DESIGN OF A MICROCOMPUTER-BASED OPEN HEART SURGERY PATIENT MONITOR

by

Karen L. Brinkman

Thesis submitted to the Graduate Faculty
of the Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

APPROVED:

L. J. Arp, Chairman

D. G. Larsen

H. H. Robertshaw

October, 1985

Blacksburg, Virginia

DESIGN OF A MICROCOMPUTER-BASED OPEN HEART SURGERY PATIENT MONITOR

by

Karen L. Brinkman

(ABSTRACT)

A patient monitor device for use during open heart surgery has been designed and constructed. The device uses a VIC 20 microcomputer along with some additional circuitry to monitor 3 separate functions. The first patient variable monitored is the blood flow rate through the extracorporeal blood circuit during surgery. The device also continuously monitors and displays 6 separate temperatures. Finally, 3 individual timers are monitored and displayed with the device. Both the hardware and the software used in the design are fully described.

ACKNOWLEDGEMENTS

I would like to thank the following people for their help and support in this research:

Dr. Leon J. Arp, for serving as my committee chairman, and for spending countless hours teaching and helping me with this project.

Dr. Harry Robertshaw and Professor David Larsen for serving on my graduate committee.

My parents, , for their support, love, and encouragement.

Finally, my husband, , for his constant love, support, and patience throughout my college education.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Problem Overview

A microcomputer-based patient monitoring device has been developed for use during open heart surgery. The device has three separate functions: (1) monitor and display the blood flow rate through the patient, (2) monitor and display six different temperatures, and (3) display three individual timers.

The blood flow rate is found using a noninvasive technique which calculates the blood pump's output, then normalizes the output for the patient's size in two ways. One flowrate is found by dividing the blood pump output by the patient's weight, the other is found by dividing by the patient's total body surface area. Three types of blood pumps can be used with this device: (1) centrifugal, (2) roller, and (3) pulsatile. The centrifugal pump is a turbo-machine which pumps the blood using a rotor. The rotor transfers its rotational energy to the blood by centrifugal forces which move the blood from the center of the rotor to the outer casing. The roller pump has rollers on the end of arms attached at their other end to a rotating shaft. The rollers pinch tubing filled with blood against the outer casing. As the rollers move the blood is pushed along in the direction of rotation. The pulsatile pump consists of a

reciprocating piston which pushes a hydraulic fluid against a flexible diaphragm. The diaphragm expands into a spherical cavity filled with blood. As the diaphragm expands, the blood is squeezed out of the cavity. Two check valves, one above and below the cavity, move the blood along in only one direction.

Six different temperature measurements are needed in order to fully monitor the patient's skin and core temperature, the temperature entering and leaving the heat exchanger water bath, as well as the temperature of the blood entering and leaving the blood oxygenator.

Two of the three individual timers can be used to time the total perfusion time, the amount of time the blood is being oxygenated, or the cross-clamping of the aorta. Finally, the remaining timer can be used to obtain the time for each individual distal anastomosis. A distal anastomosis is a surgical connection between two blood vessels.

Survey of Present Devices

The idea for the patient monitor device was given to Dr. Leon Arp by Mr. Edward C. Berger, a retired perfusionist [1]. A perfusionist runs the blood pump and oxygenator during the open heart surgery. According to Mr. Berger, the existing devices used to monitor blood flow need more flexibility in the patient input data as well as in the output display.

With both the Sarns and the Cobe-Stoekert units, either the patient weight or the body surface area are input, then the perfusionist is stuck with that flow index, either ml/min/kg or ml/min/m^2. With the monitor designed in this thesis, just the patient height and weight are input, then the body surface area is calculated using the Dubois formula [2]. This simplifies the patient input data as well as adds the flexibility of having both flow indexes displayed continuously.

Most of the existing temperature monitors have only four inputs where one or two more are often needed. Also, many of these devices display only one temperature, chosen by a switch. With the device described in this thesis, all six temperatures are displayed continuously, eliminating the confusion of switching from one temperature to another.

The timing module described in this thesis is comparable with the available devices. The advantage of incorporating the timing module with the temperature and flow index modules is the elimination of three separate devices.

Principle of Problem Solution

A dedicated microcomputer is used to monitor and display the various parameters because it is a flexible and efficient system. The system can be used with any of the three types of blood pumps now available: the centrifugal, the roller, and the pulsatile pump. Also, the inputs can be in a variety

of units, such as kilograms or pounds, which eliminates the chance of conversion errors.

II. DESIGN OVERVIEW

The patient monitor device, as shown in Fig. 1, consists of a VIC 20 microcomputer and keyboard, a television set and an interface circuit board.

The patient information is input to the microprocessor through the keyboard. The input data includes the patient's name, height, and weight; the type of blood pump; the stroke volume if a pulsatile pump is used; the number of rollers, the tubing diameter and arc length if a roller pump is used; the labels for the six temperature probes and three timers; as well as the units for the height, weight and pump data.

Also input to the microcomputer, through the interface circuit board, are the six temperature values in either Farenheit or Celsius, selectable by a switch on the front panel, the tachometer information from the roller or pulsatile pump, and a voltage representing the flowrate through the centrifugal pump.

Software, consisting of both BASIC and assembly language programs, calculates the patient body surface area, the blood pump output and the normalized blood flow rate using consistent units. This information along with the patient's name, height, and weight, the six temperatures, the three timers, as well as the current time of day are then displayed on the television set. The program continues to display and monitor the parameters until the power is turned off, or the input parameters need to be changed.

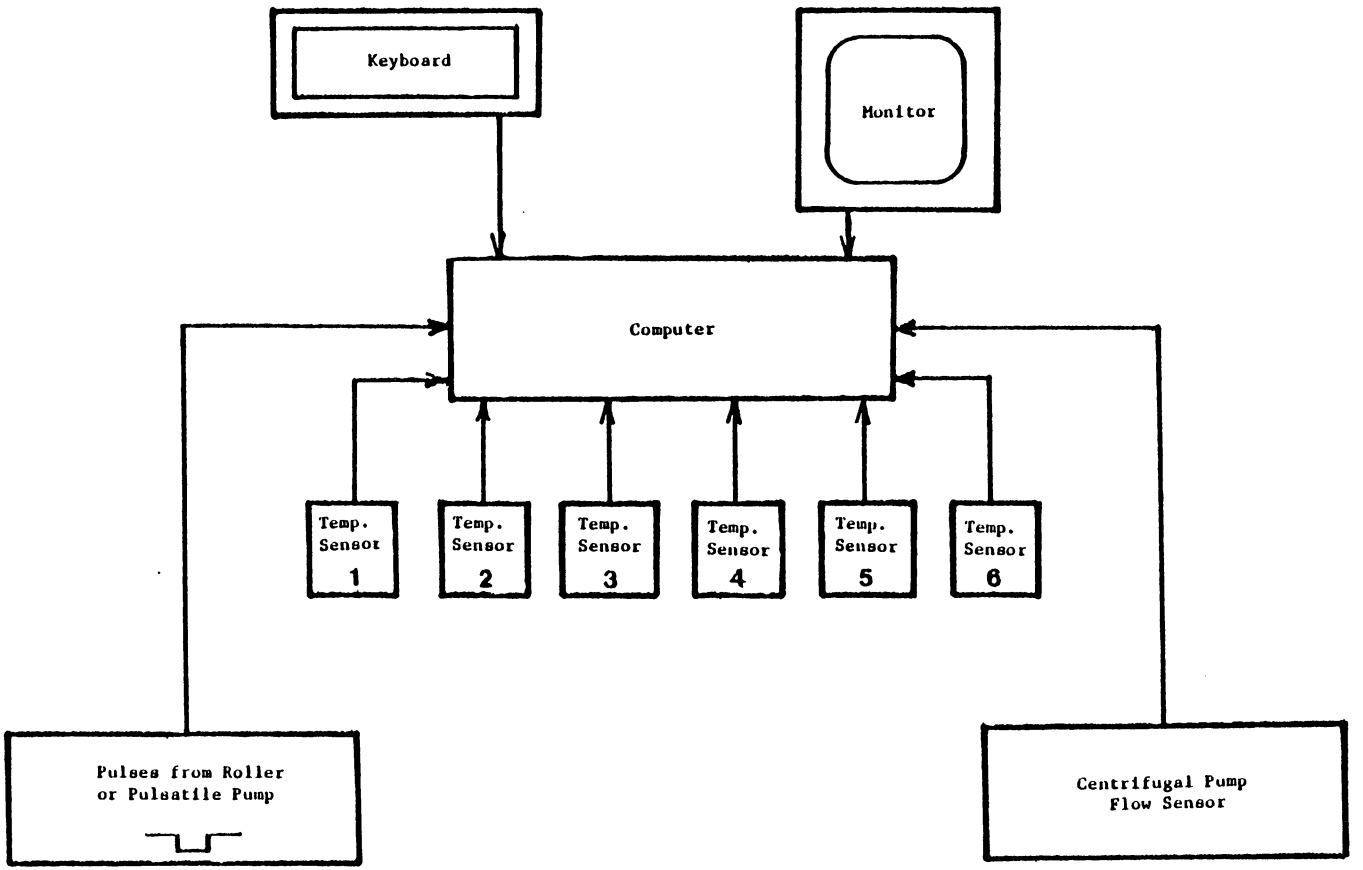


Figure 1. Open Heart Surgery Patient Monitor Device

III. HARDWARE DESIGN

Microcomputer Selection

For the design of the patient monitor a VIC 20 microcomputer was used. The design could have been implemented with dedicated circuitry, however, since the VIC 20 was available it was used because of its following advantages. The VIC 20 uses a 6502 microprocessor, contains a keyboard, a video interface chip (6560), as well as user callable machine language subroutines which input keyboard information and output characters to the video display.

Since the VIC 20 uses a 6502 microprocessor, it is compatible with the AIM 65 Advanced Interactive Microcomputer. This allows machine language subroutines to be programmed onto Electrically Programmable Read Only Memory (EPROM) chips, using the AIM 65. These programs can then be run on the VIC 20 using the memory expansion port to access the address, data and control lines from the microprocessor.

Getting information into the microprocessor is simple using the VIC 20 keyboard and input subroutines, as well as the BASIC input statement. Displaying information from the microcomputer is also easy, using the VIC 20 video interface chip and output subroutines, as well as the BASIC print statement [3,4].

As shown in the VIC 20 schematic in Appendix E, memory expansion is easily accomplished, since the VIC 20 hardware has already decoded the six most significant address lines. Therefore, only the ten least significant address lines need to be decoded when adding memory or interfacing devices. For the EPROM chips, these ten least significant address lines are inputs to the chip, eliminating any further decoding.

The VIC 20 also has an assembly language programming cartridge, the VIC MON®, which will assemble, disassemble and run assembly language programs. This helped with debugging assembly language subroutines, as well as enabled the disassembly of some of the VIC 20 assembly language subroutines.

Microcomputer Overview

The VIC 20 system along with the added memory and interface circuitry is outlined in Fig. 2. The Motorola 6502A microprocessor uses a 16-bit address bus, and an 8-bit data bus to communicate with memory and interface devices. The address bus contains the location that the data is to be sent to or received from. All devices that the microprocessor communicates with have an address, such as added memory or peripheral devices. The direction of data transfer is determined by the READ/$\overline{\text{WRITE}}$ signal from the 6502A. If the signal is high, then data is being read into the microprocessor. If

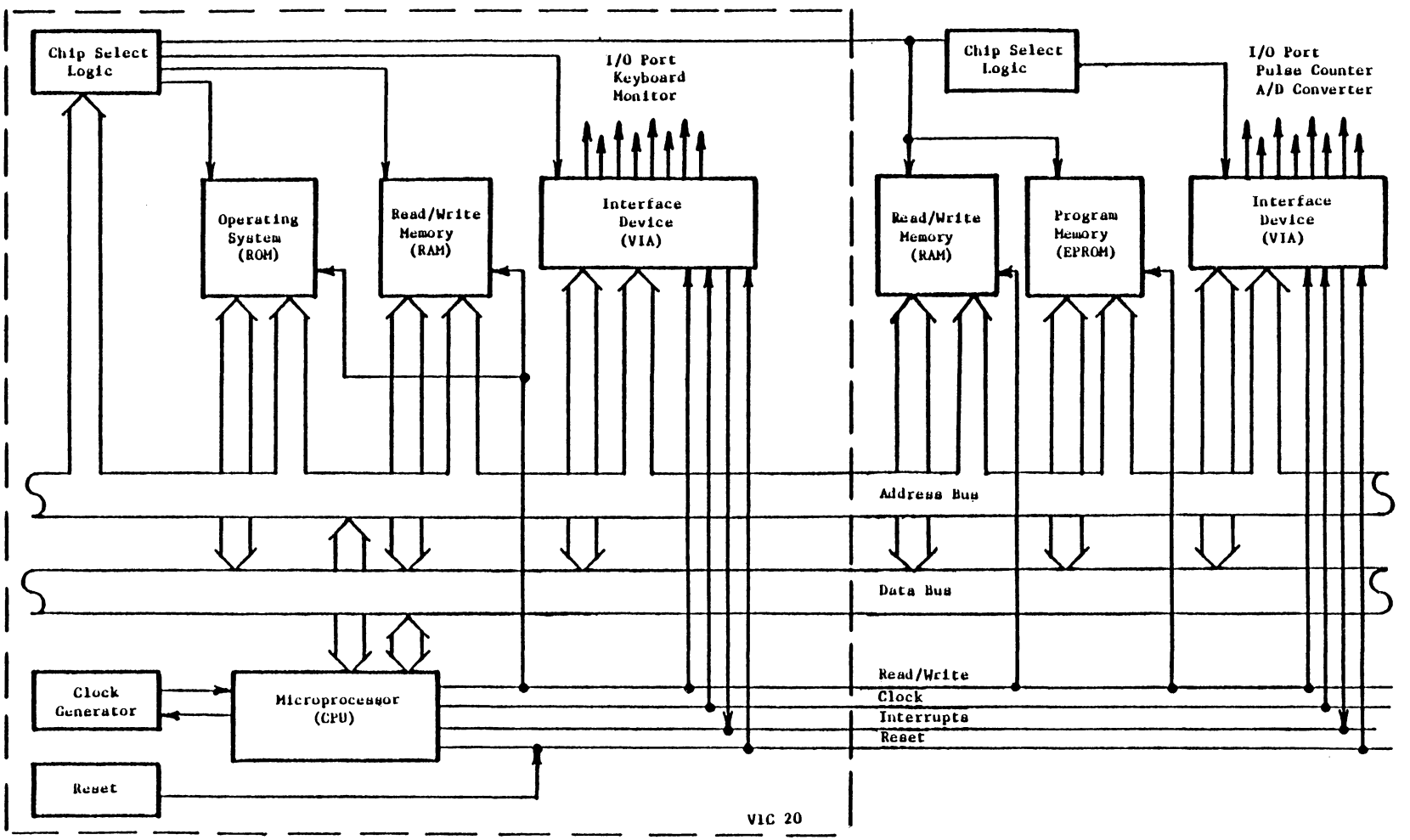


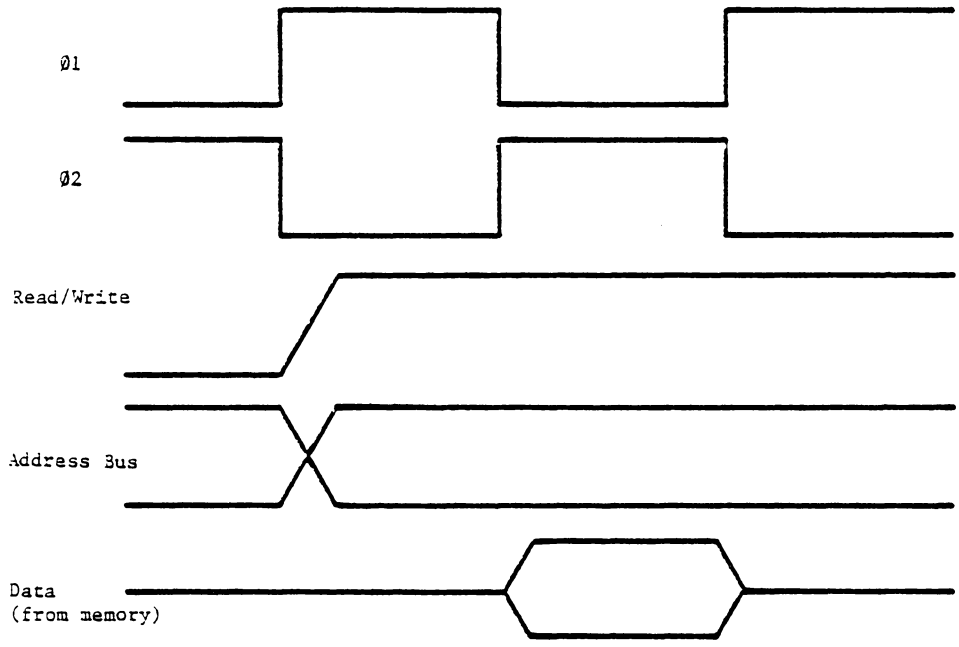
Figure 2. Microcomputer Block Diagram

the signal is low, then data is being written into memory or a peripheral device.

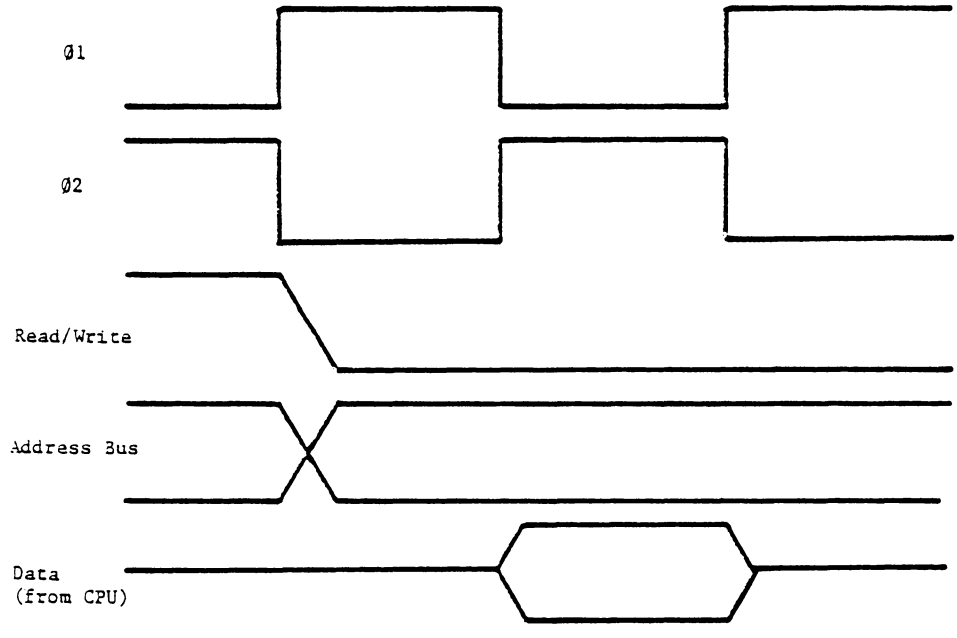
The timing which the microprocessor follows when sending or receiving data is shown in Fig. 3. As shown in the schematic, the microprocessor clocks, Ø1 and Ø2, control the timing of the data transfers. During the high portion of Ø1, the READ/$\overline{\text{WRITE}}$ line and the address lines stabilize. The data is then sent or received during the high portion of Ø2. This ensures that the correct data is sent to or received by the correct address [5].

To ensure correct operation at power-up, the VIC 20 contains reset hardware. This hardware delays the system operation until all of the supply voltages and clock cycles are stabilized.

Also available with this microprocessor is the ability to interrupt. An interrupt causes the microprocessor to finish executing the current instruction, service an interrupt routine, then return to the main program. For example, if an alarm condition occurred, the microprocessor could be interrupted, then the interrupt routine would stop execution of the main program and sound an alarm. Without the ability to interrupt, the microprocessor would have to periodically check to see if an alarm condition had occurred. This would increase the main program length, as well as, possibly, not recognize the alarm condition as quickly.



a) Memory Read Operation



b) Memory Write Operation

Figure 3. Timing Diagram for Data Transfer Between Memory and 6502A Microprocessor

VIC 20 System Review

The VIC 20 schematic is shown in Appendix E. As shown in the schematic, the VIC 20 contains reset hardware, a clock generator, random access memory, read only memory, interface devices, and chip select logic.

The reset hardware consists of a 555 timer chip which at power up pulses the reset line low. This causes the microprocessor to wait to start executing instructions until the supply voltages and clock signals have stabilized. Also, the reset signal places the interface devices in the input mode, and disables the timers, interrupts, and shift registers.

The clock generator for the microprocessor is a 14.31818 MHz quartz crystal oscillator. Its output is used to clock the 6560 video interface chip which controls the television screen input and output (I/O). The Ø1 and Ø2 clock signals are also generated by the oscillator using a 1.000 MHz signal.

There is 6K (6143 bytes) of random access memory available on the unexpanded VIC 20. This memory is used for temporary storage of variables used by the VIC 20 operating system, temporary storage of user programs, and storage of the characters and colors to be displayed on the screen.

The unexpanded VIC 20 contains 20K (20,479) of read only memory. This memory contains the KERNAL machine language subroutines, the BASIC interpreter program and the character

generator information. The addresses for both the read only memory and the random access memory are shown in Fig. 4. A hexadecimal number is noted by the dollar sign prefix, $. If there is no prefix, then a decimal number is indicated.

The interface devices contained in the unexpanded VIC 20 are a 6560 video interface device, and two 6522 versatile interface adapters. The 6560 can control the video display, sound generation, a light pen and a joystick. The 6522's perform the following I/O functions: the cassette deck, the user port, the keyboard input, the RESTORE key, the interrupt request (IRQ) timing for the real time clock and scanning the keyboard, the light pen control, and the joystick.

The chip select logic determines which chip is accessed, such as read only memory or random access memory or an interface device. This is accomplished on the VIC 20 by three 74LS138 integrated circuit chips. These chips are 3 to 8 line decoders which take an input address and output a low-going pulse on a separate line for each separate address. These pulses are sent to the different chips to enable them when their address is on the address bus.

Power-up and Initialization

When the power is turned on, after the reset line returns to the high state, the microprocessor delays 6 ns, then executes the initialization routine starting with the

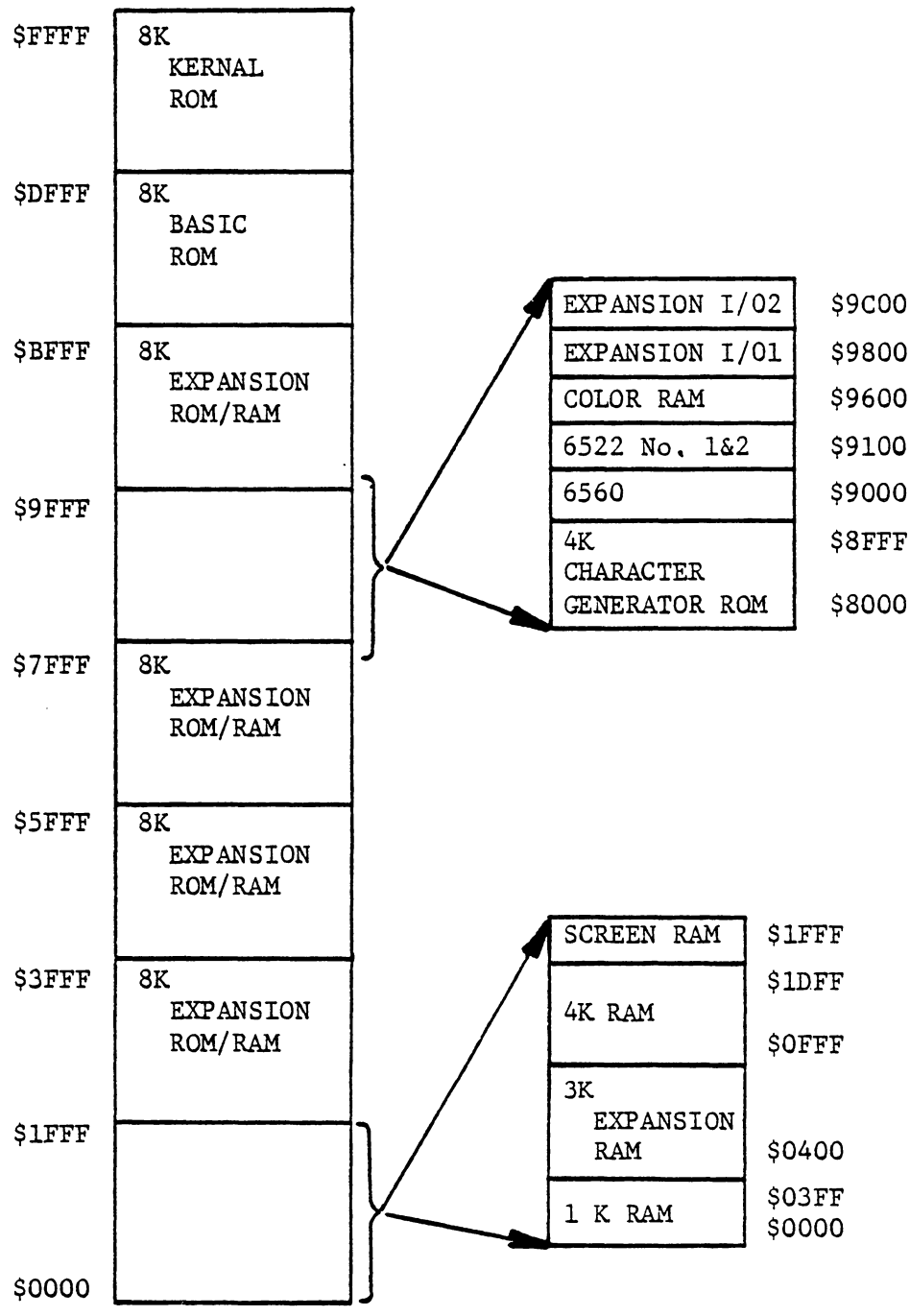


Figure 4. VIC 20 Memory Map

instruction located at address $FD22. This initialization routine is listed in Appendix C.

First, the interrupts are disabled until the initialization sequence has finished. The stack pointer is then set at $00FF. The stack pointer contains the address of the current value in the stack register. The stack register is a last-in-first-out (LIFO) register used to temporarily store variables.

Next, address $A000 is checked for the presence of ROM. If the auto-start ROM sequence:

Address -	$A004	Contents -	$41
	$A005		$30
	$A006		$C3
	$A007		$C2
	$A008		$CD

is found, then the program counter high byte is loaded with the contents of address $A001, and the program counter low byte is loaded with the contents of address $A000. The microprocessor then executes the instructions starting at the program counter address. If the auto-start sequence is not found then the normal initialization routine is continued, executing the instructions starting at address $FD2F.

The above instructions clear and test the RAM, initialize all of the indirect jump vectors, the I/O, and the video chip. Next, the interrupts are enabled since the

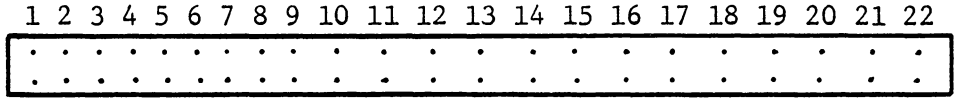
system is fully initialized. Finally, the processor jumps to the BASIC interpreter, turning control over to the user. For proper operation of the VIC 20 system, the programmer should include these instructions in his initialization routine if they are bypassed by the auto-start sequence.

Memory Expansion

The memory expansion connector located on the back of the VIC 20 is used to expand the microcomputer by adding memory or interface devices. As shown in Fig. 5, the female edge connector has 44-pins. Fig. 5 shows the location and the type of pins on the memory expansion connector. Table 1 gives brief explanations of the functions of each pin. A more detailed description follows.

Eight of the pins, CD0-CD7, are the data lines from the microprocessor. These lines are used to transfer data between the microprocessor and memory or interface devices. Fourteen of the pins, CA0-CA13, are the 14 least significant address lines from the microprocessor. These lines are used to direct the data to and from the correct chip. The 2 most significant address lines are not needed since the VIC 20 chip select logic has already decoded these lines for the block select signals.

The block select signals use 9 of the pins on the memory expansion connector. Four of these lines, $\overline{\text{BLK1}}$, $\overline{\text{BLK2}}$, $\overline{\text{BLK3}}$,



Pin #	Type
1	GND
2	CD0
3	CD1
4	CD2
5	CD3
6	CD4
7	CD5
8	CD6
9	CD7
10	$\overline{BLK1}$
11	$\overline{BLK2}$

Pin #	Type
12	$\overline{BLK3}$
13	$\overline{BLK5}$
14	$\overline{RAM1}$
15	$\overline{RAM2}$
16	$\overline{RAM3}$
17	VR/W
18	CR/W
19	$\overline{IRQ}$
20	NC
21	+5 V
22	GND

Pin #	Type
A	GND
B	CA0
C	CA1
D	CA2
E	CA3
F	CA4
H	CA5
J	CA6
K	CA7
L	CA8
M	CA9

Pin #	Type
N	CA10
P	CA11
R	CA12
S	CA13
T	$\overline{I/02}$
U	$\overline{I/03}$
V	SO2
W	$\overline{NMI}$
X	RESET
Y	NC
Z	GND

Figure 5. Location and Type of Pins on the Memory Expansion Connector

Table 1. Memory Expansion Connector Pin Functions

Type	Pin #	Description
GND	1	System Ground
(CD0-CD7)	2-9	Data bus lines 0-7
BLK1	10	8 K decoded RAM/ROM block 1, starting at $2000 (active low)
$\overline{\text{BLK2}}$	11	8 K decoded RAM/ROM block 2, starting at $4000 (active low)
$\overline{\text{BLK3}}$	12	8 K decoded RAM/ROM block 3, starting at $6000 (active low)
$\overline{\text{BLK5}}$	13	8 K decoded RAM/ROM block 5, starting at $A000 (active low)
$\overline{\text{RAM1}}$	14	1 K decoded RAM at $0400 (active low)
$\overline{\text{RAM2}}$	15	1 K decoded RAM at $0800 (active low)
$\overline{\text{RAM3}}$	16	1 K decoded RAM at $0C00 (active low)
VR/W	17	Read/Write line from buffer chip, (1=read, 0=write)
CR/W	18	Read/Write line from CPU, (1=read, 0=write)
$\overline{\text{IRQ}}$	19	6502A Interrupt request line (active low)
(NC)	20	No connection
+5V	21	+5 volt power
GND	22	System Ground
GND	A	System Ground
CA0-CA13	B-S	Address bus lines 0-13
I/O1	T	Decoded I/O block 1, starting at $9800
$\overline{\text{I/O2}}$	U	Decoded I/O block 2, starting at $9C00
SØ2	V	Phase 2 system clock
NMI	W	6502A non-maskable interrupt (active low)
$\overline{\text{RESET}}$	X	6502A reset line (active low)
(NC)	Y	No connection
GND	Z	System Ground

and $\overline{BLK5}$, go from high to low, respectively, when addresses, $2000 - $3FFF, $4000 - $5FFF, $6000 - $7FFF, and $A000 - $BFFF are accessed. These signals are used to enable a RAM or a ROM chip.

Three of the block select signals, $\overline{RAM1}$, $\overline{RAM2}$, and $\overline{RAM3}$, go from high to low, respectively, when addresses, $0400 - $07FF, $0800 - $0BFF, and $0C00 - $0FFF are accessed. These signals are used to enable a RAM chip.

The last two block select signals, $\overline{I/O1}$ and $\overline{I/O2}$, go from high to low, respectively, when addresses, $9800 - $9BFF and $9C00 - $9FFF are accessed. These signals are used to enable interface devices.

The remaining lines on the memory expansion port are used for power and for microprocessor control. The power lines consist of a system ground and a +5 supply voltage with a current rating of approximately 750 ma. The control lines consist of the Ø2 microprocessor clock, the reset line, two READ/$\overline{WRITE}$ lines, and two interrupt lines. The Ø2 clock controls the timing of data transfers between the microprocessor and memory or interface devices. The reset line when pulled low, halts the microprocessor operation until 6 clock cycles after the line returns high. The processor then executes the instruction found at address $FD22. The first READ/$\overline{WRITE}$ line, the CR/W, is directly connected to the microprocessor. This line is used for

memory and interface device timing. The second READ/$\overline{\text{WRITE}}$ line, the VR/W, is the buffered CR/W line. This signal is used by the video interface chip, the 6560. The two interrupt lines, $\overline{\text{IRQ}}$ and $\overline{\text{NMI}}$, when brought low, cause the microprocessor to finish executing the current instruction, then go to an interrupt service routine. The difference between the two lines is that the $\overline{\text{IRQ}}$ signal can be ignored by the processor by setting the interrupt disable bit.

Interrupt Processing

The VIC 20 uses the $\overline{\text{IRQ}}$ line to interrupt the microprocessor every 1/60 of a second. When the $\overline{\text{IRQ}}$ signal goes low, providing the interrupt disable bit is 0, the microprocessor transfers execution to the VIC 20 interrupt service routine located at address $FF72. This routine starts with a subroutine which saves the processor registers on the stack. Next program control jumps to the address pointed at by addresses $0314 (low byte) and $0315 (high byte). This address is set to $EABF by the VIC 20 initialization routine. This routine scans the keyboard for a pressed key and updates the system clock. If the programmer wants to bypass this routine, then the starting address of the new interrupt routine should be placed at addresses $0314 and $0315. The return from interrupt instruction (RTI) returns program control to the

instruction following the instruction that was being executed when the $\overline{IRQ}$ line went low.

VIC 20 Interface Circuitry

The interface circuitry used in this thesis to connect the VIC 20 with the added memory and interface devices is described in this section. The circuitry is based on the design used in Browning's thesis "Design of a Microcomputer-Based Microporous Membrane Process Controller" [6]. All the lines from the memory expansion port are buffered using four 74245 buffers as shown in Fig. 6. These chips can transfer data in one of two directions as chosen by the DIR input. Their outputs are 3-state with a typical propogation delay time from input to output of 8 ns. As shown in Fig. 6, 74245-3 is only enabled when either $\overline{BLK1}$, $\overline{BLK2}$, $\overline{BLK5}$, $\overline{RAM3}$, or $\overline{I/O2}$ is low and SØ2 is high. This makes sure that the data bus is only enabled when added memory or interface devices are accessed. This enabling is accomplished using a 7430 8-input nand gate in combination with a 7420 dual input nand gate as shown in Fig. 6.

