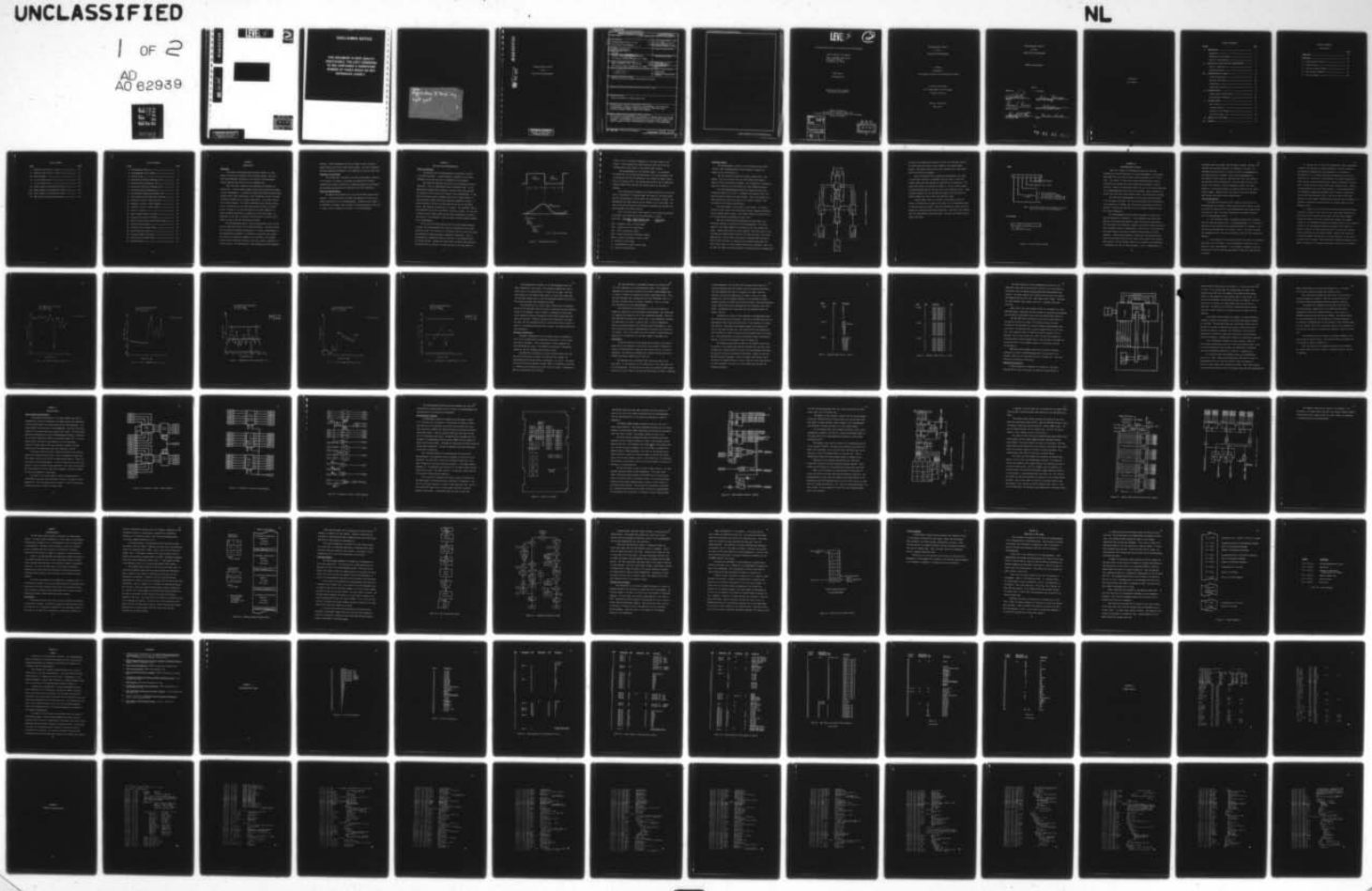
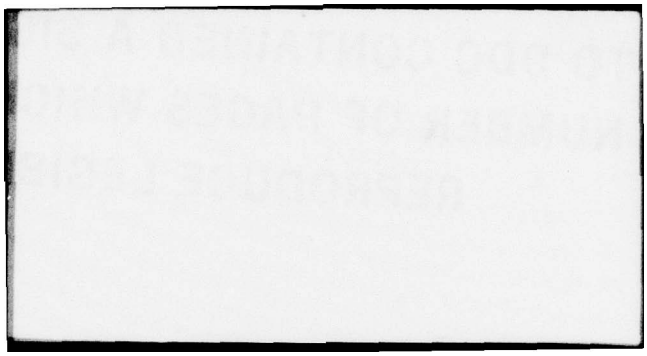
AD-A062 939 ARMY MILITARY PERSONNEL CENTER ALEXANDRIA VA F/G 6/5
MICROPROCESSOR CONTROL OF THE BLOOD FLOW PLETHYSMOGRAPH.(U)
DEC 78 W E SEIBERLING

UNCLASSIFIED NL

1 OF 2
AD
A062939





DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Note:
Appendices "B" thru E: very
light print

AD A062939

DDC FILE COPY

Microprocessor Control of the Blood Flow Plethysmograph

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*

Microprocessor Control of the Blood Flow Plethysmograph.

5. TYPE OF REPORT & PERIOD COVERED

Final Report, 15 DEC 78

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

WALTER E. SEIBERLING
CPT, MI
US Army

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Student, HQDA, MILPERCEN (DAPC-OPP-E),
200 Stovall Street, Alexandria, VA 22332

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

HQDA, MILPERCEN, ATTN: DAPC-OPP-E,
200 Stovall Street, Alexandria, VA 22332

12. REPORT DATE

15 December 1978

13. NUMBER OF PAGES

139

14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*

154p.

15. SECURITY CLASS. *(of this report)*

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Thesis, University of Akron, Akron, Ohio

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Microprocessor, Control, Blood Flow, Plethysmograph, Intel SBC 80/10, Floating-point arithmetic, 8080 Microprocessor, Block Structured Assembly Language (BSAL), Single Board Computer

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Blood Flow Plethysmograph, an instrument to measure blood flow through non-invasive techniques, is automated with an Intel SBC 80/10 single board computer. The automated system provides a real-time output capability, reduces operator's error, and improves the accuracy of the measurement.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 391 191

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

B

LEVEL II

2

Microprocessor Control of the Blood Flow Plethysmograph

Captain Walter E. Seiberling

HQDA, MILPERCEN (DAPC-OPP-E)
200 Stovall Street
Alexandria, VA 22332

Final Report

14 December 1978

Approved For Public Release;
Distribution Unlimited.

A Thesis Submitted to
The University of Akron, Akron, Ohio
in Partial Fulfillment of the Requirements for the Degree
Master of Science

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A 23 64

DDC
RECEIVED
JAN 8 1979
D

Microprocessor Control
of the
Blood Flow Plethysmograph

A Thesis

Presented To

The Graduate Faculty of the University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Walter E. Seiberling

May, 1979

Microprocessor Control

of the

Blood Flow Plethysmograph

Walter E. Seiberling

Thesis

Approved:

Adviser

Faculty Reader

Department Head

Accepted:

Dean of the College

Dean of the Graduate School

December 15th, 1978

Date

79 01 02 045

Dedicated

to my family

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
	Background	1
	Statement of the Problem	2
	Method of Implementation	2
II.	BACKGROUND OF THE BLOOD FLOW PLETHYSMOGRAPH	3
	Theory of Operation	3
	Electrical Design	6
III.	CONSIDERATIONS OF DESIGN	10
	Initial Observations	11
	Equipment Availability	18
	Input/Output	19
	Operational Description	23
IV.	HARDWARE DESIGN	27
	Slope Computer Modifications	27
	Microprocessor Interface	31
V.	SOFTWARE DESIGN	41
	Data Storage	41
	Interrupt Handler	44
	Floating Point Programs	47
	Utility Programs	50
VI.	OPERATION OF THE SYSTEM	51
VII.	SUMMARY	55

TABLE OF CONTENTS

(continued)

	PAGE
REFERENCES	56
APPENDICES	57
A. Interconnection Tables	57
B. Linked Listing	66
C. Interrupt Handler Program	69
D. Floating Point Programs	94
E. Utility Programs	116

LIST OF TABLES

TABLE		PAGE
I.	Parallel Input Ports 1, 2 and 3	21
II.	Parallel Input Ports 4, 5 and 6	22
III.	Error Messages	54
IV.	J-4 Pin Connections	58
V.	J-5 Pin Connections	59
VI.	Slope Computer Interconnection, Card 5	60
VII.	Slope Computer Interconnection, Card 6	61
VIII.	Slope Computer Interconnection, Card 7	62
IX.	SBC 80/10 and B-board Interconnection	63

LIST OF FIGURES

FIGURE		PAGE
1.	Plethysmograph Timing	4
2.	Plethysmograph Block Diagram	7
3.	Printer Format	9
4.	Calibration Difference Readings, Run 1	13
5.	First Calibration Readings, Run 1	14
6.	Calibration Difference Readings, Run 4	15
7.	First Calibration Readings, Run 4	16
8.	Circumference Difference Readings, Run 4	17
9.	Block Diagram of Microprocessor Interface	24
10.	Schematic of Card 5, Slope Computer	28
11.	Schematic of Card 6, Slope Computer	29
12.	Schematic of Card 7, Slope Computer	30
13.	Layout of B-board	32
14.	Memory Mapped Decoders, B-board	34
15.	Keyboard Input Section, B-board	36
16.	Digital Panel Printer Output Ports, B-board	38
17.	Timer Section, B-board	39
18.	Difference Data Storage Format	43
19.	Main Program Flow Chart	45
20.	Interrupt Handler Flow Chart	46
21.	Floating Point Number Format	49
22.	Output Sequence	53

CHAPTER I

INTRODUCTION

Background

One method of detecting cardio-vascular disease is by the amount of blood flow throughout the circulatory system. The blood flow plethysmograph, developed by the Institute of Environmental Stress, measures this blood flow to an isolated limb.

The blood flow is measured non-invasively by occluding the venous flow. A mercury strain gauge is then used to detect the change in limb circumference. An operational amplifier network provides bridge isolation and signal conditioning. This information is then converted to digital information by a digital panel meter. The system contains two of these channels. The digital information from the panel meters is then multiplexed into a single digital panel printer to provide output. Changes in the limb circumference are then compared to a known calibrated percentage of elongation in the strain gauge. An algorithm is then used to calculate the blood flow in terms of milliliters per minute per 100 milliliters of tissue.

The plethysmograph uses an intricate asynchronous logic network for the timing and control of the system operations. The actual blood flow is derived by applying the average of a series of measurements of the differential increases in circumference, along with the average of a series of calibration readings to an algorithm. The original system outputs individual calibration and size increase information on a panel printer. The differences of these readings were then averaged

manually. Other information such as the length of time intervals, gauge lengths and limb size were noted manually. All this information was later analyzed and applied to the algorithm to find the blood flow.

Statement of the Problem

The problem was to include an on-board microprocessor interface to the system. The goal is to provide the capability of quickly assimilating the data, obtaining the remaining parameters as necessary and providing the operator with immediate blood flow information.

Method of Implementation

The system was interfaced with an Intel SBC 80/10 single board computer. The microprocessor interface was designed to require only minimal modification of the plethysmograph. Peripherals were added to the SBC 80/10 to measure time intervals, to provide keyboard inputs and to output control information and data to the plethysmograph.

CHAPTER II

THE BLOOD FLOW PLETHYSMOGRAPH

Theory of Operation

The operation of the plethysmograph is described in detail in the User's Manual[1]. A brief summary is necessary to the understanding of this work and will be included for this purpose.

The method of obtaining blood flow is by occluding the venous blood flow. This is accomplished with an inflatable cuff. In the beginning the system waits a period of time (2.5 sec.-first period) for stability of the time rate of change in the circumference. At the end of this first period a reading of arm size is taken (Figure 1), using the strain gauge, bridge and operational amplifier network. A second reading of arm size is taken again at the end of the second delay. Each reading is printed as it is taken. The inflatable cuff is then deflated and the limb is allowed to return to normal. At this time a reading counter number indicating the first measurement is printed. This cycle is then repeated a number of times to obtain adequate samples of the measurements.

Since only the relative amount that the strain gauge stretched is known, the plethysmograph has a built in calibration circuit which simulates either a 0.2 percent or 1.0 percent stretch of the strain gauge. The calibration reading provides a standard of comparison. This is accomplished by temporarily removing resistors in the opposite leg of the bridge network. Readings are taken before and after the resistance is removed from the circuit. We then are able to tell the effect of

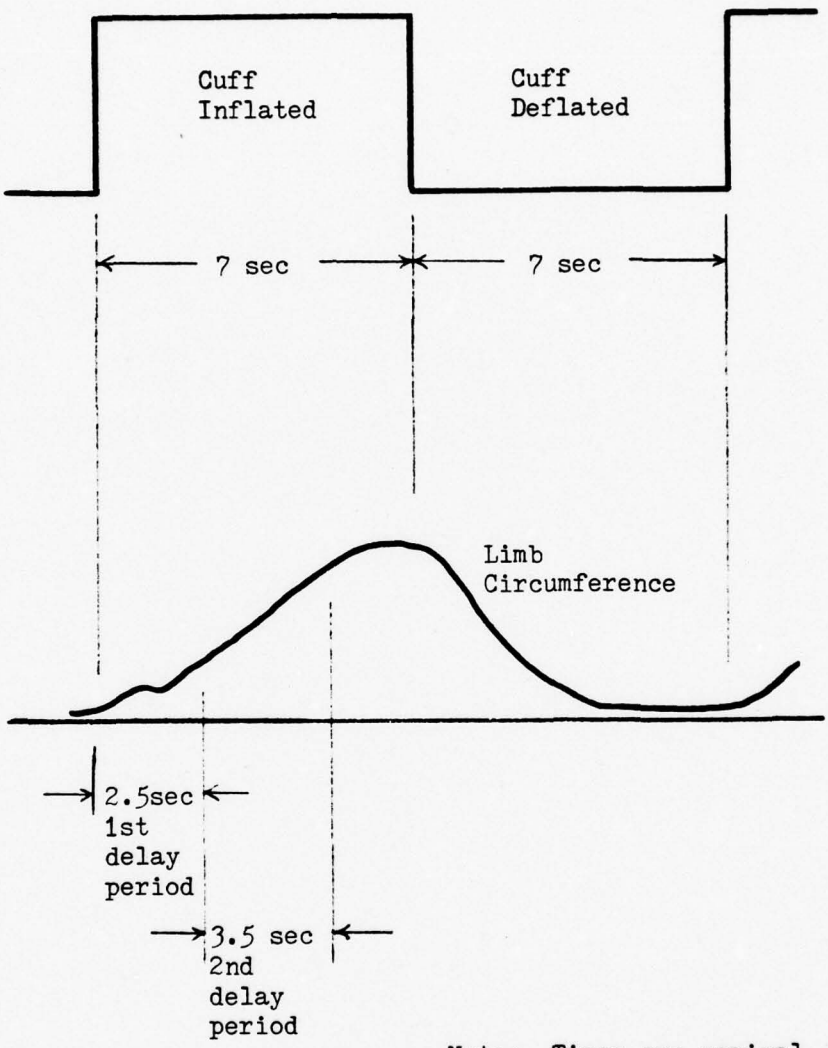


Figure 1 - Plethysmograph Timing

either a 0.2 or 1.0 percent elongation of the strain gauge on the circuit. These readings are taken before and after the blood flow readings and are also output on the digital panel printer.

The plethysmograph is a two channel system. It can measure the blood flow in two limbs simultaneously. All output readings, such as calibration and limb circumference, are provided as signed integer data in the range $\pm$ 1999. The digital panel printer has two additional leading numbers which identify the channel number and the type of reading.

Other information necessary to the final computation of the blood flow is the period of the second delay, the length of the strain gauge and the circumference of the arm where the strain gauge is placed. The measurements are taken directly by the operator with a steel tape. The time of the second delay may be adjusted as necessary and read from the plethysmograph front panel controls.

All of the above data is then applied to the following algorithm[1] to determine the blood flow per 100 milliliters of tissue.

$$\text{Blood Flow} = \frac{120\ (RDG2 - RDG1)\ (Factor)\ (Lg)}{(CAL2 - CAL1)\ (C_i)\ (T_s)} \quad \frac{\text{milliliters}}{\text{minute}}$$

RDG1 - reading at start of second delay

RDG2 - reading at end of second delay

CAL1 - first calibration reading

CAL2 - second (stretched) calibration reading

Factor - percent calibration (0.2% or 1.0%)

Lg - length of strain gauge

C_i - circumference of arm at strain gauge

T_s - period of second delay

Electrical Design

The plethysmograph consists of three principal units (cuff inflator, dual plethysmograph and slope computer) mounted in a single 19 inch rack (Figure 2).

The cuff inflator provides the overall timing cycle to the system. This unit controls when the cuff inflates and starts a series of limb circumference readings. Also included are the pressure valves for turning on and controlling the cuff inflation pressure.

The dual plethysmograph contains the analog circuitry and calibration logic for the unit. Included in the analog circuit are gain and balance controls for each channel. These are adjusted such that readings are within the range of the analog-to digital (A/D) converters of the digital panel meters.

The slope computer is the heart of the system. It contains the main control logic, the digital panel meters, digital multiplexers and the digital panel printer. The signals needed for interface with the microprocessor are available in this unit.

When limb circumference measurements are being taken, the following sequence is initialed in the plethysmograph. The timing modules in the slope computer are prompted by the cuff inflator and timer. Trigger pulses from the slope computer timers are output to the controller and digital panel meters at the start and end of the second delay period. This initiates the analog to digital (A/D) conversion. When the A/D conversion is complete and the digital panel meter has valid output data, the status signal notifies the controller. Data over or under the range of the A/D converter is indicated by the overload line.

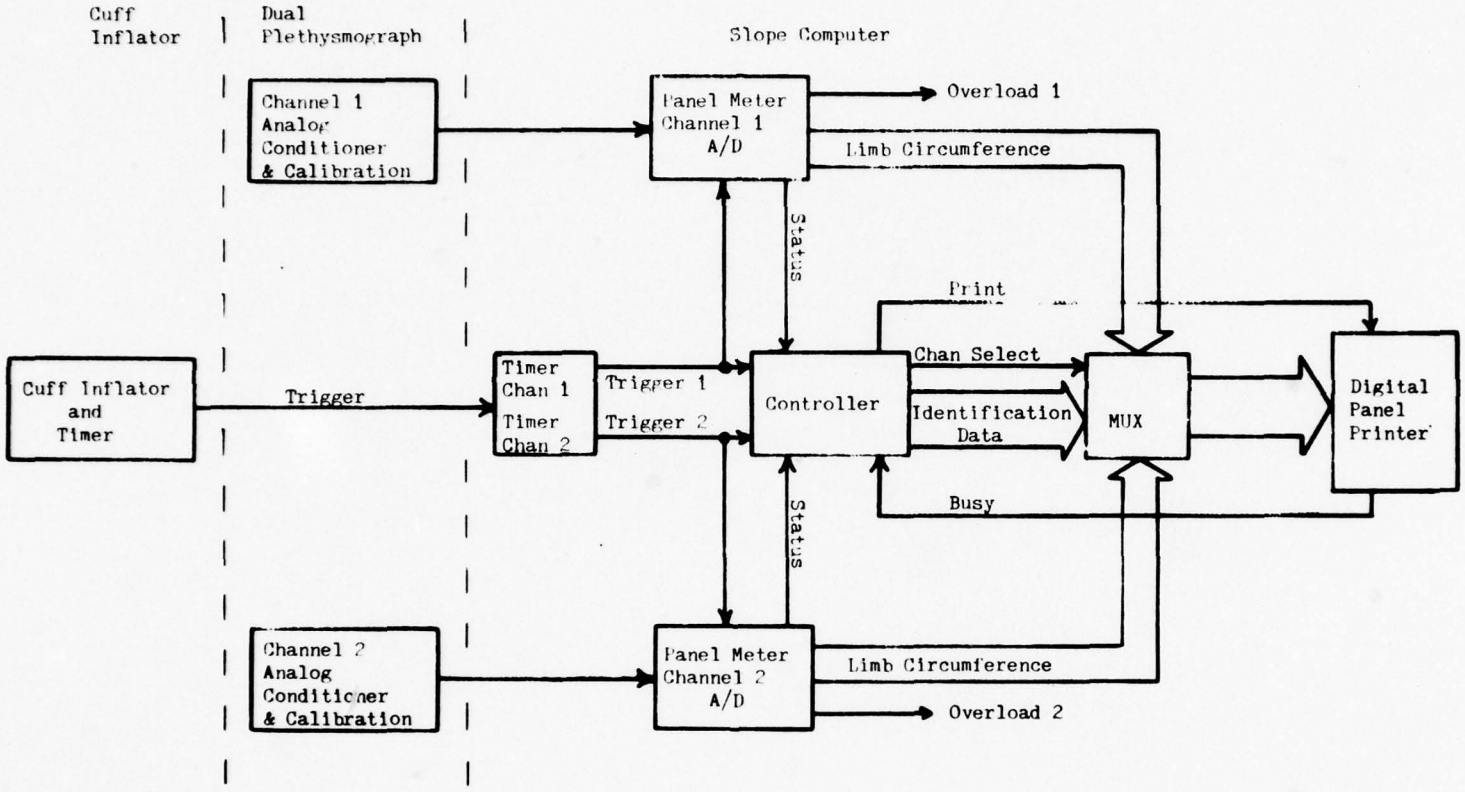


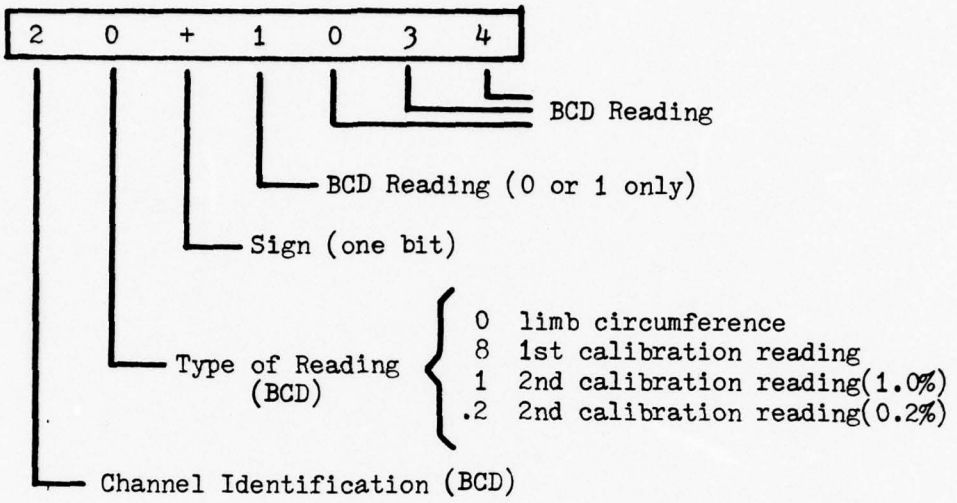
Figure 2 - Plethysmograph Block Diagram[1]

As long as the digital panel printer is free, the controller selects the multiplexer and issues a print command to the digital panel printer. The digital panel printer then responds with a busy signal until the printing is complete.

Since two channels may be operated simultaneously, the order in which data is printed is dependent upon the order in which the trigger pulses from each channel are received by the controller. The output data format is described in figure 3. The run counter number is printed sequentially at the end of each cuff inflation cycle. Unlike the data printing, the run counter uses negative true polarity and thereby signals the end of each cuff inflation cycle.

Another aspect which is critical to the printer is that of timing. The data must be valid at the input for 0.5 microsecond prior to issuing the print command. The print command must remain valid for between one microsecond and 200 milliseconds. The printer then remains busy for approximately 330 milliseconds. The busy line remains active during this time.

Data



Note: Six decimal points are available (six bits). Only one is used for 0.2% calibration.

Run Counter

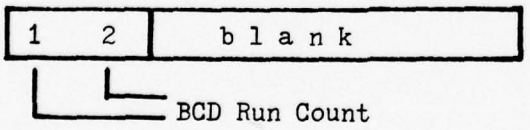


Figure 3 - Printer Output Format[1]

CHAPTER III

CONSIDERATIONS OF DESIGN

There are a number of considerations that went into the integration of a microprocessor into the plethysmograph. Of principal importance were cost, and the need to keep the system in one 19 inch rack. The system, of course, must remain somewhat mobile to allow flexibility for making blood flow measurements throughout a medical facility. These two requirements dictate the use of the printer available in the plethysmograph rather than a CRT terminal. Since the amount of manual input and output would remain minimal, this posed little problem.

The blood flow plethysmograph is a prototype system and needs to be evaluated clinically. The avoid any problems with the evaluation of the system as it existed, the circuitry was disturbed as little as possible. Changes were made only to tap off digital signals and some key timing information.

The plethysmograph requires manual adjustment of balance and gain settings prior to operation. It was suggested that these might possibly be automated and therefore reduce the operator requirements. This possibility was considered. However, there were several indications that the manual control is satisfactory. First of all it was found that the extreme range on the gain and balance settings were not really necessary. During normal operation a single gain and balance setting could be found which would keep the output within range on the digital panel meters. The fine balance needed to be adjusted only occasionally to compensate for drift or other variables. Cost of implementing an

automated system was greater than the benefit reaped. Secondly, the readings were extremely sensitive to tenseness in the limb. An automated system would have to insure it did not overcompensate for any temporary muscle tenseness. Since the plethysmograph is a prototype of limited quantity, there has been little data gathered and therefore little experience as to all the range of uses of the system. For instance, it may be necessary to output the analog signal to a strip chart recorder to observe linearity of the circumference change. Therefore, for this reason it was also concluded that the gain and balance should remain manual.

Initial Observations

The system was operated and the overall functioning of the plethysmograph was observed. These observations were critical in that any system interfacing with the plethysmograph must take these factors into account. Some of the major observations concerning interface with the plethysmograph follow:

1. The calibration logic is manually initiated and is separate from the remaining digital logic in the plethysmograph. It is therefore possible for the calibration and measurement cycles to interfere. This can cause questionable data to be output. Also, it is easy to release the calibration button too soon and therefore lose the second calibration reading.

2. Any tenseness in the limb would cause the output of the digital panel meter to be off scale. This was displayed as all zeros on the printer, a still valid reading. It was therefore necessary to use the overload line from the digital panel meter to sense this condition when it occurs.

3. Several sets of blood flow readings were taken and analyzed. A graph of the calibration reading differences is shown in figure 4. All difference readings have been normalized to a 1.0 percent calibration; i.e. 0.2 percent calibrations readings have been multiplied by five. In this set of measurements ten balance readings were taken and then blood flow readings were started. Calibration readings were continued during the seven second cuff deflation cycle. There is a large variance in the last eight readings. It was determined that the calibration reading was dependent upon the state of recovery of the arm after the inflation cycle. This can be seen by the large variance in first balance readings taken during the same run (figure 5). Therefore, calibration readings should not be taken during the cuff deflation cycle.

4. Another set of more uniform calibration difference readings are shown in figure 6. The first ten were taken before measuring the blood flow and the remaining after taking all blood flow readings. Although the value of the first calibration readings have changed (figure 7), the calibration difference readings (figure 6) are consistent. Looking at the circumference difference readings for the same run, figure 8, they are not particularly grouped. This may have resulted since the subject was not totally relaxed. However this, along with the previous graph, indicates that it may be desirable to cull some of the points from the data. This would delete difference readings, which might be indicative of momentary tenseness or movement of the subject.

