Computer controlled

blood sampling system

by

Mark William Tschopp

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Interdepartmental Program: Biomedical Engineering
Major: Biomedical Engineering

Iowa State University
Ames, Iowa

1993

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

William Robertson is responsible for designing and acquiring the Computer Controlled Blood Sampling System hardware. I would like to thank Bill for all his help throughout the project. When I refer to we in the text, I am refering to Bill and me. I would also like to acknowledge Kyle Holland's design of the voltage drop circuitry and his electronic help with the project. And I would like to thank Lyle Kesl for his help doing the cutdowns and preparing the dogs for testing.

INTRODUCTION

The Computer Controlled Blood Sampling System (CCBSS) was developed to eliminate the need for a researcher to be present when blood samples are taken from a pig. Merck, a pharmaceutical company, had the need for such a device and was willing to provide a grant for the development. This system will allow the researcher to set the appropriate sampling times, administer a test drug to a pig, and then do other work or leave for home while the system takes blood samples automatically. After the sponsor changed its requirements several times and new designs were drawn up, the present design was chosen and the necessary materials were purchased to assemble the system. Basic language was selected to develop the software based on the fact that the library routines included with the stepper motors were written in Basic. Software had to be written for the user interface, control of a hypodermic needle mounted to XY and Z tables, and to control valves which direct the blood flow.

The purpose for developing the CCBSS was to eliminate human intervention once the catheters have been inserted and a test drug administered to the pig. The situation requires that the pig(s) be confined to a very small pen to decrease the risk of having a catheter pulled out by the pig. Merck made several system requirement changes during the design phase. The current system was designed for two situations. The first situation uses three catheters in each pig, two venous and one arterial, with the ability to sample from two pigs. The second situation can sample from four pigs with one venous catheter in each pig. The system does not require that the maximum number of pigs be used in any given sampling cycle. The researcher can use the system for any number of pigs up to the maximum by entering the appropriate sampling times or lack of sampling times, indicating where no pig is present.

LITERATURE REVIEW

Robotic, closed tube processing devices are only now becoming available in laboratories [1]. Very few automatic blood sampling systems have been documented; no papers were found on a stationary blood sampling device that samples without human involvement after the initial set up.

The automatic blood sampling systems that have been developed have been designed for use with unrestrained animals. These devices have been used to sample from ducks and hens [2], Weddell seals [3], and greyhounds [4]. Each of these devices used a different method to collect and store the sampled blood. Scheid and Slama stored the sampled blood in a sampling catheter by simply kinking it, using a spring-loaded hinge which was operated by a magnet and activated by remote control [2]. Hill's multiple-sample collector collected blood from seals in amputated 20 ml syringes containing their rubber plungers from which the handles had been removed [3]. Schmalzried et al. constructed their own blood collection receptacles using a latex balloon within a 20 ml plastic syringe, and evacuating the tube [4]. These three systems were able to collect relatively few blood samples during a single testing session. Scheid and Slama's device was able to collect only one sample per testing session, Hill's, nine samples, and Schmalzried et. al., eight samples.

Glass evacuated blood-drawing tubes have been the standard device for obtaining blood from patients for clinical laboratory testing since the 1950s [1]. A vacutainer will pull in just enough blood to fill itself. As the vacutainer fills with blood, the vacuum disappears, and it is impossible to overfill it, even when the blood is under relatively high pressure. This allows a precise blood sample to be taken very easily and quickly. Vacutainers can also be purchased in a variety of sizes and diameters, which allows for various sample volumes.

Stepper motors afford simple means for controlled remote positioning of many devices. They are used in clocks, elapsed time indicators, pen drives for plotting curves,

indexing control, drivers for watt-hour meters, and many other applications [5]. A stepper motor is a special type of motor that lends itself to precise positioning under digital computer control [6]. The MD-2 Dual Stepper Motor Package from Arrick Robotics provides all of the components necessary to operate stepper motors from an IBM style personal computer including electronics, power supply, motor cables, software, and documentation. To provide three axis positioning, an 18 inch XY table and a 2 inch Z table are also available from Arrick Robotics, designed to be used with the MD-2 Dual Stepper Motor System.

MATERIALS AND METHODS

Hardware

A schematic of the system is shown in Figure 1 and photographs of the complete system are shown in Figures 2 and 3. Table 1 contains a list of all the hardware components that were used to develop the CCBSS. It was decided to store the blood in vacutainers pretreated with EDTA, an anticoagulant. Vacutainers allow a sample of a given amount of blood to be drawn easily and accurately. The vacuum will draw just enough blood to fill the tube. Our contact at Merck told us that the maximum number of samples that would be taken during any particular testing session was 72. We decided to construct a rack capable of

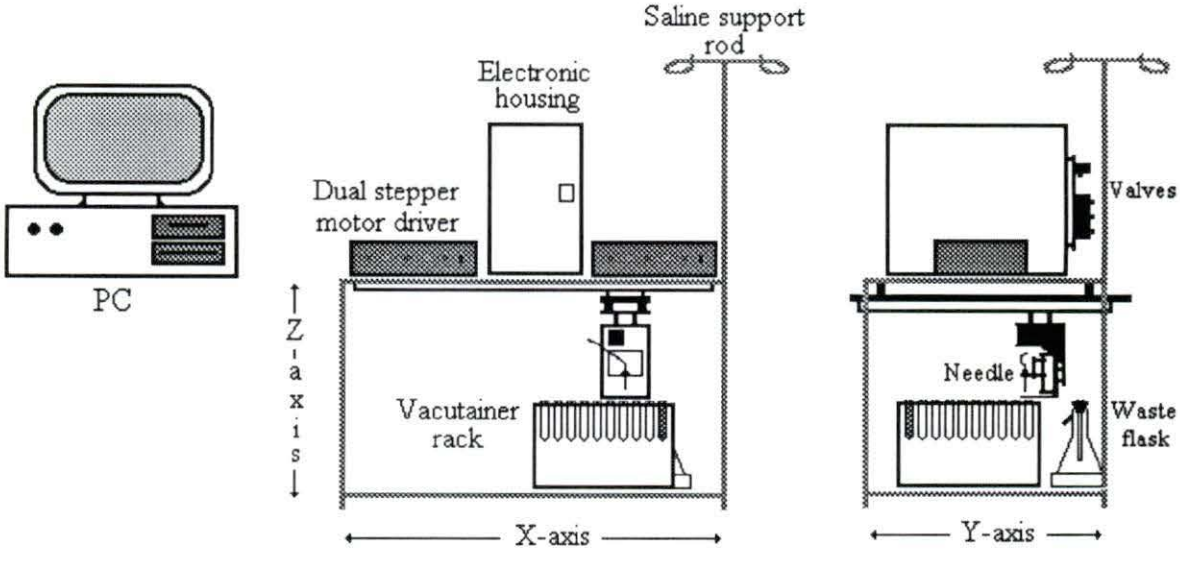


Figure 1 Schematic of the CCBSS

Figure 2 Right front view of the CCBSS

Figure 3 Front view of the CCBSS

Table 1 Hardware List

IBM compatible personal computer (PC) (386)
with Monitor, Keyboard, and Mouse
Second parallel card
CyberResearch, Inc. Digital I/O board

2 Parallel cables
1 Ribbon cable (14 line)

Arrick Robotics:
XY-18 table
Z-2 table and BR-2 right angle mounting bracket for vertical mounting of the Z-2
2 MD-2 Dual Stepper Motor Systems
includes stepper motors and 3 motor cables

Stand approximately 29.5"W X 22"D X 29.75"H
w/ aluminum bottom and saline support rod

100 Tube Plexiglas vacutainer rack, large diameter vacutainers
100 Tube Plexiglas vacutainer rack, medium diameter vacutainers

Erlenmeyer flask w/ vacuum port and stopper assembly
Stopper assembly: large rubber stopper w/ hole,
syringe barrel,
silicon septum

Reciprocating pump (Gast, model DOA)
functions as compressor and vacuum pump

Electronic housing containing:
5 and 12 volt power supply w/ fan
CyberResearch, Inc. Relay board w/ 8 relays
Voltage drop board

Mounted on the rear:
1 six inlet Teflon PTFE solenoid manifold valve (Cole Parmer, CP# 01367 - 83)
2 three-way Teflon PTFE solenoid valves (Cole Parmer, CP# 01367 - 72)

Pressure cuff for saline bags
3 Liter and 1 Liter

Hypodermic needle (16 gauge) and mounting bracket

Hypodermic needle fork

Teflon tubing and fittings

Polyether catheters with Luer locks

IV line set

Silastic tubing

holding 100 vacutainers, a 10 hole by 10 hole rack. The extra holes allow for future changes in the number of samples and allow the researcher to easily differentiate between samples from different pigs. When the system is set to use three catheters per pig, two pigs maximum, the vacutainer rack is divided as shown in Figure 4, with pig #1 samples being toward the inside of the stand. In this mode, the blood sample from catheter one is stored in every third vacutainer, starting with tube #1. Samples from catheter two are stored in tubes #2, 5, 8, 11, etc. Figure 5 shows how the vacutainers are divided between pigs when the system is set for sampling using one catheter per pig and a maximum of four pigs. Figure 6 shows a photograph of the rack. The original vacutainer rack was milled to hold 7 ml or 15 ml large diameter vacutainers. A second vacutainer rack was constructed with the top piece of Plexiglas milled with smaller diameter holes to allow the researcher to also use medium diameter vacutainers. The vacutainer rack is held in position by two pins, the racks can be easily interchanged by lifting one rack off the pins and replacing it with the other vacutainer rack. A hypodermic needle, shown in Figure 7, is required to penetrate the rubber top of the vacutainer and supply the blood path. The hypodermic needle needed to be movable

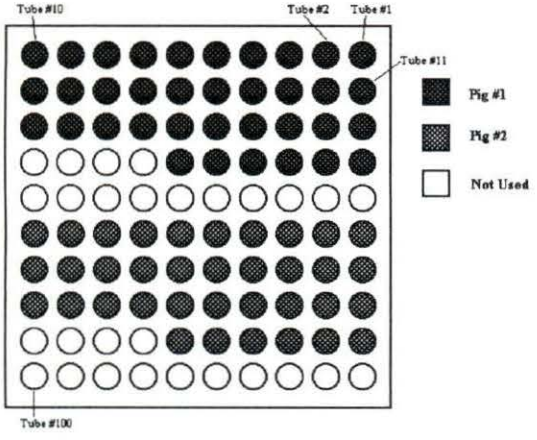


Figure 4 Tube placement using three catheters per pig

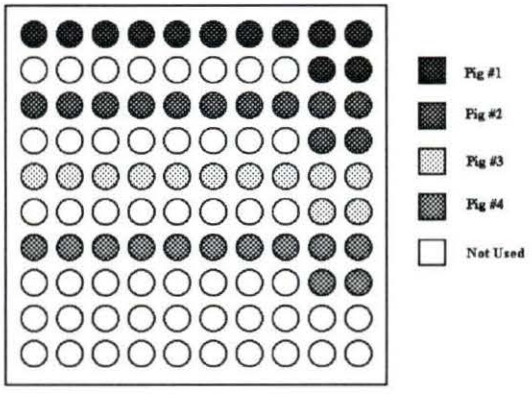


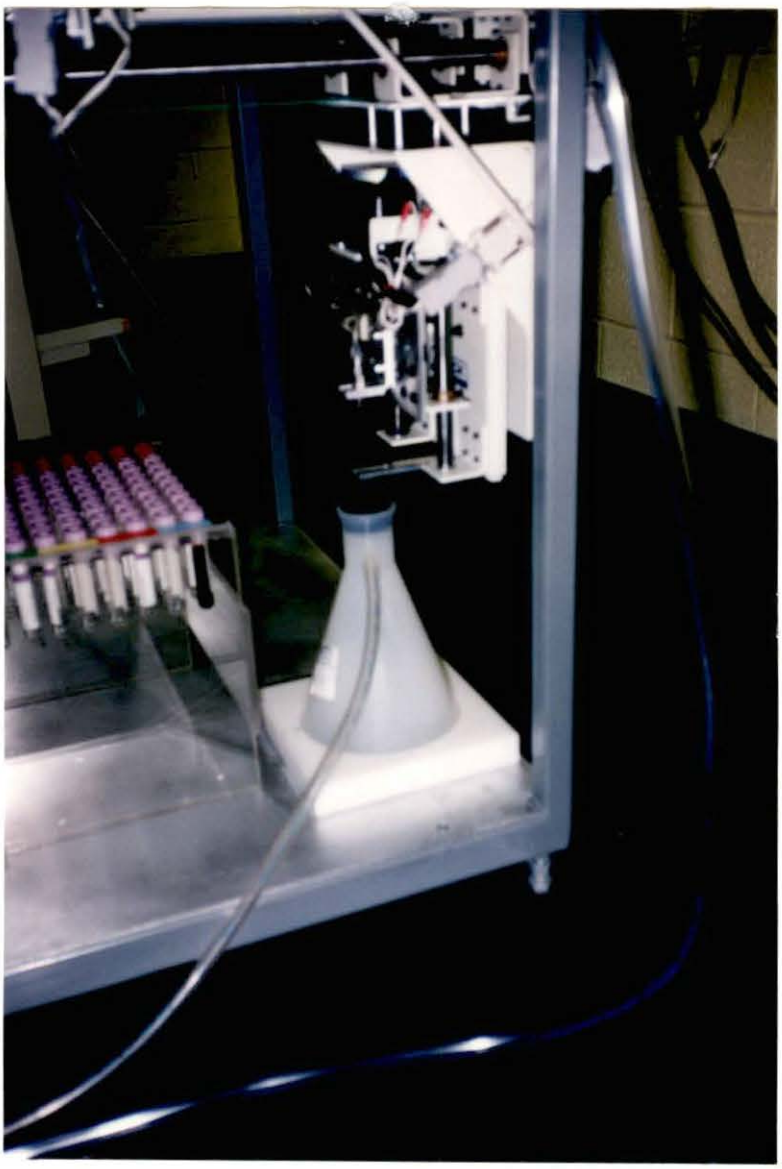
Figure 5 Tube placement using one catheter per pig

Figure 6 The vacutainer rack

movable to each vacutainer and the Erlenmeyer waste flask. The waste flask, shown in Figure 7, provides a storage container for waste saline and blood that is flushed out of the lines during sampling. The plastic waste flask is equipped with a vacuum port. A vacuum pump is attached to the waste flask vacuum port using a section of Silastic tubing. The vacuum pulls the blood and saline through the lines and into the waste flask. The waste flask also uses a syringe barrel, containing a silicon septum, within a large rubber stopper. The syringe barrel, above the silicon septum, can then be filled with Novosan to rinse the needle every time the needle pierces the silicon septum. The silicon septum was used because it is capable of receiving many more needle piercings before deteriorating, than a vacutainer stopper. The task of moving the hypodermic needle was accomplished by mounting the hypodermic needle on XY and Z tables controlled by stepper motors.

The XY-18 and Z-2 tables were purchased from Arrick Robotics along with two MD-2 Dual Stepper Motor Systems. The motor drivers are connected to the IBM compatible personal computer via two parallel cables, one for each motor driver. A second parallel board was purchased and installed in the PC for this purpose. Each motor driver is capable of controlling two motors. The manufacturer, Arrick Robotics, does not have a triple stepper motor driver available, so we were forced to purchase two dual stepper motor drivers. This gives us the ability to control four motors, but we are only using three. Each stepper motor is connected to a motor driver via a motor cable, supplied with the motor drivers. The motor cable also carries the limit switch signal, which is used to move the motor to the "home" position. The dual stepper motor drivers should be connected to the PC so that the motor driver connected to the original parallel port (left as viewed from rear of PC) is also connected to the X axis (left to right movement) stepper motor via motor 1 motor cable and to the Z axis (vertical movement) stepper motor via motor 2 motor cable. The other motor

Figure 7 The hypodermic needle and waste flask



driver should be connected to the installed (right) parallel port with motor 1 motor cable connected to the Y axis (front to back movement) stepper motor.

The XY-18 table is shown in Figure 8 and the Z-2 table is shown in Figure 9. The Z-2 table was mounted to the XY-18 table using the BR-2 bracket, allowing for motion in all

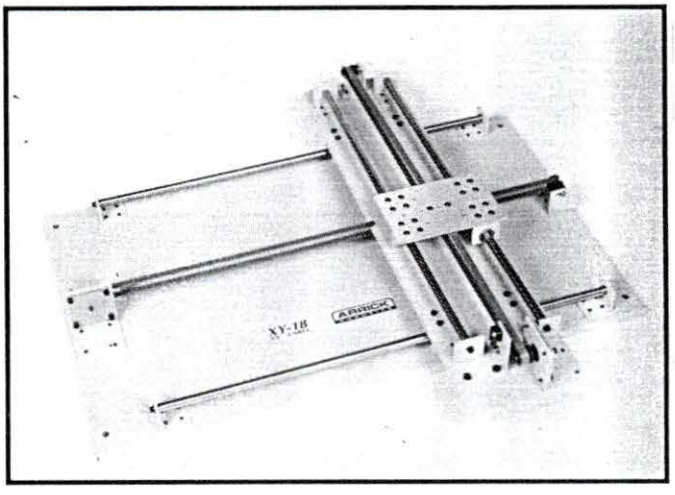


Figure 8 XY-18 table. From Arrick Robotics advertising flyer

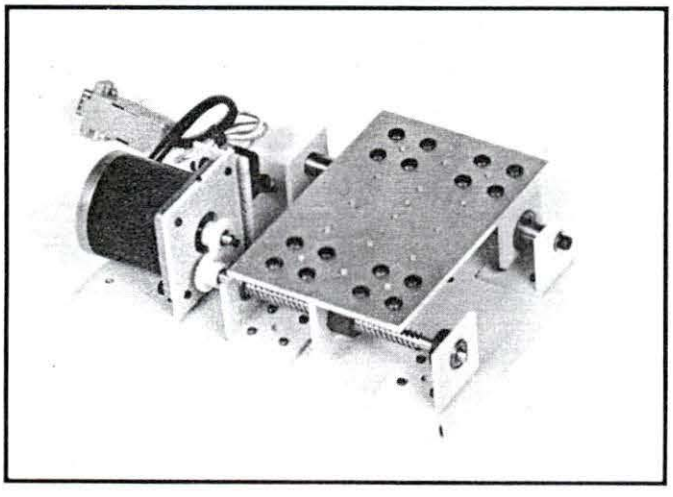


Figure 9 Z-2 table. From Arrick Robotics advertising flyer

three coordinate directions. During initial testing of the system, the X axis was found to bind and require a push to start moving. This was not acceptable; the tables could never bind; they must always move exactly as expected if the system is going to require no human involvement during sampling. Eventually, the bronze bearings were bored 1/1000th of an inch, and the steel shafts were aligned using a large caliper. This eliminated the binding and then all axes could be precisely controlled without failure.

A stand, approximately 29.5″ wide by 22″ deep by 29.75″ high, was manufactured from square steel tube to invert the XY and Z tables. The XY table frame was mounted to the stand upside down using four bolts to invert the XY and Z tables. A hypodermic needle was mounted to a bracket, which was then mounted to the Z table. The hypodermic needle could now be moved about above the vacutainers, vacutainer rack, and waste flask. A fork was mounted to the Z table frame so that the hypodermic needle can pass through the fork opening when it is driven into a vacutainer or the waste flask. When the needle is removed from the vacutainer (or waste flask), the fork keeps the vacutainer (or waste flask) from rising with the needle keeping it in its proper position.

The stand was manufactured with an adjustable saline support rod and an aluminum bottom was attached to the stand. We then placed pins in the aluminum bottom to locate the vacutainer rack and the flask rack in the proper positions. The flask rack was positioned so that the flask opening was centered below the hypodermic needle when both axes of the XY table were in the "home" position. The "home" position of a table is that position which results when the table is moved and triggers the limit switch and then the table is moved in the opposite direction only enough to deactivate the limit switch.

The vacutainer rack was located such that the rightmost column of holes in the vacutainer rack is centered under the hypodermic needle when the X axis of the XY table is

in the "home" position. Extreme care was taken to assure that the vacutainer rack was positioned square with the XY table. This allowed for the development of two equations for positioning the hypodermic needle above any of the vacutainers and eliminated the need to find the location of each vacutainer by trial and error and then store the values for future reference. Each vacutainer center is located 170 motor steps, in the X or Y direction, from the center of the vacutainer adjacent to it. The following equations were used to position the hypodermic needle based on the tube number:

$$\text{COLUMN (X axis)} = ((\text{tube} - 1) \bmod 10) * 170$$

$$\text{ROW (Y axis)} = 1070 + ((\text{tube} - 1) \setminus 10) * 170$$

where tube 1 is in row 1, column 1 (innermost, rightmost tube), tube 2 is in row 1, column 2, tube 11 is in row 2, column 1, etc., shown in Figure 4. The tube number is determined by the number of catheters per pig being used, the pig number, the catheter being sampled from, and which sampling sequence is being taken (1-12).

The saline support rod described earlier was included in the design to provide a place to hang a saline bag within a pressurized cuff. The saline is needed to flush the catheters, valves, and Teflon tubing after each sample is taken. This procedure will keep blood from different pigs from mixing and also keep the catheters patent (free from clotting). The pressurized cuff is required to push the saline through the catheters, valves, and tubing, especially when samples are being taken from an artery. The pressurized cuff must be able to push the saline against the arterial pressure, 150 mm Hg seemed to be appropriate when the system was tested. The pressure for the cuff is supplied by the compressor function of the reciprocating pump.