EPROM Circuitry

Fig. 7 shows the EPROM circuitry used in this thesis, based on the design described in Browning's thesis [6]. The assembly language software for the patient monitor device is

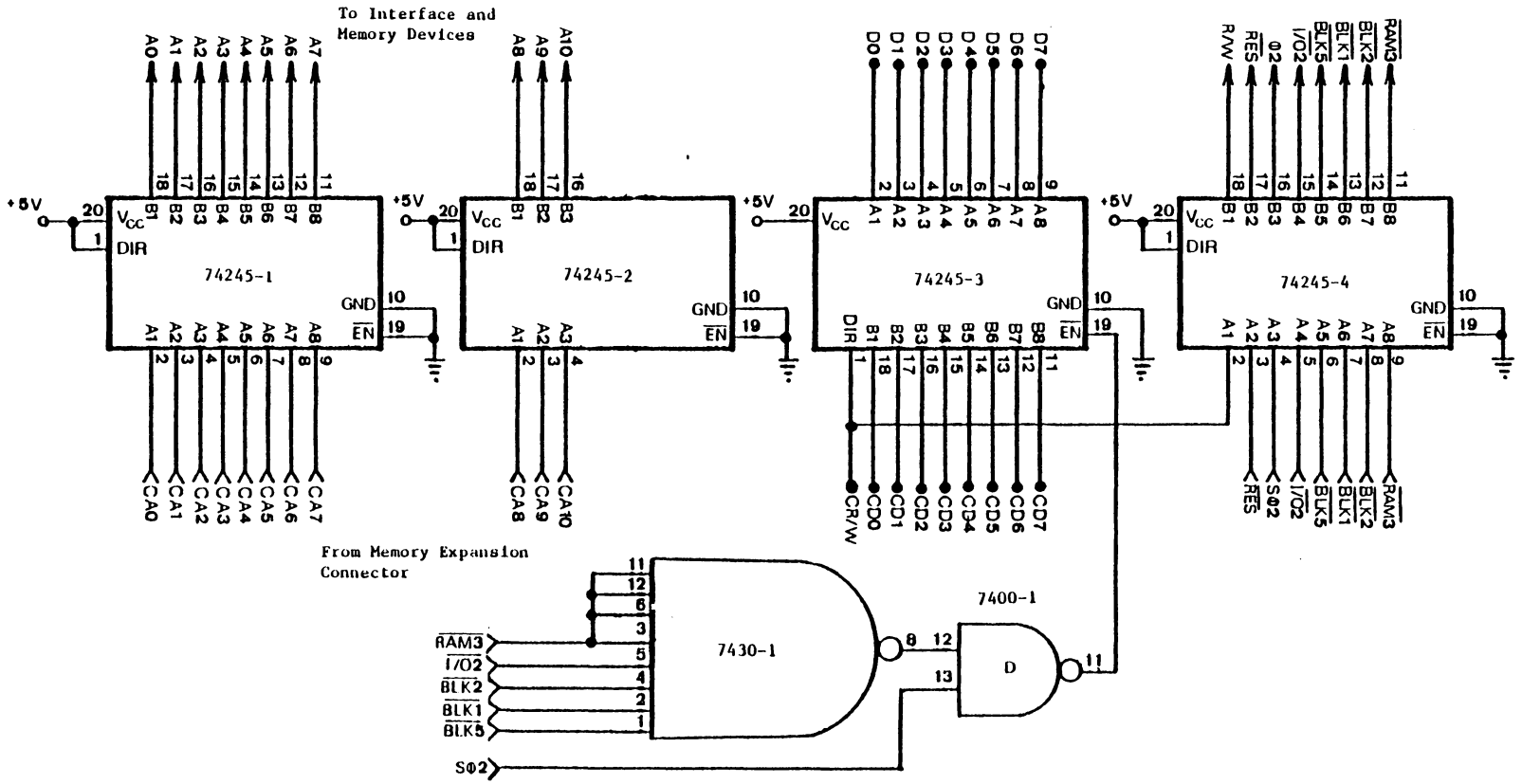


Figure 6. Buffer Circuitry

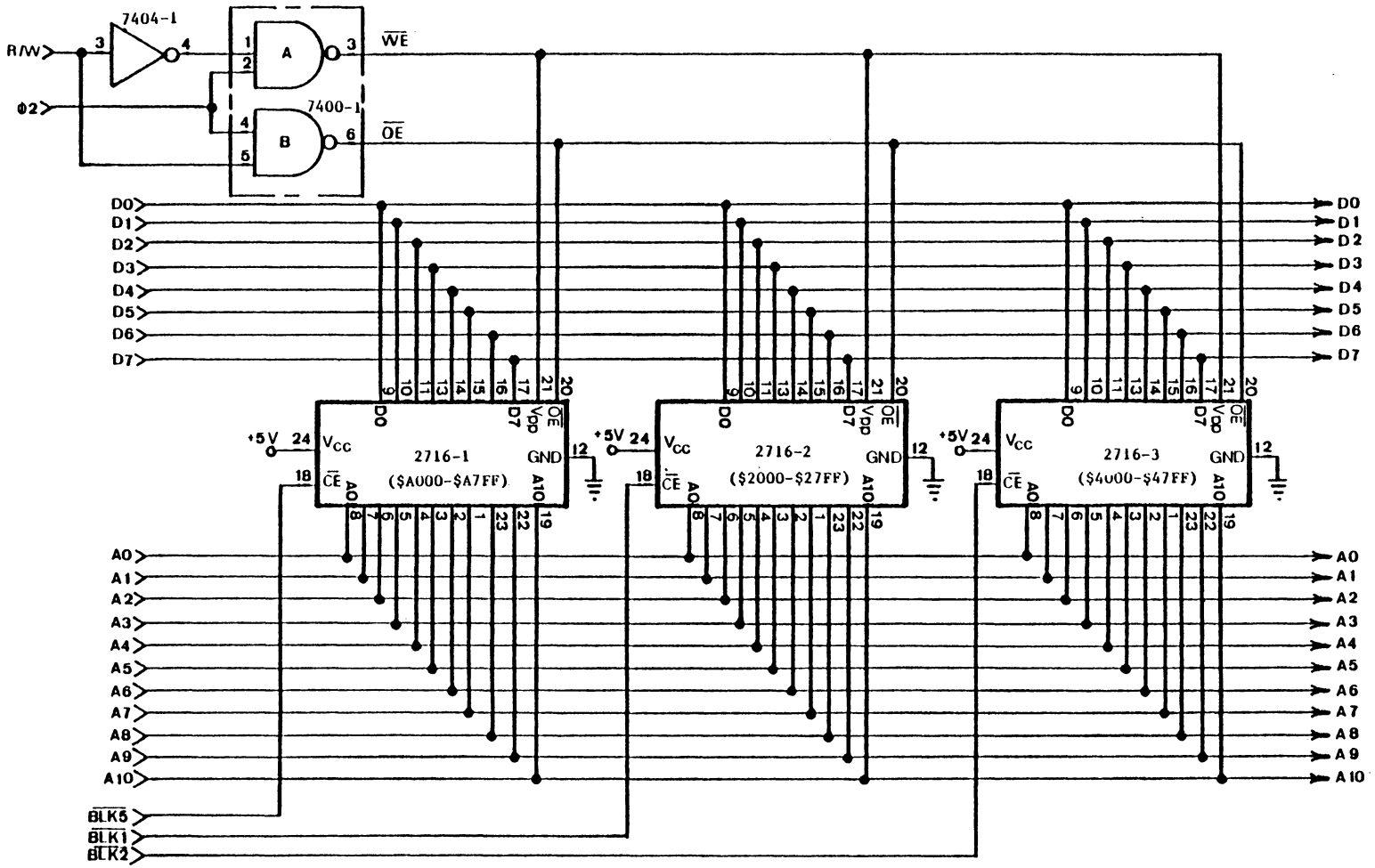


Figure 7. EPROM Circuitry

contained in these three memory chips. Each of the three 2716 EPROM chips are enabled using chip select pulses accessed directly at the memory expansion port. The chip select pulses, $\overline{\text{BLK1}}$, $\overline{\text{BLK2}}$, and $\overline{\text{BLK5}}$, each enable a separate EPROM chip. Each chip contains only 2K (2048) bytes of read only memory, while each of the chip select pulses enables 8K (8192) bytes of memory. Therefore, 6K (6144) of the addressable memory for each of the chip select pulses cannot be used with this circuitry. Table 2 shows the addresses used by the EPROM chips as well as the enabling range for the $\overline{\text{BLK}}$ signals.

Also shown in Fig. 7 is the circuitry required to achieve the correct timing of data transfers between the microprocessor and the EPROM chips. This is accomplished by logically gating the READ/$\overline{\text{WRITE}}$ signal with the Ø2 clock signal. This must be added since the EPROM chip does not have a clock input to synchronize data transfers with the microprocessor.

RAM Circuitry

In order to have enough memory to hold the BASIC program and the variables used by the assembly language program, random access memory (RAM) was added. The circuitry used is shown in Fig. 8. The 2114 RAM chips each hold 1K (1024) of addressable memory locations. However only 4 bits of data

Table 2. EPROM Address Range versus $\overline{\text{BLK}}$ Enabling Range

EPROM	EPROM Range	$\overline{\text{BLK}}$	$\overline{\text{BLK}}$ Range
2716-2	$2000-$27FF	$\overline{\text{BLK1}}$	$2000-$3FFF
2716-3	$4000-$47FF	$\overline{\text{BLK2}}$	$4000-$5FFF
2716-1	$A000-$A7FF	$\overline{\text{BLK5}}$	$A000-$BFFF

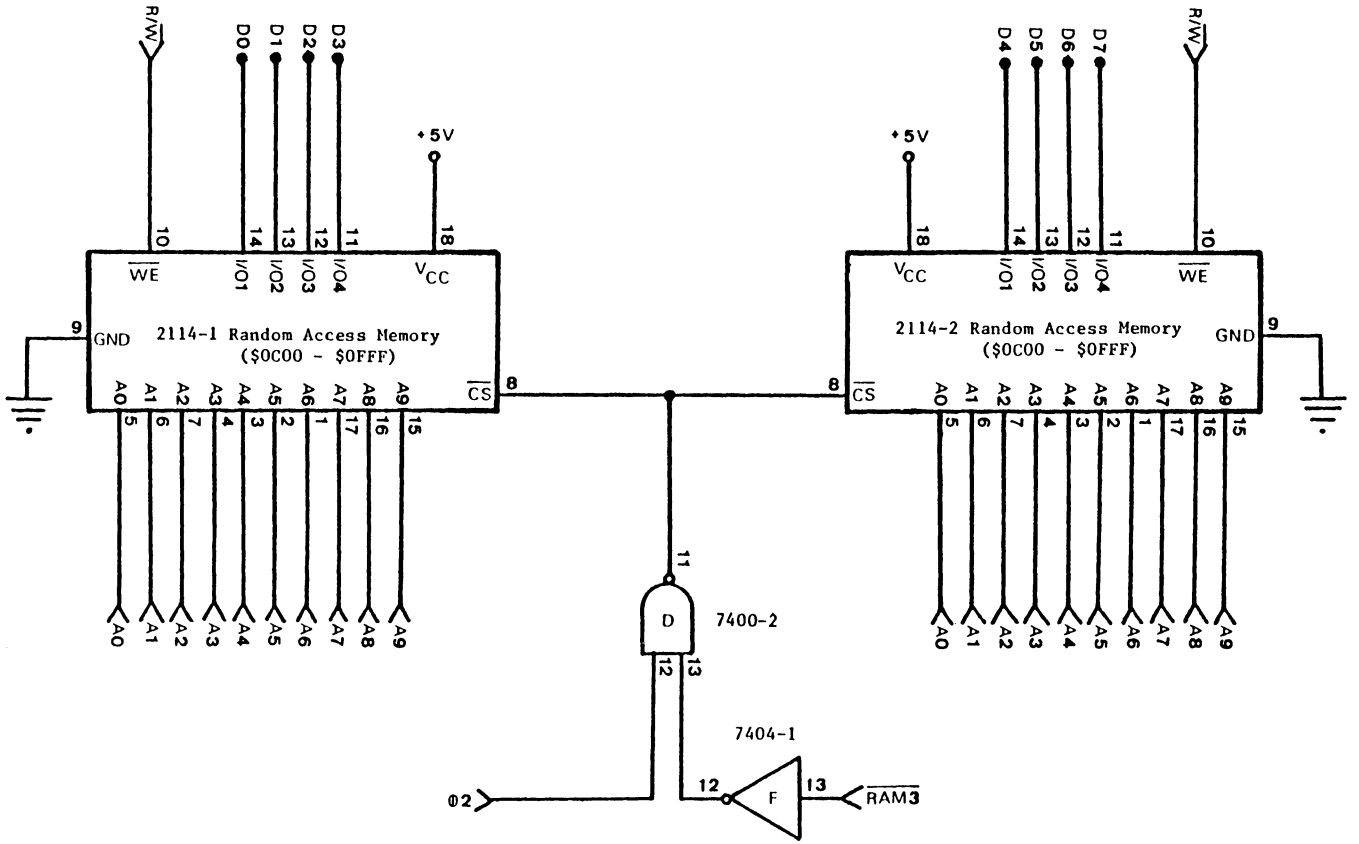


Figure 8. RAM Circuitry

are available in each chip. In order to hold the necessary 8 bits of data, two RAM chips are enabled with the same chip select pulse while the 4 least significant data bits are stored in 2114-1 and the 4 most significant data bits are stored in 2114-2.

Both the chip select logic and the correct data transfer timing is accomplished using the logic circuitry shown in Fig. 8. The $\overline{\text{RAM3}}$ chip select signal from the memory expansion port pulses low whenever a memory location at addresses $0C00 - $0FFF is accessed. This signal gated with the system clock, Ø2, enables the RAM chips at the correct time for data transfers with the desired address. The RAM is added at these locations since the BASIC interpreter program needs a continuous block of memory to run a BASIC program.

Interface Devices

In order to monitor six temperatures and the blood pump output, the circuitry shown in Fig. 9 was used. Two of the chips, the 6520-1 peripheral interface adapter, and the 6522-1 versatile interface adapter are used to interface the signals from the temperature sensors and the blood pump with the microcomputer.

The 6520 peripheral interface adapter has two bi-directional peripheral data ports which perform the following functions. Port A is configured as an input port to accept

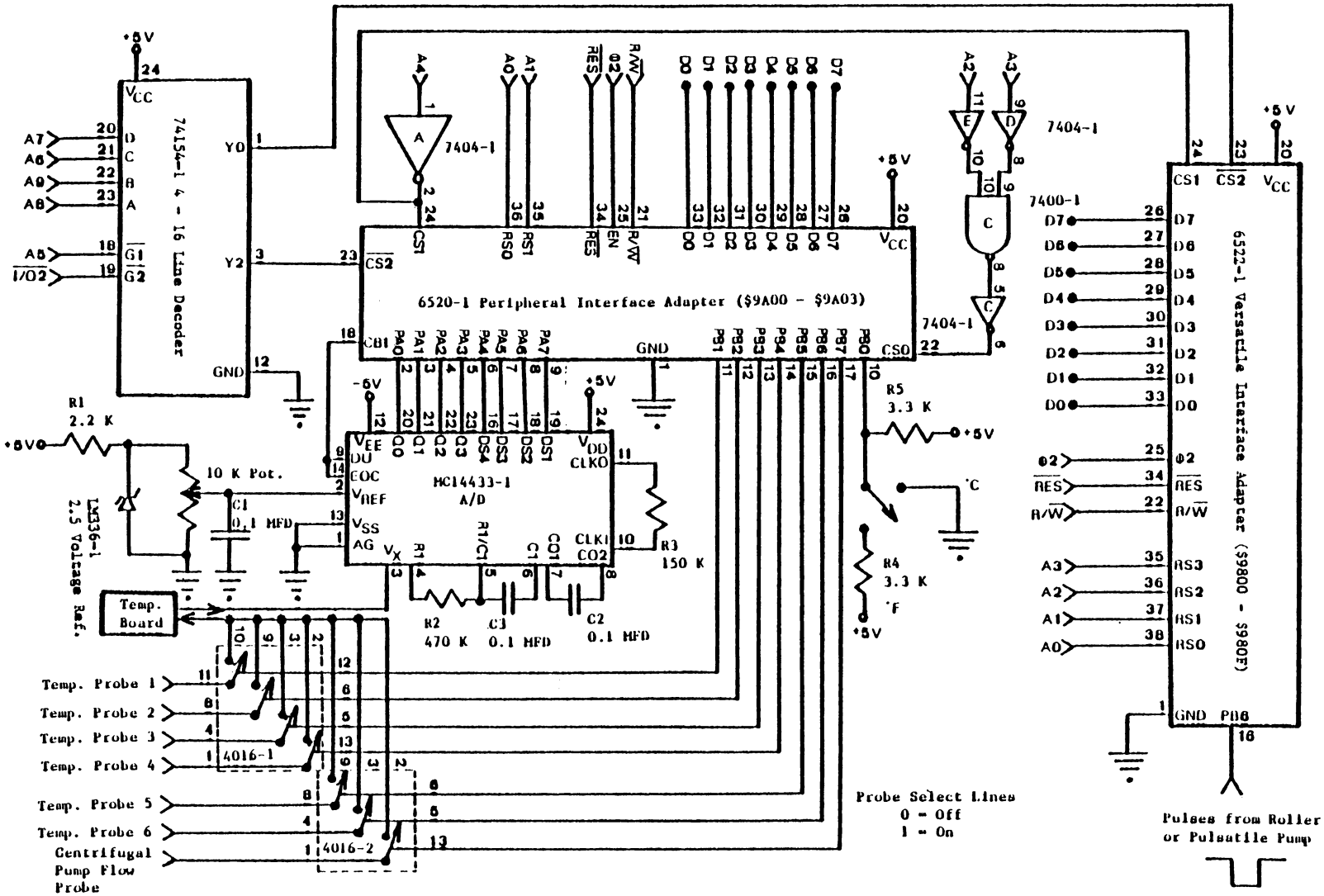


Figure 9. Interface Devices

the temperature information from the A/D converter. The 7 most significant data lines of port B are outputs that select which temperature probe is connected to the temperature board. The least significant data bit of port B is an input from a switch that selects the temperature degree unit, either Farenheit or Celsius.

In order to enable the 6520 for addresses $9A00 - $9A03, the chip select logic shown in Fig. 9 was used. The 74154-1 is a 4 - 16 line decoder with two enabling inputs. Inputting the $\overline{I/O2}$ signal from the memory expansion port along with the address lines, A5 through A9, to the 74154-1 decoder, produces a chip select pulse at pin 3 of the decoder for addresses $9A00 - $9A1F. The 6520-1, however, only uses addresses $9A00 - $9A03. Therefore, the address lines A2, A3 and A4 were decoded using the logic gates as shown in Fig. 9, to enable the 6520-1 only when addresses $9A00 - $9A03 are accessed.

The other interface device, the 6522-1, also has two bi-directional peripheral data ports in addition to two interval timers and a serial-parallel/parallel-serial shift register. For this project, the 6522-1's interval timer in connection with port B's input line 6 was used to count the low-going pulses from either the roller pump or the pulsatile pump. These pulses indicate either the revolutions per minute for

the roller pump, or the pulses per minute for the pulsatile pump.

The chip select logic for the 6522-1, as shown in Fig. 9, was accomplished using the 74154-1 decoder [6]. Using the $\overline{I/O2}$ signal from the memory expansion port along with address lines 5 through 9 as inputs to the decoder, gives a low going pulse at pin 1 of the decoder for addresses $9800 - $981F. In order to enable the 6522-1 only when addresses $9800-$980F are accessed, address line 4 is gated as shown in Fig. 9.

Temperature Detection

The temperature detection scheme used in this thesis is based on the design used in Mears' thesis "A Microcomputer-Based Temperature and Humidity Control System for Respiratory Therapy" [7].

Each temperature sensor is input to the temperature board which outputs a voltage signal proportional to the temperature. The voltages range from 0.000 to 1.999, where 0.010 V is equivalent to either 1.0 C or 1.0 F. The degree unit, C or F, is chosen by a switch connected to the temperature board.

The temperature probes are multiplexed, as shown in Fig. 9, using 6 analog switches. The integrated circuits, 4016-1 and 4016-2, each contain 4 analog switches which will

close when their respective enabling line goes high. These switches are opened and closed using the 6520-1, which is controlled by software.

The output of the 6 analog switches are input to the temperature board. The output of the temperature board is input to the analog-to-digital (A/D) converter, the MC14433-1. This is a dual slope A/D converter which outputs a number from 0 to 1999 for inputs ranging from 0.000 V to either 1.999 V or 199.9 mV [8]. The two voltage ranges are chosen by the input to the reference voltage, pin 2 on the MC14433-1. If the reference voltage is 2.000 V then the 1.999 V range is chosen. If the reference voltage is 200.0 mV then the 199.9 mV range is used. In this design, a reference voltage of 2.000 V is provided using a LM336 2.5 Voltage reference connected with a 10K trim potentiometer [9]. The accuracy of the A/D is $\pm$0.05% of reading +1 count.

The MC14433-1 makes up to 25 conversions per second, depending upon the resistor value connected to pins 10 and 11. This resistor changes the A/D converter's internal clock frequency. The conversion rate is then equal to the clock frequency divided by 16,400. In this design, as shown in Fig. 9, the A/D makes roughly 8 conversions/sec due to the 150 K resistor, R3.

The A/D converter's data outputs are connected to port A of the peripheral interface adapter, the 6520-1. These out-

puts are DS1 through DS4 and Q0 through Q3. The digit select lines, DS1 through DS4, pulse high when the most significant digit, DS1, through the least significant digit, DS4 has data on the data lines, Q0 - Q3. These data lines contain binary coded decimal (BCD) data, where Q0 is the least significant bit and Q3 is the most significant bit. Since both the data and digit select lines are read at the same time by the microprocessor through port A, the correct data is read for the desired digit. The MC14433 outputs a high going pulse at pin 14, to signal the end-of-conversion (EOC). This signal is connected to CB1 of 6520-1 which sets a flag in an internal register of the 6520-1 when an EOC pulse occurs. This flag is then monitored by software.

The MC14433 A/D converter works fine when the capacitors, C2 and C3, are mylar capacitors. If the capacitors are not mylar, then the A/D does not track the voltages correctly.

Pump Output Detection

In order to calculate the pump output both the 6520-1 and the 6522-1 are used as shown in Fig. 9. For the centrifugal blood pump, a voltage signal representing the flow through the pump is sent to the A/D converter (MC14433-1) through a switch in the 4016-2. This voltage is in the range of 0.000 V to 1.999 V which converts to 0. to 1999. ml/min.

Both the roller pump and the pulsatile pump have pick-up devices which produce 100 low-going pulses per revolution of the roller and 1 low-going pulse per pulse of the diaphragm. These pulses are input to line 6 of port B on the 6522-1. The 6522-1 interval timer counts these pulses, then the pump output is calculated using software.

Power Supply

The circuitry for this design is powered by the VIC 20 power supply and a commercially available negative power supply. A common ground between the circuitry, the two power supplies and the VIC 20 memory expansion port is connected. In order to avoid ground loops, each chip's ground is connected to the same point on the circuit board. When the grounds are not connected in this way, each chip draws a small amount of current at the ground input. This causes each ground to be at a slightly different potential causing circuit operation problems.

IV. SOFTWARE DESIGN

Program Languages

The software used in this design consists of both BASIC and assembly language programs which monitor and display the various parameters. The BASIC program contains the main program loop, the initial input section, the calculation of pump output and blood flow rate, as well as some of the display routines. These sections were written in BASIC since it is a higher level language, which greatly simplifies the programming.

The assembly language software consists of a power up initialization routine, an interrupt servicing routine, as well as display and monitor subroutines. The power up initialization routine was written in assembly language since the only way to bypass the VIC 20 initialization routine is to store the signature bytes, as described in the VIC 20 Review section, on an EPROM starting at the hexadecimal address A000. Similarly, in order to bypass the VIC 20 interrupt service routine, a machine code program must be used. The display and monitor subroutines were written in assembly language in order to operate at the fastest speed possible. Both the BASIC and assembly language programs are listed in Appendix A.

In the software discussion, the following prefixes indicate the base of the number. The dollar sign, $, indicates that a hexadecimal number follows. The per cent sign, %, indicates that a binary number follows. No prefix indicates a decimal number.

Initialization

When the VIC 20 is turned on it follows the initialization routine shown in Appendix C. The routine checks for an auto-start sequence as explained in the VIC 20 Review section. When the signature bytes are found at addresses $A004 through $A008, the VIC 20 routine jumps to the user's initialization routine at the address indicated by $A000 (low byte) and $A001 (high byte). In this design the initialization routine begins at $A009.

The initialization routine flowchart is shown in Fig. 10. First, the peripheral interface devices are configured such that the temperature and centrifugal flow probes are not connected to the A/D converter, and the pulse counter is disabled. In order to configure the PIA's, the ports must be made either outputs or inputs.

Setting the peripheral port pins to be outputs or inputs is accomplished by writing either a 1 or a 0 into the appropriate bit in the Data Direction Register (DDR). For example, if bit 7 of a port is to be an input while bits 0

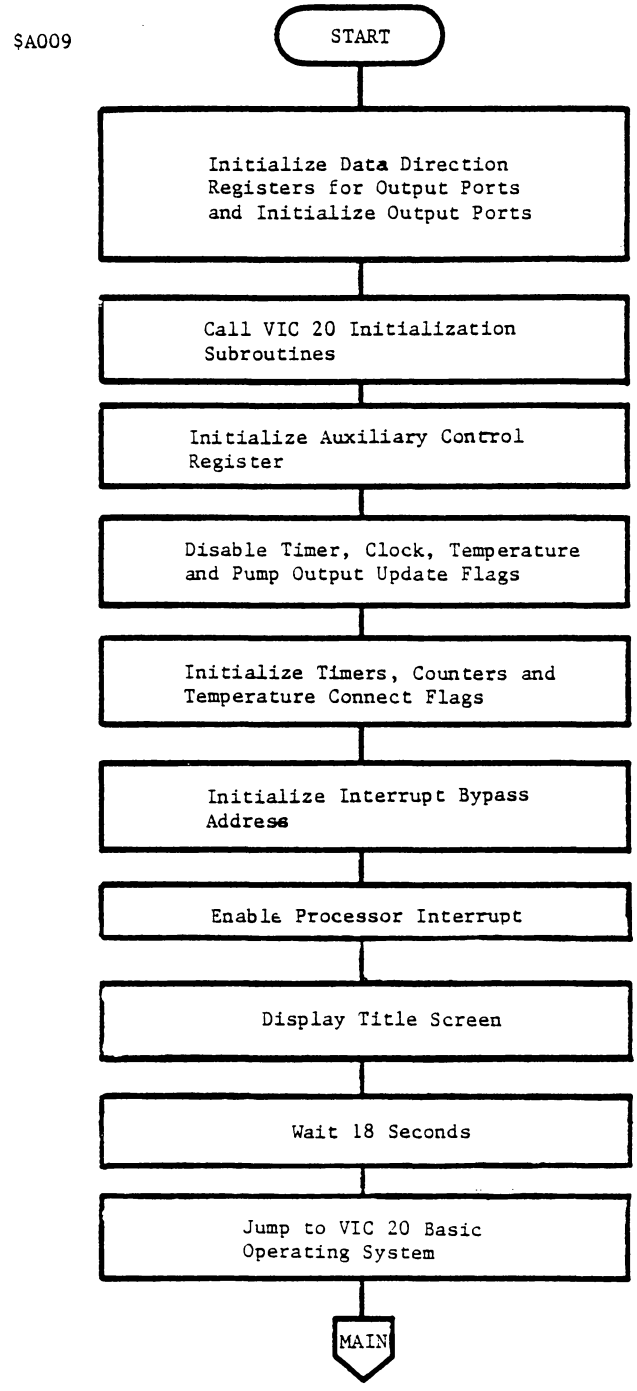


Figure 10. Initialization Routine Flowchart

through 6 are to be outputs, then %01111111 is stored in the DDR. The reset pulse at power up causes all the ports to be inputs, therefore only the output ports need to be initialized.

Once the output ports have been configured, the initial state of the pins is determined simply by writing a value to the output data register. The 6520 peripheral interface uses the same address for both the data direction register and the output register. When bit 2 in the control register is a 0, the data direction register is being accessed. When bit 2 in the control register is a 1, the output data register is being accessed. Therefore, after the data direction has been set, the data output register must be accessed to set the initial output values. These outputs are set to 0, which disconnects the probes from the A/D converter.

Once the temperature and centrifugal probes have been disabled, the routine calls subroutines which finish initializing the VIC 20 for operation. These routines clear and check the random access memory (RAM), set the VIC 20 vectors, and initialize the VIC 20 I/O and screen. Next the 6522 Versatile Interface Adapter (VIA) is configured.

In order to initialize the VIA, only the Auxiliary Control Register which controls the timers' modes of operation, and the Peripheral Control Register which determines the conditions for an interrupt to occur, need to be set.

The data direction register has already been set as inputs by the reset pulse and there are no outputs to initialize.

Following the initialization of the VIA, all of the flags are disabled and the variables are initialized. Then the interrupt by-pass address is specified and interrupts allowed. Next the title screen is displayed for 18 seconds, then the routine jumps to the BASIC operating system.

In order to use part of the RAM for storage of machine language variables, the top of BASIC memory is set below the highest RAM address. Unfortunately, this cannot be set in the initialization sequence since the BASIC operating system resets it to the highest RAM address. Therefore, the top of BASIC memory is set by the user after the initialization sequence, before the BASIC program is loaded. After the title screen goes blank, the user presses the CTRL and 2 keys simultaneously, then types in:

poke56,29:poke55,196:poke52,29:poke51,196

then hits the RETURN key. The user then fully rewinds the tape, then presses the SHIFT and RUN keys simultaneously.

Input Section

The BASIC input section is flowcharted in Fig. 11. The first section inputs all of the patient information: the patient's name; height, in either cm or in; and weight, in either kg or lb. Throughout the inputs, the user is asked if