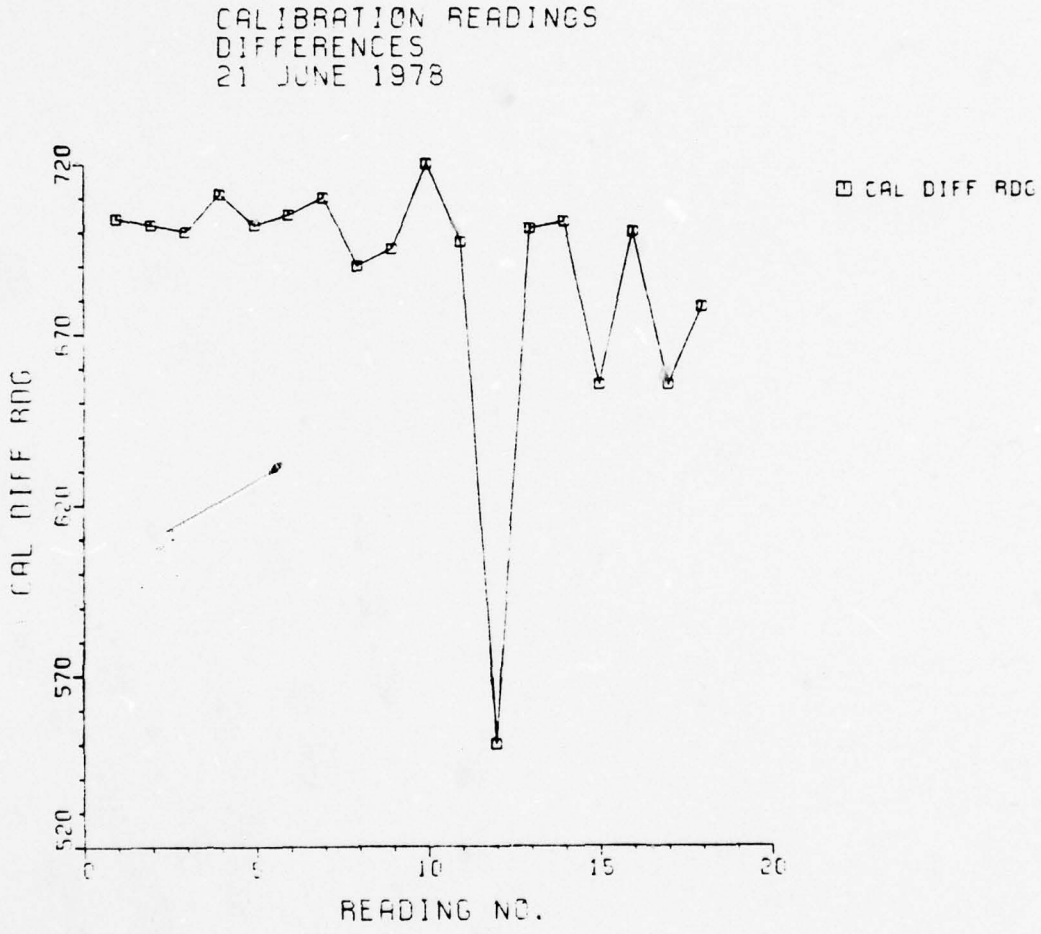


FIGURE 4 - CALIBRATION DIFFERENCE READINGS, RUN 1

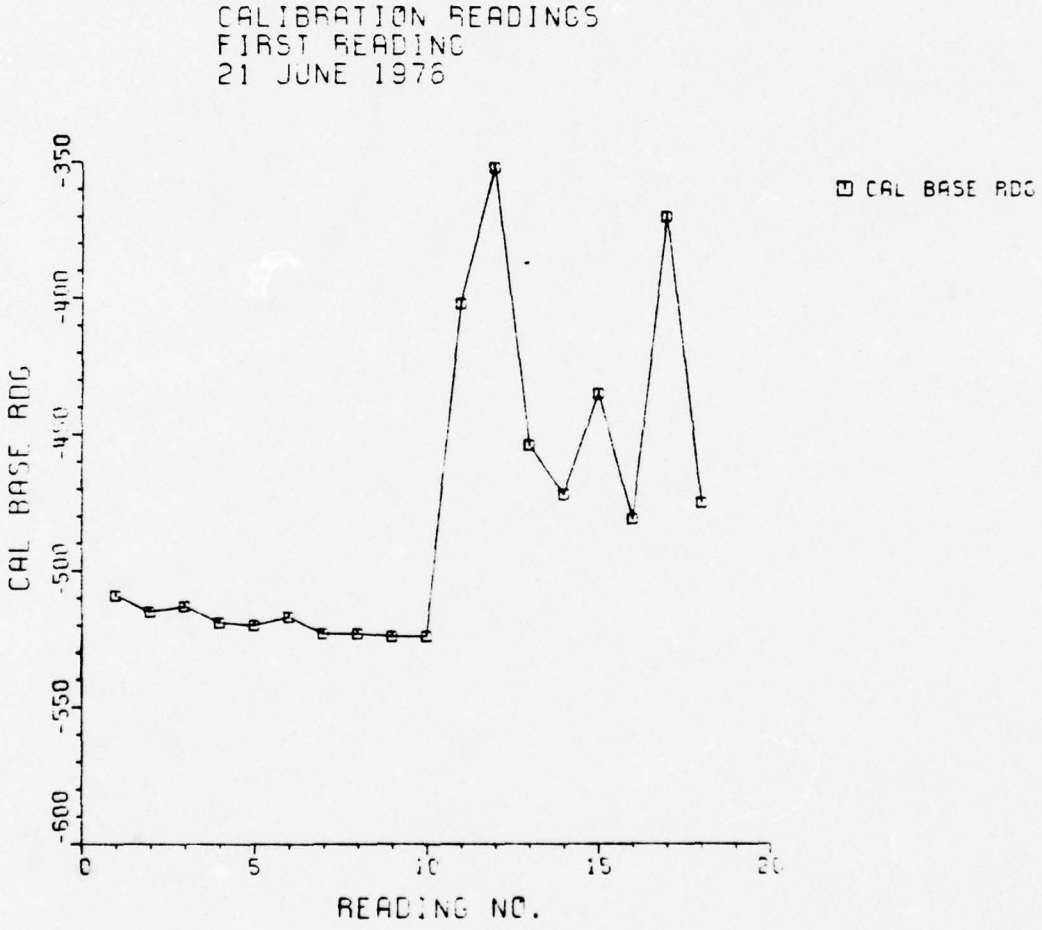


FIGURE 5 - FIRST CALIBRATION READINGS, RUN 1

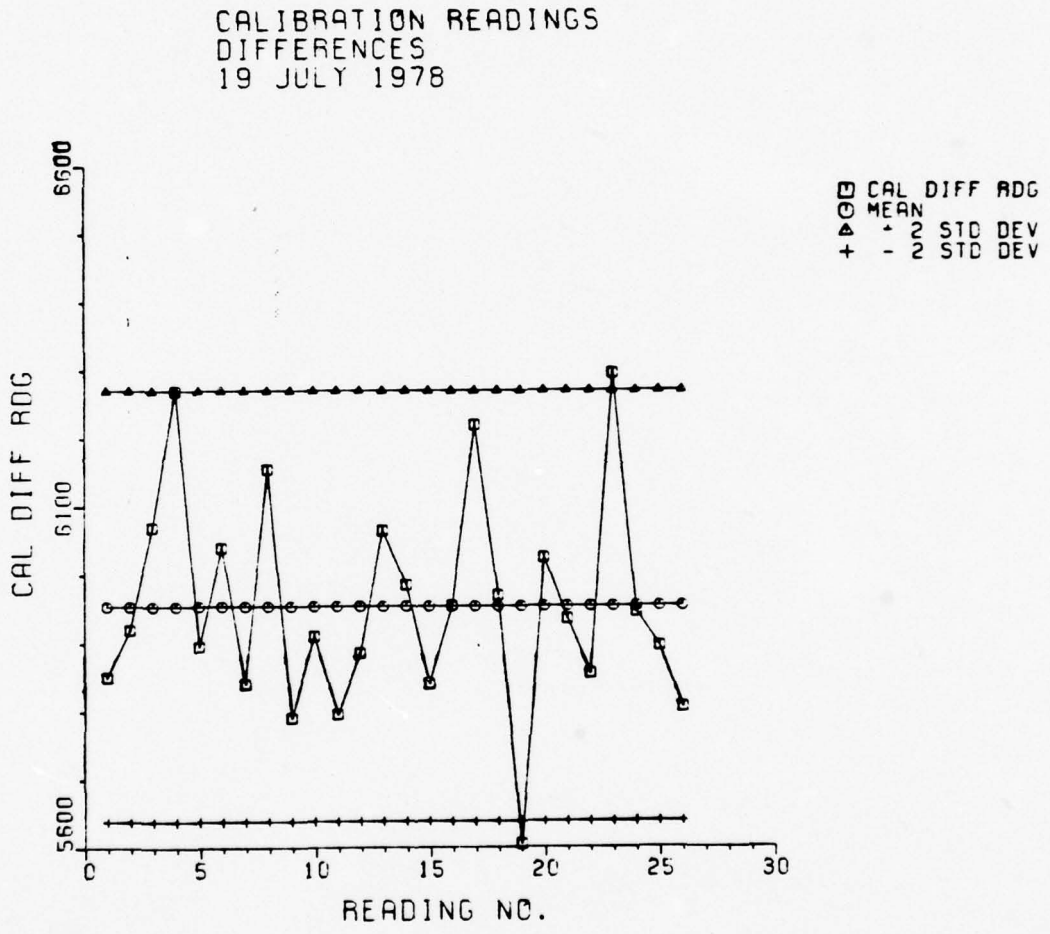


FIGURE 6 - CALIBRATION DIFFERENCE READINGS, RUN 4

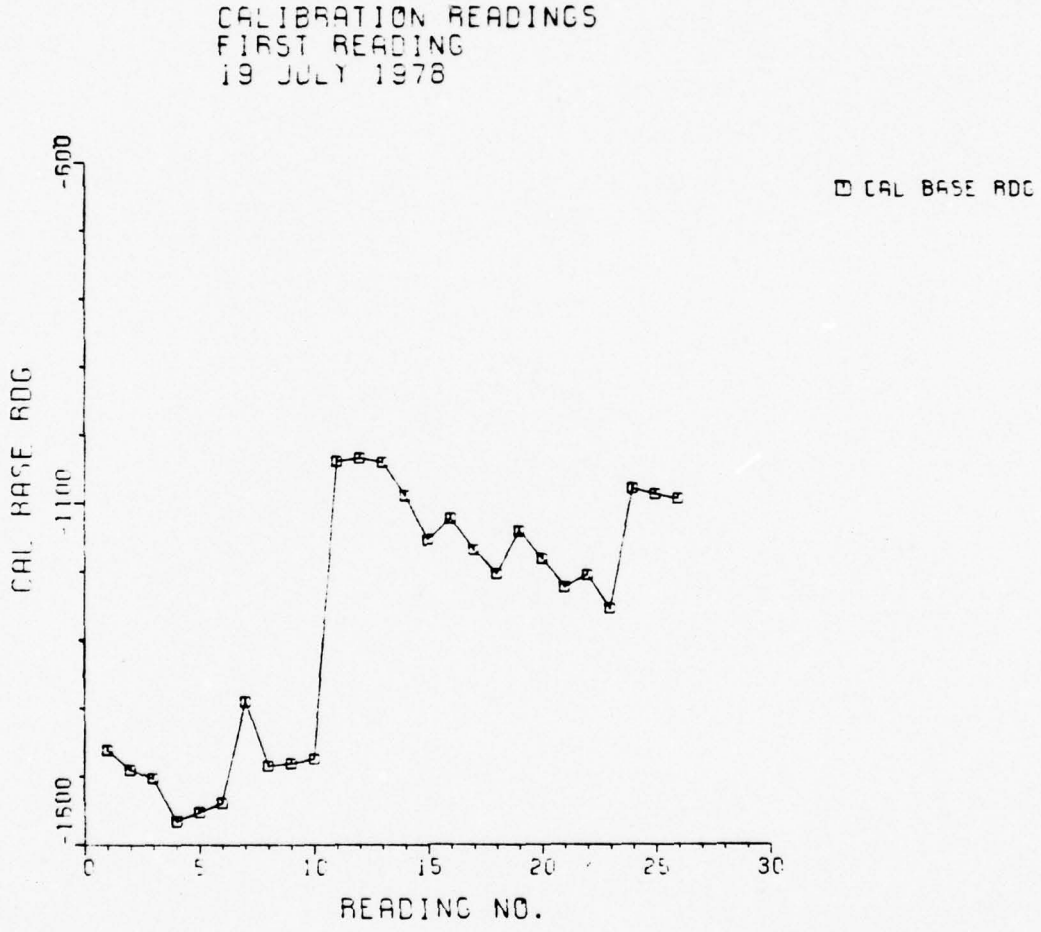


FIGURE 7 - FIRST CALIBRATION READINGS, RUN 4

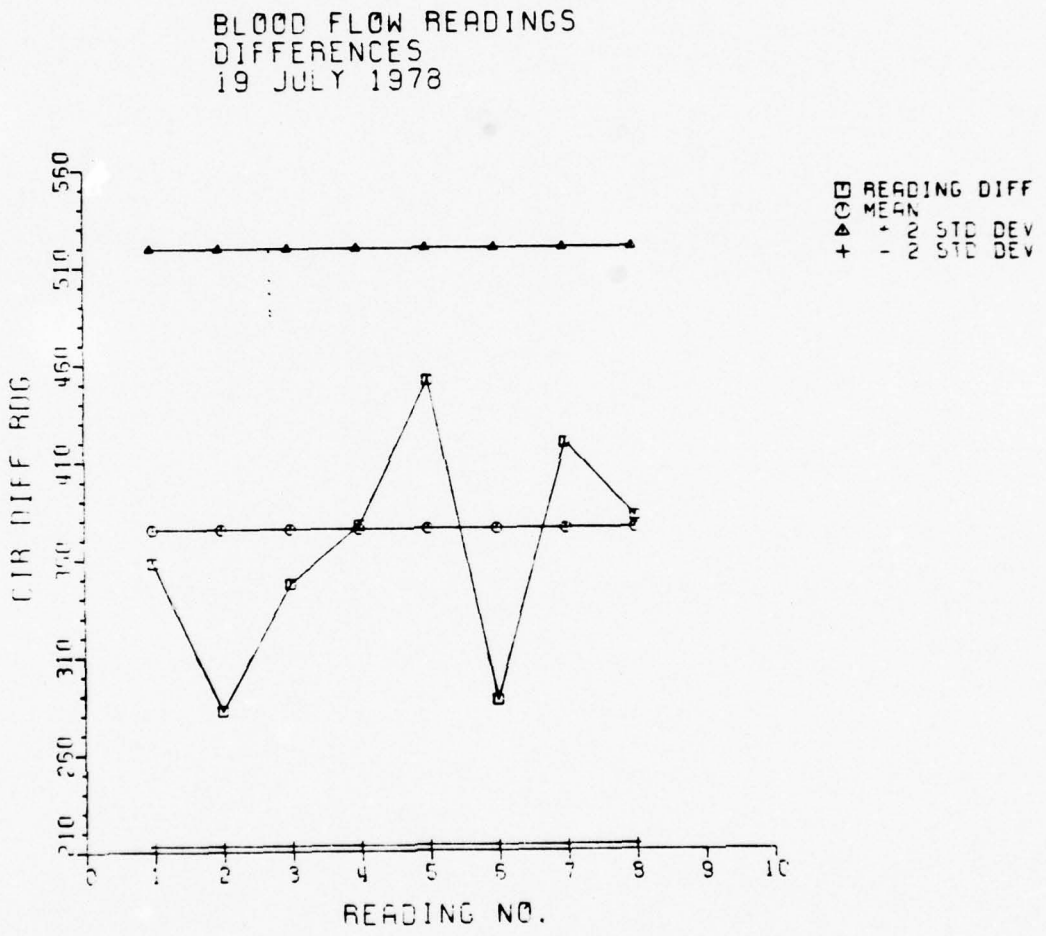


FIGURE 9 - CIRCUMFERENCE DIFFERENCE READINGS, RUN 4

The microprocessor interface to the plethysmograph took the above observations into account. The software programs were used to insure that out of sequence data (1. above) was not used. The overload condition of the digital panel meter (2. above) was noted along with the panel printer data at the microprocessor parallel input ports. The software programs then discarded this data.

The observations of 3. and 4. above indicate that the physical status of the individual whose blood flow is measured may cause discrepencies in the readings. Since a number of readings are being taken, we can therefore obtain an average and statistically delete data well away from the mean. Therefore programs were incorporated to average all data, find the standard deviation, and delete all data outside the range of two standard deviations from the mean. The resulting data was then reaveraged.

Equipment Availability

Equipment considerations were necessary from two standpoints: first, from an implementation standpoint and secondly, from the debug and testing standpoint. From a combination of both viewpoints the Intel SBC-80/10 was chosen. The MUPRO-80 microprocessor development system was used for debugging and software testing.

The MUPRO-80, configured as a single terminal system with disk drive and 40 kilobytes of memory, was used with the emulator for software debug and system checkout (references 2, 3 and 4). The system uses Block Structured Assembly Language (BSAL) as described in reference 5. Therefore, all programs were either written in BSAL or translated to BSAL for convenience and uniformity.

The Intel SBC 80/10 is expandable (reference 6) leaving room for future expansion of the plethysmograph system. This expansion capability also provides the ability to provide a memory mapped input or output to enhance the parallel and serial input/output ports. This was very necessary with a system like the Intel SBC 80/10, when it is required to handle most of its data in parallel. This is the case when interfacing with the plethysmograph.

One consideration not mentioned so far was the necessity to measure the period of the second delay in each channel. This would save the operator from having to enter this information manually. Since the Intel SBC 80/20 includes an on board timer, it was considered. However, the additional cost would not warrant this. A pair of additional twelve bit ripple counters with a 60 Hertz input was decided on. This would provide an accuracy of approximately 0.5 percent over a period of 3.5 seconds. This is well below any error caused by inconsistencies in measuring the circumference of the limb, length of the gauge, etc.

Input/Output

The manual inputs that were needed are the length of the strain gauge and the circumference of the arm for each channel. Therefore, a sixteen key keyboard plus a reset key were used as input to the microprocessor. This sixteen key keyboard with decoder and debounce circuit (74C922) was memory mapped onto the data bus.

The SBC 80/10 has six parallel 8-bit ports when used in the input mode. It was desirable to use these ports for all input data from the plethysmograph. The data from the twelve bit timers for each channel required the use of three of the parallel input ports (24 bits). Therefore

it was necessary to put the data from the panel printer plus the overload indication for each channel into the remaining three ports. Since the decimal point was not required as an input, this was deleted. This left six BCD digits (24 bits), a sign bit, a data polarity bit and two overload indicators to be included as inputs. By deleting some of the BCD bits which were unused in the plethysmograph output, these 28 bits were compressed into the remaining three input ports. The parallel port input data was then arranged as shown in tables I and II.

Indications of inputs, either keyboard or plethysmograph data, were incorporated using interrupts. Since the Intel SBC 80/10 does not distinguish between interrupting devices, it was necessary to provide this ability. The sixteen key keyboard encoder only required four bits of the eight bit data bus. The interrupt from the keyboard was then included as a fifth bit along with the keyboard input on the data bus. The microprocessor could then distinguish between interrupting devices. This will be discussed further in Chapter IV.

The inputs required all six bytes of the parallel input ports. The output to the digital panel printer required that any digit, zero through nine, could be printed in each of the six character positions. The ability to place the decimal was desirable. Control of the data polarity was also necessary. Since six binary coded decimal digits (24 bits), six decimal places (6 bits), a sign bit and a data polarity bit were necessary, a 32 bit bus to the digital panel printer was therefore required.

Port	Bit	Function
1 (E4)	7	80
	6	40
	5	20
	4	10
	3	8
	2	4
	1	2
	0	1
2 (E5)	7	8000
	6	Data Polarity
	5	Sign
	4	1000
	3	800
	2	400
	1	200
	0	100
3 (E6)	7	Overload 2
	6	Overload 1
	5	200,000
	4	100,000
	3	80,000
	2	40,000
	1	20,000
	0	10,000

Table I - Parallel Input Port 1, 2 and 3

Port	Bit	Function	Bit
4 (E8)	7	Timer Channel 1	7
	6	Timer Channel 1	6
	5	Timer Channel 1	5
	4	Timer Channel 1	4
	3	Timer Channel 1	3
	2	Timer Channel 1	2
	1	Timer Channel 1	1
	0	Timer Channel 1	0
5 (E9)	7	Timer Channel 2	11
	6	Timer Channel 2	10
	5	Timer Channel 2	9
	4	Timer Channel 2	8
	3	Timer Channel 1	11
	2	Timer Channel 1	10
	1	Timer Channel 1	9
	0	Timer Channel 1	8
6 (EA)	7	Timer Channel 2	7
	6	Timer Channel 2	6
	5	Timer Channel 2	5
	4	Timer Channel 2	4
	3	Timer Channel 2	3
	2	Timer Channel 2	2
	1	Timer Channel 2	1
	0	Timer Channel 2	0

Table II - Parallel Input Ports 4, 5 and 6

The slope computer of the plethysmograph had room for two additional cards which could be used to provide a tri-state output to the printer from the present plethysmograph multiplexed panel meter outputs. The slope computer with existing circuitry was only drawing 0.82 amperes from the five volt, three ampere power supply. Therefore, the slope computer power supply would also accommodate additional circuitry.

Space within the slope computer would not accommodate the Intel SBC 80/10 board. Therefore, the line between the existing plethysmograph and the add-on microprocessor was pretty much established (figure 9).

The SBC 80/10 uses its parallel input ports to read data from the printer bus and the timers. The remaining input and output functions of the SBC 80/10 are achieved through a memory mapped input and output. The keyboard input is memory mapped directly to the data bus. Four separate eight bit latches are used to provide a 32 bit parallel output to the printer bus. Each of these four latches are accessed through a memory mapped output. Once the data is latched onto the printer bus, a print command is given to the printer from the microprocessor.

Additional memory mapped outputs are for control functions necessary to control the plethysmograph and the functions on the B-board. These will be discussed in detail in Chapter IV. The above circuits are located on the B-board.

Operational Description

A typical sequence of operation is initiated by the microprocessor which resets the system and selects the panel meters to

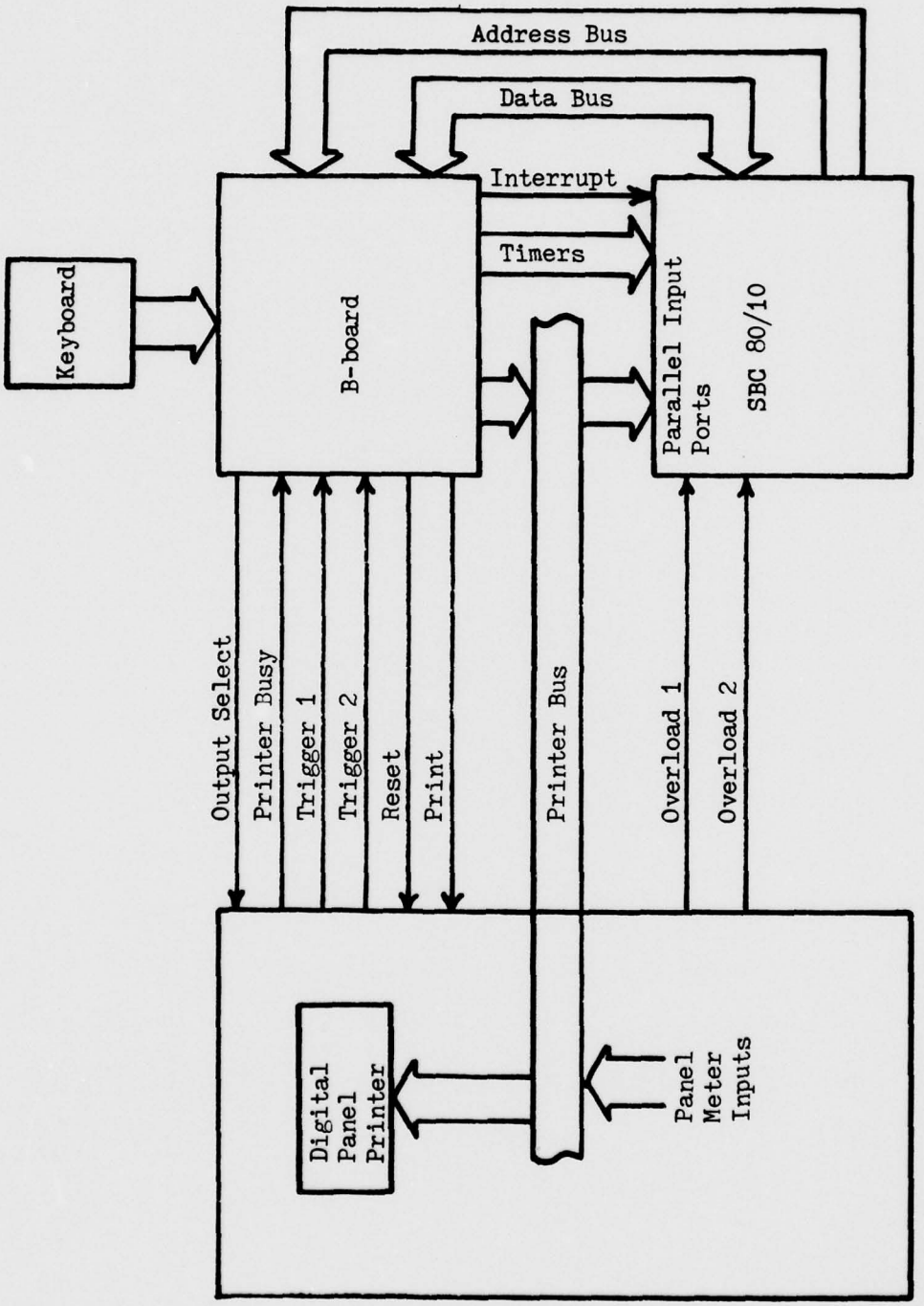


Figure 9 - Block Diagram of Microprocessor Interface

provide input to the printer bus (figure 9). A print test (all 8's) is output on the printer when the microprocessor is ready. The operator would then enter the length of the gauge and limb circumference on the keyboard for each channel which is being used. As each key is depressed, the SBC 80/10 receives an interrupt. The keyboard entry is stored in an input buffer until the enter key is depressed. The data is then stored at a specified memory location. This data is also printed on the printer as an entry check for the operator.

The operator then calibrates the system. As each calibration reading is printed on the printer, the printer busy signal initiates an interrupt. The SBC 80/10 reads each calibration reading from the printer bus through the parallel input ports. After the second of calibration reading is received, the difference is found and this difference is stored in a memory list along with other calibration difference readings. Readings which indicate an overload are discarded.

Limb circumference readings are then initiated. These are read through the parallel input ports like the calibration data. After the second reading has been received, the difference is calculated and this circumference difference is stored in a separate list.

At the end of each cuff inflation cycle, the plethysmograph prints a run count. When the microprocessor reads this run count it acknowledges the end of the cuff inflation cycle by reading the timers for each channel which presented the data. These timers measure the second delay period by using the trigger pulses from the plethysmograph.

These time periods are retained and averaged later. The timers are then reset awaiting another cuff inflation cycle.

More calibration readings are taken after the limb circumference readings are complete. These are stored along with the calibration difference readings which were obtained initially.

The operator then depresses the calculate blood flow key. The microprocessor averages all of the calibration difference readings and finds their standard deviation. Data outside two standard deviations from the mean are flagged. The same procedure is then accomplished for the circumference difference readings. A new average of unflagged data is then obtained. These averages along with the average time of the second delay period, the circumference of the limb and the length of the gauge are combined to yield the blood flow.

The microprocessor then outputs a statistical summary of all data through the panel printer. The specific format and content will be discussed in detail in Chapter VI.

After the blood flow is computed and output on the printer, the entire system is reset and memory is cleared awaiting a new set of readings.

CHAPTER IV

HARDWARE DESIGN

Slope Computer Modifications

The principal modification to the slope computer was that of providing the tri-state buffering to the panel printer and interfacing control logic between the microprocessor and the plethysmograph. The slope computer contained five 44 pin plugboards. The first four plugboards in the original system were fairly well filled. The fifth plugboard only contained two, four bit multiplexers (SN 74157). Therefore, the tri-state buffers were added to this board (figure 10). Two plugboards (card 6 and card 7) were added at the two excess edge connectors already within the slope computer to provide tri-state buffering for the remaining printer data (figures 11 and 12).

Card seven (figure 12) also included latching for the overload status of the digital panel meters. This is necessary to insure the data remains valid at the time the microprocessor reads it. There are two resets on the plethysmograph. One is to reset the run counter and another to reset the whole system sequencing. Card seven also provides the microprocessor input to these reset commands. The print commands are also ored together on this board.

The input to the slope computer from the microprocessor is accomplished using two 25-pin connectors J-4 and J-5 (tables IV and V, Appendix A). The interconnections from cards 5, 6 and 7 of the slope computer are shown in tables VI through VIII.

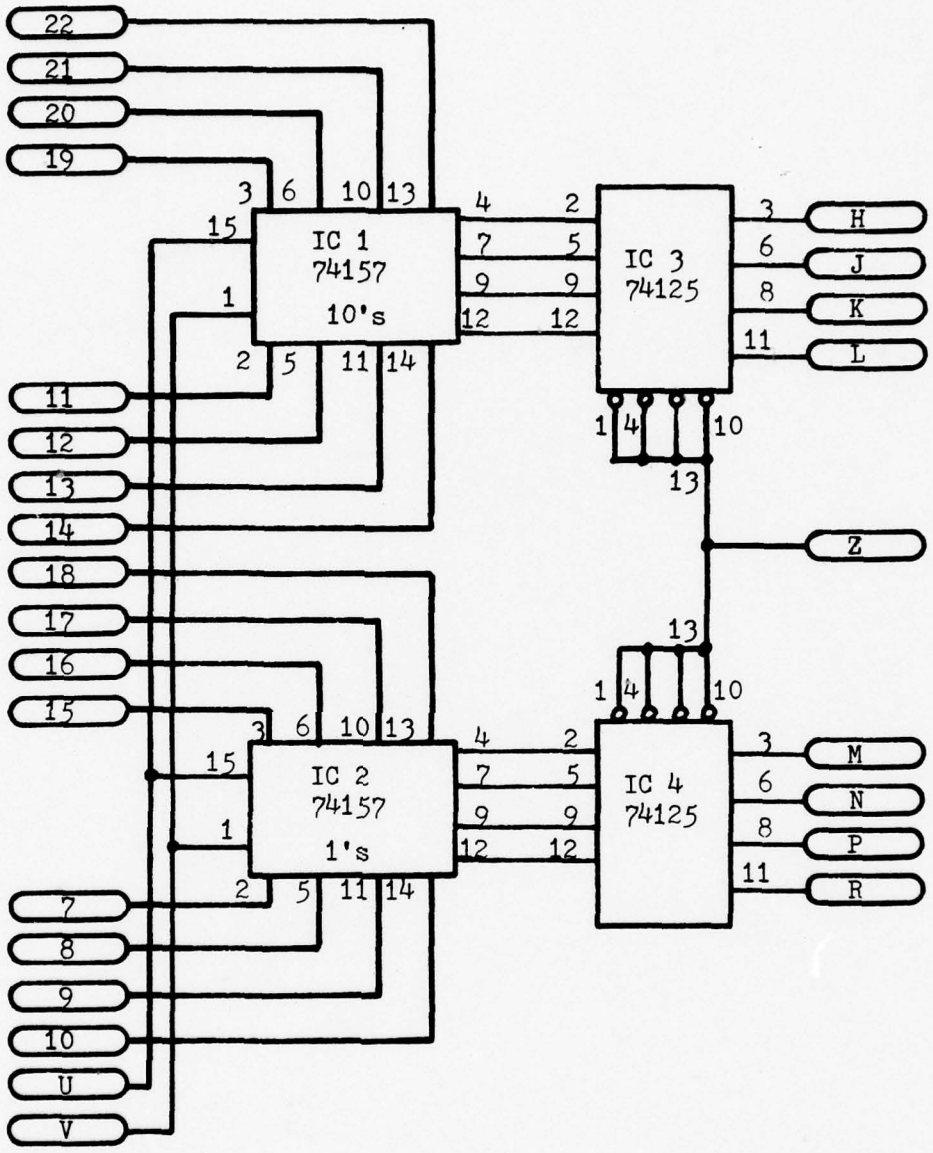


Figure 10 - Schematic of Card 5, Slope Computer

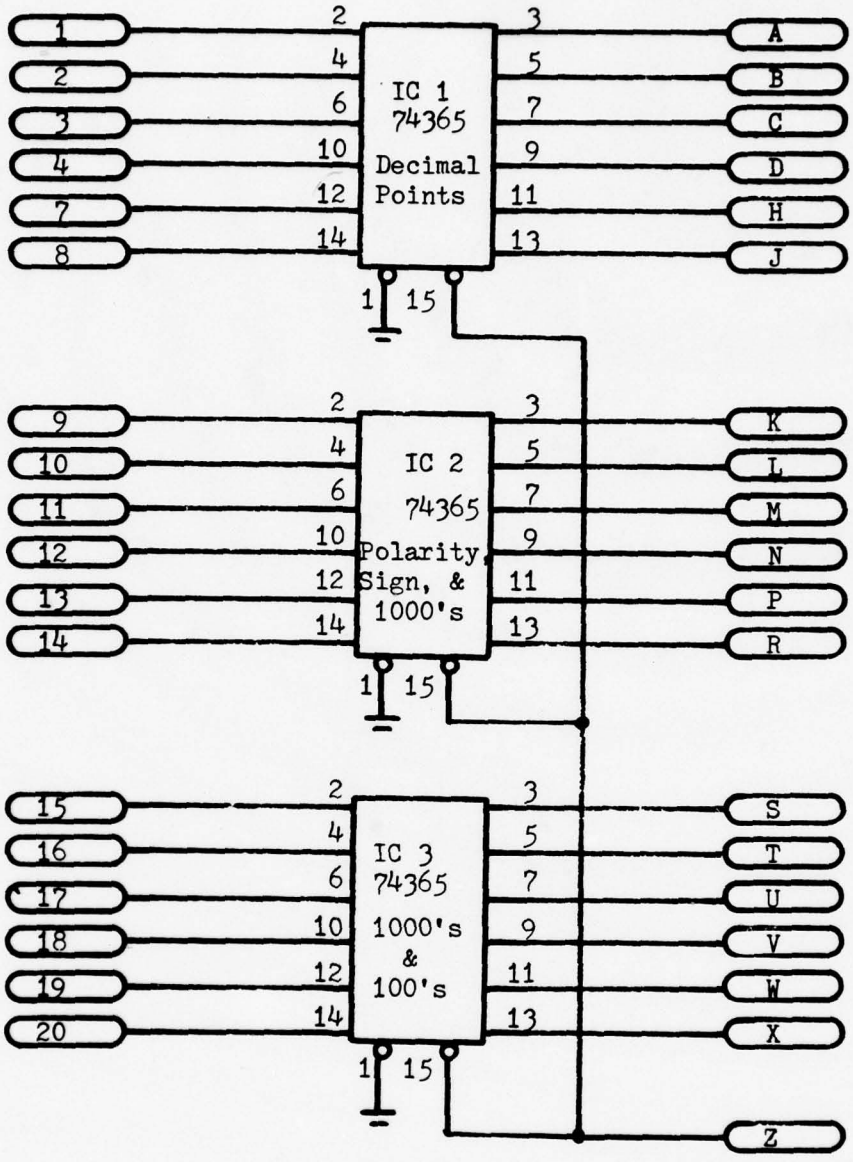


Figure 11 - Schematic of Card 6, Slope Computer

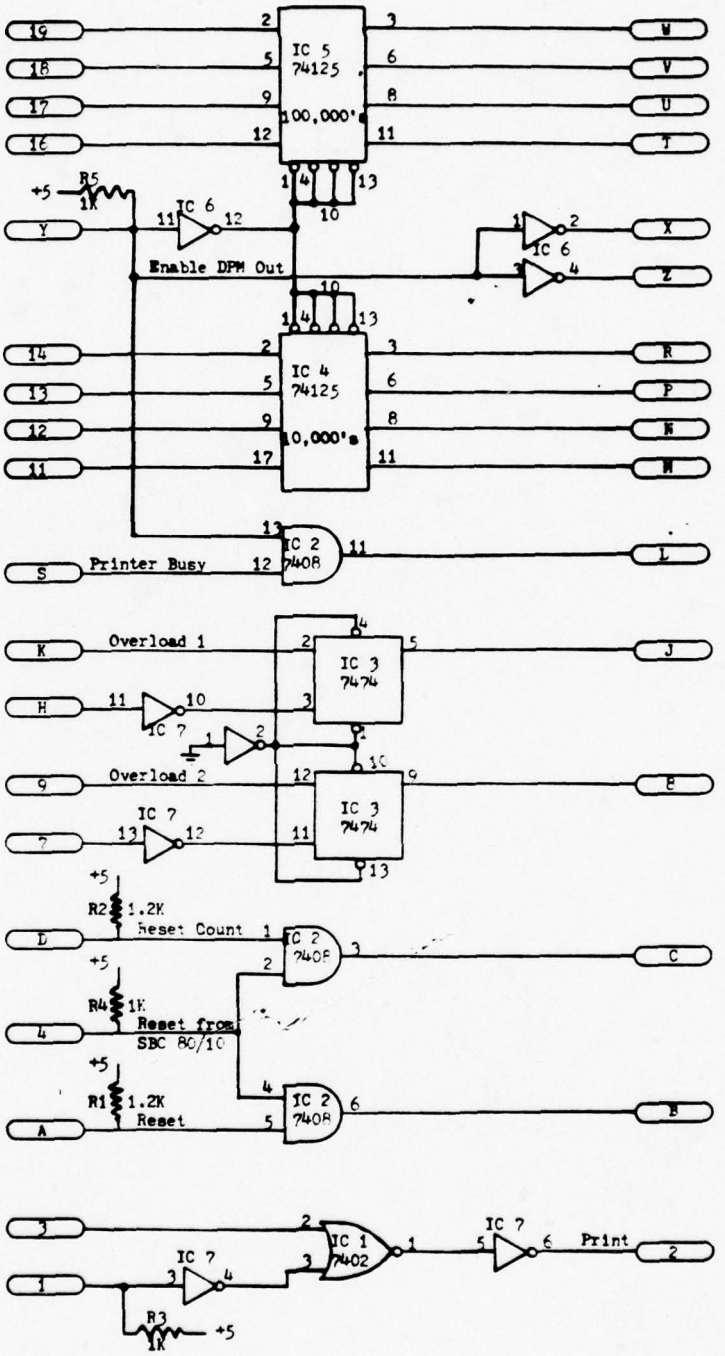


Figure 12 - Schematic of Card 7, Slope Computer

The plethysmograph modifications were designed such that disconnecting the microprocessor would not inhibit the plethysmograph from functioning as it was previously configured.

Microprocessor Interface

The SBC 80/10 is wired for operation as described in Tables 4-4, 4-13 and 4-24 of reference 6. This places the system in Mode 0 such that all of the parallel input/output ports accept input only. This wiring also disables the interrupts from the parallel and serial input/output ports. Pull-up resistors (1000 ohm) were used at each bit of the parallel input ports. The SBC 80/10 is equipped with four kilobytes of programmable read only memory (PROM) and one kilobyte of random access memory (RAM). The PROMs are addressed from 0000 hexidecimal through 0FFF hexidecimal. The RAM is addressed from 3C00 hexidecimal through 3FFF hexidecimal.

The interface between the microprocessor and the modified plethysmograph is accomplished using a plugboard the same size as the SBC 80/10. We will refer to this as the B-board (figure 13). This board consists of four major functional groups; 1) the memory mapped decoders, 2) the keyboard input section, 3) the digital panel printer output ports, and 4) the timer section. If required, room is left on the board for further memory expansion.

The P-1 and P-2 connectors are common to both the B-board and the 80/10 board. The functions are as described in reference 6. The top of the B-board connects to a 100 pin edge connector (J1). The top of the SBC 80/10 board uses two 50 pin edge connectors to the six parallel input ports. A perforated board was used to mount and

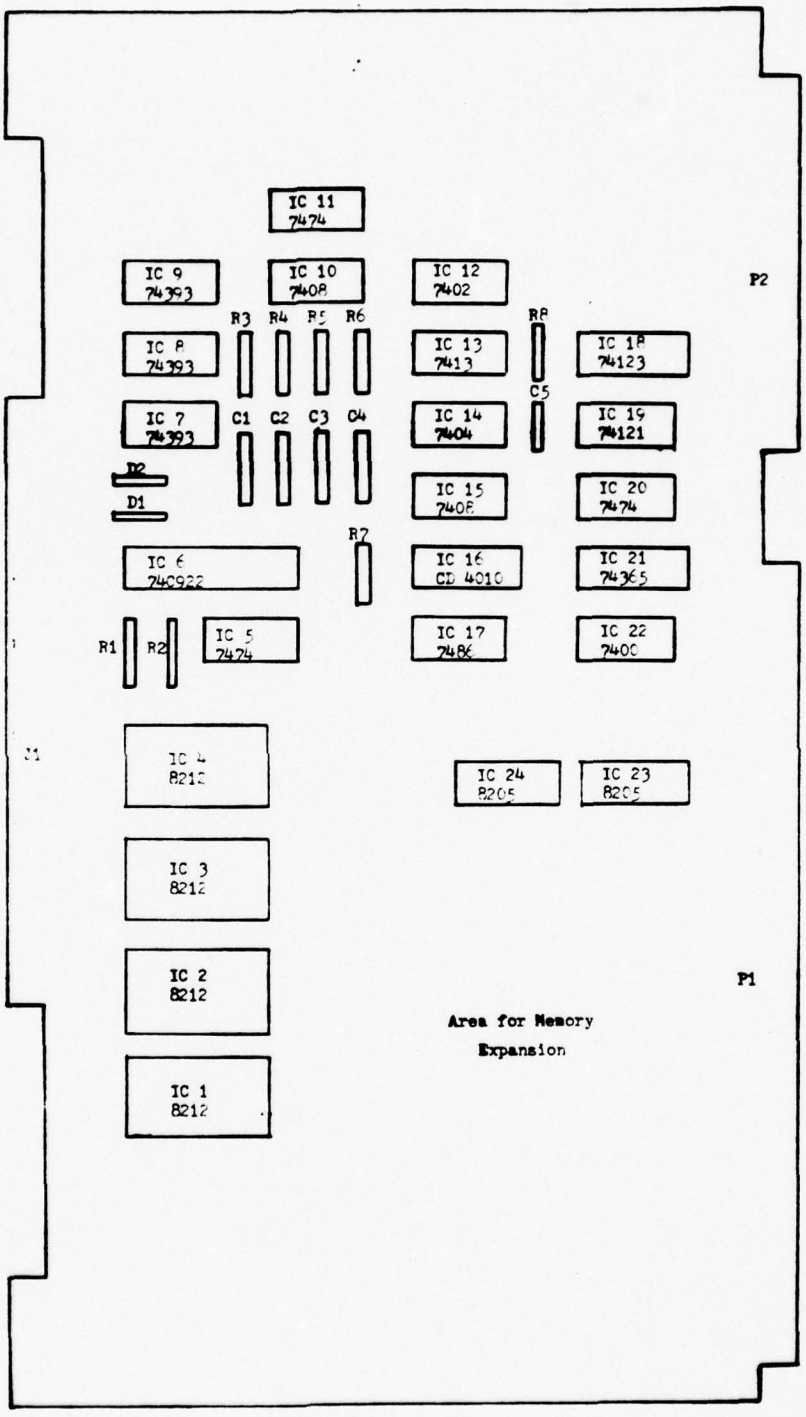


Figure 13 - Layout of B-board

interconnect these wire wrap edge connectors and also provide the base for the two 25 pin cables interfacing with the plethysmograph. The pin interconnections on this board are described in table IX, appendix A.

The memory mapped decoders (figure 14) use two, one out of eight binary decoders. They decode addresses located above the highest address of system memory (3FFF hexidecimal) and provide the designated input and output control. The decoder outputs are active low.

The first decoder provides decoding for keyboard inputs and decoding to four tri-state buffers for output to the parallel digital panel printer bus. The keyboard input signal places the keyboard data onto the data bus when activated. This is the same as reading from address location 7800 hexidecimal. The first of the decoded output signals selects an eight bit tri-state buffer to provide the decimal point, sign and data polarity output to the printer bus. The remaining three decoder outputs select each of three other buffers to output six BCD digits to the printer bus.

The second decoder is used to output control signals. The clear keyboard interrupt signal is self-explanatory. The output select signal alternately selects either the SBC 80/10 or the digital panel meters to provide output to the panel printer. The print and advance command from the memory mapped decoder initiates a print command to the digital panel printer. The command from the decoder is provided to a monstable multivibrator which widens the pulse to approximately 10 milliseconds for the printer. The signal of reset timers and DPP

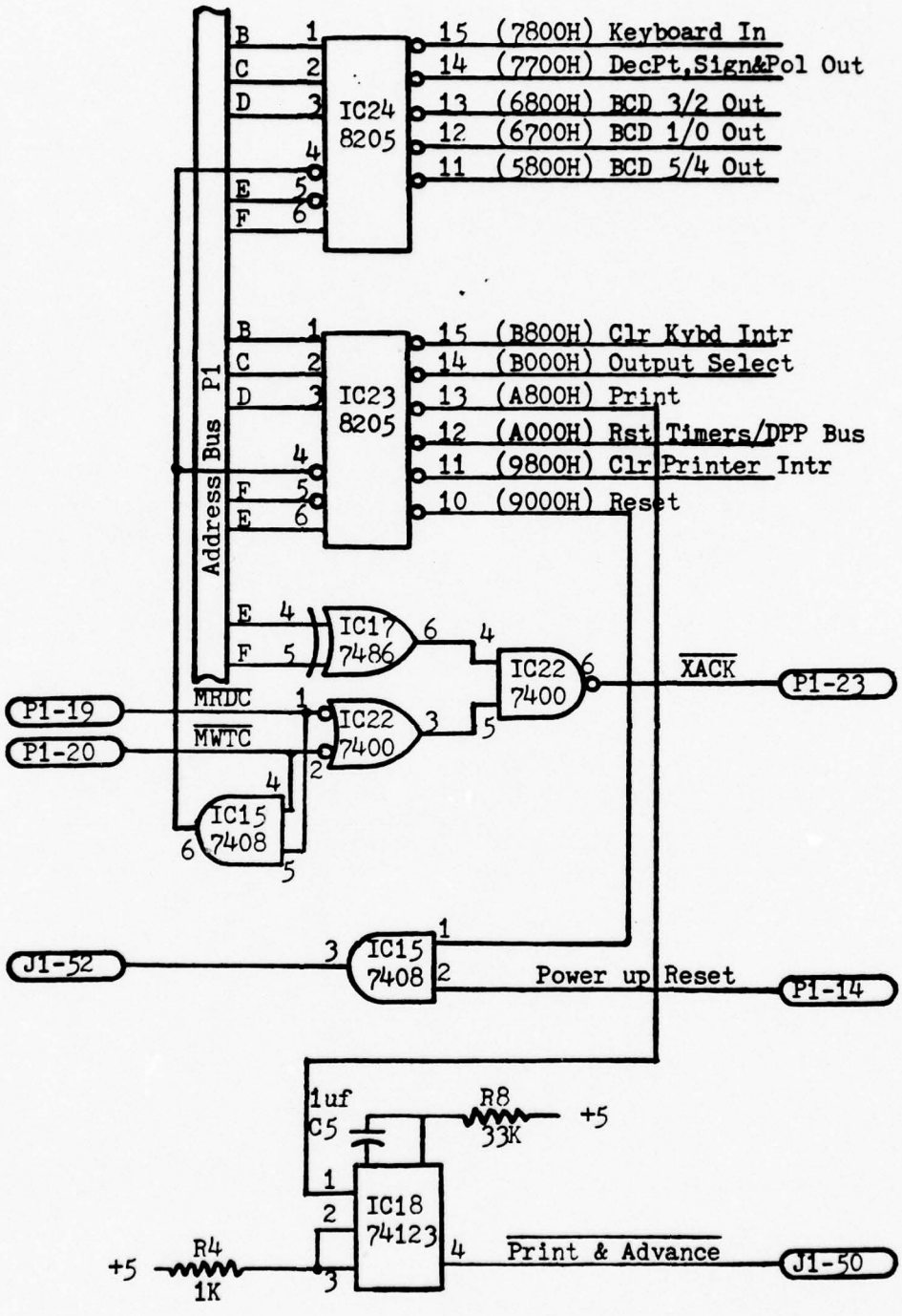


Figure 14 - Memory Mapped Decoders, B-Board

bus from the second decoder zeros the timers and selects the panel meter output onto the printer bus.