Several valves are necessary to control the flow of blood and saline. A six inlet, one outlet, Teflon PTFE solenoid manifold valve (Cole Parmer CP# 01367 - 83) and two three-way Teflon PTFE solenoid valves (Cole Parmer CP# 01367 - 72) were purchased from Cole

Parmer. The inlets of the six inlet valve are labeled 1 through 6, starting with 1 in the upper right (1 o'clock position) and increasing clockwise: 2 is the right inlet, 3 the lower right, 4 the lower left, 5 the left, and 6 the upper left. When one catheter per pig is being used, catheters 1 through 4 should be connected to pig #1 through pig #4, respectively, if the pig is present. Catheters 5 and 6 are never used in this mode. When sampling using three catheters per pig, catheters 1 through 3 should be used in pig #1, and catheters 4 through 6 are used to sample from pig # 2. These valves are connected as shown in Figure 10, and Figure 11 shows a

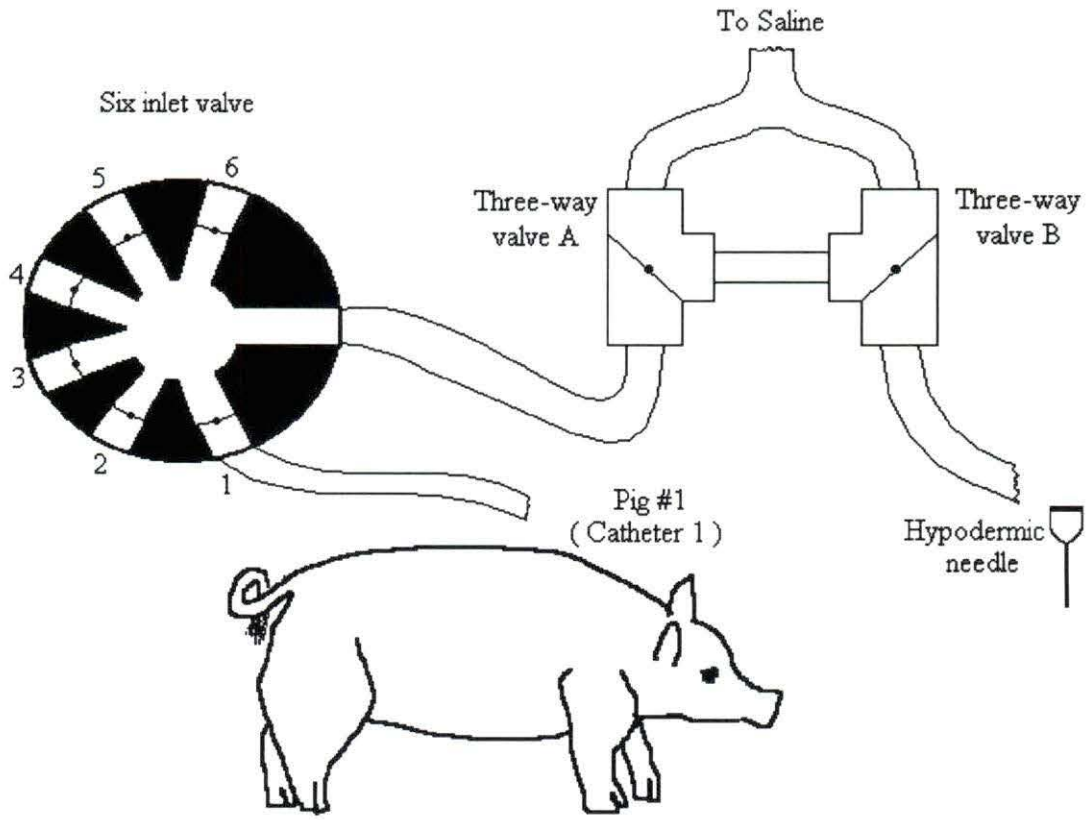
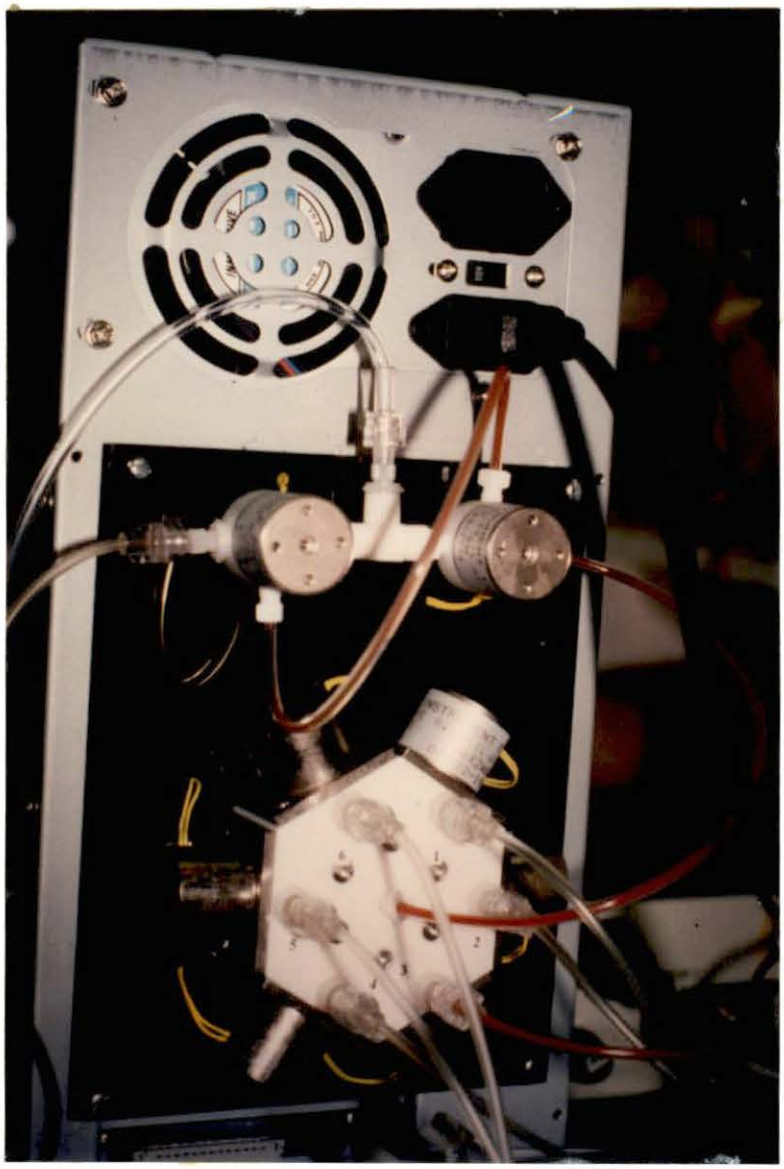


Figure 10 Valve connections

Figure 11 The solenoid valves



photograph of the valves. Solid state relays are required to control each of the valves, eight total. A relay board with eight relay sockets, eight relays, an I/O board and a ribbon cable were purchased from CyberResearch, Inc. The I/O board was installed in the IBM compatible PC. The ribbon cable is used to connect the I/O board with the relay board.

It was recommended by the valve manufacturer that the voltage to the valves be decreased from 12 volts if the valves were to remain open for more than two minutes, to avoid overheating. We decided that this was a good idea and constructed a voltage drop board, the circuit diagram is shown in Figure 12. There are eight identical circuits on the board, one for each relay / valve combination. The function of this board is to supply a valve with the voltage required to open the valve, 12 volts, when the relay first switches on for approximately one second. After this time, the voltage is decreased to approximately 3.8 volts, which is enough to keep the valve open without the risk of overheating.

An electronic housing equipped with a 5 and 12 volt power supply and cooling fan was purchased to mount the relay and voltage drop boards. A photograph of the inside of the electronic housing is shown in Figure 13. The valves were mounted to a piece of Plexiglas which was then attached to the rear of the electronic housing. The 12 volt power supply voltage sourcing the relays was measured to be only 11.04 volts. This 11.04 volts was decreased to 9.21 volts across the voltage drop circuitry before reaching the valves. It was discovered that 9.21 volts was not enough to open some of the valves, a voltage closer to 12 volts was required. It was necessary to load the 5 volt source with a 10 Ω, 25 watt resistor for the power supply to source 12.40 volts to the relays and 10.6 volts to the valves, enough to reliably open the valves.

Software

The accompanying software from Arrick Robotics to control the stepper motors was written in Basic. Basic was chosen based on this. An important consideration for this

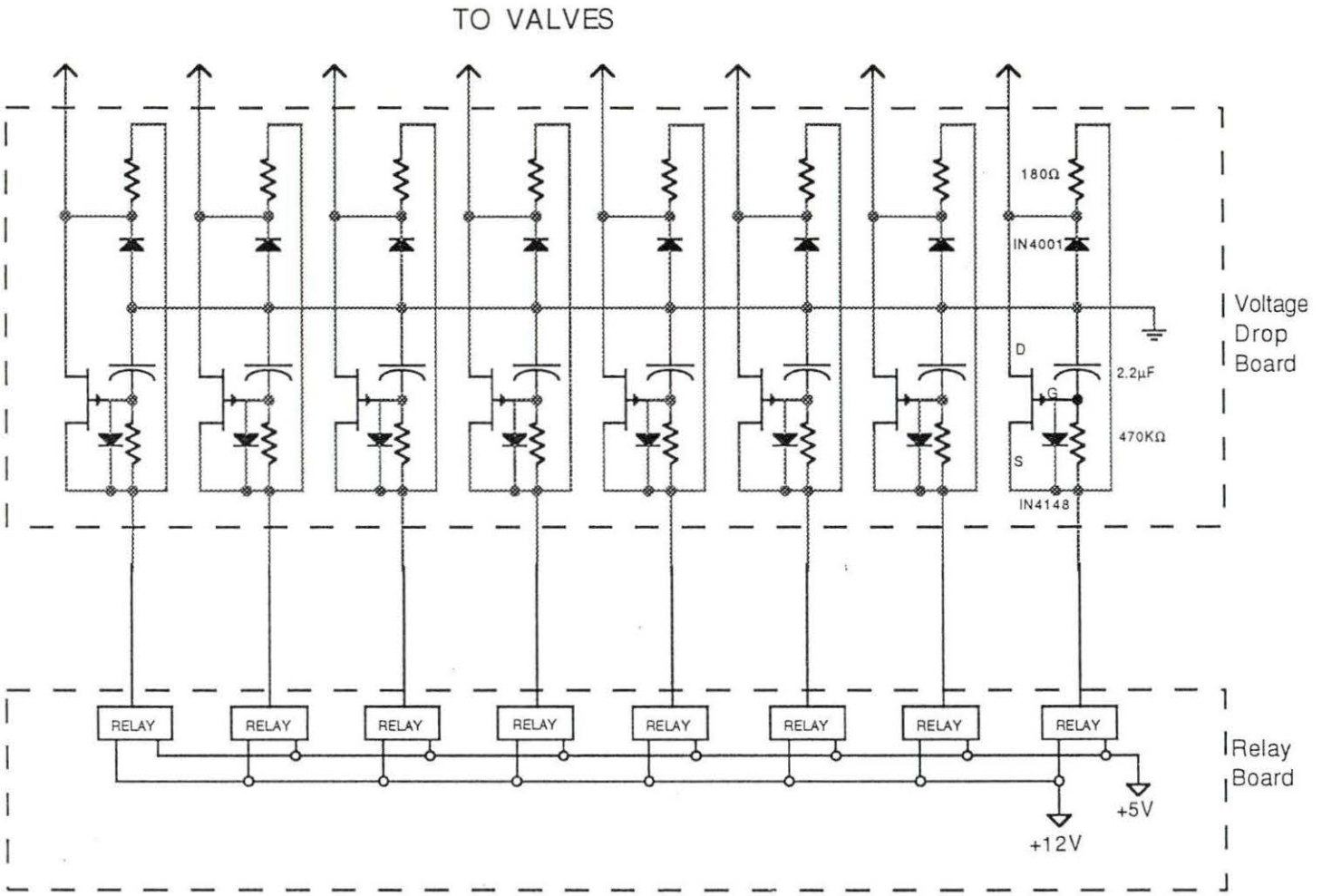
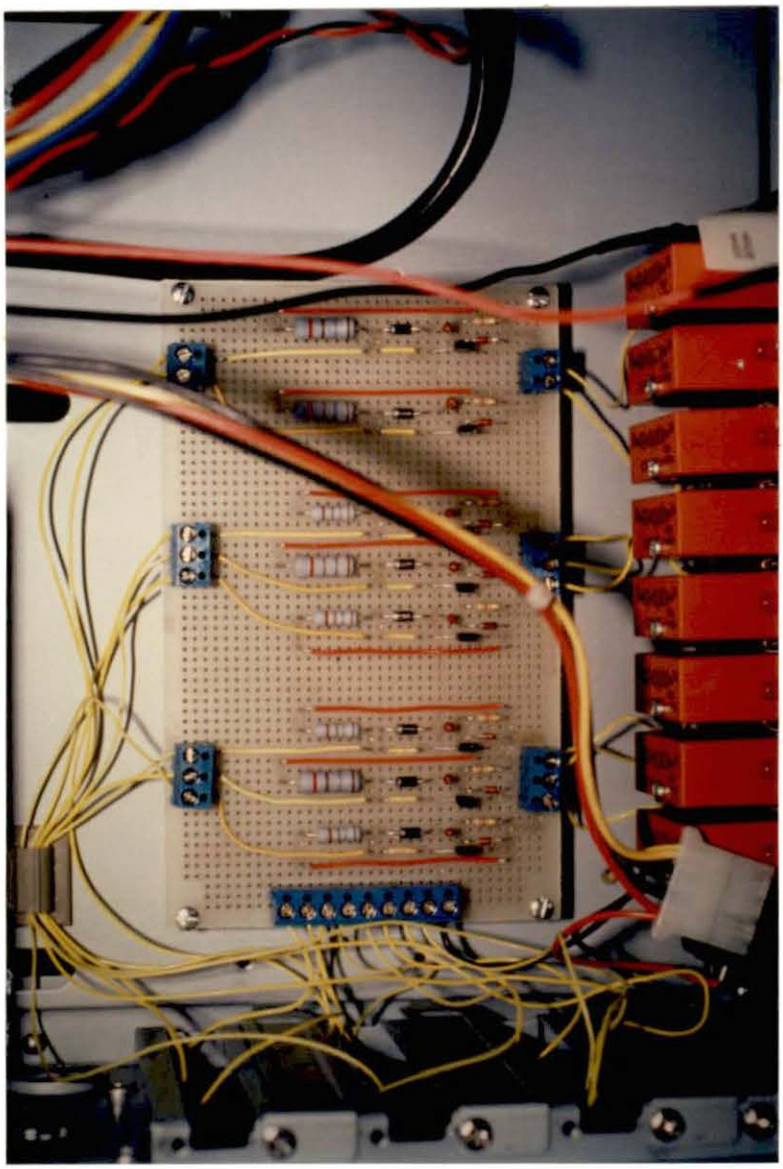


Figure 12 Voltage drop board circuit diagram and connections

Figure 13 Electronic housing showing voltage drop board and relay board



project's software was that it should be as user friendly as possible. The assumption was made that the researchers who would be using this application had little computer experience. The decision to use Visual Basic for MS-DOS was made after reviewing the Visual Basic Programmer's Guide [7]. Visual Basic provides the programmer with the ability to create screen forms very easily and quickly. The form design tool is object oriented and allows the programmer to place text fields, command buttons, labels, etc. on the screen form to create a user interface very easily. This eliminates the need to write the code to create these features, only the code that executes based on using the features needs to be written (event-driven programming). Menu bars and windows can be created just as easily using Visual Basic. These features allow the programmer to create a user interface that is simple to use even for the novice computer user. Some knowledge of Windows based software is helpful to know how to click and double click objects, chose menus and options, etc., but this knowledge can be acquired through some experimentation with the application.

A flow chart of the computer program is shown in Figure 14. The program is divided into three modules, each of which contains several sub procedures. Declaration of sub procedures, variables, arrays, etc. are included at the module level. The CMNDLG.BAS and CMNDLGF.FRM modules are provided with Visual Basic; these modules were used to acquire the necessary file information needed to save and retrieve sample times from a data file. The software provided by Arrick Robotics to control the stepper motors is contained within the take_sample procedure.

Visual Basic executes certain procedures automatically when a specific event occurs. For example, when the menu option New (CtlName = mnu_new) is chosen by the user, by clicking on it, Visual Basic automatically performs the sub procedure mnu_new_click if it is present. The sub procedure form_load is performed automatically when the program is run; this allows the programmer to set initial screen parameters, variables, or perform any

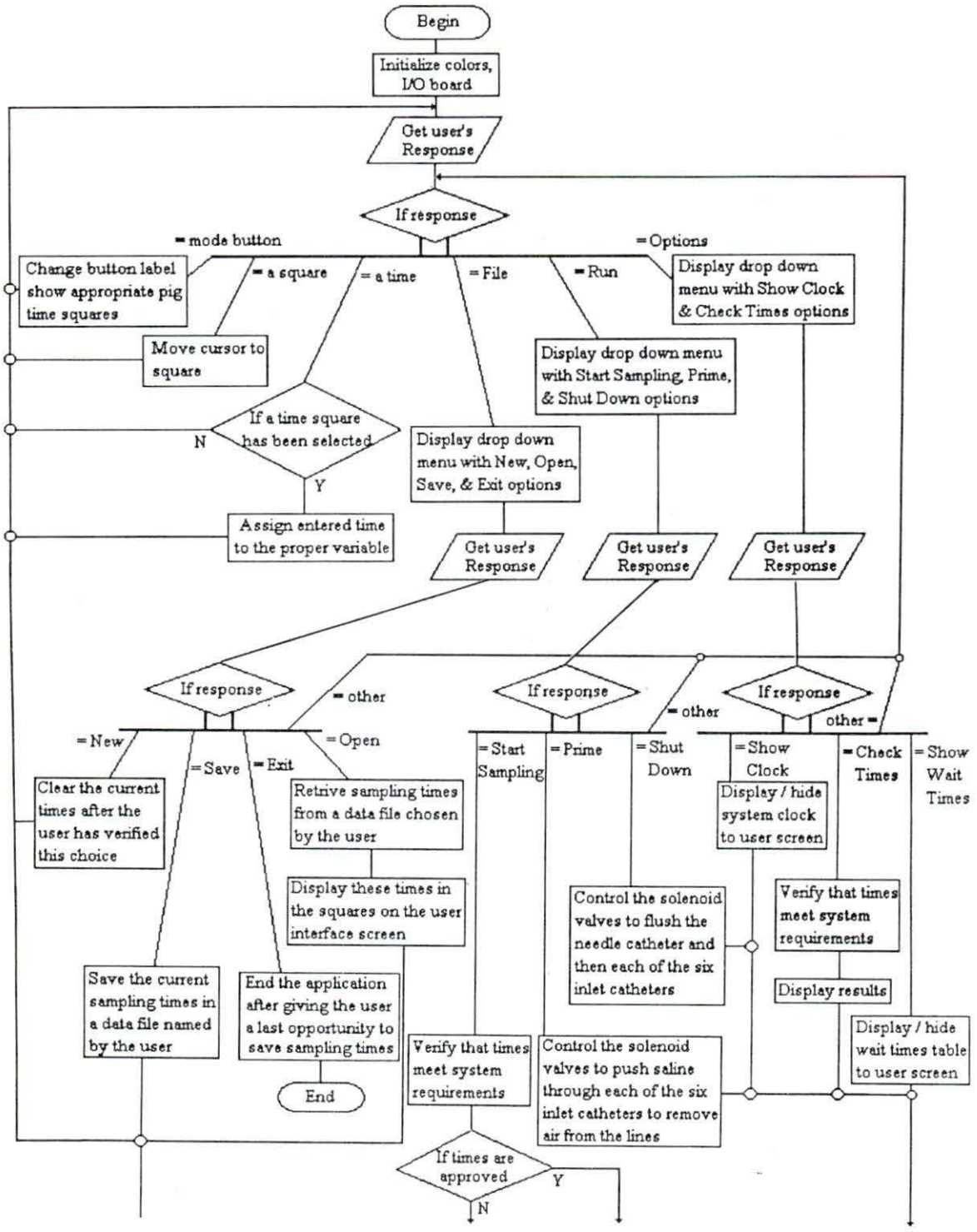


Figure 14 Flow chart of the computer program

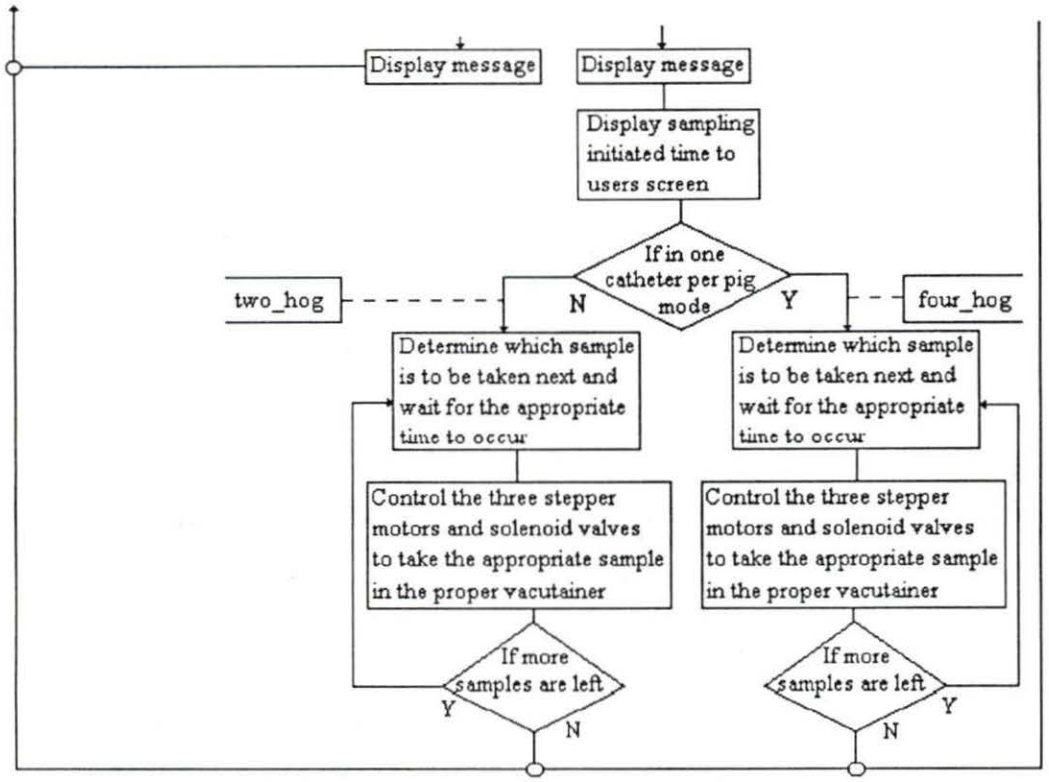


Figure 14 (continued)

important initialization procedures, such as initializing the I/O board. The CCBSS program listing can be found in Appendix A. The compiled assembly language program provided with the I/O board was used to create a .QLB file to be used by the Visual Basic program. The autoexec.bat file on the PC was modified to run the CCBSS application automatically when the PC is turned on. If the program is exited, the user will be in Visual Basic. To run the application from this point, the user can select start from the run drop down menu. If the user is in DOS, the application can be restarted by typing "sampling" and hitting Enter. When an executable version (.EXE file) of the application was created and run, the stepper motors simply buzzed without moving. It is believed that this problem is due to the faster compiled version switching the stepper motor phase windings on and off too fast, not allowing the motor enough time to turn.

OPERATION

Once all the cables have been properly connected, the IBM compatible PC should be turned on first. Then both stepper motor drivers, the switch on the front of the electronic housing, and the reciprocating pump can be turned on. The CCBSS application will appear showing the user interface screen. Initially, the screen shows four rows of twelve squares each, labeled pig #1 through pig #4. The sampling times are to be entered in these squares. In order to enter a time in a particular square, move the mouse pointer to that square and click on it. The cursor will appear in the square for the user to enter a sampling time. The Tab key can also be used to move from square to square while entering sampling times. The sampling times are entered as the number of minutes after sampling is started that the sample should be taken. If a sample is to be taken an hour and a half after sampling starts, for example, 90 should be entered in the appropriate square. Fractions of minutes may be included in expressing the sampling time, for instance, 24.5 could be entered. No squares should be left blank between sampling times in any particular row. There is a command button located toward the top of the screen at all times, which allows the user to switch between modes using one catheter or three catheters. When the application is set to sample using one catheter per pig, the initial setting, all four pigs are listed on the screen and the button is labeled "2 pigs w / 3 catheters". By clicking the button, the user can switch to sample using three catheters in each pig. When in three catheters mode, only pig #1 and pig #2 are listed on the screen, and the button label changes to "4 pigs w / 1 catheter".

A menu bar is located at the top of the screen with three selections: File, Run, and Options. A drop down menu will show several options when each of these selections are clicked. Table 2 shows all of the options in each of the drop down menus. The File selection drop down menu contains the options: New, Open, Save, and Exit. The New option clears the current times for all pigs; all squares will be shown blank. A warning window appears

Table 2 Drop Down Menu Options

File drop down menu
- New
- Open
- Save
- Exit

Options drop down menu
- Show Clock
- Check Times
- Show Wait Times

Run drop down menu
- Start Sampling
- Prime
- Shut Down

before clearing to verify that the user wants to erase the current sampling times. The Save option stores the current sampling times in a data file on disk. A window appears to request the file name and path as well as the drive, C or A (internal or 3.5 inch floppy drive, respectively). The Open option allows the user to retrieve previously saved sampling times from disk. A window requesting the file, path, and drive will also appear for this option. The Open and Save options allow the user to enter a set of frequently used sampling times only once, save the times, and then retrieve them whenever the times are to be used again. The Exit option stops the application and returns the user to Visual Basic. Before exiting the application, a warning window appears giving the user a last chance to save any sampling times that are currently entered.

The Options drop down menu contains three choices: Show Clock, Check Times, and Show Wait Times. The PC system clock is shown in the upper right of the screen. The user can chose not to display the clock by clicking on the Show Clock drop down menu option.

The dot to the left of the show clock option will disappear as will the time on the screen form. The clock can be displayed in the same manner if it has been previously hidden. The time on the screen form and the dot to the left of the Show Clock option both reappear when the Show Clock option is clicked at this point. The system clock can be changed by exiting the application, exiting Visual Basic, typing "time" at the system prompt, and then entering the new time in the same format as the current time, shown on the screen.

The Check Times option allows the user to verify that the current sampling times meet the system requirements. When sampling using one catheter per pig, every sampling time must be at least 2 minutes apart from every other sampling time. This 2 minutes allows enough time to take a sample and be ready to take the next sample. When the system is set to take samples using three catheters per pig, sampling times are required to be at least 6 minutes apart. In this mode, three blood samples are taken during each sampling sequence, thus the minimum amount of time between sampling sequences is tripled. If the sampling times meet the system requirements, a message is displayed in a window indicating the approval of sampling times. The user should click on the OK button within the window after the message has been read. If the sampling times do not meet the system requirements, a message is displayed to the user in a window to indicate that times must be altered before sampling can begin. Included in the message is a suggestion as to which pigs and columns the user should examine to correct the problem.

