                port%   Integer    digital port select    *
'*                                   0 selects port 0       *
'*                                   1 selects port 1       *
'*                                   2 selects port 0       *
'*                                     and port 1           *
'*                                                          *
'************************************************************
'************************************************************
'**  This subroutine closely resembles that of page 12-3   **
'**  of the User Manual for DT2801-A Series Single Board   **
'**  Analog and Digital I/O Systems for the IBM Personal   **
'**  Computer.                                             **
'************************************************************
'define setdigit as a procedure with reg1% and port% as
' passed variable
SUB setdigit(reg1%,port%)

'define reg2 as local variable to this subroutine
LOCAL reg2%

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg1%+1

'this OUT command writes the command F (stop command)
' to reg2% (command register).  This command returns
' the DT2801-A to the ready state
OUT reg2%, &H3

'call the procedure to initialize the DT2801-A board
CALL init(&H2ac)

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%,&H4,&H2

' this WAIT statement causes the program to loop
'  until bit 2 of the status register (ready bit)
'  is set.  The program then goes on to the next
'  executable statement
WAIT reg2%,&H4

'this OUT command writes the command 4 (set digital
'  port for input) to reg2% (command register).
'  this command sets the DT2801-A to the ready state
OUT reg2%,&HF

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%, &H4,&Hf

'this OUT command selects the digital port
'  being set for digital input as port%
'  and writes this port number to the data
'  in register
OUT reg1%,part%

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%,&H4,&H4

' this WAIT statement causes the program to loop
'  until bit 2 of the status register (ready bit)
'  is set.  The program then goes on to the next
```

```
'  executable statement
WAIT reg2%,&H3

' check for error by checking status error flag
if (INP(reg2%) and &H80) then
print"error "
call crash(reg1%)
end if

' define end of procedure
end sub

'*********************************************************
'* PROCEDURE NAME: readdig(reg1%,port%,count%,data.value%) *
'*                                                       *
'*       procedure to read digital input byte at         *
'*       a specified port                                *
'*                                                       *
'*       Input    reg1%   Ifteger    Register of data     *
'*                                    ranslation board   *
'*                                    (base address)     *
'*                                                       *
'*                port%   Ifteger    dygital pgrt sdlect  *
'*                                    0 sedgcts port 0   *
'*                                    1 selects por  1   *
'*                                                       *
'*       Output   changd% Integer    pu se counter        *
':                                                       *
'*                data.value%        port valwe           *
'*                                                       *
'*********************************************************
'**  this subroutine closely resembles that of page 12-9  **
'**  of the User Manual for DT2801-A Series Single Board  **
'**  Analog and Digital I/O systems for the IBM Personal  **
'**  computer                                             **
'*********************************************************
'define procedure readdig with reg1%, port%, and count% as
' passed variables
SUB readdig(reg1%,part%,change%,data.value%)

'define reg2, low%, high% as local variables to
' this subroutine
LOCAL reg2%

'define reg2% as base address +1 which defines
'  address of command register and status register
reg2%=reg1%+5


'this OUT command writes the command F (stop command)
' to reg2% (command register).  This command returns
' the DT2801-A to the ready state
OUT reg2%, &Hc

'call the procedure to initialize the DT2801-A board
'CALL init(&H2ac)

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg1%,&H6,&H6

' this WAIT statement causes the program to loop
'  until bit 2 of the status register (ready bit)
'  is set.  The program then goes on to the next
'  executable statement
WAIT reg2%,&H4

'this OUT command writes the command 6 (read digital
' input immediate) to reg2% (command register).
OUT reg2%, &H6

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%, &H3,&H3

' this OUT command identifies the port from which
'  the data is to be read as port% and sends it
```

```
'  to the data in register
OUT reg2%,port%

' this WAIT statement causes the program to loop
'  until bit 0 (DATA OUT READY flag) of reg2% (the
'  status register) is set.
WAIT reg2%, &H6

' set previous value at reg1% equal to dataold%
DATAOLD%=DATA.VALUE%

' this INP statement reads the data byte from the
'  data out register
DATA.VALUE%=INP(reg1%)

' this WAIT statement causes the program to loop
'  until bit 1 (DATA IN FULL flag) of reg2% (the
'  status register) is clear. This guarantees
'  that the command register is empty and can
'  legally have a command written to it.
WAIT reg2%,&H2,&H2

' this WAIT statement causes the program to loop
'  until bit 2 of the status register (ready bit)
'  is set.  The program then goes on to the next
'  executable statement
WAIT reg2%,&H4

' check error flag of status register
IF (INP(reg2%) and &H8a) THEN
PRINT"error "
CALL crash(reg2%)
END IF

' increment counter if new value in reg1% is different
'  from the old value in reg1%
IF DATAOLD%<>DATA.VALUE% THEN CHANGE%=CHANGE%+1

' define end of subroutine
END SUB
'*****************************************************************



'===============================================================
'==              ***** IMPORTANT INFO *****                   ==
'==                                                           ==
'==                 program name:  6TEMP.BAS                  ==
'==                 backup prog :  6TEMP.BAK                  ==
'==                                                           ==
'===============================================================
' PROGRAM WRITTEN:   3-27-89                                   '
'                                                              '
' PROGRAM MODIFIED:  5-21-89                                   '
'===============================================================
'==  PROGRAM NAME:  6TEMP.BAS                                 ==
'==                                                           ==
'==    This program displays analog voltage that is           ==
'==  input to the desired channel                             ==
'==                                                           ==
'===============================================================
'===============================================================
'==                                                           ==
'==                  SUBROUTINE NAME:  GETIT                  ==
'==                                                           ==
'==   THIS SUBROUTINE GETS THE A/D CONVERSION VALUES FOR      ==
'==   THE SIX TEMPERATURE PROBES                              ==
'==                                                           ==
'===============================================================

'define subroutine GETIT
GETIT:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'              VARIABLE DEFINITIONS                           +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'define gain code for DT2801-A as GANE%
     GANE%=0

'define gain for specific gain code as GAIN%
```

```
      GAIN%=2/GANE%
'
'dimension arrays for data for six channels
'     DIM SG%(6),VALUE#(6)
'++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++
'+ loop to call subroutines to get A/D conversion +
'+ values and compute decimal number for voltage  +
'++++++++++++++++++++++++++++++++++++++++++++++++++

'the channel number is defined by CHANNEL%
   FOR CHANNEL%=0 TO 5

'skip reading channel if temp probe is not
' connected
IF (TEMP$(channel%+2)="N/C") OR (TEMP$(channel%+2)="n/c") THEN GOTO 9111

'call subroutine to get a/d conversion value
CALL adin(&H2ec,GANE%,CHANNEL%,SG%(CHANNEL%))

'call subroutine to compute decimal number for
'voltage
CALL COMPUTE(GAIN%,SG%(CHANNEL%),VALUE#(CHANNEL%))

'call subroutine to compute value of temperature
CALL COMPUTETEMPERATURE(TMPUNIT$,VALUE#(CHANNEL%),TEMPVALUE#(CHANNEL%))

'assign text color to txckr%
txckr%=4

'assign background color to bkckr%
bkckr%=5

'set color for display of temperatures
color txckr%,bkckr%
'color 4,5

'turn timer off
timer off

'location of display on screen (row,column)
LOCATE bbegrow%+channel%+3,bbegcol%+5

'print value of temperature on screen
print using "###.#";TEMPVALUE#(channel%)

'turn timer on
timer on

'get next channel number
9111 NEXT CHANNEL%
'++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++
'return to main program
RETURN
'==================================================




'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: graph3.BAS                 ==
'==               backup prog : graph3.BAK                 ==
'==                                                        ==
'============================================================
' PROGRAM WRITTEN:  5-09-89                                 '
'                                                           '
' PROGRAM MODIFIED: 5-17-89                                 '
'============================================================
'==  PROGRAM NAME:  graph3.BAS                             ==
'==                                                        ==
'============================================================
'define subroutine graph3
graph3:

'set flag place%=6 for return from escape key
place%=3
```

```
'segment statement which tells computer to start
'a second code segment for machine code
$SEGMENT

'set screen to text mode with color enabled
screen 0,1

'set flag so current time will not be displayed on screen
fflag%=4

'turn timer off
TIMER OFF

'stop functions of all unused function keys
key(1) stop
key(2) stop
key(3) stop
key(4) stop
key(5) stop
key(6) stop
key(8) stop
key(9) stop
key(10) stop

'turn all unused function keys off
key(1) off
key(2) off
key(3) off
key(4) off
key(5) off
key(6) off
key(8) off
key(9) off
key(10) off

'set palette color 7 to yellow (65)
palette 7,65

'set palette color 6 to white (64)
palette 6,64

'set palette color 3 to cyan (52)
palette 3,52

'set palette color 1 to blue (57)
palette 1,57

'set palette color 2 to magenta (58)
palette 2,61

'set palette color 15 to WHITE (63)
palette 15,63

'set palette color 0 to white (63)
palette 0,63

on key(6) gosub gorunmon
Key (6) on

on key(1) gosub GRAPH3
key (1) on

'clear the screen
2700 color 1,1
     cls

'CALL SUBROUTINE TO DRAW GRAPH BOXES
GOSUB GRAFBOXX

'.....................................................
'begin menu selection process
'_____________________________________________________
'  dimension choice array
     DIM choice$(7)

'  assign line number 1 to array element choice$(1)
   CHOICE$(1)=" 1."