The signal of clear printer interrupt from the second decoder resets the interrupt from the printer after the data has been read. The signal of reset provides a reset command to the plethysmograph when selected or during the power up of the microprocessor.

The two highest bits of the address and memory read or write commands are combined to provide the transfer acknowledge (XACK) to the ready input of the clock generator and driver of the central processing unit.[9]

The keyboard input section (figure 15) is designed around the 16 key encoder[7] (74C922). The hex buffer (CD 4010) is used for the CMOS to TTL interface. The tri-state buffer (SN 74365) then can be selected to input the keyboard data onto the data bus. This is accomplished through the memory mapped signal of keyboard input.

Capacitors on the encoder are selected for a 600 hertz oscillation frequency and a 20 millisecond debounce period. These values proved reliable for the pushbutton switches which were used.

When a key is pressed, the data available line provides an output which sets the D-type flip-flop and causes an interrupt. The microprocessor can then read the keyboard input and determine if the keyboard was the interrupting device (Line 13 of IC21, figure 15, goes high for a keyboard interrupt). If it was the interrupting device, the data is kept. At the completion of the cycle the interrupt handler resets the interrupt.

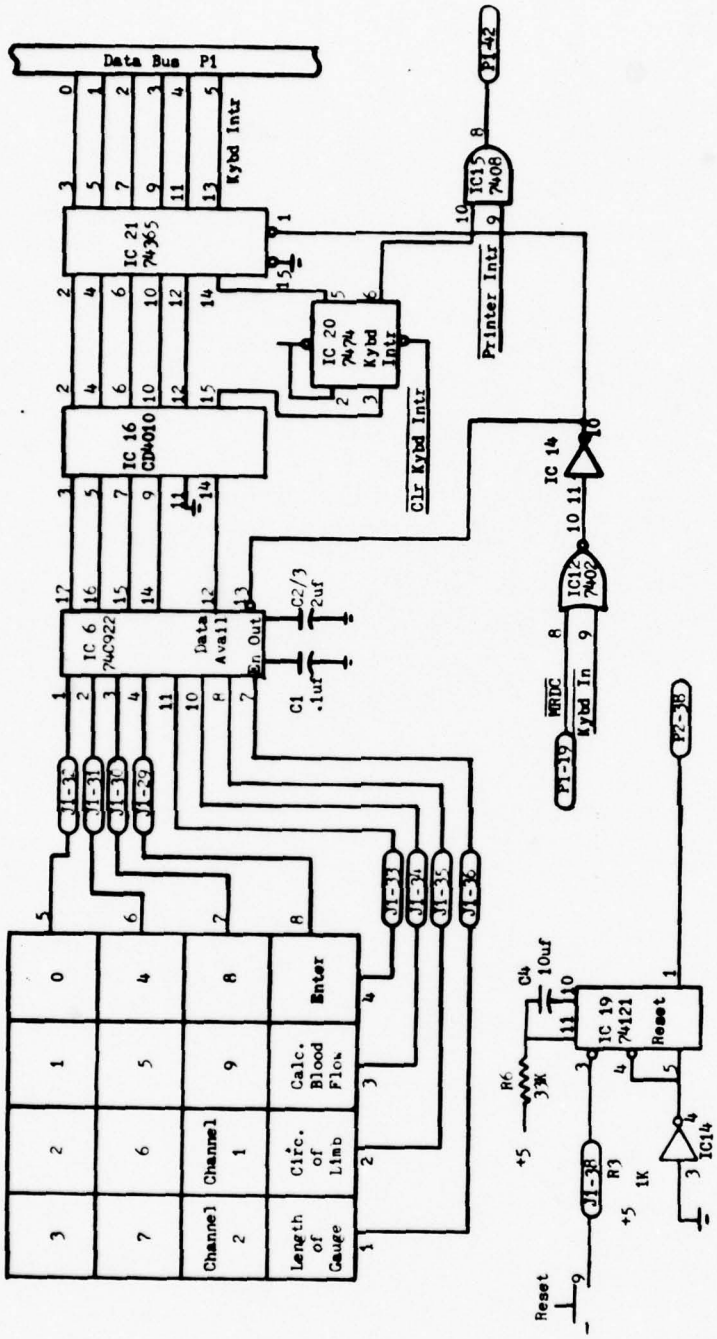


Figure 15 - Keyboard Input Section, B-board

A separate circuit (figure 15) is provided for the system reset. This provides a 230 millisecond pulse directly to the SBC 80/10 reset input.

The digital panel printer output ports (figure 16) use four, eight bit input/output ports (8212). When the SBC 80/10 printer output is selected, these ports receive data as a memory mapped output. The data is placed directly onto the printer bus. Once all the data is loaded, a print command is sent to the printer.

As a part of this circuitry the digital panel meter (DPM) output enable signal from the memory mapped decoder alternately selects either 80/10 or the panel meters for output to the printer. When the panel meters are providing data to the printer, the printer busy signal causes an interrupt signal to be sent to the microprocessor. The microprocessor then reads the data from the panel meters.

Two timers are used (figure 17) to measure the second delay period in each of the two plethysmograph channels. The timers are started and stopped by the trigger pulses at the beginning and end of the second delay period. Once the trigger pulse is received, the 60 hertz input is fed through the Schmidt trigger NAND gate directly to the counters. A second pulse shuts the timers off. The counters are twelve bit long and therefore can measure up to 68 seconds without overflow. This is well above the nominal 3.5 seconds usually used. The outputs of the counters are fed to parallel input ports 4, 5 and 6 of the SBC 80/10. The software then decides when to read the timers.

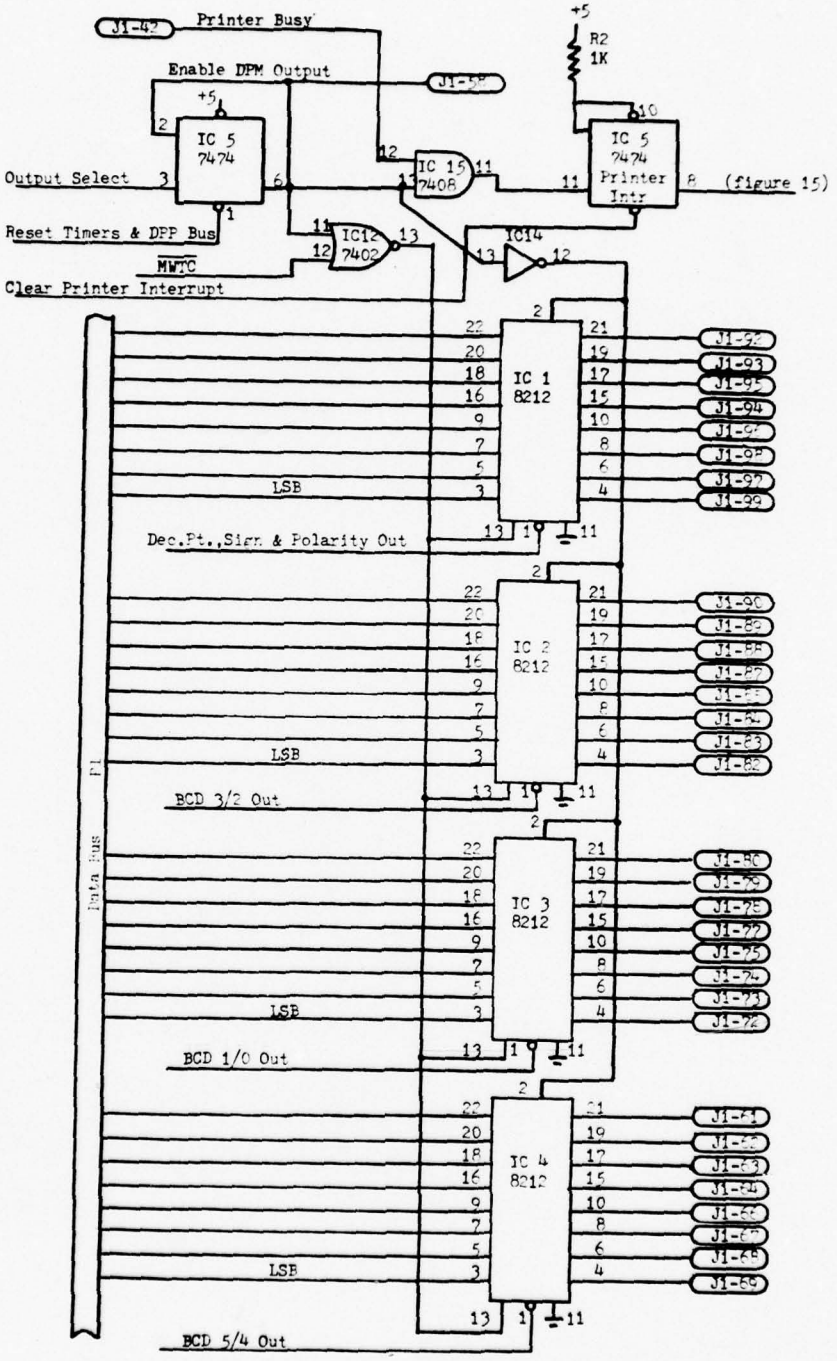


Figure 16 - Digital Panel Printer Output Ports, B-board

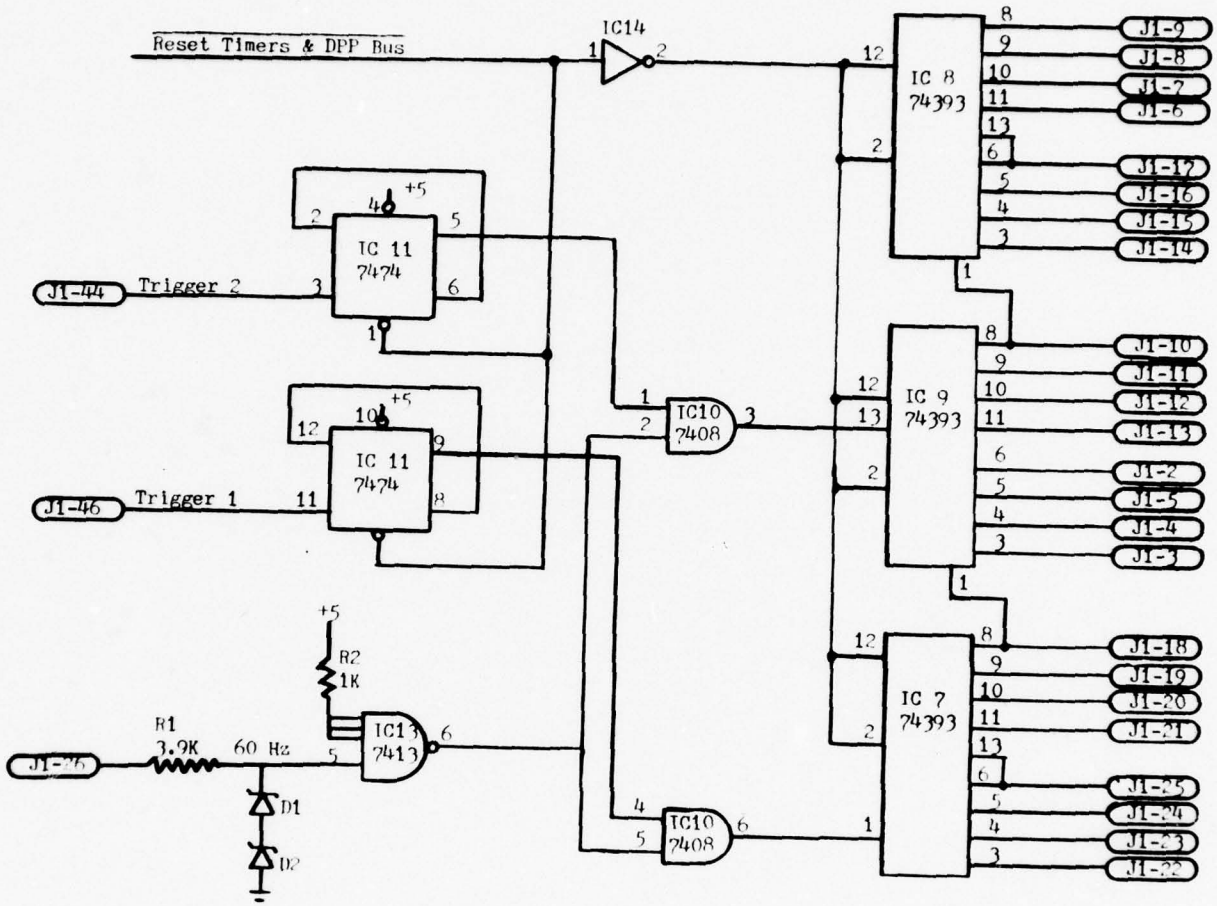


Figure 17 - Timer Section, B-board

The hardware comprising the system is now complete. The principle of the design was to provide a good balance between hardware and software without making either one overly complex or cumbersome in interfacing with the plethysmograph.

CHAPTER V

SOFTWARE DESIGN

The SBC 80/10 system provides one kilobyte of random access memory. This was a basic consideration in the design of the software. As mentioned in Chapter III, it was desirable to sort the data after it was gathered. Therefore it was necessary to store all readings prior to analysis and then calculate the blood flow. Therefore stored data must be kept as compact as possible to conserve memory.

Since it is also desirable to discard data outside the range of two standard deviations from the mean, calculations almost dictated that a floating point arithmetic be included to compute the standard deviation. The fact that there were a fair number of variables comprising the actual blood flow calculation further indicated that a floating point arithmetic would provide a more versatile software package.

The above requirements are contradictory. Storage of data in floating point format is not the most memory efficient, especially when input data can be given integer values within a very specific range. This, therefore, dictated a third requirement that the data be easily manipulated between integer and floating point format.

Data Storage

The majority of the input data was either calibration readings or blood flow readings. Calibration readings are obtained from either a 0.2 percent or a 1.0 percent calibration. The calibration factor is then used in the blood flow calculation, as described in Chapter II.

Since all calibration readings need to be compared, regardless of the calibration factor, it was decided to normalize all calibration readings to a 1.0 percent factor. This required multiplying each 0.2 percent reading difference by five.

The maximum range of the panel meters and therefore the data was between -1999 and +1999. Therefore, at worst case, the differences between two readings would be 3998. This is true for the blood flow reading only. A calibration reading of 0.2 percent might be multiplied by five, giving a maximum value of 19,990. Calibration readings would then require additional storage area. To best handle the data, it was decided to leave it in binary coded decimal (BCD) format.

The storage format (figure 18) was set up with three bytes for calibration difference data and two bytes for blood flow difference data. Neither of the values of differences would occupy the most significant bit location. Therefore, this bit location was then established as a flag for data to be discarded from the final blood flow computation. Data is flagged if it is outside two standard deviations from the mean. Data with negative differences or which overrange are not stored. They are meaningless and therefore discarded immediately. The head of each block of data contains the number of difference readings in the list. The total size of the blood flow difference lists are 75 readings per channel. The total size of the calibration difference list are 33 readings per channel. This allows more than enough readings for an accurate sample and still leaves room for data manipulation and other storage.

Calibration Differences

F	10^4
10^3	10^2
10^1	10^0

Range:
0 to +19,990

Circumference Differences

F 10^3	10^2
10^1	10^0

Range:
0 to +3998

F - Flag indicates data outside 2 standard deviations from the mean.

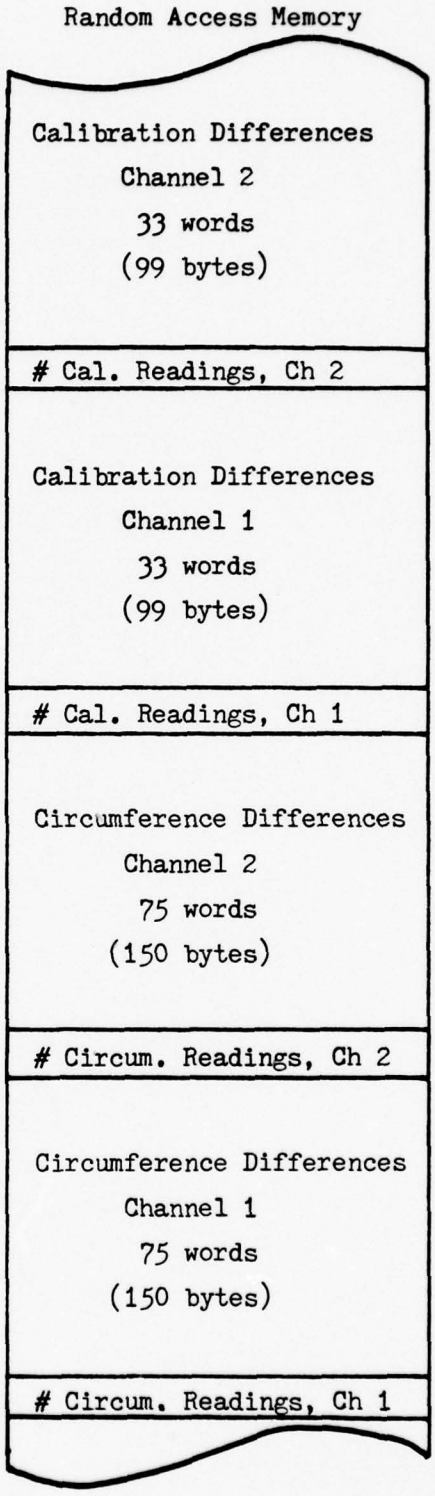


Figure 18 - Difference Data Storage Format

Other data necessary for the computation of blood flow are singular in quantity for each channel. Therefore those data are converted to floating point format immediately upon entry and stored awaiting the final blood flow calculations.

The timer information is entered in a 12-bit binary format. This binary data is converted immediately to floating point format and these times are summed for each channel. An average is later computed for use in the final calculation of blood flow.

Interrupt Handler

The main program (Appendix B) provides an initialization of the system (figure 19). A 20 second turn on delay allows time for powering the rest of the plethysmograph system. After the turn on delay of 20 seconds, memory is cleared and all of the system is reset. A print test then notifies the operator that the microprocessor is ready. The microprocessor then waits on an interrupt initiated input.

At the time of an interrupt the microprocessor first checks for a keyboard input (figure 20). If the input is from the keyboard, then it checks to see if it is an enter command. If it is an enter command and there is data in the input buffer, the input buffer is stored. The storage location depends on whether it is the length of gauge or the limb circumference and also the channel number. If it is not an enter and not a calculate blood flow command, the data is stored in the input buffer until the enter command is received.

A calculate blood flow entry calculates the blood flow and provides a statistical summary along with the blood flow reading as output to the printer. Memory is then cleared and the system reset prior to returning to the main program.

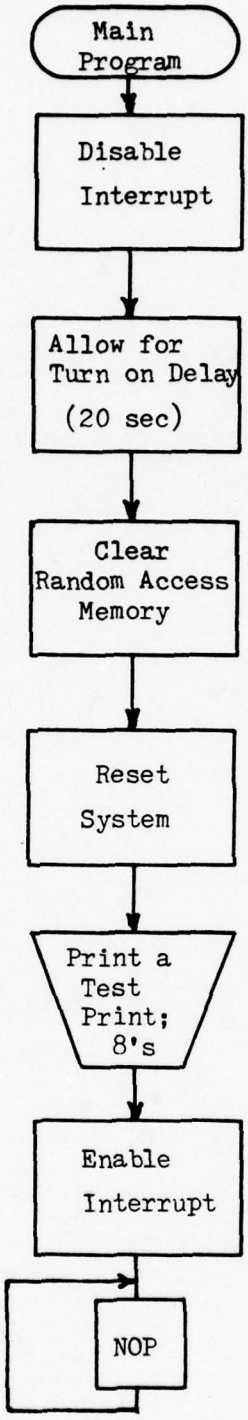


Figure 19 - Main Program Flow Chart

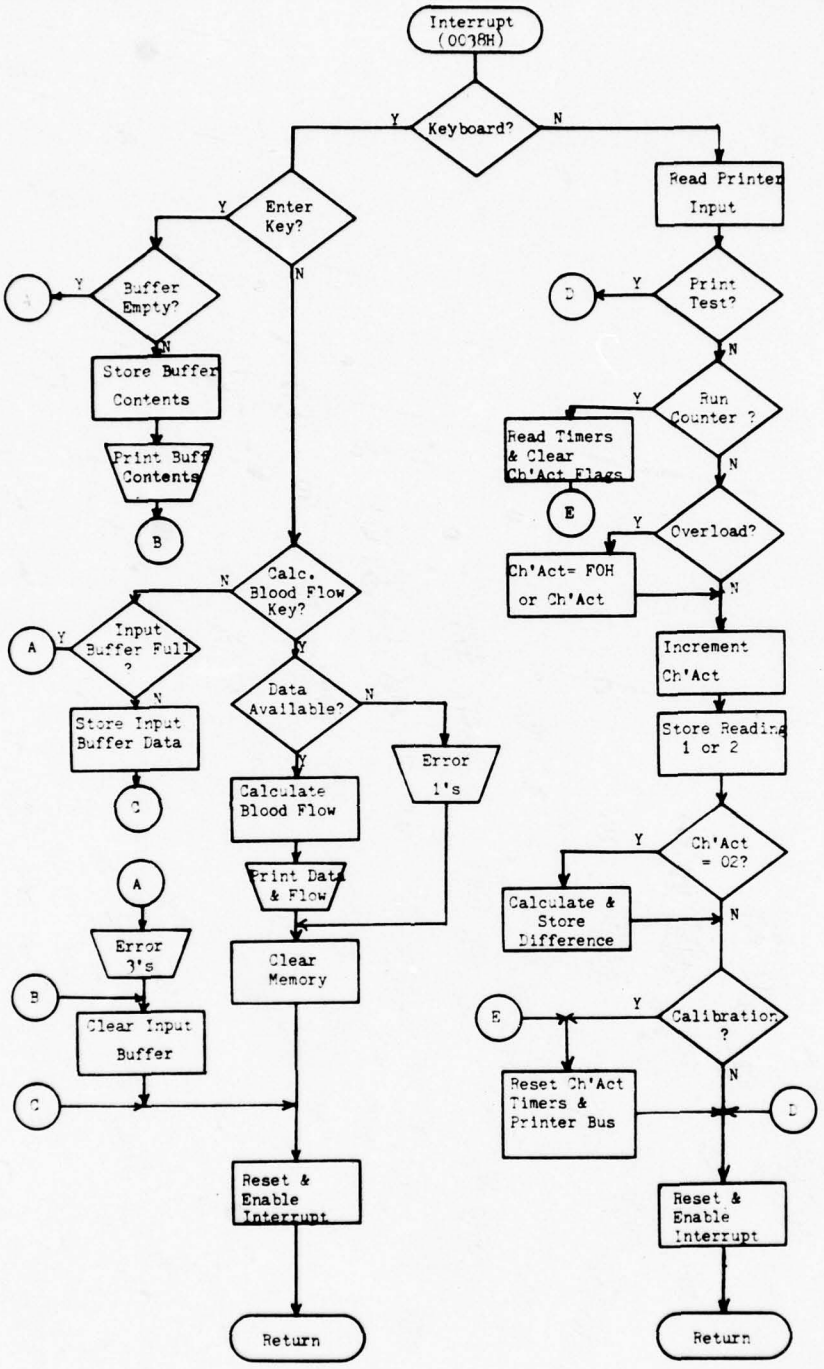


Figure 20 - Interrupt Handler Flow Chart

A panel printer interrupt causes the data to be read from the panel printer. The program then checks for a print test on the printer (all 8s) provided by pushbutton control on the plethysmograph and returns appropriately for this condition.

Along with indicating the number of inflation cycles, run count input shows that the cuff inflation cycle is complete. Therefore the timers are read at this time. Channel Active flags are used to keep track of the input data. The most significant bits are ones if either the first or second readings show an overload from the panel meters. The lease significant bits keep track of the number of entries for each channel. Once the second entry is input, valid data is stored. If Channel Active flag shows that is the second calibration reading, the Channel Active flag is cleared immediately. Otherwise the microprocessor waits for the run counter input. The Channel Active flag then indicates which of the timers to read.

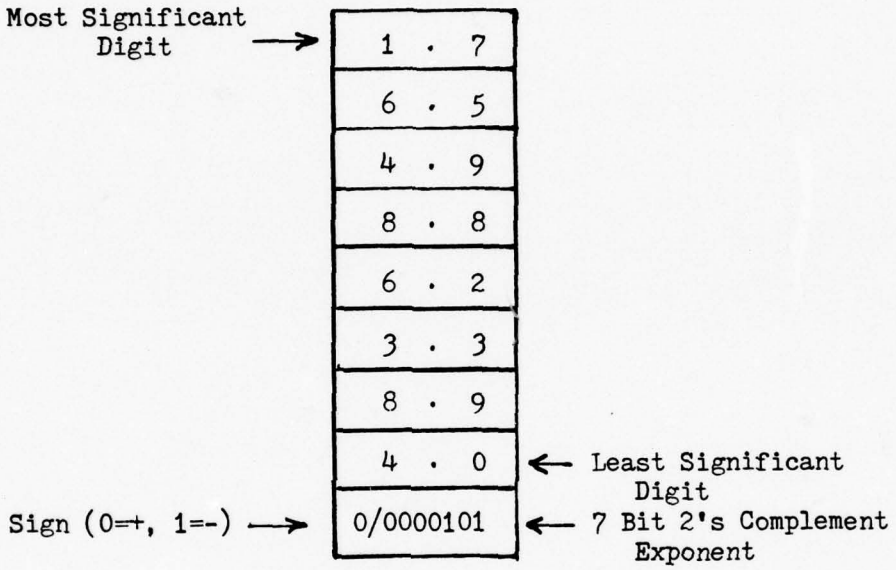
Floating Point Programs

Data manipulation to find the standard deviation of a series of readings requires full accuracy in finding the sum of the squares. The routines needed to be able to find the sum of the squares of 33 balance readings. At worst case, this would be approximately on the order of 10^{10}. References 10 and 11 offered little promise for such routines to handle binary coded decimal numbers. They did provide other routines or portions of routines to aid in the design of a set of floating points programs. These are noted in the comments of the program listings in the appendices.

After consideration of the problem, a nine byte floating point number was formatted (figure 21). This number provides eight bytes for the mantissa and one byte for characteristics. The lowest byte of the number in memory provides the sign and a seven bit signed exponent. This provides a 16 digit computational capability. An exponent of up to $\pm$ 63 might be obtained. Although this accuracy and range is more than needed for the plethysmograph, it was felt that these routines would offer an excellent package for general use with the MUPRO system in the future.

Routines were developed for signed addition, multiplication, division and square root. Subtraction is accomplished by changing the sign of the subtrahend. The multiplication routine relied heavily upon a fixed point BCD multiplication program in reference 11.

Division by zero or finding the square root of a negative number provide error outputs. Division is accomplished by successive subtractions of the divisor times powers of ten from the dividend. The square root is found by successive averaging of the divisor and quotient of the original number. The first divisor or approximation of the root is obtained by dividing the original number's exponent by two. Normally, a result of 14 digit accuracy was obtained in less than seven iterations. Since the square root was only used to find the standard deviation of the difference readings from the plethysmograph, this routine is only checked for six digit accuracy. This was accurate for the entire range of possible difference readings of 0 through 19,990.



Number Represented Above

+ 17654.98862338940

Figure 21 - Floating Point Number Format

Utility Programs

A wide range of special purpose programs were employed to aide in the handling of data within the system. These were principally data formatting for either storage, calculations or output. Other routines such as the standard deviation and mean calculation programs were used to analyze data. Other routines, such as the comparison routine, flagged undesirable data.

Annotation as to the specific function and the required input parameters in each program are contained in the specific program comments in the appendix. Therefore, the details will not be restated.

CHAPTER VI

OPERATION OF THE SYSTEM

The procedure for operation of the blood flow plethysmograph with a microprocessor is described below. This is meant to provide the operator with information in addition to that of reference 1 which is required for measuring blood flow with the automated plethysmograph.

First turn on all components of the plethysmograph. After approximately 20 seconds the microprocessor will output a print test. Allow for normal warm-up, make all gain and balance adjustments and perform operations as described in reference 1. When the system is ready, reset the microprocessor. This resets the entire system and again outputs a print test.

Next, enter the gauge length and the limb circumference in millimeters. Enter in the following order: 1) channel number, 2) gauge length or limb circumference, and 3) the measurement in millimeters. Then press enter key. The printer then displays the data that was entered. If an incorrect data was entered, reenter the correct data. Insure that all measurements are entered for the channels being used.

Calibrate the system as described in the operator's manual. Once a series of stable readings are obtained, the blood flow measurement may begin. Again calibrate the system at the end of the blood flow measurements. (Note: no more than 33 calibration readings or 75 blood flow readings may be taken per channel).

To obtain the blood flow reading, press the calculate blood flow key. The microprocessor now automatically calculates the blood flow. All difference data outside the range of two standard deviations from the mean are discarded. If you wish to retain this data and include it into the calculation, press the number '2' before the calculate blood flow key. After the blood flow is output, the microprocessor then automatically resets itself and the plethysmograph and awaits new data.

A sample of the printer output sequence is shown in figure 22. The first digit of each output is retained for channel identification. The second digit of each output is a sequential coding to identify second delay time data (2), length of gauge (3) limb circumference (4), calibration data (5), circumference reading data (6) and the blood flow (7). The standard deviations are not included in the printout when the '2' key is depressed prior to the calculate blood flow key. Any output greater than +9998 is displayed as 9999. Any output less than 0.0001 is displayed as all zeros.

Error indications are provided to the operator (table III). If there are less than two circumference readings, an error message is provided. This insures an adequate sample size and insures that a standard deviation may be computed.

Input errors are provided for an overflow of the input buffer. If the input data is not entered correctly from the keyboard an error is also output. In either case, reenter the data. The remaining error outputs are provided as a precaution only. Normal operation of the system should not require their use.

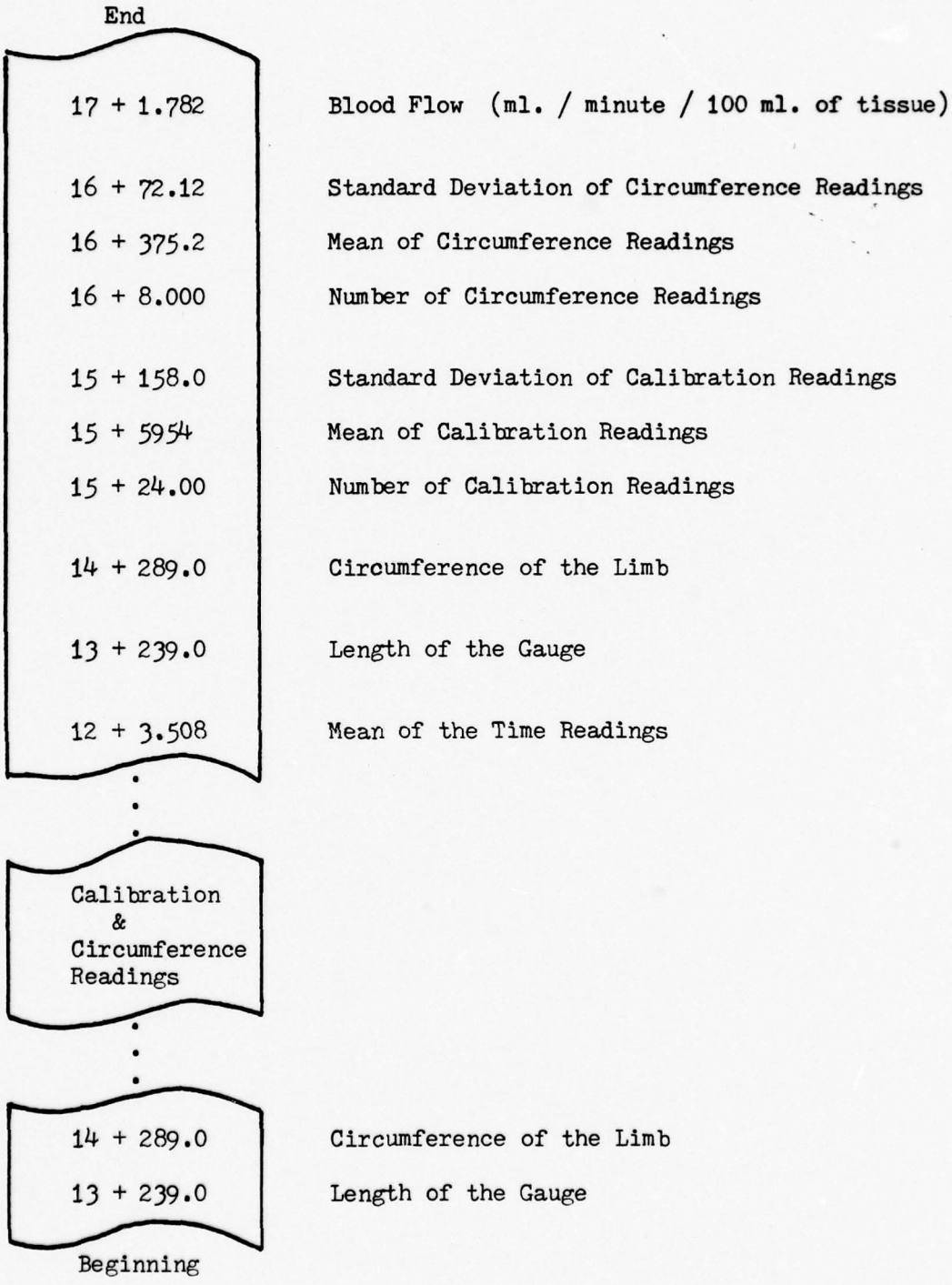


Figure 22 - Output Sequence

OUTPUT	INDICATION
1.1 + 1.1.1.1	No Data Available for Output
3.3 + 3.3.3.3	Input Error
4.4 + 4.4.4.4	Overflow in Conversion Floating Point to Integer
6.6 + 6.6.6.6	Negative Square Root
7.7 + 7.7.7.7	Divide by Zero
8.8 + 8.8.8.8	Print Test

Table III - Error Messages

CHAPTER VII

SUMMARY

Adoption of a microprocessor interface to the plethysmograph adds the capability of quickly assimilating the data, obtaining the remaining parameters as necessary and providing the operator with immediate blood flow measurements.

The interface as currently designed requires only a minimum modification to the dual plethysmograph. The Intel SBC 80/10 single board computer is integrated with the system. Peripherals to this board were added to measure time intervals, to provide keyboard inputs, and to output data to the parallel digital panel printer bus.

Introducing a microprocessor to the plethysmograph provides rapid processing of the information, reduces the chance of operator error, and allows for error checking. This provides the operator/technician with a more usable instrument which can produce a real-time blood flow measurement for the physician's use. This modification will allow a more complete evaluation of the blood flow plethysmograph. Blood flow measurements can now be quickly repeated for correlation with previous measurements.

Although the initial system was developed with the concept of minimizing changes to the dual plethysmograph circuitry, follow-on systems would be entirely microprocessor controlled. This would include replacing the entire slope computer and timing circuitry. The calibration logic and the manual gain and balance could also be entirely microprocessor controlled. An interactive terminal could be made available and most system parameters could be set through this terminal.

REFERENCES

1. Institute of Environmental Stress, User's Manual for Blood Flow Plethysmograph, Santa Barbara, California, revised March 1978, pp. 1-28, schematics and component spcifications.

2. MUPRO-80 Microcomputer and In-Circuit Emulator, Hardware Reference Manual, MUPRO Incorporated, 1976.

3. Multi-User Task Executive, MUPRO Incorporated, November 1977.

4. 2708 Prom Programmer, MUPRO Incorporated, 1977.