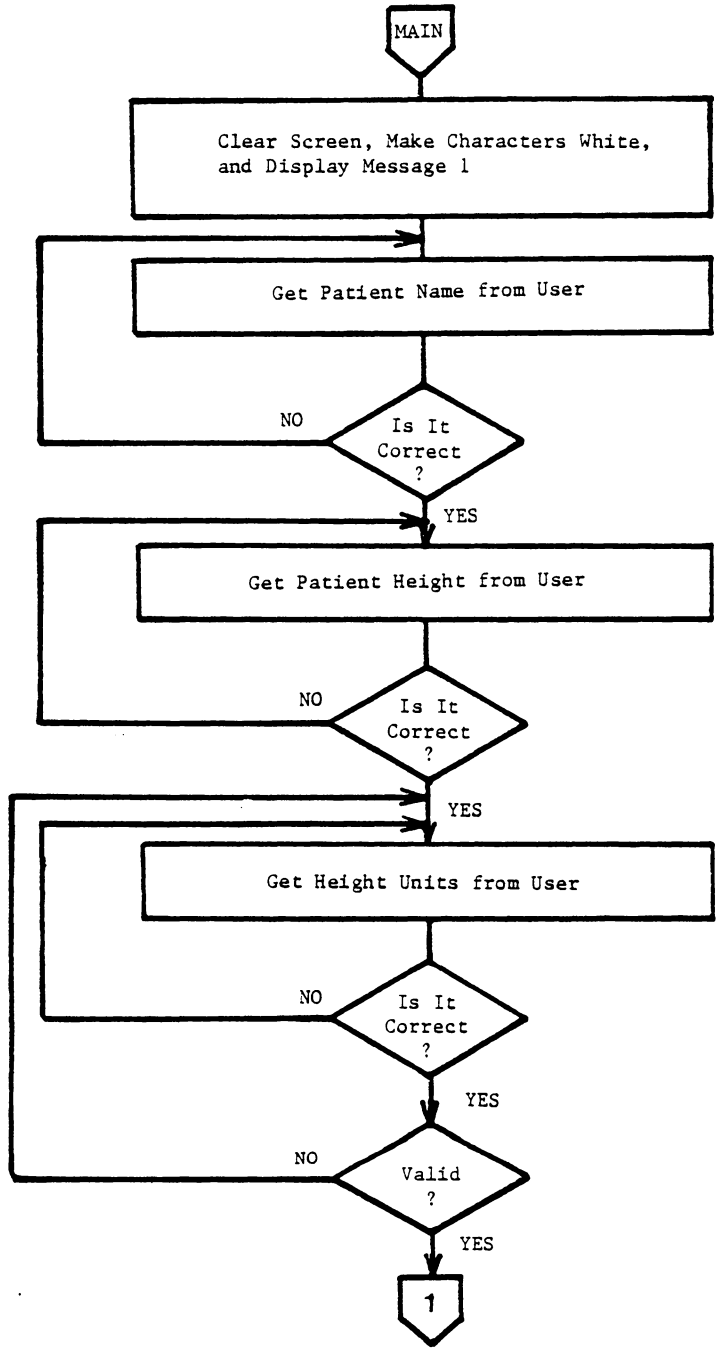


Figure 11 a. Input Section Flowchart

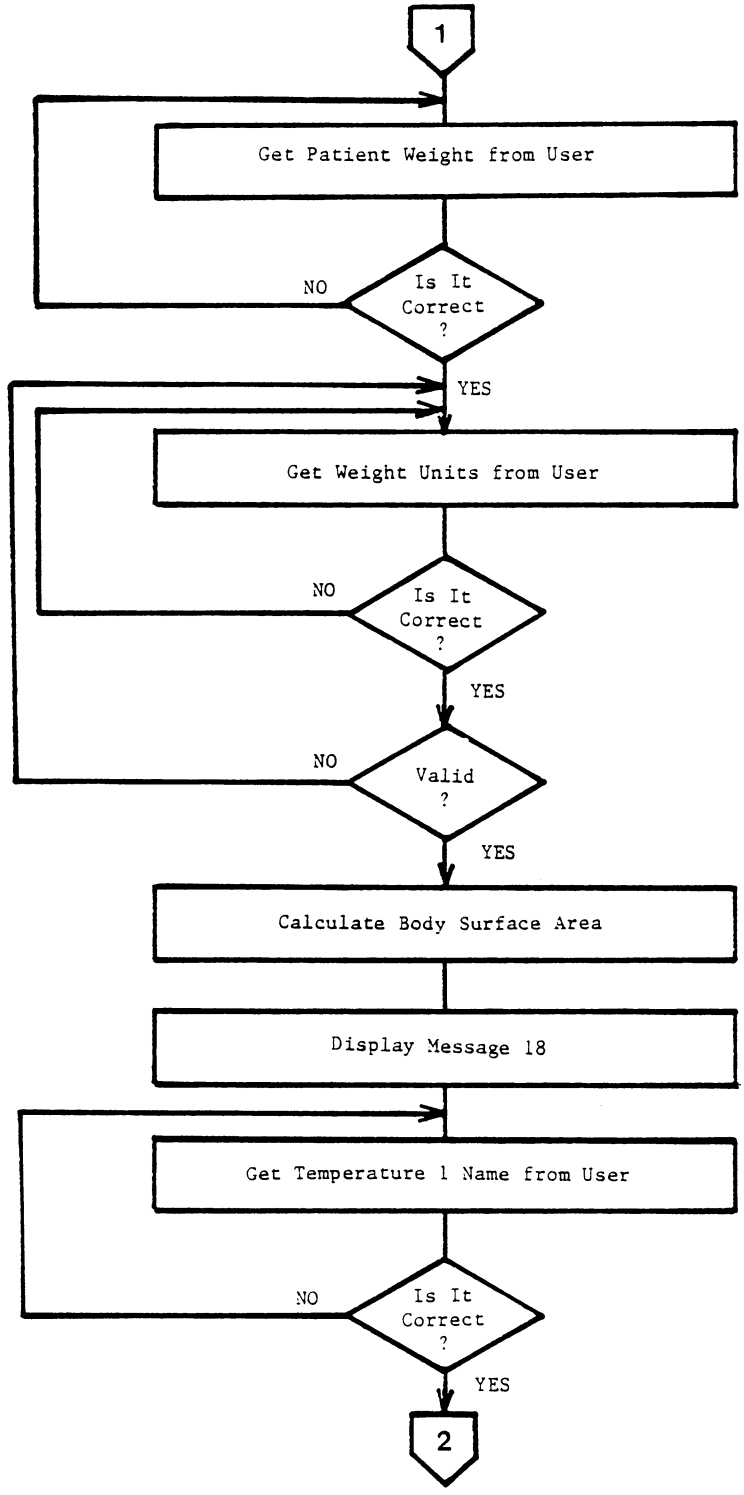


Figure 11 b. Input Section Flowchart

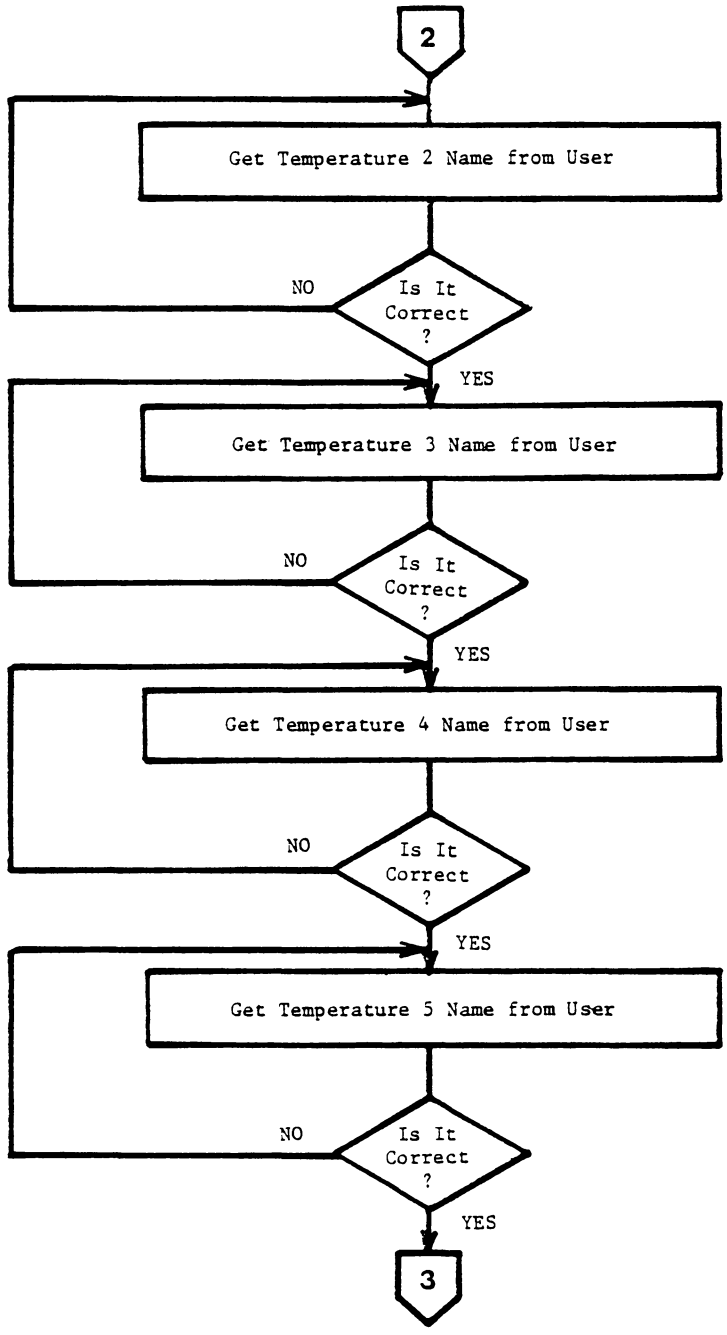


Figure 11 c. Input Section Flowchart

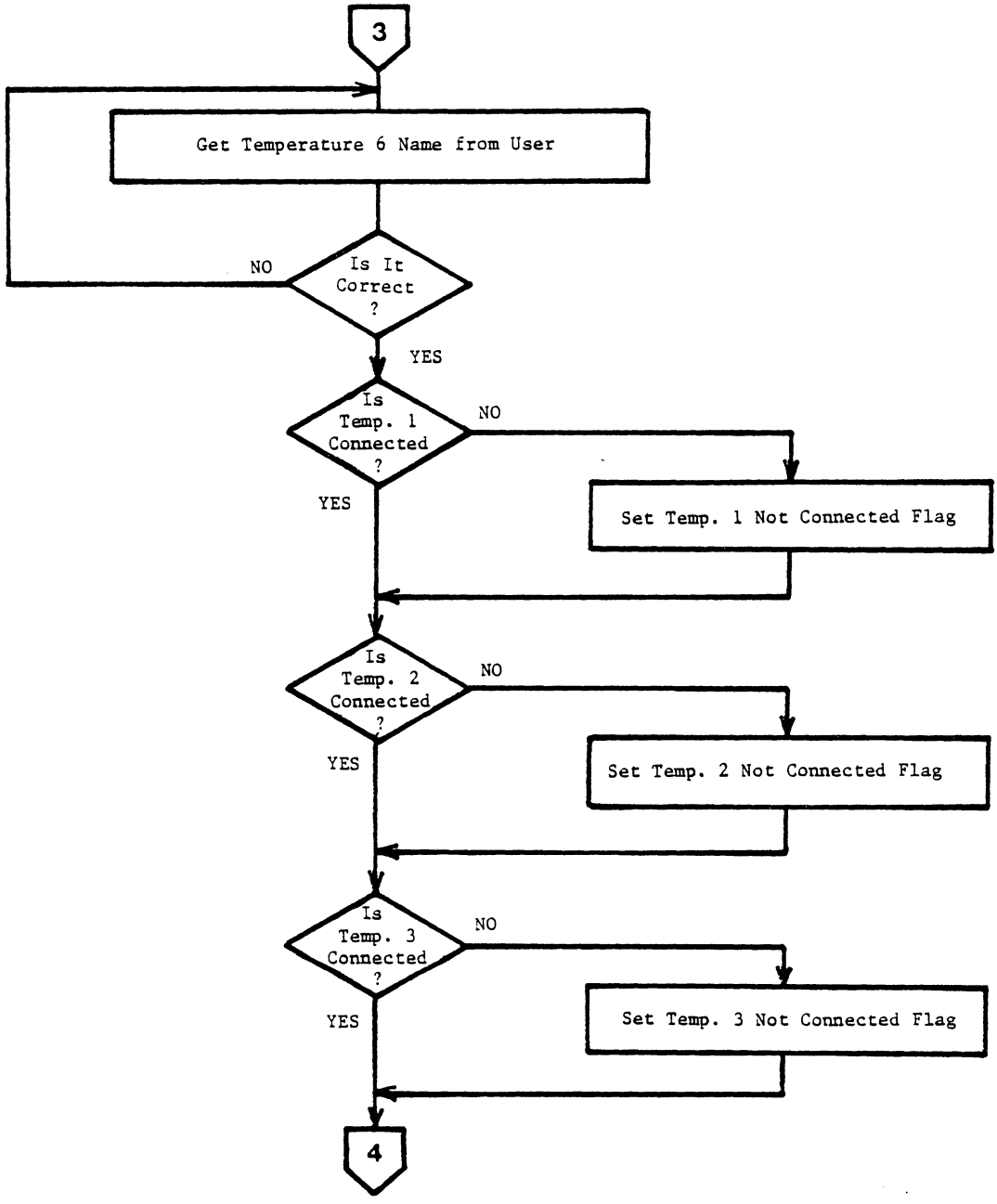


Figure 11 d. Input Section Flowchart.

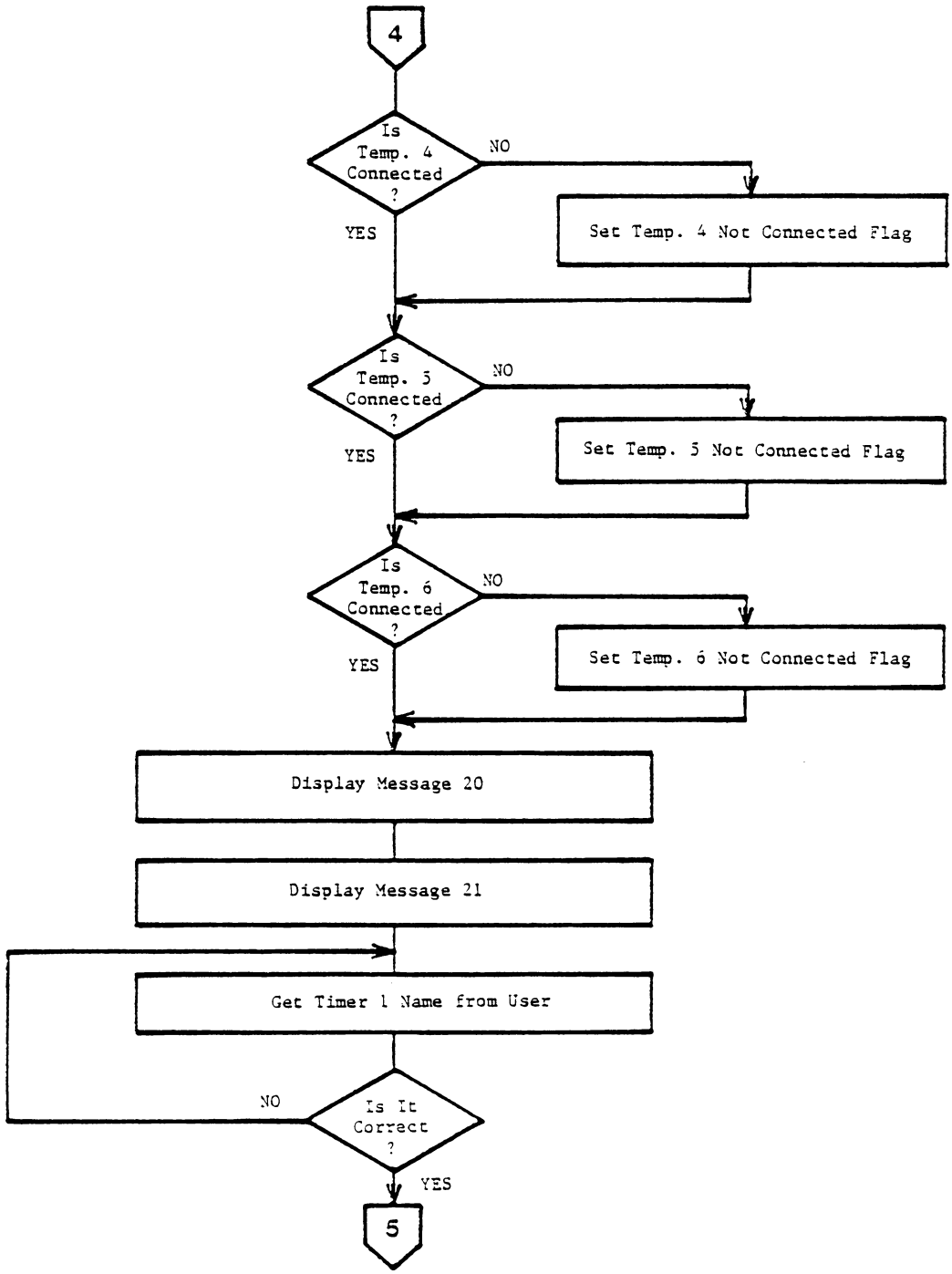


Figure 11 e. Input Section Flowchart

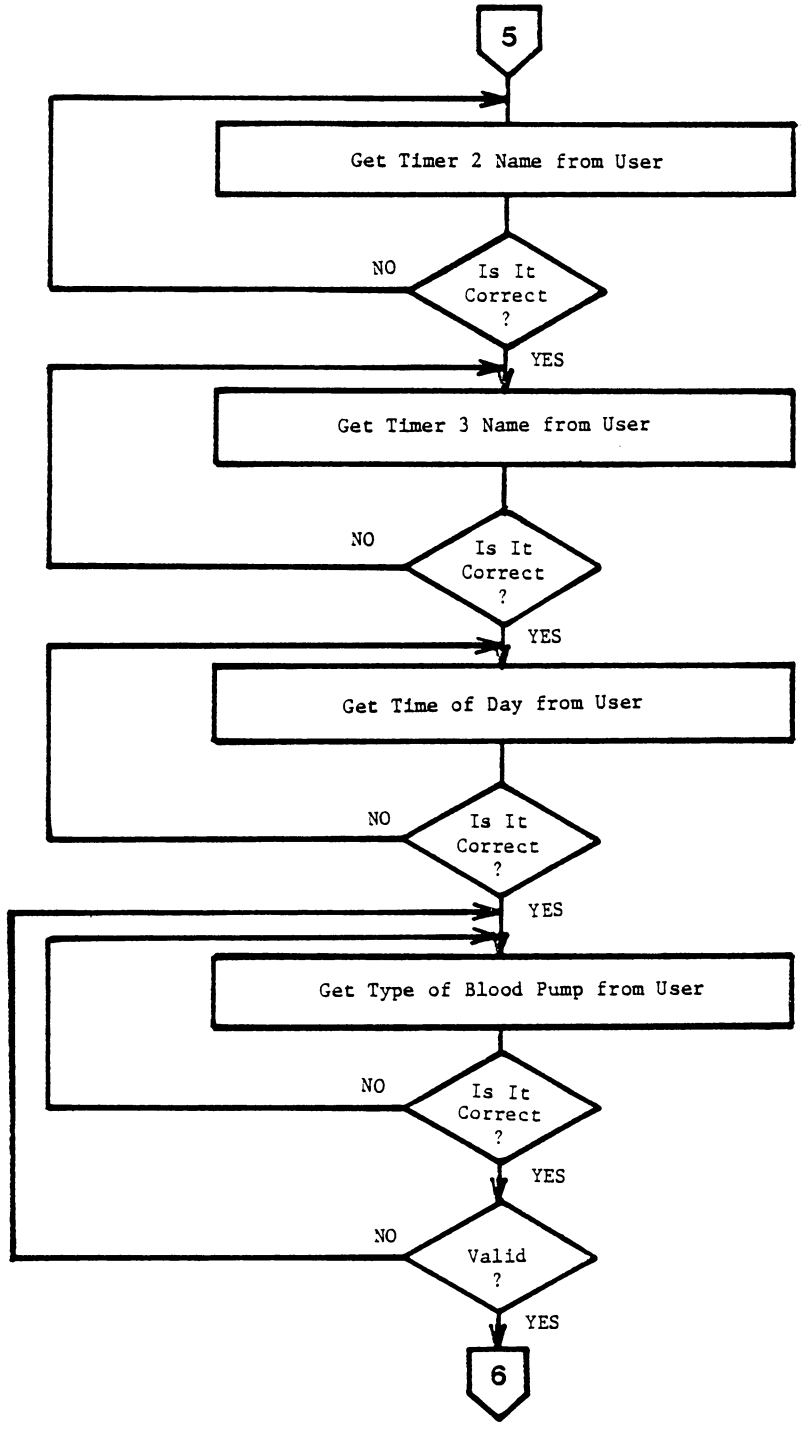


Figure 11 f. Input Section Flowchart

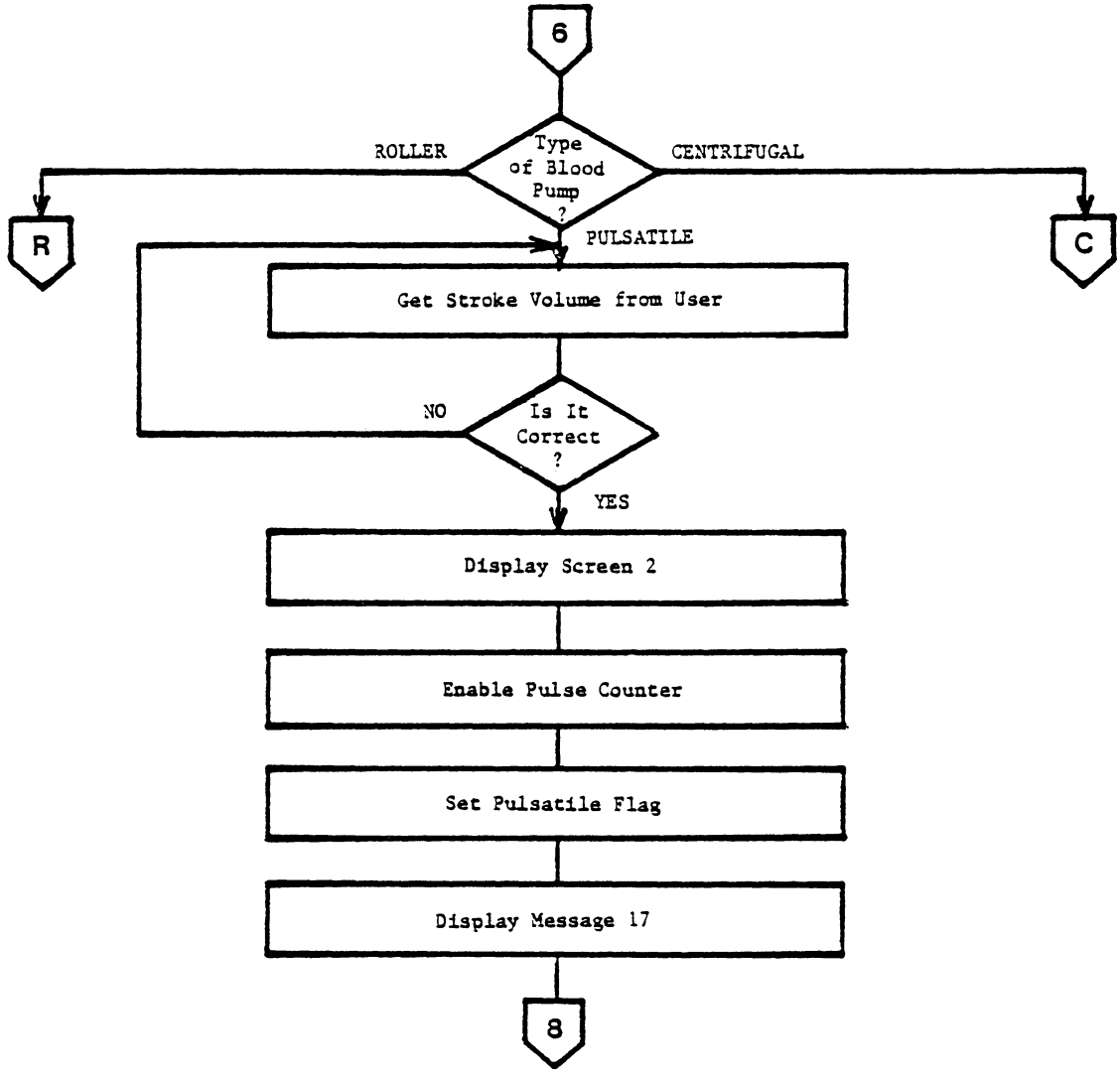


Figure 11 g. Input Section Flowchart

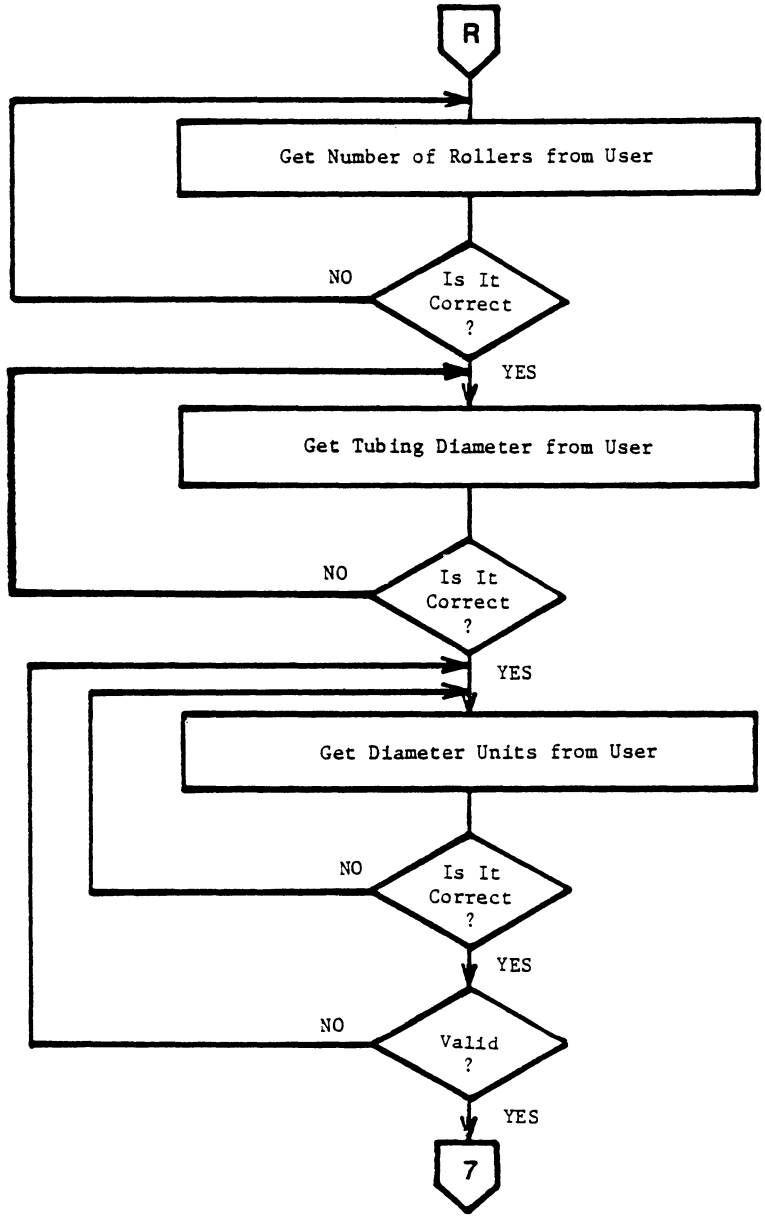


Figure 11 h. Input Section Flowchart

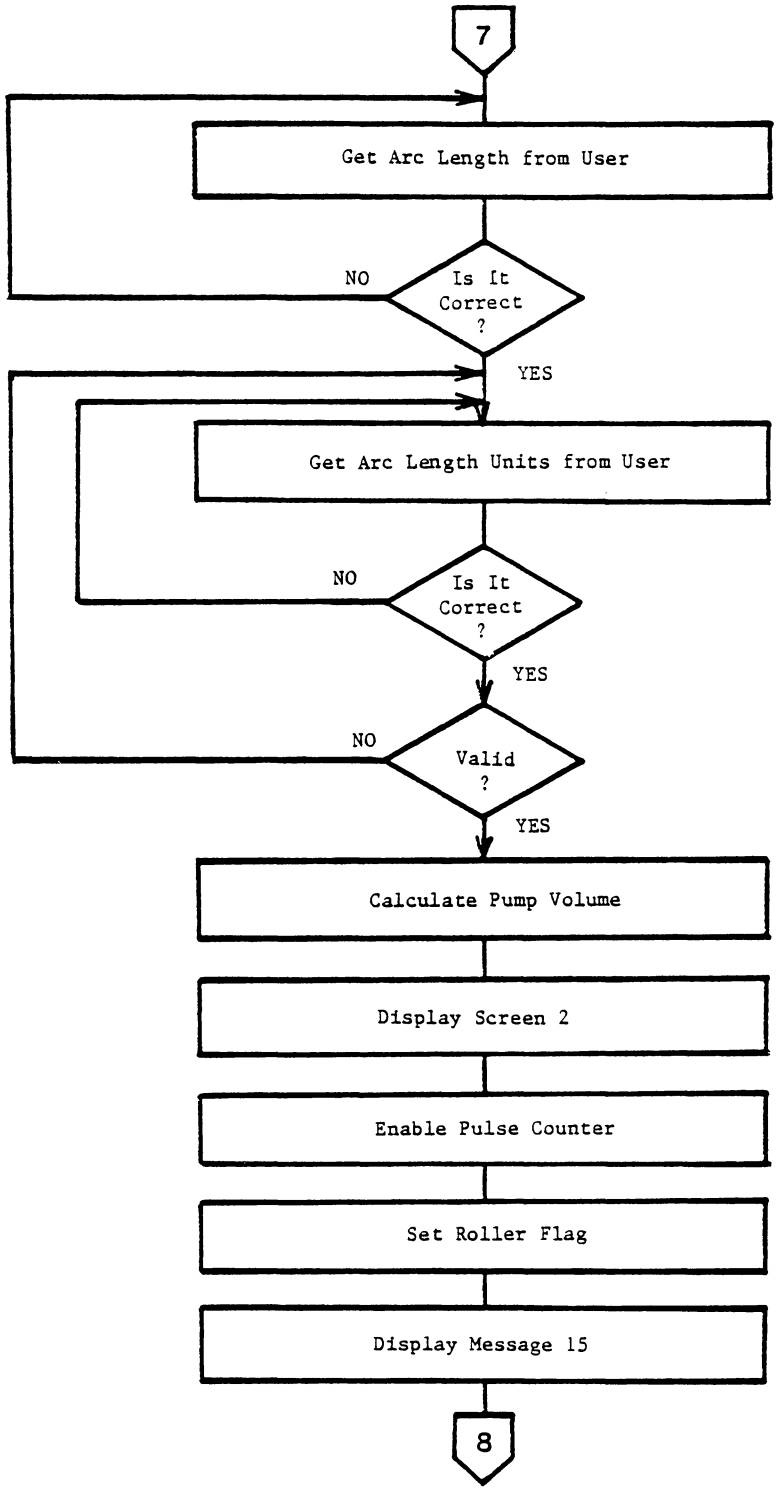


Figure 11 i. Input Section Flowchart

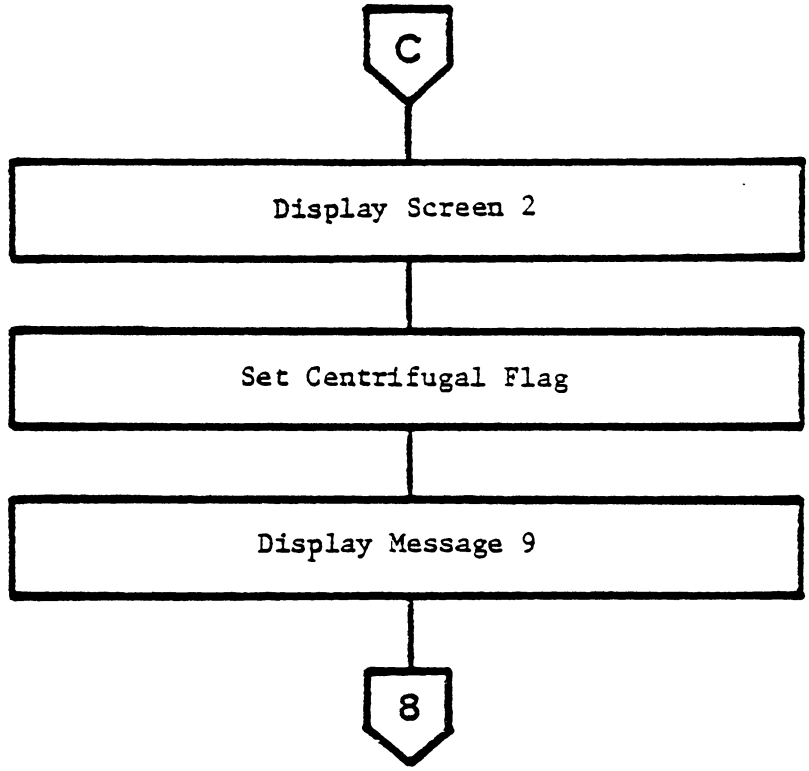


Figure 11 j. Input Section Flowchart

the entry is correct and is able to change the entry if needed. If an invalid entry is made, such as kd instead of kg, for kilograms, the program displays a message to the user to input the entry again.

Next, the temperature information is input. Labels for each of the six temperature probes are entered with a chance for correction if needed. If a temperature probe is not being used, then the user names the probe "N/C" for not connected. The program then checks for unconnected probes and sets the appropriate flags.

The third input section accepts the timer labels. Again the entries can be changed if needed. The current time is also input at this time.

Finally, the pump information is entered. First, the type of blood pump is input, either centrifugal, roller or pulsatile. Next, depending on the type of pump various parameters need to be entered.

If the centrifugal pump is being used, then screen 2 is displayed by an assembly language subroutine. Screen 2 is the monitor display as shown in Appendix B. Next the centrifugal flag is set and message 9, CENTRIFUGAL PUMP, is displayed in the correct location on the monitor screen.

If the roller pump is being used, then the user inputs the following information: the number of rollers, the tubing diameter and units, and the arc length and units. The

program checks for valid units and allows the user to correct any of the inputs if necessary. Next, the roller pump volume, PV, in ml, is calculated using the following formula:

$$PV = NR*AL*3.1416*(TD**2)/4$$

where NR is the number of rollers, AL is the arc length in cm, and TD is the tubing diameter in cm. The program then displays screen 2, the monitor screen, as shown in Appendix B. Next, the revolutions per minute pulse counter on the 6522-1 versatile interface adapter is enabled and cleared. The roller flag is then set and message 15, ROLLER PUMP, is displayed in the correct spot on the screen.

If a pulsatile pump is being used, then the user is asked to input the stroke volume of the pump in cc. Again, the user can correct his response if necessary. The monitor screen, screen 2, is then displayed, as shown in Appendix B. The pulse counter on the versatile interface adapter is then cleared and enabled, the pulsatile flag set, and message 17, PULSATILE PUMP, is displayed at the correct screen location.

Monitor and Display Loop

Next, the program displays all of the patient monitor information as flowcharted in Fig. 12. First, the names of the patient, the timers, and the temperatures are displayed on the screen. The program then prints the initial timer values of 00:00:00 (hours:minutes:seconds). Next, the

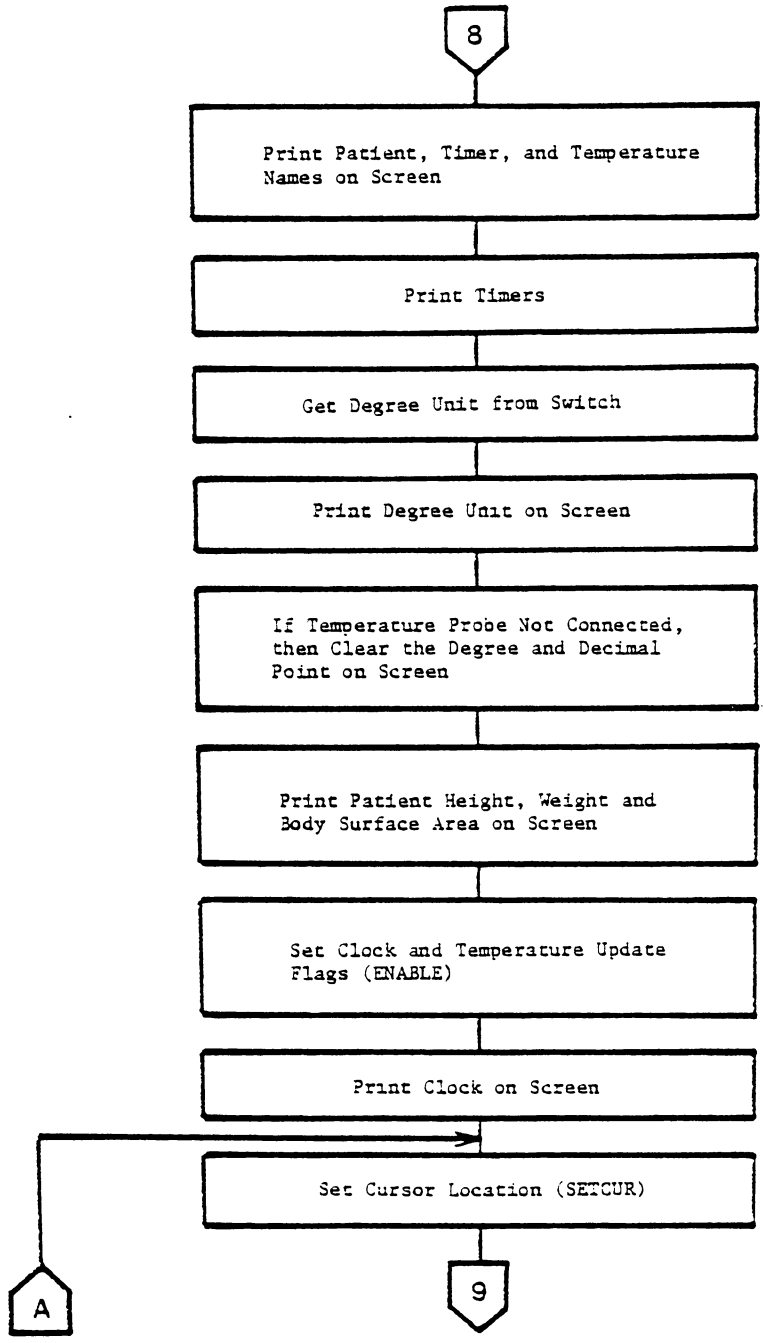


Figure 12 a. Monitor and Display Flowchart

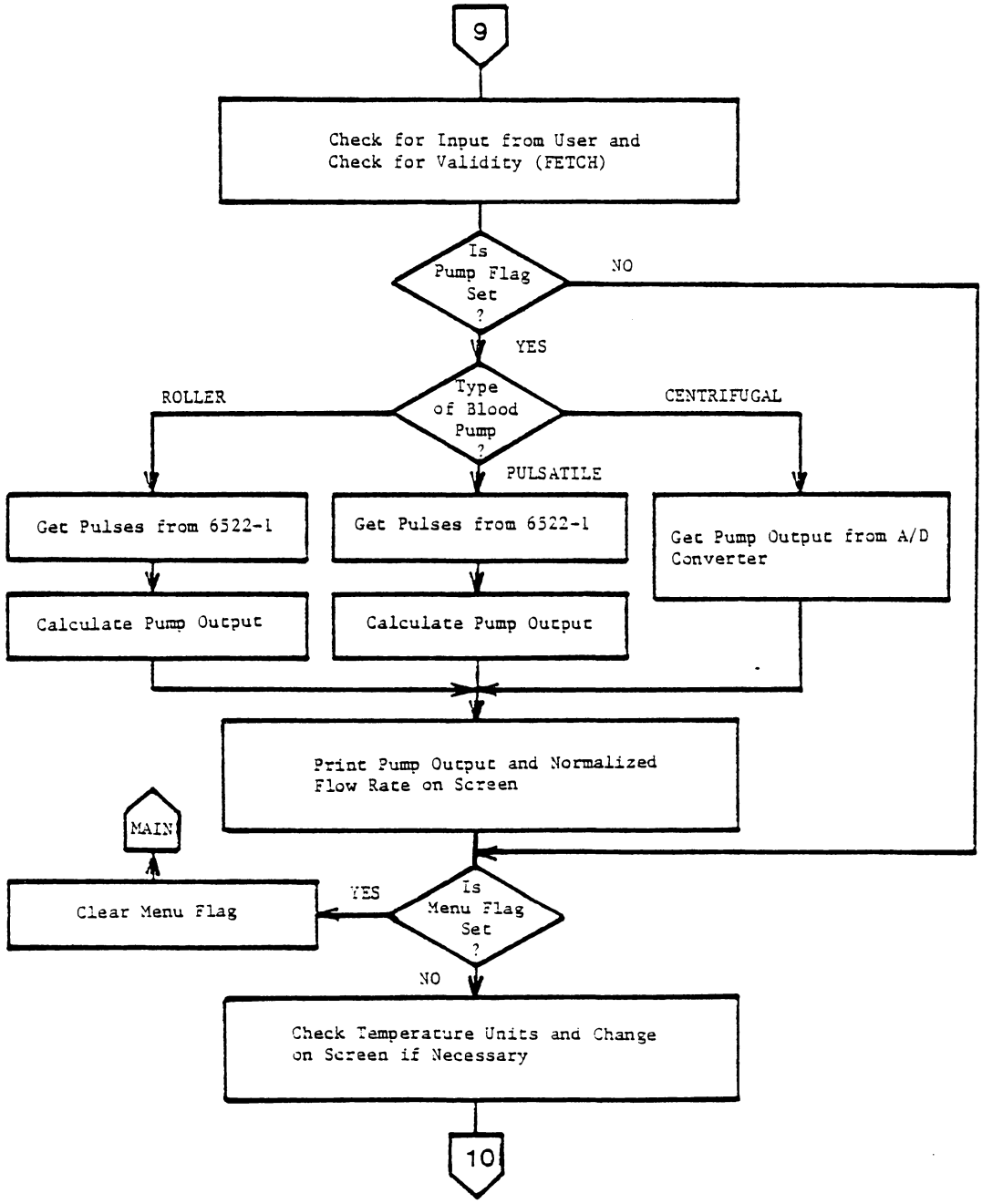


Figure 12 b. Monitor and Display Flowchart

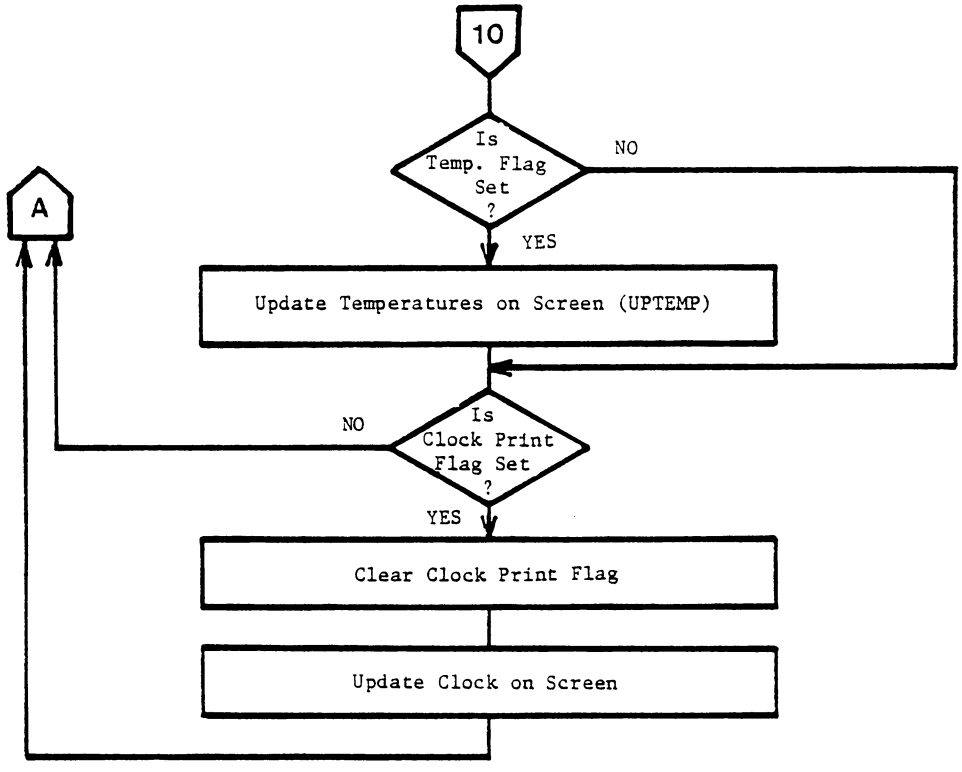


Figure 12 c. Monitor and Display Flowchart

temperature degree unit is input from the peripheral interface adapter and printed on the screen. Then, the degree unit and the decimal point are cleared on the screen, if the corresponding temperature probe is not being used. The patient's height, weight and surface area are printed next. Finally, the clock and temperature update flags are set, then the clock is printed on the screen.