When the system is set to sample using three catheters per pig, the sampling times for pig #3 and pig #4 are not displayed and are not used during sampling. But if sampling times have been entered for pig #3 and pig #4, the times must meet the requirement of being at least 6 minutes apart from all other times, even though they are not used. The best advice is to leave the times of pig #3 and pig #4 blank when sampling using three catheters. The Show Wait Times option will be discussed after the sampling operations have been described.

The options in the Run drop down menu are Start Sampling, Prime, and Shut Down. The Start Sampling option, when clicked, executes the Check Times routine to verify that the sampling times meet system requirements before sampling begins. When the sampling times meet system requirements and the user clicks the OK button, the time that sampling began is displayed on the screen above the current time. From this point on, the program waits for the next sampling time to occur. The sampling times cannot be changed after the start sampling option has been selected.

When a sampling time occurs, the user interface is removed from the screen and the hypodermic needle is moved to a position above the waste flask and is driven down through the silicon septum. All stepper motors are powered down (motors' phases are turned off) after they are run, keeping the motors and dual stepper motor drivers cool. At this time, the valves are changed from being all closed, the wait state, Figure 15, to the waste state, where the two three-way valves are both open and one inlet of the six-inlet valve is open, Figure 16. Blood is represented by light shading and saline by dark shading in these valve diagrams. The vacuum line attached to the waste flask pulls blood from the pig through the catheter attached to the open inlet of the six-inlet valve and removes the saline from the lines, replacing it with blood. Then, the open inlet of the six-inlet valve is closed and the hypodermic needle is removed from the waste flask. The hypodermic needle is then moved to a position above the correct vacutainer, which is determined by the number of catheters per pig being used, the pig number, the catheter being sampled from, and which sampling sequence is being taken (1-12). The hypodermic needle is driven through the vacutainer stopper at this time and the same inlet of the six-inlet valve is reopened, the sample state, shown in Figure 16. The vacutainer pulls the blood through the lines and almost fills the vacutainer with blood. Three-way valve A is then closed and the pressure from the saline forces some of the blood in the line into the vacutainer to fill it to an appropriate level, shown

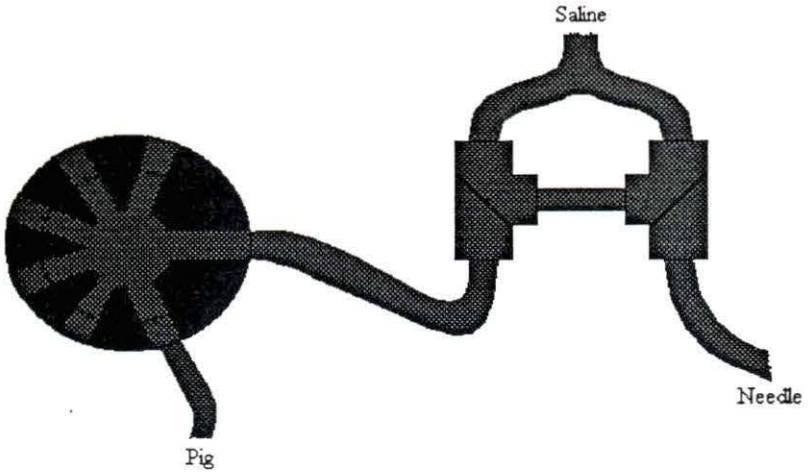


Figure 15 Valves showing the wait state

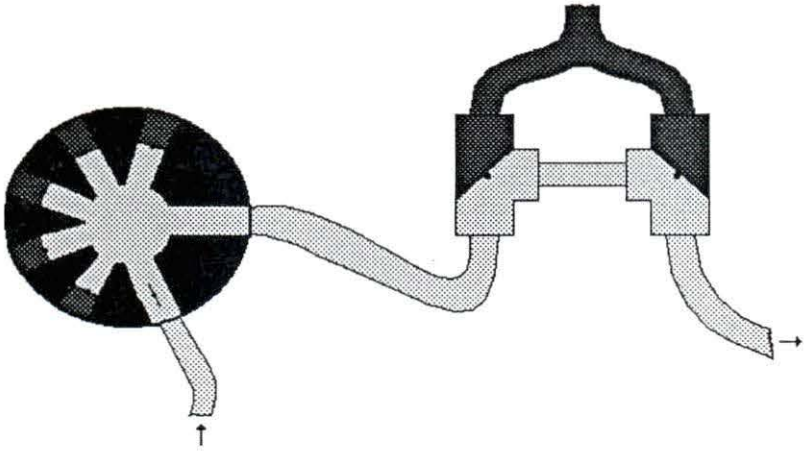


Figure 16 Valves showing the waste and sample states

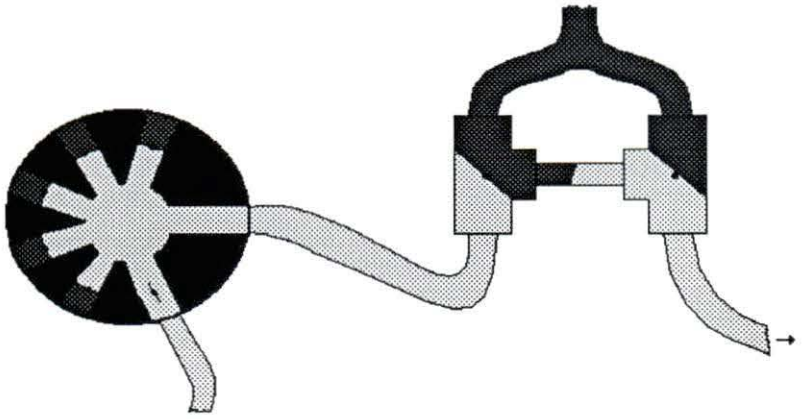


Figure 17 Valves showing the saline push to completely fill the vacutainer

in Figure 17. At this point, three-way valve B is closed, three-way valve A is opened, and the open inlet of the six-inlet valve remains open. The pressure on the saline bag forces saline through the lines and valves as shown in Figure 18, to flush the blood in the lines on the pig side of the valves back into the pig. While the pig side lines are being flushed, the hypodermic needle is removed from the vacutainer, returned to a position above the waste flask, and driven through the silicon septum of the waste flask stopper assembly. The open inlet of the six-inlet valve is then closed, three-way valve A is closed, and three-way valve B is opened, causing saline to flush the blood in the lines on the needle side of the valves into the waste flask, shown in Figure 19.

All valves are then closed; the hypodermic needle remains in the waste flask. The six-

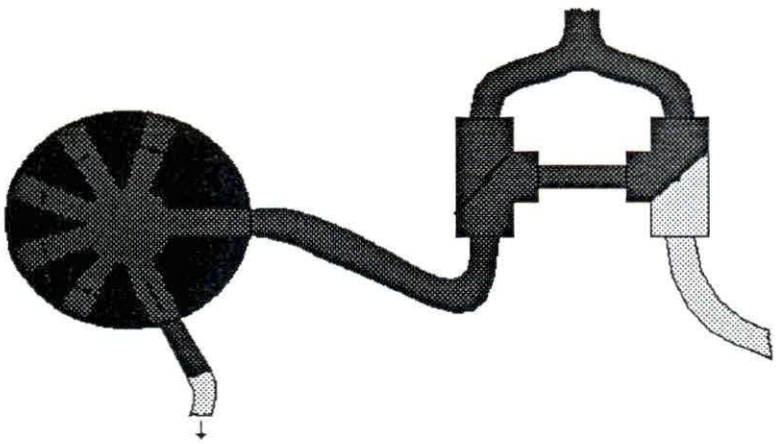


Figure 18 Valves showing a flush of the lines on the pig side of the valves

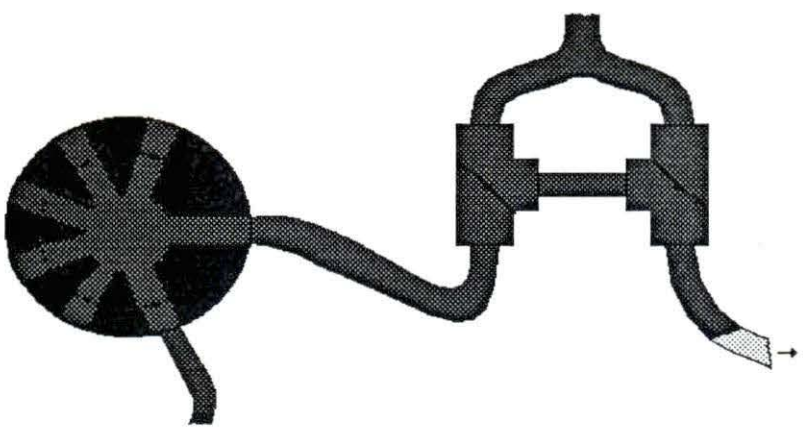


Figure 19 Valves showing a flush of the lines on the needle side of the valves

inlet valve, regrettably, is not a zero dead space valve; some residual blood can remain in the valve. To remove this blood and avoid damaging the valves, all six inlets are opened, as are both three-way valves. Blood from each pig or saline from a container is drawn about half-way through each catheter, three-way valve A is closed, all inlets of the six inlet valve are closed, and the needle side lines are flushed. Three-way valve A is then opened, three-way valve B is closed, and each inlet of the six inlet valve is opened one at a time to flush the blood or saline in the catheters back into the pig or saline container. All inlets and valves are then closed and the hypodermic needle is removed from the waste flask. Because of this routine to remove any residual blood from the six-inlet valve, six catheters must always be connected to the six-inlet valve and any unused catheters should be placed in a container of saline. The sampling procedure is summarized in Table 3.

The third option in the Options drop down menu, Show Wait Times, allows the user to modify the time delays for drawing waste, flushing back into the pig, and the pull and push times for cleaning out the six inlet valve. When the Show Wait Times option is selected, a table of delay times will appear below the rows of sampling times, and a dot will appear next to Show Wait Times menu option. If the table of delay times is currently shown on the user screen, selecting Show Wait Times will remove the table from the screen and remove the dot next to the menu option. The four delay times mentioned above can be set individually for each of the six catheters. The waste time delay is the time that the hypodermic needle remains in the waste flask at the beginning of sampling when blood and saline are pulled through the catheter. The flush time delay is the amount of time that saline is pushed back into the pig after the needle has been moved from a vacutainer to the waste flask. The pull time delay is the time in which blood is drawn into each of the catheters to agitate the six inlet

Table 3 Sampling Procedure

Remove user interface from the screen
Move needle to home position and drive into waste flask
Open both three-way valves and one of the six inlets
Wait (variable) to remove saline and fill lines with blood
Close the open inlet
Remove the needle from waste flask, move above the correct vacutainer, and drive into vacutainer
Reopen inlet
Wait for vacutainer to draw blood sample
Close three-way valve A
Wait to push extra blood into vacutainer
Open three-way valve A, close three-way valve B
Remove the needle from vacutainer, move to home position, and drive needle into waste flask
Wait (variable) to flush blood out of lines on the pig side of valves
Close open inlet, close three-way valve A, open three-way valve B
Wait to flush blood out of lines on the needle side of valves
Open both three-way valves and all inlets
Wait (variable) to draw blood or saline partially up all six catheters
Close three-way valve A and all inlets
Wait to flush out lines on the needle side of valves
Close three-way valve B, open three-way valve A, and open each of the six inlets one at a time (Wait, variable, for each catheter to flush back into the pig)
Close all valves
Remove needle from waste flask

valve, and the push time delay is the time that saline is then pushed through each of the six catheters to force the blood back into the pig(s).

The catheter sizes used, the blood vessels (arteries or veins) used, and the pigs' blood pressure are all possible factors that may require the user to adjust the delay times. A small catheter will not allow as much fluid as a larger catheter to flow through in a given amount of time at a given pressure. This decrease in fluid flow requires the blood and saline to be drawn and flushed for a longer period of time for the same volume of fluid to pass through. When sampling from an artery or a pig with high blood pressure, less time is needed to draw blood, but more time is needed to push blood and saline back into the pig because of the increased pressure. The user should try catheters of various sizes in various sampling arrangements to become familiar with the time delay changes that may be required in situations that vary from the usual. However, when the delay times are lengthened, the Check Times routine will no longer be accurate. The user must check the time duration of an individual sample and make sure that an appropriate amount of time is allowed between samples, so that one sample procedure ends before the next begins.

The second option in the Run drop down menu is Prime. This routine should be performed before any catheters are connected to the pigs. The Prime routine flushes saline through the six catheters connected to the six inlet valve to remove any air and fill the lines with saline. It is suggested that all six catheters be placed in a large container of saline and then choose the Prime option. Three-way valve A is opened and then each of the inlets of the six inlet valve are opened, one at a time. The pressurized saline bag then forces saline through all six of the catheters for approximately 5 seconds each. All of the valves are then closed. At this point, the catheters can be connected to the pigs; any unused catheters should remain in the saline container as described earlier.

In the Run drop down menu, the third option is Shut Down. This routine should be performed when sampling is completed. The valves need to be flushed with sterile water to remove the saline from the valves. It is possible for the salt to precipitate out of the saline if the saline remains in the valves for an extended period of time. The salt can then damage the valve diaphragm causing the valve to leak and require a replacement. Before this option is chosen, the catheter connected to the saline bag should be placed in a bag of sterile water, which is within the one liter pressure cuff, and the compressor air line should be changed from the three liter cuff to the one liter cuff. The catheters from the six inlets of the six inlet valve and the catheter to the hypodermic needle should be placed into a container to collect the waste sterile water. When the shut down option is chosen, the sterile water is pushed through the needle catheter by opening three-way valve B. The six inlets of the six inlet valve are then flushed one at a time by opening three-way valve A and each of the six inlets of the six inlet valve in succession. The flushing of these seven catheters is repeated four times. All the valves are closed, the valves and lines should be free of saline and blood, and the PC, dual stepper motor drivers, reciprocating pump, and electronic housing may be turned off. Table 4 contains an outline of the complete sampling cycle.

Table 4 Outline of Complete Sampling Cycle

I. Follow hardware setup procedures

II. Prime the six inlet catheters

III. Choose one or three catheters per pig

IV. Modify sampling wait time delays if necessary

V. Enter sampling times

VI. Check sampling times, correct if necessary

VII. Insert catheters in pigs, leave unused catheters in saline container

VIII. Start Sampling

IX. Inject drug when appropriate

X. Wait for sampling cycle to complete

XI. Remove catheters from pigs and hypodermic needle and place in a waste container

XII. Connect sterile water in place of saline

XIII. Perform Shut Down procedure

RESULTS

The CCBSS is capable of taking samples from two pigs with two catheters each, as well as the two modes of operation that the system was designed for, one catheter per pig and three catheters per pig. Sampling from two pigs, using two catheters in each pig, can be accomplished in either one catheter per pig mode or three catheters per pig mode. The most efficient method, saving time and vacutainers, is to use the one catheter per pig mode. Use pig #1 and pig #2 on the user screen to set the sampling times for catheter #1 and catheter #2, respectively, for one pig. Then use pig #3 and pig #4 user screen sampling times to sample from a second pig, catheter #1 and catheter #2, respectively. In most situations, it would be desirable that for each pig #1 sampling time, a pig #2 sampling time is two minutes later, and that for each pig #3 sampling time, a pig #4 sampling time is two minutes later than that time. In this way, the two samples from a single pig would be taken so that the second sample is taken immediately after the first. If connected properly, blood samples taken using pig #1 sampling times (pig 1, catheter 1) will be stored in the vacutainers for pig #1. Pig 1, catheter 2 samples will be taken in pig #2 vacutainers, pig 2, catheter 1 samples in pig #3 vacutainers, and pig 2, catheter 2 samples in pig #4 vacutainers.

The system was first tested using a beaker of dog blood. The flushing and sampling periods were adjusted to allow enough time to perform the function without wasting excess time, saline, or blood. During this testing, it was discovered that some residual blood was left in the six inlet valve. Therefore, the additional flushing routine, described in the Operations section, was developed and implemented. After using this routine, no residual blood was seen leaving the six inlet valve during the next sample. Six small flasks were partially filled with dog blood heated to approximately 37° C to simulate the six different sources when three catheters per pig are used in two pigs. Confident that the system was operating properly, we proceeded to test the system using live animals.

A 70 pound greyhound was used to test the system; a catheter was placed in both femoral veins and both jugular veins to simulate four pigs with one catheter each. Sections of the procedure were video taped, and the complete procedure progressed relatively smoothly. A timing change was made to account for the dog being at the same height as the valves and the venous pressure. A Swans Ganz catheter was used to see if the system would operate correctly with a smaller lumen catheter. It was discovered that much more time is needed to draw and push blood through this smaller catheter. A new option was developed to allow the user to adjust the timing for each catheter depending on catheter size and blood vessel pressure.

The Show Wait Times option was added, which shows a table of wait times on the bottom of the user screen, allowing the user to modify the sampling procedure for each catheter. The user can change the amount of time that waste is pulled, the catheter flush time, and the pull and push times during the six inlet valve flush. A smaller catheter will require longer times for each of the parameters. The blood volume rate through a smaller catheter is lower and thus requires more time for the same amount of blood or saline to pass through it. If a catheter is placed in an artery, the two blood pull times should be decreased and the two push or flush times increased. The higher pressure of the artery makes the blood flow out of the animal much faster, thus requiring less time for the pull operations. But the increased pressure of the artery also makes it more difficult to push blood and saline back into the animal, thus requiring more time for blood push operations.

A second 70 pound greyhound was used to test the adjustable wait time delays. The time delays could be changed so that a Swans Ganz catheter would sample and flush correctly, but the time delays were approximately three times as long as the time delays for the catheters normally used. A few of the blood samples from the first greyhound were tested for hemolysis and seen to hemolysize. The hypodermic needle size was changed from

18 gauge to 16 gauge during the second greyhound testing to try to remedy this problem. The extra blood push into the vacutainer using the saline pressure was also removed to try to eliminate any hemolysis. Changing the size of the needle seemed to reduce the hemolysis to an acceptable level, so the extra blood push remains as part of the sampling procedure.

We discovered several operational concerns while testing the system that deserve to be mentioned. Some vacuum grease should be used when connecting the catheters and fittings, this will help seal the connections and provide an airtight system. Use a relatively large Erlenmeyer flask partially filled with saline to prime the system, this same container can then hold any unused catheters. All the catheters can be placed in the same container when sampling is finished and the shut down routine flushes the system with sterile water. The silicon septum that is used in the waste flask stopper assembly should be replaced every few sampling cycles. This will prevent any vacuum leaks due to the deterioration of the silicon septum from repeated penetration by the hypodermic needle. One concern that has risen is the possibility of power failure. In this situation, the valves will all be closed and the hypodermic needle will remain in its current position. It is not possible for the pig to bleed to death, but the catheters and valves may be clotted in the event of a power failure when a blood sample is being taken. A back-up power generator may help to solve this problem.

A user's manual was created for the CCBSS to be given to Merck with the system. A copy of the CCBSS User's Manual is in Appendix B. The CCBSS User's Manual provides step-by-step instructions to set up and connect the various hardware components. The CCBSS User's Manual also contains a section on using the application which explains how and when the application's menu options can be used. An outline of the complete sampling procedure is also included in the CCBSS User's Manual, which can be used for quick reference or as a check list to make sure procedures are followed correctly. Some suggestions are provided to get the best operational results from the system.

CONCLUSION

An automatic blood sampling system was developed to take samples from pigs without human intervention after initial set up. The CCBSS was designed for two modes of operation, three catheters per pig, two pigs maximum, and one catheter per pig, four pigs maximum. An IBM compatible PC was used to provide a simple user interface capable of acquiring sampling times. When the user starts the sampling routine, the PC controls the opening and closing of the solenoid valves to control blood flow. The stepper motors, thus the position of the hypodermic needle, are also controlled by the PC during sampling.

The CCBSS was adjusted to operate correctly when tested using small flasks of dog blood and two greyhounds. A user's manual was written for the CCBSS for the researchers at Merck to reference and use as necessary in the future. Some modifications that could be made to the CCBSS if future versions of the system are developed are listed below. An XY table could be constructed so that the shafts are more precisely parallel using linear bearings to reduce friction. A vacutainer rack with twelve columns of holes may help the researcher separate and remember which vacutainers contain samples from which pig and catheter. With twelve columns, in one catheter per pig mode, all samples from a single pig would be located in a single row of vacutainers. In three catheters per pig mode, three rows of vacutainers for each pig would be necessary and samples from catheter one and four would always be located in columns 1, 4, 7, or 10. Valves with zero dead space could be used in place of the six-inlet valve, such as the new pinch valves, or a new valve configuration could be designed using only three-way valves. Use of a power supply designed for other purposes, such as the computer power supply used in this version of the CCBSS, should be avoided to obtain the expected voltages.

REFERENCES

[1] B.M. Hill, R.H. Laessig, D.D. Koch and D.J. Hassemer, Comparison of plastic vs. glass evacuated serum separator (SST) blood-drawing tubes for common clinical chemistry determinations. Clin. Chem. 38 (1992) 1474-1478

[2] P. Scheid and H. Slama, Remote-controlled device for sampling blood in unrestrained animals. Pflugers Arch. 356 (1975) 373-376

[3] R.D. Hill, Microcomputer monitor and blood sampler for free diving Weddell seals. J. Appl. Physiol. 61 (1986) 1570-1576

[4] R.T. Schmalzried, P.W. Toll, J.J. Devore and M.R. Fedde, Microcontroller-based system for collecting anaerobic blood samples from a running greyhound. Comput. Methods Programs Biomed. 37 (1992) 183-190

[5] C.G. Veinott and J.E. Martin, Stepper Motors. Fractional and subfractional horsepower electric motors. Fourth Edition (1986) 302-316 McGraw-Hill, New York

[6] MD-2 Dual Stepper Motor System User's Guide. Revision B (1991) Arrick Robotics, Hurst, TX