5. Block Structured Assembly Language, MUPRO Incorporated, November 1977.

6. SBC 80/10 Single Board Computer Hardware Reference Manual, Intel Corporation, 1976.

7. CMOS Databook, National Semiconductor, 1977.

8. The TTL Data Book for Design Engineers, Texas Instruments Inc., Second Edition, 1976.

9. Intel 8080 Microcomputer Systems User's Manual, Intel Corporation, September 1975.

10. Lance A. Leventhal, 8080A/8085 Assembly Language Programming, Osborne and Associates, 1978.

11. Intel User's Library Program Manual, Volume I, Section 5, April 1977.

APPENDIX A

INTERCONNECTION TABLES

1	Ground
14	Decimal Point .0
2	Decimal Point .000
15	Decimal Point .00
3	Decimal Point .0000
16	Decimal Point .000000
4	Decimal Point .00000
17	Data Polarity
5	Sign
18	8000
6	4000
19	2000
7	1000
20	800
8	400
21	200
9	100
22	80
10	40
23	20
11	10
24	8
12	4
25	2
13	1

Table IV - J-4 Pin Connections

Pin	Function
1	Ground
14	10,000
2	20,000
15	40,000
3	80,000
16	100,000
4	200,000
17	400,000
5	800,000
18	Enable DPM Output
6	Overload 1
19	Overload 2
7	Reset
20	Ground
8	Print and Advance
21	Ground
9	Unused
22	Ground
10	Trigger 1
23	Ground
11	Trigger 2
24	Ground
12	Printer Busy
25	Ground
13	

Table V - J-5 Pin Connections

PIN	Connector	PIN	Connector	PIN	Function
1					
2					
3					
4					
5					+5 volts
6					Ground
7	DPM 1	F			80
8		6			40
9		7			20
10		H			10
11		L			8
12		12			4
13		P			2
14		M			1
15	DPM 2	F			80
16		6			40
17		7			20
18		H			10
19		L			8
20		12			4
21		P			2
22		M			1
A					
B					
C					
D					
E					
F			J4	1	Ground
H	DPP-C1	B8		22	80
J		A13		10	40
K		A9		23	20
L	DPP-C2	B14		11	10
M	DPP-C1	B9		24	8
N		B15		12	4
P		A10		25	2
R	DPP-C2	B15		13	1
S					
T					
U					
V					
W					
X					
Y					
Z	Card 7	Z			Enable DPM Output

Table VI - Slope Computer Interconnection, Card 5

PIN	Connector	PIN	Connector	PIN	Function
1	Card 3	U			Decimal Pt. .0
2	Card 3	U			Decimal Pt. .000
3	Card 3	V			Decimal Pt. .00
4	Card 3	V			Decimal Pt. .0000
5					+5 volts
6					Ground
7	Card 4	19			Decimal Pt. .000000
8	Card 4	1			Decimal Pt. .00000
9					
10					
11	Card 3	20			Data Polarity
12	Card 3	10			Sign
13	Card 4	H			8000
14	Card 4	J			4000
15	Card 4	K			2000
16	Card 4	L			1000
17	Card 4	M			800
18	Card 4	N			400
19	Card 4	P			200
20	Card 4	R			100
21					
22					
A	DPP-C1	A6	J4	14	Decimal Pt. .0
B	DPP-C1	B6		2	Decimal Pt. .000
C	DPP-C1	A7		15	Decimal Pt. .00
D	DPP-C1	A1		3	Decimal Pt. .0000
E					
F					
H	DPP-C1	B3		16	Decimal Pt. .000000
J	DPP-C1	B2		4	Decimal Pt. .00000
K					
L					
M	DPP-C1	B11		17	Data Polarity
N	DPP-C1	A5		5	Sign
P	DPP-C2	A1		18	8000
R	DPP-C1	A15		6	4000
S	DPP-C2	B10		19	2000
T	DPP-C2	A13		7	1000
U	DPP-C2	B7		20	800
V	DPP-C1	A14		8	400
W	DPP-C2	B11		21	200
X	DPP-C2	A15		9	100
Y					
Z	Card 7	X			$\overline{\text{Enable DPM Output}}$

Table VII - Slope Computer Interconnection, Card 6

PIN	Connector	PIN	Connector	PIN	Function
1			J5	8	Print and Advance
2	DPP-C1	B14			Print and Advance
3	Card 3	9			Print and Advance
4			J5	7	Reset (80/10)
5					+5 volts
6					Ground
7	DPM 2	K			DPM 2 Status
8			J5	19	Overload 2
9	DPM 2	5			Overload 2
10					
11	Card 3	11			80,000
12		12			40,000
13		13			20,000
14		14			10,000
15					
16		16			800,000
17		17			400,000
18		18			200,000
19		19			100,000
20					
21					
22					
A	Switch				Reset
B	Card 3	Y	Card 1 & 2	21	Reset
C	Card 3	S			Reset Count
D	Switch				Reset Count
E					+5 volts
F			J5	1	Ground
H	DPM 1	K			DPM 1 Status
J			J5	6	Overload 1
K	DPM 1	5			Overload 1
L	Card 3	7			Printer Busy
M	DPP-C2	B2	J5	3	80,000
N	DPP-C1	B10		15	40,000
P	DPP-C2	B8		2	20,000
R	DPP-C1	A12		14	10,000
S	DPP-C2	B12			Printer Busy
T	DPP-C2	B1		5	800,000
U	DPP-C1	A8		17	400,000
V	DPP-C2	A8		4	200,000
W	DPP-C1	A11		16	100,000
X	Card 6	Z			Enable DPM Output
Y			J5	18	Enable DPM Output
Z	Card 5	Z			Enable DPM Output

Table VIII - Slope Computer Interconnection, Card 7

B-board PIN	SBC 80/10 Connector	PIN	Function
1	J-2	1 & all even	Ground
2		3	Timer 1, Bit 11
3		5	Timer 1, Bit 8
4		7	Timer 1, Bit 9
5		9	Timer 1, Bit 10
6		11	Timer 2, Bit 8
7		13	Timer 2, Bit 9
8		15	Timer 2, Bit 10
9		17	Timer 2, Bit 11
10		19	Timer 2, Bit 3
11		21	Timer 2, Bit 2
12		23	Timer 2, Bit 1
13		25	Timer 2, Bit 0
14		27	Timer 2, Bit 4
15		29	Timer 2, Bit 5
16		31	Timer 2, Bit 6
17		33	Timer 2, Bit 7
18		35	Timer 1, Bit 7
19		37	Timer 1, Bit 6
20		39	Timer 1, Bit 5
21		41	Timer 1, Bit 4
22		43	Timer 1, Bit 0
23		45	Timer 1, Bit 1
24		47	Timer 1, Bit 2
25		49	Timer 1, Bit 3
26			Ground
27	Terminal		60 Hertz
28	Terminal		Ground
29	J4	8	Keyboard, Y4
30	J4	7	Keyboard, Y3
31	J4	6	Keyboard, Y2
32	J4	5	Keyboard, Y1
33	J4	4	Keyboard, X1
34	J4	3	Keyboard, X2
35	J4	2	Keyboard, X3
36	J4	1	Keyboard, X4

Table IX - SBC 80/10 and B-board Interconnection

(continued)

B-board Pin	SBC 80/10 Connector	PIN	Function
37			
38	J4	10	Reset
39			
40			
41			Ground
42			Panel Printer Busy
43			Ground
44			Trigger 2
45			Ground
46			Trigger 1
47			Ground
48			Unused
49			Ground
50			Print and Advance
51			Ground
52			Reset
53			
54 (n.c.)	J-1	31	Overload 2
55			Unused
56 (n.c.)	J-1	23	Overload 1
57			Unused
58			Enable DPM Output
59			Unused
60			Unused
61			800,000
62			400,000
63	J-1	27	200,000
64		21	100,000
65			Unused
66		17	80,000
67		19	40,000
68		29	20,000
69		25	10,000
70		Even	Ground

Table IX

(continued)

B-board PIN	SBC 80/10 Connector	PIN	Function
71			Unused
72	J-1	43	1
73		41	2
74		45	4
75		47	8
76			Unused
77		39	10
78		37	20
79		35	40
80		33	80
81			Unused
82		7	100
83		5	200
84		3	400
85		1	800
86			Unused
87		9	1000
88			2000
89			4000
90		15	8000
91			Unused
92		11	Sign
93		13	Data Polarity
94			.00000
95			.000000
96			.0000
97			.00
98			.000
99			.0
100		All even	Ground

Table IX

(continued)

APPENDIX B

LINKED LISTING

```
PROGRAM FILE: DK2:SYSTEM.BFF.PGM-0                    78/11/27  15:32:11
ENTRY POINT = #0000
RUN TIME DEFAULTS:
 STACK SIZE = #0064     TASK PRIORITY = 40     FILES = 0      BUFFERS = 0
    OBJECT FILE NAME              MODULE NAME     PROGRAM LIMITS  DATA LIMITS
DK2:WALT.MAIN.OBJ-1               MAIN              0000  06E2    3C70  3F5D
DK2:WALT.ADD.OBJ-0                FP.ADD            06E3  08C0    3F5E  3F6B
DK2:WALT.DIV.OBJ-0                FP.DIV            08C1  0A01    3F6C  3F93
DK2:WALT.MULT.OBJ-0               FP.MULT           0A02  0AFE    3F94  3FC2
DK2:WALT.PRINT.OBJ-0              OUTPUTS           0AFF  0B55
DK2:WALT.SORT.OBJ-0               SORT              0B56  0BD3    3FC3  3FDD
DK2:WALT.UTIL.OBJ-0               UTIL              0BD4  0C44
DK2:WALT.UTIL1.OBJ-0              UTIL1             0C45  0DF0    3FDE  3FE6
DK2:WALT.UTIL2.OBJ-0              UTIL2             0DF1  0FA1    3FE7  3FF7
  IDENTIFIER       ADDRESS    DEFINED          REFERENCED
BIN TO BCD          0C45    UTIL1              MAIN                UTIL2
BLOOD FLOW1         3D29    MAIN
BLOOD FLOW2         3D32    MAIN
BUFF COUNT          3C76    MAIN               UTIL1
BUFFER IN           0D9C    UTIL1              MAIN
CAL DIFF 1          3E97    MAIN
CAL DIFF 2          3EFB    MAIN
CH1 ACT             3C77    MAIN
CH2 ACT             3C78    MAIN
CIRCUM LIME1        3D17    MAIN
CIRCUM LIME2        3D20    MAIN
COMPARE             0DF1    UTIL2              MAIN
DEC ADD             0C1B    UTIL               MAIN                FP.ADD
DEC SUB             0C2D    UTIL               MAIN                FP.ADD
                            FP.DIV
DELAY               0B47    OUTPUTS
DIVIDE              08C1    FP.DIV             MAIN                SORT
                            UTIL2
DPP BUFF1A          3C79    MAIN
DPP BUFF1B          3C7C    MAIN
DPP BUFF2A          3C7F    MAIN
DPP BUFF2B          3C82    MAIN
ERROR OUT           0AFF    OUTPUTS            MAIN                FP.DIV
                            SORT               UTIL1
FP.ADD              06E3    FP.ADD             MAIN                SORT
                            UTIL2
FP.NUM              3D5F    MAIN               UTIL2
FP OUT FMT          0CA9    UTIL1              MAIN
FP.TO INT           0D3E    UTIL1              UTIL2
INPUT BUFF          3C70    MAIN               UTIL1
INT TO FP           0C89    UTIL1              MAIN                UTIL2
LARGER              0B0C    UTIL               MAIN                FP.ADD
                            FP.DIV             SORT                UTIL2
LEFT JUST           0B43    FP.ADD             FP.MULT             UTIL1
LENGTH GAUGE1       3D05    MAIN
LENGTH GAUGE2       3D0E    MAIN
LOW LIM INT         3FF5    UTIL2
```

MEAN	0E88	UTIL2	MAIN	
MEAN CAL1	3CCD	MAIN		
MEAN CAL2	3CD6	MAIN		
MEAN RDG1	3CBB	MAIN		
MEAN RDG2	3CC4	MAIN		
MEAN TIME1	3CF3	MAIN		
MEAN TIME2	3CFC	MAIN		
MULT	0A02	FP*MULT	MAIN	UTIL2
MULT BCD	0A65	FP*MULT		
MULTIPLICAND	3F9D	FP*MULT		
MULTIPLIER	3F94	FP*MULT		
N MINUS 1	3FF0	UTIL2		
NUM CAL 1	3E96	MAIN		
NUM CAL 2	3EFA	MAIN		
NUM RDG 1	3D68	MAIN		
NUM RDG 2	3DFF	MAIN		
NUM REMAINING	3FF0	UTIL2		
NUM TIME1	3CF1	MAIN		
NUM TIME2	3CF2	MAIN		
NUMBER CAL1	3D3B	MAIN		
NUMBER CAL2	3D4D	MAIN		
NUMBER RDG1	3D44	MAIN		
NUMBER RDG2	3D56	MAIN		
PRINT DELAY	0B43	OUTPUT1	MAIN	
PRINTOUT	0AFF	OUTPUT1	MAIN	
PRODUCT	3F80	FP*MULT		
RDG DIFF 1	3D6B	MAIN		
RDG DIFF 2	3E00	MAIN		
SAMPLE SIZE	3FF1	UTIL2		
SPACE	0B25	OUTPUT1	MAIN	FP*DIV
SORT	0B56	SORT	UTIL2	
SQUARE	3FE7	UTIL2		
START	0000	MAIN		
STD DEV	0EEB	UTIL2	MAIN	
STD DEV CAL1	3CA9	MAIN		
STD DEV CAL2	3CB2	MAIN		
STD DEV RDG1	3C97	MAIN		
STD DEV RDG2	3CA0	MAIN		
SUM	3C8E	MAIN	UTIL2	
SUM OF SQUARES	3C85	MAIN	UTIL2	
SUM TIME1	3CDF	MAIN		
SUM TIME2	3CE8	MAIN		
TEMP2	3FDE	UTIL1		
TRANS DATA	0AF6	FP*MULT	MAIN	FP*ADD
		FP*DIV	SORT	UTIL1
TRANS DATA LEFT	0BF7	UTIL	FP*ADD	FP*DIV
		UTIL1		
UP LIM INT	3FF2	UTIL2		
ZERO MEM	0BD4	UTIL	MAIN	FP*ADD
		FP*DIV	FP*MULT	UTIL1
		UTIL2		

APPENDIX C

INTERRUPT HANDLER PROGRAM

```
 BSAL-80/85.2.3   78/11/27  12:28:58
          *****   0 ERROR(S),   0 WARNING(S). SEE     0 *****
 SOURCE FILE =   DK2:WALT_MAIN.EDT-0
 OBJECT FILE =   DK2:WALT_MAIN.OBJ-1
00001.000    1 1 06E3
00002.000    2 1 0000          $CONTROL          NAME=MAIN      .
00003.000    3 1 0000          !$CONTROL          NOLIST
00004.000    4 1 0000          $CONTROL          LIST,INNERLIST,TABLE,CROSS
00005.000    5 1 0000
00006.000    6 1 0000          <<THIS IS THE MAIN ROUTINE FOR OPERATION OF THE
00007.000    7 1 0000          PLETHYSMOGRAPH. IT INITIALIZES THE SYSTEM AND THEN
00008.000    8 1 0000          AWAITS INTERRUPTS FOR SERVICING KEYBOARD INPUT OR
00009.000    9 1 0000          INPUT FROM THE PLETHYSMOGRAPH.                  >>
00010.000   10 1 0000
00011.000   11 1 0000          ENTRY   START;
00012.000   12 1 0000          EXTERNAL          FP_ADD, DIVIDE, MULT, TRANS_DATA,
00013.000   13 1 0000                            ZERO_MEM,  LARGER,  PRINTOUT,
00014.000   14 1 0000                            ERROR_OUT,  SPACE,  PRINT_DELAY,
00015.000   15 1 0000                            BIN_TO_BCD,  STD_DEV,  MEAN,
00016.000   16 1 0000                            FP_OUT_FMT,  INT_TO_FP,  COMPARE,
00017.000   17 1 0000                            BUFFER_IN, DEC_ADD, DEC_SUB;
00018.000   18 1 0000
00019.000   19 1 0000
00020.000   20 1 0000          GLOBAL  INPUT_BUFF(6),      BUFF_COUNT,   CH1_ACT,
00021.000   21 1 0000                  CH2_ACT,            DPP_BUFF1A(3),
00021.100   22 1 0000                  DPP_BUFF1B(3),      DPP_BUFF2A(3),
00022.000   23 1 0000                  DPP_BUFF2B(3),      SUM_OF_SQUARES(9),
00023.000   24 1 0000                  SUM(9),             STD_DEV_RDG1(9),
00023.100   25 1 0000                  STD_DEV_RDG2(9),    STD_DEV_CAL1(9),
00024.000   26 1 0000                  STD_DEV_CAL2(9),    MEAN_RDG1(9),
00025.000   27 1 0000                  MEAN_RDG2(9),       MEAN_CAL1(9),
00025.100   28 1 0000                  MEAN_CAL2(9),       SUM_TIME1(9),
00026.000   29 1 0000                  SUM_TIME2(9),       NUM_TIME1,
00027.000   30 1 0000                  NUM_TIME2,          MEAN_TIME1(9),
00027.100   31 1 0000                  MEAN_TIME2(9),
00028.000   32 1 0000                  LENGTH_GAUGE1(9),   LENGTH_GAUGE2(9),
00029.000   33 1 0000                  CIRCUM_LIMB1(9),    CIRCUM_LIMB2(9),
00030.000   34 1 0000                  BLOOD_FLOW1(9),     BLOOD_FLOW2(9),
00031.000   35 1 0000                  NUMBER_CAL1(9),     NUMBER_RDG1(9),
00032.000   36 1 0000                  NUMBER_CAL2(9),     NUMBER_RDG2(9),
00033.000   37 1 0000                  FP_NUM(9),
00034.000   38 1 0000                  NUM_RDG_1,          RDG_DIFF_1(150),
00035.000   39 1 0000                  NUM_RDG_2,          RDG_DIFF_2(150),
00036.000   40 1 0000                  NUM_CAL_1,          CAL_DIFF_1(99),
00037.000   41 1 0000                  NUM_CAL_2,          CAL_DIFF_2(99);
00038.000   42 1 0000
00039.000   43 1 0000          EQUATE  STACK= #3C6F;
00039.100   44 1 0000          EQUATE  KYBD_INTR= #8800;
00040.000   45 1 0000          EQUATE  OUT_CEL=#8000;
00040.100   46 1 0000          EQUATE  PRINT_DIV= #A100;
00041.000   47 1 0000          EQUATE  BIT_TIMER_BUF= #A000;
00041.100   48 1 0000          EQUATE  DPP_PRINT_INTR= #9800;
00042.000   49 1 0000          EQUATE  RESET_DP= #A000;
00043.000   50 1 0000          EQUATE  KYBD_IN= #7800;
00044.000   51 1 0000          EQUATE  NEG_SIGN= 0;
```

```
00045.000   52 1 0000          EQUATE  POS^IGN= %10000000;
00046.000   53 1 0000          EQUATE  POS TRU POLARY= 0;
00046.100   54 1 0000          EQUATE  NEG TRU POLARY= %01000000;
00047.000   55 1 0000          EQUATE  ALL DEC^PTS= %00111111;
00047.100   56 1 0000          EQUATE  CHAN 1= #05;
00048.000   57 1 0000          EQUATE  CHAN 2= #04;
00048.100   58 1 0000          EQUATE  ENTER= #03;
00049.000   59 1 0000          EQUATE  CAL BLD FLOW= #02;
00049.100   60 1 0000          EQUATE  CIRC LIMB= #01;
00050.000   61 1 0000          EQUATE  LENGTH GAUGE= #00;
00050.100   62 1 0000          EQUATE  PORT1= #E4;
00051.000   63 1 0000          EQUATE  PORT2= #E5;
00051.100   64 1 0000          EQUATE  PORT3= #E6;
00052.000   65 1 0000          EQUATE  PORT4= #E8;
00052.100   66 1 0000          EQUATE  PORT5= #E9;
00053.000   67 1 0000          EQUATE  PORT6= #EA;
00054.000   68 1 0000          EQUATE  MAX READINGS = 50;
00055.000   69 1 0000          EQUATE  MAX CALIBRATIONS = 33;
00056.000   70 1 0000          EQUATE  BYTES^PER^PNG = 2;
00057.000   71 1 0000          EQUATE  BYTES^PER^CAL = 3;
00058.000   72 1 0000          EQUATE  TURN ON DELAY=40;  ! DELAY TURN ON^DELAY
00058.100   73 1 0000                                     !  TIME; 500 MSEC
00059.000   74 1 0000
00060.000   75 1 0000          START:
00061.000   76 1 0000 3AFFFF          A=MEM^(#FFFF);  !NOP;NOP;NOP;
00063.000   77 1 0003 F3              DISABLE INTERRUPT;
00064.000   78 1 0004 316F3C          SP=STACK;
00065.000   79 1 0007 CDCE05          CALL INITIALIZE;
00066.000   80 1 000A 000000          NOP;NOP;NOP;
00067.000   81 1 000D 00              NOP;
00068.000   82 1 000E FB              ENABLE INTERRUPT;
00068.000   83 1 000F 000000  STOP:   NOP;NOP;NOP;
00070.000   84 1 0012 C30F00          GOTO STOP;
00071.000   85 1 0015                 DS #38 - #15;
00072.000   86 1 0038
00073.000   87 1 0038          INTERRUPTION:
00074.000   88 1 0038                 << IS IT AN INTERRUPT FROM THE KEYBOARD?>>
00075.000   89 1 0038 F3              DISABLE INTERRUPT;
00076.000   90 1 0039 3A0078          A=MEM^(KYBD IN);
00077.000   91 1 003C 47              B=A;
00078.000   92 1 003D E620            A=A AND #20;    ! MASK OFF INTERRUPT
00079.000   93 1 003F C21704          IF NONZERO THEN GOTO READ^DPP;
00080.000   94 1 0042 78              A=B;
00081.000   95 1 0043 E60F            A=A AND #0F;    ! MASK OFF KYBD DATA
00082.000   96 1 0045 47              B=A;
00083.000   97 1 0046 FE03            A-ENTER;
00084.000   98 1 0048 CA6E03          IF ZERO THEN GOTO STORE^BUFF;  ! IF ENTER;
00085.000   99 1 004B FE02            A-CAL BLD FLOW;
00086.000  100 1 004D CA8100          IF ZERO THEN GOTO COMPUTE BF;
00087.000  101 1 0050
00088.000  102 1 0050                 << STORE ENTRY IN BUFFER >>
00089.000  103 1 0050 3A0000          A=BUFF^COUNT;
00090.000  104 1 0053 FE06            A-6;
00091.000  105 1 0055                 IF ZERO                                    ■
00092.000  106 1 0055 C27200            THEN BEGIN
```

```
00093.000  107 2 0058          INPUT'ERROR:<< INPUT BUFFER FULL, OUTPUT AN ERROR
00094.000  108 2 0058                         OF 3 >>
00095.000  109 2 0058                       B=POS'SIGN OR POS'TRU'POLARY
00095.100  110 2 0058 06BF                            OR ALL'DEC'PTS;
00096.000  111 2 005A 0E33                  C=#33;
00097.000  112 2 005C 113333                DE=#3333;
00098.000  113 2 005F CD0000X               CALL PRINTOUT;
00099.000  114 2 0062 CD0000'               CALL SPACE;
00100.000  115 2 0065          CLR'IN'BUFF:         ! CLEAR INPUT'BUFF & COUNT
00101.000  116 2 0065 2100006               HL=@INPUT'BUFF;
00102.000  117 2 0068 0607                  B=7;
00103.000  118 2 006A CD0000X               CALL ZERO'MEM;
00104.000  119 2 006D          RESET'KYBD'INTR:
00105.000  120 2 006D 3200B8                MEM(KYBD'INTR)=A;
00106.000  121 2 0070 FB                    ENABLE INTERRUPT;
00107.000  122 2 0071 C9                    RETURN;
00108.000  123 2 0072                     END
00109.000  124 1 0072                     ELSE BEGIN
00110.000  125 2 0072                       <<INPUT BUFFER NOT FULL.  STORE THE
00110.100  126 2 0072                       ENTRY IN THE INPUT BUFFER AND
00110.200  127 2 0072                       INCREMENT THE BUFFER COUNT.      >>
00111.000  128 2 0072 1E005F                D=0;E=A;             ! BUFF COUNT
00112.000  129 2 0075 2100006               HL=@INPUT'BUFF;
00113.000  130 2 0078 19                    HL=HL+DE;
00114.000  131 2 0079 70                    MEM'HL'=B;
00115.000  132 2 007A 2106006               HL=@BUFF'COUNT;
00116.000  133 2 007D 34                    MEM'HL'=MEM'HL'+1;
00117.000  134 2 007E 3200B8                MEM'KYBD'INTR'=A;
00118.000  135 2 0081 FB                    ENABLE INTERRUPT;
00119.000  136 2 0082 C9                    RETURN;
00120.000  137 1 0083                     END;
00121.000  138 1 0083          COMPUTE'BF:     ! ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆ ◆
00122.000  139 1 0083                     << CALCULATE THE BLOOD FLOW & PROVIDE
00123.000  140 1 0083                       THE OUTPUT                     >>
00124.000  141 1 0083 3AF8006             A=NUM PTS 1;
00125.000  142 1 0086 FE02                A-2;
00126.000  143 1 0088                     IF NEGATIVE
00126.100  144 1 0088 F29300P               THEN BEGIN
00126.200  145 2 008B 3EFF                    A=#FF;
00126.300  146 2 008D 32B900G                 BLOOD'FLOW1=A; ! SET FLAG SHOWING
00126.400  147 2 0090                                ! NO DATA FOR CHAN 1
00126.500  148 2 0090 C3EE01P                 GOTO CHANN'2;
00126.600  149 1 0093                       END;
00127.000  150 1 0093                     << CHANNEL 1 HAS AT LEAST ONE READING.
00128.000  151 1 0093                       CALCULATE THE STD DEV FOR CHAN 1.>>
00129.000  152 1 0093          CHANN'1:
00129.100  153 1 0093 3A0000G             A =INPUT BUFF;
00129.200  154 1 0096 FE0D                A-#0D;  ! 2 KEY
00129.300  155 1 0098 CACD00P             IF ZERO THEN GOTO MEAN1;
00129.400  156 1 009B                                ! SKIP STD DEV COMPUTATION
00130.000  157 1 009B 21F8006             HL=@NUM PTS 1;
00131.000  158 1 009E 1127006             DE=@STD DEV PTS1;
00132.000  159 1 00A1 014B006             BC=@MEAN PTS1;
```

```
00133.000  160 1 00A4 3E02            A=BYTE@PER@RDG;
00134.000  161 1 00A6 CD0000X         CALL STD@DEV;    ! HL=@STD DEV
00135.000  162 1 00A9 11E000          DE=@MEAN@RDG1;
00136.000  163 1 00AC 01F800          BC=@NUM RDG 1;
00137.000  164 1 00AF 3E02            A=BYTE@PER@RDG;
00138.000  165 1 00B1 CD0000X         CALL COMPARE;    << FLAG:DATA OUTSIDE
00139.000  166 1 00B4                                    +/- 2 STD DEV    >>
00140.000  167 1 00B4 212602          HL=@NUM@CAL@1;
00141.000  168 1 00B7 113800          DE=@STD DEV CAL1;
00142.000  169 1 00BA 015D00          BC=@MEAN@CAL1;
00143.000  170 1 00BD 3E01            A=BYTE@PER@CAL;
00144.000  171 1 00BF CD0000X         CALL STD@DEV;    ! HL=@STD DEV
00145.000  172 1 00C2 115D00          DE=@MEAN@CAL1;
00146.000  173 1 00C5 012602          BC=@NUM@CAL@1;
00147.000  174 1 00C8 3E03            A=BYTE@PER@CAL;
00148.000  175 1 00CA CD0000X         CALL COMPARE;
00149.000  176 1 00CD                 << FIND A NEW MEAN FOR CALIBRATION
00150.000  177 1 00CD                     AND READING:                 >>
00151.000  178 1 00CD 0602     MEAN1:  B=BYTE@PER@RDG;
00152.000  179 1 00CF 11E000          DE=@MEAN RDG1;
00153.000  180 1 00D2 21F800          HL=@NUM RDG 1;
00154.000  181 1 00D5 CD0000X         CALL MEAN;
00155.000  182 1 00D8 0609            B=9;
00156.000  183 1 00DA 11EF00          DE=@FF NUM;      ! SAMPLE SIZE
00157.000  184 1 00DD 21D400          HL=@NUMBER RDG1;
00158.000  185 1 00E0 CD0000X         CALL TRANS DATA;
00159.000  186 1 00E3 0603            B=BYTE@PER@CAL;
00160.000  187 1 00E5 115D00          DE=@MEAN CAL1;
00161.000  188 1 00E8 212602          HL=@NUM CAL 1;
00162.000  189 1 00EB CD0000X         CALL MEAN;
00163.000  190 1 00EE 0609            B=9;
00164.000  191 1 00F0 11EF00          DE=@FF@NUM;      ! SAMPLE SIZE
00165.000  192 1 00F3 21CB00          HL=@NUMBER CAL1;
00166.000  193 1 00F6 CD0000X         CALL TRANS DATA;
00167.000  194 1 00F9                 << CALCULATE THE MEAN TIME READING >>
00168.000  195 1 00F9 3A8100          A=NUM TIME1;
00169.000  196 1 00FC 16005F          D=0;E=A;
00170.000  197 1 00FF 210000          HL=@BFF BUFFER;
00170.100  198 1 0102 CD0000X         CALL BIN TO BCD; !HL=@BCD RESULT
00170.200  199 1 0105 7E              A=MEM<HL>;
00171.000  200 1 0106 328100          NUM@TIME1=A;
00172.000  201 1 0109 0601            B=1;
00173.000  202 1 010B 118100          DE=@NUM TIME1;
00174.000  203 1 010E 21EF00          HL=@FF@NUM;
00175.000  204 1 0111 CD0000X         CALL INT@TO@FP; ! HL=@FF@NUM
00176.000  205 1 0114 EB              DE<==>HL;
00177.000  206 1 0115 216F00          HL=@SUM TIME1;
00178.000  207 1 0118 011E00          BC=@SUM;
00179.000  208 1 011B CD0000X         CALL DIVIDE;     ! HL=@SUM
00180.000  209 1 011E 01CB00          BC=@MEAN TIME1;
00181.000  210 1 0121 11C505          DE=@DIVTY;
00182.000  211 1 0124 CD0000X         CALL DIVIDE;
00183.000  212 1 0127                 << FIND BLOOD FLOW                >>
00184.000  213 1 0127 01EF00          BC=@FF NUM;
```

```
00185.000  214 1 012A 114B00G          DE=@MEAN'PDG1;
00186.000  215 1 012D 2195006          HL=@LENGTH'GAUGE1;
00187.000  216 1 0130 CD0000'          CALL MULT;        ! HL=@FP'NUM'PRODUCT>
00188.000  217 1 0133 011E006          BC=@SUM;
00189.000  218 1 0136 11BC05F          DE=@ONE'TWENTY;
00190.000  219 1 0139 CD0000'          CALL MULT;        << HL=@SUM'PRODUCT>
00191.000  220 1 013C                                    NUMERATOR OF BLD FLW>>
00192.000  221 1 013C 01EF006          BC=@FP'NUM;
00193.000  222 1 013F 115D006          DE=@MEAN'CAL1;
00194.000  223 1 0142 21A7006          HL=@CIRCUM'LIMB1;
00195.000  224 1 0145 CD0000'          CALL MULT;        ! HL=@FP'NUM
00196.000  225 1 0148 0115006          BC=@SUM'OF'SQUARES;
00197.000  226 1 014B 1183006          DE=@MEAN'TIME1;
00198.000  227 1 014E CD0000X          CALL MULT;        << HL=@PRODUCT OF
00199.000  228 1 0151                                    DENOMINATOR OF BLD FLW>>
00200.000  229 1 0151 EB               DE<==>HL;
00201.000  230 1 0152 211E006          HL=@SUM;
00202.000  231 1 0155 01B9006          BC=@BLOOD FLOW1;
00203.000  232 1 0158 CD0000'          CALL DIVIDE;
00204.000  233 1 015B                  << OUTPUT CHANNEL 1 DATA >>
00204.100  234 1 015B CD0000'          CALL SPACE;
00205.000  235 1 015E 2183006          HL=@MEAN TIME1; ! OUTPUT MEAN TIME
00206.000  236 1 0161 CD0000'          CALL FP'OUT FMT;
00207.000  237 1 0164 0E12             C=#12;
00208.000  238 1 0166 CD0000X          CALL PRINTOUT;
00208.100  239 1 0169 CD0000'          CALL SPACE;
00209.000  240 1 016C 2195006          HL=@LENGTH GAUGE1;  !OUTPUT
00210.000  241 1 016F CD0000'          CALL FP OUT FMT;
00211.000  242 1 0172 0E13             C=#13;
00212.000  243 1 0174 CD0000'          CALL PRINTOUT;
00212.100  244 1 0177 21A7006          HL=@CIRCUM LIMB1;   ! OUTPUT
00214.000  245 1 017A CD0000'          CALL FP OUT FMT;
00215.000  246 1 017D 0E14             C=#14;
00216.000  247 1 017F CD0000X          CALL PRINTOUT;
00216.100  248 1 0182 CD0000'          CALL SPACE;
00217.000  249 1 0185                  << OUTPUT CALIBRATION INFO: SAMPLE SIZE,
00218.000  250 1 0185                     MEAN  & STD DEV: IF CALCULATED      >>
00219.000  251 1 0185 21CB006          HL=@NUMBER'CAL1;
00220.000  252 1 0188 CD0000'          CALL FP'OUT FMT;
00221.000  253 1 018B 0E15             C=#15;
00222.000  254 1 018D CD0000X          CALL PRINTOUT;
00223.000  255 1 0190 215D006          HL=@MEAN'CAL1;
00224.000  256 1 0193 CD0000'          CALL FP'OUT FMT;
00225.000  257 1 0196 0E15             C=#15;
00226.000  258 1 0198 CD0000X          CALL PRINTOUT;
00226.100  259 1 019B 3A0000G          A=INPUT'BUFF;
00226.200  260 1 019E FE0D             A-=0D;  ! 2 KEY
00226.300  261 1 01A0 CAB101F          IF ZERO THEN GOTO PDG1;
00227.000  262 1 01A3 2194006          HL=@STD'DEV'CAL1;
00228.000  263 1 01A6 CD0000'          CALL FP'OUT FMT;
00229.000  264 1 01A9 0E15             C=#15;
00230.000  265 1 01AB CD0000X          CALL PRINTOUT;
00230.100  266 1 01AE CD0000'          CALL SPACE;
00231.000  267 1 01B1          PDG1:   << OUTPUT READING INF: SAMPLE SIZE,MEAN
00232.000  268 1 01B1                     & STD DEV: IF CALCULATED          >>
```

```
00233.000  268 1 01B1 21D400G          HL=@NUMBER'RDG1;
00234.000  270 1 01B4 CD0000X          CALL FP'OUT'FMT;
00235.000  271 1 01B7 0E16             C=#16;
00236.000  272 1 01B9 CD0000X          CALL PRINTOUT;
00237.000  273 1 01BC 214F00G          HL=@MEAN'RDG1;
00238.000  274 1 01BF CD0000X          CALL FP'OUT'FMT;
00239.000  275 1 01C2 0E16             C=#16;
00240.000  276 1 01C4 CD0000X          CALL PRINTOUT;
00240.100  277 1 01C7 3A0000G          A=INPUT'BUFF;
00240.200  278 1 01CA FE0D             A-#0D;  ! 2 KEY
00240.300  279 1 01CC CADD01F          IF ZERO THEN GOTO BF1;
00241.000  280 1 01CF 212700G          HL=@STD'DEV'RDG1;
00242.000  281 1 01D2 CD0000X          CALL FP'OUT'FMT;
00243.000  282 1 01D5 0E16             C=#16;
00244.000  283 1 01D7 CD0000X          CALL PRINTOUT;
00245.000  284 1 01DA CD0000X          CALL SPACE;
00246.000  285 1 01DD 21B900G BF1:     HL=@BLOOD'FLOW1;  ! OUTPUT BLD FLOW
00247.000  286 1 01E0 CD0000X          CALL FP'OUT'FMT;
00248.000  287 1 01E3 0E17             C=#17;
00249.000  288 1 01E5 CD0000X          CALL PRINTOUT;
00250.000  289 1 01E8 CD0000X          CALL SPACE;
00250.100  290 1 01EB CD0000X          CALL SPACE;
00251.000  291 1 01EE          CHANN'2:
00252.000  292 1 01EE                 <<IF CHANNEL 2 HAS MORE THAN ONE READING,
00253.000  293 1 01EE                   CALCULATE AND OUTPUT ITS BLOOD FLOW >>
00254.000  294 1 01EE 3A8F01G          A=NUM'RDG 2;
00255.000  295 1 01F1 FE02             A-2;
00256.000  296 1 01F3                  IF NEGATIVE  ! NO DATA CHAN 2
00256.010  297 1 01F3 F20E02F            THEN BEGIN
00256.020  298 2 01F6 3AB900G              A=BLOOD'FLOW1; ! WAS THERE DATA
00256.030  299 2 01F9 FEFF                 A-#FF;         !   FOR CHAN 1?
00256.040  300 2 01FB                      IF ZERO
00256.050  301 2 01FB C20E02F                THEN BEGIN
00256.060  302 3 01FE                          B=POS SIGN OR POS TRU POLARY
00256.070  303 3 01FE 06FF                          OF ALL DEC PTS;
00256.080  304 3 0200 0E11                     C=#11;
00256.090  305 3 0202 111111                   DE=#1111;
00256.091  306 3 0205 CD0000X                  CALL ERROR'OUT;
00256.092  307 3 0208 CD0000X                  CALL SPACE;
00256.093  308 2 020B                        END;
00256.094  309 2 020B C3B903F              GOTO CLEAR'MEM;
00256.095  310 1 020E                    END;
00256.100  311 1 020E 3A0000G          A=INPUT'BUFF;
00256.200  312 1 0211 FE0D             A-#0D;  ! 2 KEY
00256.300  313 1 0213 CA4802F          IF ZERO THEN GOTO MEAN2;
00256.400  314 1 0216                                ! CHIP CTD DEV COMPUTATION
00257.000  315 1 0216 218F01G          HL=@NUM'RDG 2;
00258.000  316 1 0219 113000G          DE=@CTD'DEV'RDG2;
00258.100  317 1 021C 015400G          BC=@MEAN'RDG2;
00258.200  318 1 021F 3E02             A=BYTES'PER'RDG;
00261.000  319 1 0221 CD0000X          CALL CTD'DEV;   ! HL=@CTD'DEV
00262.000  320 1 0224 115400G          DE=@MEAN'RDG2;
00263.000  321 1 0227 018F01G          BC=@NUM'RDG 2;
00264.000  322 1 022A 3E02             A=BYTES'PER'RDG;
00265.000  323 1 022C CD0000X          CALL COMPARE;    << FLAGS DATA OUTSIDE
00266.000  324 1 022F                                    +-- 2 STD DEV    >>
```

```
00267.000   325 1 022F 218A026          HL=@NUM'CAL'2;
00268.000   326 1 0232 1148006          DE=@CTD'DEV'CAL2;
00269.000   327 1 0235 0166006          BC=@MEAN'CAL2;
00270.000   328 1 0238 3E03              A=BYTES'PER'CAL;
00271.000   329 1 023A CD0000X          CALL CTD'DEV;    ! HL=@CTD DEV
00272.000   330 1 023D 1166006          DE=@MEAN'CAL2;
00273.000   331 1 0240 018A026          BC=@NUM'CAL'2;
00274.000   332 1 0243 3E03              A=BYTES PER CAL;
00275.000   333 1 0245 CD0000X          CALL COMPARE;
00276.000   334 1 0248                  << FIND A NEW MEAN FOR CALIBRATION
00277.000   335 1 0248                      AND READING!                  >>
00278.000   336 1 0248 0602     MEAN2:  B=BYTES'PER'RDG;
00279.000   337 1 024A 1154006          DE=@MEAN'RDG2;
00280.000   338 1 024D 218F016          HL=@NUM'RDG'2;
00281.000   339 1 0250 CD0000X          CALL MEAN;
00282.000   340 1 0253 0609             B=9;
00283.000   341 1 0255 11EF006          DE=@FP'NUM;      ! SAMPLE SIZE
00284.000   342 1 0258 21E6006          HL=@NUMBER'RDG2;
00285.000   343 1 025B CD0000X          CALL TRANS'DATA;
00286.000   344 1 025E 0603             B=BYTES'PER'CAL;
00287.000   345 1 0260 1166006          DE=@MEAN'CAL2;
00288.000   346 1 0263 218A026          HL=@NUM'CAL'2;
00289.000   347 1 0266 CD0000X          CALL MEAN;
00290.000   348 1 0269 0609             B=9;
00291.000   349 1 026B 11EF006          DE=@FP'NUM;      ! SAMPLE SIZE
00292.000   350 1 026E 21DD006          HL=@NUMBER'CAL2;
00293.000   351 1 0271 CD0000X          CALL TRANS'DATA;
00294.000   352 1 0274                  << CALCULATE THE MEAN TIME READING>>
00295.000   353 1 0274 3A82006          A=NUM TIME2;
00296.000   354 1 0277 16005F           D=0;E=A;
00296.100   355 1 027A 210F006          HL=@DPP BUFF2A;
00296.200   356 1 027D CD0000X          CALL BIN TO BCD; !HL=@BCD RESULT
00296.300   357 1 0280 7E                A=MEM(HL);
00296.400   358 1 0281 3282006          NUM TIME2=A;
00296.500   359 1 0284 0601             B=1;
00300.000   360 1 0286 1182006          DE=@NUM'TIME2;
00301.000   361 1 0289 21EF006          HL=@FP'NUM;
00302.000   362 1 028C CD0000X          CALL INT'TO FP; ! HL=@FP'NUM
00303.000   363 1 028F EB               DE ==HL;
00304.000   364 1 0290 2178006          HL=@SUM'TIME2;
00305.000   365 1 0293 011E006          BC=@SUM;
00306.000   366 1 0296 CD0000X          CALL DIVIDE;     ! HL=@SUM
00307.000   367 1 0299 018C006          BC=@MEAN'TIME2;
00308.000   368 1 029C 11C505F          DE=@CIXTY;
00309.000   369 1 029F CD0000X          CALL DIVIDE;
00310.000   370 1 02A2                  << FIND BLOOD FLOW                >>
00311.000   371 1 02A2 01EF006          BC=@FP'NUM;
00312.000   372 1 02A5 1154006          DE=@MEAN'RDG2;
00313.000   373 1 02A8 219E006          HL=@LENGTH GAUGE2;
00314.000   374 1 02AB CD0000X          CALL MULT;       ! HL=@FP NUM(PRODUCT)
00315.000   375 1 02AE 011E006          BC=@SUM;
00316.000   376 1 02B1 11BC05F          DE=@ONE TWENTY;
00317.000   377 1 02B4 CD0000X          CALL MULT;           ! HL=@SUM(PRODUCT)
00318.000   378 1 02B7                                  NUMERATOR OF BLD FLW>>
```

```
00319.000  379 1 02B7 01EF00G        BC=@FP'NUM;
00320.000  380 1 02BA 1166006        DE=@MEAN'CAL2;
00321.000  381 1 02BD 21B000G        HL=@CIRCUM'LIMB2;
00322.000  382 1 02C0 CD0000X        CALL MULT;       ! HL=@FP'NUM
00323.000  383 1 02C3 0115006        BC=@SUM'OF'SQUARES;
00324.000  384 1 02C6 1180006        DE=@MEAN'TIME2;
00325.000  385 1 02C9 CD0000X        CALL MULT;       << HL=@PRODUCT OF
00326.000  386 1 02CC                                   DENOMINATOR OF BLD FLW>>
00327.000  387 1 02CC EB             DE'=='HL;
00328.000  388 1 02CD 211E006        HL=@SUM;
00329.000  389 1 02D0 01C2006        BC=@BLOOD'FLOW2;
00330.000  390 1 02D3 CD0000X        CALL DIVIDE;
00331.000  391 1 02D6                << OUTPUT CHANNEL 2 DATA >>
00331.100  392 1 02D6 CD0000X        CALL SPACE;
00332.000  393 1 02D9 2180006        HL=@MEAN'TIME2; ! OUTPUT MEAN TIME
00333.000  394 1 02DC CD0000X        CALL FP'OUT'FMT;
00334.000  395 1 02DF 0E22           C=#22;
00335.000  396 1 02E1 CD0000X        CALL PRINTOUT;
00335.100  397 1 02E4 CD0000X        CALL SPACE;
00336.000  398 1 02E7 219E006        HL=@LENGTH'GAUGE2;  !OUTPUT
00337.000  399 1 02EA CD0000X        CALL FP'OUT'FMT;
00338.000  400 1 02ED 0E23           C=#23;
00339.000  401 1 02EF CD0000X        CALL PRINTOUT;
00340.000  402 1 02F2 21B000G        HL=@CIRCUM'LIMB2;   ! OUTPUT
00341.000  403 1 02F5 CD0000X        CALL FP'OUT'FMT;
00342.000  404 1 02F8 0E24           C=#24;
00343.000  405 1 02FA CD0000X        CALL PRINTOUT;
00343.100  406 1 02FD CD0000X        CALL SPACE;
00344.000  407 1 0300                << OUTPUT CALIBRATION INFO: SAMPLE SIZE,
00345.000  408 1 0300                   MEAN & STD DEV: IF CALCULATED      >>
00346.000  409 1 0300 21DD006        HL=@NUMBER'CAL2;
00347.000  410 1 0303 CD0000X        CALL FP'OUT'FMT;
00348.000  411 1 0306 0E25           C=#25;
00349.000  412 1 0308 CD0000X        CALL PRINTOUT;
00350.000  413 1 030B 2166006        HL=@MEAN'CAL2;
00351.000  414 1 030E CD0000X        CALL FP'OUT'FMT;
00352.000  415 1 0311 0E25           C=#25;
00353.000  416 1 0313 CD0000X        CALL PRINTOUT;
00353.100  417 1 0316 3A0000G        A=INPUT'BUFF;
00353.200  418 1 0319 FE0D           A-=0D;  ! 2 KEY
00353.300  419 1 031B CA2C03P        IF ZERO THEN GOTO RDG2;
00354.000  420 1 031E 2142006        HL=@STD'DEV'CAL2;
00355.000  421 1 0321 CD0000X        CALL FP'OUT'FMT;
00356.000  422 1 0324 0E25           C=#25;
00357.000  423 1 0326 CD0000X        CALL PRINTOUT;
00357.100  424 1 0329 CD0000X        CALL SPACE;
00358.000  425 1 032C          RDG2: << OUTPUT READING INF: SAMPLE SIZE,MEAN
00359.000  426 1 032C                   & STD DEV: IF CALCULATED       >>
00360.000  427 1 032C 21E6006        HL=@NUMBER'RDG2;
00361.000  428 1 032F CD0000X        CALL FP'OUT'FMT;
00362.000  429 1 0332 0E26           C=#26;
00363.000  430 1 0334 CD0000X        CALL PRINTOUT;
00364.000  431 1 0337 2154006        HL=@MEAN'RDG2;
```

```
00365.000  432 1 033A CD0000X          CALL FP'OUT'FMT;
00366.000  433 1 033D 0E26             C=#26;
00367.000  434 1 033F CD0000X          CALL PRINTOUT;
00367.100  435 1 0342 3A0000G          A=INPUT'BUFF;
00367.200  436 1 0345 FE0D             A-#0D;  ! 2 KEY
00367.300  437 1 0347 CA5803F          IF ZERO THEN GOTO BF2;
00368.000  438 1 034A 2130006          HL=@STD'DEV'FDG2;
00369.000  439 1 034D CD0000X          CALL FP'OUT'FMT;
00370.000  440 1 0350 0E26             C=#26;
00371.000  441 1 0352 CD0000X          CALL PRINTOUT;
00372.000  442 1 0355 CD0000X          CALL SPACE;
00373.000  443 1 0358 21C2006 BF2:     HL=@BLOOD'FLOW2;  ! OUTPUT BLD FLOW
00374.000  444 1 035B CD0000X          CALL FP'OUT'FMT;
00375.000  445 1 035E 0E27             C=#27;
00376.000  446 1 0360 CD0000X          CALL PRINTOUT;
00377.000  447 1 0363 CD0000X          CALL SPACE;
00377.100  448 1 03ee CD0000X          CALL SPACE;
00378.000  449 1 0369            CLEAR'MEM:
00379.000  450 1 0369 0690             B=#3CFF - STACK;
00380.000  451 1 036B 217030           HL=STACK + 1;
00381.000  452 1 036E 1604             D=4;
00382.000  453 1 0370            REPEAT:
00383.000  454 1 0370 CD0000X          CALL ZERO'MEM;
00384.000  455 1 0373 05               B=B-1;
00385.000  456 1 0374 15               D=D-1;
00386.000  457 1 0375 C270003F         IF NONZERO THEN GOTO REPEAT;
00387.000  458 1 0378 3200E8           MEM(KYBD INTR)=A;
00387.100  459 1 037B 3200A0           MEM(RIT TIMER BUF)=A;
00387.200  460 1 037E 320090           MEM(RECIET DP)=A;
00387.300  461 1 0381 320098           MEM(DPP'PRINT INTR)=A;
00388.000  462 1 0384 FB               ENABLE INTERRUPT;
00389.000  463 1 0385 C9               RETURN;
00390.000  464 1 0386
00391.000  465 1 0386            STORE'BUFF:     ! * * * * * * * * *
00392.000  466 1 0386
00393.000  467 1 0386            ! THE LAST KEYBOARD INPUT WAS ENTER.  STORE
00394.000  468 1 0386            THE CONTENTS OF THE BUFFER ACCORDING TO WHETHER
00395.000  469 1 0386            IT IS CH1'CH2 AND LENGTH FO GAUGE'LIMB CIRCUM.
00396.000  470 1 0386            THEN OUTPUT THE INPUT DATA.                    >>
00397.000  471 1 0386
00398.000  472 1 0386 3A0600G          A=BUFF'COUNT;
00399.000  473 1 0389 D602             A=A-2;
00400.000  474 1 038B 47               B=A;
00401.000  475 1 038C CA5800F          IF ZERO THEN GOTO INPUT ERROR;
00402.000  476 1 038F 2100006          HL=@INPUT'BUFF;
00403.000  477 1 0392 7E               A=MEM(HL);
00404.000  478 1 0393 FE05             A-CHAN'1;
00405.000  479 1 0395                  IF ZERO         ! CHANNEL 1
00406.000  480 1 0395 C2D203F            THEN BEGIN
00407.000  481 2 0398 23                   HL=HL+1;
00408.000  482 2 0399 7E                   A=MEM(HL);
00409.000  483 2 039A FE00                 A-LENGTH GAUGE;
00410.000  484 2 039C                      IF ZERO
00411.000  485 2 039C C2B303F                THEN BEGIN ! STORE GAUGE LENGTH
00412.000  486 3 039F 1195006                 DE=@LENGTH GAUGE1;
```

```
00413.000  487 3 03A2 CD0000X          CALL BUFFER IN; !HL=@LENGTH GAUGE1
00414.000  488 3 03A5 CD0000X          CALL FP OUT FMT; !OUTPUT DATA
00415.000  489 3 03A8 0E13             C=#13;
00416.000  490 3 03AA CD0000X          CALL PRINTOUT;
00417.000  491 3 03AD CD0000X          CALL SPACE;
00418.000  492 3 03B0                  END
00419.000  493 2 03B0 C3CF03P          ELSE BEGIN   ! STORE CIRCUM OF LIMB
00420.000  494 3 03B3 FE01               A-CIRC LIMB;
00421.000  495 3 03B5                    IF ZERO
00422.000  496 3 03B5 C2CC03P              THEN BEGIN ! STORE @OUTPUT LIMB
00423.000  497 4 03B8                                 !   CIRCUMFERENCE
00424.000  498 4 03B8 11A700G                DE=@CIRCUM LIMB1;
00425.000  499 4 03BB CD0000X                CALL BUFFER IN;
00425.100  500 4 03BE                                 !  HL=@CIRCUM LIMB1
00426.000  501 4 03BE CD0000X                CALL FP OUT FMT;
00427.000  502 4 03C1 0E14                   C=#14;
00428.000  503 4 03C3 CD0000X                CALL PRINTOUT;
00429.000  504 4 03C6 CD0000X                CALL SPACE;
00430.000  505 4 03C9                      END
00431.000  506 3 03C9 C3CF03P              ELSE GOTO INPUT ERROR;
           3 03CC C35800P
00432.000  507 3 03CF                          !NEITHER LG NOR CI
00434.000  508 2 03CF                  END;
00435.000  509 2 03CF              END
00436.000  510 1 03CF C31404P      ELSE BEGIN
00437.000  511 2 03D2 FE04           A-CHAN 2;
00438.000  512 2 03D4                IF ZERO
00439.000  513 2 03D4 C21104P          THEN BEGIN  ! CHANNEL 2
00440.000  514 3 03D7 23                 HL=HL+1;
00441.000  515 3 03D8 7E                 A=MEM(HL);
00442.000  516 3 03D9 FE00               A-LENGTH GAUGE;
00443.000  517 3 03DB                    IF ZERO
00444.000  518 3 03DB C2F203P              THEN BEGIN ! STORE GAUGE LENGTH
00445.000  519 4 03DE 119E00G                DE=@LENGTH GAUGE2;
00446.000  520 4 03E1 CD0000X                CALL BUFFER IN;
00446.100  521 4 03E4                                 ! HL= @LENGTH GAUGE2
00447.000  522 4 03E4 CD0000X                CALL FP OUT FMT; !OUTPUT DATA
00448.000  523 4 03E7 0E23                   C=#23;
00449.000  524 4 03E9 CD0000X                CALL PRINTOUT;
00450.000  525 4 03EC CD0000X                CALL SPACE;
00451.000  526 4 03EF                      END
00452.000  527 3 03EF C30E04P              ELSE BEGIN !STORE CIRCUM OF LIMB
00453.000  528 4 03F2 FE01                   A-CIRC LIMB;
00454.000  529 4 03F4                        IF ZERO
00455.000  530 4 03F4 C20B04P                  THEN BEGIN ! STORE @ OUTPUT
00456.000  531 5 03F7                                   !  LIMB CIRCUMFERENCE
00457.000  532 5 03F7 11B000G                    DE=@CIRCUM LIMB2;
00458.000  533 5 03FA CD0000X                    CALL BUFFER IN;
00458.100  534 5 03FD                                   ! HL= @CIRCUM LIMB2
00459.000  535 5 03FD CD0000X                    CALL FP OUT FMT;
00460.000  536 5 0400 0E24                       C=#24;
00461.000  537 5 0402 CD0000X                    CALL PRINTOUT;
00462.000  538 5 0405 CD0000X                    CALL SPACE;
00463.000  539 5 0408                          END
```

```
00464.000   540 4 0408 C30E04F                    ELSE GOTO INPUT'ERROR;
            4 040B C35800F
00465.000   541 4 040E                                        !NEITHER LG NOR CI
00467.000   542 3 040E                          END;
00468.000   543 3 040E                        END
00469.000   544 2 040E C31404F                ELSE GOTO INPUT'ERROR; !NEITHER CH1
            2 0411 C35800F
00470.000   545 2 0414                                        !  NOR CH2
00471.000   546 2 0414
00473.000   547 1 0414                      END;
00474.000   548 1 0414 C36500F             GOTO CLR'IN'BUFF;
00475.000   549 1 0417
00476.000   550 1 0417              READ'DPP:       ! * * * * * * * * *
00477.000   551 1 0417                            <<INPUTS: FINDS THE DIFFERENCE IN RDGS, &
00478.000   552 1 0417                            STORES DATA FROM THE DIGITAL PANEL PRINTER
00479.000   553 1 0417                            (DPP) OF THE DUAL PLETHYSMOGRAPH.  SORTS
00480.000   554 1 0417                            FOR PRINT TESTS, INVALID READINGS, ETC.
00481.000   555 1 0417                            THE TIMERS OF ACTIVE CHANNELS ARE READ
00482.000   556 1 0417                            AND RESET AFTER EACH INDEX.           >>
00483.000   557 1 0417
00484.000   558 1 0417 DBE4                       A=INPUT(PORT1);
00485.000   559 1 0419 EEFF                       A=A XOR #FF;    ! INVERT
00486.000   560 1 041B 4F                         C=A;
00487.000   561 1 041C DBE5                       A=INPUT(PORT2);
00488.000   562 1 041E 47                         B=A;
00489.000   563 1 041F DBE6                       A=INPUT(PORT3);
00490.000   564 1 0421 57                         D=A;
00491.000   565 1 0422                            <<CHECK FOR PRINT TEST>>
00492.000   566 1 0422 E63F                       A=A AND #3F;    ! MASK OFF 2 MSB
00493.000   567 1 0424 FE08                       A-8;
00494.000   568 1 0426                            IF ZERO
00495.000   569 1 0426 C23C04F                      THEN BEGIN
00496.000   570 2 0429 78                           A=B;
00497.000   571 2 042A E69F                         A=A AND #9F;  ! MASK OFF NEXT 2 DIGITS
00498.000   572 2 042C FE88                         A-#88;
00499.000   573 2 042E                              IF ZERO
00500.000   574 2 042E C23C04F                         THEN BEGIN
00501.000   575 3 0431 79                              A=C;
00502.000   576 3 0432 FE88                            A-#88;
00503.000   577 3 0434                                 IF ZERO
00504.000   578 3 0434 C23C04F                            THEN BEGIN  ! PRINT TEST
00505.000   579 4 0437 320098                              MEM(DPP'PRINT'INTR)=A;
00506.000   580 4 043A FB                                 ENABLE INTERRUPT;
00507.000   581 4 043B C9                                 RETURN;
00508.000   582 3 043C                                  END;
00509.000   583 2 043C                              END;
00510.000   584 1 043C                         END;
00511.000   585 1 043C                         << IF INDEX READING THEN READ TIME >>
00512.000   586 1 043C 78                        A=B;
00513.000   587 1 043D E640                      A=A AND NEG'TRU'POLARY;
00514.000   588 1 043F                         IF NONZERO
00515.000   589 1 043F CA1304F                    THEN BEGIN    ! READ & STORE TIME INFO
00516.000   590 2 0442 3A0700G                      A=CH1'ACT;
00517.000   591 2 0445 E60F                        A=A AND #0F; ! MASK OFF # OF RDGS
```