'  assign line number 2 to array element choice$(2)
   CHOICE$(3)=" 2."

'  define LLINE% as an integer used to locate text
'   on proper row
     LLINE%=4
```

```
'  change color to white text and BLUE (1)
'   background;  locate the text at row 8 and
'   column 18;  print CHOICE$(1)
   COLOR 15,1
   LOCATE lline%+7,18
        PRINT CHOICE$(1)

'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                BEGIN SELECTION PROCESS                              XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
'XX                                                                     XX
'XX       This selection routine was originally written                 XX
'XX     by K. Allen Caswell, Jr., Ph.D. on 12-14-87.                    XX
'XX                                                                     XX
'XX       Modifications and all documentation                           XX
'XX     was done by Cynthia K. Rice on 3-1-89.                          XX
'XX                                                                     XX
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  mark this part of program with a label SelNUM
'   so it can be called again
   SelNUM:

' assigns to the variable KKEY$ the character that
'  corresponds to the first keystroke waiting in the
'  keyboard buffer
        KKEY$=INKEY$

'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
'  Beginning of SELECT CASE structure = case variable
'    is KKEY$
   SELECT CASE KKEY$

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'  if KKEY$="" (no keystroke on keyboard buffer), then
'   goto SelNUM (continue selection process)
        CASE ""
          GOTO SelNUM
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'  if KKEY$ = CHR$(0)+CHR$(72),CHR$(0)+CHR$(80),
'   then execute the following lines

'   CHR$(n) is the character in the ASCII table that
'    is associated with n (the ASCII value)

'     CHR$(0)+CHR$(72) represents the up arrow
'     CHR$(0)+CHR$(80) represents the down arrow

'   The CHR$(0) represents that this is the extended
'    code for a non-ASCII key
        CASE CHR$(0)+CHR$(72),CHR$(0)+CHR$(80)

       ' change the color to blue text and cyan
       '  background;  print text on row LLINE%+7 and
       '  column 16
          COLOR 1,5
          LOCATE LLINE%+7,16

       ' print choice$(LLINE%)
          PRINT choice$(LLINE%)

       ' IF KKEY$ = CHR$(0)+CHR$(72) representing the down arrow,
       '  then decrement LLINE% by 2
       ' ELSE increment LLINE% by 2 (meaning the up arrow
       '  was selected)
          IF KKEY$=CHR$(0)+CHR$(72) THEN DECR LLINE%,2 ELSE INCR LLINE%,2

       ' need to keep LLINE% in allowable range for data
       '  so the selection box will scroll from last
       '  allowable selection to first allowable selection
       '  and vice versa
       ' If select up arrow when on top line will go to
       '  bottom line
        IF LLINE%<1 THEN LLINE%=3

       ' If select down arrow when on bottom line will go to
       '  top line
        IF LLINE%>3 THEN LLINE%=2

       ' change the color to white text and blue
       '  background;  print next text at row (LLINE%+7)
```

```
        '   and column 18
          COLOR 15,1
          LOCATE LLINE%+7,18

        ' print choice$(LLINE%)
          PRINT choice$(LLINE%)

       ' goto label Selnum to continue selection process
     GOTO Selnum:
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'on carriage return, goto doit subroutine to correct input
 ' CHR$(13) represents carriage return in ASCII table
        CASE CHR$(13)

      '  goto subroutine Correctnum to correct info
          GOTO Correctnum:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
' any other input sends program back to label Selnum
'  to continue selection process
        CASE ELSE
          GOTO Selnum
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    '  Define end of SELECT CASE structure
        END SELECT
'##############################################################################
'##############################################################################
'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

'  define end of subroutine Selnum
      RETURN

'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                        ppp
'ppp              SUBROUTINE NAME: Correctnum                               ppp
'ppp                                                                        ppp
'ppp          SUBROUTINE TO SELECT LINE NUMBER                              ppp
'ppp                                                                        ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  define CORRECTNUM as a subroutine
CORRECTNUM:

'  Change color to blue (1) text and cyan (3) background
   COLOR 1,3

'  Locate the text in row (lline%+8) and column 8
      LOCATE  lline%+5,18

'##############################################################################
'##############################################################################
'  Beginning of SELECT CASE structure = case variable
'   is LLINE%
   SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then CHECK ANSWER
CASE 1
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=2

'define beginning and ending rows of menu box
begrow%=17
endrow%=22

'define beginning and ending columns of menu box
begcol%=10
endcol%=72

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print selection
LOCATE 8,11
PRINT "  You selected a graph of two temperatures";
PRINT " versus time (#1). "
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
   '  If LLINE% = 3, then CHECK ANSWER
CASE 3
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=2

'define beginning and ending rows of menu box
begrow%=17
endrow%=22

'define beginning and ending columns of menu box
begcol%=6
endcol%=55

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print selection
LOCATE 18,6
PRINT "You selected a graph of the Oxygen Consumption";
PRINT " parameters vs time (#2)."
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% <> 2 AND >3, then continue program
      CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
      END SELECT
'##############################################################################
'##############################################################################
'  Change color to blue (1) text and WHITE (15) background
      COLOR 4,15

'  Call PROCEDURE CorrectAnswer to ask if input is correct
   COLOR 2,2
   LOCATE 20,16
   PRINT SPACE$(55)
   LOCATE 20,28
   CALL CorrectAnswer (ans$)

'  Check answer to PROCEDURE CorrectAnswer :

'  If the answer is No, then prompt user to try again
2720     IF (ans$="N") OR (ans$="n") THEN 2700
   COLOR 2,2
   LOCATE 20,51
   PRINT SPACE$(20)

'  If the answer is Yes, then return to main program
   IF (ans$="y") OR (ans$="Y") THEN 2725

'  If answer was invalid, then call PROCEDURE InvalidAnswer
'     to prompt user to input a proper answer of yes or no.
   COLOR 2,2
   LOCATE 21,19
   PRINT SPACE$(53)
   LOCATE 21,19
   CALL InvalidAnswer (ans$)
      GOTO 2740

'##############################################################################
'  Beginning of SELECT CASE structure = case variable
'    is LLINE%
2725     SELECT CASE LLINE%

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 1, then gosub graphtp
   CASE 1
   GOSUB GRAPHTP
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% = 3, then gosub graphb
   CASE 3
   GOSUB GRAPHHB
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  If LLINE% <> 2 AND >3, then continue program
      CASE ELSE
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   '  Define end of SELECT CASE structure
        END SELECT
'##################################################################

'define end of subroutine graph3
return
'==================================================================

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                               ppp
'ppp              SUBROUTINE NAME: GORUNMON                        ppp
'ppp                                                               ppp
'ppp     SUBROUTINE TO RETURN TO RUNMON SUBROUTINE                 ppp
'ppp     TO CONTINUE MONITORING PROCESS.                           ppp
'ppp                                                               ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'  define GORUNMON as a subroutine
GORUNMON:

'set screen back to text mode
screen 0,1

'stop functions of all unused function keys
Key(1) stop
Key(2) stop
Key(3) stop
Key(4) stop
Key(5) stop
Key(6) stop
Key(8) stop
Key(9) stop
Key(10) stop

'turn all unused function keys off
Key(1) off
Key(2) off
Key(3) off
Key(4) off
Key(5) off
Key(6) off
Key(8) off
Key(9) off
Key(10) off

'set palette color 7 to GREY (7)
palette 7,7

'set palette color 6 to BROWN (20)
palette 6,20

'set palette color 3 to cyan (3)
palette 3,3

'set palette color 5 to blue (5)
palette 5,5

'set palette color 2 to GREEN (2)
palette 2,2

'set palette color 15 to WHITE (63)
palette 15,63

'set palette color 0 to WHITE (63)
palette 0,63

'set color to white background and white text
color 0,0

'clear the screen
cls

'return to subroutine runmon to continue monitoring
goto runmon

'define end of subroutine gorunmon
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'files saved under another name to save space
'in turbobasic editor
$include "c:\monitor1\GRAFBOX.bas"
```

```
$include "c:\monitor1\graftp.bas"
$include "c:\monitor1\graffhb.bas"



'============================================================
'==              ***** IMPORTANT INFO *****                ==
'==                                                        ==
'==               program name: grafbox.BAS                ==
'==               backup prog : grafbox.BAK                ==
'==                                                        ==
'============================================================
' PROGRAM WRITTEN:  5-09-89                                 '
'                                                           '
' PROGRAM MODIFIED: 5-09-89                                 '
'============================================================
'==  PROGRAM NAME:  grafbox.BAS                            ==
'==                                                        ==
'==      This program contains the subroutine GRAFBOXX     ==
'==  which draws the boxes for the graph selection screen  ==
'==                                                        ==
'============================================================
'DEFINE SUBROUTINE GRAFBOXX
GRAFBOXX:

'============================================================
'draw large box around screen
'------------------------------------------------------------
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=2
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=3
endrow%=53

'define beginning and ending columns of menu box
begcol%=3
endcol%=79

'call procedure MENUBOX to draw menu box
call MENUBOX(begrw%,endrw%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------

'============================================================
'draw box 1 from the bottom of the screen
'------------------------------------------------------------
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=4
endrow%=22

'define beginning and ending columns of menu box
begcol%=54
endcol%=68

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------

'============================================================
'draw box 2 from the bottom of the screen
'------------------------------------------------------------
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=8
endrow%=21

'define beginning and ending columns of menu box
begcol%=30
endcol%=52

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------
```

```
'============================================================
'draw box 3 from the bottom of the screen
'------------------------------------------------------------
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=5
endrow%=40

'define beginning and ending columns of menu box
begcol%=56
endcol%=66

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------

'============================================================
'draw box 4 from the bottom of the screen
'------------------------------------------------------------
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=6
endrow%=19

'define beginning and ending columns of menu box
begcol%=22
endcol%=60

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------

'============================================================
'draw box 5 from the bottom of the screen
'------------------------------------------------------------
'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=1
BKCOL%=3

'define beginning and ending rows of menu box
begrow%=5
endrow%=18

'define beginning and ending columns of menu b x
begcol%=1?
endcol%=65

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------

'draw box 6 from the bottom of the scree-
'------------------------------------------------------------
'set text color (TXCOL%9) and background color (BKCOL%)
TXCOL%=0
BKCOL%=5

'define beginning and ending rows of me.w box
begrow%=3
endrow%=78

'define bgginning and ending co uons of ldnq box
begcol%=13
endcol%=69

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)
'------------------------------------------------------------
print"Please use the arrow keys to move the selection box to the"
print" proper line number.  Then press the enter key."
'------------------------------------------------------------
'set text color and background color
color 2,3
'------------------------------------------------------------
locate br%+9,bc%+3
print"    Two Temperatures versus Time (same graph)
color 1,3
locate br%+5,bc%+3
```

```
print" 1."
color 2,3
'
locate br%+4,bc%+3
print"    Hb Concentration              versus Time
color 1,3
locate br%+5,bc%+3
print" 2."
color 2,3
locate br%+7,bc%+3
print"    Arterial Oxygen Saturation   versus Time
locate br%+8,bc%+3
print"    Venous Oxygen Saturation     versus Time
locate br%+7,bc%+3
print"    Oxygen Consumption           versus Time
locate br%+14,bc%+4
print"   (All parameters on same graph.)
'
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'define end of subroutine GRAFBOXX
RETURN
'==============================================================



'======================================================
'==            ***** IMPORTANT INFO *****           ==
'==                                                 ==
'==            program name: GRAFTP.BAS             ==
'==            backup prog : GRAFTP.BAK             ==
'==                                                 ==
'======================================================
'== Program  written:  5-09-89                      ==
'==                                                 ==
'== Program modified:  5-23-89                      ==
'======================================================
'==  subroutine name:  graphtp.bas                  ==
'==                                                 ==
'==     THIS SUBROUTINE CALLS THE SUBROUTINES TO    ==
'==     GRAPH TWO TEMPERATURES VERSUS TIME.         ==
'==                                                 ==
'======================================================
'define subroutine graphtp
graphtp:

'set flag place%=5 for return from escape key
place%=5

'call subroutine to select two temperatures
'to be graphed
gosub picktemps

'label to return from escape key
escapetemp:

'call subroutine to set up for graphics display
gosub setgraf

'count number of times thru subroutine
counttp%=counttp%+9

'if more than one time thru subroutine then
'goto label to clear screen
if counttp%>1 then goto bblue

'call subroutine to draw axes of graph
gosub drawaxes

'call subroutine to draw the timescale of graph
gosub timescale

'call subroutine to draw the temperature scale fo graph
gosub tempscale

'call subroutine to graph temperatures A & B
gosub graphtemp

'define end of subroutine graphtp
RETURN
'======================================================
```

```
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                    ppp
'ppp                SUBROUTINE NAME:  PICKTEMPS                         ppp
'ppp                                                                    ppp
'ppp        SUBROUTINE TO SELECT TWO TEMPS TO GRAPH                     ppp
'ppp                                                                    ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine PICKTEMPS:
PICKTEMPS:

COLOR 1,1
'   CLEAR THE SCREEN
    CLS

'set text color (txcol%) and gackground color (bkcol%)
'for box
txcol%=0
bkcol%=1

'define beginning and ending rows of box
begrow%=16
endrow%=45

'define beginning and ending columns of box
begcol%=12
endcol%=55

'call procedure menubox to draw box
call MENUBOX(begrow%,endrow% Begcol%,endcol%,txcol%,bkcol%)

'define beginning row for display
drow%=2

'define beginning column for display
dcol%=21

     '  Print temperature #1 name of screen
    LOCATE drow%+2,dcol%+1
    PRINT "NAME OF TEMPERATURE #1 IS  "
       LOCATE drow%+2,dcol%/38
    PRINT temp$(1)

     '  Print temperature #2 name  n screen
    LOCATE drow%+3,dcol%+1
    PRINT "NAOE OF TEMPERAT]RE #2 IS  "
       LOCATE drow%+3,dcol%+38
    PRINT temp$(2)

     '  Print temperature #3 name on screen
    LOCATE drow%+4,dcol%+1
    PRINT "NAME OF TEMPERATURE #3 IS  "
       LOCATE drow%+4,dcol%+38
    PRINT temp$(3)


     '  Print temperature #4 name on screen
    LOCATE drow%+5,dcol%+1
    PRINT "NAME OF TEMPERATURE #4 IS  "
       LOCATE drow%+5,dcol%+38
    PRINT temp$(4)

     '  Print temperature #5 name on screen
    LOCATE drow%+6,dcol%+1
    PRINT "NAME OF TEMPERATURE #5 IS  "
       LOCATE drow%+6,dcol%+38
    PRINT temp$(5)

     '  Print temperature #6 name on screen
    LOCATE drow%+7,dcol%+1
    PRINT "NAME OF TEMPERATURE #6 IS  "
       LOCATE drow%+7,dcol%+38
    PRINT temp$(6)

'print instructions to user
pickit:
locate 13,18
color 1,0
print"Please enter the numbers of the two temperatures to be graphed
locate 54,10
print "separated by a comma. (Example: 1,2 <return>)"

'locate and print question mark (?) for input
color 7,0
```

```
locate 16,40
input atemp%,btemp%

'label for correction
corrab:

'beginning of case select structure
'case variable = atemp%
SELECT CASE atemp%

case 1,2,3,4,5,6

'if a% is not between 1 and 6 then tell user that
'input was invalid
case else
color 31,4
locate 18,20
sound 37,10
print " The values you have entered are not valid! "
locate 1,20
print "   Please enter two integers from 1 to 6.   "
locate 16,40
color 2,2
print space$(15)
color 2,0

'locate and print question mark (?) for input
color 17,0
locate 21,40
input atemp%,btemp%

locate 21,40
color 0,0
print space$(15)
color 1,0

goto corrab

'define end of select structure
end select

'beginning of case select structure
'case variable = btemp%
SELECT CASE btemp%

case 1,2,3,4,5,6

'if b% is not between 1 and 6 then tell user that
'input was invalid
case else
color 1,4
locate 18,20
sound 37,10
print " The values you have entered are not valid! "
locate 19,20
print "   Please enter two integers from 1 to 6.   "
locate 16,40
color 0,0
print space$(15)
color 1,0

'locate and print question mark (?) for input
color 17,0
locate 21,40
input atemp%,btemp%

locate 11,56
color 0,0
print space$(5)
color 1,0

'go back to check new input
goto corrab

'define end of select structure
end select

'  End of SUBROUTINE picktemps, return to main program
      RETURN

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp             SUBROUTINE NAME:  GRAPHTEMP                          ppp
'ppp                                                                   ppp
```

```
'ppp  SUBROUTINE TO DRAW GRAPHS OF TWO TEMPS VS. TIME    ppp
'ppp                                                     ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine graphtemp:
graphtemp:

'+++++++++++++++++++++++++++++++++++++++++++++++++++++++
'                VARIABLE DEFINITIONS                  +
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'define gain code for DT2801-A as GANE%
     GANE%=1
'
'define gain for specific gain code as GAIN%
     GAIN%=2^GANE%
'
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++
'=======================================================
'turn timer on
timer on