The next section described is the software loop that the computer stays in while monitoring the patient information. First, software sets the cursor location for the display of the timer commands, such as s1 which starts timer 1. A machine language subroutine, SETCUR, uses a VIC 20 subroutine to set the cursor location on the screen. Then, using the machine language subroutine, FETCH, based on a subroutine described in Browning's thesis, "Design of a Microcomputer-Based Microporous Membrane Process Controller" [6], the program checks for any keyboard input from the user and checks its validity. If a valid input is found then the input is printed on the screen at the cursor location and the command is implemented. For example, an input of s1 would cause timer 1 to start and s1 would be printed on the screen. If an invalid input is found then the timer command is cleared from the screen and the program execution continues.

Next, the program checks to see if it is time to update the pump output by checking the pump update flag. If the

flag is set, then the program checks for the type of blood pump being used, either centrifugal, roller, or pulsatile.

If the centrifugal flag is set, then the program gets the pump output from the A/D converter using the machine language subroutine, TNEW, based on the subroutine described in Mears' thesis, "A Microcomputer-Based Temperature and Humidity Control System for Respiratory Therapy" [7].

If the roller flag is set, then the program takes the number of pulses from the 6522-1 and calculates the pump output, PO, in ml/min using the following formula:

PO = Pump Volume * Number of Pulses * 1.2

where the pump volume is in ml, and the pulses are counted for 1/2 sec with 100 pulses being generated per revolution.

When the pulsatile flag is set, the number of pulses from the 6522-1 is read and the pump output in ml/min is calculated using the following equation:

Pump Output = Stroke Volume * Number of Pulses * 120

where the stroke volume is in cc, and the pulses are counted for 1/2 sec.

The pump output and the two normalized blood flow rates are then printed on the screen. If the pump flag was not set then the pump output is not updated on the screen and the program continues to determine whether or not the user wants to return to the input section by checking the menu flag.

If the menu flag is set, then the program returns to the beginning of the input section. If the menu flag is not set, then the program checks to see if the operator has changed the temperature degree units to be displayed on the screen. The degree units are then changed on the screen if necessary.

Next, the program checks the temperature update flag. If the flag is set, then it is time to update the temperatures. The program updates the temperatures using the machine language subroutine, UPTEMP. If the temperature update flag is not set, then the program does not update the temperatures and continues to check the clock print flag.

When the clock print flag is set, the program reads the time of day using the VIC 20 BASIC command, TI$ [3]. This value is then displayed on the screen and the flag is cleared. The program then goes back to setting the cursor location and checking for user input. If the clock print flag was not set, then the program returns to setting the cursor and checking for keyboard input. The program continues to repeat this loop until the power is turned off.

Assembly Language Subroutines

In the following discussion of the assembly language subroutines, the starting address of the subroutine is given in parentheses after the subroutine name.

Video Display

The title and monitor screens, as shown in Appendix B, are displayed using subroutines SCREN1 ($A47E) and SCREN2 ($226A) respectively. These routines use a series of load and store commands which load a number into the processor accumulator then store the value at a memory location. Characters are displayed on the VIC 20 by storing an ASCII value in the screen memory. The screen memory starts at address $1E00 and continues to address $1FF9. These addresses describe a character grid of 22 columns by 23 rows, where $1E00 is the location of the character to be displayed in the top left corner, and $1FF9 is the location of the character to be displayed at the bottom right corner of the screen [3].

Similarly, each character has a color determined by a value placed in the color memory which starts at address $9600 and ends at address $97F9. Seven character colors can be used, black, white, red, cyan, purple, green, blue, and yellow. These colors are displayed, respectively, by storing a number from 0 to 7 at the address corresponding to the desired screen location.

All of the printed messages on the screen are accomplished using one of two subroutines, DISPLY and MSGE. Both of these subroutines are based on routines described in

Browning's thesis "Design of a Microcomputer-Based Microporous Membrane Process Controller" [6].

The first routine, DISPLY ($A38A), takes a string of ASCII values in memory and outputs them one at a time to the screen using the VIC 20 subroutine, CHROUT ($FFD2). This routine takes the ASCII value in the accumulator and displays it on the screen at the current cursor location. The routine, DISPLY, simply needs the starting address of the string of ASCII values as well as the number of characters to be displayed.

The second display routine, MSGE ($A398), stores a string of ASCII values at a certain location in screen memory. This routine needs, in addition to the starting address and the length of the character string, the starting address of the screen memory where the character string is to be displayed.

Keyboard Input

During the monitoring loop, the operator can start, stop, or reset any of the three timers, as well as return to the patient input section. Two subroutines, SETCUR and FETCH, get the user input from the keyboard then display and implement any valid commands.

SETCUR ($A20C) uses a VIC 20 subroutine, PLOT ($FFF0), which when the accumulator carry flag is clear, positions the

cursor to the column (0-21) indicated by the Y register and the row (0-22) indicated by the X register. In SETCUR, the cursor is positioned at column 18 and row 2 for display of the valid commands.

FETCH ($A215 checks to see if any keyboard entry has been made, using the VIC 20 subroutine, GETIN ($FFE4), which removes one character from the keyboard queue. If the queue is empty then after returning from subroutine, GETIN, the accumulator will be zero. If there is an entry, then FETCH checks it for validity. There are 10 valid entries:

s1 - starts timer 1

s2 - starts timer 2

s3 - starts timer 3

e1 - stops timer 1

e2 - stops timer 2

e3 - stops timer 3

r1 - resets timer 1

r2 - resets timer 2

r3 - resets timer 3

m - returns to input section

The user must press the RETURN key after each of the above entries. If one of these entries is found, it is displayed on the screen and its function is implemented. If none of these entries are recognized, then the display is cleared and program execution continues.

Temperature Update

In order to update the temperatures, a subroutine called UPTEMP ($4000), is called. This subroutine first checks for an unconnected temperature probe by looking at the respective temperature connect flag. If the flag is set, then the temperature probe is not being used and the program continues to check the next flag. If the flag is not set, then the routine closes the switch connecting that probe to the temperature board while at the same time opening all of the other probes' switches. This is accomplished by writing, to port B of the 6520-1, a 1 in the bit position connected to the switch enable line of the temperature to be updated, while storing zeros at all other bit positions.

Once the correct probe is connected to the temperature board, the output of the A/D converter is read using the subroutine, TNEW ($A423). This routine delays for 1/2 sec to make sure the converter is displaying the correct value. This delay is needed since the converter has a long recovery time from an underange condition, i.e., any voltage under 0.180 V. After the time delay, the routine waits for an end-of-conversion pulse by monitoring the B control register of the 6520-1. When the pulse occurs, bit 7 of the B control register is set to a 1. Next, the routine monitors port A or the 6520-1 to wait for digit select 1, DS1, to go high. When DS1 goes high, the value at port A is stored in memory. Both

the digit select lines and the binary coded decimal data lines are connected to port A of the 6520-1. Therefore, when port A is read the data for digit 1 is on the lines when the digit select 1 line is high. This eliminates the possibility of reading the wrong data for the desired digit. Similarly, the program reads and stores the second through the fourth digit, then returns to the routine UPTEMP.

Interrupt Processing

The VIC 20 generates an interrupt request signal every 1/60 sec. This signal causes program execution to jump to a VIC 20 interrupt servicing routine which is used to update the system clock as well as check to see if any keyboard entries have been made. This interrupt routine is easily bypassed simply by storing the address of the programmer's routine at addresses $0314 (low byte) and $0315 (high byte) in the initialization routine.

In this design, the VIC 20 routine is bypassed and the routine flowcharted in Fig. 13 is used. As shown in Fig. 13, the routine first checks to see if the roller pump is being used. If it is then the pump output update counter is incremented and compared with 30. Since the interrupt is generated every 1/60 sec, a count of 30 is equivalent to 1/2 sec. If the counter has reached 30, then the pump output update flag is set. The routine then reads the number of

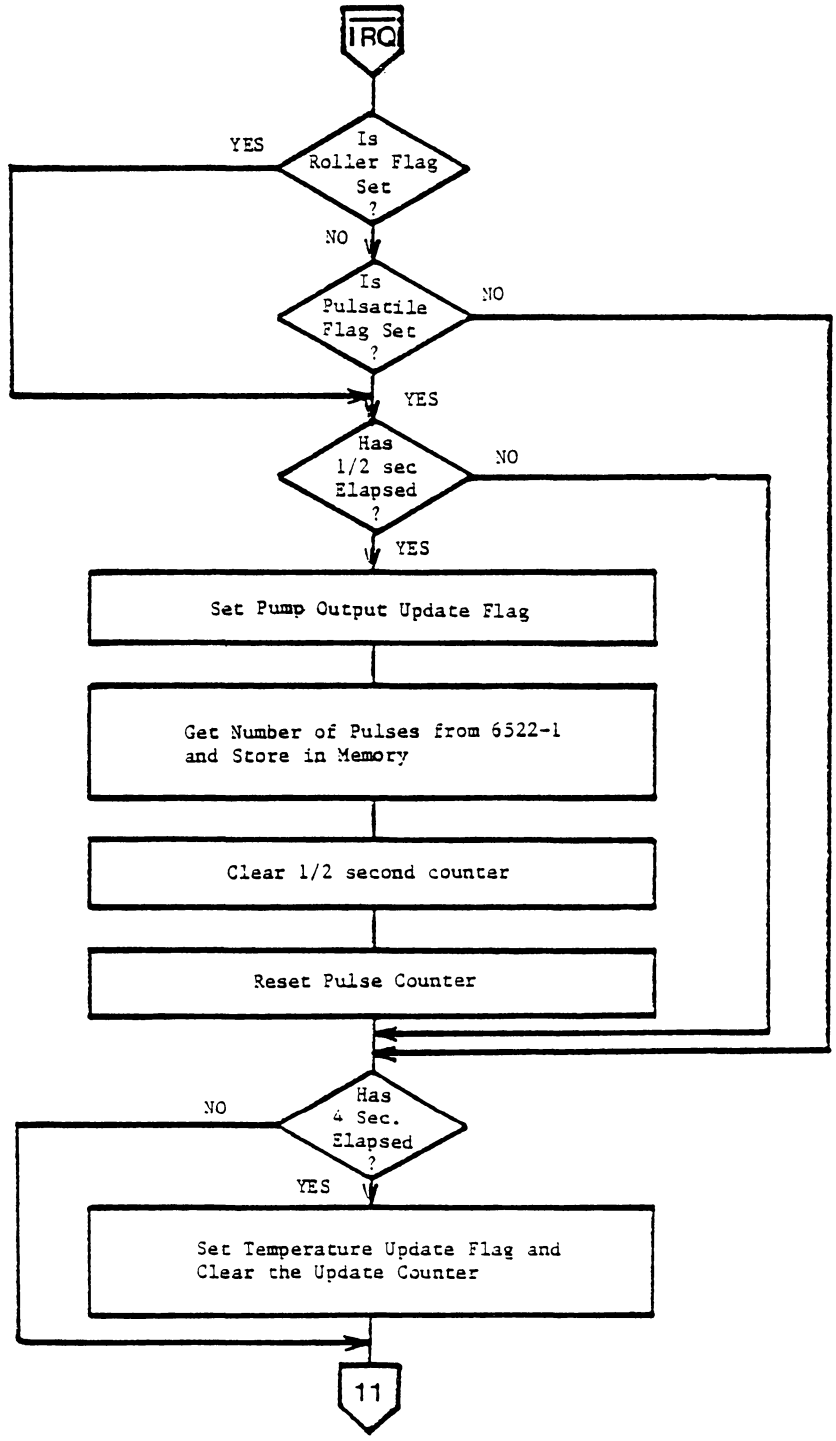


Figure 13 a. Interrupt Routine Flowchart

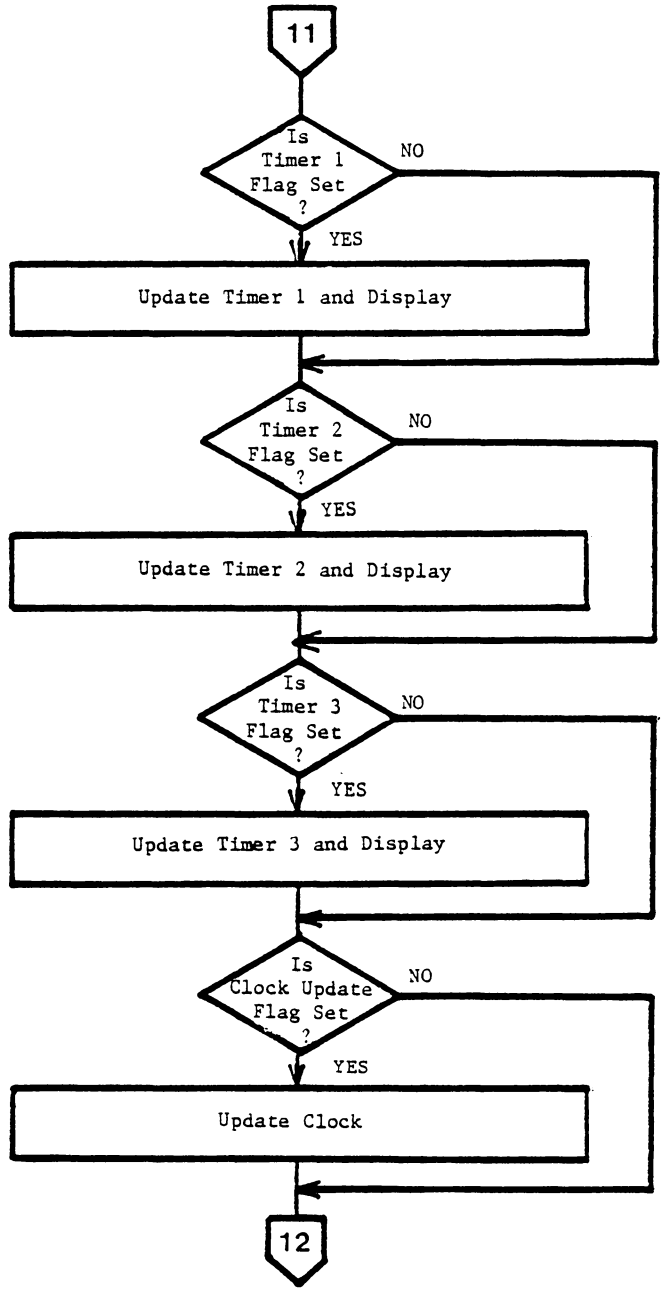


Figure 13 b. Interrupt Routine Flowchart

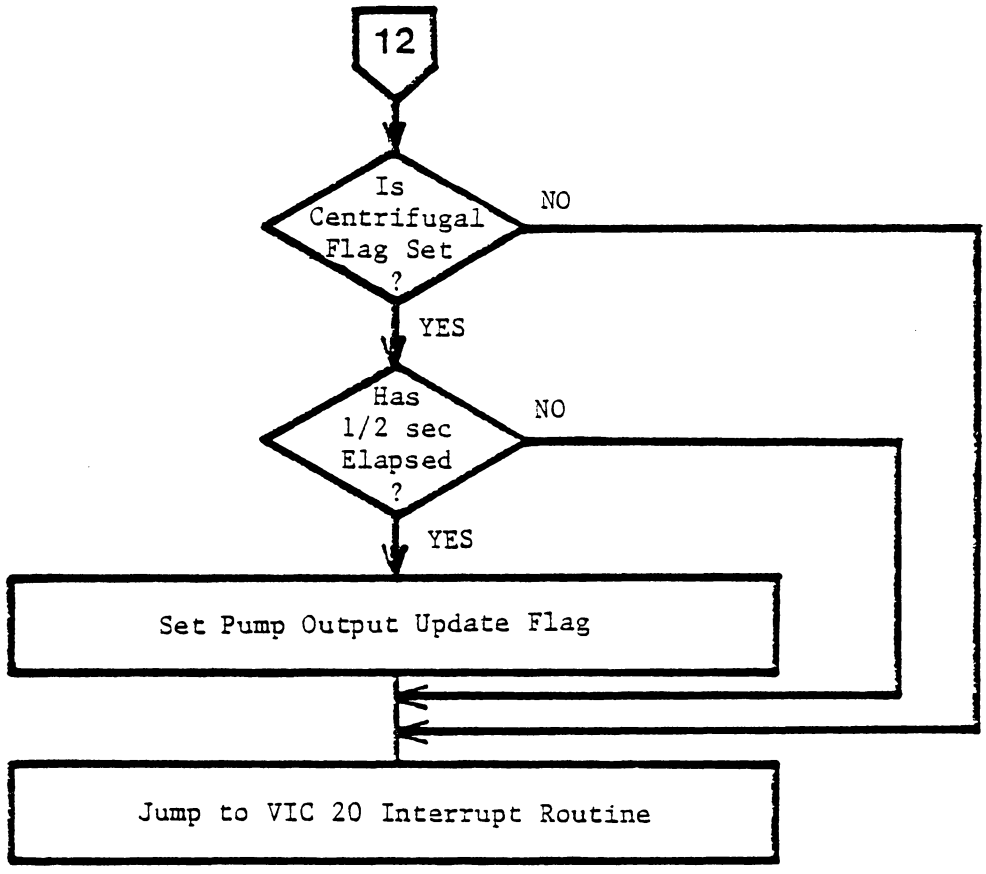


Figure 13 c. Interrupt Routine Flowchart

pulses counted by the 6522-1, the versatile interface adapter, and stores the value in memory. The pump output update counter is then cleared along with the pulse counter on the 6522-1.

If the roller pump is not being used, then the program checks for a pulsatile pump. If the pulsatile pump is being used, then the program increments the pump output update counter, compares it to 30 and sets the pump output update flag if 1/2 sec has elapsed. Then, just as if the roller pump was being used, the pulse counter is read and stored in memory, and the pump output update counter along with the pulse counter are cleared.

If neither the roller or the pulsatile pump is being used, or if 1/2 sec has not elapsed, then the program jumps to the next portion of the interrupt service routine as shown in Fig. 13.

This section increments the temperature update counter and compares it to 240 (4 sec). If 4 sec has elapsed, then the temperature update flag is set, the counter is cleared, and program execution continues to the timer update section. If 4 sec has not gone by, then the program jumps to the timer update section.

In the timer update section, first the timer 1 flag is read. If the flag is set then the timer 1 counter is incremented, stored in memory, and displayed on the screen if

1 sec has elapsed (60 counts). The program execution then continues to check the timer 2 flag. If the timer 1 flag is not set then the program branches to check the timer 2 flag. Timers 2 and 3 are checked and updated similarly to the timer 1 routine.

After checking timer 3, the program checks the clock update flag. If the flag is set then a counter is incremented and compared with 60. If the counter is equal to 60 (1 sec), then the clock print flag is set, the counter is cleared, and program execution continues to check for a centrifugal pump. If the clock update flag is not set, or if the counter has not reached 1 sec, then the program branches to the centrifugal pump routine.

If the centrifugal flag is set, then a pump update counter is incremented and compared to 30. If 1/2 sec has elapsed then the pump output update flag is set, the counter is cleared, and the program jumps to the VIC 20 interrupt service routine. If either the centrifugal pump is not being used or the 1/2 sec has not elapsed, then the program jumps to the VIC 20 interrupt routine.

V. SYSTEM EVALUATION

Electrical System

The electrical system worked as expected. The time of day clock worked perfectly, as compared with a timex quartz watch. When the clock was synchronized with the watch, after 3 hours the clock still agreed with the watch. Also, all three of the timers ran for 3 hours keeping perfect time.

As shown in Table 3, the switches to connect the 6 temperature probes and the centrifugal flow probe worked perfectly. To show this, different voltages were connected to the probes and the screen displayed the correct voltages for the corresponding probes.

In order to verify that the A/D converter worked correctly, 11 voltages ranging from 0.000 to 2.000 were input through all of the temperature probes and the centrifugal flow probe. As shown in Fig. 14, the input voltages, as measured with a digital voltmeter (DVM), compared within a millivolt to the values displayed on the patient monitor screen.

To prove that the pulse counter on the versatile interface adapter was working correctly, square waves with varying frequencies were input to the interface board using a 555 timer integrated circuit. The square waves were monitored on an oscilloscope to compare with the values

Table 3. Verification of Analog Switches

Probe	Input, V *	Screen Display
Temp. 1	0.020	2.3
	1.999	199.9
Temp. 2	0.667	67.0
	0.020	2.3
Temp. 3	0.930	93.2
	1.615	161.4
Temp. 4	1.267	126.8
	0.930	93.2
Temp. 5	1.615	161.4
	0.668	67.0
Temp. 6	1.797	179.7
	1.268	126.8
Centrifugal Flow	0.668	670.
	0.020	2.

*As measured by a digital voltmeter, Fluke model 8050A, SN 3016073, on 8/20/85

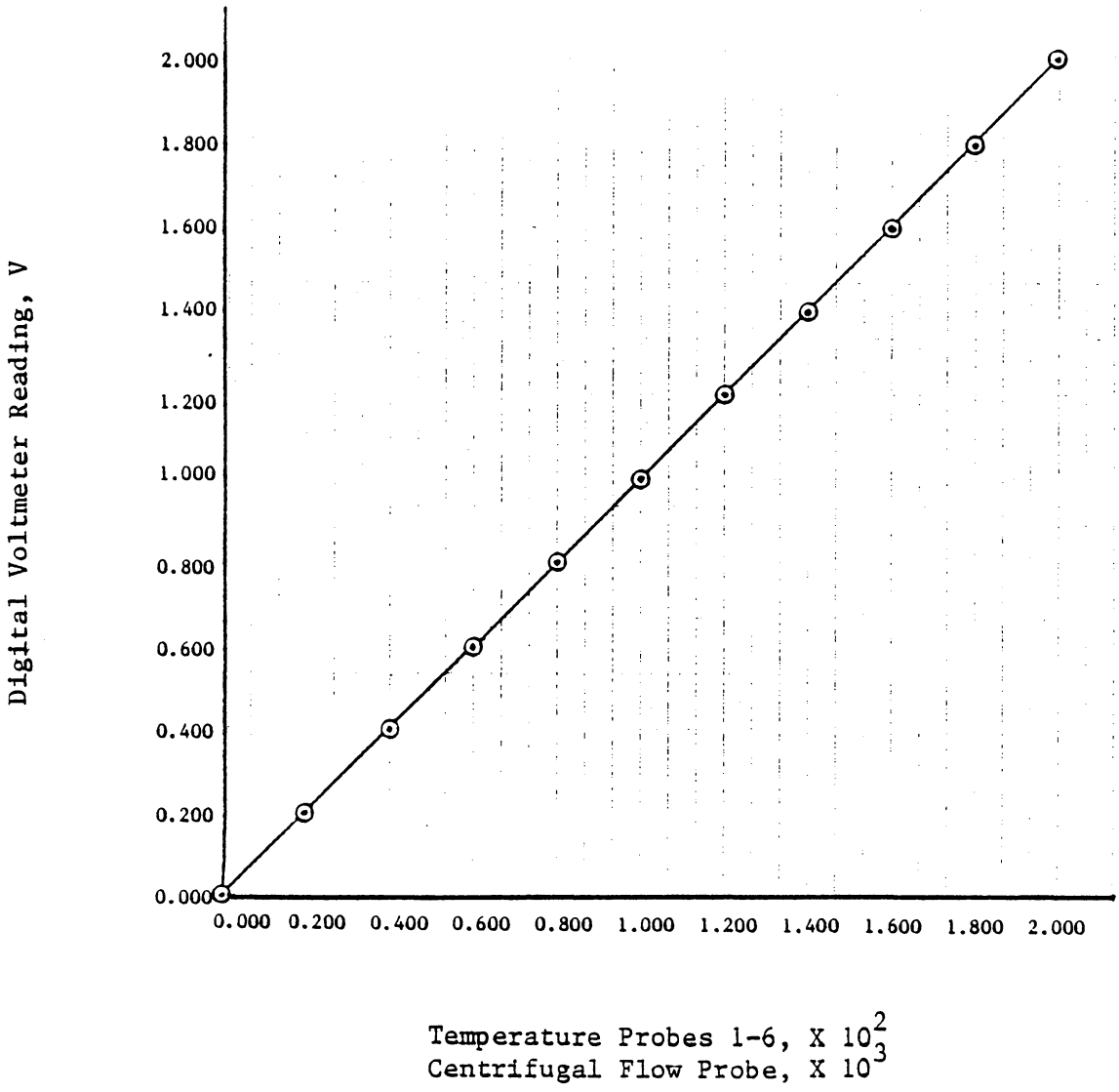


Figure 14. Accuracy of Temperature and Centrifugal Flow Probes

counted by the 6522-1. The pulses counted were calculated, for the roller pump, by dividing the displayed pump output by 1.2 times the pump volume. For the pulsatile pump, the pulses were calculated by dividing the displayed pump output by 120 times the stroke volume. Table 4 shows the pulses counted by the 6522-1 versus the oscilloscope readings for both the roller and the pulsatile pump. This table shows a slight discrepancy, at the most 13 pulses, between the oscilloscope reading and the number of pulses counted by the 6522-1. This is due to the fact that the oscilloscope cannot be read with the accuracy and precision necessary to compare with the pulse counter. For all of the data points the pulse counter predicted a reading which agreed with the oscilloscope within $\pm$ 2 small divisions.

Software System

The software worked as expected, with all calculations checked using a Hewlett-Packard 15C calculator. As shown in Tables 5 and 6, the calculated and displayed pump output for both the roller pump and the pulsatile pump agreed exactly, within the significant figures displayed on the screen.

To make sure that the calculations involving the patient data were correct, data values as shown in Table 7 were input and the displayed outputs compared with the calculated values using the hp 15C. The calculator predicted exactly the values

Table 4. Verification of Pulse Counter

Oscilloscope *			Monitor Screen		
No. of Divisions	Time Scale/Div.	Pulses Counted	No. of Divisions	Time Scale/Div.	Pulses Counted
4.3	10 ms	11.6	4.5/4.1	10 ms	11/12
4.8	5 ms	20.8	5.0	5 ms	20/21
4.1	5 ms	24.3	4.2	5 ms	24
4.2	1 ms	119.0	4.4	1 ms	113/114
4.5	0.5 ms	222.2	4.6	0.5 ms	216/217
4.0	0.5 ms	250.0	4.2	0.5 ms	237/238

* Hitachi, V-152F 15 MHz, SN 430024, used on 8/21/85

Table 5. Roller Pump Output

Inputs				Calculated and Displayed Pump Output, ml/min
Pulses/ 1/2 sec	No. of Rollers	Tubing Diameter	Arc Length	
11/12	2	0.5 in.	3 in.	255/278
11/12	2	1 cm	6 cm	124/136
20/21	2	1 cm	6 cm	226/238
20/21	2	0.5 in.	3 in.	463/487
216/217	2	0.5 in.	3 in.	5004/5027
216/217	2	1 cm	6 cm	2443/2454
237/238	2	1 cm	6 cm	2680/2692
237/238	2	0.5 in.	3 in.	5491/5514

Table 6. Pulsatile Pump Output

Inputs		Calculated and Displayed Pump Output, ml/min
Pulses/1/2 sec	Stroke Volume, cc	
11	0.1	132.
11	1.0	1320.
12	0.1	144.
12	1.0	1440.
24	0.1	288.
24	1.0	2880.
113	0.05	678.
113	0.1	1356.
114	0.05	684.
114	0.1	1368.
216	0.05	1296.
216	0.1	2592.
217	0.05	1302.
217	0.1	2604.

Table 7. Patient Data Calculations

Inputs		Calculated and Displayed Outputs				
Height	Weight	Height	Weight	Surface Area, m2	* ml/min/kg	* ml/min/m2
24 in.	10 lb	61 cm	5 kg	0.27	439.	7400.
36 in.	20 lb	91 cm	9 kg	0.48	219.	4108.
48 in.	50 lb	122 cm	23 kg	0.88	88.	2259.
52 in.	75 lb	132 cm	34 kg	1.11	58.	1794.
60 in.	100 lb	152 cm	45 kg	1.39	44.	1431.
66 in.	150 lb	168 cm	68 kg	1.77	29.	1124.
72 in.	200 lb	183 cm	91 kg	2.13	22.	934.
78 in.	250 lb	198 cm	113 kg	2.48	18.	802.
84 in.	300 lb	213 cm	136 kg	2.83	15.	703.
90 in.	320 lb	229 cm	145 kg	3.06	14.	651.
70 cm	10 kg	28 in.	22 lb	0.42	199.	4784.
80 cm	12 kg	31 in.	26 lb	0.50	166.	4019.
100 cm	15 kg	39 in.	33 lb	0.64	133.	3109.
110 cm	20 kg	43 in.	44 lb	0.78	100.	2568.
130 cm	30 kg	51 in.	66 lb	1.04	66.	1915.
140 cm	40 kg	55 in.	88 lb	1.24	50.	1606.
160 cm	50 kg	63 in.	110 lb	1.50	40.	1326.
170 cm	80 kg	67 in.	176 lb	1.92	25.	1039.
190 cm	100 kg	75 in.	220 lb	2.28	20.	872.
200 cm	120 kg	79 in.	265 lb	2.56	17.	777.

*NOTE: All blood flows calculated using a pump output of 1990. ml/min

displayed on the screen, within the significant figures displayed.

VI. RECOMMENDATIONS AND CONCLUSIONS

The system described in this thesis works perfectly. However, before using the system in the operating room, shielded cable should be added to connect the temperature board with the interface circuit board and the temperature probes. Also, shielded cable should be used to connect the pulse generator and the centrifugal flow probe with the interface circuit board.

REFERENCES

1. Berger, E. C., Letter to Leon J. Arp, April 28, 1983.

2. Berger, E. C., Letter to Leon J. Arp, May 5, 1983.

3. Finkel, A., N. Harris, P. Higginbottom, and M. Tomczyk, VIC 20 Programmer's Reference Guide, Commodore Business Machines, Inc. and Howard W. Sams and Co., Inc. King of Prussia, Pennsylvania, 1983, pp. 5-211, 241-245, 263-274.

4. Hampshire, N., VIC Revealed, Hayden Book Company, Inc., Rochelle Park, New Jersey, 1982, pp. 44-102.

5. R6500 Microcomputer System Hardware Manual, Rockwell International Corp., Document No. 29650, N31, Rev. 1, Anaheim, California, 1978.