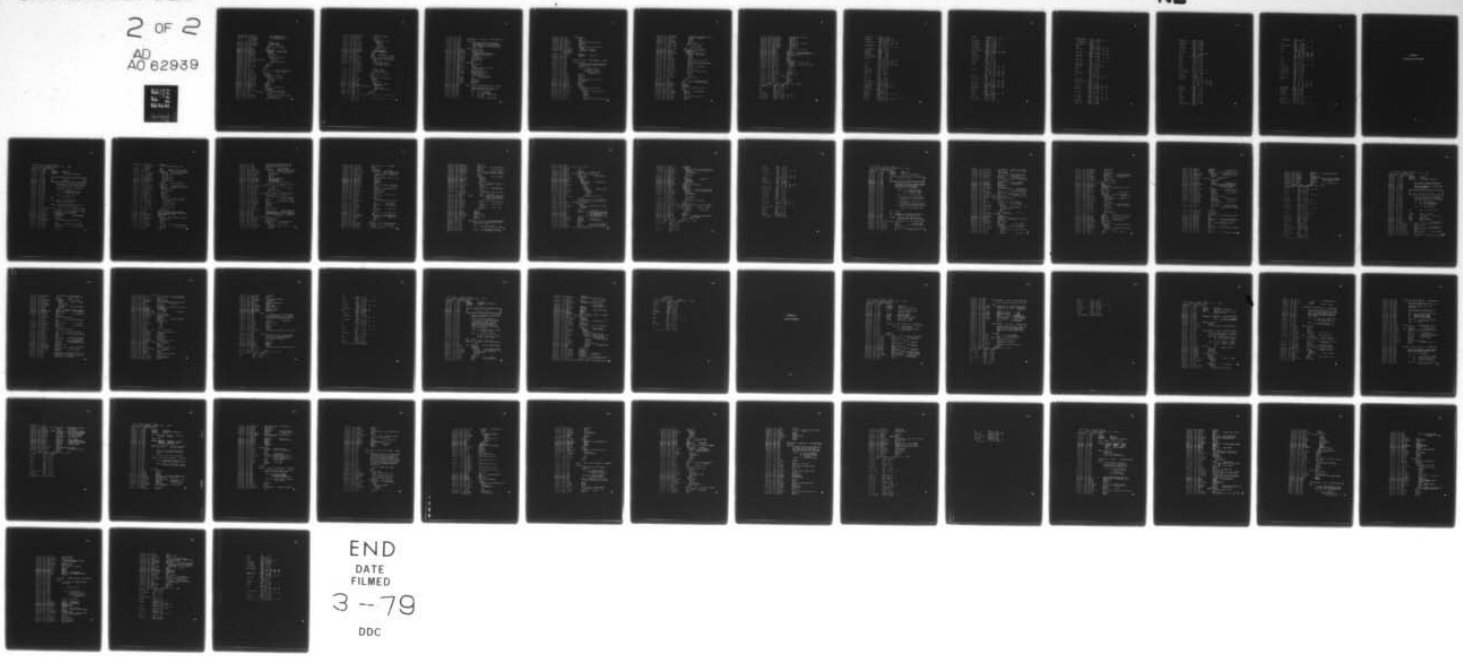
```
00518.000  592 2 0447 FE02              A=2;
00519.000  593 2 0449                   IF ZERO
00520.000  594 2 0449 C26F04F             THEN BEGIN  ! READ CH1 TIMER
00521.000  595 3 044C DBE8                  A=INPUT(PORT4);
00522.000  596 3 044E 5F                    E=A;
00523.000  597 3 044F DBE9                  A=INPUT(PORT5);
00524.000  598 3 0451 E60F                  A=A AND #0F; ! MASK OFF 4 LSB
00525.000  599 3 0453 57                    D=A;
00526.000  600 3 0454 211E00G               HL=@SUM;
00527.000  601 3 0457 CD0000X               CALL BIN TO BCD; !HL=@SUM
00528.000  602 3 045A 0603                  B=3;
00529.000  603 3 045C EB                    DE<==>HL;
00530.000  604 3 045D 21EF00G               HL=@FP NUM;
00531.000  605 3 0460 CD0000X               CALL INT TO FP; !HL=@FP NUM
00532.000  606 3 0463 016F00G               BC=@SUM TIME1;
00533.000  607 3 0466 5059                  D=B;E=C;
00534.000  608 3 0468 CD0000X               CALL FP ADD;
00535.000  609 3 046B 218100G               HL=@NUM TIME1;
00536.000  610 3 046E 34                    MEM(HL)=MEM(HL)+1;
00537.000  611 2 046F                     END;
00538.000  612 2 046F 3A0300G             A=CH2 ACT;
00539.000  613 2 0472 E60F                A=A AND #0F;
00540.000  614 2 0474 FE02                A-2;
00541.000  615 2 0476                     IF ZERO
00542.000  616 2 0476 C2A004F               THEN BEGIN !READ CH2 TIMER
00543.000  617 3 0479 DBE9                    A=INPUT(PORT5);
00544.000  618 3 047B E6F0                    A=A AND #F0;
00545.000  619 3 047D 0F0F0F                  RRC;RRC;RRC;RRC;
                   3 0480 0F
00545.000  620 3 0481 57                      D=A;
00547.000  621 3 0482 DBEA                    A=INPUT(PORT6);
00548.000  622 3 0484 5F                      E=A;
00549.000  623 3 0485 211E00G                 HL=@SUM;
00550.000  624 3 0488 CD0000X                 CALL BIN TO BCD; !HL=@SUM
00551.000  625 3 048B 0603                    B=3;
00552.000  626 3 048D EB                      DE<==>HL;
00553.000  627 3 048E 21EF00G                 HL=@FP NUM;
00554.000  628 3 0491 CD0000X                 CALL INT TO FP; !HL=@FP NUM
00555.000  629 3 0494 017800G                 BC=@SUM TIME2;
00556.000  630 3 0497 5059                    D=B;E=C;
00557.000  631 3 0499 CD0000X                 CALL FP ADD;
00558.000  632 3 049C 218200G                 HL=@NUM TIME2;
00559.000  633 3 049F 34                      MEM(HL)=MEM(HL)+1;
00560.000  634 2 04A0                       END;
00562.000  635 2 04A0 3E00                  A=0;
00562.100  636 2 04A2 CD0000                CALL PRINT DELAY;
00563.000  637 2 04A5 320700G               CH1 ACT=A;
00564.000  638 2 04A8 320800G               CH2 ACT=A;
00565.000  639 2 04AB 320A00                MEM(PIT TIMER BUSY)=A;
00566.000  640 2 04AE 320000  PTN:          MEM(DPP PRINT INTR)=A;
00567.000  641 2 04B1 FB                    ENABLE INTERRUPT;
00568.000  642 2 04B2 C9                    RETURN;
00569.000  643 1 04B3                   END;
```

```
00570.000   644 1 04B3                  << STORE INPUT DATA.  IF 2D RDG ON A CHAN
00571.000   645 1 04B3                  & NEITHER READING WAS OVERRANGE, THEN FIND
00572.000   646 1 04B3                  THE DIFFERENCE AND STORE IN TABLE FOR
00573.000   647 1 04B3                  LATER CALCULATION OF BLOOD FLOW.  NEGATIVE
00574.000   648 1 04B3                  DIFFERENCES ARE NOT RETAINED.  INPUT DATA
00575.000   649 1 04B3                  IS IN REG D,B, & C.          .       >>
00576.000   650 1 04B3 7A               A=D;
00577.000   651 1 04B4 E650             A=A AND %01010000; ! MASK OFF OVERLOAD 1 &
00578.000   652 1 04B6                                    !  CHANNEL 1 IDENT
00579.000   653 1 04B6 FE10             A-%00010000;
00580.000   654 1 04B8                  IF POSITIVE
00581.000   655 1 04B8 FA3805P            THEN BEGIN    ! CHANNNEL 1
00582.000   656 2 04BB                     IF NONZERO
00583.000   657 2 04BB CAC604P               THEN BEGIN   ! OVERLOAD 1
00584.000   658 3 04BE 3A0700G                 A=CH1'ACT;
00585.000   659 3 04C1 F6F0                   A=A OR #F0;
00586.000   660 3 04C3 3207000G               CH1'ACT=A;
00587.000   661 2 04C6                     END;
00588.000   662 2 04C6 3A0700G            A=CH1'ACT;
00589.000   663 2 04C9 3C                 A=A+1;
00590.000   664 2 04CA 3207000G           CH1'ACT=A;
00591.000   665 2 04CD 5F                 E=A;         ! SAVE CH1'ACT
00592.000   666 2 04CE E601               A=A AND #01;
00593.000   667 2 04D0                    IF NONZERO
00594.000   668 2 04D0 CAE204P              THEN BEGIN !1ST RDG, STORE IT
00595.000   669 3 04D3           FIRST'RDG1:     << CHECK IF 2ND CAL RDG >>
00596.000   670 3 04D3 7A                    A=D;
00597.000   671 3 04D4 E603                  A=A AND %00000011; !MASK CAL ID
00598.000   672 3 04D6 C2E005P               JP NONZERO THEN GOTO CHAN1'RESET;
00599.000   673 3 04D9 210400G               HL=@DPP'BUFF1A;
00600.000   674 3 04DC CD0806P               CALL SAVE'DPP'DATA;
00601.000   675 3 04DF                     END
00602.000   676 3 04DF C3F804P              ELSE BEGIN  !2ND RDG
00603.000   677 3 04E2 7A                    A=D;
00604.000   678 3 04E3 E60F                  A=A AND #0F;
00605.000   679 3 04E5 FE08                  A-8;
00606.000   680 3 04E7                      IF ZERO ! 1ST CAL RDG
00607.000   681 3 04E7 C2F504P                 THEN BEGIN
00608.000   682 4 04EA 1D                       E=E-1;
00609.000   683 4 04EB 3A0700G                  A=CH1'ACT;
00610.000   684 4 04EE 3D                       A=A-1;
00611.000   685 4 04EF 3207000G                 CH1'ACT=A;
00612.000   686 4 04F2 C3D304P                  GOTO FIRST'RDG1;
00613.000   687 3 04F5                       END;
00614.000   688 3 04F5 210C00G               HL=@DPP'BUFF1B;
00615.000   689 3 04F8 CD0806P               CALL SAVE'DPP'DATA;
00616.000   690 2 04FB                     END;
00617.000   691 2 04FB 7B                 A=E;         !CH1'ACT
00618.000   692 2 04FC D5                 TOS=DE;      !SAVE CHAN & ID INFO
00619.000   693 2 04FD FE02               A-2;
00620.000   694 2 04FF                    IF ZERO  ! 2ND RDG HAS BEEN INPUT
00621.000   695 2 04FF C21D05P              THEN BEGIN  ! & NEITHER OVERLOAD
00622.000   696 3 0502 7A                    A=D;
00623.000   697 3 0503 E60F                  A=A AND #0F;
```

AD-A062 939 ARMY MILITARY PERSONNEL CENTER ALEXANDRIA VA F/G 6/5
MICROPROCESSOR CONTROL OF THE BLOOD FLOW PLETHYSMOGRAPH.(U)
DEC 78 W E SEIBERLING