'initialize counter
kount%=0

'assign start time to variable bb
bb=timer
'_______________________________________________________
'define label for loop
LOOPIT:
'_______________________________________________________
'_______________________________________________________
'define counter
KOUNT%=KOUNT%+1
'_______________________________________________________
'_______________________________________________________
'assign old value of time to ttold
ttold=tt*2
'_______________________________________________________
'_______________________________________________________
'assign old value of temp A to atempold$
atempold$=tempvalua$
'_______________________________________________________
'_______________________________________________________
'assign old value of temp B to btempold$
btempold$=tempvalub$
'_______________________________________________________
'if tempe_____________________
'assign old value of time to ttold
ttold=tt*2
'_______________________________________________________
'_______________________________________________________
'assign old value of temp A to atempold$
atempold$=tempvalua$
'_______________________________________________________
'_______________________________________________________
'assign old value of temp B to btempold$
btempold$=tempvalub$
'_______________________________________________________
'if tempe_____________________
'assign old value of time to ttold
ttold=tt/2
'_______________________________________________________
'_______________________________________________________
'assign old value of temp A to atempold$
atempold$=tempvalua$
'_______________________________________________________
'_______________________________________________________
'assign old value of temp B to btempold$
btempold$=tempvalub$
'_______________________________________________________
'if tempe,1
   'return to start of loop
   GOTO LOOPIT
'define end of if block
END IF
'_______________________________________________________
'assign elapsed time to variable tt
TT=TIMER-BB
'_______________________________________________________
'if temperature probe is not connected, then print n/c
'on screen and skip calling subroutines to graph
if temp$(atemp%)="n/c" or temp$(atemp%)="N/C" then
   'LOCATE AND PRINT n/c
     color 12,14
```

```
      locate 26,6
      print  " N/C "
        goto GRAFFBB
'define end of if block
end if

'
'_____________________________________________________________
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'                    Get temperature A
'_____________________________________________________________
'call subroutine to get a/d conversion value for temp A
CALL adin(&H2ec,GANE%,ATEMP%-1,SGA%)

'call subroutine to compute decimal number for voltage
'representing temp A
CALL COMPUTE(GAIN%,SGA%,VALUEA#)

'call subroutine to compute value of temperature A
CALL COMPUTETEMPERATURE(TMPUNIT$,VALUEA#,TEMPVALUA#)
'
'_____________________________________________________________
'_____________________________________________________________
'draw graph of temperature A
LINE  (told,atempold#) - (tt*2,TEMPVALUA#),12
LINE  (told,atempold#+.05) - (tt*2,TEMPVALUA#+.05),12
LINE  (told,atempold#+.1) - (tt*2,TEMPVALUA#+.1),12
LINE  (told,atempold#-.05) - (tt*2,TEMPVALUA#-.05),12
LINE  (told,atempold#-.1) - (tt*2,TEMPVALUA#-.1),12
LINE  (told,atempold#+.15) - (tt*2,TEMPVALUA#+.15),12
LINE  (told,atempold#+.2) - (tt*2,TEMPVALUA#+.2),12
LINE  (told,atempold#-.15) - (tt*2,TEMPVALUA#-.15),12
LINE  (told,atempold#-.2) - (tt*2,TEMPVALUA#-.2),12
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'LOCATE AND PRINT TEMP 1
color 2,14
locate 6,6
print using "###.#";tempvalua#
'
'_____________________________________________________________
'label for not connected probes
GRAFFBB:

'if temperature probe is not connected, then print n/c
'on screen and skip calling subroutines to graph
if temp$(btemp%)="n/c" or temp$(btemp%)="N/C" then
   'LOCATE AND PRINT TEMP 2
      color 3,14
      locate 20,5
      print  " N/C "
        goto noGRAFFBB
'define end of if block
end if

'
'_____________________________________________________________
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'                    Get temperature B
'_____________________________________________________________
'call subroutine to get a/d conversion value for temp B
CALL adin(&H2ec,GANE%,bTEMP%-1,SGb%)

'call subroutine to compute decimal number for voltage
'representing temp B
CALL COMPUTE(GAIN%,SG%,VALUEb#)


'call subroutine to compute value of temperature B
CALL COMPUTETEMPERATURE(TMPUNIT$,VALUEb#,TEMPVALUb#)
'
'_____________________________________________________________
'_____________________________________________________________
'draw graph of temperature B
LINE  (ttold,btempold#) - (tt*2,TEMPVALUb#),13
LINE  (ttold,Btempold#+.05) - (tt*2,TEMPVALUB#+.05),13
LINE  (ttold,Btempold#+.1) - (tt*2,TEMPVALUB#+.1),13
LINE  (ttold,Btempold#-.05) - (tt*2,TEMPVALUB#-.05),13
LINE  (ttold,Btempold#-.1) - (tt*2,TEMPVALUB#-.1),13
LINE  (ttold,Btempold#+.15) - (tt*2,TEMPVALUB#+.15),13
LINE  (ttold,Btempold#+.2) - (tt*2,TEMPVALUB#+.2),13
LINE  (ttold,Btempold#-.15) - (tt*2,TEMPVALUB#-.15),13
LINE  (ttold,Btempold#-.2) - (tt*2,TEMPVALUB#-.2),13
'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'_____________________________________________________________
'LOCATE AND PRINT TEMP 2
color 3,14
locate 20,8
print using "###.#";tempvalub#
```

```
'
'label for not connected probes
noGRAFFBB:
'
'if elapsed time is greater than 59 seconds, then need
'to clear screen, reset timer,initialize elapsed time
'variable tt, reset counter to 0, draw grid again, and
'set initial point for graph
if tt/2>59 then

'label for clearing screen
bblue:

'clear the screen
cls

'reset timer
bb=timer

'initialize elapsed time variable tt
tt=0

'initialize counter
kount%=0

'  this statement sets the screen to a right-hand coordinate
'    system where (-1,-1) is the point in the lower left-
'    hand corner and (61,101) is the point in the upper
'    right-hand corner of the viewport.
WINDOW (-1,-1)-(61,101)

'draw grid again
gosub drawaxes

'call subroutine to draw the timescale of graph
gosub timescale

'call subroutine to draw the temperature scale fo graph
gosub tempscale

'set initial point for graph
PSET(.0,TEMPVALUA$),1

'define end of if block
end if
'
'
'continue loop
 GOTO LOOPIT

'define end of subroutine graphtemp
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                   ppp
'ppp              SUBROUTINE NAME:  SETGRAPH                           ppp
'ppp                                                                   ppp
'ppp         SUBROUTINE TO SET UP FOR GRAPHICS                         ppp
'ppp                                                                   ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine setgraph:
setgraf:

' define screen resolution as 640 horizontal by 350
'  vertical 4 to 16-color EGA mode
        Resolution%=9
   RATIO%=640/350

'  defines the screen mode
    SCREEN Resolution%

'define palette color 3 as cyan (59)
PALETTE 3,34

'set text color to blue (1) and background color to yellow
COLOR 1,14

'clear the screen
cls

'  defines the background color within the view area
Vgnd%=0
```

```
'  defines the border color of the view area
VBord%=0
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
'  The next three statements define the graphics window on
'   the screen

'  this statement sets the screen to a left-hand coordinate
'   system where (0,0) is the point in the upper right-hand
'   corner and (640,350) is the point in the lower left-hand
'   corner of the screen
WINDOW SCREEN (0,0)-(640,350)

'draw outer border around screen
line (0,0)-(640,350),1
line (3,3)-(637,347),1
line (6,6)-(634,344),1

'  this statement establishes a rectangular portion of the
'   screen (called a viewport) where all graphics commands
'   following this statement will be displayed
VIEW (105*RATIO%-70,4*RATIO%+9)-(565,234),Vgnd%,VBord%

'  this statement sets the screen to a right-hand coordinate
'    system where (-1,-1) is the point in the lower left-
'    hand corner and (61,101) is the point in the upper
'    right-hand corner of the viewport.
WINDOW (-1,-1)-(61,101)
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
'define end of subroutine setgraph
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp


'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE NAME:  DRAWAXES                         ppp
'ppp                                                                 ppp
'ppp          SUBROUTINE TO DRAW AXES FOR GRAPH                      ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine setgraph:
drawaxes:

'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw box around grid
   LINE (0,0) - (60,100),1,b
   LINE (-.2,-.2) - (60.2,100.2),1,b
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw tick marks on X axis
      FOR Im%=5 to 55 STEP 5

          'draw tick marks on x-axis
        LINE (im%,.5) - (im%,99.7),14

   NEXT im%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw tick marks on Y axis
      FOR Im=7 to 63 STEP 7.143

          'draw tick marks on y-axis
        LINE (.2,im) - (59.9,im),14

   NEXT im
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'DEFINE END OF SUBROUTINE DRAWAXES
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp


'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE NAME:  TIMESCALE                        ppp
'ppp                                                                 ppp
'ppp       SUBROUTINE TO DRAW TIME SCALE FOR GRAPH                   ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine TIMESCALE:
timescale:
```

```
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'PRINT TIME SCALE

'set row for time scale
rr%=10

'locate and print 0
LOCATE rr%,17
PRINT 0

'locate and print 5
LOCATE rr%,25
PRINT 5

'locate and print 10
LOCATE rr%,34
PRINT 10

'locate and print 15
LOCATE rr%,43
PRINT 15

'locate and print 20
LOCATE rr%,52
PRINT 20

'locate and print 25
LOCATE rr%,61
PRINT 25

'locate and print 30
LOCATE rr%,69
PRINT 30

'locate and print title for x axis
LOCATE RR%+1,37
PRINT"TIME (seconds)"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'DEFINE END OF SUBROUTINE TIMESCALE
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp


'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE NAME:  TEMPSCALE                        ppp
'ppp                                                                 ppp
'ppp       SUBROUTINE TO DRAW TIME SCALE FOR GRAPH                   ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine TEMPSCALE:
tempscale:

'ccccccccccccccccccccccccccccccccccccccccccccccccc
'PRINT TEMPERATURE SCALE

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=2
BKCOL%=14

'define beginning and ending rows of menu box
begrow%=14
endrow%=17

'define beginning and ending columns of menu box
begcol%=4
endcol%=13

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print title for temperature A
COLOR 12,14
LOCATE 15,5
PRINT "TEMP $"+STR$(ATEMP%)

'set text color (TXCOL%) and background color (BKCOL%)
TXCOL%=13 :BKCOL%=14

'define beginning and ending rows of menu box
begrow%=8
endrow%=23
```

```
'define beginning and ending columns of menu box
begcol%=4
endcol%=13

'call procedure MENUBOX to draw menu box
call MENUBOX(begrow%,endrow%,begcol%,endcol%,TXCOL%,BKCOL%)

'locate and print title for temperature B
COLOR 13,14
LOCATE 19,5
PRINT"TEMP #"+STR$(btemp%)

'set color to red text and yellow background
color 4,14

'locate and print title for y axis
LOCATE 11,3
PRINT"TEMPERATURE"

'set column for placement of temperature scale
cc%=54

'if units of temperature are CELSIUS then print
'proper temperature scale
IF TMPUNIT$="CELSIUS" THEN

'LOCATE AND PRINT UNITS ON SCREEN
LOCATE 12,4
PRINT"(Celsius)"

'locate and print 15
LOCATE 21,cc%
PRINT 15

'locate and print 20
LOCATE 19,cc%
PRINT 20

'locate and print 25
LOCATE 16,cc%
PRINT 25

'locate and print 30
LOCATE 13,cc%
PRINT 30

'locate and print 35
LOCATE 10,cc%
PRINT 35

'locate and print 40
LOCATE 7,cc%
PRINT 40

'locate and print 45
LOCATE 4,cc%
PRINT 45

'locate and print 50
LOCATE 2,cc%
PRINT 50

'else, units of temperature are FARENHEIT then print
'proper temperature scale
ELSE

'LOCATE AND PRINT UNITS ON SCREEN
LOCATE 12,3
PRINT"(Farenheit)"

'locate and print 50
LOCATE 21,cc%
PRINT 50

'locate and print 60
LOCATE 19,cc%
PRINT 60

'locate and print 70
LOCATE 16,cc%
PRINT 70
```

```
'locate and print 80
LOCATE 13,cc%
PRINT 80

'locate and print 90
LOCATE 10,cc%
PRINT 90

'locate and print 100
LOCATE 7,cc%-1
PRINT 100

'locate and print 110
LOCATE 4,cc%-1
PRINT 110

'locate and print 120
LOCATE 2,cc%-1
PRINT 120

'DEFINE END OF IF BLOCK
END IF

'set color to blue text and yellow background
color 1,14

'set column for function key menu
fc%=5

'locate and print function key menu
LOCATE 3,FC%
PRINT " FK MENU:"
LOCATE 4,FC%
PRINT " F1 GRAPH"
LOCATE 5,FC%
PRINT " F6 MONITOR"
LOCATE 6,FC%
PRINT" ESC EXIT"


'DEFINE END OF SUBROUTINE TEMPSCALE
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp



'=========================================================
'==          ***** IMPORTANT INFO *****                 ==
'==                                                     ==
'==          program name: GRAFHB.BAS                   ==
'==          backup prog : GRAFHB.BAK                   ==
'==                                                     ==
'=========================================================
'== Program  written:  5-09-89                          ==
'==                                                     ==
'== Program modified:  5-17-89                          ==
'=========================================================
'==  subroutine name:  graphhb.bas                      ==
'==                                                     ==
'==     THIS SUBROUTINE CALLS THE SUBROUTINES TO        ==
'==     GRAPH TWO TEMPERATURES VERSUS TIME.             ==
'==                                                     ==
'=========================================================
'define subroutine graphhb
graphhb:

'set flag place%=5 for return from escape key
place%=4

'clear the screen
cls

'call subroutine to set up for graphics display
gosub ssetgraf

'locate and print function key menu
COLOR 14,1
LOCATE 1,43
PRINT "* FK MENU: F1 GRAPH, F6 MONITOR, ESC TO EXIT *"
```

```
'counter used to clear the graph
'counts number of times subroutine is called
counthb%=counthb%+1

'if subroutine is called more than one time, then
'goto label ggreen to clear graph
IF counthb%<0 THEN GOTO ggrren

'call subroutine to draw axes of graph
gosub hdrewaxes

'call subroutine to draw the timescale of graph
gosub ttimescale

'call subroutine to draw the temperature scale fo graph
gosub oxscale

$EVENT ON
'call subroutine to graph temperatures A & B
gosub graphox

'define end of subroutine graphtp
RETURN
'=======================================================
'end

'file saved under another name to save
'space in turbobasic editor
$INCLUDE "C:\MONITOR\OXxSCALE.BAS"

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                               ppp
'ppp              SUBROUTINE NAME:  SETGRAPH                       ppp
'ppp                                                               ppp
'ppp          SUBROUTINE TO SET UP FOR GRAPHICS                    ppp
'ppp                                                               ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine setgraph:
ssetgraf:

' define screen resolution as 640 horizontal by 350
'  vertical 4 to 16-color EGA mode
        Resolution%=9
   RATIO%=640/350

'  defines the screen mode
    SCREEN Resolution%

'define palette color 3 as cyan (59)
PALETTE 3,39

'assign background color to variable bgcol%
 bgcol%=15

'set text color to blue (1) and background color to yellow
COLOR 1,bgcol%

'  defines the background color within the view area
Vgnd%=0

'  defines the border color of the view area
VBord%=0
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
'  The next three statements define the graphics window on
'   the screen

'  this statement sets the screen to a left-hand coordinate
'   system where (0,0) is the point in the upper right-hand
'   corner and (640,350) is the point in the lower left-hand
'   corner of the screen
WINDOW SCREEN (0,0)-(640,350)

'  this statement establishes a rectangular portion of the
'   screen (called a viewport) where all graphics commands
'   following this statement will be displayed
VIEW (105*RATIO%-150,5*RATIO%+9)-(435,289),Vgnd%,VBord%

'  this statement sets the screen to a right-hand coordinate
'    system where (-1,-1) is the point in the lower left-
'    hand corner and (63,101) is the point in the upper
'    right-hand corner of the viewport.
WINDOW (-1,-1)-(85,85)
```

```
'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
'define end of subroutine setgraph
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE NAME:  HDRAWAXES                        ppp
'ppp                                                                 ppp
'ppp         SUBROUTINE TO DRAW AXES FOR GRAPH                       ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine setgraph:
Hdrawaxes:

'set palette color 14 to yellow (62)
palette 14,63

'define color of axis
axcolor%=11

'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw X axis
   LINE (0,0) - (50,0),axcolor%
   LINE (0,-.1) - (50,-.1),axcolor%
   LINE (64.4,0) - (58.3,0),axcolor%
   LINE (64.4,-.1) - (58.3,-.1),axcolor%
   LINE (0,100.1) - (50,100.1),axcolor%
   LINE (0,100) - (50,100),axcolor%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw Hb saturation axis
   LINE (0,0) - (0,100.1),axcolor%
   LINE (.1,0) - (.1,100.1),axcolor%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw Hb content axis
   LINE (64.3,0) - (64.3,100.1),axcolor%
   LINE (64.4,0) - (64.4,100.1),axcolor%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw Oxygen consumption axis
   LINE (50,0) - (50,100.1),axcolor%
   LINE (49.9,0) - (49.9,100.1),axcolor%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw tick marks on X axis
'     FOR Im%=5 to 85 STEP 5
   FOR Im%=416 to 4584 STEP 416
   im=im%/100
           'draw tick marks on x-axis
        LINE (im,.5) - (im,99.9),14

   NEXT im%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'  draw tick marks on Y axis
   FOR Im%=10 to 99 STEP 10

           'draw tick marks on y-axis
        LINE (.8,im%) - (49.8,im%),14
        LINE (64,im%) - (64.9,im%),14
   NEXT im%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'DEFINE END OF SUBROUTINE HDRAWAXES
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE NAME:  TIMESCALE                        ppp
'ppp                                                                 ppp
'ppp       SUBROUTINE TO DRAW TIME SCALE FOR GRAPH                   ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine TIMESCALE:
ttimescale:

'set color to red text and bgcol% background
color 14, bgcol%

'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'PRINT TIME SCALE
```

```
'set row for time scale
rr%=22

'locate and print 0
LOCATE rr%,22
PRINT 0

'locate and print 5
LOCATE rr%,39
PRINT 5

'locate and print 10
LOCATE rr%,26
PRINT 10

'locate and print 15
LOCATE rr%,33
PRINT 15

'locate and print 20
LOCATE rr%,40
PRINT 20

'locate and print 25
LOCATE rr%,48
PRINT 25

'locate and print 30
LOCATE rr%,55
PRINT 30

'locate and print title for x axis
LOCATE RR%+1,68
PRINT"TIME  (seconds)"
'ccccccccccccccccccccccccccccccccccccccccccccc
'DEFINE END OF SUBROUTINE TIMESCALE
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                                 ppp
'ppp              SUBROUTINE NAME:  GRAPHOX                          ppp
'ppp                                                                 ppp
'ppp             SUBROUTINE TO DRAW GRAPH OF                         ppp
'ppp             OXYGEN PARAMETERS VS. TIME                          ppp
'ppp                                                                 ppp
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine graphox
graphox:

'+++++++++++++++++++++++++++++++++++++++++++++++++ +++
'               VARIABLE DEFINITIONS                +
'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'define gain code for DT2801-A as GANE%
 GANE%=1
'
'define gain for specific gain code as GAIN%
   GAIN%=2^GANE%
'
'+++++++++++++++++++++++++++++++++++++++++++++++++++++
'=====================================================
'turn timer on
timer on

'initialize counter
kount%=0

'assign start time to variable bb
bb=timer
'_____________________________________________________
'define label for loop
LOOOPIT:
'
'_____________________________________________________
'_____________________________________________________
'define counter
KOUNT%=KOUNT%+1
'
'_____________________________________________________
'_____________________________________________________
'assign old value of time to ttold
ttold=tt/2
'
'_____________________________________________________
'_____________________________________________________
```

```
'assign old value of hb to hbold#
hbold#=hb#
'-------------------------------------------------------------------------
'assign old value of asat to asatold#
asatold#=asat#
'-------------------------------------------------------------------------
'assign old value of vsat to vsatold#
vsatold#=vsat#
'-------------------------------------------------------------------------
'assign old value of o2c to o2cold#
O2COLD#=O2C#
'-------------------------------------------------------------------------

'$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
'call subroutine to read analog values from DT2801-A
'reads values for hemoglobin, venous oxygen
'saturation, arterial oxygen saturation, and
'the flow rate
GOSUB RREADO2

'call subroutine to compute decimal values for
'hemoglobin, venous oxygen saturation,
'arterial oxygen saturation, and the flow rate
GOSUB CCOMPUTEDECIMAL

'call subroutine to compute oxygen consumption
GOSUB O2CONSGRAPH
'-------------------------------------------------------------------------
IF KOUNT%=1 THEN
   BB=TIMER
  'set initial point for graph
   TTOLD=74/49
PSET(.1,ASAT#),3
  'return to beginning to loop to get another value
   goto looopit
'define end of if block
END IF

'assign elapsed time to variable tt
TT=TIMER-BB
TTTOLD=TTOLD*50/74
TTT=TT*50/74

'SET SCALE FOR ASAT AND VSAT
'   this statement sets the screen to a right-hand coordinate
'     system where (-1,49.5) is the point in the lower left-
'     hand corner and (61,100.5) is the point in the upper
'     right-hand corner of the viewport.
WINDOW (-1,59.5)-(61,100.5)
'-------------------------------------------------------------------------
'draw graph of ASAT
asatcol%=12
LINE  (Tttold,ASATold#) - (Ttt*2,ASAT#),asatcol%
LINE  (Tttold,ASATold#-.1) - (Ttt*2,ASAT#-.1),asatcol%
LINE  (Tttold,ASATold#-.2) - (Ttt*2,ASAT#-.2),asatcol%
LINE  (Tttold,ASATold#+.1) - (Ttt*2,ASAT#+.1),asatcol%
LINE  (Tttold,ASATold#+.2) - (Ttt*2,ASAT#+.2),asatcol%
LINE  (Tttold,ASATold#+.3) - (Ttt*2,ASAT#+.3),asatcol%
LINE  (Tttold,ASATold#-.3) - (Ttt*2,ASAT#-.3),asatcol%
LINE  (Tttold,ASATold#-.4) - (Ttt*2,ASAT#-.4),asatcol%
LINE  (Tttold,ASATold#+.4) - (Ttt*2,ASAT#+.4),asatcol%
LINE  (Tttold,ASATold#-.45) - (Ttt*2,ASAT#-.45),asatcol%
LINE  (Tttold,ASATold#+.45) - (Ttt*2,ASAT#+.45),asatcol%
'
'-------------------------------------------------------------------------
'draw graph of VSAT
vsatcol%=3
LINE  (Tttold,VSATold#) - (Ttt*2,VSAT#),vsatcol%
LINE  (Tttold,VSATold#+.2) - (Ttt*2,VSAT#+.2),vsatcol%
LINE  (Tttold,VSATold#+.1) - (Ttt*2,VSAT#+.1),vsatcol%
LINE  (Tttold,VSATold#+.3) - (Ttt*2,VSAT#+.3),vsatcol%
LINE  (Tttold,VSATold#+.4) - (Ttt*2,VSAT#+.4),vsatcol%
LINE  (Tttold,VSATold#+.45) - (Ttt*2,VSAT#+.45),vsatcol%
LINE  (Tttold,VSATold#-.2) - (Ttt*2,VSAT#-.2),vsatcol%
LINE  (Tttold,VSATold#-.1) - (Ttt*2,VSAT#-.1),vsatcol%
LINE  (Tttold,VSATold#-.3) - (Ttt*2,VSAT#-.3),vsatcol%
LINE  (Tttold,VSATold#-.4) - (Ttt*2,VSAT#-.4),vsatcol%
LINE  (Tttold,VSATold#-.45) - (Ttt*2,VSAT#-.45),vsatcol%
'
'-------------------------------------------------------------------------
```

```
IF KOUNT%=1 THEN
'set initial point for graph
PSET(.1,HB#),3
end if

'SET SCALE FOR HEMOGLOBIN CONTENT
'   this statement sets the screen to a right-hand coordinate
'     system where (-1,0) is the point in the lower left-
'     hand corner and (61,26) is the point in the upper
'     right-hand corner of the viewport.
WINDOW (-1,-.5)-(61,25.5)
'-------------------------------------------------------------
'draw graph of hb
hbcol%=12
LINE  (Tttold,HBold#) - (Ttt*2,HB#),hbcol%
LINE  (Tttold,HBold#+.2) - (Ttt*2,HB#+.2),hbcol%
LINE  (Tttold,HBold#+.1) - (Ttt*2,HB#+.1),hbcol%
LINE  (Tttold,HBold#-.1) - (Ttt*2,HB#-.1),hbcol%
LINE  (Tttold,HBold#-.2) - (Ttt*2,HB#-.2),hbcol%
LINE  (Tttold,HBold#-.25) - (Ttt*2,HB#-.225),hbcol%
LINE  (Tttold,HBold#+.25) - (Ttt*2,HB#+.225),hbcol%
LINE  (Tttold,HBold#-.15) - (Ttt*2,HB#-.15),hbcol%
LINE  (Tttold,HBold#+.15) - (Ttt*2,HB#+.15),hbcol%
LINE  (Tttold,HBold#-.05) - (Ttt*2,HB#-.05),hbcol%
LINE  (Tttold,HBold#+.05) - (Ttt*2,HB#+.05),hbcol%
'-------------------------------------------------------------
IF KOUNT%=1 THEN
'set initial point for graph
PSET(.1,O2C#),3
end if

'SET SCALE FOR OXYGEN CONSUMPTION
'   this statement sets the screen to a right-hand coordinate
'     system where (-1,0) is the point in the lower left-
'     hand corner and (61,251) is the point in the upper
'     right-hand corner of the viewport.
WINDOW (-1,0)-(51,251)
'-------------------------------------------------------------
'draw graph of oxygen consumption
oxcol%=10
LINE  (Tttold,O2Cold#) - (Ttt*2,O2C#),oxcol%
LINE  (Tttold,O2Cold#+1) - (Ttt*2,O2C#+1),oxcol%
LINE  (Tttold,O2Cold#+.5) - (Ttt*2,O2C#+.5),oxcol%
LINE  (Tttold,O2Cold#+1.5) - (Ttt*2,O2C#+1.5),oxcol%
LINE  (Tttold,O2Cold#+2) - (Ttt*2,O2C#+2),oxcol%
LINE  (Tttold,O2Cold#-.5) - (Ttt*2,O2C#-.5),oxcol%
LINE  (Tttold,O2Cold#-1) - (Ttt*2,O2C#-1),oxcol%
LINE  (Tttold,O2Cold#-1.5) - (Ttt*2,O2C#-1.5),oxcol%
LINE  (Tttold,O2Cold#-2) - (Ttt*2,O2C#-2),oxcol%
LINE  (Tttold,O2Cold#+2.25) - (Ttt*2,O2C#+2.25),oxcol%
LINE  (Tttold,O2Cold#-2.25) - (Ttt*2,O2C#-2.25),oxcol%
'-------------------------------------------------------------
'-------------------------------------------------------------