6. Browning, D. R., "Design of a Microcomputer-Based Microporous Membrane Process Controller," Master Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1985.

7. Mears, D. T., "A Microcomputer-Based Temperature and Humidity Control System for Respiratory Therapy," Master Thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1980.

8. MC1443, Motorola Semi Conductor Products Inc., Document O. DS-9423, Austin, Texas, 1976.

9. Linear Databook, National Semiconductor Corporation, "Voltage References," Santa Clara, California, 1980, pp. 2-26 - 2-31.

APPENDIX A

Software Listing

This appendix lests both the BASIC and the assembly language programs used in the patient monitor device.

Basic Program

In the listing of the BASIC program, the statements are listed in the left column separated by colons. In the right column, comments are written, separated by colons, to help explain the BASIC statements.

The symbols and abbreviations used in this listing follow.

```
?          = BASIC keyword for PRINT
sys  XXXX  = BASIC command which jumps to a machine language
             subroutine at address XXXX
↑          = BASIC exponent operator
pn$        = Patient's name
ph         = Patient's height
hu$        = Height unit, either cm or in.
pw         = Patient's weight
wu$        = Weight unit, either kg or lb
sa         = Patient's body surface area
t1$        = Temperature 1 name
t2$        = Temperature 2 name
t3$        = Temperature 3 name
t4$        = Temperature 4 name
t5$        = Temperature 5 name
t6$        = Temperature 6 name
tw$        = Timer 1 name
tt$        = Timer 2 name
th$        = Timer 3 name
ti$        = Time of day (HHMMSS)
pt$        = Type of blood pump, either centrifugal or
             roller or pulsatile
nr         = Number of rollers
td         = Tubing diameter
du$        = Tubing diameter units, either cm or in.
```

```
al        = Tubing arc length
au$       = Tubing arc length units, either cm or in.
pv        = Pump volume in ml
sv        = Stoke volume in cc
z         = Variable screen location
x$        = Variable character string
x         = Length of character string
u         = maximum length of character string
```

Listing

`1 print"E♡":sys 41056:?`	Clears the screen and makes the characters white:Displays message 1 on the screen:Prints a carriage return.
`2 sys 41107:?:input"name";pn$: gosub500`	Displays message 4 on the screen:Prints a carraige return:Inputs the patient's name:Asks if the answer is correct.
`3 if y$="n" then 2`	If the answer is no, then inputs the name again.
`4 if y$="y" then 6`	If the answer is yes, then inputs the patient's height.
`5 gosub600:go to 3`	The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.
`6 sys 41107:?:input"height";ph: gosub 500`	Displays message 4:Prints a carriage return:Inputs the patient's height:Asks if the answer is correct.
`7 if y$="n" then 6`	If the answer is no, then inputs the height again.
`8 if y$="y" then 10`	If the answer is yes, then inputs the height unit.
`9 gosub600:goto 7`	The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.
`10 sys 41124:?:input"or in";hu$: gosub500`	Displays message 5:Prints a carriage return:Inputs the height unit:Asks if the answer is correct.
`11 if y$="n" then 10`	If the answer is no, then inputs the height unit again

```
12 if y$="y" then 14
```
If the answer is yes, then inputs the weight.

```
13 gosub 600:goto 11
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
14 gosub 700
```
Checks for a valid height unit, and inputs again if invalid.

```
15 sys 41107:?:input"weight";pw:
   gosub 500
```
Displays message 4:Prints a carriage return:Inputs the patient's weight:Asks if the answer is correct.

```
16 if y$="n" then 15
```
If the answer is no, then inputs the weight again.

```
17 if y$="y" then 19
```
If the answer is yes, then inputs the weight unit.

```
18 gosub 600:goto 16
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
19 sys 41141:?:input"or lb";wu$:
   gosub 500
```
Displays message 6:Prints a carriage return:Inputs the weight unit:Asks if the answer is correct.

```
20 if y$="n" then 19
```
If the answer is no, then inputs the weight unit again.

```
21 if y$="y" then 23
```
If the answer is yes, then checks the validity of the weight unit entered.

```
22 gosub 600:goto 20
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
23 if wu$="kg" then 27
```
If the weight unit is kg, then calculates the body surface area.

```
24 if wu$="lb" then 26
```
If the weight unit is lb, then converts the weight to kg.

```
25 sys 41090:?:input"in kg
   or lb";wu$:goto 23
```
Displays message 3:Prints a carriage return:Inputs the weight unit again:Checks its validity.

```
26 pw=pw*0.45359237
```
Converts the weight to kg.

```
27 sa=pw↑0.425*ph↑0.725*0.007184
```
Calculates the body surface area using the Dubois formula.

```
28 sys 41390:?
```
Displays message 18:Prints a carriage return.

```
29 sys 41407:?:input"1
   measuring";t1$:gosub 500
```
Displays message 19:Prints a carriage return:Inputs the name of temperature 1: Asks if the answer is correct.

```
30 if y$="y" then 33
```
If the answer is yes, then inputs the name of temperature 2.

```
31 if y$="n" then 29
```
If the answer is no, then inputs the name of temperature 1 again.

```
32 gosub 600:goto 30
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
33 sys 41407:?:input"2
   measuring";t2$:gosub 500
```
Displays message 19:Prints a carriage return:Inputs the name of temperature 2: Asks if the answer is correct.

```
34 if y$="y" then 37
```
If the answer is yes, then inputs the name of temperature 3.

```
35 if y$="n" then 33
```
If the answer is no, then inputs the name of temperature 2 again.

```
36 gosub 600:goto 34
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
37 sys 41407:?:input"3
   measuring";t3$:gosub 500
```
Displays message 19:Prints a carriage return:Inputs the name of temperature 3: Asks if the answer is correct.

```
38 if y$="y" then 41
```
If the answer is yes, then inputs the name of temperature 4.

```
39 if y$="n" then 37
```
If the answer is no, then inputs the name of temperature 3 again.

```
40 gosub 600:goto 38
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
41 sys 41407:?:input"4
   measuring";t4$:gosub 500
```
Displays message 19:Prints a carriage return:Inputs the name of temperature 4: Asks if the answer is correct.

```
42 if y$="y" then 45
```
If the answer is yes, then inputs the name of temperature 5.

```
43 if y$="n" then 41
```
If the answer is no, then inputs the name of temperature 4 again.

```
44 gosub 600:goto 42
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
45 sys 41407:?:input"5
   measuring";t5$:gosub 500
```
Displays message 19:Prints a carriage return:Inputs the name of temperature 5: Asks if the answer is correct.

```
46 if y$="y" then 49
```
If the answer is yes, then inputs the name of temperature 6.

```
47 if y$="n" then 45
```
If the answer is no, then inputs the name of temperature 5 again.

```
48 gosub 600:goto 46
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
49 sys 41407:?:input"6
   measuring";t6$:gosub 500
```
Displays message 19:Prints a carriage return:Inputs the name of temperature 6: Asks if the answer is correct.

```
50 if y$="y" then 53
```
If the answer is yes, then checks to see if temperature 1 is connected.

```
51 if y$="n" then 49
```
If the answer is no, then inputs the name of temperature 6 again.

```
52 gosub 600:goto 50
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
53 if t1$="N/C" then poke
   7656,255
```
If temperature 1 is not connected, then sets the temperature 1 not connected flag.

```
54 if t4$="N/C" then poke
   7659,255
```
If temperature 4 is not connected, then sets the temperature 4 not connected flag.

```
55 sys 41424:?:sys 41441:?:
   if t2$="N/C" then poke
   7657,255
```
Displays message 20:Prints a carriage return:Displays message 21:Prints a carriage return:If temperature 2 is not connected, then sets the temperature 2 not connected flag.

```
56 sys 41458:?:input"1
   measuring";tw$:gosub 500:
   if t3$="N/C" then poke
   7658,255
```
Displays message 22:Prints a carriage return:Inputs the name of timer 1:Asks if the answer is correct:If temperature 3 is not connected, then sets the temperature 3 not connected flag.

```
57 if y$="y" then 60
```
If the answer is yes, then inputs the name of timer 2.

```
58 if y$="n" then 56
```
If the answer is no, then inputs the name of timer 1 again.

```
59 gosub 600:goto 57
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
60 sys 41458:?:input"2
   measuring";tt$:gosub 500:
   if t5$="N/C" then poke
   7660,255
```
Displays message 22:Prints a carriage return:Inputs the name of timer 2:Asks if the answer is correct:If temperature 5 is not connected, then sets the temperature 5 not connected flag.

```
61 if y$="y" then 64
```
If the answer is yes, then inputs the name of timer 3.

```
62 if y$="n" then 60
```
If the answer is no, then inputs the name of timer 2 again.

```
63 gosub 600:goto 61
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
64 sys 41458:?:input"3
   measuring";th$:gosub 500:
   if t6$="N/C" then poke
   7661,255
```
Displays message 22:Prints a carriage return:Inputs the name of timer 3:Asks if the answer is correct:If temperature 6 is not connected, then sets the temperature 6 not connected flag.

```
65 if y$="y" then 68
```
If the answer is yes, then inputs the time of day.

```
66 if y$="n" then 64
```
If the answer is no, then inputs the name of timer 3 again.

```
67 gosub 600:goto 65
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
68 sys 41852:?:input"time
   (HHMMSS)";ti$:goto 132
```
Displays message 23:Prints a carriage return:Inputs the time of day:Asks if the answer is correct.

```
69 sys 41158:?:input"pump
   being used";pt$:gosub 500
```
Displays message 7:Prints a carriage return:Inputs the type of blood pump being used:Asks if the answer is correct.

```
70 if y$="y" then 73
```
If the answer is yes, then checks for a centrifugal pump being used.

```
71 if y$="n" then 69
```
If the answer is no, then inputs the type of blood pump again.

```
72 gosub 600:goto 70
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
73 if pt$="centrifugal" then 75
```
If a centrifugal pump is being used, then sets the centrifugal flag, and displays the monitor screen with CENTRIFUGAL PUMP displayed in the right location.

```
74 goto 76
```
Check for a roller pump being used.

```
75 sys 8810:sys 41192:goto 111
```
Displays the monitor screen:Sets the centrifugal flag and displays message 9:Prints the patient's and timers' names.

```
76 if pt$="roller" then 79
```
If the roller pump is being used, then inputs roller pump data.

```
77 if pt$="pulsatile" then 106
```
If the pulsatile pump is being used, then inputs pulsatile pump data.

```
78 sys 41090:?:sys 41175:?:
   input"roller, or pulsatile";
   pt$:goto 73
```
Displays message 3:Prints a carriage return:Displays message 8:Prints a carriage return:Inputs the pump type again:Checks the validity of the response.

```
79 sys 41222:?:input"are there"
   ;nr:gosub 500
```
Displays message 10:Prints a carriage return:Inputs the number of rollers:Asks if the answer is correct.

```
80 if y$="y" then 83
```
If the answer is yes, then inputs the tubing diameter

```
81 if y$="n" then 79
```
If the answer is no, then inputs the number of rollers again.

```
82 gosub 600:goto 80
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
83 sys 41239:?:input"diameter"
   ;td:gosub 500
```
Displays message 11:Prints a carriage return:Inputs the tubing diameter:Asks if the answer is correct.

```
84 if y$="y" then 87
```
If the answer is yes, then inputs the tubing diameter unit.

```
85 if y$="n" then 83
```
If the answer is no, then inputs the tubing diameter again.

```
86 gosub 600:goto 84
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
87 sys 41256:?:input"cm or in"
   ;du$:gosub 500
```
Displays message 12:Prints a carriage return:Inputs the diameter unit:Asks if the answer is correct.

```
88 if y$="y" then 91
```
If the answer is yes, then checks the validity of the diameter unit.

```
89 if y$="n" then 87
```
If the answer is no, then inputs the diameter unit again.

```
90 gosub 600:goto 88
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
91 if du$="cm" then 94
```
If the diameter unit is cm, then inputs the arc length.

```
92 if du$="in" then td=td*2.54:
   goto 94
```
If the diameter unit is in, then converts the diameter to cm and inputs the arc length.

```
93 sys 41090:?:input"in cm or
   in";du$:goto 91
```
Displays message 3:Prints a carriage return:Inputs the diameter unit again:Checks the validity of the response.

```
94 sys 41273:?:input"length";
   al:gosub 500
```
Displays message 13:Prints a carriage return:Asks if the answer is correct.

```
95 if y$="y" then 98
```
If the answer is yes, then inputs the arc length unit.

```
96 if y$="n" then 94
```
If the answer is no, then inputs the arc length again.

```
97 gosub 600:goto 95
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
98 sys 41290:?:input"in cm or
   in";au$:gosub 500
```
Displays message 14:Prints a carriage return:Inputs the arc length unit:Asks if the answer is correct.

```
99 if y$="y" then 102
```
If the answer is yes, then checks for a valid unit.

```
100 if y$="n" then 98
```
If the answer is no, then inputs the arc length unit again.

```
101 gosub 600:goto 99
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
102 if au$="cm" then 105
```
If the arc length is in cm, then calculates the pump volume.

```
103 if au$="in" then al=al*2.54:
    goto 105
```
If the arc length is in inches, then convert the arc length to cm:Calculates the pump volume.

```
104 sys 41090:?:input"in cm or
    in";au$:goto 102
```
Displays message 3:Prints a carriage return:Inputs the arc length unit again: Checks the validity of the response.

```
105 pv=nr*al*3.1416*(td↑2)/4:
    sys 8810:sys 41307:goto 111
```
Calculates the pump volume: Displays the monitor screen:Sets the roller pump flag, displays message 15, and enables the pulse counter on the 6522-1: Prints the patient and timer names.

```
106 sys 41340:?:input"volume in
    cc";sv:gosub 500
```
Displays message 16:Prints a carriage return:Inputs the stroke volume:Asks if the answer is correct.

```
107 if y$="y" then 110
```
If the answer is yes, then displays the monitor screen and message 17, sets the pulsatile flag and enables the pulse counter on the 6522-1.

```
108 if y$="n" then 106
```
If the answer is no, then inputs the stroke volume again.

```
109 gosub 600:goto 107
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

```
110 sys 8810:sys 41357
```
Displays the monitor screen:Displays message 17, sets the pulsatile flag and enables the pulse counter on the 6522-1.

```
111 x$=pn$:z=7680:u=21:gosub
    800:x$=tw$:z=8120:u=6:
    gosub 800:x$=tt$:z=8142:
    gosub 800
```
:::Displays the patient name on the monitor screen: :::Displays the name of timer 1:::Displays the name of timer 2.

```
112 x$=th$:z=8164:gosub 800:
    x$=t1$:z=7968:u=8:gosub
    800:x$=t2$:z=7979:gosub
    800:x$=t3$
```
::Displays the name of timer 3::::Displays the name of temperature 1::: Displays the name of temperature 2:

```
113 z=8012:gosub 800:x$=t4$:
    z=8023:gosub 800:x$=t5$:
    z=8056:gosub 800:x$=t6$:
    z=8067:gosub 800
```
:Displays the name of temperature 3:::Displays the name of temperature 4::: Displays the name of temperature 5:::Displays the name of temperature 6.

```
114 sys 41902:sys 41942:sys
    41982:sys 8682:x=ph:z=7711:
    gosub 900:x=ph/2.54:z=7733:
    gosub 900
```

Displays timer 1:Displays timer 2:Displays timer 3: Prints the temperature degree unit and clears the display for unconnected temperature probes:Displays the patient's height in cm: Displays the patient's height in inches.

```
115 x=pw:z=7755:gosub 900:x=pw/
    0.45359237:z=7777:gosub 900:
    u=int(sa):poke 7822,u+48
```

::Displays the patient's weight in kg::Displays the weight in lb:Gets the unit's place of the surface area:Displays it.

```
116 t=int((sa-u)*10):poke 7824,
    t+48:h=int((sa-u-t/10)*100):
    poke 7825,h+48
```

Gets the tenth's digit of the surface area:Prints it: Gets the hundredth's digit of the surface area:Prints it.

```
117 if (sa-u-t/10-h/100)<0.005
    then 122
```

If the thousandth's digit is less than 5, then enables the clock and temperature update flags.

```
118 h=h+1:if h<10 then 121
```

The throusandth's digit is greater than 5, so increments the hundredth's digit:if the hundredth's digit is less than 10, then updates the surface area display.

```
119 h=0:t=t+1:if t<10 then 121
```

The hundredth's digit was ten, so clears the digit: Increments the tenth's digit:If the tenth's digit is less than 10, then updates the surface area display.

```
120 t=0:u=u+1
```

The tenth's digit was 10, so clears the digit:Increments the unit's digit.

```
121 poke 7822,u+48:poke 7824,
    t+48:poke7825,h+48
```

Displays the unit's digit of the surface area:Displays the tenth's digit of the surface area:Displays the hundredth's digit of the surface area.

```
122 sys 41475:gosub 913
```

Enables the clock and temperature update flags: Displays the clock.

```
123 sys 41484:sys 41493:if
    peek(7621)=0 then 128
```
Sets the cursor location: If the pump output update flag is not set, then checks the menu flag.

```
124 poke 7621,0:if peek(7622)
    =255 then gosub 920
```
Clears the pump output update flag:If the roller flag is set then updates the roller pump output and displays.

```
125 if peek(7623)=255 then
    gosub 930
```
If the pulsatile flag is set, then updates the pulsatile pump output and displays.

```
126 if peek(7624)=255 then
    gosub 940
```
If the centrifugal flag is set, then updates and displays the centrifugal pump output.

```
127 x=po:z=7866:gosub 900:
    x=po/pw:z=7904:gosub 900:
    x=po/sa:z=7926:gosub 900
```
::Displays the pump output:
::Displays the blood flow rate in ml/min/kg:::
Displays the blood flow rate in ml/min/m**2.

```
128 if peek(7662)=0 then 130
```
If the menu flag is not set then checks the temperature degree units and the temperature and print clock update flags.

```
129 poke 7662,0:goto 1
```
The menu flag is set so clears the flag:Returns to the input section.

```
130 sys 41807:sys 16384:if
    peek(7650)=0 then 123
```
Checks the temperature degree units and updates them if changed:Checks the temperature update flag and updates the temperatures if it set:If the print clock flag is not set then returns to statement 123 to set the position of the cursor.

```
131 poke 7650,0:gosub 913:
    goto 123
```
The print clock flag is set so clears the flag:Updates the clock on the screen: Returns to statement 123 to set the cursor location.

```
132 gosub 500
```
Asks if the the answer is correct.

```
133 if y$="y" then 69
```
If the answer is yes, then inputs the type of blood pump.

```
134 if y$="n" then 68
```
If the answer is no, then inputs the time of day again.

```
135 gosub 600:goto 133
```
The answer is invalid, so asks again if the answer is correct:Checks the validity of the response.

Subroutines

```
400 if th=0 then poke z,32:goto
    402
```
If the most significant digit is 0 then clears the digit on the screen:Checks the second most significant digit.

```
401 poke z,th+48
```
Displays the most significant digit.

```
402 if h=0 and th=0 then poke
    z+1,32:goto 404
```
If both the most and the next most significant digits are zero, then clears the next most significant digit on the screen: Checks the third most significant digit.

```
403 poke z+1,h+48
```
Displays the next most significant digit.

```
404 if t=0 and th=0 and h=0
    then poke z+2,32:goto 406
```
If the 3 most significant are zero, then clears the third most significant digit:Displays then least significant digit.

```
405 poke z+2,t+48
```
Displays the third most significant digit.

```
406 poke z+3,u+48:return
```
Displays the least significant digit:Returns from subroutine.

```
500 sys 41073:?:input"(y/n)";
    y$:return
```
Displays message 2:Prints a carriage return:Inputs the answer to "Is this correct?":Returns from subroutine.

```
600 sys 41090:?:input"in y or
    n";y$:return
```
Displays message 3:Prints a carriage return:Inputs the answer to "Invalid entry, type in y or n?":Returns from subroutine.

```
700 if hu$="cm" then return
```
If the height unit is cm then returns from subroutine.

```
710 if hu$="in" then 740
```
If the height unit is in. then converts the height to cm and returns from subroutine.

```
720 sys 41090:?:input"in cm or
    in";hu$:goto 700
```
Displays message 3:Prints a carriage return:Inputs the height unit:Checks the validity of the response.

```
800 y=1:x=len(x$):if x>u then
    x=u
```
Sets the pointer, y, to the first character:Calculates the length of the character string:If the string is too long, then resets the string length.

```
801 w=asc(mid$(x$,y,1))-64:if
    w<0 then w=w+64
```
Calculates the screen code for a character:Converts the screen code if necessary.

```
802 if w>128 then w=w-64
```
Converts the screen code if necessary.

```
803 poke z+y,w:y=y+1:x=x-1:
    if x>0 then 801
```
Displays a character on the screen:Increments the character pointer:Decrements the length of the string: Displays another character if not through with string.

```
804 return
```
Returns from subroutine.

```
900 th=int(x/1000):h=int((x-
    th*1000)/100):t=int((x-
    th*1000-h*100)/10)
```
Gets the thousand's digit: Gets the hundred's digit: Gets the ten's digit.

```
901 u=int(x-th*1000-h*100-t*10)
```
Gets the unit's digit.

```
902 if (x-th*1000-h*100-t*10-u)
    <0.5 then 907
```
If the tenth's digit is less than 5 then, displays the number on the screen.

```
903 u=u+1:if u<10 then 907
```
The tenth's digit is more than 5 so increments the unit's digit:If the unit's digit is less than 10 then displays the number.

```
904 t=t+1:u=0:if t<10 then 907
```
Increments the ten's digit: Clears the unit's digit:If the ten's digit is less than 10 then displays the number.

```
905 h=h+1:t=0:ifh<10 then 907      Increments the hundred's
                                   digit:Clears the ten's
                                   digit:If the hundred's
                                   digit is less than 10 then
                                   displays the number.
906 th=th+1:h=0                    Increments the thousand's
                                   digit:Clears the hundred's
                                   digit.
907 gosub 400:return               Displays the number:Returns
                                   from the subroutine.

913 poke 8112,asc(mid$(ti$,1,1):   Displays the high byte of
    poke 8113,asc(mid$(ti$,2,1):   the clock hours:Displays
    poke 8115,asc(mid$(ti$,3,1):   the low byte of the clock
                                   hours:Displays the high
                                   byte of the clock minutes.
914 poke 8116,asc(mid$(ti$,4,1):   Displays the low byte of
    poke 8118,asc(mid$(ti$,5,1):   the clock minutes:Displays
    poke 8119,asc(mid$(ti$,6,1)    the high byte of the clock
                                   seconds:Displays the low
                                   byte of the clock seconds.
915 return                         Returns from subroutine.

920 po=pv*peek(7653)*1.2:return    Calculates roller pump out-
                                   put:Returns from subroutine.

930 po=sv*peek(7653)*120:return    Calculates pulsatile pump
                                   output:Returns from subrou-
                                   tine.

940 poke39426,128:sys42022         Enables centrifugal flow
                                   probe:Gets value from A/D
                                   converter.
941 po=(peek(7667)-48)*1000+       Calculates centrifugal pump
    (peek(7669)-48)*100+(peek      output
    (7669)-48)*10+(peek(7670)-
    48)
942 return                         Returns from subroutine
```

Assembly Language Listing

In the following listing, the hexadecimal address, XXXX, of the upcoming machine code byte is noted by ==XXXX. The hexadecimal machine code is shown in the left hand column. The assembly language mneumonic and operand are shown in the middle column and comments are listed following a semicolon in the right hand column. The assembler directive, *=$XXXX, causes the assembled program's machine code to start at

hexadecimal address XXXX. The abbreviation LB is used to signify low byte and the abbreviation HB is used to signify HB. A prefix of a dollar sign, $, indicates a hexadecimal number while no prefix indicates a decimal number.

```
==0000
        *=$A000
;EQUATE STATEMENTS
==A000   PMPFLG=$1DC5    ;  Pump output update flag
                             0 = don't update / $FF = update
==A000   ROLFLG=$1DC6    ;  Roller pump flag
                             0 = is not / $FF = is
==A000   PULFLG=$1DC7    ;  Pulsatile pump flag
==A000   CENFLG=$1DC8    ;  Centrifugal pump flag
==A000   CENCNT=$1DC9    ;  1/2 sec counter for updating cen-
                             trifugal pump output
==A000   TM1FLG=$1DCA    ;  Timer 1 update flag
==A000   TMR1HH=$1DCB    ;  Timer 1 hours high byte
==A000   TMR1HL=$1DCC    ;  Timer 1 hours low byte
==A000   TMR1MH=$1DCD    ;  Timer 1 minutes high byte
==A000   TMR1ML=$1DCE    ;  Timer 1 minutes low byte
==A000   TMR1SH=$1DCF    ;  Timer 1 seconds high byte
==A000   TMR1SL=$1DD0    ;  Timer 1 seconds low byte
==A000   TMR1JF=$1DD1    ;  Timer 1 jiffy (1/60 sec) counter
==A000   TM2FLG=$1DD2    ;  Timer 2 update flag
==A000   TMR2HH=$1DD3    ;  Timer 2 hours high byte
==A000   TMR2HL=$1DD4    ;  Timer 2 hours low byte
==A000   TMR2MH=$1DD5    ;  Timer 2 minutes high byte
==A000   TMR2ML=$1DD6    ;  Timer 2 minutes low byte
==A000   TMR2SH=$1DD7    ;  Timer 2 seconds high byte
==A000   TMR2SL=$1DD8    ;  Timer 2 seconds low byte
==A000   TMR2JF=$1DD9    ;  Timer 2 jiffy counter
==A000   TM3FLG=$1DDA    ;  Timer 3 update flag
==A000   TMR3HH=$1DDB    ;  Timer 3 hours high byte
==A000   TMR3HL=$1DDC    ;  Timer 3 hours low byte
==A000   TMR3MH=$1DDD    ;  Timer 3 minutes high byte
==A000   TMR3ML=$1DDE    ;  Timer 3 minutes low byte
==A000   TMR3SH=$1DDF    ;  Timer 3 seconds high byte
==A000   TMR3SL=$1DE0    ;  Timer 3 seconds low byte
==A000   TMR3JF=$1DE1    ;  Timer 3 jiffy counter
==A000   PRCLKF=$1DE2    ;  Print clock flag
==A000   CLKFLG=$1DE3    ;  Update screen clock flag
==A000   CLKJIF=$1DE4    ;  Clock jiffy counter
==A000   PULSES=$1DE5    ;  Pulses counted per 1/2 sec
==A000   TMPFLG=$1DE6    ;  Update temperatures flag
==A000   TMPCNT=$1DE7    ;  4 sec counter for updating temp-
                             eratures
==A000   TMP1NC=$1DE8    ;  Temperature 1 flag
                             0 = connected/$FF = not connected
```

```
==A000    TMP2NC=$1DE9    ;  Temperature 2 flag
==A000    TMP3NC=$1DEA    ;  Temperature 3 flag
==A000    TMP4NC=$1DEB    ;  Temperature 4 flag
==A000    TMP5NC=$1DEC    ;  Temperature 5 flag
==A000    TMP6NC=$1DED    ;  Temperature 6 flag
==A000    MENFLG=$1DEE    ;  Menu flag
                             0 = don't return to input
                             $FF = return to input
==A000    DEGUNT=$1DEF    ;  Temperature degree unit
                             0 =  C / $FF =  F
==A000    MSGADL=$FB      ;  Low byte of starting address of
                             message to be displayed
==A000    MSGADH=$FC      ;  High byte of starting address of
                             message to be displayed
==A000    NOCHAR=$1DF0    ;  Number of characters in message
==A000    MSGPRL=$FD      ;  Low byte of starting screen loca-
                             tion of message
==A000    MSGPRH=$FE      ;  High byte of starting screen loca-
                             tion of message
==A000    CHRCNT=$1DF9    ;  Character counter for input
==A000    MEMORY=$1DFB    ;  Temporary storage of input char-
                             acters
==A000    HALFCT=$1DF1    ;  1/2 sec counter for pulse counter
==A000    IO1ACR=$980B    ;  6522-1 Auxilliary control register
==A000    IO1IFR=$980D    ;  6522-1 Interrupt flag register
==A000    IO1IER=$980E    ;  6522-1 Interrupt enable register
==A000    IO1T2L=$9808    ;  6522-1 Timer 2 low byte latch
==A000    IO1T2H=$9809    ;  6522-1 Timer 2 high byte latch
==A000    INTH=$20        ;  Interrupt bypass address-high byte
==A000    INTL=$00        ;  Interrupt bypass address-low byte
==A000    M1L=$F5         ;  Message 1 starting address, LB
==A000    M1H=$24         ;  Message 1 starting address, HB
==A000    M2L=$50         ;  Message 2 starting address, LB
==A000    M2H=$25         ;  Message 2 starting address, HB
==A000    M3L=$5F         ;  Message 3 starting address, LB
==A000    M3H=$25         ;  Message 3 starting address, HB
==A000    M4L=$72         ;  Message 4 starting address, LB
==A000    M4H=$25         ;  Message 4 starting address, HB
==A000    M5L=$87         ;  Message 5 starting address, LB
==A000    M5H=$25         ;  Message 5 starting address, HB
==A000    M6L=$9A         ;  Message 6 starting address, LB
==A000    M6H=$25         ;  Message 6 starting address, HB
==A000    M7L=$AD         ;  Message 7 starting address, LB
==A000    M7H=$25         ;  Message 7 starting address, HB
==A000    M8L=$D1         ;  Message 8 starting address, LB
==A000    M8H=$25         ;  Message 8 starting address, HB
==A000    M9L=$E0         ;  Message 9 starting address, LB
==A000    M9H=$25         ;  Message 9 starting address, HB
==A000    M10L=$F0        ;  Message 10 starting address, LB
==A000    M10H=$25        ;  Message 10 starting address, HB
```

```
==A000    M11L=$00        ;  Message 11 starting address, LB
==A000    M11H=$26        ;  Message 11 starting address, HB
==A000    M12L=$12        ;  Message 12 starting address, LB
==A000    M12H=$26        ;  Message 12 starting address, HB
==A000    M13L=$24        ;  Message 13 starting address, LB
==A000    M13H=$26        ;  Message 13 starting address, HB
==A000    M14L=$33        ;  Message 14 starting address, LB
==A000    M14H=$26        ;  Message 14 starting address, HB
==A000    M15L=$44        ;  Message 15 starting address, LB
==A000    M15H=$26        ;  Message 15 starting address, HB
==A000    M16L=$4F        ;  Message 16 starting address, LB
==A000    M16H=$26        ;  Message 16 starting address, HB
==A000    M17L=$61        ;  Message 17 starting address, LB
==A000    M17H=$26        ;  Message 17 starting address, HB
==A000    M18L=$6F        ;  Message 18 starting address, LB
==A000    M18H=$26        ;  Message 18 starting address, HB
==A000    M19L=$32        ;  Message 19 starting address, LB
==A000    M19H=$27        ;  Message 19 starting address, HB
==A000    M20L=$45        ;  Message 20 starting address, LB
==A000    M20H=$27        ;  Message 20 starting address, HB
==A000    M21L=$88        ;  Message 21 starting address, LB
==A000    M21H=$27        ;  Message 21 starting address, HB
==A000    M22L=$D5        ;  Message 22 starting address, LB
==A000    M22H=$27        ;  Message 22 starting address, HB
==A000    M23L=$E2        ;  Message 23 starting address, LB
==A000    M23H=$27        ;  Message 23 starting address, HB
==A000    DIGSEL=$1DF2    ;  Selects digit during A/D conver-
                             sion
==A000    TEMPH=$1DF3     ;  Temperature - hundred's digit
==A000    TEMPT=$1DF4     ;  Temperature - ten's digit
==A000    TEMPU=$1DF5     ;  Temperature - unit's digit
==A000    TEMPTH=$1DF6    ;  Temperature - tenth's digit
==A000    COUNT=$1DF7     ;  Counter used in subroutine SCREN1
==A000    CNT=$1DF8       ;  Counter used in subroutine SCREN2
;INITIALIZATION ROUTINE
09        .BYT 09,$A0,0   ;  Auto-start sequence
  9,$A0,$41,$30,$C3,$C
  2,$CD
A0
09                        ;  Start of program - LB
A0                        ;  Start of program - HB
41                        ;  Signature bytes
30
C3
C2
CD
A902      LDA #02         ;  Accesses DDRB on 6520-1
8D039A    STA $9A03       ;  Control register B on 6520-1
A9FE      LDA #$FE        ;  All bits outputs except bit 0
```

```
==A010
8D029A    STA $9A02       ;  DDRB on 6520-1
A906      LDA #06         ;  Accesses port B on 6520-1
8D039A    STA $9A03       ;  Control register B on 6520-1
A900      LDA #00         ;  Initializes outputs to zero
8D029A    STA $9A02
208DFD    JSR $FD8D       ;  RAM clear and memory check
==A020
2052FD    JSR $FD52       ;  Sets VIC 20 vectors
20F9FD    JSR $FDF9       ;  Initializes VIC 20 I/O
2018E5    JSR $E518       ;  Initializes screen
A920      LDA #$20        ;  Initializes 6522-1 ACR
8D0B98    STA IO1ACR
A904      LDA #04         ;  Accesses port A on 6520-1
==A030
8D019A    STA $9A01       ;  Control register A on 6520-1
A9FF      LDA #$FF        ;  Initializes low byte of timer 2
8D0898    STA IO1T2L        latch
20BA40    JSR DISABL      ;  Disables all flags, initializes
                             constants
A920      LDA #INTH       ;  Sets interrupt bypass address
8D1503    STA $0315
==A040
A900      LDA #INTL
8D1403    STA $0314
58        CLI             ;  Allows interrupts
2081A4    JSR $A481       ;  Displays title screen (SCREN1)
A930      LDA #$30        ;  18 sec time delay
8DFA1D    STA $1DFA
==A04E    DCRM
A2FF      LDX #$FF
==A050    DCRX
A0FF      LDY #$FF
==A052    DCRY
88        DEY
D0FD      BNE DCRY
CA        DEX
D0F8      BNE DCRX
CEFA1D    DEC $1DFA
D0F1      BNE DCRM
6C00C0    JMP ($C000)     ;  Jumps to basic
==A060    MSG1            ;  Prints first message
A9F5      LDA #M1L        ;  Sets starting address, LB
85FB      STA MSGADL
A924      LDA #M1H        ;  Sets starting address, HB
85FC      STA MSGADH
A95B      LDA #$5B        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
```

```
==A070
60        RTS             ;  Returns from subroutine
==A071    MSG2            ;  Prints second message
A950      LDA #M2L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M2H        ;  Sets starting address, HB
85FC      STA MSGADH
A90F      LDA #$0F        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A081
60        RTS             ;  Returns from subroutine
==A082    MSG3            ;  Prints third message
A95F      LDA #M3L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M3H        ;  Sets starting address, HB
85FC      STA MSGADH
A913      LDA #$13        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A092
60        RTS             ;  Returns from subroutine
==A093    MSG4            ;  Prints fourth message
A972      LDA #M4L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M4H        ;  Sets starting address, HB
85FC      STA MSGADH
A915      LDA #$15        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A0A3
60        RTS             ;  Returns from subroutine
==A0A4    MSG5            ;  Prints fifth message
A987      LDA #M5L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M5H        ;  Sets starting address, HB
85FC      STA MSGADH
A913      LDA #$13        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A0B4
60        RTS             ;  Returns from subroutine
==A0B5    MSG6            ;  Prints sixth message
A99A      LDA #M6L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M6H        ;  Sets starting address, HB
85FC      STA MSGADH
A913      LDA #$13        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
```