UNCLASSIFIED NL

2 OF 2
AD
A062939



END
DATE
FILMED
3-79
DDC

```
00624.000  698 3 0505 FE00                    A-0;   ! RDG VS CAL?
00625.000  699 3 0507 C21005F                 IF ZERO THEN HL=@NUM'RDG'1
00626.000  700 3 050A 21F800G                    ELSE HL=@NUM'CAL'1;
           3 050D C31305F
           3 0510 2126026
00627.000  701 3 0513 7A                      A=D;
00628.000  702 3 0514 0109006                 BC=@DPP'BUFF1A;
00629.000  703 3 0517 110C006                 DE=@DPP'BUFF1B;
00630.000  704 3 051A CD1F06F                 CALL CALC'STORE'DIFF;
00630.100  705 2 051D                       END;
00631.000  706 2 051D C1                    BC=TOS; !B= CHAN & ID INFO
00632.000  707 2 051E 78                    A=B;
00633.000  708 2 051F E60F                  A=A AND #0F;
00634.000  709 2 0521                       IF NONZERO   ! CALIBRATION RDG
00635.000  710 2 0521 CA3405F                 THEN BEGIN
00636.000  711 3 0524 79                      A=C;
00637.000  712 3 0525 E60F                    A=A AND #0F;
00638.000  713 3 0527 FE02                    A-2;
00639.000  714 3 0529                         IF ZERO  ! 2ND CAL RDG
00640.000  715 3 0529 C23405F                   THEN BEGIN
00641.000  716 4 052C 3E00     CHAN1 RESET:       A=0;
00642.000  717 4 052E 3207006                     CH1'ACT=A;
00643.000  718 4 0531 3200A0                      MEM'PIT'TIMER'BUCK=A;
00644.000  719 3 0534                           END;
00645.000  720 2 0534                       END;
00646.000  721 2 0534 C3AE04F               GOTO RTN;
00647.000  722 2 0537 00                    NOP;
00649.000  723 1 0538                     END;
00650.000  724 1 0538 7A                  A=D;
00651.000  725 1 0539 E6A0                A=A AND '10100000; ! MASK OFF OVERLOAD 2 &
00652.000  726 1 053B                                        !  CHANNEL 2 IDENT
00653.000  727 1 053B FE20                A-'00100000;
00654.000  728 1 053D                     IF POSITIVE
00655.000  729 1 053D FABC05F               THEN BEGIN    ! CHANNNEL 2
00656.000  730 2 0540                        IF NONZERO
00657.000  731 2 0540 CA4B05F                  THEN BEGIN   ! OVERLOAD 2
00658.000  732 3 0543 3A0800G                    A=CH2'ACT;
00659.000  733 3 0546 F6F0                       A=A OR #F0;
00660.000  734 3 0548 3208006                    CH2'ACT=A;
00661.000  735 2 054B                          END;
00662.000  736 2 054B 3A0800G                 A=CH2'ACT;
00663.000  737 2 054E 3C                      A=A+1;
00664.000  738 2 054F 3208006                 CH2'ACT=A;
00665.000  739 2 0552 5F                      E=A;         ! SAVE CH2'ACT
00666.000  740 2 0553 E601                    A=A AND #01;
00667.000  741 2 0555                         IF NONZERO
00668.000  742 2 0555 CA6705F                   THEN BEGIN !1ST RDG, STORE IT
00669.000  743 3 0558          FIRST'RDG2:        << CHECK IF 2ND CAL RDG >>
00670.000  744 3 0558 7A                        A=D;
00671.000  745 3 0559 E603                      A=A AND '00000011; !MASK CAL ID
00672.000  746 3 055B C2B105F                   IF NONZERO THEN GOTO CHAN2 RESET;
00673.000  747 3 055E 210F006                   HL=@DPP BUFF2A;
00674.000  748 3 0561 CD0306F                   CALL SAVE DPP DATA;
00675.000  749 3 0564                         END
```

```
00676.000   750 2 0564 C38005P                    ELSE BEGIN  !2ND RDG
00677.000   751 3 0567 7A                           A=D;
00678.000   752 3 0568 E60F                         A=A AND #0F;
00679.000   753 3 056A FE08                         A-8;
00680.000   754 3 056C                              IF ZERO ! 1ST CAL RDG
00681.000   755 3 056C C27A05P                       THEN BEGIN
00682.000   756 4 056F 1D                             E=E-1;
00683.000   757 4 0570 3A0800G                       A=CH2'ACT;
00684.000   758 4 0573 3D                           A=A-1;
00685.000   759 4 0574 320800G                      CH2'ACT=A;
00686.000   760 4 0577 C3580SP                      GOTO FIRST'RDG2;
00687.000   761 3 057A                             END;
00688.000   762 3 057A 2112006                     HL=@DPP'BUFF2B;
00689.000   763 3 057D CD0806P                     CALL SAVE'DPP'DATA;
00690.000   764 2 0580                           END;
00691.000   765 2 0580 7B                        A=E;        !CH2'ACT
00692.000   766 2 0581 D5                        TOI=DE;     !SAVE CHAN & ID INFO
00693.000   767 2 0582 FE02                      A-2;
00694.000   768 2 0584                         IF ZERO  ! 2ND RDG HAS BEEN INPUT
00695.000   769 2 0584 C2A205P                   THEN BEGIN  ! & NEITHER OVERLOAD
00696.000   770 3 0587 7A                          A=D;
00697.000   771 3 0588 E60F                        A=A AND #0F;
00698.000   772 3 058A FE00                        A-0;    ! RDG VS CAL?
00699.000   773 3 058C C2950SP                     IF ZERO THEN HL=@NUM'RDG'2
00700.000   774 3 058F 218F01G                        ELSE HL=@NUM'CAL'2;
            3 0592 C39805P
            3 0595 218A02G
00701.000   775 3 0598 7A                          A=D;
00702.000   776 3 0599 010F00G                     BC=@DPP'BUFF2A;
00703.000   777 3 059C 1112006                     DE=@DPP'BUFF2B;
00704.000   778 3 059F CD1F0eP                     CALL CALC'STORE'DIFF;
00705.000   779 2 05A2                          END;
00706.000   780 2 05A2 D1                      BC=TOI; !B= CHAN & ID INFO
00707.000   781 2 05A3 78                      A=B;
00708.000   782 2 05A4 E60F                    A=A AND #0F;
00709.000   783 2 05A6                        IF NONZERO   ! CALIBRATION RDG
00710.000   784 2 05A6 CAE905P                   THEN BEGIN
00711.000   785 3 05A9 79                        A=C;
00712.000   786 3 05AA E60F                      A=A AND #0F;
00713.000   787 3 05AC FE02                      A-2;
00714.000   788 3 05AE                          IF ZERO  ! 2ND CAL RDG
00715.000   789 3 05AE C2B905P                     THEN BEGIN
00716.000   790 4 05B1 3E00     CHAN2'RESET:        A=0;
00717.000   791 4 05B3 320800G                     CH2'ACT=A;
00718.000   792 4 05B6 3200A0                      MEM'BIT TIMER'BUCS:=A;
00719.000   793 3 05B9                           END;
00720.000   794 2 05B9                        END;
00721.000   795 2 05B9 C3AE04P                GOTO RTN;
00724.000   796 1 05BC                      END;
00725.000   797 1 05BC 030000   ONE TWENTY:      DB #03,0,0,0,0,0,0,0,#12; ! 120.0
            1 05BF 000000
            1 05C2 000012
00726.000   798 1 05C5 020000   SIXTY:          DB #02,0,0,0,0,0,0,0,#60; !  60.0
            1 05C8 000000
            1 05CB 000060
```

```
00797.000  799 1 05CE          <<
00797.100  800 1 05CE          ••••••••••••••••     INITIALIZE     ••••••••••••••••••
00797.200  801 1 05CE                                                              >>
00798.000  802 1 05CE          INITIALIZE:
00799.000  803 1 05CE                 << PROVIDES A DELAY TO ALLOW FOR THE
00800.000  804 1 05CE                 PLETHYSMOGRAPH TURN ON, CLEARS MEMORY,
00801.000  805 1 05CE                 CLEARS INTERRUPTS, & RESETS THE PLETHYSMO-
00802.000  806 1 05CE                 GRAPH.  A PRINT TEST IS OUTPUT WHEN THE
00802.100  807 1 05CE                 SYSTEM IS READY.                        >>
00803.000  808 1 05CE          ON'DELAY:
00804.000  809 1 05CE 2E28           L=TURN'ON'DELAY;
00805.000  810 1 05D0 CD0000X        CALL PRINT'DELAY;
00806.000  811 1 05D3 2D             L=L-1;
00807.000  812 1 05D4 C2D005P        IF NONZERO THEN GOTO ON'DELAY(2);
00808.000  813 1 05D7                << CLEAR RAM, EXCEPT STACK >>
00809.000  814 1 05D7 21003C         HL=#3C00;
00810.000  815 1 05DA 066A           B=STACK - #3C00 - 5;
00811.000  816 1 05DC CD0000X        CALL ZERO MEM;
00812.000  817 1 05DF 21703C         HL=STACK+1;
00813.000  818 1 05E2 0690           B=#3CFF - STACK;
00814.000  819 1 05E4 1604           D=4;
00815.000  820 1 05E6          MORE'ZERO:
00816.000  821 1 05E6 CD0000X        CALL ZERO MEM;
00817.000  822 1 05E9 05             B=B-1;
00818.000  823 1 05EA 15             D=D-1;
00819.000  824 1 05EB C2E605P        IF NONZERO THEN GOTO MORE'ZERO;
00820.000  825 1 05EE                << CLEAR SYSTEM >>
00821.000  826 1 05EE 3200E8         MEM<SYSID INTR>=A;
00822.000  827 1 05F1 3200A0         MEM<PIT'TIMER BUF>=A;
00823.000  828 1 05F4 320098         MEM<DPP'PRINT'INTR>=A;
00824.000  829 1 05F7 320090         MEM<RESET'DP>=A;
00825.000  830 1 05FA                << OUTPUT PRINT TEST >>
00826.000  831 1 05FA                B=POLSIGN OF POL'TRU'POLARY
00826.100  832 1 05FA 063F                   OR ALL'DEC PTS;
00827.000  833 1 05FC 0E88           C=#88;
00828.000  834 1 05FE 118888         DE=#8888;
00829.000  835 1 0601 CD0000X        CALL PRINTOUT;
00830.000  836 1 0604 CD0000X        CALL SPACE;
00831.000  837 1 0607 C9             RETURN;
00832.000  838 1 0608                <<
00832.100  839 1 0608          •••••   SAVE DIGITIAL PANEL PRINTER DATA   •••••••
00832.200  840 1 0608
00833.000  841 1 0608                PROGRAM STORES INPUT DATA INTO A 3-BYTE
00834.000  842 1 0608                STORAGE BUFF AWAITING FURTHER CALC.
00835.000  843 1 0608
00836.000  844 1 0608                INPUTS     BC  4 LSD FROM DPP
00837.000  845 1 0608                           D   2 MSD FROM DPP
00838.000  846 1 0608                           HL  ADDR OF STORAGE BUFFER
00839.000  847 1 0608                OUTPUT     BC  SAME
00840.000  848 1 0608                           DE  SAME
00841.000  849 1 0608                           HL  & TOP OF STORAGE BUFFER  >>
```

```
00772.000  850 1 0608          SAVE'DPP'DATA:
00773.000  851 1 0608 79               A=C;
00774.000  852 1 0609 77               MEM(HL)=A;
00775.000  853 1 060A 78               A=B;
00776.000  854 1 060B E69F             A=A AND #9F;  !MASK OFF 2 DIGITS
00777.000  855 1 060D 23               HL=HL+1;
00778.000  856 1 060E 77               MEM(HL)=A;
00779.000  857 1 060F 78               A=B;
00780.000  858 1 0610 E620             A=A AND %00100000; !MASK OFF SIGN
00781.000  859 1 0612                  IF ZERO
00782.000  860 1 0612 C21A06P            THEN A=#FF  ! NEGATIVE
00783.000  861 1 0615 3EFF               ELSE A=0;   ! POSITIVE
           1 0617 C31C06P
           1 061A 3E00
00784.000  862 1 061C 23               HL=HL+1;
00785.000  863 1 061D 77               MEM(HL)=A;    ! STORE SIGN
00786.000  864 1 061E C9               RETURN;
00787.000  865 1 061F
00787.100  866 1 061F          <<
00787.200  867 1 061F          ******    CALCULATE & STORE DIFFERENCE    *******
00787.300  868 1 061F                                                        >>
00788.000  869 1 061F          CALC'STORE'DIFF:
00789.000  870 1 061F                  << CALCULATES THE DIFFERENCE BETWEEN THE
00790.000  871 1 061F                  INPUT READINGS AND STORES THE POSITIVE
00791.000  872 1 061F                  DIFFERENCES ONLY.
00792.000  873 1 061F
00793.000  874 1 061F                  INPUT   A   CHANNEL ID INFO
00794.000  875 1 061F                          BC  POINTER TO 1ST RDG
00795.000  876 1 061F                          DE  POINTER TO 2ND RDG
00796.000  877 1 061F                          HL  POINTER TO START OF
00797.000  878 1 061F                              DIFFERENCE LIST    >>
00798.000  879 1 061F E5               TOS=HL;
00799.000  880 1 0620 F5               TOS=AFLAGS;
00800.000  881 1 0621 C5               TOS=BC;
00801.000  882 1 0622 D5               TOS=DE;
00802.000  883 1 0623 0303             BC=BC+1;BC=BC+1;  !POINT TO SIGN
00803.000  884 1 0625 1313             DE=DE+1;DE=DE+1;
00804.000  885 1 0627 EB               HL<==>DE;
00805.000  886 1 0628 0A               A=MEM(BC);
00806.000  887 1 0629 57               D=A;
00807.000  888 1 062A FE00             A-0;
00808.000  889 1 062C                  IF ZERO
00809.000  890 1 062C C25306P            THEN BEGIN  ! 1ST RDG POSITIVE
00810.000  891 2 062F B6                   A=A OR MEM(HL);
00811.000  892 2 0630                      IF ZERO
00812.000  893 2 0630 C24D06P                THEN BEGIN  ! 2ND RDG POSITIVE
00813.000  894 3 0633 E1                       HL=TOS;
00814.000  895 3 0634 D1                       DE=TOS;
00815.000  896 3 0635 D5                       TOS=DE;
00816.000  897 3 0636 E5                       TOS=HL;
00817.000  898 3 0637 0602                     B=2;
00818.000  899 3 0639 CD0000                   CALL LARGER; !B=1 IF 2ND 1ST RDG
00819.000  900 3 063C 05                       B=B-1;
```

```
00820.000  901 3 063D C2DD06P              IF NONZERO THEN GOTO POP'AND'RET;
00821.000  902 3 0640 011E00G              BC=@SUM;  ! DIFFERENCE
00822.000  903 3 0643 E1                   HL=TOS;
00823.000  904 3 0644 D1                   DE=TOS;
00824.000  905 3 0645 3E02                 A=2;
00825.000  906 3 0647 CD0000X              CALL DEC'SUB;
00826.000  907 3 064A                    END
00827.000  908 2 064A C35006P            ELSE GOTO POP'AND'RET;
           2 064D C3DD06P
00828.000  909 2 0650                      <<2ND RDG NEGATIVE >>
00829.000  910 2 0650                  END
00830.000  911 1 0650 C37B06P          ELSE BEGIN  ! 1ST RDG NEGATIVE
00831.000  912 2 0653 AE                 A=A XOR MEM(HL);
00832.000  913 2 0654                    IF ZERO
00833.000  914 2 0654 C27106P              THEN BEGIN ! 2ND RDG NEG
00834.000  915 3 0657 D1                     DE=TOS;  ! CHECK REL MAG
00835.000  916 3 0658 E1                     HL=TOS;
00836.000  917 3 0659 E5                     TOS=HL;
00837.000  918 3 065A D5                     TOS=DE;
00838.000  919 3 065B 0602                   B=2;
00839.000  920 3 065D CD0000X                CALL LARGEF; !B=1 IF 2ND 1ST
00840.000  921 3 0660 05                     B=B-1;
00841.000  922 3 0661 C2DD06P                IF NONZERO THEN GOTO POP'AND'RET;
00842.000  923 3 0664 D1                     DE=TOS;
00843.000  924 3 0665 E1                     HL=TOS;
00844.000  925 3 0666 011E00G                BC=@SUM;  ! @ DIFFERENCE
00845.000  926 3 0669 3E02                   A=2;
00846.000  927 3 066B CD0000X                CALL DEC'SUB;
00847.000  928 3 066E                      END
00848.000  929 2 066E C37B06P              ELSE BEGIN  ! 2ND RDG POSITIVE
00849.000  930 3 0671 011E00G                BC=@SUM;
00849.100  931 3 0674 D1                     DE=TOS;
00850.000  932 3 0675 E1                     HL=TOS;
00851.000  933 3 0676 3E02                   A=2;
00852.000  934 3 0678 CD0000X                CALL DEC'ADD;
00853.000  935 2 067B                      END;
00854.000  936 1 067B                  END;
00855.000  937 1 067B                << IF THIS IS A 0.2% CALIBRATION RDG.
00856.000  938 1 067B                MULTIPLY DIFF BY 5 PRIOR TO STORING.>>
00857.000  939 1 067B 3E00           A=0;
00858.000  940 1 067D 3220006        SUM(2)=A;
00859.000  941 1 0680 F1             AFLAGS=TOS;
00860.000  942 1 0681 F5             TOS=AFLAGS;
00861.000  943 1 0682 E60F           A=A AND #0F;  ! MASK OFF ID INFO
00862.000  944 1 0684 FE02           A-2;
00863.000  945 1 0686                IF ZERO
00864.000  946 1 0686 C2B306P          THEN BEGIN  ! .2% CALIBRATION
00865.000  947 2 0689 111E00G            DE=@SUM;
00866.000  948 2 068C 212100G            HL=@SUM(3);
00867.000  949 2 068F 0603               B=3;
00868.000  950 2 0691 CD0000X            CALL TRANS'DATA; !DE=@SUM(3)
00869.000  951 2 0694 211E00G            HL=@SUM;
00870.000  952 2 0697 012400G            BC=@SUM(6);
00871.000  953 2 069A 3E03               A=3;
```

```
00872.000  954 2 069C CD0000X             CALL DEC'ADD;  !2X @SUM(6)
00873.000  955 2 069F 2124006             HL=@SUM(6);
00874.000  956 2 06A2 111E006             DE=@SUM;
00875.000  957 2 06A5 0121006             BC=@SUM(3);
00876.000  958 2 06A8 3E03                A=3;
00877.000  959 2 06AA CD0000X             CALL DEC'ADD;  !3X @SUM(3)
00878.000  960 2 06AD 011E006             BC=@SUM;
00879.000  961 2 06B0 2121006             HL=@SUM(3);
00880.000  962 2 06B3 1124006             DE=@SUM(6);
00881.000  963 2 06B6 3E03                A=3;
00882.000  964 2 06B8 CD0000X             CALL DEC'ADD;  !5X @SUM
00883.000  965 1 06BB                   END;
00884.000  966 1 06BB E1                HL=TOS;
00885.000  967 1 06BC E3                HL<=>TOS;  ! START OF DIFFERENCE LIST
00886.000  968 1 06BD 7E                A=MEM(HL);  ! NUM RDGS IN LIST
00887.000  969 1 06BE 34                MEM(HL)=MEM(HL)+1;
00888.000  970 1 06BF 06004F            B=0;C=A;
00889.000  971 1 06C2 81                A=A+C;
00890.000  972 1 06C3 3C                A=A+1;
00891.000  973 1 06C4 5F1600            E=A;D=0;
00892.000  974 1 06C7 F1                AFLAGS=TOS;
00893.000  975 1 06C8 E60F              A=A AND #0F; ! MASK OFF ID INFO
00894.000  976 1 06CA                   IF NONZERO
00895.000  977 1 06CA CAD306F             THEN BEGIN  ! CAL INFO(3 BYTES/NUM)
00896.000  978 2 06CD 09                    HL=HL+BC;
00897.000  979 2 06CE 0603                  B=3;
00898.000  980 2 06D0                     END
00899.000  981 1 06D0 C3D506F             ELSE B=2;
                1 06D3 0602
00900.000  982 1 06D5 19                HL=HL+DE;
00901.000  983 1 06D6 111E006           DE=@SUM;
00902.000  984 1 06D9 CD0000X           CALL TRANS'DATA;
00903.000  985 1 06DC C9                RETURN;
00904.000  986 1 06DD             POP' AND' RET:
00905.000  987 1 06DD 210800            HL=8;
00906.000  988 1 06E0 39                HL=HL+SP;
00907.000  989 1 06E1 F9                SP=HL;
00908.000  990 1 06E2 C9                RETURN;
00909.000  991 0 06E3                   END.
           ***** 0 ERROR(S), 0 WARNING(S). SEE 0 *****
  BSAL-80/85.2.3  78/11/27  13:10:01
ALL'DEC'PTS       EQUATE    #003F
                    55   110   303   832
BF1               LABEL     #01DD
                   285   279
BF2               LABEL     #0358
                   443   437
BIN TO BCD        EXTERNAL  #0014
                    15   198   356   601   624
BLOOD FLOW1       GLOBAL    #00B9
                    34   146   231   285   298
BLOOD FLOW2       GLOBAL    #00C2
                    34   389   443
```

```
BUFF_COUNT          GLOBAL    #0006
                      20   103   132   472
BUFFER_IN           EXTERNAL  #001A
                      17   487   499   520   533
BYTES_PER_CAL       EQUATE    #0003
                      71   170   174   186   328
                     332   344
BYTES_PER_RDG       EQUATE    #0002
                      70   160   164   178   318
                     322   336
CAL_BLD_FLOW        EQUATE    #0002
                      59    99
CAL_DIFF_1          GLOBAL    #0227
                      40
CAL_DIFF_2          GLOBAL    #028B
                      41
CALC_STORE_DIFF     LABEL     #061F
                     869   704   778
CH1_ACT             GLOBAL    #0007
                      20   590   637   658   660
                     662   664   683   685   717
CH2_ACT             GLOBAL    #0008
                      21   612   638   732   734
                     736   738   757   759   791
CHAN_1              EQUATE    #0005
                      56   478
CHAN_2              EQUATE    #0004
                      57   511
CHAN1_RESET         LABEL     #052C
                     716   672
CHAN2_RESET         LABEL     #05B1
                     790   746
CHAN_1              LABEL     #0093
                     152
CHAN_2              LABEL     #01EE
                     291   148
CIRC_LIMB           EQUATE    #0001
                      60   494   528
CIRCUM_LIMB1        GLOBAL    #00A7
                      33   223   244   498
CIRCUM_LIMB2        GLOBAL    #00B0
                      33   381   402   532
CLEAR_MEM           LABEL     #0369
                     449   309
CLR_IN_BUFF         LABEL     #0065
                     115   548
COMPARE             EXTERNAL  #0019
                      16   165   175   323   333
COMPUTE_BF          LABEL     #0083
                     138   100
DEC_ADD             EXTERNAL  #001E
                      17   934   954   959   964
DEC_SUB             EXTERNAL  #001C
                      17   906   927
```

```
DIVIDE              EXTERNAL  #000B
                      12    208    211    232    366
                     369    390
DPP'BUFF1A          GLOBAL    #0009
                      21    197    673    702
DPP'BUFF1B          GLOBAL    #000C
                      22    688    703
DPP'BUFF2A          GLOBAL    #000F
                      22    355    747    776
DPP'BUFF2B          GLOBAL    #0012
                      23    762    777
DPP'PRINT'INTR      EQUATE    #9800
                      48    461    579    640    828
ENTER               EQUATE    #0003
                      58     97
ERROR'OUT           EXTERNAL  #0011
                      14    306
FIRST'RIG1          LABEL     #04D3
                     669    686
FIRST'RIG2          LABEL     #0558
                     743    760
FP'ADD              EXTERNAL  #000A
                      12    608    631
FP'NUM              GLOBAL    #00EF
                      37    183    191    203    213
                     221    341    349    361    371
                     379    604    627
FP'OUT'FMT          EXTERNAL  #0017
                      16    236    241    245    252
                     256    263    270    274    281
                     286    394    399    403    410
                     414    421    428    432    439
                     444    488    501    522    535
INITIALIZE          LABEL     #050E
                     802     79
INPUT'BUFF          GLOBAL    #0000
                      20    116    129    153    259
                     277    311    417    435    476
INPUT'ERROR         LABEL     #0058
                     107    475    506    540    544
INT'TO'FP           EXTERNAL  #0018
                      16    204    362    605    628
INTERRUPTION        LABEL     #0038
                      87
KYBD'IN             EQUATE    #7800
                      50     90
KYBD'INTR           EQUATE    #B800
                      44    120    134    458    826
LARGEP              EXTERNAL  #000F
                      13    899    920
LENGTH'GAUGE        EQUATE    #0000
                      61    483    516
LENGTH'GAUGE1       GLOBAL    #0035
                      32    215    240    486
```

```
LENGTH_GAUGE2       GLOBAL    #009E
                      32   373   398   519
MAX_CALIBRATION     EQUATE    #0021
                      69
MAX_READING         EQUATE    #0032
                      68
MEAN                EXTERNAL  #0016
                      15   181   189   339   347
MEAN_CAL1           GLOBAL    #005D
                      27   169   172   187   222
                     255
MEAN_CAL2           GLOBAL    #0066
                      28   327   330   345   380
                     413
MEAN_RDG1           GLOBAL    #004B
                      26   159   162   179   214
                     273
MEAN_RDG2           GLOBAL    #0054
                      27   317   320   337   372
                     431
MEAN_TIME1          GLOBAL    #0083
                      30   209   226   235
MEAN_TIME2          GLOBAL    #008C
                      31   367   384   393
MEAN1               LABEL     #00CD
                     173   155
MEAN2               LABEL     #0248
                     336   313
MORE_ZERO1          LABEL     #05E6
                     820   824
MULT                EXTERNAL  #000C
                      12   216   219   224   227
                     374   377   382   385
NEG_SIGN            EQUATE    #0000
                      51
NEG_TRU_POLARY      EQUATE    #0040
                      54   587
NUM_CAL_1           GLOBAL    #028E
                      40   167   173   188   700
NUM_CAL_2           GLOBAL    #028A
                      41   325   331   346   774
NUM_RDG_1           GLOBAL    #00F8
                      38   141   157   163   180
                     699
NUM_RDG_2           GLOBAL    #018F
                      39   294   315   321   338
                     773
NUM_TIME1           GLOBAL    #0081
                      29   195   200   202   609
NUM_TIME2           GLOBAL    #0082
                      30   353   358   360   632
NUMBER_CAL1         GLOBAL    #00CB
                      35   192   851
NUMBER_CAL2         GLOBAL    #00DD
                      36   350   409
```

```
NUMBER_PDG1          GLOBAL    #00D4
                       35   184   269
NUMBER_PDG2          GLOBAL    #00E6
                       36   342   427
ON_DELAY             LABEL     #05CE
                      808   812
ONE_TWENTY           LABEL     #05BC
                      797   218   376
OUT_CEL              EQUATE    #B000
                       45
POP_AND_RET          LABEL     #06DD
                      986   901   908   922
PORT1                EQUATE    #00E4
                       62   558
PORT2                EQUATE    #00E5
                       63   561
PORT3                EQUATE    #00E6
                       64   563
PORT4                EQUATE    #00E8
                       65   595
PORT5                EQUATE    #00E9
                       66   597   617
PORT6                EQUATE    #00EA
                       67   621
PDI_SIGN             EQUATE    #0080
                       52   109   302   831
PDI_TRU_POLARY       EQUATE    #0000
                       53   109   302   831
PRINT_ADV            EQUATE    #A800
                       46
PRINT_DELAY          EXTERNAL  #0013
                       14   636   810
PRINTOUT             EXTERNAL  #0010
                       13   113   238   243   247
                      254   258   265   272   276
                      283   288   396   401   405
                      412   416   423   430   434
                      441   446   490   503   524
                      537   835
RDG_DIFF_1           GLOBAL    #00F9
                       38
RDG_DIFF_2           GLOBAL    #0190
                       39
PDG1                 LABEL     #01B1
                      267   261
PDG2                 LABEL     #03BC
                      425   419
READ_DPP             LABEL     #0417
                      550    93
REPEAT               LABEL     #0370
                      453   457
RESET_DF             EQUATE    #9000
                       49   460   829
RESET_VBD_INTP       LABEL     #006D
                      119
```

```
PIT^TIMER^BUSS      EQUATE    #A000
                      47   459   639   718   798
                     827
RTN                 LABEL     #04AE
                     640   721   795
SAVE^DPP^DATA       LABEL     #0608
                     850   674   689   748   763
SINTV               LABEL     #05C5
                     798   210   368
SPACE               EXTERNAL  #0012
                      14   114   234   239   248
                     266   284   289   290   307
                     392   397   406   424   442
                     447   448   491   504   525
                     538   836
STACK               EQUATE    #306F
                      43    78   450   451   815
                     817   818
START               ENTRY     #0000
                      11    75
STD^DEV             EXTERNAL  #0015
                      15   161   171   319   329
STD^DEV^CAL1        GLOBAL    #0039
                      25   168   262
STD^DEV^CAL2        GLOBAL    #0042
                      26   326   420
STD^DEV^RDG1        GLOBAL    #0087
                      24   158   280
STD^DEV^RDG2        GLOBAL    #0030
                      25   316   438
STOP                LABEL     #000F
                      83    84
STORE^BUFF          LABEL     #0386
                     485    98
SUM                 GLOBAL    #001E
                      24   207   217   230   365
                     375   383   600   623   902
                     925   930   940   947   948
                     951   952   955   956   957
                     960   961   962   983
SUM^OF^SQUARES      GLOBAL    #0015
                      23   225   383
SUM^TIME1           GLOBAL    #006F
                      28   206   606
SUM^TIME2           GLOBAL    #0078
                      29   364   629
TRANS^DATA          EXTERNAL  #0001
                      12   185   193   343   351
                     950   984
TURN^ON^DELAY       EQUATE    #0088
                      72   809
ZERO^MEM            EXTERNAL  #000E
                      13   118   454   816   821
  BIAL-80/85.2.3  78^11^27  13:16:40
```