[7] Visual Basic Programmer's Guide. (1992) Microsoft Corporation, Redmond, WA

APPENDIX A

CCBSS PROGRAM LISTING

```
DECLARE SUB valve_flush ()
DECLARE SUB mnu_save_click ()
DECLARE SUB FileSave (FileName AS STRING, PathName AS STRING, DefaultExt AS
STRING, DialogTitle AS STRING, ForeColor AS INTEGER, BackColor AS INTEGER, Flags AS
INTEGER, cancel AS INTEGER)
DECLARE SUB FileOpen (FileName AS STRING, PathName AS STRING, DefaultExt AS
STRING, DialogTitle AS STRING, ForeColor AS INTEGER, BackColor AS INTEGER, Flags AS
INTEGER, cancel AS INTEGER)
DECLARE SUB chk_time (min_delay AS INTEGER, ok AS INTEGER)
DECLARE SUB timer1_timer ()
DECLARE SUB md2_init ()
DECLARE SUB take_sample (tube AS INTEGER, valve AS INTEGER)
DECLARE SUB four_hog ()
DECLARE SUB two_hog ()
DECLARE SUB CIODIO (MD AS INTEGER, BYVAL ARR AS INTEGER, F AS INTEGER)

  DIM D%(10)
  COMMON SHARED D%()

  DIM SHARED cancel AS INTEGER
  DIM SHARED FileName AS STRING
  DIM SHARED PathName AS STRING
  DIM SHARED BACKLASH%(6)
  DIM SHARED CHECK.KEY$(6)
  DIM SHARED CHECK.SWITCH$(6)
  DIM SHARED CURRENT.POSITION&(6)
  DIM SHARED DIRECTION$(6)
  DIM SHARED MD2.LAST.DIRECTION$(6)
  DIM SHARED MD2.LAST.PATTERN%(6)
  DIM SHARED MD2.PATTERN%(8)
  DIM SHARED MD2.NEW.PATTERN%(8)
  DIM SHARED POWER.DOWN$(6)
  DIM SHARED SPEED%(6)
  DIM SHARED STEPS.TO.MOVE&(6)
  DIM SHARED step.type$(6)
  DIM SHARED SWITCH$(6)
  DIM SHARED TARGET.POSITION&(6)
```

```
SUB chk_time (min_delay AS INTEGER, ok AS INTEGER)

' Procedure verifys that all times entered in the sampling
' times squares are at least min_delay minutes apart from
' every other time.  ok is passed back to the calling
' procedure to indicate the result of the checking routine.
' (ok=1 times approved, ok=0 times must be changed before
' sampling can begin.)

   FOR i = 0 TO 11
      FOR j = 0 TO 11
         IF i < j THEN IF ((VAL(text1(i).text) + min_delay) > VAL(text1(j).text) AND
VAL(text1(i).text) <> 0 AND VAL(text1(j).text) <> 0) OR (VAL(text1(i).text) = 0 AND
VAL(text1(i + 1).text) <> 0) THEN response% = MSGBOX("The current times do not meet system
requirements.  Sampling cannot begin until times are modified.  Please examine pig #1, columns " +
STR$(i + 1) + " and " + STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
         IF ((VAL(text1(i).text) + min_delay) > VAL(text2(j).text) AND (VAL(text1(i).text) -
min_delay) < VAL(text2(j).text)) AND VAL(text1(i).text) <> 0 AND VAL(text2(j).text) <> 0 THEN
response% = MSGBOX("The current times do not meet system requirements.  Sampling cannot begin
until times are modified.  Please examine pig #1 and pig #2, columns " + STR$(i + 1) + " and " +
STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
         IF ((VAL(text1(i).text) + min_delay) > VAL(text3(j).text) AND (VAL(text1(i).text) -
min_delay) < VAL(text3(j).text)) AND VAL(text1(i).text) <> 0 AND VAL(text3(j).text) <> 0 THEN
response% = MSGBOX("The current times do not meet system requirements.  Sampling cannot begin
until times are modified.  Please examine pig #1 and pig #3, columns " + STR$(i + 1) + " and " +
STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
         IF ((VAL(text1(i).text) + min_delay) > VAL(text4(j).text) AND (VAL(text1(i).text) -
min_delay) < VAL(text4(j).text)) AND VAL(text1(i).text) <> 0 AND VAL(text4(j).text) <> 0 THEN
response% = MSGBOX("The current times do not meet system requirements.  Sampling cannot begin
until times are modified.  Please examine pig #1 and pig #4, columns " + STR$(i + 1) + " and " +
STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
         IF i < j THEN IF ((VAL(text2(i).text) + min_delay) > VAL(text2(j).text) AND
VAL(text2(i).text) <> 0 AND VAL(text2(j).text) <> 0) OR (VAL(text2(i).text) = 0 AND
VAL(text2(i + 1).text) <> 0) THEN response% = MSGBOX("The current times do not meet system
requirements.  Sampling cannot begin until times are modified.  Please examine pig #2, columns " +
STR$(i + 1) + " and " + STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
         IF ((VAL(text2(i).text) + min_delay) > VAL(text3(j).text) AND (VAL(text2(i).text) -
min_delay) < VAL(text3(j).text)) AND VAL(text2(i).text) <> 0 AND VAL(text3(j).text) <> 0 THEN
response% = MSGBOX("The current times do not meet system requirements.  Sampling cannot begin
until times are modified.  Please examine pig #2 and pig #3, columns " + STR$(i + 1) + " and " +
STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
         IF ((VAL(text2(i).text) + min_delay) > VAL(text4(j).text) AND (VAL(text2(i).text) -
min_delay) < VAL(text4(j).text)) AND VAL(text2(i).text) <> 0 AND VAL(text4(j).text) <> 0 THEN
response% = MSGBOX("The current times do not meet system requirements.  Sampling cannot begin
```

```
until times are modified.  Please examine pig #2 and pig #4, columns " + STR$(i + 1) + " and " +
STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
        IF i < j THEN IF ((VAL(text3(i).text) + min_delay) > VAL(text3(j).text) AND
VAL(text3(i).text) <> 0 AND VAL(text3(j).text) <> 0) OR (VAL(text3(i).text) = 0 AND
VAL(text3(i + 1).text) <> 0) THEN response% = MSGBOX("The current times do not meet system
requirements.  Sampling cannot begin until times are modified.  Please examine pig #3, columns " +
STR$(i + 1) + " and " + STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
        IF ((VAL(text3(i).text) + min_delay) > VAL(text4(j).text) AND (VAL(text3(i).text) -
min_delay) < VAL(text4(j).text)) AND VAL(text3(i).text) <> 0 AND VAL(text4(j).text) <> 0 THEN
response% = MSGBOX("The current times do not meet system requirements.  Sampling cannot begin
until times are modified.  Please examine pig #3 and pig #4, columns " + STR$(i + 1) + " and " +
STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
        IF i < j THEN IF ((VAL(text4(i).text) + min_delay) > VAL(text4(j).text) AND
VAL(text4(i).text) <> 0 AND VAL(text4(j).text) <> 0) OR (VAL(text4(i).text) = 0 AND
VAL(text4(i + 1).text) <> 0) THEN response% = MSGBOX("The current times do not meet system
requirements.  Sampling cannot begin until times are modified.  Please examine pig #4, columns " +
STR$(i + 1) + " and " + STR$(j + 1), 0, "Warning"): ok = 0: IF response% = 1 THEN GOTO done
      NEXT j
   NEXT i
   response% = MSGBOX("The current times meet system requirements.", 1, "Approved")
   IF response% = 1 THEN ok = 1: GOTO done
done:

END SUB
```

```
SUB command1_click ()

' Switch between one catheter per pig and three catheters per
' pig modes.  Change button label and display appropriate
' pig time squares.

   label3.visible = NOT label3.visible
   label4.visible = NOT label4.visible
   FOR i = 0 TO 11
      text3(i).visible = NOT text3(i).visible
      text4(i).visible = NOT text4(i).visible
   NEXT i
   IF label3.visible = -1 THEN command1.caption = "2 Pigs w/ 3 catheters" ELSE command1.caption
= "4 Pigs w/ 1 catheter"
END SUB
```

```
SUB form_load ()

' Set colors, background pattern.  Place dot next to Show
' Clock menu option.  Initialize the I/O board, turn all
' relays off (close all valves).

   screen.controlpanel(4) = 4
   screen.controlpanel(9) = 9
   screen.controlpanel(16) = 12
   screen.controlpanel(7) = 206
   screen.controlpanel(5) = 0
   mnu_clock.checked = -1

  F% = 0                  'declare error flag as '0'
  MD% = 0                  'set MODE
  D%(0) = &H380              'I/O ADDRESS OF DIO XX
  D%(1) = 2
  D%(2) = 1
  CALL CIODIO(MD%, VARPTR(D%(0)), F%)

  MD% = 55
  D%(0) = 0              'PORT A OUTPUT
  D%(1) = 0              'PORT B OUTPUT
  D%(2) = 0              'PORT C LOWER AS OUTPUT
  D%(3) = 0              'PORT C UPPER AS OUTPUT
  D%(4) = 1              'FIRST 8255 AT BASE + 0 -> BASE + 3
  CALL CIODIO(MD%, VARPTR(D%(0)), F%)

   MD% = 58
   D%(0) = 1
   D%(1) = 255
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

   mnu_mod_wait.checked = 0
   label8.visible = false
   label9.visible = false
   label10.visible = false
   label11.visible = false
   label12.visible = false
   FOR i = 0 TO 5
      waste(i).visible = false
      flush(i).visible = false
      pull(i).visible = false
      push(i).visible = false
```

```
    NEXT i

    FOR i = 0 TO 5
        waste(i).text = "27"
        flush(i).text = "1.5"
        pull(i).text = "2.7"
        push(i).text = "3"
    NEXT i

END SUB
```

```
SUB four_hog ()

' Used when sampling is started in one catheter per pig mode.
' Determines which sample is to be taken next and waits for the
' appropriate time to occur before calling the take_sample
' procedure.

   DIM dstart AS STRING
   current1 = 0: current2 = 0: current3 = 0: current4 = 0
   tstart = VAL(MID$(TIME$, 1, 2)) * 60 + VAL(MID$(TIME$, 4, 2)) + VAL(MID$(TIME$, 7,
2)) / 60
   dstart = DATE$

   IF VAL(text1(current1).text) = 0 THEN text1(current1).text = "9999": text1(current1).visible = 0
   IF VAL(text2(current2).text) = 0 THEN text2(current2).text = "9999": text2(current2).visible = 0
   IF VAL(text3(current3).text) = 0 THEN text3(current3).text = "9999": text3(current3).visible = 0
   IF VAL(text4(current4).text) = 0 THEN text4(current4).text = "9999": text4(current4).visible = 0

  DO WHILE VAL(text1(current1).text) <> 9999 OR VAL(text2(current2).text) <> 9999 OR
VAL(text3(current3).text) <> 9999 OR VAL(text4(current4).text) <> 9999
   IF VAL(text1(current1).text) < VAL(text2(current2).text) AND VAL(text1(current1).text) <
VAL(text3(current3).text) AND VAL(text1(current1).text) < VAL(text4(current4).text) THEN
      waittime = VAL(text1(current1).text): tube% = current1 + 1: valve% = 0: current1 = current1 +
1: IF VAL(text1(current1).text) = 0 THEN text1(current1).text = "9999": text1(current1).visible = 0
   ELSEIF VAL(text2(current2).text) < VAL(text1(current1).text) AND VAL(text2(current2).text) <
VAL(text3(current3).text) AND VAL(text2(current2).text) < VAL(text4(current4).text) THEN
      waittime = VAL(text2(current2).text): tube% = current2 + 21: valve% = 1: current2 = current2
+ 1: IF VAL(text2(current2).text) = 0 THEN text2(current2).text = "9999": text2(current2).visible =
0
   ELSEIF VAL(text3(current3).text) < VAL(text1(current1).text) AND VAL(text3(current3).text) <
VAL(text2(current2).text) AND VAL(text3(current3).text) < VAL(text4(current4).text) THEN
      waittime = VAL(text3(current3).text): tube% = current3 + 41: valve% = 2: current3 = current3
+ 1: IF VAL(text3(current3).text) = 0 THEN text3(current3).text = "9999": text3(current3).visible =
0
   ELSEIF VAL(text4(current4).text) < VAL(text1(current1).text) AND VAL(text4(current4).text) <
VAL(text2(current2).text) AND VAL(text4(current4).text) < VAL(text3(current3).text) THEN
      waittime = VAL(text4(current4).text): tube% = current4 + 61: valve% = 3: current4 = current4
+ 1: IF VAL(text4(current4).text) = 0 THEN text4(current4).text = "9999": text4(current4).visible =
0
   END IF

   DO
      IF DATE$ <> dstart THEN tstart = tstart - 1440: dstart = DATE$
      CALL timer1_timer
```

```
   LOOP WHILE VAL(MID$(TIME$, 1, 2)) * 60 + VAL(MID$(TIME$, 4, 2)) +
VAL(MID$(TIME$, 7, 2)) / 60 < tstart + waittime

   screen.HIDE

   CALL take_sample(tube%, valve%)
   screen.SHOW
  LOOP
  FOR i = 0 TO 11
     IF text1(i).visible = 0 THEN text1(i).text = "": text1(i).visible = -1
     IF text2(i).visible = 0 THEN text2(i).text = "": text2(i).visible = -1
     IF text3(i).visible = 0 THEN text3(i).text = "": text3(i).visible = -1
     IF text4(i).visible = 0 THEN text4(i).text = "": text4(i).visible = -1
  NEXT i
END SUB
```

```
SUB mnu_check_times_click ()

' Initiates procedure to check times with appropriate minimum
' time between samples.

   DIM ok AS INTEGER
   IF label3.visible = -1 THEN CALL chk_time(2, ok) ELSE CALL chk_time(6, ok)
END SUB
```

```
SUB mnu_clock_click ()

' Display / Hide clock

   label6.visible = NOT label6.visible
   mnu_clock.checked = NOT mnu_clock.checked
END SUB
```

```
SUB mnu_exit_click ()

' Have user verify that they want to exit the application and
' give them the opportunity to save the current sampling times.

   response% = MSGBOX("Exiting Application.  Unsaved data will be lost.  Exit without saving
data.", 515, "Warning")
   IF response% = 6 THEN END
   IF response% = 7 THEN CALL mnu_save_click: IF NOT cancel THEN END
END SUB
```

```
SUB mnu_mod_wait_click ()

   mnu_mod_wait.checked = -1
   label8.visible = NOT label8.visible
   label9.visible = NOT label9.visible
   label10.visible = NOT label10.visible
   label11.visible = NOT label11.visible
   label12.visible = NOT label12.visible
   FOR i = 0 TO 5
      waste(i).visible = NOT waste(i).visible
      flush(i).visible = NOT flush(i).visible
      pull(i).visible = NOT pull(i).visible
      push(i).visible = NOT push(i).visible
   NEXT i

END SUB
```

```
SUB mnu_new_click ()

' Clear the current sampling times after the user has
' verified this choice.

   FOR i = 0 TO 11
      text1(i).text = ""
      text2(i).text = ""
      text3(i).text = ""
      text4(i).text = ""
   NEXT i
END SUB
```

```
SUB mnu_open_click ()

' Retrieve sampling times from a data file chosen by the
' user.  Then display these times in the squares on the
' user interface screen.

   DIM Flags AS INTEGER
   DIM DefaultExt AS STRING
   DIM DialogTitle AS STRING
   DialogTitle = "Open Data File"
   DefaultExt = "*.DAT"
   CALL FileOpen(FileName, PathName, DefaultExt, DialogTitle, 3, 0, Flags, cancel)
   IF NOT cancel THEN
      OPEN PathName + "\" + FileName FOR INPUT AS 1
      FOR i = 0 TO 11
         INPUT #1, temp$
         text1(i).text = temp$
         INPUT #1, temp$
         text2(i).text = temp$
         INPUT #1, temp$
         text3(i).text = temp$
         INPUT #1, temp$
         text4(i).text = temp$
      NEXT i
      CLOSE #1
   END IF
END SUB
```

```
SUB mnu_prime_click ()

' Controls solenoid valves to prime the system.  Each of
' the six inlet catheters connected to the six inlet valve
' are flushed in order (1 through 6).  This removes air
' from the lines and fills them with saline.

   response% = MSGBOX("Prime will push saline out all six inlets of the six inlet valve to remove
any air and fill the catheters with saline. Collect excess saline. The Prime routine should be performed
before the catheters are placed in the pigs.", 257, "Warning")
   IF response% = 2 THEN GOTO fini

   FOR valve = 0 TO 5
   MD% = 58
   D%(0) = 1
   D%(1) = 191 - 2 ^ valve
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)
   FOR i = 1 TO 20000: NEXT i
   NEXT valve

   MD% = 58
   D%(0) = 1
   D%(1) = 255
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)
fini:
END SUB
```

```
SUB mnu_save_click ()

' Save the current sampling times in a data file named
' by the user.

   DIM Flags AS INTEGER
   DIM DefaultExt AS STRING
   DIM DialogTitle AS STRING
   DialogTitle = "Save Data File"
   DefaultExt = "*.DAT"
   CALL FileSave(FileName, PathName, DefaultExt, DialogTitle, 3, 0, Flags, cancel)
   IF NOT cancel THEN
      OPEN PathName + "\" + FileName FOR OUTPUT AS 1
      FOR i = 0 TO 11
         WRITE #1, text1(i).text
         WRITE #1, text2(i).text
         WRITE #1, text3(i).text
         WRITE #1, text4(i).text
      NEXT i
      CLOSE #1
   END IF
END SUB
```

```
SUB mnu_shut_click ()

' Controls solenoid valves to flush the system.  The needle
' catheter is flushed first and then each of the six inlet
' catheters connected to the six inlet valve are flushed
' in order (1 through 6).  This process is repeated 4 times.

   response% = MSGBOX("Make sure deionized water is connected (instead of saline) and catheters
are in a waste container.  All six inlets of the six inlet valve and the catheter connected to the needle
will be flushed.", 257, "Warning")
   IF response% = 2 THEN GOTO canc

  FOR j = 1 TO 4
   MD% = 58
   D%(0) = 1
   D%(1) = 127
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)
   FOR i = 1 TO 70000: NEXT i

   FOR valve = 0 TO 5
   MD% = 58
   D%(0) = 1
   D%(1) = 191 - 2 ^ valve
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)
   FOR i = 1 TO 70000: NEXT i
   NEXT valve

   MD% = 58
   D%(0) = 1
   D%(1) = 255
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)
  NEXT j

canc:
END SUB
```

```
SUB mnu_start_click ()

' Call chk_time to verify that sampling times meet system
' requirements.  If so, print the current time (sampling
' intiated) to the user screen and call the appropriate
' procedure (depending on the mode of operation chosen).

   DIM temp_tim AS STRING
   DIM hours AS INTEGER
   DIM ok AS INTEGER

   IF label3.visible = -1 THEN CALL chk_time(2, ok) ELSE CALL chk_time(6, ok)
   IF ok = 1 THEN
      temp_tim = TIME$
      hours = VAL(temp_tim)
      IF hours > 12 THEN MID$(temp_tim, 1, 2) = STR$(hours - 12)

      label7.caption = "Sampling initiated: " + temp_tim
      label7.visible = -1
      IF label3.visible = -1 THEN CALL four_hog ELSE CALL two_hog
   END IF
   label7.visible = 0
END SUB
```

```
SUB take_sample (tube AS INTEGER, valve AS INTEGER)

' Controls the three stepper motors and the solenoid
' valves to take the appropriate blood sample in the
' proper vacutainer.  Procedure contains the routines
' provided by Arrick Robotics to control the stepper
' motors.

'Initialize ports
  IF init = 1 THEN GOTO start
    FOR MOTOR% = 3 TO 5
       GOSUB md2.init
    NEXT MOTOR%
    init = 1

' Move all motors to the home position.  Z axis first.
start:
    move.action$ = "P"
    MOTOR% = 4
    SPEED%(MOTOR%) = 300
    POWER.DOWN$(MOTOR%) = "Y"
    CHECK.KEY$(MOTOR%) = "N"
    GOSUB md2.home
    MOTOR% = 3
    SPEED%(MOTOR%) = 400
    POWER.DOWN$(MOTOR%) = "Y"
    CHECK.KEY$(MOTOR%) = "N"
    GOSUB md2.home
    MOTOR% = 5
    SPEED%(MOTOR%) = 300
    POWER.DOWN$(MOTOR%) = "N"
    CHECK.KEY$(MOTOR%) = "N"
    GOSUB md2.home
    MOTOR% = 5
    TARGET.POSITION&(MOTOR%) = 1
    POWER.DOWN$(MOTOR%) = "Y"
    GOSUB md2.move

' Drive needle into waste flask.
    MOTOR% = 4
    TARGET.POSITION&(MOTOR%) = 1500
    GOSUB md2.move

' Open both three-way valves and the appropriate inlet
' of the six inlet valve.
```

```
    MD% = 58
    D%(0) = 1
    D%(1) = 63 - 2 ^ valve
    D%(2) = 1
    CALL CIODIO(MD%, VARPTR(D%(0)), F%)

' Wait to pull blood through the lines and valves.
    begin = VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2))
    DO
    LOOP WHILE VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2)) < begin +
VAL(waste(valve).text)

' Close open inlet of the six inlet valve.
    MD% = 58
    D%(0) = 1
    D%(1) = 63
    D%(2) = 1
    CALL CIODIO(MD%, VARPTR(D%(0)), F%)

' Remove needle from waste flask.
    MOTOR% = 4
    GOSUB md2.home

' Move needle to a position above the proper vacutainer.
    MOTOR% = 3
    TARGET.POSITION&(MOTOR%) = ((tube - 1) MOD 10) * 170
    GOSUB md2.move
    MOTOR% = 5
    TARGET.POSITION&(MOTOR%) = 1070 + ((tube - 1) \ 10) * 170
    POWER.DOWN$(MOTOR%) = "Y"
    GOSUB md2.move

' Drive needle into the proper vacutainer.
    MOTOR% = 4
    TARGET.POSITION&(MOTOR%) = 1500
    GOSUB md2.move

' Reopen the previously opened and then closed inlet of
' the six inlet valve.
    MD% = 58
    D%(0) = 1
    D%(1) = 63 - 2 ^ valve
    D%(2) = 1
    CALL CIODIO(MD%, VARPTR(D%(0)), F%)
```

```
' Wait to fill the vacutainer with blood.
   FOR i = 1 TO 140000: NEXT i

' Close three-way valve A
   MD% = 58
   D%(0) = 1
   D%(1) = 127 - 2 ^ valve
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

' Wait to fill the vacutainer using a push of saline.
   FOR i = 1 TO 40000: NEXT i

' Open three-way valve A and close three-way valve B to
' flush blood and saline back into the pig.
   MD% = 58
   D%(0) = 1
   D%(1) = 191 - 2 ^ valve
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

' Remove needle from the vacutainer.
   MOTOR% = 4
   GOSUB md2.home

' Move the needle to the home position, above the waste
' flask.
   MOTOR% = 3