```
==A0C5
60        RTS             ;  Returns from subroutine
==A0C6    MSG7            ;  Prints seventh message
A9AD      LDA #M7L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M7H        ;  Sets starting address, HB
85FC      STA MSGADH
A924      LDA #$24        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A0D6
60        RTS             ;  Returns from subroutine
==A0D7    MSG8            ;  Prints eighth message
A9D1      LDA #M8L        ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M8H        ;  Sets starting address, HB
85FC      STA MSGADH
A90F      LDA #$0F        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A0E7
60        RTS             ;  Returns from subroutine
==A0E8    CENPMP          ;  Centrifugal pump being used

A9FF      LDA #$FF        ;  Sets centrifugal flag
8DC81D    STA CENFLG
A9E0      LDA #M9L        ;  Sets starting address for message
                             9, LB
85FB      STA MSGADL
A925      LDA #M9H        ;  Sets starting address, HB
85FC      STA MSGADH
A99B      LDA #$9B        ;  Sets starting screen location, LB
85FD      STA MSGPRL
==A0F9
A91E      LDA #$1E        ;  Sets starting screen location, HB
85FE      STA MSGPRH
A910      LDA #$10        ;  Sets number of characters
8DF01D    STA NOCHAR
209BA3    JSR $A39B       ;  Displays message (MSGE)
60        RTS             ;  Returns from subroutine
==A106    MSG10           ;  Prints tenth message
A9F0      LDA #M10L       ;  Sets starting address, LB
85FB      STA MSGADL
A925      LDA #M10H       ;  Sets starting address, HB
85FC      STA MSGADH
A910      LDA #$10        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A116
60        RTS             ;  Returns from subroutine
```

```
==A117   MSG11          ;  Prints eleventh message
A900     LDA #M11L      ;  Sets starting address, LB
85FB     STA MSGADL
A926     LDA #M11H      ;  Sets starting address, HB
85FC     STA MSGADH
A912     LDA #$12       ;  Sets number of characters
8DF01D   STA NOCHAR
208DA3   JSR $A38D      ;  Displays message (DISPLY)
==A127
60       RTS            ;  Returns from subroutine
==A128   MSG12          ;  Prints twelfth message
A912     LDA #M12L      ;  Sets starting address, LB
85FB     STA MSGADL
A926     LDA #M12H      ;  Sets starting address, HB
85FC     STA MSGADH
A912     LDA #$12       ;  Sets number of characters
8DF01D   STA NOCHAR
208DA3   JSR $A38D      ;  Displays message (DISPLY)
==A138
60       RTS            ;  Returns from subroutine
==A139   MSG13          ;  Prints thirteenth message
A924     LDA #M13L      ;  Sets starting address, LB
85FB     STA MSGADL
A926     LDA #M13H      ;  Sets starting address, HB
85FC     STA MSGADH
A90F     LDA #$0F       ;  Sets number of characters
8DF01D   STA NOCHAR
208DA3   JSR $A38D      ;  Displays message (DISPLY)
==A149
60       RTS            ;  Returns from subroutine
==A14A   MSG14          ;  Prints fourteenth message
A933     LDA #M14L      ;  Sets starting address, LB
85FB     STA MSGADL
A926     LDA #M14H      ;  Sets starting address, HB
85FC     STA MSGADH
A911     LDA #$11       ;  Sets number of characters
8DF01D   STA NOCHAR
208DA3   JSR $A38D      ;  Displays message (DISPLY)
==A15A
60       RTS            ;  Returns from subroutine
==A15B   MSG15          ;  Prints fifteenth message
A944     LDA #M15L      ;  Sets starting address, LB
85FB     STA MSGADL
A926     LDA #M15H      ;  Sets starting address, HB
85FC     STA MSGADH
A90B     LDA #$0B       ;  Sets number of characters
8DF01D   STA NOCHAR
A99B     LDA #$9B       ;  Sets starting screen location, LB
85FD     STA MSGPRL
```

```
==A16C
A91E      LDA #$1E        ;  Sets starting screen location, HB
85FE      STA MSGPRH
209BA3    JSR $A39B       ;  Displays message (MSGE)
A9FF      LDA #$FF        ;  Sets roller flag
8DC61D    STA ROLFLG
20A8A3    JSR $A3A8       ;  Enables pulse counter (PULSE)
60        RTS             ;  Returns from subroutine
==A17C    MSG16           ;  Prints sixteenth message
A94F      LDA #M16L       ;  Sets starting address, LB
85FB      STA MSGADL
A926      LDA #M16H       ;  Sets starting address, HB
85FC      STA MSGADH
A912      LDA #$12        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A18C
60        RTS             ;  Returns from subroutine
==A18D    MSG17           ;  Prints seventeenth message
20A8A3    JSR $A3A8       ;  Enables pulse counter (PULSE)
A9FF      LDA #$FF        ;  Sets pulsatile flag
8DC71D    STA PULFLG
A961      LDA #M17L       ;  Sets starting address, LB
85FB      STA MSGADL
A926      LDA #M17H       ;  Sets starting address, HB
85FC      STA MSGADH
==A19D
A90E      LDA #$0E        ;  Sets number of characters
8DF01D    STA NOCHAR
A99B      LDA #$9B        ;  Sets starting screen location, LB
85FD      STA MSGPRL
A91E      LDA #$1E        ;  Sets starting screen location, HB
85FE      STA MSGPRH
209BA3    JSR $A39B       ;  Displays message (MSGE)
==A1AD
60        RTS             ;  Returns from subroutine
==A1AE    MSG18           ;  Prints eighteenth message
A96F      LDA #M18L       ;  Sets starting address, LB
85FB      STA MSGADL
A926      LDA #M18H       ;  Sets starting address, HB
85FC      STA MSGADH
A9C3      LDA #$C3        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A1BE
60        RTS             ;  Returns from subroutine
==A1BF    MSG19           ;  Prints nineteenth message
A932      LDA #M19L       ;  Sets starting address, LB
85FB      STA MSGADL
A927      LDA #M19H       ;  Sets starting address, HB
```

```
85FC      STA MSGADH
A913      LDA #$13          ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D         ;  Displays message (DISPLY)
==A1CF
60        RTS               ;  Returns from subroutine
==A1D0    MSG20             ;  Prints twentieth message
A943      LDA #$43          ;  Sets number of characters
8DF01D    STA NOCHAR
A945      LDA #M20L         ;  Sets starting address, LB
85FB      STA MSGADL
A927      LDA #M20H         ;  Sets starting address, HB
85FC      STA MSGADH
208DA3    JSR $A38D         ;  Displays message (DISPLY)
==A1E0
60        RTS               ;  Returns from subroutine
==A1E1    MSG21             ;  Prints twenty-first message
A988      LDA #M21L         ;  Sets starting address, LB
85FB      STA MSGADL
A927      LDA #M21H         ;  Sets starting address, HB
85FC      STA MSGADH
A94D      LDA #$4D          ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D         ;  Displays message (DISPLY)
==A1F1
60        RTS               ;  Returns from subroutine
==A1F2    MSG22             ;  Prints twenty-second message
A9D5      LDA #M22L         ;  Sets starting address, LB
85FB      STA MSGADL
A927      LDA #M22H         ;  Sets starting address, HB
85FC      STA MSGADH
A90D      LDA #$0D          ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D         ;  Displays message (DISPLY)
==A202
60        RTS               ;  Returns from subroutine
==A203    ENABLE            ;  Enables clock and temperature up-
                               date flags
A9FF      LDA #$FF
8DE31D    STA CLKFLG
8DE61D    STA TMPFLG
60        RTS               ;  Returns from subroutine
==A20C    SETCUR            ;  Sets the cursor location
18        CLC
A202      LDX #02           ;  Sets row to 2
A012      LDY #$12          ;  Sets column to 18
20F0FF    JSR $FFF0         ;  Sets the cursor location
60        RTS               ;  Returns from subroutine
```

```
==A215   FETCH          ;  Checks for keyboard input and if
                           any checks its validity
A900     LDA #00        ;  Initializes character counter
8DF91D   STA CHRCNT
==A21A   LOOP1
20E4FF   JSR $FFE4      ;  Gets character from keyboard queue
C900     CMP #00        ;  Is it empty?
D00A     BNE STORE      ;  It isn't so branches
ADF91D   LDA CHRCNT     ;  It is so checks if character
                           counter is greater than or equal
                           to 1
C900     CMP #00
F04C     BEQ RETURN     ;  No characters so returns from sub-
                           routine
4C1AA2   JMP $A21A      ;  Is a character so loops and waits
==A22B   STORE
ACF91D   LDY CHRCNT     ;  Restores character counter
99FB1D   STA MEMORY,Y   ;  Stores character counter in memory
C8       INY            ;  Increments character counter
8CF91D   STY CHRCNT     ;  Saves character counter
C90D     CMP #$0D       ;  Carriage return?
F003     BEQ PRCHK      ;  If so, prints first 2 characters
                           and checks validity
4C1AA2   JMP $A21A      ;  If not, gets another character
==A23C   PRCHK
ADFB1D   LDA MEMORY     ;  Gets first character
20D2FF   JSR $FFD2      ;  Prints it
ADFC1D   LDA MEMORY+1   ;  Gets second character
20D2FF   JSR $FFD2      ;  Prints it
A901     LDA #01        ;  Colors the characters white
8D3E96   STA 38462
==A24D
8D3F96   STA 38463
ADFB1D   LDA MEMORY     ;  Gets first character
C953     CMP #$53       ;  Is it a s?
F01E     BEQ STRTIM     ;  It is so branches to start a timer
C952     CMP #$52       ;  Is it a r?
F03F     BEQ RESTIM     ;  It is so branches to reset a timer
C945     CMP #$45       ;  Is it an e?
==A25D
D003     BNE LOOP4      ;  If it's not checks for a m
4C16A3   JMP $A316      ;  It is so jumps to end a timer
==A262   LOOP4
C94D     CMP #$4D       ;  Is it a m?
D003     BNE CLRCOD     ;  If it's not clears the screen
4C3BA3   JMP $A33B      ;  It is so jumps to set menu flag
==A269   CLRCOD         ;  Clears the invalid command on the
                           screen
200CA2   JSR $A20C      ;  Sets cursor position
A920     LDA #$20       ;  Spaces over code
```

```
20D2FF    JSR $FFD2
20D2FF    JSR $FFD2
==A274    RETURN
60        RTS              ;  Returns from subroutine
==A275    STRTIM           ;  Starts a timer
ADFC1D    LDA MEMORY+1     ;  Gets timer number
C931      CMP #$31         ;  Is it 1?
F00C      BEQ TIMER1       ;  If it is branches to start timer 1
C932      CMP #$32         ;  Is it 2?
F00E      BEQ TIMER2       ;  If it is branches to start timer 2
C933      CMP #$33         ;  Is it 3?
F010      BEQ TIMER3       ;  If it is branches to start timer 3
2069A2    JSR $A269        ;  Invalid timer number, so jumps to
                              clear code
==A287
60        RTS              ;  Returns from subroutine
==A288    TIMER1           ;  Starts timer 1
A9FF      LDA #$FF         ;  Sets timer 1 flag
8DCA1D    STA TM1FLG
60        RTS              ;  Returns from subroutine
==A28E    TIMER2           ;  Starts timer 2
A9FF      LDA #$FF         ;  Sets timer 2 flag
8DD21D    STA TM2FLG
60        RTS              ;  Returns from subroutine
==A294    TIMER3           ;  Starts timer 3
A9FF      LDA #$FF         ;  Sets timer 3 flag
8DDA1D    STA TM3FLG
60        RTS              ;  Returns from subroutine
==A29A    RESTIM           ;  Resets a timer
ADFC1D    LDA MEMORY+1     ;  Gets timer number
C931      CMP #$31         ;  Is it a 1?
F00C      BEQ RT1          ;  If it is branches to reset timer 1
C932      CMP #$32         ;  Is it a 2?
F02B      BEQ RT2          ;  If it is branches to reset timer 2
C933      CMP #$33         ;  Is it a 3?
F04A      BEQ RT3          ;  If it is branches to reset timer 3
2069A2    JSR $A269        ;  Invalid timer numer, so jumps to
                              clear code
==A2AC
60        RTS              ;  Returns from subroutine
==A2AD    RT1              ;  Resets timer 1
A900      LDA #00          ;  Clears timer 1 flag and sets timer
                              1 to zero
8DCA1D    STA TM1FLG
8DD11D    STA TMR1JF
A930      LDA #$30         ;  ASCII code for zero
8DCB1D    STA TMR1HH
8DCC1D    STA TMR1HL
==A2BD
8DCD1D    STA TMR1MH
```

```
8DCE1D    STA TMR1ML
8DCF1D    STA TMR1SH
8DD01D    STA TMR1SL
20AEA3    JSR $A3AE       ;  Prints timer 1
200CA2    JSR $A20C       ;  Sets cursor location
==A2CF
60        RTS             ;  Returns from subroutine
==A2D0    RT2             ;  Resets timer 2
A900      LDA #00         ;  Clears timer 2 flag and sets timer
                             2 to zero
8DD21D    STA TM2FLG
8DD91D    STA TMR2JF
A930      LDA #$30        ;  ASCII code for zero
8DD31D    STA TMR2HH
8DD41D    STA TMR2HL
==A2E0
8DD51D    STA TMR2MH
8DD61D    STA TMR2ML
8DD71D    STA TMR2SH
8DD81D    STA TMR2SL
20D6A3    JSR $A3D6       ;  Prints timer 2
200CA2    JSR $A20C       ;  Sets cursor location
==A2F2
60        RTS             ;  Returns from subroutine
==A2F3    RT3             ;  Resets timer 3
A900      LDA #00         ;  Clears timer 3 flag and sets timer
                             3 to zero
8DDA1D    STA TM3FLG
8DE11D    STA TMR3JF
A930      LDA #$30        ;  ASCII code for zero
8DDB1D    STA TMR3HH
8DDC1D    STA TMR3HL
==A303
8DDD1D    STA TMR3MH
8DDE1D    STA TMR3ML
8DDF1D    STA TMR3SH
8DE01D    STA TMR3SL
20FEA3    JSR $A3FE       ;  Prints timer 3
200CA2    JSR $A20C       ;  Sets cursor location
==A315
60        RTS             ;  Returns from subroutine
==A316    ENDTIM          ;  Stops a timer
ADFC1D    LDA MEMORY+1    ;  Gets timer number
C931      CMP #$31        ;  Is it timer 1?
F00C      BEQ ET1         ;  If it is, branches to stop timer 1
C932      CMP #$32        ;  Is it timer 2?
F00E      BEQ ET2         ;  If it is, branches to stop timer 2
C933      CMP #$33        ;  Is it timer 3?
F010      BEQ ET3         ;  If it is, branches to stop timer 3
```

```
2069A2   JSR $A269       ;  Invalid timer number so clears
                            code on screen
==A328
60       RTS             ;  Returns from subroutine
==A329   ET1             ;  Stops timer 1
A900     LDA #00         ;  Clears timer 1 flag
8DCA1D   STA TM1FLG
60       RTS             ;  Returns from subroutine
==A32F   ET2             ;  Stops timer 2
A900     LDA #00         ;  Clears timer 2 flag
8DD21D   STA TM2FLG
60       RTS             ;  Returns from subroutine
==A335   ET3             ;  Stops timer 3
A900     LDA #00         ;  Clears timer 3 flag
8DDA1D   STA TM3FLG
60       RTS             ;  Returns from subroutine
==A33B   MENU            ;  Returns to input section
20BA40   JSR DISABL      ;  Disables all flags, clears all
                            constants
A9FF     LDA #$FF        ;  Sets menu flag
8DEE1D   STA MENFLG
60       RTS             ;  Returns from subroutine
ADE61D   LDA TMPFLG      ;  Gets temperature flag
C9FF     CMP #$FF        ;  Is it set?
D003     BNE RETRN       ;  If it's not returns from subrou-
                            tine
==A34B
200040   JSR UPTEMP      ;  It is set, so updates temperatures
==A34E   RETRN
60       RTS             ;  Returns from subroutine
AD029A   LDA $9A02       ;  Gets the temperature degree unit
                            from the 6520-1
2901     AND #01         ;  Masks off other bits
CDEF1D   CMP DEGUNT      ;  Compares with the previous unit
F0F5     BEQ RETRN       ;  If the same returns from subrou-
                            tine
8DEF1D   STA DEGUNT      ;  Saves the degree unit
C901     CMP #01         ;  It's not the same so is it  F?
==A35E
F004     BEQ F           ;  If it is branches to print F
A943     LDA #$43        ;  It's not F so gets screen code for
                            C
D002     BNE POKE        ;  Branches to print  C
==A364   F               ;  Prints  F
A946     LDA #$46        ;  Screen code for F
==A366   POKE            ;  Prints degree units
8D3D1F   STA 7997
8D481F   STA 8008
8D691F   STA 8041
8D741F   STA 8052
```

```
8D951F    STA 8085
8DA01F    STA 8096
==A378
200F22    JSR $220F       ;  Clears the degree unit if the
                             temperature probe is not connected
60        RTS             ;  Returns from subroutine
==A37C    MSG23           ;  Prints message 23
A9E2      LDA #M23L       ;  Sets starting address, LB
85FB      STA MSGADL
A927      LDA #M23H       ;  Sets starting address, HB
85FC      STA MSGADH
A913      LDA #$13        ;  Sets number of characters
8DF01D    STA NOCHAR
208DA3    JSR $A38D       ;  Displays message (DISPLY)
==A38C
60        RTS             ;  Returns from subroutine
==A38D    DISPLY          ;  Displays a message to the screen
                             at the current cursor location
A000      LDY #00         ;  Clears the character counter
==A38F    MOREC
B1FB      LDA (MSGADL),Y  ;  Gets a character
20D2FF    JSR $FFD2       ;  Displays it
C8        INY             ;  Increments character counter
CCF01D    CPY NOCHAR      ;  Finished message?
D0F5      BNE MOREC       ;  If not gets another character
60        RTS             ;  Returns from subroutine
==A39B    MSGE            ;  Displays a message at a specified
                             screen location
A000      LDY #00         ;  Clears character counter
==A39D    MORECH
B1FB      LDA (MSGADL),Y  ;  Gets a character
91FD      STA (MSGPRL),Y  ;  Displays it
C8        INY             ;  Increments character counter
CCF01D    CPY NOCHAR      ;  Finished message?
D0F6      BNE MORECH      ;  If not gets another character
60        RTS             ;  Returns from subroutine
==A3A8    PULSE           ;  Enables pulse counter on 6522-1
A900      LDA #00
8D0998    STA IO1T2H      ;  Starts counting pulses
60        RTS             ;  Returns from subroutine
==A3AE    PRTIM1          ;  Prints timer 1
ADCB1D    LDA TMR1HH      ;  Gets timer 1 hours HB
8DC61F    STA 8134        ;  Prints it
ADCC1D    LDA TMR1HL      ;  Gets timer 1 hours LB
8DC71F    STA 8135        ;  Prints it
ADCD1D    LDA TMR1MH      ;  Gets timer 1 minutes HB
8DC91F    STA 8137        ;  Prints it
==A3C0
ADCE1D    LDA TMR1ML      ;  Gets timer 1 minutes LB
8DCA1F    STA 8138        ;  Prints it
```

```
ADCF1D   LDA TMR1SH      ;  Gets timer 1 seconds HB
8DCC1F   STA 8140        ;  Prints it
ADD01D   LDA TMR1SL      ;  Gets timer 1 seconds LB
8DCD1F   STA 8141        ;  Prints it
==A3D2
200CA2   JSR $A20C       ;  Sets cursor location
60       RTS             ;  Returns from subroutine
==A3D6   PRTIM2          ;  Prints timer 2
ADD31D   LDA TMR2HH      ;  Gets timer 2 hours HB
8DDC1F   STA 8156        ;  Prints it
ADD41D   LDA TMR2HL      ;  Gets timer 2 hours LB
8DDD1F   STA 8157        ;  Prints it
ADD51D   LDA TMR2MH      ;  Gets timer 2 minutes HB
8DDF1F   STA 8159        ;  Prints it
==A3E8
ADD61D   LDA TMR2ML      ;  Gets timer 2 minutes LB
8DE01F   STA 8160        ;  Prints it
ADD71D   LDA TMR2SH      ;  Gets timer 2 seconds HB
8DE21F   STA 8162        ;  Prints it
ADD81D   LDA TMR2SL      ;  Gets timer 2 seconds LB
8DE31F   STA 8163        ;  Prints it
==A3FA
200CA2   JSR $A20C       ;  Sets cursor location
60       RTS             ;  Returns from subroutine
==A3FE   PRTIM3          ;  Prints timer 3
ADDB1D   LDA TMR3HH      ;  Gets timer 3 hours HB
8DF21F   STA 8178        ;  Prints it
ADDC1D   LDA TMR3HL      ;  Gets timer 3 hours LB
8DF31F   STA 8179        ;  Prints it
ADDD1D   LDA TMR3MH      ;  Gets timer 3 minutes HB
8DF51F   STA 8181        ;  Prints it
==A410
ADDE1D   LDA TMR3ML      ;  Gets timer 3 minutes LB
8DF61F   STA 8182        ;  Prints it
ADDF1D   LDA TMR3SH      ;  Gets timer 3 seconds HB
8DF81F   STA 8184        ;  Prints it
ADE01D   LDA TMR3SL      ;  Gets timer 3 seconds LB
8DF91F   STA 8185        ;  Prints it
==A422
200CA2   JSR $A20C       ;  Sets cursor location
60       RTS             ;  Returns from subroutine
==A426   TNEW            ;  Gets temperature from A/D convert-
                            er
202CA6   JSR $A62C       ;  Time delay (DELAY)
A980     LDA #$80        ;  Initializes test byte (first
                            digit)
8DF21D   STA DIGSEL
AD029A   LDA $9A02       ;  Resets EOC flag in 6520-1 CRB
==A431   NEOC
2C039A   BIT $9A03       ;  Has EOC occurred?
```

```
10FB      BPL NEOC        ;  Loops until EOC flag set
==A436    DS1
AD009A    LDA $9A00       ;  Loads digit select
2CF21D    BIT DIGSEL      ;  Compares with test byte
F0F8      BEQ DS1         ;  Loops until first digit
2908      AND #08         ;  Isolates first digit
4908      EOR #08
4A        LSR A
4A        LSR A
4A        LSR A
0930      ORA #$30        ;  Makes it ASCII
==A447
8DF31D    STA TEMPH       ;  Stores hundred's digit
4EF21D    LSR DIGSEL      ;  Shifts test byte
==A44D    DS2
AD009A    LDA $9A00       ;  Loads digit select byte
2CF21D    BIT DIGSEL      ;  Compares with test byte
F0F8      BEQ DS2         ;  Loops until second digit
290F      AND #$0F        ;  Removes digit select bits
0930      ORA #$30        ;  Makes it ASCII
8DF41D    STA TEMPT       ;  Stores ten's digit
4EF21D    LSR DIGSEL      ;  Shifts test byte
==A45F    DS3
AD009A    LDA $9A00       ;  Loads digit select byte
2CF21D    BIT DIGSEL      ;  Compares with test byte
F0F8      BEQ DS3         ;  Loops until third digit
290F      AND #$0F        ;  Removes digit select bits
0930      ORA #$30        ;  Makes it ASCII
8DF51D    STA TEMPU       ;  Stores unit's digit
4EF21D    LSR DIGSEL      ;  Shifts test byte
==A471    DS4
AD009A    LDA $9A00       ;  Loads digit select byte
2CF21D    BIT DIGSEL      ;  Compares with test byte
F0F8      BEQ DS4         ;  Loops until fourth digit
290F      AND #$0F        ;  Removes digit select bits
0930      ORA #$30        ;  Makes it ASCII
8DF61D    STA TEMPTH      ;  Store tenth's digit
60        RTS             ;  Returns from subroutine
==A481    SCREN1          ;  Displays title screen
2018E5    JSR $E518       ;  Clears the screen
18        CLC             ;  Clears the carry bit
A9F2      LDA #242        ;  Sets upper and lower case
8D0590    STA $9005
A200      LDX #00         ;  Prints 22 cyan asterisks in two
                             rows
A000      LDY #00
A916      LDA #22
8DF71D    STA COUNT
==A493    HERE
A92A      LDA #42         ;  Gets screen code for asterisk
```

```
9D161E    STA 7702,X      ;  Prints first row
9D4A1F    STA 8010,X      ;  Prints second row
E8        INX
A903      LDA #03         ;  Gets screen code for cyan char-
                             acter
991696    STA 38422,Y     ;  Colors first row
994A97    STA 38730,Y     ;  Colors second row
==A4A4
C8        INY
CEF71D    DEC COUNT
D0E9      BNE HERE
A200      LDX #00         ;  Prints 2 columns of 11 cyan aster-
                             isks
A000      LDY #00
A90B      LDA #11
8DF71D    STA COUNT
==A4B3    COL
A92A      LDA #42         ;  Screen code for asterisk
9D2C1E    STA 7724,X      ;  Prints first column
9D411E    STA 7745,X      ;  Prints second column
A903      LDA #03         ;  Gets screen code for cyan char-
                             acter
992C96    STA 38444,Y     ;  Colors first column
994196    STA 38465,Y     ;  Colors second column
==A4C3
8A        TXA
6916      ADC #22
AA        TAX
98        TYA
6916      ADC #22
A8        TAY
CEF71D    DEC COUNT
D0E3      BNE COL
A92A      LDA #42         ;  Gets screen code for asterisk
8D1E1F    STA 7966        ;  Prints first asterisk
==A4D5
8D341F    STA 7988        ;  Prints second asterisk
8D331F    STA 7987        ;  Prints third asterisk
8D491F    STA 8009        ;  Prints fourth asterisk
A903      LDA #03         ;  Gets screen code for cyan char-
                             acter
8D1E97    STA 38686       ;  Colors first asterisk
8D3497    STA 38708       ;  Colors second asterisk
==A4E6
8D3397    STA 38707       ;  Colors third asterisk
8D4997    STA 38729       ;  Colors fourth asterisk
A94F      LDA #79         ;  Gets screen code for O
8D441E    STA 7748        ;  Prints first O
8D7B1E    STA 7803        ;  Prints second O
8D7F1E    STA 7807        ;  Prints third O
```

```
==A4F7
8DB61E    STA 7862        ;  Prints fourth O
8DB71E    STA 7863        ;  Prints fifth O
8DBC1E    STA 7868        ;  Prints sixth O
8D121F    STA 7954        ;  Prints seventh O
A950      LDA #80         ;  Gets screen code for P
8D451E    STA 7749        ;  Prints first P
==A508
8D721E    STA 7794        ;  Prints second P
8DE71E    STA 7911        ;  Prints third P
8D131F    STA 7955        ;  Prints fourth P
A945      LDA #69         ;  Gets screen code for E
8D461E    STA 7750        ;  Prints first E
8D4A1E    STA 7754        ;  Prints second E
==A519
8D531E    STA 7763        ;  Prints third E
8D761E    STA 7798        ;  Prints fourth E
8DC21E    STA 7874        ;  Prints fifth E
8DE51E    STA 7909        ;  Prints sixth E
8DE81E    STA 7912        ;  Prints seventh E
8DEE1E    STA 7918        ;  Prints eighth E
==A52B
8D0F1F    STA 7951        ;  Prints ninth E
8D101F    STA 7952        ;  Prints tenth E
8D141F    STA 7956        ;  Prints eleventh E
8D291F    STA 7977        ;  Prints twelfth E
A94E      LDA #78         ;  Gets screen code for N
8D471E    STA 7751        ;  Prints first N
==A53C
8D771E    STA 7799        ;  Prints second N
8D7C1E    STA 7804        ;  Prints third N
8D191F    STA 7961        ;  Prints fourth N
A948      LDA #72         ;  Gets screen code for H
8D491E    STA 7753        ;  Prints first H
8D0D1F    STA 7949        ;  Prints second H
==A54D
A941      LDA #65         ;  Gets screen code for A
8D4B1E    STA 7755        ;  Prints first A
8D731E    STA 7795        ;  Prints second A
8DC01E    STA 7872        ;  Prints third A
8DEA1E    STA 7914        ;  Prints fourth A
8D161F    STA 7958        ;  Prints fifth A
==A55E
A952      LDA #82         ;  Gets screen code for R
8D4C1E    STA 7756        ;  Prints first R
8D511E    STA 7761        ;  Prints second R
8D541E    STA 7764        ;  Prints third R
8D801E    STA 7808        ;  Prints fourth R
8DBF1E    STA 7871        ;  Prints fifth R
```

```
==A56F
8DE91E    STA 7913        ;  Prints sixth R
8DED1E    STA 7917        ;  Prints seventh R
8D0E1F    STA 7950        ;  Prints eighth R
8D151F    STA 7957        ;  Prints ninth R
A954      LDA #84         ;  Gets screen code for T
8D4D1E    STA 7757        ;  Prints first T
==A580
8D741E    STA 7796        ;  Prints second T
8D781E    STA 7800        ;  Prints third T
8D7E1E    STA 7806        ;  Prints fourth T
8DC11E    STA 7873        ;  Prints fifth T
8DE41E    STA 7908        ;  Prints sixth T
8DEB1E    STA 7915        ;  Prints seventh T
==A592
8D0C1F    STA 7948        ;  Prints eighth T
8D171F    STA 7959        ;  Prints ninth T
8D261F    STA 7974        ;  Prints tenth T
A953      LDA #83         ;  Gets screen code for S
8D4F1E    STA 7759        ;  Prints first S
8DE01E    STA 7904        ;  Prints second S
==A5A3
8DEF1E    STA 7919        ;  Prints third S
8D2A1F    STA 7978        ;  Prints fourth S
A955      LDA #85         ;  Gets screen code for U
8D501E    STA 7760        ;  Prints first U
8DEC1E    STA 7916        ;  Prints second U
A947      LDA #71         ;  Gets screen code for G
==A5B3
8D521E    STA 7762        ;  Prints first G
8D1A1F    STA 7962        ;  Prints second G
A959      LDA #89         ;  Gets screen code for Y
8D551E    STA 7765        ;  Prints first Y
A949      LDA #73         ;  Gets screen code for I
8D751E    STA 7797        ;  Prints first I
==A5C3
8D7D1E    STA 7805        ;  Prints second I
8DE11E    STA 7905        ;  Prints third I
8D181F    STA 7960        ;  Prints fourth I
8D271F    STA 7975        ;  Prints fifth I
A94D      LDA #77         ;  Gets screen code for M
8D7A1E    STA 7802        ;  Prints first M
==A5D4
8DE61E    STA 7910        ;  Prints second M
8D281F    STA 7976        ;  Prints third M
A95E      LDA #94         ;  Gets screen code for a bullet
8DB21E    STA 7858        ;  Prints first bullet
8DDE1E    STA 7902        ;  Prints second bullet
8D0A1F    STA 7946        ;  Prints third bullet
```

```
==A5E5
A942      LDA #66          ;  Gets screen code for B
8DB41E    STA 7860         ;  Prints first B
A94C      LDA #76          ;  Gets screen code for L
8DB51E    STA 7861         ;  Prints first L
8DBB1E    STA 7867         ;  Prints second L
A944      LDA #68          ;  Gets screen code for D
8DB81E    STA 7864         ;  Prints first D
==A5F7
A946      LDA #70          ;  Gets screen code for F
8DBA1E    STA 7866         ;  Prints first F
A957      LDA #87          ;  Gets screen code for W
8DBD1E    STA 7869         ;  Prints first W
A958      LDA #88          ;  Gets screen code for X
8DE21E    STA 7906         ;  Prints first X
A96E      LDA #110         ;  Gets screen code for blue back-
                              ground
==A608
8D0F90    STA 36879        ;  Colors the background blue
A200      LDX #00
A90B      LDA #11          ;  11 rows of white characters
8DF81D    STA CNT
==A612    THERE
A912      LDA #18          ;  18 characters in a row
8DF71D    STA COUNT
==A617    LOOP
A901      LDA #01          ;  Gets screen code for white charac-
                              ters
9D4496    STA 38468,X      ;  Colors a row
E8        INX
CEF71D    DEC COUNT
D0F5      BNE LOOP
8A        TXA
6904      ADC #04
AA        TAX
CEF81D    DEC CNT
==A629
D0E7      BNE THERE
60        RTS              ;  Returns from subroutine
==A62C    DELAY            ;  Delays for roughly 1/2 sec
A200      LDX #00
A000      LDY #00
==A630    COMPY
C00A      CPY #10
F06B      BEQ RET          ;  Branches to return from subroutine
                              if finished with delay
==A634    COMPX
E0FF      CPX #255
F063      BEQ CONT
```

```
EA          NOP             ;  Waits
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
==A644
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
==A654
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
EA          NOP
==A664
EA          NOP
EA          NOP
EA          NOP
```

```
EA        NOP             ;  Waits
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
==A674
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
==A684
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
EA        NOP
==A694
EA        NOP
EA        NOP
```

```
EA        NOP
E8        INX
4C34A6    JMP $A634       ;  Repeats loop
==A69B    CONT
C8        INY
4C30A6    JMP $A630       ;  Repeats loop
==A69F    RET
60        RTS             ;  Returns from subroutine
;INTERRUPT ROUTINE
          *=$2000
==2000
ADC61D    LDA ROLFLG      ;  Gets roller flag
C9FF      CMP #$FF        ;  Is it set?
F007      BEQ GETRPM      ;  If it is then updates pump output
ADC71D    LDA PULFLG      ;  Gets pulsatile flag
C9FF      CMP #$FF        ;  Is it set?
D01E      BNE NEXT        ;  If it's not then updates tempera-
                             ture update counter
==200E    GETRPM          ;  Updates roller or pulsatile pump
                             output
ACF11D    LDY HALFCT      ;  Gets update counter
C8        INY             ;  Increments update counter
8CF11D    STY HALFCT      ;  Stores update counter
C01E      CPY #30         ;  Is it 1/2 sec yet?
D013      BNE NEXT        ;  If it isn't then updates tempera-
                             ture update counter
A9FF      LDA #$FF
8DC51D    STA PMPFLG      ;  Sets pump output update flag
==201E
ED0898    SBC IO1T2L      ;  Gets number of pulses counted
8DE51D    STA PULSES      ;  Stores number of pulses counted
A900      LDA #00
8DF11D    STA HALFCT      ;  Clears 1/2 sec counter
8D0998    STA IO1T2H      ;  Resets pulse counter
==202C    NEXT
ACE71D    LDY TMPCNT      ;  Gets temperature update counter
C8        INY             ;  Increments it
8CE71D    STY TMPCNT      ;  Stores it
C0FF      CPY #$FF        ;  Has it been 4 sec?
D00A      BNE TM1UPD      ;  If it hasn't branches to check
                             timer 1
A9FF      LDA #$FF
8DE61D    STA TMPFLG      ;  It has so sets temperature update
                             flag
==203C
A900      LDA #00
8DE71D    STA TMPCNT      ;  Clears temperature update counter
==2041    TM1UPD          ;  Updates timer 1 if necessary
ADCA1D    LDA TM1FLG      ;  Gets timer 1 flag
C9FF      CMP #$FF        ;  Is it set?
```