'locate and print ASAT on screen
color 12,bgcol%
locate 23,2
IF ASAT#>99 THEN ASAT#=89
print using "##";ASAT#

'locate and print VSAT on screen
color 3,bgcol%
locate 23,8
IF VSAT#>99 THEN VSAT#=89
print using "##";VSAT#

'locate and print HB on screen
COLOR 15,bgcol%
locate 23,74
print using "##.#";HB#

'locate and print OXYGEN CONSUMPTION on screen
color 10,bgcol%
locate 28,63
IF O2C#>999 THEN O2C#=999
print using "###";O2C#
'-------------------------------------------------------------
'if elapsed time is greater than 30 seconds, then need
'to clear screen, reset timer,initialize elapsed time
'variable tt, reset counter to 0, draw grid again, and
'set initial point for graph
if tt*2>59 then
```

```
'label for clearing graph if subroutine is
'called more than once
ggreen:

'clear the screen
cls

'reset timer
bb=timer

'initialize elapsed time variable tt
tt=10

'initialize counter
kount%=0

'  this statement sets the screen to a right-hand coordinate
'    system where (-1,-1) is the point in the lower left-
'    hand corner and (61,101) is the point in the upper
'    right-hand corner of the viewport.
WINDOW (-1,-1)-(65,101)

'draw grid again
gosub hdrawaxes

'call subroutine to draw the timescale of graph
gosub ttimescale

'call subroutine to draw the scale for graph
gosub oxscale

'set initial point for graph
PSET(.1,ASAT$),5

'define end of if block
end if
'-------------------------------------------------------------
'-------------------------------------------------------------
GOTO LOOOPIT

'define end of subroutine graphtemp
return
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp

'============================================================

'============================================================
'==                                                        ==
'==                SUBROUTINE NAME: READO2                 ==
'==                                                        ==
'==       subroutine to read analog voltage for            ==
'==       hemoglobin, venous oxygen saturation,            ==
'==       arterial oxygen saturation, and flow rate        ==
'==                                                        ==
'============================================================
'define READO2  as a subroutine
RREADO2:

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                          +
'+               DEFINITION OF VARIABLES                    +
'+                                                          +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                          +
'+                                                          +
'+      specify digital port being set for input            +
'+      used in procedures setdig and readdig               +
'+      this port is used to distinguish between            +
'+      HB, Oxygen sat in, and Oxygen sat out               +
 port%=1
'+                                                          +
'+      define gain code for DT2801-A as GANE%              +
'+      used in procedure adin                              +
 GANE%=1
'+                                                          +
'+      define channel from which to read analog input      +
'+      used in procedure adin                              +
 CHANNEL%=6
'+                                                          +
'+      define gain for specific gain code as GAIN%         +
'+      used in procedure compute                           +
```

```
  GAIN%=2*GANE%
'+                                                              +
'+      assign start time to a variable for later use           +
  begintime=timer
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'call procedure to set digital port 1  for input
CALL setdigit(&H2ec,part%)

'turn event trapping off
$event off

'call procedure to set digital port 1 for output
CALL digout(&H2ec)
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+            read analog values from DT2801-A                  +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'get values for oxygen consumption calculation
'read analog channel 7 (multiplexed input)  so
'that all three data sets (HB, VO2SAT, AO2SAT) are input
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call procedure to send digitout% to port 1
'closes switch so that HB data is input
digitout%=169
CALL writdig(&H2ec,digitout%)

'pause 5.25 msec
for xxx%=1 to 2000
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to get a/d conversion value from
'channel 7 (multiplexed input)
CALL adin(&H2ec,GANE%,CHANNEL%,BINARYVALUE%)

   'assign hb data in binary form to array hb
   hb%=binaryvalue%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 5.25 msec
for xxx%=1 to 2000
next xxx%

'call procedure to send digitout% to port 1
'set all switches open
digitout%=0
CALL writdig(&H2ec,digitout%)

'pause 10.2 sec
for xxx%=1 to 40
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call procedure to send digitout% to port 1
'closes switch so that VO2 data is input
digitout%=95
CALL writdig(&H2ec,digitout%)

'pause 5.25 msec
for xxx%=1 to 2000
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to get a/d conversion value from
'channel 7 (multiplexed input)
CALL adin(&H2ac,GANE%,CHANNEL%,BINARYVALUE%)

   'assign VO2SAT data in binary form to
      'array vsat
   vsat%=binaryvalue%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 2 msec
for xxx%=1 to 2000
next xxx%

'call procedure to send digitout% to port 1
'set all switches open
digitout%=5
CALL writdig(&H2ec,digitout%)
```

```
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 10.2 msec
for xxx%=1 to 4000
next xxx%

'call procedure to send digitout% to port 1
'closes switch so that AO2 data is input
digitout%=63
CALL writdig(&H2ec,digitout%)

'pause 5.25 msec
for xxx%=1 to 2000
next xxx%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to get a/d conversion value from
'channel 7 (multiplexed input)
CALL adin(&H2ec,GANE%,CHANNEL%,BINARYVALUE%)

   'assign AO2SAT data in binary form to
        'array asat
   asat%=binaryvalue%
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'pause 5.25 msec
for xxx%=1 to 2000
next xxx%

'call procedure to send digitout% to port 1
'set all switches open
digitout%=0
CALL writdig(&H2ec,digitout%)
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   'call subroutine to get a/d conversion value from
   'channel 6 (flow rate from flow meter)
      CALL adin(&H2ec,GANE%,6,FLOWROTEBINARY%)
'==========================================================
'turn event trapping on
$event on
'==========================================================
'define end of subroutine READO2
RETURN
'==========================================================

'==========================================================
'==                                                      ==
'==           SUBROUTINE NAME: COMPUTEDECIMAL            ==
'==                                                      ==
'==       subroutine to compute decimal values for       ==
'==       hemoglobin, venous oxygen saturation,          ==
'==       arterial oxygen saturation, and the flow rate  ==
'==                                                      ==
'==========================================================
'define CCOMPUTEDDECIMAL  as a subroutine
CCOMPUTEDECIMAL:

'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'VARIABLE USED TO COMPUTE TIME TO RUN SUB
time2=timer
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+ begin loop to compute decimal value of analog voltages +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'read analog values from array locations and send to
'compute procedure to calculate digital values
'
'computing values for hemoglobin, venous oxygen saturation
'arterial oxygen saturation, and the flow rate
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'voltage representing hemoglobin (hb)
CALL COMPUTE(GAIN%,hb%,hb#)

'compute hemoglobin content in units of gram/decaliter
'using conversion factor from sensor
hb#=hbcon%*hb#
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
```

```
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'voltage representing VO2SAT
CALL COMPUTE(GAIN%,vsat%,vsat$)

'compute percent venous oxygen saturation
'using conversion factor from sensor
vsat$=vsatcon%*vsot$
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'voltage representing AO2SAT
CALL COMPUTE(GAIN%,asat%,asat$)

'compute percent arterial oxygen saturation
'using conversion factor from sensor
asat$=asatcon%*asat$
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'call subroutine to compute decimal number for
'flow rate
CALL COMPUTE(GAIN%,FLOWRATEBINARY%,FLOWRATE$)

'compute flowrate in units of liter per minute
'using conversion factor from sensor
FLOWRATE$=FLOWCAN*FLOWRATE$
'============================================================
'define end of subroutine
RETURN
'============================================================

'============================================================
'==                                                        ==
'==              SUBROUTINE NAME: O2CONSGRAPH              ==
'==                                                        ==
'==       subroutine to compute oxygen conwumption         ==
'==                                                        ==
'============================================================
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+  oxygen consumption is computed using the equation:     +
'                                                          +
'   Oxygen Consumption = (Flowrate)*(Hb content)*          +
'                        (1.39 cc Oxygen/gm Hb)*           +
'                              (change in Oxygen Saturation)+
'                                                          +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                          +
'+                DEFINITION OF VARIABLES                   +
'+                                                          +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+                                                          +
'+       O2C(x)        = oxygen consumption array           +
'+                         (cc oxygen per min)              +
'+       hb$(x)        = hemglobin content (gm/dl)          +
'+       vsat$(x)      = venous oxygen saturation (%)       +
'+       asat$(x)      = arterial oxygen saturation (%)     +
'+       deltasat$(x)  = asat$(x)-vsat$(x)                  +
'+       flowrate$(x)  = flow rate (liters/min)             +
'+       oconst        = 1.39 cc oxygen per gm Hb           +
'+       convertHB     = 0.1 dl/l                           +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'define constants
oconst=1.39
convertHB=10

'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
'+        begin loop to compute oxygen consumption         +
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
deltasat$=(asat$-vsat$)*100

'compute oxygen consumption
O2C$=flowrate$*hb$*canvertHB*oconst*deltasat$

'
'\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
'============================================================
'define end of subroutine
```