   GOSUB md2.home
   MOTOR% = 5
   POWER.DOWN$(MOTOR%) = "N"
   GOSUB md2.home
   MOTOR% = 5
   TARGET.POSITION&(MOTOR%) = 1
   POWER.DOWN$(MOTOR%) = "Y"
   GOSUB md2.move

' Drive the needle into the waste flask.
   MOTOR% = 4
   TARGET.POSITION&(MOTOR%) = 1500
   GOSUB md2.move

' Wait to flush the lines on the pig side of the valves.
   begin = VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2))
   DO
```

```
   LOOP WHILE VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2)) < begin +
VAL(flush(valve).text)

' Close three-way valve A, close all inlets of the six
' inlet valve and open three-way valve B.
   MD% = 58
   D%(0) = 1
   D%(1) = 127
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

' Wait to flush the lines on the needle side of the valves.
   FOR i = 1 TO 70000: NEXT i

' Close all valves.
   MD% = 58
   D%(0) = 1
   D%(1) = 255
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

   CALL valve_flush

' Remove needle from waste flask.
   MOTOR% = 4
   GOSUB md2.home

   GOTO finish

'**************** MD-2 INITIALIZATION *****************
'
'DESCRIPTION:
'  USE AT BEGINNING OF PROGRAM TO INITIALIZE VARIABLES AND PORTS.
'  PORTS ARE NOT INITIALIZED IF MOTOR%=0.
'
'VARIABLES RECEIVED:
'  MOTOR% = 1 - 6, 12, 34 OR 56.
'
'VARIABLES RETURNED:
'  ALL LISTED BELOW WITH DEFAULT VALUES.
'
md2.init:

   'SET PORT ADDRESSES.
   MD2.ADDR.12% = &H3BC     'MOTOR 1 & 2
```

```
  MD2.ADDR.34% = &H378    'MOTOR 3 & 4
  MD2.ADDR.56% = &H278    'MOTOR 5 & 6

  'FIND PORTS.
  OUT MD2.ADDR.12%, &HAA
  OUT MD2.ADDR.34%, &HAA
  OUT MD2.ADDR.56%, &HAA
  IF INP(MD2.ADDR.12%) = &HAA THEN PORT.12$ = "Y" ELSE PORT.12$ = "N"
  IF INP(MD2.ADDR.34%) = &HAA THEN PORT.34$ = "Y" ELSE PORT.34$ = "N"
  IF INP(MD2.ADDR.56%) = &HAA THEN PORT.56$ = "Y" ELSE PORT.56$ = "N"

  'TURN OFF ALL PHASES OF ALL MOTORS.
  OUT MD2.ADDR.12%, &HFF
  OUT MD2.ADDR.34%, &HFF
  OUT MD2.ADDR.56%, &HFF

  'SET PARAMETER DEFAULTS.
  move.action$ = "P"
  MD2.BACKLASH$ = "Y"
  FOR MD2.I% = 0 TO 6
    BACKLASH%(MD2.I%) = 0
    CHECK.KEY$(MD2.I%) = "Y"
    CHECK.SWITCH$(MD2.I%) = "N"
    CURRENT.POSITION&(MD2.I%) = 0
    DIRECTION$(MD2.I%) = "F"
    MD2.LAST.DIRECTION$(MD2.I%) = "F"
    MD2.LAST.PATTERN%(MD2.I%) = &HE
    POWER.DOWN$(MD2.I%) = "N"
    SPEED%(MD2.I%) = 1000
    STEPS.TO.MOVE&(MD2.I%) = 0
    step.type$(MD2.I%) = "H"
    SWITCH$(MD2.I%) = "N"
    TARGET.POSITION&(MD2.I%) = 0
  NEXT MD2.I%

  'TURN ON MOTOR DRIVER AND SET PHASES OF MOTOR% TO LAST PATTERN.
  'IF ONLY ONE MOTOR SELECTED THEN THE OTHER MOTOR AT THAT PORT IS LEFT
  'OFF. THIS ROUTINE CAN BE USED SEPERATE FROM MD2.INIT WHEN VARIABLES
  'MUST BE INITIALIZED BUT MOTORS TO BE USED ARE NOT KNOWN.
MD2.ON:
  IF MOTOR% = 1 THEN
    OUT MD2.ADDR.12%, ((INP(MD2.ADDR.12%) AND &HF0) OR
MD2.LAST.PATTERN%(1))
    OUT MD2.ADDR.12% + 2, &HED
  END IF
```

```
   IF MOTOR% = 2 THEN
      OUT MD2.ADDR.12%, ((INP(MD2.ADDR.12%) AND &HF) OR
(MD2.LAST.PATTERN%(2) * &H10))
      OUT MD2.ADDR.12% + 2, &HED
   END IF
   IF MOTOR% = 3 THEN
      OUT MD2.ADDR.34%, ((INP(MD2.ADDR.34%) AND &HF0) OR
MD2.LAST.PATTERN%(3))
      OUT MD2.ADDR.34% + 2, &HED
   END IF
   IF MOTOR% = 4 THEN
      OUT MD2.ADDR.34%, ((INP(MD2.ADDR.34%) AND &HF) OR
(MD2.LAST.PATTERN%(4) * &H10))
      OUT MD2.ADDR.34% + 2, &HED
   END IF
   IF MOTOR% = 5 THEN
      OUT MD2.ADDR.56%, ((INP(MD2.ADDR.56%) AND &HF0) OR
MD2.LAST.PATTERN%(5))
      OUT MD2.ADDR.56% + 2, &HED
   END IF
   IF MOTOR% = 6 THEN
      OUT MD2.ADDR.56%, ((INP(MD2.ADDR.56%) AND &HF) OR
(MD2.LAST.PATTERN%(6) * &H10))
      OUT MD2.ADDR.56% + 2, &HED
   END IF
   IF MOTOR% = 12 THEN
      OUT MD2.ADDR.12%, ((MD2.LAST.PATTERN%(2) * &H10) OR
MD2.LAST.PATTERN%(1))
      OUT MD2.ADDR.12% + 2, &HED
   END IF
   IF MOTOR% = 34 THEN
      OUT MD2.ADDR.34%, ((MD2.LAST.PATTERN%(4) * &H10) OR
MD2.LAST.PATTERN%(3))
      OUT MD2.ADDR.34% + 2, &HED
   END IF
   IF MOTOR% = 56 THEN
      OUT MD2.ADDR.56%, ((MD2.LAST.PATTERN%(6) * &H10) OR
MD2.LAST.PATTERN%(5))
      OUT MD2.ADDR.56% + 2, &HED
   END IF

'************** READ MD-2 SWITCHES *************
'
'DESCRIPTION:
'  RETURNS SWITCH STATUS IN SWITCH$(MOTOR%) VARIABLES.
```

```
'
'VARIABLES RECEIVED:
'  NONE
'
'VARIABLES RETURNED:
'  SWITCH$(MOTOR%)
'
MD2.SWITCH:
   IF (INP(MD2.ADDR.12% + 1) AND &H20) = 0 THEN
      SWITCH$(1) = "Y"
   ELSE
      SWITCH$(1) = "N"
   END IF

   IF (INP(MD2.ADDR.12% + 1) AND &H10) = 0 THEN
      SWITCH$(2) = "Y"
   ELSE
      SWITCH$(2) = "N"
   END IF

   IF (INP(MD2.ADDR.34% + 1) AND &H20) = 0 THEN
      SWITCH$(3) = "Y"
   ELSE
      SWITCH$(3) = "N"
   END IF

   IF (INP(MD2.ADDR.34% + 1) AND &H10) = 0 THEN
      SWITCH$(4) = "Y"
   ELSE
      SWITCH$(4) = "N"
   END IF

   IF (INP(MD2.ADDR.56% + 1) AND &H20) = 0 THEN
      SWITCH$(5) = "Y"
   ELSE
      SWITCH$(5) = "N"
   END IF

   IF (INP(MD2.ADDR.56% + 1) AND &H10) = 0 THEN
      SWITCH$(6) = "Y"
   ELSE
      SWITCH$(6) = "N"
   END IF

   RETURN
```

```
'************* RESET PARALLEL PRINTER PORT ***************
'
'DESCRIPTION:
'  RETURN THE PARALLEL PRINTER PORT TO ITS PREVIOUS STATUS.
'
'VARIABLES RECEIVED:
'  MOTOR% = 1 - 6, 12, 34 OR 56.
'
'VARIABLES RETURNED:
'  NONE
'
MD2.RESET:
   'STB PIN OFF(HIGH), ALF PIN OFF(HIGH),
   'SEL PIN ON(LOW), IRQ DISABLED,
   'INIT PIN ON(LOW).
   IF (MOTOR% = 1) OR (MOTOR% = 2) OR (MOTOR% = 12) THEN
      OUT MD2.ADDR.12% + 2, &H8: OUT MD2.ADDR.12%, &HFF
   END IF

   IF (MOTOR% = 3) OR (MOTOR% = 4) OR (MOTOR% = 34) THEN
      OUT MD2.ADDR.34% + 2, &H8: OUT MD2.ADDR.34%, &HFF
   END IF

   IF (MOTOR% = 5) OR (MOTOR% = 6) OR (MOTOR% = 56) THEN
      OUT MD2.ADDR.56% + 2, &H8: OUT MD2.ADDR.56%, &HFF
   END IF

   'DELAY.
   MD2.I = TIMER: DO: LOOP UNTIL TIMER > MD2.I + .2

   'TURN INIT PIN OFF(HIGH).
   IF (MOTOR% = 1) OR (MOTOR% = 2) OR (MOTOR% = 12) THEN
      OUT MD2.ADDR.12% + 2, &HC
   END IF

   IF (MOTOR% = 3) OR (MOTOR% = 4) OR (MOTOR% = 34) THEN
      OUT MD2.ADDR.34% + 2, &HC
   END IF

   IF (MOTOR% = 5) OR (MOTOR% = 6) OR (MOTOR% = 56) THEN
      OUT MD2.ADDR.56% + 2, &HC
   END IF
```

```
   RETURN


'********************** MD-2 HOME ***********************
'
'DESCRIPTION:
'  PERFORMS THE FOLLOWING HOME POSITIONING SEQUENCE:
'     1. MOVE SELECTED MOTOR(S) REVERSE UNTIL THE SWITCH IS ACTIVATED.
'     2. MOVE FORWARD UNTIL THE SWITCH IS DEACTIVATED.
'     3. SET CURRENT POSITION VARIABLE(S) TO 0.
'
'VARIABLES RECEIVED:
'  MOTOR% = 1 - 6, 12, 34 OR 56.
'  ALL MOTOR PARAMETERS.
'
'VARIABLES RETURNED:
'  RETURN.STATUS$          O=OK, K=STOPED BY KEYPRESS.
'  CURRENT.POSITION&(MOTOR%)  SET TO ZERO. INVALID WHEN MOVEMENT
'                    STOPPED BY KEYPRESS.
'
md2.home:

   'SET MOTOR NUMBERS AND DO HOME FUNCTION.
   IF MOTOR% = 1 THEN
      MD2.HOME.FIRST.MOTOR% = 1: MD2.HOME.SECOND.MOTOR% = 0
      GOSUB MD2.HOME.WORK
      RETURN
   END IF

   IF MOTOR% = 2 THEN
      MD2.HOME.FIRST.MOTOR% = 2: MD2.HOME.SECOND.MOTOR% = 0
      GOSUB MD2.HOME.WORK
      RETURN
   END IF

   IF MOTOR% = 3 THEN
      MD2.HOME.FIRST.MOTOR% = 3: MD2.HOME.SECOND.MOTOR% = 0
      GOSUB MD2.HOME.WORK
      RETURN
   END IF

   IF MOTOR% = 4 THEN
      MD2.HOME.FIRST.MOTOR% = 4: MD2.HOME.SECOND.MOTOR% = 0
      GOSUB MD2.HOME.WORK
      RETURN
```

```
  END IF

  IF MOTOR% = 5 THEN
    MD2.HOME.FIRST.MOTOR% = 5: MD2.HOME.SECOND.MOTOR% = 0
    GOSUB MD2.HOME.WORK
    RETURN
  END IF

  IF MOTOR% = 6 THEN
    MD2.HOME.FIRST.MOTOR% = 6: MD2.HOME.SECOND.MOTOR% = 0
    GOSUB MD2.HOME.WORK
    RETURN
  END IF

  IF MOTOR% = 12 THEN
    MD2.HOME.FIRST.MOTOR% = 1: MD2.HOME.SECOND.MOTOR% = 2
    GOSUB MD2.HOME.WORK
    RETURN
  END IF

  IF MOTOR% = 34 THEN
    MD2.HOME.FIRST.MOTOR% = 3: MD2.HOME.SECOND.MOTOR% = 4
    GOSUB MD2.HOME.WORK
    RETURN
  END IF

  IF MOTOR% = 56 THEN
    MD2.HOME.FIRST.MOTOR% = 5: MD2.HOME.SECOND.MOTOR% = 6
    GOSUB MD2.HOME.WORK
    RETURN
  END IF

  RETURN


'******** DO THE HOME SEQUENCE FOR THE SELECTED MOTOR(S) *******
'
MD2.HOME.WORK:
  'SAVE VARIABLES.
  MD2.MOTOR.SAVE% = MOTOR%
  MD2.MOVE.ACTION.SAVE$ = move.action$
  MD2.CHECK.SWITCH.SAVE1$ = CHECK.SWITCH$(MD2.HOME.FIRST.MOTOR%)
  MD2.CHECK.SWITCH.SAVE2$ = CHECK.SWITCH$(MD2.HOME.SECOND.MOTOR%)
  MD2.DIRECTION.SAVE1$ = DIRECTION$(MD2.HOME.FIRST.MOTOR%)
  MD2.DIRECTION.SAVE2$ = DIRECTION$(MD2.HOME.SECOND.MOTOR%)
```

```
MD2.STEPS.TO.MOVE.SAVE1& = STEPS.TO.MOVE&(MD2.HOME.FIRST.MOTOR%)
MD2.STEPS.TO.MOVE.SAVE2& = STEPS.TO.MOVE&(MD2.HOME.SECOND.MOTOR%)

'DISABLE BACKLASH COMPENSATION DURING HOME SEQUENCE.
MD2.BACKLASH$ = "N"

'TURN ON SWITCH CHECKING.
CHECK.SWITCH$(MD2.HOME.FIRST.MOTOR%) = "Y"
CHECK.SWITCH$(MD2.HOME.SECOND.MOTOR%) = "Y"

'MOVE REVERSE CONTINOUSLY UNTIL SWITCH IS PRESSED.
move.action$ = "C"
DIRECTION$(MD2.HOME.FIRST.MOTOR%) = "R"
DIRECTION$(MD2.HOME.SECOND.MOTOR%) = "R"
GOSUB md2.move
IF RETURN.STATUS$ = "K" THEN GOTO MD2.HOME.WORK.DONE

'TURN OFF SWITCH CHECKING.
CHECK.SWITCH$(MD2.HOME.FIRST.MOTOR%) = "N"
CHECK.SWITCH$(MD2.HOME.SECOND.MOTOR%) = "N"

'MOVE FIRST MOTOR FORWARD UNTIL SWITCH IS OFF.
IF MD2.HOME.FIRST.MOTOR% <> 0 THEN
   DO:
      move.action$ = "S"
      DIRECTION$(MD2.HOME.FIRST.MOTOR%) = "F"
      STEPS.TO.MOVE&(MD2.HOME.FIRST.MOTOR%) = 1
      MOTOR% = MD2.HOME.FIRST.MOTOR%
      GOSUB md2.move
      IF RETURN.STATUS$ = "K" THEN GOTO MD2.HOME.WORK.DONE
      GOSUB MD2.SWITCH
   LOOP WHILE SWITCH$(MD2.HOME.FIRST.MOTOR%) = "Y"
END IF

'MOVE SECOND MOTOR FORWARD UNTIL SWITCH IS OFF.
IF MD2.HOME.SECOND.MOTOR% <> 0 THEN
   DO:
      move.action$ = "S"
      DIRECTION$(MD2.HOME.SECOND.MOTOR%) = "F"
      STEPS.TO.MOVE&(MD2.HOME.SECOND.MOTOR%) = 1
      MOTOR% = MD2.HOME.SECOND.MOTOR%
      GOSUB md2.move
      IF RETURN.STATUS$ = "K" THEN GOTO MD2.HOME.WORK.DONE
      GOSUB MD2.SWITCH
   LOOP WHILE SWITCH$(MD2.HOME.SECOND.MOTOR%) = "Y"
```

```
   END IF

MD2.HOME.WORK.DONE:
   'SET CURRENT POSITION TO 0 - HOME
   CURRENT.POSITION&(MD2.HOME.FIRST.MOTOR%) = 0
   CURRENT.POSITION&(MD2.HOME.SECOND.MOTOR%) = 0

   'RESTORE VARIABLES.
   MOTOR% = MD2.MOTOR.SAVE%
   move.action$ = MD2.MOVE.ACTION.SAVE$
   CHECK.SWITCH$(MD2.HOME.FIRST.MOTOR%) = MD2.CHECK.SWITCH.SAVE1$
   CHECK.SWITCH$(MD2.HOME.SECOND.MOTOR%) = MD2.CHECK.SWITCH.SAVE2$
   DIRECTION$(MD2.HOME.FIRST.MOTOR%) = MD2.DIRECTION.SAVE1$
   DIRECTION$(MD2.HOME.SECOND.MOTOR%) = MD2.DIRECTION.SAVE2$
   STEPS.TO.MOVE&(MD2.HOME.FIRST.MOTOR%) = MD2.STEPS.TO.MOVE.SAVE1&
   STEPS.TO.MOVE&(MD2.HOME.SECOND.MOTOR%) = MD2.STEPS.TO.MOVE.SAVE2&

   'TURN BACKLASH COMPENSATION BACK ON.
   MD2.BACKLASH$ = "Y"
   RETURN


'********************** MD-2 MOVE ***********************
'
'DESCRIPTION:
'  USED TO MOVE MOTOR(S). CAN MOVE MOTOR(S) 1,2,3,4,5,6 OR
'  1 & 2, 3 & 4 OR 5 & 6.  WHEN MOVING 2 MOTORS USES THE SPEED
'  VARIABLE THAT IS SLOWEST.
'
'VARIABLES RECEIVED:
'  BACKLASH%(MOTOR%)       BACKLASH COMPENSATION (# OF STEPS)
'  CHECK.KEY$(MOTOR%)      CHECK FOR KEYSTROKE, Y=YES,N=NO
'  CHECK.SWITCH$(MOTOR%)   CHECK FOR SWITCH, Y=YES,N=NO
'  DIRECTION$(MOTOR%)      F=FORWARD, R=REVERSE
'  MOVE.ACTION$            P=TO A POSITION, S=# STEPS, C=CONTINOUSLY
'  MOTOR%                  MOTOR NUMBER 1-6,12,34,56
'  POWER.DOWN$(MOTOR%)     POWER DOWN WHEN DONE Y=YES,N=NO
'  SPEED%(MOTOR%)          MOTOR SPEED 0 - 32000, 0=FASTEST
'  STEPS.TO.MOVE&(MOTOR%)  # OF STEPS TO MOVE MOTOR
'  STEP.TYPE$(MOTOR%)      S=SINGLE, D=DOUBLE, H=HALF
'  TARGET.POSITION&(MOTOR%) TARGET POSITION RELATIVE TO HOME
'
'VARIABLES RETURNED:
'  RETURN.STATUS$          O=OK, K=STOPED BY KEYPRESS, S=STOPED BY SWITCH
```

```
'  CURRENT.POSITION&(MOTOR%) # OF STEPS FROM HOME, +=FORWARD, -
=REVERSE
'
'
md2.move:
   'SET UP VARIABLES FOR MOVE.LOOP BASED ON MOTOR #.
   'SET SPEED TO SLOWEST MOTOR WHEN MOVING 2 MOTORS.
   IF MOTOR% = 1 THEN
      MD2.ADDRESS% = MD2.ADDR.12%: MD2.KEYPRESS$ = CHECK.KEY$(1)
      MD2.DELAY% = SPEED%(1): MD2.FIRST.MOTOR% = 1: MD2.SECOND.MOTOR% = 0
      GOTO MD2.SC
   END IF

   IF MOTOR% = 2 THEN
      MD2.ADDRESS% = MD2.ADDR.12%: MD2.KEYPRESS$ = CHECK.KEY$(2)
      MD2.DELAY% = SPEED%(2): MD2.FIRST.MOTOR% = 0: MD2.SECOND.MOTOR% = 2
      GOTO MD2.SC
   END IF

   IF MOTOR% = 3 THEN
      MD2.ADDRESS% = MD2.ADDR.34%: MD2.KEYPRESS$ = CHECK.KEY$(3)
      MD2.DELAY% = SPEED%(3): MD2.FIRST.MOTOR% = 3: MD2.SECOND.MOTOR% = 0
      GOTO MD2.SC
   END IF

   IF MOTOR% = 4 THEN
      MD2.ADDRESS% = MD2.ADDR.34%: MD2.KEYPRESS$ = CHECK.KEY$(4)
      MD2.DELAY% = SPEED%(4): MD2.FIRST.MOTOR% = 0: MD2.SECOND.MOTOR% = 4
      GOTO MD2.SC
   END IF

   IF MOTOR% = 5 THEN
      MD2.ADDRESS% = MD2.ADDR.56%: MD2.KEYPRESS$ = CHECK.KEY$(5)
      MD2.DELAY% = SPEED%(5): MD2.FIRST.MOTOR% = 5: MD2.SECOND.MOTOR% = 0
      GOTO MD2.SC
   END IF

   IF MOTOR% = 6 THEN
      MD2.ADDRESS% = MD2.ADDR.56%: MD2.KEYPRESS$ = CHECK.KEY$(6)
      MD2.DELAY% = SPEED%(6): MD2.FIRST.MOTOR% = 0: MD2.SECOND.MOTOR% = 6
      GOTO MD2.SC
   END IF

   IF MOTOR% = 12 THEN
      MD2.ADDRESS% = MD2.ADDR.12%: MD2.KEYPRESS$ = CHECK.KEY$(1)
```

```
      MD2.FIRST.MOTOR% = 1: MD2.SECOND.MOTOR% = 2
      IF SPEED%(1) > SPEED%(2) THEN
         MD2.DELAY% = SPEED%(1)
      ELSE
         MD2.DELAY% = SPEED%(2)
      END IF
      GOTO MD2.SC
   END IF

   IF MOTOR% = 34 THEN
      MD2.ADDRESS% = MD2.ADDR.34%: MD2.KEYPRESS$ = CHECK.KEY$(3)
      MD2.FIRST.MOTOR% = 3: MD2.SECOND.MOTOR% = 4
      IF SPEED%(3) > SPEED%(4) THEN
         MD2.DELAY% = SPEED%(3)
      ELSE
         MD2.DELAY% = SPEED%(4)
      END IF
      GOTO MD2.SC
   END IF

   IF MOTOR% = 56 THEN
      MD2.ADDRESS% = MD2.ADDR.56%: MD2.KEYPRESS$ = CHECK.KEY$(5)
      MD2.FIRST.MOTOR% = 5: MD2.SECOND.MOTOR% = 6
      IF SPEED%(5) > SPEED%(6) THEN
         MD2.DELAY% = SPEED%(5)
      ELSE
         MD2.DELAY% = SPEED%(6)
      END IF
      GOTO MD2.SC
   END IF

   'INVALID MOTOR.
   RETURN

MD2.SC:
   'SET SWITCH CHECKING.
   IF MD2.FIRST.MOTOR% = 0 THEN
      MD2.SWITCH.1$ = "N"
   ELSE
      MD2.SWITCH.1$ = CHECK.SWITCH$(MD2.FIRST.MOTOR%)
   END IF

   IF MD2.SECOND.MOTOR% = 0 THEN
      MD2.SWITCH.2$ = "N"
   ELSE
```

```
    MD2.SWITCH.2$ = CHECK.SWITCH$(MD2.SECOND.MOTOR%)
  END IF

  'SET POWER DOWN WHEN DONE.
  IF MD2.FIRST.MOTOR% = 0 THEN
    MD2.POWER.DOWN.1$ = "N"
  ELSE
    MD2.POWER.DOWN.1$ = POWER.DOWN$(MD2.FIRST.MOTOR%)
  END IF

  IF MD2.SECOND.MOTOR% = 0 THEN
    MD2.POWER.DOWN.2$ = "N"
  ELSE
    MD2.POWER.DOWN.2$ = POWER.DOWN$(MD2.SECOND.MOTOR%)
  END IF

  'SET STEP COUNT FOR 'CONTINOUS' MOVEMENT.
  IF move.action$ = "C" THEN
    IF MD2.FIRST.MOTOR% = 0 THEN MD2.STEPS.1& = 0 ELSE MD2.STEPS.1& = 8
    IF MD2.SECOND.MOTOR% = 0 THEN MD2.STEPS.2& = 0 ELSE MD2.STEPS.2& = 8
  END IF

  'SET STEP COUNT FOR 'STEP COUNT' MOVEMENT.
  IF move.action$ = "S" THEN
    IF MD2.FIRST.MOTOR% = 0 THEN
      MD2.STEPS.1& = 0
    ELSE
      MD2.STEPS.1& = STEPS.TO.MOVE&(MD2.FIRST.MOTOR%)
    END IF
    IF MD2.SECOND.MOTOR% = 0 THEN
      MD2.STEPS.2& = 0
    ELSE
      MD2.STEPS.2& = STEPS.TO.MOVE&(MD2.SECOND.MOTOR%)
    END IF
  END IF

  'SET STEP COUNT AND DIRECTION FOR 'POSITION' MOVEMENT.
  IF move.action$ = "P" THEN
    IF MD2.FIRST.MOTOR% = 0 THEN
      MD2.STEPS.1& = 0
    ELSE
      MD2.STEPS.1& = TARGET.POSITION&(MD2.FIRST.MOTOR%) -
CURRENT.POSITION&(MD2.FIRST.MOTOR%)
      IF MD2.STEPS.1& < 0 THEN
        MD2.STEPS.1& = MD2.STEPS.1& * -1
```

```
          DIRECTION$(MD2.FIRST.MOTOR%) = "R"
        ELSE
          DIRECTION$(MD2.FIRST.MOTOR%) = "F"
        END IF
      END IF
      IF MD2.SECOND.MOTOR% = 0 THEN
        MD2.STEPS.2& = 0
      ELSE
        MD2.STEPS.2& = TARGET.POSITION&(MD2.SECOND.MOTOR%) -
CURRENT.POSITION&(MD2.SECOND.MOTOR%)
        IF MD2.STEPS.2& < 0 THEN
          MD2.STEPS.2& = MD2.STEPS.2& * -1
          DIRECTION$(MD2.SECOND.MOTOR%) = "R"
        ELSE
          DIRECTION$(MD2.SECOND.MOTOR%) = "F"
        END IF
      END IF
    END IF

    'ADJUST FOR BACKLASH.
    'ZERO BACKLASH FOR POSITION ADJUSTMENT AFTER MOVE.
    MD2.BACKLASH.ADJ.1% = 0: MD2.BACKLASH.ADJ.2% = 0
    'ONLY ADJUST IF TURNED ON.
    IF MD2.BACKLASH$ = "Y" THEN
      'DONT ADJUST IF STEP COUNT IS 0.
      IF MD2.STEPS.1& <> 0 THEN
        'ADD BACKLASH IF DIRECTION HAS CHANGED.
        IF DIRECTION$(MD2.FIRST.MOTOR%) <>
MD2.LAST.DIRECTION$(MD2.FIRST.MOTOR%) THEN
          MD2.STEPS.1& = MD2.STEPS.1& + BACKLASH%(MD2.FIRST.MOTOR%)
          'SAVE FOR CURRENT POSITION ADJUSTMENT AFTER MOVE.
          MD2.BACKLASH.ADJ.1% = BACKLASH%(MD2.FIRST.MOTOR%)
        END IF
      END IF
      'DO THE SAME FOR THE SECOND MOTOR.
      IF MD2.STEPS.2& <> 0 THEN
        IF DIRECTION$(MD2.SECOND.MOTOR%) <>
MD2.LAST.DIRECTION$(MD2.SECOND.MOTOR%) THEN
          MD2.STEPS.2& = MD2.STEPS.2& + BACKLASH%(MD2.SECOND.MOTOR%)
          MD2.BACKLASH.ADJ.2% = BACKLASH%(MD2.SECOND.MOTOR%)
        END IF
      END IF
    END IF

    'ADJUST LAST DIRECTION EVEN IF NO BACKLASH COMPENSATION.
```

```
  IF MD2.STEPS.1& <> 0 THEN
    MD2.LAST.DIRECTION$(MD2.FIRST.MOTOR%) =
DIRECTION$(MD2.FIRST.MOTOR%)
  END IF
  IF MD2.STEPS.2& <> 0 THEN
    MD2.LAST.DIRECTION$(MD2.SECOND.MOTOR%) =
DIRECTION$(MD2.SECOND.MOTOR%)
  END IF

  'CREATE STEP PATTERNS.
  MD2.STEP.TYPE$ = step.type$(MD2.SECOND.MOTOR%)
  MD2.MOTOR% = MD2.SECOND.MOTOR%
  GOSUB MD2.GET.PATTERN

  'PUT PATTERN IN HIGH NIBBLE.
  FOR MD2.I% = 1 TO 8
    MD2.NEW.PATTERN%(MD2.I%) = MD2.PATTERN%(MD2.I%) * &H10
  NEXT MD2.I%
  MD2.STEP.TYPE$ = step.type$(MD2.FIRST.MOTOR%)
  MD2.MOTOR% = MD2.FIRST.MOTOR%
  GOSUB MD2.GET.PATTERN

  'PUT PATTERN IN LOW NIBBLE.
  FOR MD2.I% = 1 TO 8
    MD2.NEW.PATTERN%(MD2.I%) = MD2.NEW.PATTERN%(MD2.I%) OR
MD2.PATTERN%(MD2.I%)
  NEXT MD2.I%

  'PUT NEW FOUND PATTERN IN MD2.PATTERN.1%-MD2.PATTERN.8%
  'SINCE NON-ARRAY VARIABLES ARE FASTER.
  MD2.PATTERN.1% = MD2.NEW.PATTERN%(1): MD2.PATTERN.2% =
MD2.NEW.PATTERN%(2)
  MD2.PATTERN.3% = MD2.NEW.PATTERN%(3): MD2.PATTERN.4% =
MD2.NEW.PATTERN%(4)
  MD2.PATTERN.5% = MD2.NEW.PATTERN%(5): MD2.PATTERN.6% =
MD2.NEW.PATTERN%(6)
  MD2.PATTERN.7% = MD2.NEW.PATTERN%(7): MD2.PATTERN.8% =
MD2.NEW.PATTERN%(8)

  'MOVE THE MOTORS!
  GOSUB MD2.MOVE.LOOP

  'ADJUST POSITION VALUES.
  'DONT ADJUST IF CONTINIOUS MOVE ACTION.
  IF move.action$ = "C" THEN RETURN
```

```
   'ONLY ADJUST IF MOVED.
   IF MD2.FIRST.MOTOR% <> 0 THEN
      'IF WENT FORWARD THEN ADD FINAL COUNT, SUBTRACT BACKLASH.
      IF DIRECTION$(MD2.FIRST.MOTOR%) = "F" THEN
         CURRENT.POSITION&(MD2.FIRST.MOTOR%) =
(CURRENT.POSITION&(MD2.FIRST.MOTOR%) + MD2.COMPLETE.1&) -
MD2.BACKLASH.ADJ.1%
      END IF
      'IF WENT REVERSE THEN SUBTRACT FINAL COUNT, ADD BACKLASH.
      IF DIRECTION$(MD2.FIRST.MOTOR%) = "R" THEN
         CURRENT.POSITION&(MD2.FIRST.MOTOR%) =
(CURRENT.POSITION&(MD2.FIRST.MOTOR%) - MD2.COMPLETE.1&) +
MD2.BACKLASH.ADJ.1%
      END IF
   END IF
   'DO THE SAME FOR MOTOR 2
   IF MD2.SECOND.MOTOR% <> 0 THEN
      IF DIRECTION$(MD2.SECOND.MOTOR%) = "F" THEN
         CURRENT.POSITION&(MD2.SECOND.MOTOR%) =
(CURRENT.POSITION&(MD2.SECOND.MOTOR%) + MD2.COMPLETE.2&) -
MD2.BACKLASH.ADJ.2%
      END IF
      IF DIRECTION$(MD2.SECOND.MOTOR%) = "R" THEN
         CURRENT.POSITION&(MD2.SECOND.MOTOR%) =
(CURRENT.POSITION&(MD2.SECOND.MOTOR%) - MD2.COMPLETE.2&) +
MD2.BACKLASH.ADJ.2%
      END IF
   END IF
   RETURN

MD2.GET.PATTERN:
   'GET STEP PATTERN BASED ON STEP TYPE.
   IF MD2.MOTOR% = 0 THEN
      FOR i = 1 TO 8
         MD2.PATTERN%(i) = &HF
      NEXT i
      RETURN
   END IF
   IF MD2.STEP.TYPE$ = "H" THEN
      MD2.PATTERN%(1) = &HE: MD2.PATTERN%(2) = &HC
      MD2.PATTERN%(3) = &HD: MD2.PATTERN%(4) = &H9
      MD2.PATTERN%(5) = &HB: MD2.PATTERN%(6) = &H3
      MD2.PATTERN%(7) = &H7: MD2.PATTERN%(8) = &H6
   END IF
   IF MD2.STEP.TYPE$ = "S" THEN
```

```
    MD2.PATTERN%(1) = &HE: MD2.PATTERN%(2) = &HD
    MD2.PATTERN%(3) = &HB: MD2.PATTERN%(4) = &H7
    MD2.PATTERN%(5) = &HE: MD2.PATTERN%(6) = &HD
    MD2.PATTERN%(7) = &HB: MD2.PATTERN%(8) = &H7
  END IF
  IF MD2.STEP.TYPE$ = "D" THEN