```
D073      BNE TM2UPD     ;  If it isn't branches to check
                            timer 2
ACD11D    LDY TMR1JF     ;  Gets timer 1 jiffy counter (1/60
                            sec)
C8        INY            ;  Increments it
8CD11D    STY TMR1JF     ;  Stores it
C03C      CPY #60        ;  Has it been 1 sec?
==2051
D068      BNE TM2UPD     ;  If it hasn't branches to check
                            timer 2
A900      LDA #00
8DD11D    STA TMR1JF     ;  It has so clears jiffy counter
ACD01D    LDY TMR1SL     ;  Gets timer 1 seconds LB
C8        INY            ;  Increments it
8CD01D    STY TMR1SL     ;  Stores it
C03A      CPY #$3A       ;  Has it been 10 sec?
==2061
D055      BNE PTIMR1     ;  If it hasn't then branches to
                            print timer 1
A930      LDA #$30
8DD01D    STA TMR1SL     ;  It has so clears timer 1 sec LB
ACCF1D    LDY TMR1SH     ;  Gets timer 1 seconds HB
C8        INY            ;  Increments it
8CCF1D    STY TMR1SH     ;  Stores it
C036      CPY #$36       ;  Has it been 60 seconds?
==2071
D045      BNE PTIMR1     ;  If it hasn't branches to print
                            timer 1
A930      LDA #$30       ;  It has so clears timer 1 sec HB
8DCF1D    STA TMR1SH
ACCE1D    LDY TMR1ML     ;  Gets timer 1 minutes LB
C8        INY            ;  Increments it
8CCE1D    STY TMR1ML     ;  Stores it
C03A      CPY #$3A       ;  Has it been 10 minutes?
==2081
D035      BNE PTIMR1     ;  If it hasn't branches to print
                            timer 1
A930      LDA #$30
8DCE1D    STA TMR1ML     ;  It has so clears timer 1 min LB
ACCD1D    LDY TMR1MH     ;  Gets timer 1 minutes HB
C8        INY            ;  Increments it
8CCD1D    STY TMR1MH     ;  Stores it
C036      CPY #$36       ;  Has it been 60 minutes?
==2091
D025      BNE PTIMR1     ;  If it hasn't branches to print
                            timer 1
A930      LDA #$30
8DCD1D    STA TMR1MH     ;  It has so clears timer 1 min HB
ACCC1D    LDY TMR1HL     ;  Gets timer 1 hours LB
C8        INY            ;  Increments it
```

```
8CCC1D    STY TMR1HL      ;  Stores it
C03A      CPY #$3A        ;  Has it been 10 hours?
==20A1
D015      BNE PTIMR1      ;  If it hasn't branches to print
                             timer 1
A930      LDA #$30
8DCC1D    STA TMR1HL      ;  It has so clears timer 1 hours LB
ACCB1D    LDY TMR1HH      ;  Gets timer 1 hours HB
C8        INY             ;  Increments it
8CCB1D    STY TMR1HH      ;  Stores it
C036      CPY #$36        ;  Has it been 60 hours?
==20B1
D005      BNE PTIMR1      ;  If it hasn't branches to print
                             timer 1
A930      LDA #$30
8DCB1D    STA TMR1HH      ;  It has so clears timer 1 hours HB
==20B8    PTIMR1          ;  Prints timer 1
20AEA3    JSR PRTIM1
==20BB    TM2UPD          ;  Checks timer 2
ADD21D    LDA TM2FLG      ;  Gets timer 2 flag
C9FF      CMP #$FF        ;  Is it set?
D073      BNE TM3UPD      ;  If it isn't branches to check
                             timer 3
ACD91D    LDY TMR2JF      ;  Gets timer 2 jiffy counter
C8        INY             ;  Increments it
8CD91D    STY TMR2JF      ;  Stores it
C03C      CPY #60         ;  Has it been 1 sec?
==20CB
D068      BNE TM3UPD      ;  If it hasn't branches to check
                             timer 3
A900      LDA #00
8DD91D    STA TMR2JF      ;  It has so clears timer 2 jiffy
                             counter
ACD81D    LDY TMR2SL      ;  Gets timer 2 seconds LB
C8        INY             ;  Increments it
8CD81D    STY TMR2SL      ;  Stores it
C03A      CPY #$3A        ;  Has it been 10 seconds?
==20DB
D055      BNE PTIMR2      ;  If it hasn't branches to print
                             timer 2
A930      LDA #$30
8DD81D    STA TMR2SL      ;  It has so clears timer 2 sec LB
ACD71D    LDY TMR2SH      ;  Gets timer 2 seconds HB
C8        INY             ;  Increments it
8CD71D    STY TMR2SH      ;  Stores it
C036      CPY #$36        ;  Has it been 60 seconds?
==20EB
D045      BNE PTIMR2      ;  If it hasn't branches to print
                             timer 2
A930      LDA #$30
```

```
8DD71D    STA TMR2SH     ;  It has so clears timer 2 sec HB
ACD61D    LDY TMR2ML     ;  Gets timer 2 minutes LB
C8        INY            ;  Increments it
8CD61D    STY TMR2ML     ;  Stores it
C03A      CPY #$3A       ;  Has it been 10 minutes?
==20FB
D035      BNE PTIMR2     ;  If it hasn't branches to print
                            timer 2
A930      LDA #$30
8DD61D    STA TMR2ML     ;  It has so clears timer 2 min LB
ACD51D    LDY TMR2MH     ;  Gets timer 2 minutes HB
C8        INY            ;  Increments it
8CD51D    STY TMR2MH     ;  Stores it
C036      CPY #$36       ;  Has it been 60 minutes?
=210B
D025      BNE PTIMR2     ;  If it hasn't branches to print
                            timer 2
A930      LDA #$30
8DD51D    STA TMR2MH     ;  It has so clears timer 2 min HB
ACD41D    LDY TMR2HL     ;  Gets timer 2 hours LB
C8        INY            ;  Increments it
8CD41D    STY TMR2HL     ;  Stores it
C03A      CPY #$3A       ;  Has it been 10 hours?
==211B
D015      BNE PTIMR2     ;  If it hasn't branches to print
                            timer 2
A930      LDA #$30
8DD41D    STA TMR2HL     ;  It has so clears timer 2 hours LB
ACD31D    LDY TMR2HH     ;  Gets timer 2 hours HB
C8        INY            ;  Increments it
8CD31D    STY TMR2HH     ;  Stores it
C036      CPY #$36       ;  Has it been 60 hours?
==212B
D005      BNE PTIMR2     ;  If it hasn't branches to print
                            timer 2
A930      LDA #$30
8DD31D    STA TMR2HH     ;  It has so clears timer 2 hours HB
==2132    PTIMR2         ;  Prints timer 2
20D6A3    JSR PRTIM2
==2135    TM3UPD         ;  Checks timer 3
ADDA1D    LDA TM3FLG     ;  Gets timer 3 flag
C9FF      CMP #$FF       ;  Is it set?
D073      BNE CLKUPD     ;  If it isn't branches to check
                            clock
ACE11D    LDY TMR3JF     ;  It is so gets timer 3 jiffy
                            counter
C8        INY            ;  Increments it
8CE11D    STY TMR3JF     ;  Stores it
C03C      CPY #60        ;  Has it been 1 sec?
```

```
==2145
D068     BNE CLKUPD      ;  If it hasn't branches to check
                            clock
A900     LDA #00
8DE11D   STA TMR3JF      ;  It has so clears timer 3 jiffy
                            counter
ACE01D   LDY TMR3SL      ;  Gets timer 3 seconds LB
C8       INY             ;  Increments it
8CE01D   STY TMR3SL      ;  Stores it
C03A     CPY #$3A        ;  Has it been 10 seconds?
==2155
D055     BNE PTIMR3      ;  If it hasn't branches to print
                            timer 3
A930     LDA #$30
8DE01D   STA TMR3SL      ;  It has so clears timer 3 sec LB
ACDF1D   LDY TMR3SH      ;  Gets timer 3 seconds HB
C8       INY             ;  Increments it
8CDF1D   STY TMR3SH      ;  Stores it
C036     CPY #$36        ;  Has it been 60 seconds?
==2165
D045     BNE PTIMR3      ;  If it hasn't branches to print
                            timer 3
A930     LDA #$30
8DDF1D   STA TMR3SH      ;  It has so clears timer 3 sec HB
ACDE1D   LDY TMR3ML      ;  Gets timer 3 minutes LB
C8       INY             ;  Increments it
8CDE1D   STY TMR3ML      ;  Stores it
C03A     CPY #$3A        ;  Has it been 10 minutes?
==2175
D035     BNE PTIMR3      ;  If it hasn't branches to print
                            timer 3
A930     LDA #$30
8DDE1D   STA TMR3ML      ;  It has so clears timer 3 min LB
ACDD1D   LDY TMR3MH      ;  Gets timer 3 minutes HB
C8       INY             ;  Increments it
8CDD1D   STY TMR3MH      ;  Stores it
C036     CPY #$36        ;  Has it been 60 minutes?
==2185
D025     BNE PTIMR3      ;  If it hasn't branches to print
                            timer 3
A930     LDA #$30
8DDD1D   STA TMR3MH      ;  It has so clears timer 3 min HB
ACDC1D   LDY TMR3HL      ;  Gets timer 3 hours LB
C8       INY             ;  Increments it
8CDC1D   STY TMR3HL      ;  Stores it
C03A     CPY #$3A        ;  Has it been 10 hours?
==2195
D015     BNE PTIMR3      ;  If it hasn't branches to print
                            timer 3
A930     LDA #$30
```

```
8DDC1D    STA TMR3HL     ;  It has so clears timer 3 hours LB
ACDB1D    LDY TMR3HH     ;  Gets timer 3 hours HB
C8        INY            ;  Increments it
8CDB1D    STY TMR3HH     ;  Stores it
C036      CPY #$36       ;  Has it been 60 hours?
==21A5
D005      BNE PTIMR3     ;  If it hasn't branches to print
                            timer 3
A930      LDA #$30
8DDB1D    STA TMR3HH     ;  It has so clears timer 3 hours HB
==21AC    PTIMR3         ;  Prints timer 3
20FEA3    JSR PRTIM3
==21AF    CLKUPD         ;  Checks clock
ADE31D    LDA CLKFLG     ;  Gets clock update flag
C9FF      CMP #$FF       ;  Is it set?
D015      BNE CENTRF     ;  If it isn't branches to check cen-
                            trifugal pump output update
ACE41D    LDY CLKJIF     ;  It is so gets clock jiffy counter
C8        INY            ;  Increments it
8CE41D    STY CLKJIF     ;  Stores it
C03C      CPY #60        ;  Has it been a second?
==21BF
D00A      BNE CENTRF     ;  If it hasn't branches to check
                            centrifugal pump update
A900      LDA #00
8DE41D    STA CLKJIF     ;  It has so clears clock jiffy
                            counter
A9FF      LDA #$FF
8DE21D    STA PRCLKF     ;  Sets print clock flag
==21CB    CENTRF         ;  Checks centrifugal pump update
ADC81D    LDA CENFLG     ;  Gets centrifugal update flag
C9FF      CMP #$FF       ;  Is it set?
D015      BNE VICINT     ;  If it isn't branches to VIC 20
                            interrupt routine
ACC91D    LDY CENCNT     ;  Gets centrifugal update counter
C8        INY            ;  Increments it
8CC91D    STY CENCNT     ;  Stores it
C01E      CPY #$1E       ;  Has it been 1/2 second?
==21DB
D00A      BNE VICINT     ;  If it hasn't branches to VIC 20
                            interrupt routine
A900      LDA #00
8DC91D    STA CENCNT     ;  It has so clears centrifugal up-
                            date counter
A9FF      LDA #$FF
8DC51D    STA PMPFLG     ;  Sets pump output update flag
==21E7    VICINT         ;  Jumps to VIC 20 interrupt routine
4CBFEA    JMP $EABF
```

```
;SUBROUTINES
==21EA    CONTMP          ;  Prints temperature degree units
                             and check for unconnected probes
AD029A    LDA $9A02       ;  Gets degree unit
2901      AND #$01        ;  Masks off other bits
8DEF1D    STA DEGUNT      ;  Stores it
C901      CMP #$01        ;  Is it degree F?
D005      BNE DEGC        ;  If it isn't branches to degree C
A946      LDA #70         ;  Gets screen code for F
4CFD21    JMP $21FD       ;  Prints the degree units
==21FB    DEGC
A943      LDA #67         ;  Gets screen code for C
==21FD    PRDEG           ;  Prints the degree units
8D3D1F    STA 7997
8D481F    STA 8008
8D691F    STA 8041
8D741F    STA 8052
8D951F    STA 8085
8DA01F    STA 8096
==220F
ADE81D    LDA TMP1NC      ;  Gets temperature 1 flag
C900      CMP #00         ;  Is it connected?
F008      BEQ CKTMP2      ;  If it is branches to check temp-
                             erature 2
A920      LDA #$20        ;  It isn't so clears its display
8D3A1F    STA 7994
8D3D1F    STA 7997
==221E    CKTMP2          ;  Checks temperature 2
ADE91D    LDA TMP2NC      ;  Gets temperature 2 flag
C900      CMP #00         ;  Is it connected?
F008      BEQ CKTMP3      ;  If it is branches to check temp-
                             erature 3
A920      LDA #$20        ;  It isn't so clears its display
8D481F    STA 8008
8D451F    STA 8005
==222D    CKTMP3          ;  Checks temperature 3
ADEA1D    LDA TMP3NC      ;  Gets temperature 3 flag
C900      CMP #00         ;  Is it connected?
F008      BEQ CKTMP4      ;  If it is branches to check temp-
                             erature 4
A920      LDA #$20        ;  It isn't so clears its display
8D691F    STA 8041
8D661F    STA 8038
==223C    CKTMP4          ;  Checks temperature 4
ADEB1D    LDA TMP4NC      ;  Gets temperature 4 flag
C900      CMP #00         ;  Is it connected?
F008      BEQ CKTMP5      ;  If it is branches to check temp-
                             erature 5
A920      LDA #$20        ;  It isn't so clears its display
8D741F    STA 8052
```

```
8D711F    STA 8049
==224B    CKTMP5          ;  Checks temperature 5
ADEC1D    LDA TMP5NC      ;  Gets temperature 5 flag
C900      CMP #00         ;  Is it connected?
F008      BEQ CKTMP6      ;  If it is branches to check temp-
                             erature 6
A920      LDA #$20        ;  It isn't so clears its display
8D951F    STA 8085
8D921F    STA 8082
==225A    CKTMP6          ;  Checks temperature 6
ADED1D    LDA TMP6NC      ;  Gets temperature 6 flag
C900      CMP #00         ;  Is it connected?
F008      BEQ RETURN      ;  If it is branches to return from
                             subroutine
A920      LDA #$20        ;  It isn't so clears its display
8DA01F    STA 8096
8D9D1F    STA 8093
==2269    RETURN
60        RTS             ;  Returns from subroutine
==226A    SCREN2          ;  Displays monitor screen
2018E5    JSR $E518       ;  Clears the screen
18        CLC             ;  Clears the carry bit
A92E      LDA #$2E        ;  Gets screen code for .
8D9D1F    STA 8093        ;  Prints first .
A948      LDA #72         ;  Gets screen code for H
8D171E    STA 7703        ;  Prints first H
8D1B1E    STA 7707        ;  Prints second H
==227B
8D471E    STA 7751        ;  Prints third H
A945      LDA #69         ;  Gets screen code for E
8D441E    STA 7748        ;  Prints first E
8D181E    STA 7704        ;  Prints second E
8D7A1E    STA 7802        ;  Prints third E
8D7E1E    STA 7806        ;  Prints fourth E
==228C
8DD51E    STA 7893        ;  Prints fifth E
8D0E1F    STA 7950        ;  Prints sixth E
8D111F    STA 7953        ;  Prints seventh E
8D171F    STA 7959        ;  Prints eighth E
8DA71F    STA 8103        ;  Prints ninth E
8DAE1F    STA 8110        ;  Prints tenth E
==229E
8DC31F    STA 8131        ;  Prints eleventh E
8DD91F    STA 8153        ;  Prints twelfth E
8DEF1F    STA 8175        ;  Prints thirteenth E
A949      LDA #73         ;  Gets screen code for I
8D191E    STA 7705        ;  Prints first I
8D451E    STA 7749        ;  Prints second I
==22AF
8DAC1F    STA 8108        ;  Prints third I
```

```
8DC11F    STA 8129        ;  Prints fourth I
8DD71F    STA 8151        ;  Prints fifth I
8DED1F    STA 8173        ;  Prints sixth I
A909      LDA #09         ;  Gets screen code for i
8D3B1E    STA 7739        ;  Prints first i
==22C0
8DC41E    STA 7876        ;  Prints second i
8DEA1E    STA 7914        ;  Prints third i
8D001F    STA 7936        ;  Prints fourth i
A947      LDA #71         ;  Gets screen code for G
8D1A1E    STA 7706        ;  Prints first G
8D461E    STA 7750        ;  Prints second G
==22D1
A907      LDA #07         ;  Gets screen code for g
8D521E    STA 7762        ;  Prints first g
8DEE1E    STA 7918        ;  Prints second g
A954      LDA #84         ;  Gets screen code for T
8D1C1E    STA 7708        ;  Prints first T
8D481E    STA 7752        ;  Prints second T
==22E1
8DB31E    STA 7859        ;  Prints third T
8DB61E    STA 7862        ;  Prints fourth T
8DD41E    STA 7892        ;  Prints fifth T
8D0D1F    STA 7949        ;  Prints sixth T
8D141F    STA 7956        ;  Prints seventh T
8DA91F    STA 8105        ;  Prints eighth T
==22F3
8DAB1F    STA 8107        ;  Prints ninth T
8DC01F    STA 8128        ;  Prints tenth T
8DD61F    STA 8150        ;  Prints eleventh T
8DEC1F    STA 8172        ;  Prints twelfth T
A93D      LDA #61         ;  Gets screen code for =
8D1E1E    STA 7710        ;  Prints first =
==2304
8D341E    STA 7732        ;  Prints second =
8D4A1E    STA 7754        ;  Prints third =
8D601E    STA 7776        ;  Prints fourth =
8D8C1E    STA 7820        ;  Prints fifth =
8DB81E    STA 7864        ;  Prints sixth =
8DDE1E    STA 7902        ;  Prints seventh =
==2316
8DF41E    STA 7924        ;  Prints eighth =
A903      LDA #03         ;  Gets screen code for c
8D251E    STA 7717        ;  Prints first c
A943      LDA #67         ;  Gets screen code for C
8D791E    STA 7801        ;  Prints first C
8DA31F    STA 8099        ;  Prints second C
==2326
A90D      LDA #13         ;  Gets screen code for m
8D261E    STA 7718        ;  Prints first m
```

```
8D931E    STA 7827        ;  Prints second m
8DC01E    STA 7872        ;  Prints third m
8DC31E    STA 7875        ;  Prints fourth m
8DE61E    STA 7910        ;  Prints fifth m
==2337
8DE91E    STA 7913        ;  Prints sixth m
8DFC1E    STA 7932        ;  Prints seventh m
8DFF1E    STA 7935        ;  Prints eighth m
8D031F    STA 7939        ;  Prints ninth m
A94D      LDA #77         ;  Gets screen code for M
8DAD1F    STA 8109        ;  Prints first M
==2348
8D0F1F    STA 7951        ;  Prints second M
8DC21F    STA 8130        ;  Prints third M
8DD81F    STA 8152        ;  Prints fourth M
8DEE1F    STA 8174        ;  Prints fifth M
A90E      LDA #14         ;  Gets screen code for n
8D3C1E    STA 7740        ;  Prints first n
==2359
8DC51E    STA 7877        ;  Prints second n
8DEB1E    STA 7915        ;  Prints third n
8D011F    STA 7937        ;  Prints fourth n
A94E      LDA #78         ;  Gets screen code for N
8DA81F    STA 8104        ;  Prints first N
A957      LDA #87         ;  Gets screen code for W
==2369
8D431E    STA 7747        ;  Prints first W
8DD01E    STA 7888        ;  Prints second W
A90B      LDA #11         ;  Gets screen code for k
8D511E    STA 7761        ;  Prints first k
8DED1E    STA 7917        ;  Prints second k
A90C      LDA #12         ;  Gets screen code for l
==2379
8D671E    STA 7783        ;  Prints first l
8DC11E    STA 7873        ;  Prints second l
8DE71E    STA 7911        ;  Prints third l
8DFD1E    STA 7933        ;  Prints fourth l
A94C      LDA #76         ;  Gets screen code for L
8DC81E    STA 7880        ;  Prints first L
==238A
8DCE1E    STA 7886        ;  Prints second L
A902      LDA #02         ;  Gets screen code for b
8D681E    STA 7784        ;  Prints first b
A942      LDA #66         ;  Gets screen code for B
8D6F1E    STA 7791        ;  Prints first B
8DC71E    STA 7879        ;  Prints second B
==239A
A94F      LDA #79         ;  Gets screen code for O
8D701E    STA 7792        ;  Prints first O
8DB11E    STA 7857        ;  Prints second O
```

```
8DC91E    STA 7881        ;  Prints third O
8DCA1E    STA 7882        ;  Prints fourth O
8DCF1E    STA 7887        ;  Prints fifth O
==23AB
A944      LDA #68         ;  Gets screen code for D
8D711E    STA 7793        ;  Prints first D
8DCB1E    STA 7883        ;  Prints second D
A959      LDA #89         ;  Gets screen code for Y
8D721E    STA 7794        ;  Prints first Y
A953      LDA #83         ;  Gets screen code for S
8D741E    STA 7796        ;  Prints first S
==23BD
8D181F    STA 7960        ;  Prints second S
A955      LDA #85         ;  Gets screen code for U
8D751E    STA 7797        ;  Prints first U
8DB21E    STA 7858        ;  Prints second U
8DB51E    STA 7861        ;  Prints third U
8D151F    STA 7957        ;  Prints fourth U
==23CE
8DA41F    STA 8100        ;  Prints fifth U
A952      LDA #82         ;  Gets screen code for R
8D761E    STA 7798        ;  Prints first R
8D7D1E    STA 7805        ;  Prints second R
8DD21E    STA 7890        ;  Prints third R
8D121F    STA 7954        ;  Prints fourth R
==23DF
8D161F    STA 7958        ;  Prints fifth R
8DA51F    STA 8101        ;  Prints sixth R
8DA61F    STA 8102        ;  Prints seventh R
A946      LDA #70         ;  Gets screen code for F
8D771E    STA 7799        ;  Prints first F
8DCD1E    STA 7885        ;  Prints second F
==23F0
A941      LDA #65         ;  Gets screen code for A
8D781E    STA 7800        ;  Prints first A
8D7C1E    STA 7804        ;  Prints second A
8D7F1E    STA 7807        ;  Prints third A
8DD31E    STA 7891        ;  Prints fourth A
8D131F    STA 7955        ;  Prints fifth A
==2401
A92A      LDA #42         ;  Gets screen code for *
8D941E    STA 7828        ;  Prints first *
8D951E    STA 7829        ;  Prints second *
8D041F    STA 7940        ;  Prints third *
8D051F    STA 7941        ;  Prints fourth *
A92E      LDA #46         ;  Gets screen code for .
==2411
8D231E    STA 7715        ;  Prints first .
8D391E    STA 7737        ;  Prints second .
8D4F1E    STA 7759        ;  Prints third .
```

```
8D651E    STA 7781       ;  Prints fourth .
8D8F1E    STA 7823       ;  Prints fifth .
8DBE1E    STA 7870       ;  Prints sixth .
==2423
8DE41E    STA 7908       ;  Prints seventh .
8DFA1E    STA 7930       ;  Prints eighth .
A932      LDA #50        ;  Gets screen code for 2
8D961E    STA 7830       ;  Prints first 2
8D061F    STA 7942       ;  Prints second 2
8D2A1F    STA 7978       ;  Prints third 2
==2434
8DDA1F    STA 8154       ;  Prints fourth 2
A950      LDA #80        ;  Gets screen code for P
8DB41E    STA 7860       ;  Prints first P
8D101F    STA 7952       ;  Prints second P
A92F      LDA #47        ;  Gets screen code for /
8DC21E    STA 7874       ;  Prints first /
==2444
8DE81E    STA 7912       ;  Prints second /
8DEC1E    STA 7916       ;  Prints third /
8DFE1E    STA 7934       ;  Prints fourth /
8D021F    STA 7938       ;  Prints fifth /
A92E      LDA #$2E       ;  Gets screen code for .
8D3A1F    STA 7994       ;  Prints first .
==2455
8D451F    STA 8005       ;  Prints second .
8D661F    STA 8038       ;  Prints third .
8D711F    STA 8049       ;  Prints fourth .
8D921F    STA 8082       ;  Prints fifth .
A92D      LDA #45        ;  Gets screen code for -
8D201F    STA 7968       ;  Prints first -
==2466
8D2B1F    STA 7979       ;  Prints second -
8D4C1F    STA 8012       ;  Prints third -
8D571F    STA 8023       ;  Prints fourth -
8D781F    STA 8056       ;  Prints fifth -
8D831F    STA 8067       ;  Prints sixth -
A93A      LDA #58        ;  Gets screen code for :
==2477
8DB21F    STA 8114       ;  Prints first :
8DB51F    STA 8117       ;  Prints second :
8DC81F    STA 8136       ;  Prints third :
8DCB1F    STA 8139       ;  Prints fourth :
8DDE1F    STA 8158       ;  Prints fifth :
8DE11F    STA 8161       ;  Prints sixth :
==2489
8DF41F    STA 8180       ;  Prints seventh :
8DF71F    STA 8183       ;  Prints eighth :
A931      LDA #49        ;  Gets screen code for 1
8D1F1F    STA 7967       ;  Prints first 1
```

```
8DC41F    STA 8132        ;  Prints second 1
A933      LDA #51         ;  Gets screen code for 3
==2499
8D4B1F    STA 8011        ;  Prints first 3
8DF01F    STA 8176        ;  Prints second 3
A934      LDA #52         ;  Gets screen code for 4
8D561F    STA 8022        ;  Prints first 4
A935      LDA #53         ;  Gets screen code for 5
8D771F    STA 8055        ;  Prints first 5
==24A9
A936      LDA #54         ;  Gets screen code for 6
8D821F    STA 8066        ;  Prints first 6
A96E      LDA #110        ;  Gets screen code for blue back-
                             ground
8D0F90    STA 36879       ;  Colors background
A200      LDX #00
==24B5    ROW
A907      LDA #07         ;  Gets screen code for yellow
9D0096    STA 38400,X     ;  Colors first row yellow
E8        INX
E016      CPX #22
D0F6      BNE ROW
A200      LDX #00
==24C1    ROW1
A903      LDA #03         ;  Gets screen code for cyan
9D1696    STA 38422,X     ;  Colors second - ninth row cyan
E8        INX
E0B0      CPX #176
D0F6      BNE ROW1
A200      LDX #00
==24CD    ROW2
A907      LDA #07         ;  Gets screen code for yellow
9DC696    STA 38598,X     ;  Colors tenth - twelfth row yellow
E8        INX
E042      CPX #66
D0F6      BNE ROW2
A200      LDX #00
==24D9    ROW3
A901      LDA #01         ;  Gets screen code for white
9D0897    STA 38664,X     ;  Colors thirteenth - nineteenth row
                             white
E8        INX
E09A      CPX #154
D0F6      BNE ROW3
A200      LDX #00
==24E5    ROW4
A903      LDA #03         ;  Gets screen code for cyan
9DA297    STA 38818,X     ;  Colors twentieth - twenty-third
                             row cyan
E8        INX
```

```
E058      CPX #88
D0F6      BNE ROW4
A9F2      LDA #242        ;  Sets upper and lower case
8D0590    STA $9005
60        RTS             ;  Returns from subroutine
==24F5    MSG1            ;  Message 1
C6        .BYT $C6,$4F,   ;  F
  $52,$20,$54,$48,$45,
  $20,$46,$4F,$4C,$4C,
4F                        ;  o
52                        ;  r
20                        ;  space
54                        ;  t
48                        ;  h
45                        ;  e
20                        ;  space
46                        ;  f
4F                        ;  o
4C                        ;  l
4C                        ;  l
4F        .BYT $4F,$57,   ;  o
  $49,$4E,$47,$0D,$49,
  $4E,$50,$55,$54,$53,
  $2C,$20
57                        ;  w
49                        ;  i
4E                        ;  n
==2505
47                        ;  g
0D                        ;  carriage return
49                        ;  i
4E                        ;  n
50                        ;  p
55                        ;  u
54                        ;  t
53                        ;  s
2C                        ;  ,
20                        ;  space
41        .BYT $41,$46,   ;  a
  $54,$45,$52,$20,$54,
  $48,$45,$0D,$41,$4E,
  $53,$57
46                        ;  f
54                        ;  t
45                        ;  e
52                        ;  r
20                        ;  space
==2515
54                        ;  t
48                        ;  h
```

```
45                          ;  e
0D                          ;  carriage return
41                          ;  a
4E                          ;  n
53                          ;  s
57                          ;  w
45          .BYT $45,$52,   ;  e
  $20,$54,$4F,$20,$54,
  $48,$45,$0D,$51,$55,
  $45,$53
52                          ;  r
20                          ;  space
54                          ;  t
4F                          ;  o
20                          ;  space
54                          ;  t
48                          ;  h
==2525
45                          ;  e
0D                          ;  carriage return
51                          ;  q
55                          ;  u
45                          ;  e
53                          ;  s
54          .BYT $54,$49,   ;  t
  $4F,$4E,$20,$49,$53,
  $20,$54,$59,$50,$45,
  $44,$20
49                          ;  i
4F                          ;  o
4E                          ;  n
20                          ;  space
49                          ;  i
53                          ;  s
20                          ;  space
54                          ;  t
59                          ;  y
==2535
50                          ;  p
45                          ;  e
44                          ;  d
20                          ;  space
49          .BYT $49,$4E,   ;  i
  $2C,$20,$48,$49,$54,
  $20,$54,$48,$45,$20,
  $D2,$C5
4E                          ;  n
2C                          ;  ,
20                          ;  space
48                          ;  h
```

```
49                          ;  i
54                          ;  t
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
==2545
D2                          ;  R
C5                          ;  E
D4          .BYT $D4,$D5,   ;  T
  $D2,$CE,$20,$4B,$45,
  $59,$2E
D5                          ;  U
D2                          ;  R
CE                          ;  N
20                          ;  space
4B                          ;  k
45                          ;  e
59                          ;  y
2E                          ;  .
==2550    MSG2              ;  Message 2
C9          .BYT $C9,$53,   ;  I
  $20,$54,$48,$49,$53,
  $20,$43,$4F,$52,$52
53                          ;  s
20                          ;  space
54                          ;  t
48                          ;  h
49                          ;  i
53                          ;  s
20                          ;  space
43                          ;  c
4F                          ;  o
52                          ;  r
52                          ;  r
45          .BYT $45,$43,   ;  e
  $54
43                          ;  c
54                          ;  t
==255F    MSG3              ;  Message 3
C9          .BYT $C9,$4E,   ;  I
  $56,$41,$4C,$49,$44,
  $20,$45,$4E,$54,$52
4E                          ;  n
56                          ;  v
41                          ;  a
4C                          ;  l
49                          ;  i
44                          ;  d
```

```
20                      ;  space
45                      ;  e
4E                      ;  n
54                      ;  t
52                      ;  r
59        .BYT $59,$2C, ;  y
  $20,$54,$59,$50,$45
2C                      ;  ,
20                      ;  space
54                      ;  t
==256F
59                      ;  y
50                      ;  p
45                      ;  e
==2572    MSG 4         ;  Message 4
D7        .BYT $D7,$48, ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$48,$45,$20
48                      ;  h
41                      ;  a
54                      ;  t
20                      ;  space
49                      ;  i
53                      ;  s
20                      ;  space
54                      ;  t
48                      ;  h
45                      ;  e
20                      ;  space
50        .BYT $50,$41, ;  p
  $54,$49,$45,$4E,$54,
  $27,$53
41                      ;  a
54                      ;  t
49                      ;  i
==2582
45                      ;  e
4E                      ;  n
54                      ;  t
27                      ;  '
53                      ;  s
==2587    MSG5          ;  Message 5
C9        .BYT $C9,$53, ;  I
  $20,$54,$48,$45,$20,
  $48,$45,$49,$47,$48
53                      ;  s
20                      ;  space
54                      ;  t
48                      ;  h
45                      ;  e
```

```
20                          ;  space
48                          ;  h
45                          ;  e
49                          ;  i
47                          ;  g
48                          ;  h
54          .BYT $54,$20,   ;  t
  $49,$4E,$20,$43,$4D
20                          ;  space
49                          ;  i
4E                          ;  n
==2597
20                          ;  space
43                          ;  c
4D                          ;  m
==259A    MSG6              ;  Message 6
C9          .BYT $C9,$53,   ;  I
  $20,$54,$48,$45,$20,
  $57,$45,$49,$47,$48
53                          ;  s
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
57                          ;  w
45                          ;  e
49                          ;  i
47                          ;  g
48                          ;  h
54          .BYT $54,$20,   ;  t
  $49,$4E,$20,$4B,$47
20                          ;  space
49                          ;  i
4E                          ;  n
==25AA
20                          ;  space
4B                          ;  k
47                          ;  g
==25AD    MSG7              ;  Message 7
C9          .BYT $C9,$53,   ;  I
  $20,$41,$20,$43,$45,
  $4E,$54,$52,$49,$46
53                          ;  s
20                          ;  space
41                          ;  a
20                          ;  space
43                          ;  c
45                          ;  e
4E                          ;  n
```

```
54                          ;  t
52                          ;  r
49                          ;  i
46                          ;  f
55          .BYT $55,$47,   ;  u
  $41,$4C,$0D,$52,$4F,
  $4C,$4C,$45,$52,$20,
  $4F,$52
47                          ;  g
41                          ;  a
4C                          ;  l
==25BD
0D                          ;  carriage return
52                          ;  r
4F                          ;  o
4C                          ;  l
4C                          ;  l
45                          ;  e
52                          ;  r
20                          ;  space
4F                          ;  o
52                          ;  r
20          .BYT $20,$50,   ;  space
  $55,$4C,$53,$41,$54,
  $49,$4C,$45
50                          ;  p
55                          ;  u
4C                          ;  l
53                          ;  s
41                          ;  a
==25CD
54                          ;  t
49                          ;  i
4C                          ;  l
45                          ;  e
==25D1    MSG8              ;  Message 8
49          .BYT $49,$4E,   ;  i
  $20,$43,$45,$4E,$54,
  $52,$49,$46,$55,$47
4E                          ;  n
20                          ;  space
43                          ;  c
45                          ;  e
4E                          ;  n
54                          ;  t
52                          ;  r
49                          ;  i
46                          ;  f
55                          ;  u
47                          ;  g
```