APPENDIX D

FLOATING POINT PROGRAMS

```
 BSAL-80 85.2.3   78/11/26  12:18:57
          *****    0 ERROR(S),    0 WARNING(S). SEE     0 *****
 SOURCE FILE =    DK2:WALT/ADD.EDT-0
 OBJECT FILE =    DK2:WALT/ADD.OBJ-0
00001.000     1  1 01DE
00002.000     2  1 0000          $CONTROL          NAME=FP/ADD
00003.000     3  1 0000          !$CONTROL          NOLIST
00004.000     4  1 0000          $CONTROL          LIST,INNERLIST,TABLE,CROSS
00005.000     5  1 0000
00006.000     6  1 0000          <<
00007.000     7  1 0000          ********************************************************
00008.000     8  1 0000          **                                                    **
00010.000     9  1 0000          **                  ADDITION                          **
00011.000    10  1 0000          **                                                    **
00012.000    11  1 0000          ********************************************************
00014.000    12  1 0000
00015.000    13  1 0000                  ADDS TWO NINE BYTE FLOATING POINT NUMBERS
00015.100    14  1 0000                  FOLLOWING FORM TO GIVE A NINE BYTE RESULT.
00017.000    15  1 0000
00018.000    16  1 0000                  /DD/DD/DD/DD/DD/DD/DD/DD/SIGN,EXPONENT/
00019.000    17  1 0000
00020.000    18  1 0000                  INPUT    BC    POINTER TO SUM
00021.000    19  1 0000                           DE    POINTER TO 1ST NUMBER
00022.000    20  1 0000                           HL    POINTER TO 2D NUMBER
00023.000    21  1 0000
00024.000    22  1 0000                  RETURN   HL    POINTER TO SUM
00025.000    23  1 0000          >>
00026.000    24  1 0000          ENTRY             FP/ADD,LEFT JUST;
00027.000    25  1 0000
00028.000    26  1 0000          LOCAL    TEMP(SUM(8), M/SIG NUM PT(2),
00029.000    27  1 0000                   L/SIG NUM PT(2), EXPON DIFF,
00029.100    28  1 0000                   $EXPON  =  $EXPON DIFF;
00030.000    29  1 0000
00031.000    30  1 0000          EXTERNAL    LARGER, TRANS/DATA, TRANS/DATA/LEFT,
00032.000    31  1 0000                      ZERO MEM, DEC ADD, DEC SUB;
00033.000    32  1 0000
00034.000    33  1 0000 220B00L FP/ADD:  L/SIG NUM PT=HL; ! SAVE POINTER 2D NUMBER
00035.000    34  1 0003 EB              DE <=> HL;
00036.000    35  1 0004 220900L         M/SIG NUM PT=HL; ! SAVE POINTER 1ST NUMBER
00037.000    36  1 0007 C5              TDI=BC;          ! SUM POINTER
00038.000    37  1 0008                 << FIND THE MORE SIGNIFICANT NUMBER                >>
00039.000    38  1 0008 1A              A=MEM(DE);
00040.000    39  1 0009 E67F            A=A AND #7F;             ! MASK OFF EXPON
00041.000    40  1 000B 07              RLC;
00042.000    41  1 000C A7              A=A AND A;
00043.000    42  1 000D 37              CARRY=1;
00044.000    43  1 000E FA1200P         IF POSITIVE THEN CARRY=NOT CARRY;
                  1 0011 3F
00045.000    44  1 0012 1F              RAR;
00046.000    45  1 0013 4F              C=A;         ! EIGHT BIT EXPON 2D NUMBER
00047.000    46  1 0014 7E              A=MEM(HL);
00048.000    47  1 0015 E67F            A=A AND #7F;             ! MASK OFF EXPON
00049.000    48  1 0017 07              RLC;
00050.000    49  1 0018 A7              A=A AND A;
```

```
00051.000    50 1 0019 37              CARRY=1;
00052.000    51 1 001A FA1E00F         IF POSITIVE THEN CARRY=NOT CARRY;
                1 001D 3F
00053.000    52 1 001E 1F              RAR;
00054.000    53 1 001F 47              B=A;          ! EIGHT BIT EXPON 1CT NUMBER
00055.000    54 1 0020 91              A=A-C;        ! ORDER 1CT MINUS ORDER 2D
00056.000    55 1 0021                 IF NEGATIVE      ! EXPON 1CT # < EXPON 2D #
00057.000    56 1 0021 F23400F           THEN BEGIN     ! SWITCH POINTERS
00058.000    57 2 0024 2A0900L             HL=M.SIG.NUM.PT;
00059.000    58 2 0027 EB                  HL <== DE;
00060.000    59 2 0028 2A0B00L             HL=L.SIG.NUM.PT;
00061.000    60 2 002B 220900L             M.SIG.NUM.PT=HL;
00062.000    61 2 002E EB                  HL<==>DE;
00063.000    62 2 002F 220B00L             L.SIG.NUM.PT=HL;
00064.000    63 2 0032 79                  A=C;
00065.000    64 2 0033 90                  A=A-B;        ! DIFF IN EXPONENTS
00066.000    65 1 0034                   END;
00067.000    66 1 0034 320D00L         EXPON.DIFF=A;    ! SAVE DIFF IN EXPON
00068.000    67 1 0037                 IF ZERO          ! SAME EXPON. FIND LARGER
00069.000    68 1 0037 C25D00F           THEN BEGIN     !  MANTISSA
00070.000    69 2 003A 2A0900L             HL=M.SIG.NUM.PT;
00071.000    70 2 003D EB                  HL <== DE;
00072.000    71 2 003E 2A0B00L             HL=L.SIG.NUM.PT;
00073.000    72 2 0041 23                  HL=HL+1;
00074.000    73 2 0042 13                  DE=DE+1;
00075.000    74 2 0043 0608                B=8;
00076.000    75 2 0045 CD0000.             CALL LARGER;
00077.000    76 2 0048 05                  B=B-1;
00078.000    77 2 0049                     IF ZERO      ! LEAST SIG # > MORE SIG #
00079.000    78 2 0049 C25A00F               THEN BEGIN   ! SWITCH POINTERS
00080.000    79 3 004C 2A0900L                 HL=M.SIG.NUM.PT;
00081.000    80 3 004F EB                      HL <== DE;
00082.000    81 3 0050 2A0B00L                 HL=L.SIG.NUM.PT;
00083.000    82 3 0053 220900L                 M.SIG.NUM.PT=HL;
00084.000    83 3 0056 EB                      DE <== HL;
00085.000    84 3 0057 220B00L                 L.SIG.NUM.PT=HL;
00086.000    85 2 005A                       END;
00087.000    86 2 005A C33F00F             GOTO TRANSFER;
00088.000    87 1 005D                   END;
00089.000    88 1 005D                 IF THE DIFFERENCE IN EXPON IS .GE. 16
00089.100    89 1 005D                 DIGITS, LEAST SIGNIFICANT NUMBER APPROX 0.
00090.000    90 1 005D                 SUM = HIGHER ORDER NUMBER.
00091.000    91 1 005D 3A0D00L         A=EXPON.DIFF;
00092.000    92 1 0060 FE10            A-16;
00093.000    93 1 0062                 IF CARRY=0  ! SMALLER NUMBER INSIGNIFICANT
00094.000    94 1 0062 DA7400F           THEN BEGIN
00095.000    95 2 0065 2A0900L             HL=M.SIG.NUM.PT;
00096.000    96 2 0068 EB                  DE <== HL;
00097.000    97 2 0069 E1                  HL=TOS;
00098.000    98 2 006A 0609                B=9;
00099.000    99 2 006C CD0000              CALL TRANS.DATA; ! SUM= HIGHER ORDER #
00100.000   100 2 006F 21F7FF             HL= -9;
00101.000   101 2 0072 19                  HL=HL+DE;          ! POINTER TO SUM
00102.000   102 2 0073 C9                  RETURN;
00103.000   103 1 0074                   END;
```

```
00104.000  104 1 0074          <<       SHIFT THE LESSER NUMBER RIGHT BY THE
00104.100  105 1 0074                   DIFFERENCE IN EXPONENT AND PLACE THE
00105.000  106 1 0074                   RESULT IN THE SUM.                >>
00106.000  107 1 0074 1F                RAR;              ! DIFF IN EXPON
00107.000  108 1 0075                   IF CARRY=1        ! DIFF IN EXPON ODD
00108.000  109 1 0075 D28F00P             THEN BEGIN  ! SHIFT LEAST SIG #(DE) LEFT
00109.000  110 2 0078                                 !  ONE DIGIT INTO SUM (HL)
00110.000  111 2 0078 110000L               DE=@TEMP'SUM;        ! SUM POINTER
00111.000  112 2 007B 2A0E00L               HL=L'SIG'NUM'PT;
00112.000  113 2 007E EB                    DE<==>HL;
00113.000  114 2 007F 13                    DE=DE+1;
00114.000  115 2 0080 0608                  B=8;
00115.000  116 2 0082 CD0000X               CALL TRANS'DATA'LEFT;
00116.000  117 2 0085 210D00L               HL=@EXPON'DIFF;
00117.000  118 2 0088 35                    MEM(HL)=MEM(HL)-1;
00118.000  119 2 0089 CA8300P               IF ZERO THEN GOTO SIGN'CHECK;
00119.000  120 2 008C                     END
00120.000  121 1 008C C39F00P             ELSE BEGIN    ! PUT LEAST SIG # INTO SUM
00121.000  122 2 008F 2A0E00L TRANSFER:           HL=L'SIG'NUM'PT;
00122.000  123 2 0092 EB                    HL<==DE;
00123.000  124 2 0093 110000L               HL=@TEMP'SUM;        ! SUM POINTER
00124.000  125 2 0096 0609                  B=9;
00125.000  126 2 0098 CD0000X               CALL TRANS'DATA;
00126.000  127 1 009B                     END;
00127.000  128 1 009B 3A0D00L SHIFT'BYTE:   A=EXPON;    ! SHIFT RIGHT BY BYTES
00128.000  129 1 009E 0F                  RRC;    ! DIVIDE BY 2, # BYTE SHIFT PT
00129.000  130 1 009F A7                  A=A AND A;
00130.000  131 1 00A0 CA8300P             IF ZERO THEN GOTO SIGN'CHECK;
00130.100  132 1 00A3                                    ! SHIFT COMPLETE
00131.000  133 1 00A3 06004F              B=0;C=A;          ! # BYTES TO BE SHIFTED
00132.000  134 1 00A6 3E08                A=8;
00133.000  135 1 00A8 91                  A=A-C;        ! # BYTE REMAINING IN NUMBER
00134.000  136 1 00A9 210100L             HL=@TEMP'SUM+1;
00135.000  137 1 00AC 545D                D=H;E=L;
00136.000  138 1 00AE 09                  HL=HL+BC;
00137.000  139 1 00AF EB                  HL<==DE;
00138.000  140 1 00B0 47                  B=A;
00139.000  141 1 00B1 CD0000X             CALL TRANS'DATA;
00140.000  142 1 00B4 41                  B=C;
00141.000  143 1 00B5 CD0000X             CALL ZERO'MEM;        ! ZERO LEADING BYTES
00142.000  144 1 00B8          <<         EXPONENTS NOW EQUAL.  MANTISSA OF LEAST
00143.000  145 1 00B8                     SIGNIFICANT NUMBER IS IN SUM.  CHECK SIGNS
00144.000  146 1 00B8                     AND CALL DEC'ADD OR DEC'SUB AS REQUIRED.
00144.100  147 1 00B8                                                        >>
00145.000  148 1 00B8 2A0E00L SIGN'CHECK:       HL=L'SIG'NUM'PT;
00146.000  149 1 00BB 7E                  A=MEM(HL);
00147.000  150 1 00BC 2A0900L             HL=M'SIG'NUM'PT;
00148.000  151 1 00BF AE                  A=A XOR MEM(HL);
00149.000  152 1 00C0 E680                A=A AND #80;     ! MASK OFF XOR OF SIGNS
00150.000  153 1 00C2                     IF ZERO          ! SIGNS THE SAME
00151.000  154 1 00C2 C21C01P               THEN BEGIN             ! ADD
00152.000  155 2 00C5 210100L                 HL=@TEMP'SUM+1;
00153.000  156 2 00C8 EB                      DE <==HL;   ! DE 2D # MANTISSA IN SUM
00154.000  157 2 00C9 424B                    B=D;C=E;
```

```
00155.000  158 2 00CB 2A0900L          HL=M(DIG NUM PT);     ! 1ST NUMBER
00156.000  159 2 00CE 23               HL=HL+1;
00157.000  160 2 00CF 3E08             A=8;
00158.000  161 2 00D1 CD0000X          CALL DEC(ADD);
00159.000  162 2 00D4                  IF CARRY=1            ! OVERFLOW
00160.000  163 2 00D4 D21301P            THEN BEGIN  ! SHIFT RT 1 BYTE & LEFT
00161.000  164 3 00D7                                !  1 DIGIT & INCR EXPONENT
00162.000  165 3 00D7 210000L              HL=@TEMP(SUM);    ! SUM POINTER
00163.000  166 3 00DA 110100L              DE=@TEMP(SUM(1));
00164.000  167 3 00DD 0608                 B=8;
00165.000  168 3 00DF CD0000X              CALL TRAN(DATA);  ! MOVE RT 1 BYTE
00166.000  169 3 00E2 3600                 MEM(HL)=0;
00167.000  170 3 00E4 21F7FF               HL= -9;
00168.000  171 3 00E7 19                   HL=HL+DE;
00169.000  172 3 00E8 545D                 D=H;E=L;
00170.000  173 3 00EA 0608                 B=8;
00171.000  174 3 00EC CD0000X              CALL TRAN(DATA(LEFT); ! MOVE LEFT
00172.000  175 3 00EF 110800               DE=8;                 ! ONE DIGIT
00173.000  176 3 00F2 19                   HL=HL+DE;
00174.000  177 3 00F3 3E10                 A=#10;       ! ADD OVERFLOW DIGIT
00175.000  178 3 00F5 B6                   A=A OR MEM(HL);
00176.000  179 3 00F6 77                   MEM(HL)=A;
00177.000  180 3 00F7 2A0900L              HL=M(DIG NUM PT);
00178.000  181 3 00FA 7E                   A=MEM(HL);
00179.000  182 3 00FB 47                   B=A;              ! SAVE FOR SIGN
00180.000  183 3 00FC E67F                 A=A AND #7F;
00181.000  184 3 00FE 07                   RLC;
00182.000  185 3 00FF A7                   A=A AND A;
00183.000  186 3 0100 37                   CARRY=1;
00184.000  187 3 0101 FA0501P              IF POSITIVE THEN CARRY= NOT CARRY;
           3 0104 3F
00185.000  188 3 0105 1F                   RAR;              ! 8 BIT EXPON
00186.000  189 3 0106 3C                   A=A+1;            ! INCR EXPON
00187.000  190 3 0107 E67F                 A=A AND #7F;      ! MASK OFF EXPON
00188.000  191 3 0109 4F                   C=A;
00189.000  192 3 010A 78                   A=B;
00190.000  193 3 010B E680                 A=A AND #80;      ! MASK OFF SIGN
00191.000  194 3 010D B1                   A=A OR C;
00192.000  195 3 010E E1                   HL=TOS;
00193.000  196 3 010F 77                   MEM(HL)=A;  ! SIGN & EXPON OF SUM
00194.000  197 3 0110                    END
00195.000  198 2 0110 C31901P            ELSE BEGIN      ! NO OVERFLOW DIGIT
00196.000  199 3 0113 2A0900L              HL=M(DIG NUM PT);  ! STORE SIGN &
00197.000  200 3 0116 7E                   A=MEM(HL);         !  EXPON IN SUM
00198.000  201 3 0117 E1                   HL=TOS;
00199.000  202 3 0118 77                   MEM(HL)=A;
00200.000  203 2 0119                    END;
00201.000  204 2 0119                  END
00202.000  205 1 0119 C35901P          ELSE BEGIN  ! SIGNS DIFFERENT, SUBTRACT
00203.000  206 2 011C 210100L            HL=@TEMP(SUM(1));    ! SUM POINTER
00204.000  207 2 011F EB                 HL <= DE;
00205.000  208 2 0120 424B               B=D;C=E;    ! POINTER TO SUM MANTISSA
00206.000  209 2 0122 2A0900L            HL=M(DIG NUM PT);
```

```
00207.000  210 2 0125 23                    HL=HL+1;
00208.000  211 2 0126 3E08                  A=8;
00209.000  212 2 0128 CD0000'               CALL DEC'SUB;      ! BC  MSD OF RESULT
00210.000  213 2 012B 21F8FF                HL= -8;
00211.000  214 2 012E 09                    HL=HL+BC;            ! MANTISSA OF SUM
00212.000  215 2 012F 0608                  B=8;
00213.000  216 2 0131 CD6501P               CALL LEFT'JUST;  ! RETURN: B  # DIGITS
00214.000  217 2 0134                             ! SHIFTED;  B  #FF IF NUMBER 0
00215.000  218 2 0134 3EFF                  A:=FF;
00216.000  219 2 0136 B8                    A-B;
00217.000  220 2 0137                       IF ZERO              ! RESULT IS ZERO
00218.000  221 2 0137 C24101P                 THEN BEGIN
00219.000  222 3 013A E1                        HL=TOS;
00220.000  223 3 013B 3E00                      A=0;
00221.000  224 3 013D 77                        MEM(HL)=A;
00222.000  225 3 013E                         END
00223.000  226 2 013E C35401P                 ELSE BEGIN ! DECREMENT EXPONENT BY B
00224.000  227 3 0141 2A0400L                   HL=M'SIG NUM PT;
00225.000  228 3 0144 7E                        A=MEM(HL);
00226.000  229 3 0145 57                        D=A;               ! SAVE SIGN
00227.000  230 3 0146 E67F                      A=A AND #7F;       ! MASK OFF EXPON
00228.000  231 3 0148 07                        RLC;
00229.000  232 3 0149 A7                        A=A AND A;
00230.000  233 3 014A F24E01P                   IF NEGATIVE THEN CARRY= NOT CARRY;
           3 014D 3F
00231.000  234 3 014E 1F                        RAR;     ! 8 BIT EXPON
00232.000  235 3 014F 90          DECR'A:       A=A-B;  !EXPON OF MDIT SIGNIFICANT
00233.000  236 3 0150                                   !  NUMBER MINUS NUMBER OF
00234.000  237 3 0150                                   !  DIGITS SHIFTED LEFT AFTER
00235.000  238 3 0150                                   !  SUBTRACTION
00235.100  239 3 0150 E67F                      A=A AND #7F;
00236.000  240 3 0152 4F                        C=A;               ! EXPON OF RESULT
00237.000  241 3 0153 7A                        A=D;
00238.000  242 3 0154 E680                      A=A AND #80;       ! MASK OFF SIGN
00239.000  243 3 0156 B1                        A=A OR C;
00240.000  244 3 0157 E1                        HL=TOS;
00241.000  245 3 0158 77                        MEM(HL)=A;         ! SIGN & EXPON
00242.000  246 2 0159                         END;
00243.000  247 1 0159                     END;
00244.000  248 1 0159 E5                  TOS=HL;
00245.000  249 1 015A 23                  HL=HL+1;
00246.000  250 1 015B 110100L             DE=@TEMP'SUM(1);
00247.000  251 1 015E 0608                B=8;
00248.000  252 1 0160 CD0000'             CALL TRANS'DATA;
00249.000  253 1 0163 E1                  HL=TOS;
00250.000  254 1 0164 C9                  RETURN;
00251.000  255 1 0165               <<
00252.000  256 1 0165               ◆◆◆◆◆◆◆◆◆◆◆◆   LEFT JUSTIFY   ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00253.000  257 1 0165               LEFT JUSTIFIES A NUMBER PACKED TWO DIGITS
00254.000  258 1 0165               PER BYTE IN A STRING.
00255.000  259 1 0165
00256.000  260 1 0165               INPUT    B  # BYTES IN NUMBER
00257.000  261 1 0165                        HL POINTER TO MANTISSA OF NUMBER
00258.000  262 1 0165
00259.000  263 1 0165               RETURN   B    # OF DIGITS SHIFTED LEFT
00260.000  264 1 0165                       OR #FF IF NUMBER IS ALL ZEROS
```

```
00261.000  265 1 0165          >>
00262.000  266 1 0165 E5       LEFT JUST:      TOI=HL;
00263.000  267 1 0166 1600             D=0;C=0;
                   1 0168 0E00
00264.000  268 1 016A 58               E=B;
00265.000  269 1 016B 19               HL=HL+DE;
00266.000  270 1 016C 2B               HL=HL-1;        ! POINTER TO MID
00267.000  271 1 016D 7E       CHK NEXT BYTE:  A=MEM(HL);
00268.000  272 1 016E 57               D=A;            ! SAVE DIGIT
00269.000  273 1 016F E6F0             A=A AND #F0;            ! MASK OFF MID
00270.000  274 1 0171                  IF ZERO THEN
00271.000  275 1 0171 C28801P            BEGIN
00272.000  276 2 0174 0C                   C=C+1;
00273.000  277 2 0175 7A                   A=D;
00274.000  278 2 0176 E60F                 A=A AND #0F;        ! MASK OFF LSD
00275.000  279 2 0178                      IF ZERO THEN
00276.000  280 2 0178 C28801P                BEGIN
00277.000  281 3 017B 0C                       C=C+1;
00278.000  282 3 017C 2B                       HL=HL-1;
00279.000  283 3 017D 05                       B=B-1;
00280.000  284 3 017E                          IF ZERO         ! NUMBER IS ZERO
00281.000  285 3 017E C28501P                    THEN BEGIN
00282.000  286 4 0181 06FF                         B=#FF;
00283.000  287 4 0183 E1                           HL=TOI;
00284.000  288 4 0184 C9                           RETURN;
00285.000  289 4 0185                            END
00286.000  290 3 0185 C36D01P                    ELSE GOTO CHK NEXT BYTE;
00287.000  291 2 0188                        END;
00288.000  292 1 0188                    END;
00289.000  293 1 0188          <<      HL IS BYTE WITH MID, C IS # OF DIGITS TO
00290.000  294 1 0188                  BD MOVED LEFT.  E IS TOTAL # OF BYTES.  >>
00291.000  295 1 0188 4379     MOVE LEFT:      B=E;A=C;
00292.000  296 1 018A A7               A=A AND A;
00293.000  297 1 018B                  IF ZERO
00294.000  298 1 018B C29201P            THEN BEGIN    ! ALREADY LEFT JUSTIFIED
00295.000  299 2 018E 0600E1               B=0;HL=TOI;
00296.000  300 2 0191 C9                   RETURN;
00297.000  301 1 0192                    END;
00298.000  302 1 0192 1F               RAR;            ! DIVIDE BY 2
00299.000  303 1 0193 F5               TOI=AFLAGS;  ! SAVE CARRY INDICATING MOVE
00300.000  304 1 0194                               !   ODD NUMBER OF DIGITS LEFT
00301.000  305 1 0194                               !   & A- NUMBER OF BYTES TO
00302.000  306 1 0194                               !   BE FILLED AT LEFT
00303.000  307 1 0194 C5               TOI=BC;      ! B # BYTES, C # DIGITS SHIFT
00304.000  308 1 0195 06004F           B=0;C=A;
00305.000  309 1 0198 545D             D=H;E=L;        ! SOURCE BYTE
00306.000  310 1 019A 09               HL=HL+BC;       ! DESTINATION BYTE
00307.000  311 1 019B C1               BC=TOI;
00308.000  312 1 019C 78               A=B;
00309.000  313 1 019D 07               RLC;            ! MULT BY 2 (TOT # DIGITS)
00310.000  314 1 019E 91               A=A-C;          ! DIGITS TO BE SHIFTED
00311.000  315 1 019F 414F             B=C;C=A;   ! # DIGITS SHIFTED, FOR RETURN
00312.000  316 1 01A1 F1       MOV NEXT BYTE:  AFLAGS=TOI;
```

```
00313.000   317 1 01A2 F5              TDI=AFLAGI;
00314.000   318 1 01A3                 IF CARRY=1
00315.000   319 1 01A3 D2B201P           THEN BEGIN  ! MOVE LSD OF SOURCE BYTE TO
00316.000   320 2 01A6 1A                 A=MEM(DE);  !  MSD OF DESTINATION BYTE
00317.000   321 2 01A7 E60F               A=A AND #0F;
00318.000   322 2 01A9 070707             RLC;RLC;RLC;RLC;
                2 01AC 07
00319.000   323 2 01AD 77                 MEM(HL)=A;
00320.000   324 2 01AE 0D                 C=C-1;
00321.000   325 2 01AF                  END
00322.000   326 1 01AF C3B601P           ELSE BEGIN             ! MOVE BYTE
00323.000   327 2 01B2 1A                 A=MEM(DE);
00324.000   328 2 01B3 77                 MEM(HL)=A;
00325.000   329 2 01B4 0D0D               C=C-1;C=C-1;
00326.000   330 1 01B6                  END;
00327.000   331 1 01B6 CAD001P         IF ZERO THEN GOTO ZERO_LSD;
00328.000   332 1 01B9 1B              DE=DE-1;
00329.000   333 1 01BA F1              AFLAGI=TDI;
00330.000   334 1 01BB F5              TDI=AFLAGI;
00331.000   335 1 01BC                 IF CARRY=1       ! MOVE MSD SOURCE BYTE TO
00332.000   336 1 01BC D2CC01P           THEN BEGIN     !  LSD OF DESTINATION BYTE
00333.000   337 2 01BF 1A                 A=MEM(DE);
00334.000   338 2 01C0 E6F0               A=A AND #F0;
00335.000   339 2 01C2 0F0F0F             RRC;RRC;RRC;RRC;
                2 01C5 0F
00336.000   340 2 01C6 B6                 A=A OR MEM(HL);
00337.000   341 2 01C7 77                 MEM(HL)=A;
00338.000   342 2 01C8 0D                 C=C-1;
00339.000   343 2 01C9 CAD001P            IF ZERO THEN GOTO ZERO_LSD;
00340.000   344 1 01CC                  END;
00341.000   345 1 01CC 2B              HL=HL-1;
00342.000   346 1 01CD C3A101P         GOTO MOV_NEXT_BYTE;
00343.000   347 1 01D0 F1       ZERO_LSD:  AFLAGI=TDI;  ! A  = BYTES TO BE ZEROED
00344.000   348 1 01D1 A7                 A=A AND A;
00345.000   349 1 01D2           NEXT_ZERO:     IF ZERO        ! = LEFT JUSTIFIED
00346.000   350 1 01D2 C2D701P             THEN BEGIN
00347.000   351 2 01D5 E1                   HL=TDI;              ! CLEAR STACK
00348.000   352 2 01D6 C9                   RETURN;
00349.000   353 1 01D7                    END;
00350.000   354 1 01D7 2B              HL=HL-1;
00351.000   355 1 01D8 3600            MEM(HL)=0;
00352.000   356 1 01DA 3D              A=A-1;       ! NUMBER OF LSB REMAINING TO
00353.000   357 1 01DB                             !   BE ZEROED
00354.000   358 1 01DB C3D201P         GOTO NEXT_ZERO;
00355.000   359 0 01DE                 END.
          ***** 0 ERRORS, 0 WARNINGS, SEE 0 *****
  BIAL-80 05.2.3  78-11-26  12:34:30
CHK_NEXT_BYTE       LABEL     #016D
                     271  290
DE_ADD              EXTERNAL  #000E
                       31  161
DEC_SUB             EXTERNAL  #000F
                       31  212
DECR_A              LABEL     #014F
                     235
```

```
EXPON                LOCAL      #000D
                       28    128
EXPON_DIFF           LOCAL      #000D
                       27     28     66     91    117
FP_ADD               ENTRY      #0000
                       24     33
L_SIG_NUM_PT         LOCAL      #000B
                       27     33     59     62     71
                       81     84    112    122    148
LARGER               EXTERNAL   #000A
                       30     75
LEFT_JUST            ENTRY      #0165
                       24    216    266
M_SIG_NUM_PT         LOCAL      #0009
                       26     35     57     60     69
                       79     82     95    150    158
                      180    199    209    227
MOV_NEXT_BYTE        LABEL      #01A1
                      316    346
MOVE_LEFT            LABEL      #0188
                      295
NEXT_ZERO            LABEL      #01D2
                      349    353
SHIFT_BYTE           LABEL      #009B
                      128
SIGN_CHECK           LABEL      #00B2
                      143    113    131
TEMP_SUM             LOCAL      #0000
                       24    111    134    136    155
                      165    166    206    250
TRANS_DATA           EXTERNAL   #000E
                       30     99    126    141    168
                      252
TRANS_DATA_LEFT      EXTERNAL   #000C
                       30    116    174
TRANSFER             LABEL      #008F
                      122     86
ZERO_LID             LABEL      #01D0
                      347    331    343
ZERO_MEM             EXTERNAL   #000D
                       31    143
  BSAL-80/85.2.3   78/11/26  12:35:38
```

```
 BSAL-80/85.2.3   78/11/26  12:35:44
          *****   0 ERRORS,   0 WARNING(S), SEE    0 *****
 SOURCE FILE =   DK2:WALT/DIV.EDT-0
 OBJECT FILE =   DK2:WALT/DIV.OBJ-0
00000.100    1 1 0141          $CONTROL        NAME= FP/DIV
00000.200    2 1 0000          !$CONTROL        NOLIST      .
00000.300    3 1 0000          $CONTROL        LIST,INNERLIST,TABLE,CROSS
00001.000    4 1 0000          <<
00003.000    5 1 0000          ************************************************
00004.000    6 1 0000          **                                            **
00005.000    7 1 0000          **            D I V I S I O N                 **
0000e.000    8 1 0000          **                                            **
00007.000    9 1 0000          ************************************************
00009.000   10 1 0000
00010.000   11 1 0000                  DIVIDES TWO EIGHT BYTE (2 DIGITS PER BYTE)
00010.100   12 1 0000                  BCD NUMBERS.  EACH NUMBER CONTAINS A NINTH
00011.000   13 1 0000                  BYTE WITH THE SIGN AND EXPONENT.
00012.000   14 1 0000                  THE QUOTIENT IS OF THE SAME FORM.
00013.000   15 1 0000
00014.000   16 1 0000                      INPUT   DE-  1ST ADDR OF DIVISOR
00015.000   17 1 0000                              HL-  1ST ADDR OF DIVIDEND
00016.000   18 1 0000                              BC-  1ST ADDR OF QUOTIENT
00016.100   19 1 0000
0001e.200   20 1 0000                      OUTPUT  HL-  1ST ADDR OF QUOTIENT
00017.000   21 1 0000
00018.000   22 1 0000                      %DIVIDEND(17 BYTES)-  LEAD 0,MANTISSA
00019.000   23 1 0000                          OF DIVIDEND, FOLLOWED BY 8 BYTES
00019.010   24 1 0000                          OF ZEROS.
00019.100   25 1 0000                      DIVISOR(10 BYTES)-  DIVISOR RIGHT
00019.200   26 1 0000                          JUSTIFIED
00020.000   27 1 0000                      %DIVISOR(9 BYTES)- MANTISSA OF DIVISOR
00021.000   28 1 0000                          SHIFTED 1 DIGIT LEFT(DIVISOR * 10)
00023.000   29 1 0000                      DIVIDEND PT(2 BYTES)- ADDR OF DIVIDEND
00024.000   30 1 0000
00025.000   31 1 0000                          ERROR OUTPUT FOR DIVIDE BY ZERO
0002e.000   32 1 0000          >>
00027.000   33 1 0000
00031.000   34 1 0000          ENTRY   DIVIDE;
00032.000   35 1 0000
00033.000   36 1 0000          LOCAL   %DIVIDEND(17), DIVISOR(10), %DIVISOR(9),
00034.000   37 1 0000                  DIVIDEND PT(2), BYTE COUNT, DIGITS;
00035.000   38 1 0000
00036.000   39 1 0000          EXTERNAL    TRANS DATA LEFT, ERROR OUT, SPACE,
00037.000   40 1 0000                      ZERO MEM, LARGER, DEC SUB, TRANS DATA;
00038.000   41 1 0000
00039.000   42 1 0000 222400L DIVIDE:   DIVIDEND PT=HL;  ! SAVE DIVIDEND POINTER
00040.000   43 1 0003 210900          HL=8;
00041.000   44 1 0006 09            HL=HL+BC;                ! MID OF QUOTIENT
00042.000   45 1 0007 E5            TOS=HL;          ! KEEP QUOTIENT POINTER
00044.100   4e 1 0008 211100L       HL=%DIVISOR;
00044.200   47 1 000B 0e0=          B=9;
00044.300   48 1 000D CD0000.       CALL TRANS DATA; !DIVISOR INTO DIVISOR REG
00044.400   49 1 0010 0e01          B=1;             ! MOB OF DIVISOR
```

```
00044.500   50 1 0018 CD0000X          CALL ZERO_MEM;   ! ZERO MSB OF DIVISOR
00045.000   51 1 0015 0608             B=8;             ! ZERO 8 LSB OF KDIVIDEND
00046.000   52 1 0017 210000L          HL=@KDIVIDEND;
00047.000   53 1 001A CD0000X          CALL ZERO_MEM;
00048.000   54 1 001D 0608             B=8;             ! TRANSFER MANTISSA OF
00049.000   55 1 001F 2A2400L          HL=DIVIDEND_PT;  !  DIVIDEND INTO 8 MSB
00050.000   56 1 0022 EB               DE<==>HL;        !  OF KDIVIDEND
00051.000   57 1 0023 13               DE=DE+1;         ! LSD OF DIVIDEND MANTISSA
00052.000   58 1 0024 210800L          HL=@KDIVIDEND+8;! MOST SIG 8 BYTES KDIVID
ND
00053.000   59 1 0027 CD0000X          CALL TRANS_DATA;
00053.100   60 1 002A 3600             MEM<HL>=0;       ! 0 MSB OF KDIVIDEND
00054.000   61 1 002C 0608             B=8;             ! SETUP KDIVISOR
00055.000   62 1 002E 211200L          HL=@DIVISOR(1);
00056.000   63 1 0031 EB               DE<==>HL;
00058.000   64 1 0032 211B00L          HL=@KDIVISOR;
00059.000   65 1 0035 CD0000X          CALL TRANS_DATA_LEFT;   ! KDIVISOR=DIVISOR
00059.100   66 1 0038                                          !  MANTISSA * 10
00060.000   67 1 0038 3E09             A=9;             ! BYTE COUNT OF QUOTIENT
00061.000   68 1 003A 322600L          BYTE_COUNT=A;
00062.000   69 1 003D 3E00             A=0;             ! VALUE OF DIGIT OF CURRENT
00063.000   70 1 003F 322700L          DIGIT=A;    !  BYTE OF QUOTIENT
00064.000   71 1 0042 0608     ZERO_DIVISOR:   B=8;         ! CHECK FOR 0 DIVISOR
00065.000   72 1 0044 211200L          HL=@DIVISOR(1);
00067.000   73 1 0047 BE       ZERO_CHECK:     A=A OR MEM<HL>;
00068.000   74 1 0048 23               HL=HL+1;
00069.000   75 1 0049 05               B=B-1;
00070.000   76 1 004A C2470OP          IF NONZERO THEN GOTO ZERO_CHECK;
00071.000   77 1 004I A7               A=A AND A;               ! SET FLAGS
00072.000   78 1 004E                  IF ZERO          ! DIVISOR IS ZERO
00073.000   79 1 004E C25E00P            THEN BEGIN
00074.000   80 3 0051 0177FF              BC=#FF77;    ! SET ALL DEC PTS & SIGN
00075.000   81 3 0054 117777              DE=#7777;    ! ALL 7'S OUT
00076.000   82 3 0057 CD0000X             CALL ERROR_OUT;
00077.000   83 3 005A C9                  RETURN;
00078.000   84 1 005E                    END;
00079.000   85 1 005E C3B600P           GOTO LOOP_BYTE;        ! 1ST BYTE
00082.000   86 1 005E 3A2600L MID_OF_BYTE:    A=BYTE_COUNT;
00083.000   87 1 0061 3D               A=A-1;
00084.000   88 1 0062 16005F           D=0;E=A;
00085.000   89 1 0065 210000L          HL=@KDIVIDEND;
00086.000   90 1 0068 19               HL=HL+DE;    ! KDIVIDEND BYTE W/ 16TH MSD
00087.000   91 1 0069 EB               HL<==>DE;
00088.000   92 1 006A 211B00L          HL=@KDIVISOR;
00089.000   93 1 006D 0609             B=9;
00090.000   94 1 006F CD0000X          CALL LARGER;             ! PICK LARGER
00090.100   95 1 0072 05               B=B-1;
00091.000   96 1 0073                  IF NONZERO       ! MANTISSA OF DIVISOR .LE.
00092.000   97 1 0073 CA9600P            THEN BEGIN     !           DIVIDEND
00093.000   98 2 0076 212700L              HL=@DIGIT;
00094.000   99 2 0079 3E10                 A=#10;       ! ADD 10 DECIMAL TO DIGIT;
00095.000  100 2 007B 86                   A=A+MEM<HL>;
00096.000  101 2 007C 27                   DAA;
00097.000  102 2 007D 77                   MEM<HL>=A;           ! SAVE DIGIT;
```

```
00098.000  103 2 007E 3A2600L              A=BYTE#COUNT;
00099.000  104 2 0081 3D                   A=A-1;
00100.000  105 2 0082 16005F               D=0; E=A;
00101.000  106 2 0085 210000L              HL=@%DIVIDEND;
00102.000  107 2 0088 19                   HL=HL+DE;          ! POINT TO MINUEND
00103.000  108 2 0089 444D                 B=H; C=L;          ! POINT TO DIFFERENCE
00104.000  109 2 008B 111B00L              DE=@%DIVISOR;      ! SUBTRAHEND
00105.000  110 2 008E 3E09                 A=9;
00106.000  111 2 0090 CD0000(              CALL DEC#SUB;
00107.000  112 2 0093 C35E00P              GOTO MID OF #BYTE;
00108.000  113 1 0096                    END;
00109.000  114 1 0096 3A2600L LOD OF BYTE:    A=BYTE#COUNT;
00110.000  115 1 0099 3D                   A=A-1;
00111.000  116 1 009A 16005F               D=0;E=A;
00112.000  117 1 009D 210000L              HL=@%DIVIDEND;
00113.000  118 1 00A0 19                   HL=HL+DE;     ! %DIVIDEND BYTE W# 16TH MOD
00114.000  119 1 00A1 EB                   HL<==>DE;
00115.000  120 1 00A2 211200L              HL=@DIVISOR(1);
00116.000  121 1 00A5 0609                 B=9;
00117.000  122 1 00A7 CD0000(              CALL LARGER;            ! PICK LARGER
00118.000  123 1 00AA 05                   B=B-1;
00119.000  124 1 00AB                      IF NONZERO      ! MANTISSA OF DIVISOR .LE.
00120.000  125 1 00AB CACC00P                THEN BEGIN    !          DIVIDEND
00121.000  126 2 00AE 212700L                 HL=@DIGIT;
00122.000  127 2 00B1 34                      MEM(HL)=MEM(HL)+1;  ! INCREMENT LOD
00123.000  128 2 00B2 3A2600L                 A=BYTE#COUNT;
00124.000  129 2 00B5 3D                      A=A-1;
00125.000  130 2 00B6 16005F                  D=0; E=A;
00126.000  131 2 00B9 210000L                 HL=@%DIVIDEND;
00127.000  132 2 00BC 19                      HL=HL+DE;        ! POINT TO MINUEND
00128.000  133 2 00BD 444D                    B=H; C=L;        ! POINT TO DIFFERENCE
00129.000  134 2 00BF EB                      DE <==> HL;
00130.100  135 2 00C0 211200L                 HL=@DIVISOR(1);
00130.200  136 2 00C3 EB                      HL <==> DE;
00131.000  137 2 00C4 3E09                    A=9;
00132.000  138 2 00C6 CD0000(                 CALL DEC#SUB;
00133.000  139 2 00C9 C35E00P                 GOTO LOD OF #BYTE;
00134.000  140 1 00CC                      END;
00135.000  141 1 00CC E1                   HL=TOS;          ! CURRENT DIGIT OF QUOTIENT
00136.000  142 1 00CD 3A2700L              A=DIGIT;
00137.000  143 1 00D0 77                   MEM(HL)=A;       ! STORE DIGITS IN QUOTIENT
00137.100  144 1 00D1 3E00                 A=0;
00137.200  145 1 00D3 322700L              DIGIT=A;         ! INITIALIZE DIGITS
00138.000  146 1 00D6 3A2600L              A=BYTE#COUNT;
00139.000  147 1 00D9 3D                   A=A-1;
00140.000  148 1 00DA 322600L              BYTE#COUNT=A;
00141.000  149 1 00DD                      IF NONZERO       ! DIVICION NOT COMPLETE
00142.000  150 1 00DD CAE500P                THEN BEGIN
00143.000  151 2 00E0 2B                      HL=HL-1;
00144.000  152 2 00E1 E5                      TOS=HL;       ! CURRENT BYTE OF QUOTIENT
00145.000  153 2 00E2 C35E00P                 GOTO MID OF #BYTE;
00146.000  154 1 00E5                      END;
00147.000  155 1 00E5 E5                   TOS=HL;          ! LOD OF QUOTIENT
00148.000  156 1 00E6 110000               DE=0;
```

```
00151.000  157 1 00E9 19              HL=HL+DE;
00152.000  158 1 00EA 7E              A=MEM(HL);                ! MSB OF QUOTIENT
00153.000  159 1 00EB A7              A=A AND A;                ! SET FLAGS
00154.000  160 1 00EC                 IF ZERO     ! MSD OF QUOTIENT= 0 (QUOTIENT
00155.000  161 1 00EC C20101P           THEN BEGIN       !   OF MANTISSAS < 1.0)
00156.000  162 2 00EF 545D                D=H; E=L;       ! SHIFT ALL BYTES LEFT
00157.000  163 2 00F1 1B                  DE=DE-1;
00158.000  164 2 00F2 0608                B=8;            ! BYTE COUNT
00159.000  165 2 00F4 1A       SHIFT/BYTE:          A=MEM(DE);
00160.000  166 2 00F5 77                  MEM(HL)=A;
00161.000  167 2 00F6 1B                  DE=DE-1;
00162.000  168 2 00F7 2B                  HL=HL-1;
00163.000  169 2 00F8 05                  B=B-1;                ! DECREMENT COUNT
00164.000  170 2 00F9 C2F400P             IF NONZERO THEN GOTO SHIFT/BYTE;
00165.000  171 2 00FC                                  ! ALL BYTES SHIFTED LEFT
00166.000  172 2 00FC 0600                B=0;            ! ADD 0 TO EXPONENT
00167.000  173 2 00FE                   END
00168.000  174 1 00FE C31A01P           ELSE BEGIN          ! QUOTIENT .GE. 1.0
00169.000  175 2 0101 E1                  HL=TOS;               ! LSD OF QUOTIENT
00170.000  176 2 0102 E5                  TOS=HL;
00171.000  177 2 0103 F5                  TOS=AFLAGS;           ! SAVE 1ST DIGIT
00172.000  178 2 0104 545D                D=H; E=L;
00173.000  179 2 0106 0608                B=8;
00174.000  180 2 0108 CD0000X             CALL TRANS/DATA LEFT;
00175.000  181 2 010B 110800              DE=#0008;
00176.000  182 2 010E 19                  HL=HL+DE;             ! MSB OF QUOTIENT
00177.000  183 2 010F F1                  AFLAGS=TOS;
00178.000  184 2 0110 E60F                A=A AND #0F;          ! MASK OUT MSD
00179.000  185 2 0112 070707              RLC;RLC;RLC;RLC;
           2 0115 07
00180.000  186 2 0116 B6                  A=A OR MEM(HL);    ! PUT 1ST DIGIT W/
00181.000  187 2 0117 77                  MEM(HL)=A;            ! REMAINING QUOTIENT
00182.000  188 2 0118 0601                B=1;          ! SAVE TO ADD 1 TO EXPONEN

00183.000  189 1 011A                   END;
00184.000  190 1 011A 211100L           HL=#DIVISOR;
00185.000  191 1 011D EB                HL <= DE;
00186.000  192 1 011E 2A2400L           HL=DIVIDEND PT;
00187.000  193 1 0121 1A                A=MEM(DE);
00188.000  194 1 0122 E67F              A=A AND #7F;    ! MASK OUT EXPON OF DIVISOR
00189.000  195 1 0124 4F                C=A;
00190.000  196 1 0125 07                RLC;
00191.000  197 1 0126 E680              A=A AND #80;    ! MASK OUT SIGN OF EXPON
00192.000  198 1 0128 B1                A=A OR C;       ! 8 BIT EXPON OF DIVISOR
00193.000  199 1 0129 4F                C=A;
00194.000  200 1 012A 78                A=B;     ! EXPON FACTOR FROM DIVISION
00195.000  201 1 012B 91                A=A-C;
00196.000  202 1 012C 47                B=A;            ! SAVE IN B
00197.000  203 1 012D 7E                A=MEM(HL);
00198.000  204 1 012E E67F              A=A AND #7F;  ! MASK OUT EXPON OF DIVIDEND
00199.000  205 1 0130 4F                C=A;
00200.000  206 1 0131 07                RLC;
00201.000  207 1 0132 E680              A=A AND #80;
00202.000  208 1 0134 B1                A=A OR C;       ! 8 BIT EXPON OF DIVIDEND
```

```
00203.000  209 1 0135 80              A=A + B;
00204.000  210 1 0136 E67F            A=A AND #7F;     ! EXPONENT OF QUOTIENT
00205.000  211 1 0138 47              B=A;
00206.000  212 1 0139 1A              A=MEM(DE);
00207.000  213 1 013A AE              A=A XOR MEM(HL);
00208.000  214 1 013B E680            A=A AND #80;     ! MASK OFF SIGN OF QUOTIENT
00209.000  215 1 013D B0              A=A OR B;        ! SIGN & EXPON OF QUOTIENT
00210.000  216 1 013E E1              HL=TOS;           ! LSB OF QUOTIENT
00211.000  217 1 013F 77              MEM(HL)=A;
00212.000  218 1 0140 C9              RETURN;           ! HL - LSB OF QUOTIENT
00213.000  219 0 0141                 END.
          ◆◆◆◆◆   0 ERRORS,   0 WARNINGS. SEE    0 ◆◆◆◆◆
  BSAL-80/85.2.3   78/11/26  12:45:31
BYTE COUNT          LOCAL     #0026
                      37    68    86   103   114
                     128   146   148
DEC SUB             EXTERNAL  #000F
                      40   111   138
DIGIT               LOCAL     #0027
                      37    70    98   126   142
                     145
DIVIDE              ENTRY     #0000
                      34    42
DIVIDEND PT         LOCAL     #0024
                      37    42    55   192
DIVISOR             LOCAL     #0011
                      36    46    62    72   120
                     135   190
ERROR OUT           EXTERNAL  #000B
                      39    82
+DIVIDEND           LOCAL     #0000
                      36    52    58    89   106
                     117   131
+DIVISOR            LOCAL     #001B
                      36    64    92   109
LAPSEP              EXTERNAL  #000E
                      40    94   122
LOW OF BYTE         LABEL     #0096
                     114   139    85
MID OF BYTE         LABEL     #005E
                      86   112   153
SHIFT BYTE          LABEL     #00F4
                     165   170
SPACE               EXTERNAL  #000C
                      39
TRANS DATA          EXTERNAL  #0010
                      40    48    59
TRANS DATA LEFT     EXTERNAL  #000A
                      39    65   180
ZERO CHECK          LABEL     #0047
                      73    76
ZERO DIVISOR        LABEL     #0042
                      71
ZERO MEM            EXTERNAL  #000D
                      40    50    53
  BSAL-80 85.2.3   78/11/26  12:45:30
```

```
 BIAL-80/85.2.3   78/11/26  12:46:36
          *****   0 ERROR(S),   0 WARNING(S). SEE     0 *****
 SOURCE FILE =   D#2:WALT/MULT.EDT-0
 OBJECT FILE =   D#2:WALT/MULT.OBJ-0
00001.000    1 1 00FD          $CONTROL          NAME=FP/MULT
00002.000    2 1 0000          !$CONTROL          NOLIST
00002.100    3 1 0000          $CONTROL          LIST,INNERLIST,TABLE,CROSS
00002.200    4 1 0000
00003.000    5 1 0000          <<
00004.000    6 1 0000                  REF INTEL USER'S LIBRARY PROGRAM MANUAL,
00005.000    7 1 0000                  APRIL 1977, PROGRAM BA14 BY RENAULT C.
00006.000    8 1 0000
00007.000    9 1 0000                  MODIFIED FOR GENERAL USE AS A BSAL FLOAT-
00008.000   10 1 0000                  ING POINT SUBROUTINE.
00009.000   11 1 0000
00010.000   12 1 0000          ;;;
```

```
00048.000   51 1 001B CDC000G        CALL LEFT'JUCT; ! B= #FF IF ZERO
00049.000   52 1 001E                               !  OR NUM DIGITS MOVED
00050.000   53 1 001E 04             B=B+1;
00051.000   54 1 001F                IF ZERO          ! PRODUCT IS ZERO
00052.000   55 1 001F C22D00F          THEN BEGIN
00053.000   56 2 0022 E1                 HL=TOS;          ! POINTER TO PRODUCT
00054.000   57 2 0023 0609               B=9;
00055.000   58 2 0025 CD0000X            CALL ZERO'MEM;
00056.000   59 2 0028 11F7FF             DE= -9;
00057.000   60 2 002B 19                 HL=HL+DE;        ! POINTER TO PRODUCT
00058.000   61 2 002C C9                 RETURN;
00059.000   62 1 002D                  END;
00060.000   63 1 002D 05             B=B-1;      ! NUMBER OF DIGITS TO DECR EXP
00061.000   64 1 002E 3A0000G        A=MULTIPLIER;            ! SIGN & EXPON
00062.000   65 1 0031 57             D=A;
00063.000   66 1 0032 E67F           A=A AND #7F;             ! MASK OFF EXPON
00064.000   67 1 0034 07             RLC;
00065.000   68 1 0035 A7             A=A AND A;
00066.000   69 1 0036 FA3A00F        IF POSITIVE THEN CARRY=1;
            1 0039 37
00067.000   70 1 003A 1F             RAR;         ! 8 BIT EXPON OF MULTIPLIER
00068.000   71 1 003B 90             A=A-B;
00069.000   72 1 003C 47             B=A;
00070.000   73 1 003D 3A0900G        A=MULTIPLICAND;          ! SIGN & EXPON
00071.000   74 1 0040 5F             E=A;
00072.000   75 1 0041 E67F           A=A AND #7F;             ! MASK OFF EXPON
00072.010   76 1 0043 07             RLC;
00072.020   77 1 0044 A7             A=A AND A;
00072.030   78 1 0045 FA4900F        IF POSITIVE THEN CARRY=1;
            1 0048 37
00072.040   79 1 0049 1F             RAR;     ! 8 BIT EXPON OF MULTIPLICAND
00072.050   80 1 004A 80             A=A+B;
00072.060   81 1 004B E67F           A=A AND #7F;    ! MASK OFF EXPON OF PRODUCT
00072.070   82 1 004D 47             B=A;
00072.080   83 1 004E 7A             A=D;
00072.090   84 1 004F AB             A=A XOR E;       ! DETERMINE SIGN OF PROD
00072.100   85 1 0050 E680           A=A AND #80;             ! MASK OFF SIGN
00072.110   86 1 0052 B0             A=A OR B;        ! SIGN & EXPON OF PRODUCT
00072.120   87 1 0053 E1             HL=TOS;          ! POINTER TO PRODUCT
00072.130   88 1 0054 77             MEM(HL)=A;      ! SIGN & EXPON INTO PRODUCT
00072.140   89 1 0055 23             HL=HL+1;
00072.150   90 1 0056 1124006        DE=@PRODUCT(8);
00072.160   91 1 0059 0608           B=8;
00072.170   92 1 005B CDF400F        CALL TRANS'DATA;  ! MANTISSA INTO PRODUCT
00072.180   93 1 005E 11F7FF         DE=-9;
00072.190   94 1 0061 19             HL=HL+DE;        ! POINTER TO PRODUCT
00072.200   95 1 0062 C9             RETURN;
00073.000   96 1 0063
00074.000   97 1 0063          <<   MULTIPLIES N1 BYTES(MULTIPLIER) * N2 BYTES
00075.000   98 1 0063               (MULTIPLICAND) IN BCD FORMAT.  THE UNSIGNED
00075.100   99 1 0063               RESULT OF 2*(N1 + N2) DIGITS IS PLACED
00076.000  100 1 0063               IN THE PRODUCT STORAGE BUFFER.        >>
00077.000  101 1 0063
```

```
00078.000  102 1 0063          MULT'BCD:
00078.100  103 1 00e3 110A00G          DE=@MULTIPLICAND(1); !MOVE MULTIPLICAND
00078.200  104 1 00e6                                      !  INTO *MLTCAND
00079.000  105 1 00e6 211300L          HL=@*MLTCAND;
00080.000  106 1 00e9 0608             B=N2;
00081.000  107 1 00eI CDF400P          CALL TRANS'DATA;
00082.000  108 1 00eE AF               A= A XOR A;  ! CLEAR MSB IN MULTIPLICAND
00082.100  109 1 00eF                              !     & *MLTCAND
00083.000  110 1 00eF 12               MEM(DE)=A;
00084.000  111 1 0070 77               MEM(HL)=A;
00085.000  112 1 0071 211C00G          HL=@PRODUCT; ! CLEAR PRODUCT & LEFT'RT'FF
00086.000  113 1 0074 0611             B= N1+N2+1;
00086.100  114 1 0076 CD0000X          CALL ZERO'MEM;
00087.000  115 1 0079 0E01     UN:           C=1;
00088.000  116 1 007B 210000   DEUX:         HL=0;              ! CLEAR INDEX
00089.000  117 1 007E 222D00L          INDEX=HL;
00090.000  118 1 0081 2A2D00L TROIS:          HL=INDEX;
00091.000  119 1 0084 1101006          DE=@MULTIPLIER(1);
00092.000  120 1 0087 19               HL=HL+DE;
00093.000  121 1 0088 3A2C00L          A=LEFT'RT'FF;
00094.000  122 1 008B B7               A=A OR A;          ! ZERO FLAG AFFECTED
00095.000  123 1 008C 7E               A=MEM(HL);
00096.000  124 1 008D                  IF NONZERO THEN
00097.000  125 1 008D CA9400P            BEGIN
00098.000  126 2 0090 0F                   RRC;
00099.000  127 2 0091 0F                   RRC;
00100.000  128 2 0092 0F                   RRC;
00101.000  129 2 0093 0F                   RRC;
00102.000  130 1 0094                    END;
00103.000  131 1 0094 E60F     QUAT:         A=A AND #0F;
00104.000  132 1 0096 B9               A-C;
00105.000  133 1 0097 C2B500P          IF NONZERO THEN GOTO CIN0;
00106.000  134 1 009A 2A2D00L          HL=INDEX;
00107.000  135 1 009D 111C00G          DE=@PRODUCT;
00108.000  136 1 00A0 19               HL=HL+DE;
00109.000  137 1 00A1 111300L          DE=@*MLTCAND;
00110.000  138 1 00A4 0609             B=N2+1;
00111.000  139 1 00A6 CDE800P          CALL ABIHI;
00112.000  140 1 00A9 D2B500P          IF CARRY=0 THEN GOTO CIN0;
00113.000  141 1 00AC 3E00     JEAN:         A=0;
00114.000  142 1 00AE 8E               A=A+MEM(HL)+CARRY;
00115.000  143 1 00AF 27               DAA;
00116.000  144 1 00B0 77               MEM(HL)=A;
00117.000  145 1 00B1 23               HL=HL+1;
00118.000  146 1 00B2 DAAC00P          IF CARRY=1 THEN GOTO JEAN;
00119.000  147 1 00B5 212D00L CIN0:           HL=@INDEX;
00120.000  148 1 00B8 34               MEM(HL)=MEM(HL)+1;
00121.000  149 1 00B9 3E08             A=N1;
00122.000  150 1 00BB BE               A-MEM(HL);
00123.000  151 1 00BC C2B100P          IF NONZERO THEN GOTO TROIS;
00124.000  152 1 00BF 0C               C=C+1;
00125.000  153 1 00C0 211300L          HL=@*MLTCAND;
00126.000  154 1 00C3 110A00G          DE=@MULTIPLICAND(1);
00127.000  155 1 00C6 0609             B=N2+1;
```

```
00138.000   156 1 00C8 CDE800P          CALL ABDHD;
00139.000   157 1 00CB 3E0A              A=#0A;
00140.000   158 1 00CD B9                A-C;
00141.000   159 1 00CE C27B00P           IF NONZERO THEN GOTO DEUX;
00142.000   160 1 00D1 3A2C00L           A=LEFT'RT'FF;
00143.000   161 1 00D4 B7                A=A OR A;
00144.000   162 1 00D5 C0                IF NONZERO THEN RETURN;
00145.000   163 1 00D6 3C                A=A+1;
00146.000   164 1 00D7 322C00L           LEFT'RT'FF=A;
00147.000   165 1 00DA 111300L           DE=@MLTCAND;
00148.000   166 1 00DD 210A00G           HL=@MULTIPLICAND(1);
00149.000   167 1 00E0 0609              B=N2+1;
00140.000   168 1 00E2 CDF400P          CALL TRANS'DATA;
00141.000   169 1 00E5 C37900P          GOTO UN;
00142.000   170 1 00E8
00143.000   171 1 00E8          ABDHD:
00143.100   172 1 00E8                  <<  SUBROUTINE ADDS (B) BYTES OF MEMORY
00143.200   173 1 00E8                  AREA POINTED TO BE DE TO (B) BYTES OF
00144.000   174 1 00E8                  MEMORY AREA POINTED TO BY HL.  AT RETURN
00145.000   175 1 00E8                  HL AND DE POINT TO FOLLOWING ADDRESS.
00146.000   176 1 00E8                                                      >>
00147.000   177 1 00E8 AF                A=A XOR A;              ! CLEAR CARRY
00148.000   178 1 00E9 1A                A=MEM(DE);
00149.000   179 1 00EA 8E                A=A+MEM(HL)+CARRY;
00150.000   180 1 00EB 27                DAA;
00151.000   181 1 00EC 77                MEM(HL)=A;
00152.000   182 1 00ED 13                DE=DE+1;
00153.000   183 1 00EE 23                HL=HL+1;
00154.000   184 1 00EF 05                B=B-1;
00155.000   185 1 00F0 C2E900P           IF NONZERO THEN GOTO ABDHD(1) ELSE RETURN;
                1 00F3 C9
00156.000   186 1 00F4
00157.000   187 1 00F4          TRANS'DATA:
00157.100   188 1 00F4                  <<  SUBROUTINE MOVES (B) BYTES OF DATA
00157.200   189 1 00F4                  POINTED TO BY DE REG INTO AREA POINTED
00158.000   190 1 00F4                  TO BY HL REG.  AT RETURN DE AND HL
00159.000   191 1 00F4                  POINT ON FOLLOWING ADDRESS.
00160.000   192 1 00F4                                                      >>
00161.000   193 1 00F4 1A                A=MEM(DE);
00162.000   194 1 00F5 77                MEM(HL)=A;
00163.000   195 1 00F6 13                DE=DE+1;
00164.000   196 1 00F7 23                HL=HL+1;
00165.000   197 1 00F8 05                B=B-1;
00166.000   198 1 00F9 C8                IF ZERO THEN RETURN ELSE GOTO TRANS'DATA;
                1 00FA C3F400P
00167.000   199 1 00FD
00168.000   200 0 00FD                  END.
             ***** 0 ERROR(S), 0 WARNING(S), SEE 0 *****
  BIAL-80 05.2.3   78/11/26  12:54:55
ABDHD               LABEL      #00E8
                     171   185   199   156
LOOP                LABEL      #00E5
                     147   133   140
```

```
DEUX                LABEL     #007E
                     116   159
INDEX               LOCAL     #002D
                      37   117   118   134   147
JEAN                LABEL     #00AC
                     141   146
MLTCAND             LOCAL     #0013
                      35   105   137   153   165
LEFT JUST           EXTERNAL  #000A
                      32    51
LEFTRT/FF           LOCAL     #002C
                      37   121   160   164
MULT                ENTRY     #0000
                      29    39
MULT BCD            ENTRY     #0063
                      29    48   102
MULTIPLICAND        GLOBAL    #0009
                      34    41    73   103   154
                     166
MULTIPLIER          GLOBAL    #0000
                      34    45    64   119
N1                  EQUATE    #0008
                      33   113   149
N2                  EQUATE    #0003
                      33   106   113   138   155
                     167
PRODUCT             GLOBAL    #001C
                      36    50    90   112   135
QUAT                LABEL     #0094
                     131
TRANS DATA          ENTRY     #00F4
                      29    43    47    92   107
                     168   187   198
TROIS               LABEL     #0081
                     113   151
UN                  LABEL     #0079
                     115   169
ZERO MEM            EXTERNAL  #000E
                      32    58   114
  BIAL-80 85.2.3   78/11/26  12:55:56
```

```
 BIAL-80 85.2.3   78/11/26  12:56:03
          *****   0 ERROR(S),   0 WARNING(S). SEE    0 *****
 SOURCE FILE =   DK2:WALT/SQRT.EDT-0
 OBJECT FILE =   DK2:WALT/SQRT.OBJ-0
00001.000    1 1 007E          $CONTROL        NAME=SQRT
00002.000    2 1 0000          $CONTROL        LIST,INNERLIST,TABLE,CROSS
00003.000    3 1 0000          !$CONTROL        NOLIST
00004.000    4 1 0000
00005.000    5 1 0000          <<
00006.000    6 1 0000          ************************************************
00007.000    7 1 0000          **                                            **
00008.000    8 1 0000          **        S Q U A R E    R O O T             **
00009.000    9 1 0000          **                                            **
00010.000   10 1 0000          ************************************************
00011.000   11 1 0000
00012.000   12 1 0000                  FINDS THE SQUARE ROOT OF A FLOATING-POINT
00014.000   13 1 0000                  NUMBER OF THE FORM M * (10 ** N).
00015.000   14 1 0000
00016.000   15 1 0000                  THE FIRST ESTIMATE IS TAKEN TO BE
00017.000   16 1 0000                  M * (10 ** N/2). THIS NUMBER IS DIVIDED
00017.100   17 1 0000                  INTO THE ORIGINAL NUMBER.  THE FIRST
00017.200   18 1 0000                  ESTIMATE AND THE QUOTIENT ARE THEN
00018.000   19 1 0000                  AVERAGED TO FORM A NEW ESTIMATE.  THIS
00019.000   20 1 0000                  PROCESS IS REPEATED UNTIL THE MOST
00020.000   21 1 0000                  SIGNIFICANT SIX DIGITS OF THE ESTIMATE
00020.100   22 1 0000                  AND THE QUOTIENT MATCH.
00021.000   23 1 0000
00024.000   24 1 0000                  INPUTS  BC   POINTER TO RESULTING SQ ROOT
00025.000   25 1 0000                          HL   POINTER TO NUMBER
00026.000   26 1 0000
00027.000   27 1 0000                  OUTPUT  HL   POINTER TO SQUARE ROOT
00028.000   28 1 0000
00029.000   29 1 0000                          AN ERROR IS OUTPUT FOR THE SQUARE
00030.000   30 1 0000                          ROOT OF A NEGATIVE NUMBER
00031.000   31 1 0000                                                        >>
00032.000   32 1 0000          ENTRY   SQRT;
00033.000   33 1 0000          LOCAL   ROOT1(9), ROOT2(9), TEMP/SUM(9);
00034.000   34 1 0000          EXTERNAL    TRANS DATA, FP ADD, DIVIDE, ERROR/OUT,
00035.000   35 1 0000                      LARGER;
00035.100   36 1 0000
00036.000   37 1 0000 7E        SQRT:   A=MEM(HL);     ! CHECK FOR NEGATIVE NUMBER
00037.000   38 1 0001 A7                A=A AND A;
00038.000   39 1 0002                  IF NEGATIVE
00039.000   40 1 0003 F20F00#             THEN BEGIN     ! SQRT OF NEGATIVE NUMBER
00040.000   41 2 0005 016#BF                BC=#BF66;    ! SET DECIMAL POINT & SIGN
00041.000   42 2 0008 116666                DE=#6666;    ! ALL 6'S- NEG SQRT
00042.000   43 2 000B CD0000(               CALL ERROR OUT;
00043.000   44 2 000E C9                    RETURN;
00044.000   45 1 000F                     END;
00045.000   46 1 000F E67F              A=A AND #7F;
00046.000   47 1 0011 C5                TOS=BC;          ! RESULT POINTER
00047.000   48 1 0012 E5                TOS=HL;          ! POINTER TO NUMBER
00048.000   49 1 0013 07                RLC;
```

```
00049.000   50 1 0014 A7              A=A AND A;
00051.000   51 1 0015 F21900P         IF NEGATIVE THEN CARRY= NOT CARRY;
               1 0018 3F
00052.000   52 1 0019 1F              RAR;              ! EIGHT BIT EXPONENT
00053.000   53 1 001A A7              A=A AND A;
00054.000   54 1 001B F21F00P         IF NEGATIVE THEN CARRY=NOT CARRY;
               1 001E 3F
00054.100   55 1 001F                                  !  SIGN NEGATIVE
00055.000   56 1 001F 1F              RAR;              ! EXPON DIVIDED BY 2
00056.000   57 1 0020 110000L        DE=@ROOT1;
00057.000   58 1 0023 EB              DE<==>HL;
00058.000   59 1 0024 77              MEM<HL>=A;        ! PLACE EXPON IN ROOT1
00059.000   60 1 0025 13              DE=DE+1;
00060.000   61 1 0026 23              HL=HL+1;
00061.000   62 1 0027 0608            B=8;
00062.000   63 1 0029 CD0000X         CALL TRANC'DATA;  ! MOVE 'M' INTO ROOT1
00063.000   64 1 002C E1        NEXT'ROOT2:     HL=TOS;
00064.000   65 1 002D E5              TOS=HL;
00065.000   66 1 002E 110000L        DE=@ROOT1;
00066.000   67 1 0031 010900L        BC=@ROOT2;
00067.000   68 1 0034 CD0000X        CALL DIVIDE;      ! QUOTIENT @ROOT2, (HL)
00068.000   69 1 0037 0603            B=3;
00069.000   70 1 0039 210E00L        HL=@ROOT1+6;
00070.000   71 1 003C 110F00L        DE=@ROOT2+6;
00071.000   72 1 003F CD0000'        CALL LARGER;      ! B=#FF IF ROOT1=ROOT2
00072.000   73 1 0042 04              B=B+1;
00073.000   74 1 0043                 IF ZERO           ! MANTISSAS SAME
00074.000   75 1 0043 C25D00P           THEN BEGIN     ! COMPARE EXPONENTS
00075.000   76 2 0046 3A0000L             A=ROOT1;
00075.500   77 2 0049 210900L             HL=@ROOT2;
00076.000   78 2 004C AE                  A=A XOR MEM<HL>;
00077.000   79 2 004D                     IF ZERO       ! NUMBER THE SAME
00078.000   80 2 004D C25D00P                THEN BEGIN   ! ROOT1 IS RESULT SQRT
00079.000   81 3 0050 E1                        HL=TOS; ! CLEAR POINTER TO
00080.000   82 3 0051                                   !   ORIGINAL NUMBER
00081.000   83 3 0051 E1                        HL=TOS;         ! RESULT POINTER
00082.000   84 3 0052 E5                        TOS=HL;
00083.000   85 3 0053 110000L                 DE=@ROOT1;
00084.000   86 3 0056 0609                    B=9;
00085.000   87 3 0058 CD0000X                 CALL TRANC'DATA;
00086.000   88 3 005B E1                      HL=TOS;         ! POINTER TO SQRT
00087.000   89 3 005C C9                      RETURN;
00088.000   90 2 005D                       END;
00089.000   91 1 005D                   END;
00090.000   92 1 005D 011200L          BC=@TEMP'SUM;
00091.000   93 1 0060 110000L          DE=@ROOT1;
00092.000   94 1 0063 210900L          HL=@ROOT2;
00093.000   95 1 0066 CD0000X          CALL FP'ADD;      ! HL=@TEMP'SUM
00094.000   96 1 0069 117500P          DE=@TWO;          ! DIVISOR= 2.0
00095.000   97 1 006C 010000L          BC=@ROOT1;
00096.000   98 1 006F CD0000X          CALL DIVIDE; ! FIND THE AVERAGE FOR A NEW
00097.000   99 1 0072                                 !   ESTIMATE OF THE SQRT
00098.000  100 1 0072 C32C00P          GOTO NEXT'ROOT2;
00098.100  101 1 0075 010000  TWO: DB #01,0,0,0,0,0,0,0,#20; ! FLOATING POINT 2.0
```

```
                1 0078 000000
                1 007B 000020
00099,000  102 0 007E                    END.
          ***** 0 ERROR(S),  0 WARNING(S). SEE    0 *****
  BIAL-80 85.2.3   78/11/26  13:00:42
DIVIDE              EXTERNAL  #000C
                      34     68     98
ERROR'OUT           EXTERNAL  #000D
                      34     43
FP'ADD              EXTERNAL  #000B
                      34     95
LARGER              EXTERNAL  #000E
                      35     72
NEXT'ROOT2          LABEL     #002C
                      64    100
ROOT1               LOCAL     #0000
                      33     57     66     70     76
                      85     93     97
ROOT2               LOCAL     #0009
                      33     67     71     77     94
SORT                ENTRY     #0000
                      32     37
TEMP'SUM            LOCAL     #0012
                      33     92
TRANS'DATA          EXTERNAL  #000A
                      34     63     87
TWO                 LABEL     #0075
                     101     96
  BIAL-80 85.2.3   78/11/26  13:01:14
```