    MD2.PATTERN%(1) = &HC: MD2.PATTERN%(2) = &H9
    MD2.PATTERN%(3) = &H3: MD2.PATTERN%(4) = &H6
    MD2.PATTERN%(5) = &HC: MD2.PATTERN%(6) = &H9
    MD2.PATTERN%(7) = &H3: MD2.PATTERN%(8) = &H6
  END IF

  'ADJUST FOR CORRECT DIRECTION.
  IF DIRECTION$(MD2.MOTOR%) = "F" THEN
    SWAP MD2.PATTERN%(1), MD2.PATTERN%(8)
    SWAP MD2.PATTERN%(2), MD2.PATTERN%(7)
    SWAP MD2.PATTERN%(3), MD2.PATTERN%(6)
    SWAP MD2.PATTERN%(4), MD2.PATTERN%(5)
  END IF

  'ADJUST FOR CORRECT STARTING PATTERN.
  'MD2.ROTATE% = # OF TIMES TO ROTATE PATTERN.
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(1) THEN
MD2.ROTATE% = 1
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(2) THEN
MD2.ROTATE% = 2
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(3) THEN
MD2.ROTATE% = 3
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(4) THEN
MD2.ROTATE% = 4
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(5) THEN
MD2.ROTATE% = 5
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(6) THEN
MD2.ROTATE% = 6
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(7) THEN
MD2.ROTATE% = 7
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = MD2.PATTERN%(8) THEN
MD2.ROTATE% = 8
  IF MD2.LAST.PATTERN%(MD2.MOTOR%) = 0 THEN MD2.ROTATE% = 8
  FOR MD2.I% = 1 TO MD2.ROTATE%
    MD2.PATTERN% = MD2.PATTERN%(1)
    MD2.PATTERN%(1) = MD2.PATTERN%(2): MD2.PATTERN%(2) = MD2.PATTERN%(3)
    MD2.PATTERN%(3) = MD2.PATTERN%(4): MD2.PATTERN%(4) = MD2.PATTERN%(5)
    MD2.PATTERN%(5) = MD2.PATTERN%(6): MD2.PATTERN%(6) = MD2.PATTERN%(7)
    MD2.PATTERN%(7) = MD2.PATTERN%(8): MD2.PATTERN%(8) = MD2.PATTERN%
```

```
   NEXT MD2.I%
   RETURN


'****************** MD-2 MOVE LOOP ******************
'
'DESCRIPTION:
'  THIS ROUTINE IS NEEDED BY MD2.MOVE ONLY.
'  FAST MOVE LOOP THAT OUTPUTS STEP PATTERNS, COUNTS
'  STEPS, CHECKS SWITCHES AND KEYBOARD.
'
'VARIABLES RECEIVED:
'  MD2.ADDRESS%          DRIVER ADDRESS
'  MD2.DELAY%           DELAY BETWEEN STEPS, 0=FAST
'  MD2.KEYPRESS$         CHECK FOR KEY PRESSED, Y=YES, N=NO
'  MD2.STEPS.1&         # OF STEPS TO MOVE FOR FIRST MOTOR, 0=NONE
'  MD2.STEPS.2&         # OF STEPS TO MOVE FOR SECOND MOTOR, 0=NONE
'  MD2.SWITCH.1$         CHECK FOR SWITCHES, Y=YES, N=NO
'  MD2.SWITCH.2$         CHECK FOR SWITCHES, Y=YES, N=NO
'  MD2.PATTERN.1% - MD2.PATTERN.8%   STEP PATTERNS
'  MD2.POWER.DOWN.1$      POWER DOWN WHEN DONE, Y=YES, N=NO
'  MD2.POWER.DOWN.2$      POWER DOWN WHEN DONE, Y=YES, N=NO
'
'VARIABLES RETURNED:
'  RETURN.STATUS$        O=OK, K=STOPED BY KEYPRESS, S=STOPED BY SWITCH
'  MD2.COMPLETE.1&       # OF STEPS COMPLETED ON FIRST MOTOR
'  MD2.COMPLETE.2&       # OF STEPS COMPLETED ON SECOND MOTOR
'
'
MD2.MOVE.LOOP:

     MD2.DONE.1$ = "N": MD2.DONE.2$ = "N"
     MD2.COMPLETE.1& = MD2.STEPS.1&: MD2.COMPLETE.2& = MD2.STEPS.2&
     RETURN.STATUS$ = "O"

     'CHECK FOR SPACE BAR TO EMERGENCY STOP MOTORS.
ML1:   IF MD2.KEYPRESS$ = "N" THEN GOTO ML2
     IF INKEY$ = " " THEN RETURN.STATUS$ = "K": GOSUB MD2.STOP.1: GOSUB
MD2.STOP.2

     'STEP #1.
     'CHECK SWITCHES.
     IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
```

```
    IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
ML2:  'CHECK & UPDATE STEP COUNTS.
    IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
    MD2.STEPS.1& = MD2.STEPS.1& - 1
    IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
    MD2.STEPS.2& = MD2.STEPS.2& - 1
    'OUTPUT THE STEP PATTERN.
    OUT MD2.ADDRESS%, MD2.PATTERN.1%
    'DO DELAY LOOP FOR SPEED CONTROL.
    IF MD2.DELAY% = 0 THEN GOTO ML4
    MD2.I% = MD2.DELAY%
ML3:  MD2.I% = MD2.I% - 1
    IF MD2.I% <> 0 THEN GOTO ML3

    'ADD TO STEP COUNT FOR CONTINOUS MOVEMENT.
ML4:  IF move.action$ = "C" THEN MD2.STEPS.1& = MD2.STEPS.1& + 8: MD2.STEPS.2& =
MD2.STEPS.2& + 8

    'STEP #2.
    IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
    IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
    IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
    MD2.STEPS.1& = MD2.STEPS.1& - 1
    IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
    MD2.STEPS.2& = MD2.STEPS.2& - 1
    OUT MD2.ADDRESS%, MD2.PATTERN.2%
    IF MD2.DELAY% = 0 THEN GOTO ML6
    MD2.I% = MD2.DELAY%
ML5:  MD2.I% = MD2.I% - 1
    IF MD2.I% <> 0 THEN GOTO ML5

    'STEP #3.
ML6:  IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0
THEN RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
    IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
    IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
    MD2.STEPS.1& = MD2.STEPS.1& - 1
    IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
    MD2.STEPS.2& = MD2.STEPS.2& - 1
    OUT MD2.ADDRESS%, MD2.PATTERN.3%
    IF MD2.DELAY% = 0 THEN GOTO ML8
```

```
      MD2.I% = MD2.DELAY%
ML7:  MD2.I% = MD2.I% - 1
      IF MD2.I% <> 0 THEN GOTO ML7

      'STEP #4.
ML8:  IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0
THEN RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
      IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
      IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
      MD2.STEPS.1& = MD2.STEPS.1& - 1
      IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
      MD2.STEPS.2& = MD2.STEPS.2& - 1
      OUT MD2.ADDRESS%, MD2.PATTERN.4%
      IF MD2.DELAY% = 0 THEN GOTO ML10
      MD2.I% = MD2.DELAY%
ML9:  MD2.I% = MD2.I% - 1
      IF MD2.I% <> 0 THEN GOTO ML9

      'STEP #5.
ML10: IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0
THEN RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
      IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
      IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
      MD2.STEPS.1& = MD2.STEPS.1& - 1
      IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
      MD2.STEPS.2& = MD2.STEPS.2& - 1
      OUT MD2.ADDRESS%, MD2.PATTERN.5%
      IF MD2.DELAY% = 0 THEN GOTO ML12
      MD2.I% = MD2.DELAY%
ML11: MD2.I% = MD2.I% - 1
      IF MD2.I% <> 0 THEN GOTO ML11

      'STEP #6.
ML12: IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0
THEN RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
      IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
      IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
      MD2.STEPS.1& = MD2.STEPS.1& - 1
      IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
      MD2.STEPS.2& = MD2.STEPS.2& - 1
      OUT MD2.ADDRESS%, MD2.PATTERN.6%
      IF MD2.DELAY% = 0 THEN GOTO ML14
```

```
      MD2.I% = MD2.DELAY%
ML13:  MD2.I% = MD2.I% - 1
      IF MD2.I% <> 0 THEN GOTO ML13

      'BOTH MOTORS DONE?
ML14:  IF MD2.DONE.1$ = "Y" AND MD2.DONE.2$ = "Y" THEN RETURN

      'STEP #7.
      IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
      IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
      IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
      MD2.STEPS.1& = MD2.STEPS.1& - 1
      IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
      MD2.STEPS.2& = MD2.STEPS.2& - 1
      OUT MD2.ADDRESS%, MD2.PATTERN.7%
      IF MD2.DELAY% = 0 THEN GOTO ML16
      MD2.I% = MD2.DELAY%
ML15:  MD2.I% = MD2.I% - 1
      IF MD2.I% <> 0 THEN GOTO ML15

      'STEP #8.
ML16:  IF MD2.SWITCH.1$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H20) = 0
THEN RETURN.STATUS$ = "S": GOSUB MD2.STOP.1
      IF MD2.SWITCH.2$ = "Y" THEN IF (INP(MD2.ADDRESS% + 1) AND &H10) = 0 THEN
RETURN.STATUS$ = "S": GOSUB MD2.STOP.2
      IF MD2.STEPS.1& = 0 THEN GOSUB MD2.STOP.1
      MD2.STEPS.1& = MD2.STEPS.1& - 1
      IF MD2.STEPS.2& = 0 THEN GOSUB MD2.STOP.2
      MD2.STEPS.2& = MD2.STEPS.2& - 1
      OUT MD2.ADDRESS%, MD2.PATTERN.8%
      IF MD2.DELAY% = 0 THEN GOTO ML1
      MD2.I% = MD2.DELAY%
ML17:  MD2.I% = MD2.I% - 1
      IF MD2.I% <> 0 THEN GOTO ML17

      GOTO ML1

'MOTOR # 1 DONE.
MD2.STOP.1:
   IF MD2.DONE.1$ = "Y" THEN RETURN
   IF (INP(MD2.ADDRESS%) AND &HF) <> &HF THEN
MD2.LAST.PATTERN%(MD2.FIRST.MOTOR%) = INP(MD2.ADDRESS%) AND &HF
   'TURN POWER OFF MOTOR 1?
```

```
  IF MD2.POWER.DOWN.1$ = "Y" THEN
    MD2.PATTERN.1% = MD2.PATTERN.1% OR &HF
    MD2.PATTERN.2% = MD2.PATTERN.2% OR &HF
    MD2.PATTERN.3% = MD2.PATTERN.3% OR &HF
    MD2.PATTERN.4% = MD2.PATTERN.4% OR &HF
    MD2.PATTERN.5% = MD2.PATTERN.5% OR &HF
    MD2.PATTERN.6% = MD2.PATTERN.6% OR &HF
    MD2.PATTERN.7% = MD2.PATTERN.7% OR &HF
    MD2.PATTERN.8% = MD2.PATTERN.8% OR &HF
  ELSE
  'SET CURRENT MOTOR 1 PATTERN IN ALL MD2.PATTERN%'S.
    MD2.TEMP% = (INP(MD2.ADDRESS%) AND &HF)
    MD2.PATTERN.1% = ((MD2.PATTERN.1% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.2% = ((MD2.PATTERN.2% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.3% = ((MD2.PATTERN.3% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.4% = ((MD2.PATTERN.4% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.5% = ((MD2.PATTERN.5% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.6% = ((MD2.PATTERN.6% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.7% = ((MD2.PATTERN.7% AND &HF0) OR MD2.TEMP%)
    MD2.PATTERN.8% = ((MD2.PATTERN.8% AND &HF0) OR MD2.TEMP%)
  END IF
  MD2.DONE.1$ = "Y": MD2.SWITCH.1$ = "N": MD2.COMPLETE.1& =
MD2.COMPLETE.1& - MD2.STEPS.1&
  RETURN

'MOTOR # 2 DONE.
MD2.STOP.2:
  IF MD2.DONE.2$ = "Y" THEN RETURN
  IF (INP(MD2.ADDRESS%) AND &HF0) <> &HF0 THEN
MD2.LAST.PATTERN%(MD2.SECOND.MOTOR%) = (INP(MD2.ADDRESS%) AND &HF0) /
&H10
  'TURN POWER OFF MOTOR 2?
  IF MD2.POWER.DOWN.2$ = "Y" THEN
    MD2.PATTERN.1% = MD2.PATTERN.1% OR &HF0
    MD2.PATTERN.2% = MD2.PATTERN.2% OR &HF0
    MD2.PATTERN.3% = MD2.PATTERN.3% OR &HF0
    MD2.PATTERN.4% = MD2.PATTERN.4% OR &HF0
    MD2.PATTERN.5% = MD2.PATTERN.5% OR &HF0
    MD2.PATTERN.6% = MD2.PATTERN.6% OR &HF0
    MD2.PATTERN.7% = MD2.PATTERN.7% OR &HF0
    MD2.PATTERN.8% = MD2.PATTERN.8% OR &HF0
  ELSE
    MD2.TEMP% = (INP(MD2.ADDRESS%) AND &HF0)
    MD2.PATTERN.1% = ((MD2.PATTERN.1% AND &HF) OR MD2.TEMP%)
    MD2.PATTERN.2% = ((MD2.PATTERN.2% AND &HF) OR MD2.TEMP%)
```

```
      MD2.PATTERN.3% = ((MD2.PATTERN.3% AND &HF) OR MD2.TEMP%)
      MD2.PATTERN.4% = ((MD2.PATTERN.4% AND &HF) OR MD2.TEMP%)
      MD2.PATTERN.5% = ((MD2.PATTERN.5% AND &HF) OR MD2.TEMP%)
      MD2.PATTERN.6% = ((MD2.PATTERN.6% AND &HF) OR MD2.TEMP%)
      MD2.PATTERN.7% = ((MD2.PATTERN.7% AND &HF) OR MD2.TEMP%)
      MD2.PATTERN.8% = ((MD2.PATTERN.8% AND &HF) OR MD2.TEMP%)
   END IF
   MD2.DONE.2$ = "Y": MD2.SWITCH.2$ = "N": MD2.COMPLETE.2& =
MD2.COMPLETE.2& - MD2.STEPS.2&
   RETURN

   END
finish:
END SUB
```

```
SUB timer1_timer ()

' Keep the clock displayed on the user interface screen
' current.  Convert 24 hour system clock to 12 hour clock.

   DIM temp_tim AS STRING
   DIM hours AS INTEGER
   temp_tim = TIME$
   hours = VAL(temp_tim)
   IF hours > 12 THEN MID$(temp_tim, 1, 2) = STR$(hours - 12)
   label6.caption = "Time: " + temp_tim
END SUB
```

```
SUB two_hog ()

' Used when sampling is started in three catheters per pig mode.
' Determines which sample is to be taken next and waits for the
' appropriate time to occur before calling the take_sample
' procedure.

   DIM dstart AS STRING
   current1 = 0: current2 = 0
   tstart = VAL(MID$(TIME$, 1, 2)) * 60 + VAL(MID$(TIME$, 4, 2)) + VAL(MID$(TIME$, 7,
2)) / 60
   dstart = DATE$

  DO WHILE VAL(text1(current1).text) <> 0 OR VAL(text2(current2).text) <> 0
   IF (VAL(text1(current1).text) < VAL(text2(current2).text) AND VAL(text1(current1).text) <> 0)
OR VAL(text2(current2).text) = 0 THEN
       waittime = VAL(text1(current1).text): tube% = current1 * 3 + 1: valve% = 0: current1 =
current1 + 1
   ELSE waittime = VAL(text2(current2).text): tube% = 51 + current2 * 3: valve% = 3: current2 =
current2 + 1
   END IF

   DO
      IF DATE$ <> dstart THEN tstart = tstart - 1440: dstart = DATE$
      CALL timer1_timer
   LOOP WHILE VAL(MID$(TIME$, 1, 2)) * 60 + VAL(MID$(TIME$, 4, 2)) +
VAL(MID$(TIME$, 7, 2)) / 60 < tstart + waittime

   screen.HIDE

   CALL take_sample(tube%, valve%)

   tube% = tube% + 1
   valve% = valve% + 1
   CALL take_sample(tube%, valve%)

   tube% = tube% + 1
   valve% = valve% + 1
   CALL take_sample(tube%, valve%)

   screen.SHOW
  LOOP
END SUB
```

```
SUB valve_flush ()

' Procedure removes any residual blood left in the six
' inlet valve.

' Open all valves and all inlets.  Allow blood or saline
' to draw partially into each catheter.
   MD% = 58
   D%(0) = 1
   D%(1) = 0
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

   begin = VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2))
10  total = 0
   FOR i = 0 TO 5
      IF VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2)) > begin + VAL(pull(i).text)
THEN total = total + 2 ^ i
   NEXT i

   MD% = 58
   D%(0) = 1
   D%(1) = total
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

   IF total <> 63 THEN GOTO 10

' Close three-way valve A.  Wait to flush the lines on the
' needle side of the valves.
   MD% = 58
   D%(0) = 1
   D%(1) = 64
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)
   FOR i = 1 TO 40000: NEXT i

' Close three-way valve B, open three-way valve A, then
' open each of the six inlets one at a time to flush the
' blood or saline back into the pig or saline container.

   FOR valve = 0 TO 5
      MD% = 58
      D%(0) = 1
      D%(1) = 191 - 2 ^ valve
```

```
      D%(2) = 1
      CALL CIODIO(MD%, VARPTR(D%(0)), F%)
      begin = VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2))
      DO
      LOOP WHILE VAL(MID$(TIME$, 4, 2)) * 60 + VAL(MID$(TIME$, 7, 2)) < begin +
VAL(push(valve).text)
   NEXT valve

' Close all valves.
   MD% = 58
   D%(0) = 1
   D%(1) = 255
   D%(2) = 1
   CALL CIODIO(MD%, VARPTR(D%(0)), F%)

END SUB
```

```
' ---------------------------------------------------------------------
' Visual Basic for MS-DOS Common Dialog Toolkit
'
' The Common Dialog Toolkit (CMNDLG.BAS and CMNDLGF.FRM)
' provides support for the following dialogs:
'     FileOpen
'     FileSave
'     FilePrint
'     FindText
'     ChangeText
'     ColorPalette
'     About
'
' Support for each dialog is provided via procedures with
' these same names that create the corresponding dialog
' and return user input to your program.  These procedures
' only provide the user interface and return user input.
' They do not actually carry out the corresponding actions
' such as opening the file.  Detailed descriptions of
' these procedures are contained in the comment headers
' above each.
'
' Special routines to preload (CmnDlgRegister) and unload
' (CmnDlgClose) the common dialog form for better
' performance (loaded forms display faster than unloaded
' forms) are also provided.  These routines are optional
' however as the common dialog form will automatically load
' and unload each time you invoke a common dialog.  Preloading
' the common dialog form will make common dialog access
' faster but will require more memory.
'
' All common dialogs are created from the same form (CMNDLGF.FRM).
' The necessary controls for each dialog are children of
' a container picture box for the dialog.  Thus the
' form (CMNDLGF.FRM) contains a picture box with
' appropriate controls for common dialog listed above.
' When a particular common dialog is created and displayed,
' the container picture box for that dialog is made visible
' (thus all controls on that picture box become visible)
' and the form is centered and sized to match the
' container picture box.
'
' To use these common dialogs in your programs, include
' CMNDLG.BAS and CMNDLGF.FRM in your program or use the
```

```
' supplied library (CMNDLG.LIB, CMNDLGA.LIB - AltMath version
' for Professional Edition only) and Quick library (CMNDLG.QLB)
' and call the appropriate procedure to invoke the dialog
' you need.
'
' Copyright (C) 1982-1992 Microsoft Corporation
'
' You have a royalty-free right to use, modify, reproduce
' and distribute the sample applications and toolkits provided with
' Visual Basic for MS-DOS (and/or any modified version)
' in any way you find useful, provided that you agree that
' Microsoft has no warranty, obligations or liability for
' any of the sample applications or toolkits.
' ----------------------------------------------------------------------

' Include file containing declarations for called procedures.
'$INCLUDE: 'CMNDLG.BI'

CONST FALSE = 0
CONST TRUE = NOT FALSE
```

```
' Click event procedure for Open/Save dialog Cancel button.
' Makes dialog invisible to return control to FileOpen or FileSave
' procedure (dialog was shown modally).
'
SUB cmdOpenCancel_Click ()
   txtOpenFile.SETFOCUS
   Visible = FALSE
END SUB
```

```
' Click event procedure for Open/Save dialog OK button.
' Determines whether user has selected a file or whether
' path and pattern need to be changed.
'
SUB cmdOpenOK_CLick ()
   ' Set up error handling for directory/drive change errors.
   ON LOCAL ERROR GOTO OKError

   cmdOpenOK.SETFOCUS                ' Set focus to button, so focus can be reset to edit field if
needed.

   ' Update Directory listbox path if user single
   ' clicked or used arrow keys in Directory listbox
   ' (only double click automatically changes path).
   dirOpenList.Path = dirOpenList.List(dirOpenList.ListIndex)

   ' If edit field filename does not match File listbox filename
   ' then assign edit field value to File listbox filename
   ' and let it determine if path or pattern need to be
   ' changed.
   IF filOpenList.FileName <> txtOpenFile.Text THEN
      OldPattern$ = filOpenList.Pattern   ' Save old pattern.

      ' Let File listbox control determine if path
      ' or pattern or filename needs to be updated.
      ' PathChange event will be triggered if path needs
      ' updating, PatternChange event will be triggered if
      ' pattern needs updating, and DblClick event will
      ' be triggered if a valid filename has been given.
      filOpenList.FileName = txtOpenFile.Text

      ' If a valid filename was not given (dialog is
      ' still visible to user after DblClick event),
      ' then update the edit field appropriately.
      IF Visible = TRUE THEN
         ' If no pattern change was indicated then either
         ' a new filename was specified for Save dialog
         ' or file was not found for Open dialog.
         IF (INSTR(txtOpenFile.Text, "*") + INSTR(txtOpenFile.Text, "?") < 1) THEN
            IF INSTR(Tag, "SAVE") THEN
               CALL filOpenList_DblClick
            ELSE
               MSGBOX "File not found", 0, Caption
               filOpenList.Pattern = OldPattern$   ' Restore old File listbox search pattern.
               txtOpenFile.SETFOCUS
```

```
            END IF
          ' Pattern change was indicated so just update
          ' textbox with pattern.
          ELSE
            txtOpenFile.Text = filOpenList.Pattern
            txtOpenFile.SETFOCUS
          END IF
        END IF
      ' File has been selected by user so close dialog
      ' and return control to FileOpen or FileSave routine.
      ELSE
        CALL filOpenList_DblClick
      END IF

OKErrorReturn:
      EXIT SUB

' Drive/Path error handling routine.
OKError:
      MSGBOX ERROR$, 0, Caption          ' Display error message.
      txtOpenFile.SETFOCUS               ' Set focus to edit field so error can be fixed.
      RESUME OKErrorReturn               ' Exit procedure.
END SUB
```

```
' Change event procedure for Open/Save dialog Directory listbox.
' Updates the File listbox path to reflect
' the directory change.
'
SUB dirOpenList_Change ()
   ' Set up error handling for path errors.
   ON LOCAL ERROR GOTO DirChangeError

   ' Update file listbox path.
   filOpenList.Path = dirOpenList.Path

   ' Display new path to the user.
   lblOpenPath.Caption = dirOpenList.Path

   ' Update text box with search pattern.
   txtOpenFile.Text = filOpenList.Pattern

DirChangeErrorReturn:
   EXIT SUB

' Path change error handling routine.
DirChangeError:
   MSGBOX ERROR$, 0, Caption          ' Display error message.