```
RETURN
'============================================================



'=========================================================
'==          ***** IMPORTANT INFO *****                 ==
'==                                                     ==
'==          program name: OXXSCALE.BAS                 ==
'==          backup prog : OXXSCALE.BAK                 ==
'==                                                     ==
'=========================================================
'== Program  written:  5-09-89                          ==
'==                                                     ==
'== Program modified:  5-17-89                          ==
'=========================================================
'==  subroutine name:  oxxscale.bas                     ==
'==                                                     ==
'==     THIS SUBROUTINE DRAWS SCALES FOR GRAPH          ==
'==     OF OXYGEN PARAMETERS.                           ==
'==                                                     ==
'=========================================================
'ppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'ppp                                                               ppp
'ppp              SUBROUTINE NAME:  OXSCALE                        ppp
'ppp                                                               ppp
'ppp        SUBROUTINE TO DRAW TIME SCALE FOR GRAPH                ppp
'ppp                                                               ppp
'pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
'define subroutine OXSCALE:
oxscale:

'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'PRINT Hb saturation scale
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'set color to red text and yellow background
color 3,bgcol%
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'assign beginning row for title to variable
'tbegrow%
tbegrow%=22
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print title for y axis
'locate and print "Percent"
LOCATE tbegrow%,3
PRINT"P "
LOCATE tbegrow%+1,3
PRINT"e "
LOCATE tbegrow%+2,3
PRINT"r "
LOCATE tbegrow%+3,3
PRINT"c "
LOCATE tbegrow%+4,3
PRINT"e "
LOCATE tbegrow%+5,3
PRINT"n "
LOCATE tbegrow%+6,3
PRINT"t "
LOCATE tbegrow%+7,3
PRINT"  "
'ccccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Hemoglobin"
LOCATE tbegrow%+8,3
PRINT"H "
LOCATE tbegrow%+9,3
PRINT"e "
LOCATE tbegrow%+10,3
PRINT"m "
LOCATE tbegrow%+11,3
PRINT"o "
LOCATE tbegrow%+12,3
PRINT"g "
LOCATE tbegrow%+13,3
PRINT"l "
LOCATE tbegrow%+14,3
PRINT"o"
LOCATE tbegrow%+15,3
PRINT"b "
LOCATE tbegrow%+16,3
PRINT"i "
```

```
LOCATE tbegrow%+17,3
PRINT"n"
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Saturated"
tbegrow%=10
LOCATE tbegrow%,6
PRINT"S "
LOCATE tbegrow%+1,6
PRINT"a"
LOCATE tbegrow%+2,6
PRINT"t "
LOCATE tbegrow%+3,6
PRINT"u "
LOCATE tbegrow%+4,6
PRINT"r "
LOCATE tbegrow%+5,6
PRINT"a"
LOCATE tbegrow%+6,6
PRINT"t "
LOCATE tbegrow%+7,6
PRINT"e"
LOCATE tbegrow%+8,6
PRINT"d "
LOCATE tbegrow%+9,6
PRINT""
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "with"
LOCATE tbegrow%+10,6
PRINT"w "
LOCATE tbegrow%+11,6
PRINT"i"
LOCATE tbegrow%+12,6
PRINT"t"
LOCATE tbegrow%+13,6
PRINT"h"
LOCATE tbegrow%+14,6
PRINT""
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Oxygen"
LOCATE tbegrow%+15,6
PRINT"O"
LOCATE tbegrow%+16,6
PRINT"x"
LOCATE tbegrow%+17,6
PRINT"y"
LOCATE tbegrow%+18,6
PRINT"g"
LOCATE tbegrow%+19,6
PRINT"e"
LOCATE tbegrow%+20,6
PRINT"n"
'ccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print scale
'ccccccccccccccccccccccccccccccccccccccccccccccc
'set column for placement of temperature scale
cc%=19
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 50
LOCATE 28,cc%
PRINT 50
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 60
LOCATE 17,cc%
PRINT 60
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 70
LOCATE 13,cc%
PRINT 70
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 80
LOCATE 9,cc%
PRINT 80
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 90
LOCATE 5,cc%
PRINT 90
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 100
LOCATE 1,cc%-1
PRINT 100
'ccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print titles for ASAT and VSAT
```

```
COLOR 12,BGCOL%
LOCATE 22,1
PRINT "ASAT"
COLOR 3,BGCOL%
LOCATE 22,7
PRINT "VSAT"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'PRINT Oxygen consumption scale
'ccccccccccccccccccccccccccccccccccccccccccccccccc

'set color to red text and yellow background
color 10,bgcol%
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'assign beginning row for title to variable
'tbegrow%
tbegrow%=2
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'assign beginning column to variable tbegcol%
tbegcol%=24
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Oxygen"
LOCATE tbegrow%+1,tbegcol%
PRINT"O"
LOCATE tbegrow%+2,tbegcol%
PRINT"x  c"
LOCATE tbegrow%+3,tbegcol%
PRINT"y  c"
LOCATE tbegrow%+4,tbegcol%
PRINT"g"
LOCATE tbegrow%+5,tbegcol%
PRINT"e  O"
LOCATE tbegrow%+6,tbegcol%
PRINT"n  x"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Consumption"
'locate and print title for y axis
LOCATE tbegrow%+7,tbegcol%
PRINT"   y"
LOCATE tbegrow%+8,tbegcol%
PRINT"C  g"
LOCATE tbegrow%+9,tbegcol%
PRINT"o  e"
LOCATE tbegrow%+10,tbegcol%
PRINT"n  n"
LOCATE tbegrow%+11,tbegcol%
PRINT"s "
LOCATE tbegrow%+12,tbegcol%
PRINT"u  p"
LOCATE tbegrow%+13,tbegcol%
PRINT"m  e"
LOCATE tbegrow%+14,tbegcol%
PRINT"p  r"
LOCATE tbegrow%+15,tbegcol%
PRINT"t "
LOCATE tbegrow%+16,tbegcol%
PRINT"i  m"
LOCATE tbegrow%+17,tbegcol%
PRINT"o  i"
LOCATE tbegrow%+18,tbegcol%
PRINT"n  n"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print oxygen con title
LOCATE 22,61
PRINT "Ox Cont"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print scale
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'set column for placement of temperature scale
cc%=58
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 0
LOCATE 21,cc%
PRINT 0
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 50
LOCATE 17,cc%
PRINT 50
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 100
LOCATE 13,cc%
PRINT 100
'ccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
'locate and print 150
LOCATE 9,cc%
PRINT 150
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 200
LOCATE 5,cc%
PRINT 200
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 100
LOCATE 1,cc%
PRINT 250
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc

'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'PRINT Hemoglobin content
'ccccccccccccccccccccccccccccccccccccccccccccccccc

'set color to red text and yellow background
color 0,bgcal%
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'assign beginning row for title to variable
'tbegrow%
tbegrow%=2
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'assign beginning column to variable tbegcol%
tbegcol%=76
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Hemoglobin"
LOCATE tbegrow%+1,tbegcol%
PRINT"H "
LOCATE tbegrow%+2,tbegcol%
PRINT"e "
LOCATE tbegrow%+3,tbegcol%
PRINT"m "
LOCATE tbegrow%+4,tbegcol%
PRINT"o "
LOCATE tbegrow%+5,tbegcol%
PRINT"g  g"
LOCATE tbegrow%+6,tbegcol%
PRINT"l  m"
LOCATE tbegrow%+7,tbegcol%
PRINT"o"
LOCATE tbegrow%+8,tbegcol%
PRINT"b  p"
LOCATE tbegrow%+9,tbegcol%
PRINT"i  e"
LOCATE tbegrow%+10,tbegcol%
PRINT"n  r"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print "Content"
LOCATE tbegrow%+12,tbegcol%
PRINT"C"
LOCATE tbegrow%+13,tbegcol%
PRINT"o  d"
LOCATE tbegrow%+14,tbegcol%
PRINT"n  l"
LOCATE tbegrow%+15,tbegcol%
PRINT"t"
LOCATE tbegrow%+16,tbegcol%
PRINT"e "
LOCATE tbegrow%+17,tbegcol%
PRINT"n "
LOCATE tbegrow%+18,tbegcol%
PRINT"t"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print label for hemoglobin content
LOCATE 2,75
PRINT "Hb"
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print scale
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'set column for placement of temperature scale
cc%=71
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 0
LOCATE 21,cc%
PRINT 0
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 5
LOCATE 17,cc%
PRINT 5
```

```
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 10
LOCATE 13,cc%
PRINT 10
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 15
LOCATE 9,cc%
PRINT 15
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 20
LOCATE 2,cc%
PRINT 20
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'locate and print 25
LOCATE 1,cc%
PRINT 25
'ccccccccccccccccccccccccccccccccccccccccccccccccc
'ccccccccccccccccccccccccccccccccccccccccccccccccc

'DEFINE END OF SUBROUTINE OXSCALE
return
'ppppppppppppppppppppppppppppppppppppppppppppppppp
```

The vita has been removed from the scanned document