```
41          .BYT $41,$4C,   ;  a
  $2C
4C                          ;  l
2C                          ;  ,
==25E0    MSG9              ;  Message 9
43          .BYT $43,$45,   ;  C
  $4E,$54,$52,$49,$46,
  $55,$47,$41,$4C,$20
45                          ;  E
4E                          ;  N
54                          ;  T
52                          ;  R
49                          ;  I
46                          ;  F
55                          ;  U
47                          ;  G
41                          ;  A
4C                          ;  L
20                          ;  space
50          .BYT $50,$55,   ;  P
  $4D,$50
55                          ;  U
4D                          ;  M
50                          ;  P
==25F0    MSG10             ;  Message 10
C8          .BYT $C8,$4F,   ;  H
  $57,$20,$4D,$41,$4E,
  $59,$20,$52,$4F,$4C
4F                          ;  o
57                          ;  w
20                          ;  space
4D                          ;  m
41                          ;  a
4E                          ;  n
59                          ;  y
20                          ;  space
52                          ;  r
4F                          ;  o
4C                          ;  l
4C          .BYT $4C,$45,   ;  l
  $52,$53
45                          ;  e
52                          ;  r
53                          ;  s
==2600    MSG11             ;  Message 11
D7          .BYT $D7,$48,   ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$48,$45,$20
48                          ;  h
41                          ;  a
```

```
54                          ;  t
20                          ;  space
49                          ;  i
53                          ;  s
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
54          .BYT $54,$55,   ;  t
  $42,$49,$4E,$47
55                          ;  u
42                          ;  b
49                          ;  i
==2610
4E                          ;  n
47                          ;  g
==2612    MSG12             ;  Message 12
C9          .BYT $C9,$53,   ;  I
  $20,$54,$48,$45,$20,
  $44,$49,$41,$4D,$45
53                          ;  s
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
44                          ;  d
49                          ;  i
41                          ;  a
4D                          ;  m
45                          ;  e
54          .BYT $54,$45,   ;  t
  $52,$20,$49,$4E
45                          ;  e
52                          ;  r
20                          ;  space
==2622
49                          ;  i
4E                          ;  n
==2624    MSG13             ;  Message 13
D7          .BYT $D7,$48,   ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$48,$45,$20
48                          ;  h
41                          ;  a
54                          ;  t
20                          ;  space
49                          ;  i
53                          ;  s
```

```
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
41          .BYT $41,$52,   ;  a
  $43
52                          ;  r
43                          ;  c
==2633    MSG14             ;  Message 14
C9          .BYT $C9,$53,   ;  I
  $20,$54,$48,$45,$20,
  $41,$52,$43,$20,$4C
53                          ;  s
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
41                          ;  a
52                          ;  r
43                          ;  c
20                          ;  space
4C                          ;  l
45          .BYT $45,$4E,   ;  e
  $47,$54,$48
4E                          ;  n
47                          ;  g
54                          ;  t
==2643
48                          ;  h
==2644    MSG15             ;  Message 15
52          .BYT $52,$4F,   ;  R
  $4C,$4C,$45,$52,$20,
  $50,$55,$4D,$50
4F                          ;  O
4C                          ;  L
4C                          ;  L
45                          ;  E
52                          ;  R
20                          ;  space
50                          ;  P
55                          ;  U
4D                          ;  M
50                          ;  P
==264F    MSG16             ;  Message 16
D7          .BYT $D7,$48,   ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$48,$45,$20
48                          ;  h
```

```
41                          ;  a
54                          ;  t
20                          ;  space
49                          ;  i
53                          ;  s
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
53          .BYT $53,$54,   ;  s
  $52,$4F,$4B,$45
54                          ;  t
52                          ;  r
4F                          ;  o
==265F
4B                          ;  k
45                          ;  e
==2661    MSG17             ;  Message 17
50          .BYT $50,$55,   ;  P
  $4C,$53,$41,$54,$49,
  $4C,$45,$20,$50,$55
55                          ;  U
4C                          ;  L
53                          ;  S
41                          ;  A
54                          ;  T
49                          ;  I
4C                          ;  L
45                          ;  E
20                          ;  space
50                          ;  P
55                          ;  U
4D          .BYT $4D,$50    ;  M
50                          ;  P
==266F    MSG18             ;  Message 18
C6          .BYT $C6,$4F,   ;  F
  $52,$20,$54,$48,$45,
  $20,$46,$4F,$4C,$4C
4F                          ;  o
52                          ;  r
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
46                          ;  f
4F                          ;  o
4C                          ;  l
4C                          ;  l
```

```
4F        .BYT $4F,$57,   ;  o
  $49,$4E,$47,$0D,$51,
  $55,$45,$53,$54,$49,
  $4F,$4E
57                        ;  w
49                        ;  i
4E                        ;  n
==267F
47                        ;  g
0D                        ;  carriage return
51                        ;  q
55                        ;  u
45                        ;  e
53                        ;  s
54                        ;  t
49                        ;  i
4F                        ;  o
4E                        ;  n
53        .BYT $53,$2C,   ;  s
  $20,$54,$48,$45,$0D,
  $54,$45,$4D,$50,$45,
  $52,$41
2C                        ;  ,
20                        ;  space
54                        ;  t
48                        ;  h
45                        ;  e
==268F
0D                        ;  carriage return
54                        ;  t
45                        ;  e
4D                        ;  m
50                        ;  p
45                        ;  e
52                        ;  r
41                        ;  a
54        .BYT $54,$55,   ;  t
  $52,$45,$53,$20,$42,
  $45,$49,$4E,$47,$0D,
  $4D,$45
55                        ;  u
52                        ;  r
45                        ;  e
53                        ;  s
20                        ;  space
42                        ;  b
45                        ;  e
==269F
49                        ;  i
4E                        ;  n
```

```
47                      ;  g
0D                      ;  carriage return
4D                      ;  m
45                      ;  e
41        .BYT $41,$53, ;  a
  $55,$52,$45,$44,$20,
  $53,$48,$4F,$55,$4C,
  $44,$20
53                      ;  s
55                      ;  u
52                      ;  r
45                      ;  e
44                      ;  d
20                      ;  space
53                      ;  s
48                      ;  h
4F                      ;  o
==26AF
55                      ;  u
4C                      ;  l
44                      ;  d
20                      ;  space
42        .BYT $42,$45, ;  b
  $0D,$4E,$41,$4D,$45,
  $44,$20,$49,$4E,$20,
  $38,$20
45                      ;  e
0D                      ;  carriage return
4E                      ;  n
41                      ;  a
4D                      ;  m
45                      ;  e
44                      ;  d
20                      ;  space
49                      ;  i
4E                      ;  n
20                      ;  space
==26BF
38                      ;  8
20                      ;  space
43        .BYT $43,$48, ;  c
  $41,$52,$41,$43,$54,
  $45,$52,$53,$20,$4F,
  $52,$20
48                      ;  h
41                      ;  a
52                      ;  r
41                      ;  a
43                      ;  c
54                      ;  t
```

```
45                          ;  e
52                          ;  r
53                          ;  s
20                          ;  space
4F                          ;  o
52                          ;  r
20                          ;  space
==26CF
4C          .BYT $4C,$45,   ;  l
  $53,$53,$2E,$20,$20,
  $C9,$46,$20,$41,$0D,
  $54,$45
45                          ;  e
53                          ;  s
53                          ;  s
2E                          ;  .
20                          ;  space
20                          ;  space
C9                          ;  I
46                          ;  f
20                          ;  space
41                          ;  a
0D                          ;  carriage return
54                          ;  t
45                          ;  e
4D          .BYT $4D,$50,   ;  m
  $45,$52,$41,$54,$55,
  $52,$45,$20,$50,$52,
  $4F,$42
50                          ;  p
==26DF
45                          ;  e
52                          ;  r
41                          ;  a
54                          ;  t
55                          ;  u
52                          ;  r
45                          ;  e
20                          ;  space
50                          ;  p
52                          ;  r
4F                          ;  o
42                          ;  b
45          .BYT $45,$20,   ;  e
  $49,$53,$20,$20,$4E,
  $4F,$54,$20,$42,$45,
  $49,$4E
20                          ;  space
49                          ;  i
53                          ;  s
```

```
==26EF
20                          ;  space
20                          ;  space
4E                          ;  n
4F                          ;  o
54                          ;  t
20                          ;  space
42                          ;  b
45                          ;  e
49                          ;  i
4E                          ;  n
47          .BYT $47,$20,   ;  g
  $55,$53,$45,$44,$2C,
  $20,$54,$48,$45,$4E,
  $20,$20
20                          ;  space
55                          ;  u
53                          ;  s
45                          ;  e
44                          ;  d
==26FF
2C                          ;  ,
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
4E                          ;  n
20                          ;  space
20                          ;  space
4E          .BYT $4E,$41,   ;  n
  $4D,$45,$20,$54,$48,
  $45,$20,$54,$45,$4D,
  $50,$45
41                          ;  a
4D                          ;  m
45                          ;  e
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
==270F
20                          ;  space
54                          ;  t
45                          ;  e
4D                          ;  m
50                          ;  p
45                          ;  e
52          .BYT $52,$41,   ;  r
  $54,$55,$52,$45,$20,
  $20,$CE,$2F,$C3,$20,
```

```
  $46,$4F
41                          ;  a
54                          ;  t
55                          ;  u
52                          ;  r
45                          ;  e
20                          ;  space
20                          ;  space
CE                          ;  N
2F                          ;  /
==271F
C3                          ;  C
20                          ;  space
46                          ;  f
4F                          ;  o
52          .BYT $52,$20,   ;  r
  $4E,$4F,$54,$20,$43,
  $4F,$4E,$4E,$45,$43,
  $54,$45
20                          ;  space
4E                          ;  n
4F                          ;  o
54                          ;  t
20                          ;  space
43                          ;  c
4F                          ;  o
4E                          ;  n
4E                          ;  n
45                          ;  e
43                          ;  c
==272F
54                          ;  t
45                          ;  e
44          .BYT $44        ;  d
==2732      MSG19           ;  Message 19
D7          .BYT $D7,$48,   ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$45,$4D,$50
48                          ;  h
41                          ;  a
54                          ;  t
20                          ;  space
49                          ;  i
53                          ;  s
20                          ;  space
54                          ;  t
45                          ;  e
4D                          ;  m
50                          ;  p
45          .BYT $45,$52,   ;  e
```

```
  $41,$54,$55,$52,$45
52                          ;  r
41                          ;  a
54                          ;  t
==2742
55                          ;  u
52                          ;  r
45                          ;  e
==2745    MSG20             ;  Message 20
C6          .BYT $C6,$4F,   ;  F
  $52,$20,$54,$48,$45,
  $20,$54,$45,$4D,$50
4F                          ;  o
52                          ;  r
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
54                          ;  t
45                          ;  e
4D                          ;  m
50                          ;  p
45          .BYT $45,$52,   ;  e
  $41,$54,$55,$52,$45,
  $0D,$44,$49,$53,$50,
  $4C,$41
52                          ;  r
41                          ;  a
54                          ;  t
==2755
55                          ;  u
52                          ;  r
45                          ;  e
0D                          ;  carriage return
44                          ;  d
49                          ;  i
53                          ;  s
50                          ;  p
4C                          ;  l
41                          ;  a
59          .BYT $59,$2C,   ;  y
  $20,$53,$45,$4C,$45,
  $43,$54,$0D,$45,$49,
  $54,$48
2C                          ;  ,
20                          ;  space
53                          ;  s
45                          ;  e
4C                          ;  l
```

```
==2765
45                          ;  e
43                          ;  c
54                          ;  t
0D                          ;  carriage return
45                          ;  e
49                          ;  i
54                          ;  t
48                          ;  h
45        .BYT $45,$52,     ;  e
  $20,$C6,$41,$52,$45,
  $4E,$48,$45,$49,$54,
  $0D,$4F
52                          ;  r
20                          ;  space
C6                          ;  F
41                          ;  a
52                          ;  r
45                          ;  e
4E                          ;  n
==2775
48                          ;  h
45                          ;  e
49                          ;  i
54                          ;  t
0D                          ;  carriage return
4F                          ;  o
52        .BYT $52,$20,     ;  r
  $C3,$45,$4C,$53,$49,
  $55,$53,$2E,$20,$20,
  $20
20                          ;  space
C3                          ;  C
45                          ;  e
4C                          ;  l
53                          ;  s
49                          ;  i
55                          ;  u
53                          ;  s
2E                          ;  .
==2785
20                          ;  space
20                          ;  space
20                          ;  space
==2788   MSG21              ;  Message 21
C6        .BYT $C6,$4F,     ;  F
  $52,$20,$54,$48,$45,
  $20,$46,$4F,$4C,$4C
4F                          ;  o
52                          ;  r
```

```
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
20                          ;  space
46                          ;  f
4F                          ;  o
4C                          ;  l
4C                          ;  l
4F          .BYT $4F,$57,   ;  o
  $49,$4E,$47,$0D,$51,
  $55,$45,$53,$54,$49,
  $4F,$4E
57                          ;  w
49                          ;  i
4E                          ;  n
==2798
47                          ;  g
0D                          ;  carriage return
51                          ;  q
55                          ;  u
45                          ;  e
53                          ;  s
54                          ;  t
49                          ;  i
4F                          ;  o
4E                          ;  n
53          .BYT $53,$2C,   ;  s
  $20,$54,$48,$45,$0D,
  $54,$49,$4D,$45,$52,
  $53,$20
2C                          ;  ,
20                          ;  space
54                          ;  t
48                          ;  h
45                          ;  e
==27A8
0D                          ;  carriage return
54                          ;  t
49                          ;  i
4D                          ;  m
45                          ;  e
52                          ;  r
53                          ;  s
20                          ;  space
4E          .BYT $4E,$41,   ;  n
  $4D,$45,$53,$20,$53,
  $48,$4F,$55,$4C,$44,
  $0D,$42
41                          ;  a
```

```
4D                      ;  m
45                      ;  e
53                      ;  s
20                      ;  space
53                      ;  s
48                      ;  h
==27B8
4F                      ;  o
55                      ;  u
4C                      ;  l
44                      ;  d
0D                      ;  carriage return
42                      ;  b
45        .BYT $45,$20, ;  e
  $36,$20,$43,$48,$41,
  $52,$41,$43,$54,$45,
  $52,$53
20                      ;  space
36                      ;  6
20                      ;  space
43                      ;  c
48                      ;  h
41                      ;  a
52                      ;  r
41                      ;  a
43                      ;  c
==27C8
54                      ;  t
45                      ;  e
52                      ;  r
53                      ;  s
20        .BYT $20,$4F, ;  space
  $52,$0D,$4C,$45,$53,
  $53,$2E
4F                      ;  o
52                      ;  r
0D                      ;  carriage return
4C                      ;  l
45                      ;  e
53                      ;  s
53                      ;  s
2E                      ;  .
==27D5    MSG22         ;  Message 22
D7        .BYT $D7,$48, ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$49,$4D,$45
48                      ;  h
41                      ;  a
54                      ;  t
20                      ;  space
```

```
49                      ;  i
53                      ;  s
20                      ;  space
54                      ;  t
49                      ;  i
4D                      ;  m
45                      ;  e
52        .BYT $52      ;  r
==27E2    MSG23         ;  Message 23
D7        .BYT $D7,$48, ;  W
  $41,$54,$20,$49,$53,
  $20,$54,$48,$45,$20
48                      ;  h
41                      ;  a
54                      ;  t
20                      ;  space
49                      ;  i
53                      ;  s
20                      ;  space
54                      ;  t
48                      ;  h
45                      ;  e
20                      ;  space
43        .BYT $43,$55, ;  c
  $52,$52,$45,$4E,$54
55                      ;  u
52                      ;  r
52                      ;  r
==27F2
45                      ;  e
4E                      ;  n
54                      ;  t
          *=4000
==4000    UPTEMP        ;  Updates the temperatures
ADE61D    LDA TMPFLG    ;  Gets temperature update flag
C9FF      CMP #$FF      ;  Is it set?
D07D      BNE RTURN     ;  If it isn't branches to return
                           from subroutine
A900      LDA #00
8DE61D    STA TMPFLG    ;  It is so clears update flag
ADE81D    LDA TMP1NC
C900      CMP #00       ;  Is temperature 1 connected?
==4011
D00D      BNE CHKTM2    ;  If it isn't branches to check
                           temperature 2
A902      LDA #02       ;  It is connected, so enables temp.
                           probe 1
8D029A    STA $9A02
2026A4    JSR TNEW      ;  Gets ASCII value from A/D
A000      LDY #00       ;  Sets screen location
```

```
208540    JSR PRTEMP      ;  Prints temperature 1
==4020    CHKTM2          ;  Checks temperature 2
ADE91D    LDA TMP2NC
C900      CMP #00         ;  Is temperature 2 connected?
D00D      BNE CHKTM3      ;  If it isn't branches to check
                             temperature 3
A904      LDA #04         ;  It is so enables temp. probe 2
8D029A    STA $9A02
2026A4    JSR TNEW        ;  Gets ASCII value from A/D
A00B      LDY #11         ;  Sets screen location
==4031
208540    JSR PRTEMP      ;  Prints temperature 2
==4034    CHKTM3          ;  Checks temperature 3
ADEA1D    LDA TMP3NC
C900      CMP #00         ;  Is temperature 3 connected?
D00D      BNE CHKTM4      ;  If it isn't branches to check
                             temperature 4
A908      LDA #08         ;  It is so enables temp. probe 3
8D029A    STA $9A02
2026A4    JSR TNEW        ;  Gets ASCII value from A/D
A02C      LDY #44         ;  Sets screen location
==4045
208540    JSR PRTEMP      ;  Prints temperature 3
==4048    CHKTM4          ;  Checks temperature 4
ADEB1D    LDA TMP4NC
C900      CMP #00         ;  Is temperature 4 connected?
D00D      BNE CHKTM5      ;  If it isn't branches to check
                             temperature 5
A910      LDA #16         ;  It is so enables temp. probe 4
8D029A    STA $9A02
2026A4    JSR TNEW        ;  Gets ASCII value from A/D
A037      LDY #55         ;  Sets screen location
==4059
208540    JSR PRTEMP      ;  Prints temperature 4
==405C    CHKTM5          ;  Checks temperature 5
ADEC1D    LDA TMP5NC
C900      CMP #00         ;  Is temperature 5 connected?
D00D      BNE CHKTM6      ;  If it isn't branches to check
                             temperature 6
A920      LDA #32         ;  It is so enables temp. probe 5
8D029A    STA $9A02
2026A4    JSR TNEW        ;  Gets ASCII value from A/D
A058      LDY #88         ;  Sets screen location
==406D
208540    JSR PRTEMP      ;  Prints temperature 5
==4070    CHKTM6          ;  Checks temperature 6
ADED1D    LDA TMP6NC
C900      CMP #00         ;  Is temperature 6 connected?
D00D      BNE RTURN       ;  If it isn't branches to return
                             from subroutine
```

```
A940      LDA #64          ;  It is so enables temp. probe 6
8D029A    STA $9A02
2026A4    JSR TNEW         ;  Gets ASCII value from A/D
A063      LDY #99          ;  Sets screen location
==4081
208540    JSR PRTEMP       ;  Prints temperature 6
==4084    RTURN
60        RTS              ;  Return from subroutine
==4085    PRTEMP           ;  Prints a temperature
ADF31D    LDA TEMPH        ;  Gets hundred's digit
C930      CMP #$30         ;  Is it a zero?
D005      BNE PRONE        ;  If it isn't branches to print it
A920      LDA #$20
8DF31D    STA TEMPH        ;  It is so space it
==4091    PRONE            ;  Prints hundred's digit
99371F    STA 7991,Y
ADF31D    LDA TEMPH
C920      CMP #$20         ;  Is the hundred's digit a space?
D00C      BNE PRINTT       ;  If it isn't branches to print
                              ten's digit
ADF41D    LDA TEMPT
C930      CMP #$30         ;  It is so is ten's digit 0?
D005      BNE PRINTT       ;  If it isn't branches to print it
==40A2
A920      LDA #$20
8DF41D    STA TEMPT        ;  It is so space it
==40A7    PRINTT           ;  Prints ten's digit
ADF41D    LDA TEMPT
99381F    STA 7992,Y
ADF51D    LDA TEMPU
99391F    STA 7993,Y       ;  Prints unit's digit
ADF61D    LDA TEMPTH
993B1F    STA 7995,Y       ;  Prints tenth's digit
==40B9
60        RTS              ;  Returns from subroutine
==40BA    DISABL           ;  Disables all flags and resets all
                              constants
A900      LDA #00
8DF91D    STA CHRCNT       ;  Resets input character counter to
                              zero
8DF31D    STA TEMPH        ;  Sets temperature hundred's digit
                              to zero
8DF41D    STA TEMPT        ;  Sets temperature ten's digit to
                              zero
8DF51D    STA TEMPU        ;  Sets temperature unit's digit to
                              zero
8DF61D    STA TEMPTH       ;  Sets temperature tenth's digit to
                              zero
==40CB
8DC61D    STA ROLFLG       ;  Roller pump not being used
```

```
8DC71D    STA PULFLG     ;  Pulsatile pump not being used
8DE51D    STA PULSES     ;  Sets number of pulses counted to
                             zero
8DEF1D    STA DEGUNT     ;  Sets degree unit to C
8DEE1D    STA MENFLG     ;  Do not return to menu
8DF11D    STA HALFCT     ;  Sets half second counter to zero
==40DD
8DCA1D    STA TM1FLG     ;  Do not update timer 1
8DE71D    STA TMPCNT     ;  Sets temperature update counter to
                             zero
8DE61D    STA TMPFLG     ;  Do not update temperatures
8DD11D    STA TMR1JF     ;  Clears timer 1 jiffy counter to 0
A930      LDA #$30       ;  Gets ascii code for 0
8DD01D    STA TMR1SL     ;  Sets timer 1 second's LB to 0
==40EE
8DCF1D    STA TMR1SH     ;  Sets timer 1 second's HB to 0
8DCE1D    STA TMR1ML     ;  Sets timer 1 minute's LB to 0
8DCD1D    STA TMR1MH     ;  Sets timer 1 minute's HB to 0
8DCC1D    STA TMR1HL     ;  Sets timer 1 hour's LB to 0
8DCB1D    STA TMR1HH     ;  Sets timer 1 hour's HB to 0
A900      LDA #00
==40FF
8DD21D    STA TM2FLG     ;  Do not update timer 2
8DD91D    STA TMR2JF     ;  Clears timer 2 jiffy counter to 0
A930      LDA #$30       ;  Gets ascii code for 0
8DD81D    STA TMR2SL     ;  Sets timer 2 second's LB to 0
8DD71D    STA TMR2SH     ;  Sets timer 2 second's HB to 0
8DD61D    STA TMR2ML     ;  Sets timer 2 minute's LB to 0
==4110
8DD51D    STA TMR2MH     ;  Sets timer 2 minute's HB to 0
8DD41D    STA TMR2HL     ;  Sets timer 2 hour's LB to 0
8DD31D    STA TMR2HH     ;  Sets timer 2 hour's HB to 0
A900      LDA #00
8DDA1D    STA TM3FLG     ;  Do not update timer 3
8DE11D    STA TMR3JF     ;  Clears timer 3 jiffy counter to 0
==4121
A930      LDA #$30       ;  Gets ascii code for 0
8DE01D    STA TMR3SL     ;  Sets timer 3 second's LB to 0
8DDF1D    STA TMR3SH     ;  Sets timer 3 second's HB to 0
8DDE1D    STA TMR3ML     ;  Sets timer 3 minute's LB to 0
8DDD1D    STA TMR3MH     ;  Sets timer 3 minute's HB to 0
8DDC1D    STA TMR3HL     ;  Sets timer 3 hour's LB to 0
==4132
8DDB1D    STA TMR3HH     ;  Sets timer 3 hour's HB to 0
A900      LDA #00
8DED1D    STA TMP6NC     ;  Temperature probe 6 is connected
8DE31D    STA CLKFLG     ;  Do not update clock on screen
8DE41D    STA CLKJIF
8DE21D    STA PRCLKF
```

```
==4143
8DC81D    STA CENFLG      ;  Centrifugal pump is not being used
8DC91D    STA CENCNT
8DC51D    STA PMPFLG      ;  Do not update pump output
8DEF1D    STA DEGUNT      ;  Sets degree unit to C
8DE81D    STA TMP1NC      ;  Temperature probe 1 is connected
8DE91D    STA TMP2NC      ;  Temperature probe 2 is connected
==4155
8DEA1D    STA TMP3NC      ;  Temperature probe 3 is connected
8DEB1D    STA TMP4NC      ;  Temperature probe 4 is connected
8DEC1D    STA TMP5NC      ;  Temperature probe 5 is connected
A940      LDA #$40        ;  Disables interrupt enable register
8D0E98    STA IO1IER
60        RTS             ;  Returns from subroutine
```

APPENDIX B

Screen Displays

This appendix shows the title and monitor screens displayed following the calls to subroutines SCREN1 and SCREN2, respectively.

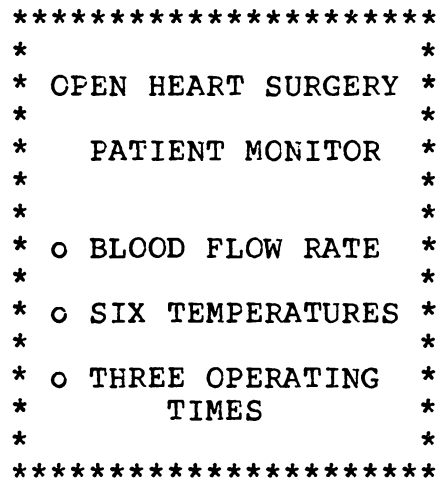


Figure B1. Title Screen Display after SCREN1 Subroutine Call

```
HEIGHT =     . cm
       =     . in
WEIGHT =     . kg
       =     . lb
BODY SURFACE AREA
       =   .   m**2

OUTPUT =      . ml/min
BLOOD FLOW RATE
 =      . ml/min/kg
 =      . ml/min/m**2
    TEMPERATURES
1-          2-
     . C          . C
3-          4-
     . C          . C
5-          6-
     . C          . C
CURRENT TIME    :  :
       TIME1    :  :
       TIME2    :  :
       TIME3    :  :
```

Figure B2. Monitor Screen Display after SCREN2 Subroutine Call

APPENDIX C

VIC 20 Power-Up Initialization Sequence

The VIC 20 power-up initialization routine is listed in this appendix.

Hex Address	Op Code		Comment
FD22	LDX #$FF	;	
	SEI	;	Disable interrupts
	TXS	;	Set stack pointer to $00FF
	CLD	;	
	JSR $FD3F	;	Test for Auto-Start sequence
	BNE $FD2F	;	If not found, continue
	JMP ($A000)	;	If found, jump to user routine
FD2F	JSR $FD8D	;	RAM test and clear
	JSR $FD52	;	Set vectors
	JSR $FDF9	;	Initialize I/O
	JSR $E518	;	Initialize screen
	CLI	;	Enable Interrupts
	JMP ($C000)	;	BASIC Area

APPENDIX D

This appendix shows the layout of the interface circuit board. All connections were made using wire wrap on tin or gold plated sockets. The sockets were attached to perf board using silicon rubber.

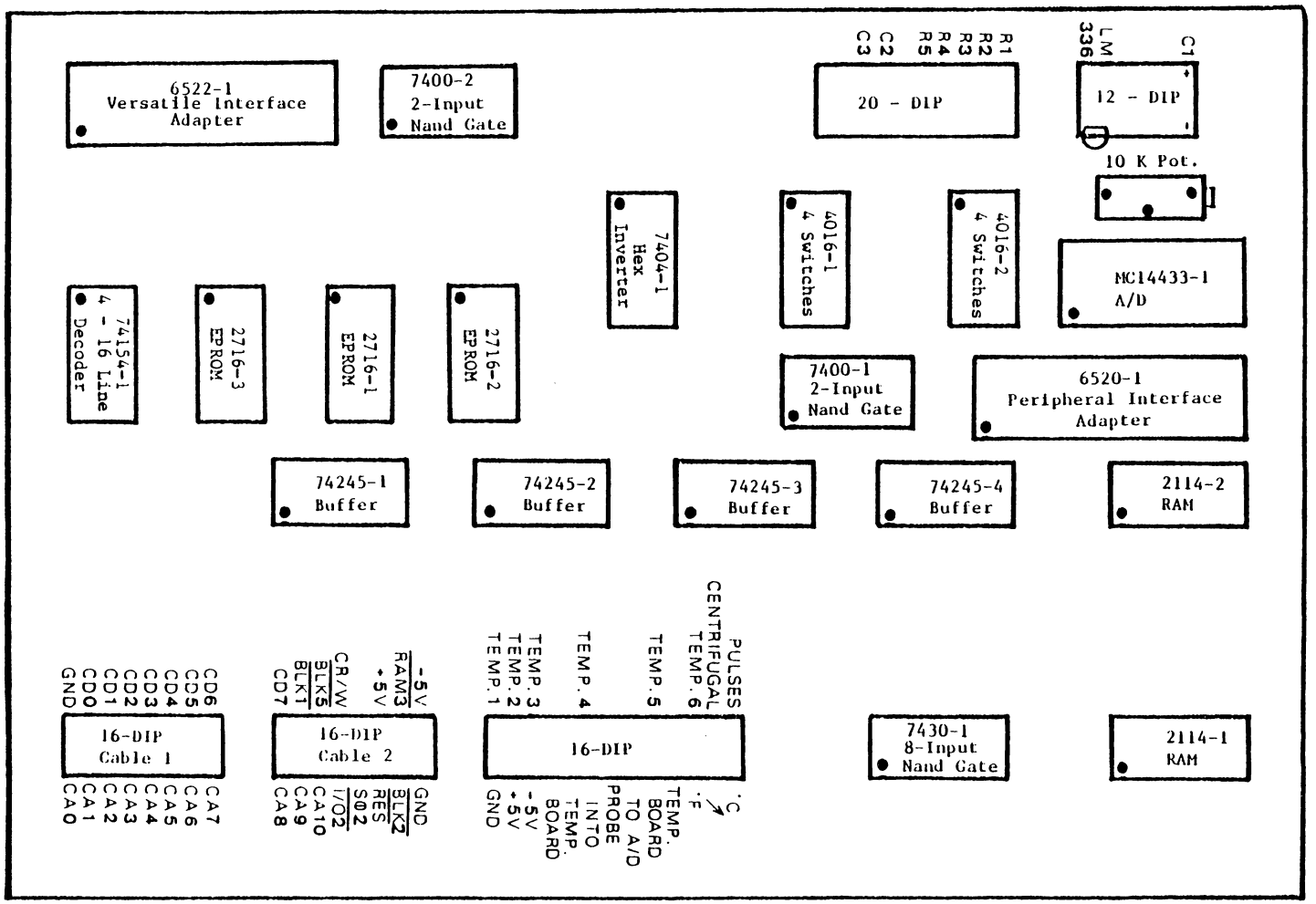


Figure D1. Interface Circuit Board Layout

APPENDIX E

This appendix shows the VIC 20 schematic. It should be noted, however, that the coding for the components shown in the schematic does not always agree with the coding on the the actual printed circuit board.

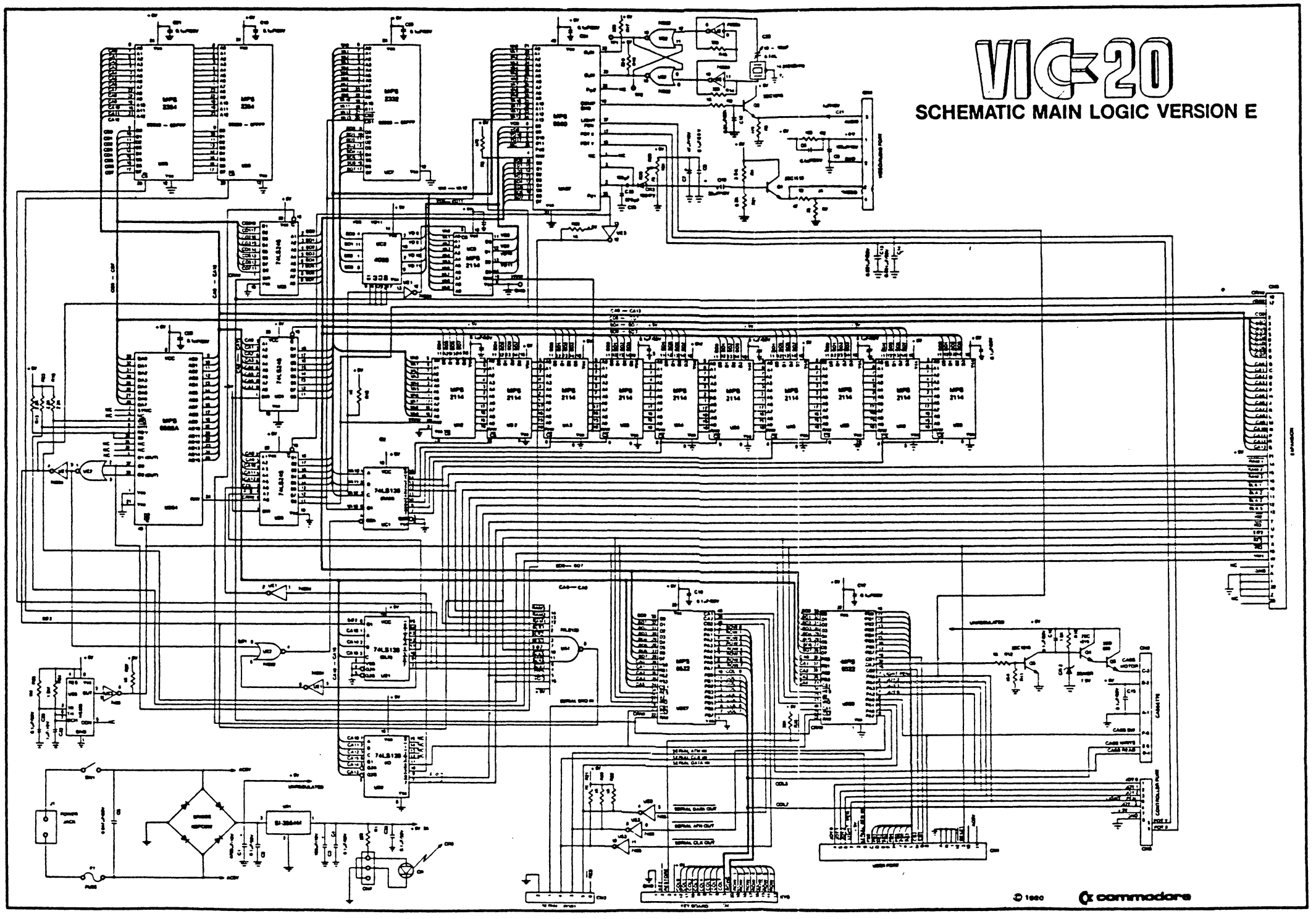


Figure E1. VIC 20 Schematic

The vita has been removed from the scanned document