   txtOpenFile.SETFOCUS               ' Set focus to edit field so error can be fixed.
   RESUME DirChangeErrorReturn        ' Exit procedure.
END SUB
```

```
' Change event procedure for Open/Save dialog Drive listbox.
' Updates the Directory listbox path to reflect
' the drive change.
'
SUB drvOpenList_Change ()
   ' Set up error handling for path errors.
   ON LOCAL ERROR GOTO DriveChangeError

   ' Update Dir listbox path.
   dirOpenList.Path = drvOpenList.Drive

DriveChangeErrorReturn:
   EXIT SUB

' Path change error handling routine.
DriveChangeError:
   MSGBOX ERROR$, 0, Caption              ' Display error message.
   drvOpenList.Drive = dirOpenList.Path   ' Reset drive.
   RESUME DriveChangeErrorReturn          ' Exit procedure.
END SUB
```

```
' Click event procedure for Open/Save dialog File listbox.
' Selects the file and updates the edit field.
'
SUB filOpenList_Click ()
   txtOpenFile.Text = filOpenList.FileName
END SUB
```

```
' Double Click event procedure for Open/Save dialog File listbox.
' File has been selected by the user so make dialog
' invisible to return control to FileOpen or FileSave
' procedure (dialog was shown modally).
'
SUB filOpenList_DblClick ()
   txtOpenFile.SETFOCUS
   Visible = FALSE
   cmdOpenCancel.Tag = "FALSE"
END SUB
```

```
' PathChange event procedure for Open/Save dialog File listbox.
' Updates the Drive listbox drive and Directory
' listbox path to reflect the change.
'
SUB filOpenList_PathChange ()
   ' Set up error handling for path errors.
   ON LOCAL ERROR GOTO FileChangeError

   ' Update drive and path.
   drvOpenList.Drive = filOpenList.Path
   dirOpenList.Path = filOpenList.Path

FileChangeErrorReturn:
   EXIT SUB

' Path change error handling routine.
FileChangeError:
   MSGBOX ERROR$, 0, Caption            ' Display error message.
   drvOpenList.Drive = dirOpenList.Path   ' Reset drive.
   filOpenList.Path = dirOpenList.Path    ' Reset path.
   RESUME FileChangeErrorReturn          ' Exit procedure.
END SUB
```

```
' PatternChange event procedure for Open/Save dialog File listbox.
' Uppercases search pattern for subsequent display
' in edit field.
'
SUB filOpenList_PatternChange ()
   filOpenList.Pattern = UCASE$(filOpenList.Pattern)
END SUB
```

```
' GotFocus event procedure for OpenFile textbox.
' Selects all textbox text when textbox receives focus
' (for easier replacement of text).
'
SUB txtOpenFile_GotFocus ()
   txtOpenFile.SelStart = 0
   txtOpenFile.SelLength = LEN(txtOpenFile.Text)
END SUB
' ---------------------------------------------------------------------
' Visual Basic for MS-DOS Common Dialog Toolkit
'
' The Common Dialog Toolkit (CMNDLG.BAS and CMNDLGF.FRM)
' provides support for the following dialogs:
'      FileOpen
'      FileSave
'      FilePrint
'      FindText
'      ChangeText
'      ColorPalette
'      About
'
' Support for each dialog is provided via procedures with
' these same names that create the corresponding dialog
' and return user input to your program.  These procedures
' only provide the user interface and return user input.
' They do not actually carry out the corresponding actions
' such as opening the file.  Detailed descriptions of
' these procedures are contained in the comment headers
' above each.
'
' Special routines to preload (CmnDlgRegister) and unload
' (CmnDlgClose) the common dialog form for better
' performance (loaded forms display faster than unloaded
' forms) are also provided.  These routines are optional
' however as the common dialog form will automatically load
' and unload each time you invoke a common dialog.  Preloading
' the common dialog form will make common dialog access
' faster but will require more memory.
'
' All common dialogs are created from the same form (CMNDLGF.FRM).
' The necessary controls for each dialog are children of
' a container picture box for the dialog.  Thus the
' form (CMNDLGF.FRM) contains a picture box with
' appropriate controls for common dialog listed above.
' When a particular common dialog is created and displayed,
```

```
' the container picture box for that dialog is made visible
' (thus all controls on that picture box become visible)
' and the form is centered and sized to match the
' container picture box.
'
' To use these common dialogs in your programs, include
' CMNDLG.BAS and CMNDLGF.FRM in your program or use the
' supplied library (CMNDLG.LIB, CMNDLGA.LIB - AltMath version
' for Professional Edition only) and Quick library (CMNDLG.QLB)
' and call the appropriate procedure to invoke the dialog
' you need.
'
' Copyright (C) 1982-1992 Microsoft Corporation
'
' You have a royalty-free right to use, modify, reproduce
' and distribute the sample applications and toolkits provided with
' Visual Basic for MS-DOS (and/or any modified version)
' in any way you find useful, provided that you agree that
' Microsoft has no warranty, obligations or liability for
' any of the sample applications or toolkits.
' ----------------------------------------------------------------------

' Include file containing declarations for called procedures.
'$INCLUDE: 'CMNDLG.BI'

' Common dialog form
'$FORM frmCmnDlg

CONST FALSE = 0
CONST TRUE = NOT FALSE

'-------------------------------------------------
' Sample usage of the common dialogs. This code is
' only executed if CMNDLG.BAS is the start-up file.
' Parameter information for each common dialog
' routine is contained in the header comments for
' the routine.
'-------------------------------------------------

' Set desktop attributes for demonstration.
SCREEN.ControlPanel(16) = 5
SCREEN.ControlPanel(5) = 1
```

```
' Load and register common dialog form before using it
' to obtain better performance (loaded forms display faster
' than unloaded forms). Form will remain loaded (but
' invisible) until this routine is called again to
' unload it. Thus, all common dialog usage in your
' program will be faster. Keeping the form loaded
' requires more memory, however, than loading and
' unloading it each time a common dialog is used.
'
CmnDlgRegister Success%

IF NOT Success% THEN END

' Displays Open dialog.
' Returns filename and pathname for open operation.
'
FileOpen FileName$, PathName$, "", "", 0, 7, Flags%, Cancel%


' Displays Save dialog.
' Returns filename and pathname for save operation.
'
FileSave FileName$, PathName$, "*.TXT", "My Save Dialog", 0, 7, Flags%, Cancel%



' Unload common dialog form (if you have preloaded it for
' better performance) so program will terminate,
' otherwise common dialog form will remain loaded but
' invisible.
'
CmnDlgClose
```

```
'
' Unloads common dialog form (if you have preloaded it for
' better performance) so program will terminate,
' otherwise common dialog form will remain loaded but
' invisible.  This routine should be called if
' CmnDlgRegister was used to preload the form.  If
' CmnDlgRegister was not used, the form will be unloaded
' after each use.
'
SUB CmnDlgClose ()
   UNLOAD frmCmnDlg          ' Unload form.
END SUB
```

```
' CmnDlgRegister common dialog support routine
'
' Loads and registers common dialog form before using it
' to obtain better performance (loaded forms display faster
' than unloaded forms). Form will remain loaded (but
' invisible) until this routine is called again to
' unload it. Thus, all common dialog usage in your
' program will be faster (form is not loaded and unload
' each time a common dialog is invoked). Keeping the
' form loaded requires more memory, however, than loading
' and unloading it each time a common dialog is used.
'
' Use of this routine is optional since the common dialog
' form does not need to be loaded before it is used (each
' common dialog routine will load the form is it is not
' loaded).
'
' Parameters:
'   Success - returns TRUE (-1) if the load or unload
'         attempt was successful, otherwise returns
'         FALSE (0).
'
SUB CmnDlgRegister (Success AS INTEGER)
   ' Set up error handling.
   ON LOCAL ERROR GOTO RegisterError

   LOAD frmCmnDlg            ' Load form.
   frmCmnDlg.Tag = "H"       ' Set flag for keeping form loaded after
                             ' each common dialog usage.

   Success = TRUE
   EXIT SUB

' Option error handling routine.
' Trap errors that occur when preloading dialog.
RegisterError:
   SELECT CASE ERR
   CASE 7:                           ' Out of memory.
      MSGBOX "Out of memory.  Can't load Common Dialogs.", 0, "Common Dialog"
      Success = FALSE
      EXIT SUB
   CASE ELSE
      MSGBOX ERROR$ + ".  Can't load Common Dialogs.", 0, "Common Dialog"
      Success = FALSE
      EXIT SUB
```

```
    END SELECT
END SUB
```

```
' FileOpen common dialog support routine
'
' Displays Open dialog which allows users to select a
' file from disk.  This procedure only provides
' the user interface and returns user input.  It does
' not actually carry out the corresponding action.
'
' Parameters:
'   FileName - returns the name (without path) of the
'         file the user wants to open.  To supply
'         default filename in dialog, assign default
'         to FileName then pass it to this procedure.
'   PathName - returns the path (without filename) of
'         the file the user wants to open.  To supply
'         default path in dialog, assign default to
'         PathName then pass it to this procedure.
'         Note, only pass a valid drive and path. Do
'         not include a filename or file pattern.
'   DefaultExt - sets the default search pattern for the
'         File Listbox.  Default pattern when DefaultExt
'         is null is "*.*".  To specify a different
'         search pattern (i.e. "*.BAS"), assign new
'         value to DefaultExt then pass it to this
'         procedure.
'   DialogTitle - sets the dialog title.  Default title
'         when DialogTitle is null is "Open".  To
'         specify a different title (i.e. "Open My File"),
'         assign new value to DialogTitle then pass it to
'         this procedure.
'   ForeColor - sets the dialog foreground color.  Does not affect
'         SCREEN.ControlPanel color settings.
'   BackColor - sets the dialog background color.  Does not affect
'         SCREEN.ControlPanel color settings.
'   Flags - unused.  Use this to customize dialog action if needed.
'   Cancel - returns whether or not user pressed the dialog's Cancel
'         button.  True (-1) means the user cancelled the dialog.
'
SUB FileOpen (FileName AS STRING, PathName AS STRING, DefaultExt AS STRING,
DialogTitle AS STRING, ForeColor AS INTEGER, BackColor AS INTEGER, Flags AS INTEGER,
Cancel AS INTEGER)
   ' Set up error handling for option validation.
   ON LOCAL ERROR GOTO FileOpenError

   ' Set form caption.
   IF DialogTitle = "" THEN
```

```
    frmCmnDlg.Caption = "Open"
  ELSE
    frmCmnDlg.Caption = DialogTitle
  END IF

  ' Determine search pattern for file listbox.
  IF DefaultExt <> "" THEN
    frmCmnDlg.filOpenList.Pattern = DefaultExt
  ELSE
    frmCmnDlg.filOpenList.Pattern = "*.*"
  END IF

  ' Determine default path.
  IF PathName <> "" THEN
    ' Set drive and path for file-system controls.
    ' Set Directory listbox path.  If PathName is different
    ' than current path, PathChange event will be triggered
    ' which updates Drive listbox drive and File listbox path.
    frmCmnDlg.dirOpenList.Path = PathName
  END IF
  ' Display current path to the user.
  frmCmnDlg.lblOpenPath.Caption = frmCmnDlg.filOpenList.Path

  ' Determine default filename to display in edit field.
  IF FileName <> "" THEN
    frmCmnDlg.txtOpenFile.Text = UCASE$(FileName)
  ELSE
    frmCmnDlg.txtOpenFile.Text = frmCmnDlg.filOpenList.Pattern
  END IF

  ' Set default and cancel command buttons.
  frmCmnDlg.cmdOpenOK.Default = TRUE
  frmCmnDlg.cmdOpenCancel.Cancel = TRUE

  ' Size and position Open/Save container.
  frmCmnDlg.pctFileOpen.BorderStyle = 0
  frmCmnDlg.pctFileOpen.visible = TRUE

  ' Size and center dialog.
  frmCmnDlg.MOVE frmCmnDlg.Left, frmCmnDlg.Top, frmCmnDlg.pctFileOpen.Width + 2,
frmCmnDlg.pctFileOpen.Height + 2
  frmCmnDlg.MOVE (SCREEN.Width - frmCmnDlg.Width) \ 2, ((SCREEN.Height -
frmCmnDlg.Height) \ 2) - 2

  ' Set dialog colors.
```

```
    frmCmnDlg.ForeColor = ForeColor
    frmCmnDlg.BackColor = BackColor
    frmCmnDlg.pctFileOpen.ForeColor = ForeColor
    frmCmnDlg.pctFileOpen.BackColor = BackColor
    frmCmnDlg.lblOpenFile.ForeColor = ForeColor
    frmCmnDlg.lblOpenFile.BackColor = BackColor
    frmCmnDlg.txtOpenFile.ForeColor = ForeColor
    frmCmnDlg.txtOpenFile.BackColor = BackColor
    frmCmnDlg.lblOpenPath.ForeColor = ForeColor
    frmCmnDlg.lblOpenPath.BackColor = BackColor
    frmCmnDlg.filOpenList.ForeColor = ForeColor
    frmCmnDlg.filOpenList.BackColor = BackColor
    frmCmnDlg.drvOpenList.ForeColor = ForeColor
    frmCmnDlg.drvOpenList.BackColor = BackColor
    frmCmnDlg.dirOpenList.ForeColor = ForeColor
    frmCmnDlg.dirOpenList.BackColor = BackColor
    frmCmnDlg.cmdOpenOK.BackColor = BackColor
    frmCmnDlg.cmdOpenCancel.BackColor = BackColor

    ' Display dialog modally.
    frmCmnDlg.SHOW 1

    ' Determine if user canceled dialog.
    IF frmCmnDlg.cmdOpenCancel.Tag <> "FALSE" THEN
       Cancel = TRUE
    ' If not, return FileName and PathName.
    ELSE
       Cancel = FALSE
       FileName = frmCmnDlg.txtOpenFile.Text
       PathName = frmCmnDlg.filOpenList.Path
       frmCmnDlg.cmdOpenCancel.Tag = ""
    END IF

    ' Hide or unload dialog and return control to user's program.
    ' (Hide if user chose to preload form for performance.)
    IF LEFT$(frmCmnDlg.Tag, 1) = "H" THEN
       frmCmnDlg.pctFileOpen.visible = FALSE
       frmCmnDlg.HIDE
    ELSE
       UNLOAD frmCmnDlg
    END IF

    EXIT SUB

' Option error handling routine.
```

```
' Ignore errors here and let dialog's controls
' handle the errors.
FileOpenError:
   SELECT CASE ERR
   CASE 7:                              ' Out of memory.
      MSGBOX "Out of memory.  Can't load dialog.", 0, "FileOpen"
      Cancel = TRUE
      EXIT SUB
   CASE ELSE
      RESUME NEXT
   END SELECT
END SUB
```

```
' FileSave common dialog support routine
'
' Displays Save dialog which allows users to specify
' filename for subsequent file save operation.
' This procedure only provides the user interface and
' returns user input.  It does not actually carry out
' the corresponding action.
'
' Parameters:
'   FileName - returns the name (without path) of the
'         file for the save operation.  To supply
'         default filename in dialog, assign default
'         to FileName then pass it to this procedure.
'   PathName - returns the path (without filename) of
'         the file for the save operation.  To supply
'         default path in dialog, assign default to
'         PathName then pass it to this procedure.
'         Note, only pass a valid drive and path. Do
'         not include a filename or file pattern.
'   DefaultExt - sets the default search pattern for the
'         File Listbox.  Default pattern when DefaultExt
'         is null is "*.*".  To specify a different
'         search pattern (i.e. "*.BAS"), assign new
'         value to DefaultExt then pass it to this
'         procedure.
'   DialogTitle - sets the dialog title.  Default title
'         when DialogTitle is null is "Save As".  To
'         specify a different title (i.e. "Save My File"),
'         assign new value to DialogTitle then pass it to
'         this procedure.
'   ForeColor - sets the dialog foreground color.  Does not affect
'         SCREEN.ControlPanel color settings.
'   BackColor - sets the dialog background color.  Does not affect
'         SCREEN.ControlPanel color settings.
'   Flags - unused.  Use this to customize dialog action if needed.
'   Cancel - returns whether or not user pressed the dialog's Cancel
'         button.  True (-1) means the user cancelled the dialog.
'
SUB FileSave (FileName AS STRING, PathName AS STRING, DefaultExt AS STRING,
DialogTitle AS STRING, ForeColor AS INTEGER, BackColor AS INTEGER, Flags AS INTEGER,
Cancel AS INTEGER)
   ' Set up error handling for option validation.
   ON LOCAL ERROR GOTO FileSaveError

   ' Set form caption.
```

```
   IF DialogTitle = "" THEN
      frmCmnDlg.Caption = "Save As"
   ELSE
      frmCmnDlg.Caption = DialogTitle
   END IF
   frmCmnDlg.Tag = frmCmnDlg.Tag + "SAVE"          ' Set form tag for common unload
procedure.

   ' Determine search pattern for file listbox.
   IF DefaultExt <> "" THEN
      frmCmnDlg.filOpenList.Pattern = DefaultExt
   ELSE
      frmCmnDlg.filOpenList.Pattern = "*.*"
   END IF

   ' Determine default path.
   IF PathName <> "" THEN
      ' If the path ends with a backslash, remove it.
      IF RIGHT$(PathName, 1) = "\" THEN
         PathName = LEFT$(PathName, LEN(PathName) - 1)
      END IF
      ' Set drive and path for file-system controls.

      ' Set File listbox path.  If PathName is different
      ' than current path, PathChange event will be triggered
      ' which updates Drive listbox drive and Directory listbox path.
      frmCmnDlg.filOpenList.Path = PathName
   END IF
   ' Display current path to the user.
   frmCmnDlg.lblOpenPath.Caption = frmCmnDlg.filOpenList.Path

   ' Determine default filename to display in edit field.
   IF FileName <> "" THEN
      frmCmnDlg.txtOpenFile.Text = UCASE$(FileName)
   ELSE
      frmCmnDlg.txtOpenFile.Text = frmCmnDlg.filOpenList.Pattern
   END IF

   ' Set default and cancel command buttons.
   frmCmnDlg.cmdOpenOK.Default = TRUE
   frmCmnDlg.cmdOpenCancel.Cancel = TRUE

   ' Size and position Open/Save container.
   frmCmnDlg.pctFileOpen.BorderStyle = 0
   frmCmnDlg.pctFileOpen.visible = TRUE
```

```
   ' Size and center dialog.
   frmCmnDlg.MOVE frmCmnDlg.Left, frmCmnDlg.Top, frmCmnDlg.pctFileOpen.Width + 2,
frmCmnDlg.pctFileOpen.Height + 2
   frmCmnDlg.MOVE (SCREEN.Width - frmCmnDlg.Width) \ 2, ((SCREEN.Height -
frmCmnDlg.Height) \ 2) - 2

   ' Set dialog colors.
   frmCmnDlg.ForeColor = ForeColor
   frmCmnDlg.BackColor = BackColor
   frmCmnDlg.pctFileOpen.ForeColor = ForeColor
   frmCmnDlg.pctFileOpen.BackColor = BackColor
   frmCmnDlg.lblOpenFile.ForeColor = ForeColor
   frmCmnDlg.lblOpenFile.BackColor = BackColor
   frmCmnDlg.txtOpenFile.ForeColor = ForeColor
   frmCmnDlg.txtOpenFile.BackColor = BackColor
   frmCmnDlg.lblOpenPath.ForeColor = ForeColor
   frmCmnDlg.lblOpenPath.BackColor = BackColor
   frmCmnDlg.filOpenList.ForeColor = ForeColor
   frmCmnDlg.filOpenList.BackColor = BackColor
   frmCmnDlg.drvOpenList.ForeColor = ForeColor
   frmCmnDlg.drvOpenList.BackColor = BackColor
   frmCmnDlg.dirOpenList.ForeColor = ForeColor
   frmCmnDlg.dirOpenList.BackColor = BackColor
   frmCmnDlg.cmdOpenOK.BackColor = BackColor
   frmCmnDlg.cmdOpenCancel.BackColor = BackColor

   ' Display dialog modally.
   frmCmnDlg.SHOW 1

   ' Determine if user canceled dialog.
   IF frmCmnDlg.cmdOpenCancel.Tag <> "FALSE" THEN
      Cancel = TRUE
   ' If not, return FileName and PathName.
   ELSE
      Cancel = FALSE
      FileName = frmCmnDlg.txtOpenFile.Text
      PathName = frmCmnDlg.filOpenList.Path
      frmCmnDlg.cmdOpenCancel.Tag = ""
   END IF

   ' Hide or unload dialog and return control to user's program.
   ' (Hide if user chose to preload form for performance.)
   IF LEFT$(frmCmnDlg.Tag, 1) = "H" THEN
      frmCmnDlg.pctFileOpen.visible = FALSE
```

```
      frmCmnDlg.HIDE
      frmCmnDlg.Tag = "H"            ' Reset tag.
   ELSE
      UNLOAD frmCmnDlg
   END IF

   EXIT SUB

' Option error handling routine.
' Ignore errors here and let dialog's controls
' handle the errors.
FileSaveError:
   SELECT CASE ERR
   CASE 7:                           ' Out of memory.
      MSGBOX "Out of memory.  Can't load dialog.", 0, "FileSave"
      Cancel = TRUE
      EXIT SUB
   CASE ELSE
      RESUME NEXT
   END SELECT
END SUB
```

Appendix B

Computer Controlled Blood Sampling System User's Manual

C S

C S

B

Computer Controlled Blood Sampling System

User's Manual

The Computer Controlled Blood Sampling System (CCBSS) User's Manual should be used to set up and connect the several pieces of hardware. The Introduction gives some background information about why the system was developed and how it can be used. A section covering the details of using the CCBSS program follows the Hardware Setup. An outline of the procedures follows the section on using the application, along with some helpful suggestions.

Introduction

The CCBSS was developed to eliminate the need for a researcher to be present when blood samples are taken from a pig. This system will allow the researcher to set the appropriate sampling times, administer a test drug to a pig, and then do other work or leave for home while the system takes blood samples automatically. The situation requires that the pig(s) be confined to a very small pen to decrease the risk of having a catheter pulled out by the pig. The system was designed for two situations. The first situation uses three catheters in each pig, two venous and one arterial, with the ability to sample from two pigs. The second situation can sample from four pigs with one venous catheter in each pig. The system does not require that the maximum number of pigs be used in any given sampling cycle. The researcher can use the system for any number of pigs up to the maximum by entering the appropriate sampling times or lack of sampling times, indicating where no pig is present. A schematic of the system is shown in figure 1. Table 1 contains a list of all the hardware components that were used to develop the CCBSS.

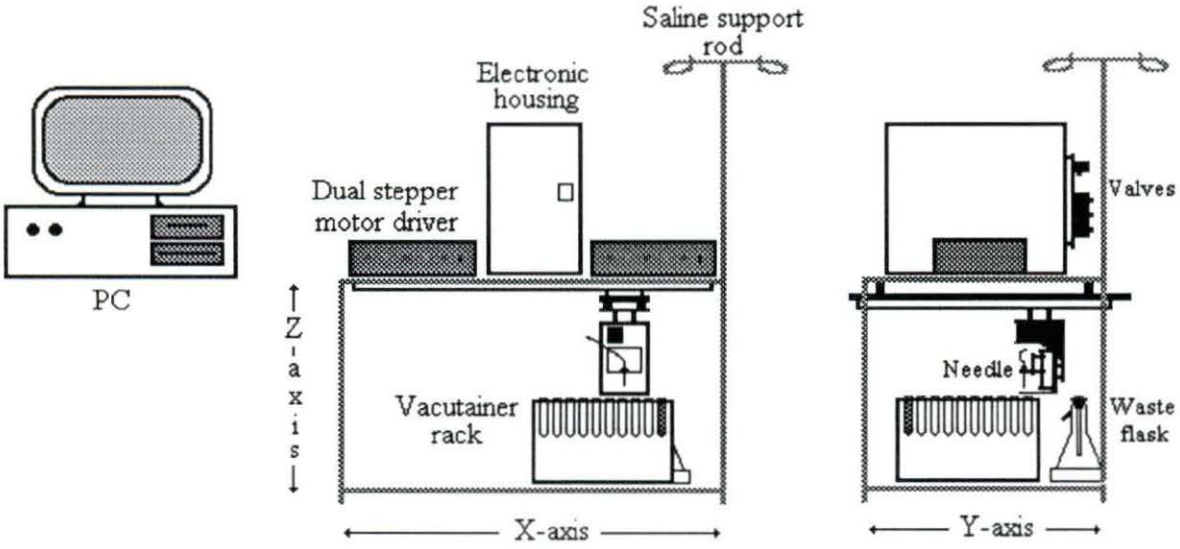


Figure 1 Schematic of the CCBSS

Hardware Setup

The stand should be placed on relatively level surface, the adjustable feet allow the user to level the stand further. If the XY and Z tables have been removed from the stand, the Z table and bracket should be mounted to the XY table frame, which should be remounted to the stand, perfectly square with the stand. Four bolts are used to hold the XY and Z tables upside down to the stand. The flask rack, a one inch thick plastic square with a circular hole, should be placed on the pins toward the rear of the aluminum bottom of the stand. The size of vacutainer to be used, large or medium diameter, needs to be determined. The proper

Table 1 Hardware List

IBM compatible personal computer (PC) (386)
with Monitor, Keyboard, and Mouse
Second parallel card
CyberResearch, Inc. Digital I/O board

2 Parallel cables
1 Ribbon cable (14 line)

Arrick Robotics:
XY-18 table
Z-2 table and BR-2 right angle mounting bracket for vertical mounting of the Z-2
2 MD-2 Dual Stepper Motor Systems
includes stepper motors and 3 motor cables

Stand approximately 29.5"W X 22"D X 29.75"H
w/ aluminum bottom and saline support rod

100 Tube Plexiglas vacutainer rack, large diameter vacutainers
100 Tube Plexiglas vacutainer rack, medium diameter vacutainers

Erlenmeyer flask w/ vacuum port and stopper assembly
Stopper assembly: large rubber stopper w/ hole,
syringe barrel,
silicon septum

Reciprocating pump (Gast, model DOA)
functions as compressor and vacuum pump

Electronic housing containing:
5 and 12 volt power supply w/ fan
CyberResearch, Inc. Relay board w/ 8 relays
Voltage drop board

Mounted on the rear:
1 six inlet Teflon PTFE solenoid manifold valve (Cole Parmer, CP# 01367 - 83)
2 three-way Teflon PTFE solenoid valves (Cole Parmer, CP# 01367 - 72)

Pressure cuff for saline bags
3 Liter and 1 Liter

Hypodermic needle (16 gauge) and mounting bracket

Hypodermic needle fork

Teflon tubing and fittings

Polyether catheters with Luer locks

IV line set

Silastic tubing

vacutainer rack can then be placed on the pins toward the front of the stand. When the system is set to use three catheters per pig, two pigs maximum, the vacutainer rack is divided as shown in figure 2, with pig #1 samples being toward the inside of the stand. Figure 3 shows how the vacutainers are divided between pigs when the system is set for sampling using one catheter per pig and a maximum of four pigs. The vacutainer rack can then be filled appropriately with vacutainers.

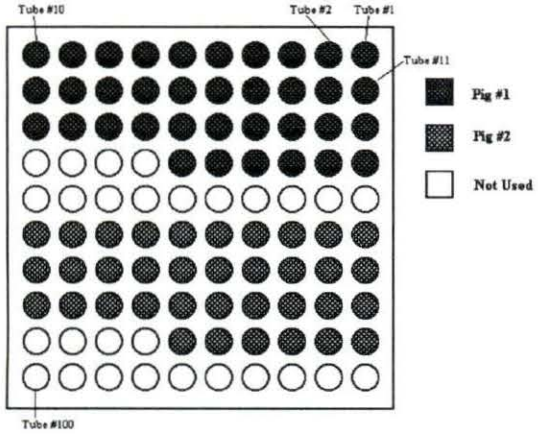


Figure 2 Tube placement using three catheters per pig

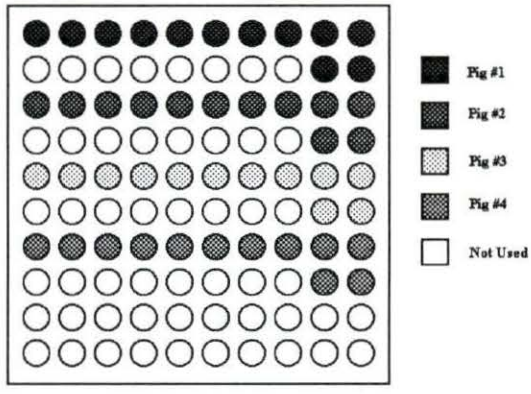


Figure 3 Tube placement using one catheter per pig

The electronic housing and both dual stepper motor drivers can be placed on top of the stand, the under side of the XY table. One of the two motor drivers' motor 1 port should be connected to the X axis stepper motor (left to right movement) using one of the motor cables. The motor 2 port of the same motor driver should be connected to the Z axis stepper motor (vertical movement) via a second motor cable. This motor driver can then be connected to the IBM compatible personal computer's (PC) original parallel port (left as viewed from the rear of the PC) via a parallel cable. The second dual stepper motor driver motor 1 port should be connected to the Y axis stepper motor (front to back

movement) using another motor cable. This motor driver can then be connected to the installed parallel port (right) using a second parallel cable. The relay board within the electronic housing, at the bottom, should be connected to the I/O board installed in the PC using the 14 line ribbon cable. The electronic housing and both dual stepper motor drivers power cords can then be connected and plugged into a wall outlet.

The valves are mounted on the rear of the electronic housing and should be connected with Teflon tubing and connectors as shown in figure 4. The IV line set, which connects to the saline bag should be connected to the top of the T-connector between the two three-way valves. Six catheters should be connected to the six inlets of the six inlet valve. The catheter

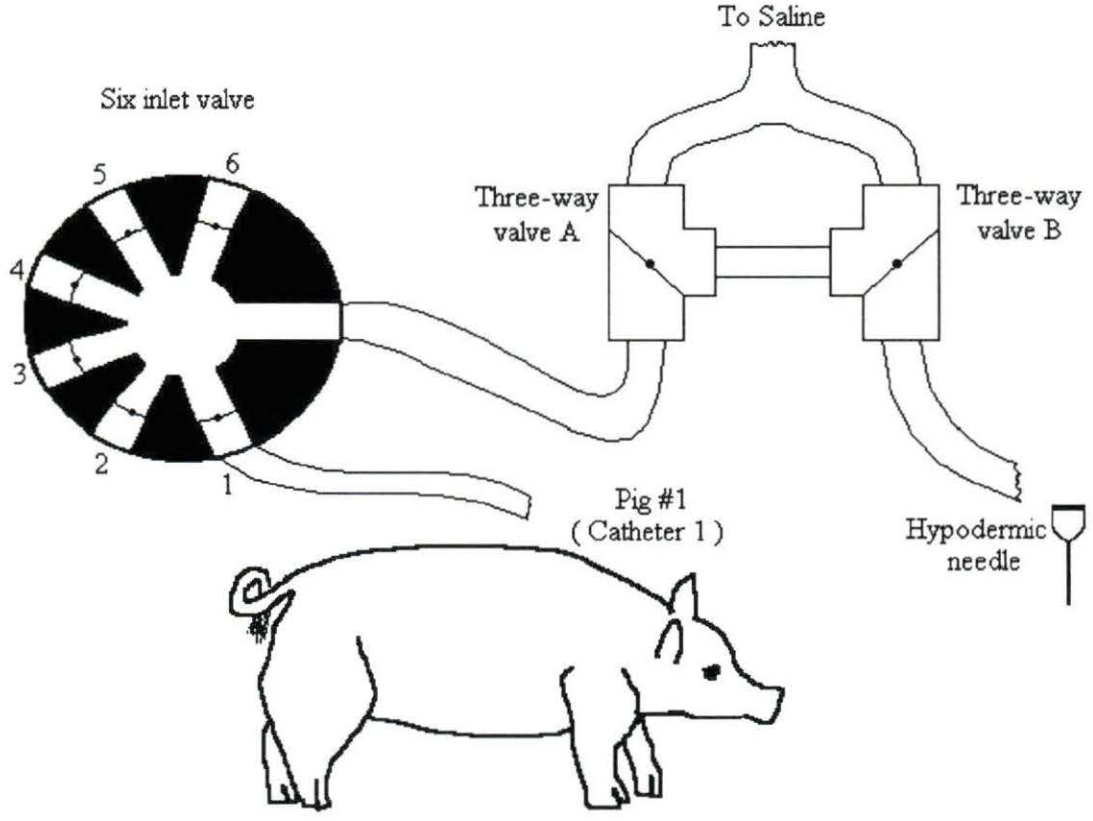


Figure 4 Valve connections

to lead to the hypodermic needle should be connected to three-way valve B. The hypodermic needle should be mounted to the Z table using the mounting bracket. The hypodermic needle fork should be mounted to the Z table frame if it is not presently mounted. The free end of the catheter leading to the hypodermic needle can then be connected to the hypodermic needle.

The saline support rod should be adjusted to a height to hold the three liter pressure cuff above the stand. A saline bag should be placed in the pressure cuff and hung on the saline bag hook. The pressure cuff and saline bag can then be hung on the saline support rod. The IV line set should be inserted into the saline bag's drain port. The air line of the pressure cuff can now be connected to the compressor of the reciprocating pump using a length of Silastic tubing, and the stop cock turned to allow inflation.

The waste flask stopper-tube assembly consists of a silicon septum, a syringe barrel, and a large rubber stopper with a hole. The silicon septum should be placed in the syringe barrel, toward the top, and the syringe barrel should be inserted through the large stopper hole. This stopper assembly should be placed in the waste flask, which can then be set in the flask rack. The hole in the large rubber stopper is not precisely centered, so the waste flask should be rotated to center the silicon septum below the hypodermic needle, when the X and Y axes are in the "home" position. The "home" position of a table is that position which results when the table is moved and triggers the limit switch and then the table is moved in the opposite direction only enough to deactivate the limit switch. A length of Silastic tubing should be connected from the waste flask vacuum port to the vacuum pump of the reciprocating pump.

The computer monitor should be connected to the PC using the monitor cable and the monitor power cord should also be connected to the PC. The computer mouse and keyboard should be connected to the appropriate ports, see the computer manuals if necessary. The

PC power cord should be connected to the PC and plugged into a wall outlet. A container should be partially filled with saline to finish the set up procedures. At this point, the system is ready for use.

Using the Application

Once all the setup procedures are completed, the IBM compatible PC should be turned on first. Then both stepper motor drivers, the switch on the front of the electronic housing, and the reciprocating pump can be turned on. The CCBSS application will appear showing the user interface screen. Initially, the screen shows four rows of twelve squares each, labeled pig #1 through pig #4. The sampling times are to be entered in these squares. In order to enter a time in a particular square, move the mouse pointer to that square and click on it. The cursor will appear in the square for the user to enter a sampling time. The Tab key can also be used to move from square to square while entering sampling times. The sampling times are entered as the number of minutes after sampling is started that the sample should be taken. If a sample is to be taken an hour and a half after sampling starts, for example, 90 should be entered in the appropriate square. Fractions of minutes may be included in expressing the sampling time, for instance, 24.5 could be entered.

The decision to use three catheters per pig, maximum of two pigs, or one catheter per pig, maximum of four pigs, needs to be made. There is a command button located toward the top of the screen at all times, which allows the user to switch between modes using one catheter or three catheters. When the application is set to sample using one catheter per pig, the initial setting, all four pigs are listed on the screen, and the button is labeled "2 pigs w / 3 catheters". By clicking the button, the user can switch to sample using three catheters in each pig. When in three catheters mode, only pig #1 and pig #2 are listed on the screen, and the

button label changes to "4 pigs w / 1 catheter". The sampling times of pig #3 and pig #4 are not cleared when the mode is changed to sample from two pigs, the application simply does not use this information.

A menu bar is located at the top of the screen with three selections: File, Run, and Options. Several menu options will be displayed in a drop down menu upon clicking one of the menu bar selections. Sample times can be saved, retrieved, and cleared using the menu options. Menu options also allow the user to prime the system, modify delay times, take samples, shut down the system, hide / display the clock, and verify that sample times meet system requirements. The File selection drop down menu contains the options: New, Open, Save, and Exit. The New option clears the current times for all pigs; all squares will be shown blank. A warning window appears before clearing to verify that the user wants to erase the current sampling times. The Save option stores the current sampling times in a data file on disk. A window appears to request the file name and path as well as the drive (hard or 3.5 inch floppy drive). The Open option allows the user to retrieve previously saved sampling times from disk. A window requesting the file, path, and drive will also appear for this option. The Open and Save options allow the user to enter a set of frequently used sampling times only once, save the times, and then retrieve them whenever the times are to be used again. The Exit option stops the application and returns the user to the system prompt. Before exiting the application, a warning window appears giving the user a last chance to save any sampling times that are currently entered.

The computer's system clock is also displayed on the opening screen, in the upper right corner. The clock can be hidden by clicking the menu bar selection Option and then clicking on the menu option Show Clock. When hidden, the clock can be displayed by following the same procedure used to hide. The drop down menu will display a dot next to the Show Clock menu option when the clock is being displayed. The system clock can be

changed by exiting the application, typing "time" at the system prompt, and then entering the new time in the same format as the current time, shown on the screen.

The second menu option in the Options drop down menu is the Check Times option, which allows the user to verify that the current sampling times meet the system requirements. When sampling using one catheter per pig, every sampling time must be at least 2 minutes apart from every other sampling time. This 2 minutes allows enough time to take a sample and be ready to take the next sample. When the system is set to take samples using three catheters per pig, sampling times are required to be at least 6 minutes apart. In this mode, three blood samples are taken during each sampling sequence, thus the minimum amount of time between sampling sequences is tripled. If the sampling times meet the system requirements, a message is displayed in a window indicating the approval of sampling times. The user should click on the OK button within the window once the message has been read. If the sampling times do not meet the system requirements, a message is displayed to the user in a window to indicate that times must be altered before sampling can begin. Included in the message is a suggestion as to which pigs and columns the user should examine to correct the problem.

When the system is set to sample using three catheters per pig, the sampling times for pig #3 and pig #4 are not displayed and are not used during sampling. But if sampling times have been entered for pig #3 and pig #4, the times must meet the requirement of being at least 6 minutes apart from all other times, even though they are not used. The best advice is to leave the times of pig #3 and pig #4 blank when sampling using three catheters.

The third option in the Options drop down menu, Show Wait Times, allows the user to modify the time delays for drawing waste, flushing back into the pig, and the pull and push times for cleaning out the six inlet valve. When the Show Wait Times option is selected, a table of delay times will appear below the rows of sampling times and a dot will appear next

to Show Wait Times menu option. If the table of delay times is currently shown on the user screen, selecting Show Wait Times will remove the table from the screen and remove the dot next to the menu option. The four delay times mentioned above can be set individually for each of the six catheters. The waste time delay is the time that the hypodermic needle remains in the waste flask at the beginning of sampling when blood and saline are pulled through the catheter. The flush time delay is the amount of time that saline is pushed back into the pig after the needle has been moved from a vacutainer to the waste flask. The pull time delay is the time in which blood is drawn into each of the catheters to agitate the six inlet valve and the push time delay is the time that saline is then pushed through each of the six catheters to force the blood back into the pig(s).

The catheter sizes used, the blood vessels (arteries or veins) used, and the pigs' blood pressure are all possible factors that may require the user to adjust the delay times. A small catheter will not allow as much fluid as a larger catheter to flow through in a given amount of time at a given pressure. This decrease in fluid flow requires the blood and saline to be drawn and flushed for a longer period of time for the same volume of fluid to pass through. When sampling from an artery or a pig with high blood pressure, less time is needed to draw blood, but more time is needed to push blood and saline back into the pig because of the increased pressure. The user should try catheters of various sizes in various sampling arrangements to become familiar with the time delay changes that may be required in situations that vary from the usual. However, when the delay times are lengthened, the Check Times routine will no longer be accurate. The user must check the time duration of an individual sample and make sure that an appropriate amount of time is allowed between samples, so that one sample procedure ends before the next begins.

The options in the Run drop down menu are Start Sampling, Prime, and Shut Down. The Start Sampling option, when clicked, executes the Check Times routine to verify that the

sampling times meet system requirements before sampling begins. When the sampling times meet system requirements and the user clicks the OK button, the time that sampling began is displayed on the screen above the current time. From this point on, the program waits for the next sampling time to occur. The sampling times cannot be changed after the start sampling option has been selected.

The second option in the Run drop down menu is Prime. This routine should be performed before any catheters are connected to the pigs. The Prime routine flushes saline through the six catheters connected to the six inlet valve to remove any air and fill the lines with saline. I suggest placing all six catheters in a large container of saline and then choosing the Prime option. At this point, the catheters can be connected to the pigs. Any unused catheters should remain in the saline container, the saline is used in a routine used to remove any residual blood from the six-inlet valve.

In the Run drop down menu, the third option is Shut Down. This routine should be performed when sampling is completed. The valves need to be flushed with sterile water to remove the saline from the valves. It is possible for the salt to precipitate out of the saline if the saline remains in the valves for an extended period of time. The salt can then damage the valve diaphragm causing the valve to leak and require a replacement. Before this option is chosen, the IV line set connected to the saline bag should be placed in a bag of sterile water, which is within the one liter pressure cuff, and the compressor air line should be changed from the three liter cuff to the one liter cuff. The catheters from the six inlets of the six inlet valve and the catheter to the hypodermic needle should be placed into a container to collect the waste sterile water. When the Shut Down option is selected, each of the seven catheters is flushed four times. All the valves are closed, the valves and lines should be free of saline and blood, and the PC, dual stepper motor drivers, reciprocating pump, and electronic housing may be turned off.

The CCBSS is capable of taking samples from two pigs with two catheters each, as well as the two modes of operation that the system was designed for, one catheter per pig and three catheters per pig. Sampling from two pigs, using two catheters in each pig, can be accomplished in either one catheter per pig mode or three catheters per pig mode. The most efficient method, saving time and vacutainers, is to use the one catheter per pig mode. Use pig #1 and pig #2 on the user screen to set the sampling times for catheter #1 and catheter #2, respectively, for one pig. Then use pig #3 and pig #4 user screen sampling times to sample from a second pig, catheter #1 and catheter #2, respectively. In most situations, it would be desirable that for each pig #1 sampling time, a pig #2 sampling time is two minutes later, and that for each pig #3 sampling time, a pig #4 sampling time is two minutes later than that time. In this way, the two samples from a single pig would be taken so that the second sample is taken immediately after the first. If connected properly, blood samples taken using pig #1 sampling times (pig 1, catheter 1) will be stored in the vacutainers for pig #1. Pig 1, catheter 2 samples will be taken in pig #2 vacutainers, pig 2, catheter 1 samples in pig #3 vacutainers, and pig 2, catheter 2 samples in pig #4 vacutainers.

Outline of Complete Sampling Cycle

I. Follow hardware setup procedures

II. Prime the six inlet catheters

III. Choose one or three catheters per pig

IV. Modify sampling wait time delays if necessary

V. Enter sampling times

VI. Check sampling times, correct if necessary

VII. Insert catheters in pigs, leave unused catheters in saline container

VIII. Start Sampling

IX. Inject drug when appropriate

X. Wait for sampling to complete

XI. Remove catheters from pigs and hypodermic needle and place in a waste container

XII. Connect sterile water in place of saline

XIII. Perform Shut Down procedure

Suggestions

Use some vacuum grease when connecting the catheters and fittings, this will help seal the connections and provide an airtight system.

Use a relatively large Erlenmeyer flask partially filled with saline to prime the system, this same container can then hold any unused catheters. All the catheters can be placed in the same container when sampling is finished and the shut down routine flushes the system with sterile water.

The silicon septum that is used in the waste flask stopper assembly should be replaced every few sampling cycles. This will prevent any vacuum leaks due to the deterioration of the silicon septum from repeated penetration by the hypodermic needle.