APPENDIX E

UTILITY PROGRAMS

```
 BIAL-80/85.2.3   78/11/27  12:04:59
          *****   0 ERROR(,   0 WARNING(, SEE     0 *****
 SOURCE FILE =   DK2:WALT/PRINT.EDT-0
 OBJECT FILE =   DK2:WALT/PRINT.OBJ-0
00001.000    1 1 0057
00002.000    2 1 0000          $CONTROL          NAME= OUTPUT;
00003.000    3 1 0000          !$CONTROL          NOLIST
00004.000    4 1 0000          $CONTROL          LIST,INNERLIST,TABLE,CROSS
00005.000    5 1 0000
00006.000    6 1 0000          ENTRY             PRINTOUT, ERROR/OUT, SPACE,
0006e.100    7 1 0000                            PRINT/DELAY, DELAY;
00007.000    8 1 0000
00008.000    9 1 0000          EQUATE            PRINT/ADV = #A800;
00008.100   10 1 0000          EQUATE            OUT/SEL = #B000;
00009.000   11 1 0000          EQUATE            DEC/SIGN/POL/OUT = #7700;
00010.000   12 1 0000          EQUATE            BCD 3/2/OUT  = #6800;
00011.000   13 1 0000          EQUATE            BCD 1/0/OUT  = #6700;
00012.000   14 1 0000          EQUPTE            BCD 5 4/OUT  = #5800;
00013.000   15 1 0000
00014.000   16 1 0000          <<
00015.000   17 1 0000          ********   PRINTOUT / ERROR OUTPUT   **************
00016.000   18 1 0000
00017.000   19 1 0000                  INPUT:  B-  SIGN, DATA POLARITY, & DEC PTS
00018.000   20 1 0000                          C-  2 MID OF OUTPUT
00019.000   21 1 0000                          D-  2 CENTER DIGITS OF OUTPUT
00020.000   22 1 0000                          E-  2 LSD OF OUTPUT
00021.000   23 1 0000                  >>
00022.000   24 1 0000
00023.000   25 1 0000          PRINTOUT:
00023.100   26 1 0000          ERROR/OUT:
00024.000   27 1 0000 3200B0          MEM(OUT/SEL)=A;   ! DPP BUS GIVEN TO 8010
00025.000   28 1 0003 7B              A=B;              !  SIGN,POLARITY & DEC PTS
00025.100   29 1 0004 EEFF            A=A XOR #FF;      !  COMPLEMENT
00026.000   30 1 0006 320077          MEM(DEC SIGN POL OUT)=A;
00027.000   31 1 0009 79              A=C;              !  2 MID
00027.100   32 1 000A EEFF            A=A XOR #FF;      !  COMPLEMENT
00028.000   33 1 000C 320058          MEM(BCD 5 4 OUT)=A;
00029.000   34 1 000F 7A              A=D;              !  2 CENTER DIGITS
00029.100   35 1 0010 EEFF            A=A XOR #FF;      !  COMPLEMENT
00030.000   36 1 0012 320068          MEM(BCD/3/2/OUT)=A;
00031.000   37 1 0015 7B              A=E;              !  2 LSD
00031.100   38 1 0016 EEFF            A=A XOR #FF;      !  COMPLEMENT
00032.000   39 1 0018 320067          MEM(BCD 1/0/OUT)=A;
00033.000   40 1 001B 00              NOP;    !  1 USEC REQUIRED BEFORE PRINT
00034.000   41 1 001C 3200A8          MEM(PRINT/ADV)=A;     !  PRINT & ADVANCE
00035.000   42 1 001F CD4400R         CALL PRINT DELAY;     !  FINISH PRINTING
00036.000   43 1 0022 3200B0          MEM(OUT/SEL)=A;  !  PTN DPP BUS TO
00036.100   44 1 0025                                   !   PLETHYSMOGRAPH
00037.000   45 1 0025 C9              RETURN;
00038.000   46 1 0026
```

```
00039.000   47 1 0026          <<
00040.000   48 1 0026          ******************   SPACE   **********************
00041.000   49 1 0026
00042.000   50 1 0026                SPACES THE DIGITAL PANEL PRINTER (DPP) ONCE.
00043.000   51 1 0026          >>
00044.000   52 1 0026          SPACE:
00045.000   53 1 0026 3200B0          MEM(OUT'SEL)=A;  ! DPP BUS CONTROL TO 8010
00046.000   54 1 0029 3E00            A=%000000000;    !  NEG TRUE DATA POLARITY
00047.000   55 1 002B 320077          MEM(DEC'SIGN POL'OUT)=A;   !  NO SIGNS OR
00047.100   56 1 002E                                            !   DEC PTS
00048.000   57 1 002E 3EFF            A=#FF;
00049.000   58 1 0030 320058          MEM(BCD'5'4'OUT)=A;        !  BLANK
00050.000   59 1 0033 320068          MEM(BCD'3'2'OUT)=A;        !  BLANK
00051.000   60 1 0036 320067          MEM(BCD'1'0'OUT)=A;        !  BLANK
00052.000   61 1 0039 00              NOP;                       !  1 USEC BEFORE PRINT
00053.000   62 1 003A 3200A8          MEM(PRINT'ADV)=A;          !  PRINT & ADVANCE
00054.000   63 1 003D CD4400P         CALL PRINT'DELAY;          !  FINISH PRINTING
00055.000   64 1 0040 3200B0          MEM(OUT'SEL)=A;   !  RTN DPP BUS TO
00055.100   65 1 0043                                  !   PLETHYSMOGRAPH
00056.000   66 1 0043 C9              RETURN;
00057.000   67 1 0044          <<
00058.000   68 1 0044          ****************    PRINT DELAY    ***************
00060.000   69 1 0044
00061.000   70 1 0044                PROVIDES APPROX 500 MSEC DELAY AFTER PRINT
00062.000   71 1 0044                COMMAND IS GIVEN TO THE DIGITAL PANEL
00063.000   72 1 0044                PRINTER(DPP).  INSURES DATA ON DPP BUS
00063.100   73 1 0044                REMAINS VALID DURING REQUIRED PRINTING
00063.200   74 1 0044                TIME.
00064.000   75 1 0044          >>
00064.100   76 1 0044          PRINT'DELAY:
00065.000   77 1 0044 0E05            C=5;      ! # OF 100 MSEC DELAYS
00066.000   78 1 0046 3E64            A=100;    ! # OF 1 MSEC DELAYS
00067.000   79 1 0048 0687    DELAY:  B=135;    ! COUNT FOR 1 MSEC DELAY
00068.000   80 1 004A 05      DLY1:   B=B-1;
00069.000   81 1 004B C24A00P         IF NONZERO THEN GOTO DLY1;
00070.000   82 1 004E 3D              A=A-1;
00071.000   83 1 004F C24800P         IF NONZERO THEN GOTO DELAY;
00072.000   84 1 0052 0D              C=C-1;
00073.000   85 1 0053 C24600P         IF NONZERO THEN GOTO PRINT'DELAY(2);
00075.000   86 1 0056 C9              RETURN;
00076.000   87 0 0057                END.
            ***** 0 ERROR(S),  0 WARNINGS. SEE    0 *****
  BIAL-80/85.2.3   78/11/27  12:08:53
BCD 1 0 OUT       EQUATE    #6700
                    13     39     60
BCD 3 2 OUT       EQUATE    #6800
                    12     36     59
BCD 5 4 OUT       EQUATE    #5800
                    14     33     58
DEC SIGN POL OU   EQUATE    #7700
                    11     30     55
DELAY             ENTRY     #0048
                     7     79     83
DLY1              LABEL     #004A
                    80     81
```

```
ERROR_OUT           ENTRY     #0000
                      6    26
OUT_CEL             EQUATE    #B000
                     10    27     43     53     64
PRINT_ADV           EQUATE    #A800
                      9    41     62
PRINT_DELAY         ENTRY     #0044
                      7    42     63     76     85
PRINTOUT            ENTRY     #0000
                      6    25
SPACE               ENTRY     #0026
                      6    52
  BSAL-80/85.2.3  78/11/27  12:09:27
```

```
 BIAL-80 85.2.3   78/11/26  13:01:21
        *****   0 ERROR(S),   0 WARNING(S), SEE     0 *****
 SOURCE FILE =   DK2:WALT_UTIL.EDT-0
 OBJECT FILE =   DK2:WALT_UTIL.OBJ-0
00001.000    1 1 0071
00002.000    2 1 0000          $CONTROL          NAME =UTIL
00003.000    3 1 0000          $CONTROL          LIST,INNERLIST,TABLE,CROSS
00004.000    4 1 0000          !$CONTROL          NOLIST
00005.000    5 1 0000          ENTRY             ZERO_MEM, LARGER, TRANS_DATA_LEFT,
00005.100    6 1 0000                            DEC_ADD,  DEC_SUB;
00006.000    7 1 0000
00007.000    8 1 0000          <<
00008.000    9 1 0000          ************   ZERO MEMORY   ************************
00009.000   10 1 0000
00010.000   11 1 0000                              INPUT-   B  # BYTES TO BE ZEROED
00011.000   12 1 0000                                       HL 1ST ADDR TO BE ZEROED
00012.000   13 1 0000          >>
00013.000   14 1 0000 3600    ZERO_MEM:    MEM(HL)=0;
00014.000   15 1 0002 23                 HL=HL+1;
00015.000   16 1 0003 05                 B=B-1;
00016.000   17 1 0004 C8                 IF ZERO THEN RETURN ELSE GOTO ZERO_MEM;
            1 0005 C300000F
00017.000   18 1 0008
00018.000   19 1 0008          <<
00019.000   20 1 0008          ******************   LARGER   ***********************
00020.000   21 1 0008
00021.000   22 1 0008                  COMPARES TWO CHARACTER STRINGS POINTED TO
00022.000   23 1 0008                  BY DE AND HL OF B BYTES EACH.  DETERMINES
00023.000   24 1 0008                  IF HL STRING > DE STRING.
00024.000   25 1 0008                  INPUT-    B    # BYTES IN EACH STRING
00025.000   26 1 0008                            DE   1ST LOC (LSB) OF 1ST STRING
00026.000   27 1 0008                            HL   1ST LOC (LSB) OF 2ND STRING
00027.000   28 1 0008
00028.000   29 1 0008                  OUTPUT-   B = #00    IF MEM(HL) < MEM(DE)
00029.000   30 1 0008                            B = #01    IF MEM(HL) > MEM(DE)
00030.000   31 1 0008                            B = #FF    IF MEM(HL) = MEM(DE) >>
00031.000   32 1 0008
00032.000   33 1 0008 48       LARGER:          C=B;
00033.000   34 1 0009 0600              B=0;              !  DEFAULT OF OUTPUT
00034.000   35 1 000B 09                HL=HL + BC;
00035.000   36 1 000C EB                HL<==>DE;
00036.000   37 1 000D 09                HL=HL + BC;
00037.000   38 1 000E EB                DE <==> HL;
00038.000   39 1 000F 2B       NEXT_LOC:        HL=HL-1;
00039.000   40 1 0010 1B                DE=DE-1;
00040.000   41 1 0011 1A                A=MEM(DE);
00041.000   42 1 0012 BE                A-MEM(HL);
00042.000   43 1 0013                   IF CARRY=1          !  MEM(HL) > MEM(DE)
00043.000   44 1 0013 D21900F             THEN BEGIN
00044.000   45 2 0016 0601                 B=#01;
00045.000   46 2 0018 C9                   RETURN;
0004E.000   47 1 0019                     END;
00047.000   48 1 0019                   IF ZERO             !  BYTES THE SAME
00048.000   49 1 0019 C22800F             THEN BEGIN
00049.000   50 2 001C 0D                   C=C-1;
00050.000   51 2 001D C22F00F               IF NONZERO THEN GOTO NEXT_LOC;
```

```
00051.000   52 2 0020                                  !  CHECK REMAINING BYTES
00052.000   53 2 0020 06FF                 B=#FF;
00053.000   54 1 0022                    END;
00054.000   55 1 0022 C9                 RETURN;       !  MEM(HL) < MEM(DE)
00055.000   56 1 0023
00056.000   57 1 0023              <<
00057.000   58 1 0023              ++++++++++++   TRANSFER DATA LEFT   ++++++++++++++++
00058.000   59 1 0023
00059.000   60 1 0023                    TAKES B BYTES OF MEMORY POINTED ON BY DE
00059A.100  61 1 0023                    AND TRANSFERS THEM INTO B+1 BYTES POINTED
00060.000   62 1 0023                    ON BY HL BY SHIFTING ONE DIGIT(BCD) TO THE
00061.000   63 1 0023                    LEFT.  A LEADING AND TRAILING ZERO IS
00062.000   64 1 0023                    ADDED.  THIS IS THE SAME AS MULTIPLYING
00062.100   65 1 0023                    BY 10 DECIMAL.
00063.000   66 1 0023
00064.000   67 1 0023                    INPUT-    B   # OF BYTES IN DE REG STRING
00065.000   68 1 0023                              DE  LOC OF SOURCE STRING
00066.000   69 1 0023                              HL  LOC OF DESTINATION STRING
00067.000   70 1 0023
00068.000   71 1 0023                    OUTPUT-   DE  LOC OF SOURCE STRING
00069.000   72 1 0023                              HL  LOC OF DESTINATION STRING
00070.000   73 1 0023              >>
00071.000   74 1 0023              TRANS DATA LEFT:
00072.000   75 1 0023 EB                    DE<==>HL;
00073.000   76 1 0024 48                    C=B;
00074.000   77 1 0025 0600                  B=0;
00075.000   78 1 0027 09                    HL=HL+BC;
00076.000   79 1 0028 2B                    HL=HL-1;      ! MID OF SOURCE
00077.000   80 1 0029 EB                    HL<== DE;
00078.000   81 1 002A 09                    HL=HL+BC;     ! MID OF DESTINATION
00079.000   82 1 002B 1A              SHIFT:    A=MEM(DE);
00080.000   83 1 002C E6F0                  A=A AND #F0;  ! MASK OFF MID OF BYTE
00081.000   84 1 002E 0F0F0F                RRC;RRC;RRC;RRC;   ! MAKE LOC OF BYTE
            1 0031 0F
00082.000   85 1 0032 B0                    A=A OR B;   ! CONCATENATE WITH MID OF BYTE
00083.000   86 1 0033 77                    MEM(HL)=A;        ! STORE BYTE
00084.000   87 1 0034 2B                    HL=HL-1;
00085.000   88 1 0035 1A                    A=MEM(DE);    ! GET MID NEXT BYTE
00086.000   89 1 0036 E60F                  A=A AND #0F;  ! MASK OFF LOC OF BYTE
00087.000   90 1 0038 070707                RLC;RLC;RLC;RLC;  ! MAKE MID OF BYTE
            1 003B 07
00088.000   91 1 003C 0D                    C=C-1;        ! DECREMENT BYTE COUNT
00089.000   92 1 003D                       IF ZERO
00090.000   93 1 003D C24200P                 THEN BEGIN
00091.000   94 2 0040 77                        MEM(HL)=A;        ! STORE LAST DIGIT
00092.000   95 2 0041                          RETURN
00093.000   96 2 0041 C9                    END;
00094.000   97 1 0042 47                    B=A;          ! SAVE MID OF NEXT BYTE
00095.000   98 1 0043 1B                    DE=DE-1;
00096.000   99 1 0044 C32B00P              GO TO SHIFT;
00097.000  100 1 0047
00098.000  101 1 0047              <<
```

```
00100.000  102 1 0047          ♦♦♦♦♦♦♦♦♦♦♦♦   DECIMAL ADDITION    ♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦
00101.000  103 1 0047
00102.000  104 1 0047                  REF LEVENTHAL, LANCE A.: 8080/8085
00103.000  105 1 0047                  ASSEMBLY LANG PROG, OSBORNE & ASSOC, 1978.
00104.000  106 1 0047
00105.000  107 1 0047                  MODIFIED FOR GENERAL USE AS A BCAL
00105.100  108 1 0047                  SUBROUTINE.
00106.000  109 1 0047
00107.000  110 1 0047                  ADDS MULTIPLE BYTE (TWO DIGITS PER BYTE)
00108.000  111 1 0047                  DECIMAL NUMBERS OF THE SAME LENGTH.
00109.000  112 1 0047
00110.000  113 1 0047                  INPUT:  A  -    BYTE COUNT OF EACH NUMBER
00111.000  114 1 0047                  BC -    POINTER OF RESULT ADDRESS
00112.000  115 1 0047                  HL -    POINTER TO FIRST NUMBER
00113.000  116 1 0047                  DE -    POINTER TO SECOND NUMBER
00114.000  117 1 0047
00115.000  118 1 0047                  OUTPUT: BC -  POINTER TO MOST SIGNIFICANT
00115.100  119 1 0047                            DIGITS OF RESULT
00116.000  120 1 0047                  CARRY-  1 IF OVERFLOW
00117.000  121 1 0047
00118.000  122 1 0047                                          >>
00119.000  123 1 0047 C5       DEC'ADD:  TOS=BC;     ! BEGINNING ADDR OF RESULT
00120.000  124 1 0048 47                 B=A;          ! NO. OF BYTES
00121.000  125 1 0049 A7                 A=A AND A;              ! CLEAR CARRY
00122.000  126 1 004A 1A       LOOP:     A=MEM(DE);    ! TWO DIGITS OF 2ND NO.
00123.000  127 1 004B 8E                 A=A+MEM(HL)+CARRY; ! ADD DIGITS OF 1ST NO.
00124.000  128 1 004C 27                 DAA;
00125.000  129 1 004D E3                 HL<== TOS;    ! RESULT ADDRESS
00126.000  130 1 004E 77                 MEM(HL)=A;    ! STORE BYTE OF RESULT
00127.000  131 1 004F 23                 HL=HL+1;      ! INCREMENT RESULT POINTER
00128.000  132 1 0050 E3                 HL<== TOS;    ! SAVE RESULT ADDR
00129.000  133 1 0051 13                 DE=DE+1;      ! UPDATE INDICES
00130.000  134 1 0052 23                 HL=HL+1;
00131.000  135 1 0053 05                 B=B-1;
00132.000  136 1 0054 C24A00             IF NONZERO THEN GOTO LOOP;
00133.000  137 1 0057 C1                 BC=TOS;       ! POINTER TO RESULT MSD
00134.000  138 1 0058 C9                 RETURN;
00135.000  139 1 0059
00136.000  140 1 0059                  <<
00137.000  141 1 0059          ♦♦♦♦♦♦♦♦♦   DECIMAL SUBTRACTION    ♦♦♦♦♦♦♦♦♦♦♦♦♦♦
00138.000  142 1 0059
00139.000  143 1 0059                  SUBTRACTS MULTIPLE BYTE (TWO DIGITS PER
00140.000  144 1 0059                  BYTE) DECIMAL NUMBERS OF THE SAME LENGTH.
00140.100  145 1 0059                  BASED ON THE GENERAL ALGORITHM:
00141.000  146 1 0059                                          ------
00142.000  147 1 0059                  X - Y = X + 99 - Y + BORROW
00143.000  148 1 0059
00144.000  149 1 0059                  INPUT:  A  -    BYTE COUNT OF EACH NUMBER
00145.000  150 1 0059                          BC -    POINTER OF RESULT ADDRESS
00146.000  151 1 0059                          HL -    POINTER TO MINUEND
00147.000  152 1 0059                          DE -    POINTER TO SUBTRAHEND
00148.000  153 1 0059
00149.000  154 1 0059                  OUTPUT: BC -  POINTER TO MOST SIGNIFICANT
00150.000  155 1 0059                            DIGITS OF RESULT
00151.000  156 1 0059
```

```
00152.000  156 1 005>                                                    >>
00153.000  157 1 0059 C5        DEC/SUB:  TOS=BC;       ! BEGINNING ADDR OF RESULT
00154.000  158 1 005A 47                  B=A;          ! NO. OF BYTES
00155.000  159 1 005B 37                  CARRY=1;      ! SET CARRY
00156.000  160 1 005C 3E99      LOOOP:      A=#99;      ! FIND 99 OR 100 COMPLEMENT
00157.000  161 1 005E CE00                A=A+0+CARRY; !  BASED ON STATUS OF BORROW
00158.000  162 1 0060 EB                  HL<==>DE;     ! GET SUBTRAHEND ADDRESS
00159.000  163 1 0061 96                  A=A-MEM(HL);  ! SUBTRACT SUBTRAHEND
00160.000  164 1 0062 EB                  HL<==>DE;     ! GET MINUEND ADDRESS
00161.000  165 1 0063 86                  A=A+MEM(HL);  ! ADD MINUEND
00162.000  166 1 0064 27                  DAA;
00163.000  167 1 0065 E3                  HL<==>TOS;    ! RESULT ADDRESS
00164.000  168 1 0066 77                  MEM(HL)=A;    ! STORE BYTE OF RESULT
00165.000  169 1 0067 23                  HL=HL+1;      ! INCREMENT RESULT POINTER
00166.000  170 1 0068 E3                  HL<==>TOS;    ! SAVE RESULT ADDR
00167.000  171 1 0069 13                  DE=DE+1;      ! UPDATE INDICES
00168.000  172 1 006A 23                  HL=HL+1;
00169.000  173 1 006B 05                  B=B-1;
00170.000  174 1 006C C25C00P             IF NONZERO THEN GOTO LOOOP;
00171.000  175 1 006F C1                  BC=TOS;       ! POINTER TO RESULT MID
00172.000  176 1 0070 C9                  RETURN;
00173.000  177 0 0071                     END.
            ***** 0 ERROR(S), 0 WARNING(S). SEE 0 *****
  BSAL-80/85.2.3  78/11/26  13:08:58
DEC/ADD           ENTRY     #0047
                    6   123
DEC/SUB           ENTRY     #0059
                    6   157
LARGER            ENTRY     #0008
                    5    33
LOOOP             LABEL     #005C
                  160   174
LOOP              LABEL     #004A
                  126   136
NEXT LSD          LABEL     #000F
                   39    51
SHIFT             LABEL     #002B
                   82    99
TRANS DATA LEFT   ENTRY     #0023
                    5    74
ZERO/MEM          ENTRY     #0000
                    5    14    17
  BSAL-80/85.2.3  78/11/26  13:09:22
```

```
 BSAL-80 05.2.3   78/11/27  15:11:39
          *****   0 ERRORS.   0 WARNINGS. SEE    0 *****
 SOURCE FILE =   DK2:WALT/UTIL1.EDT-0
 OBJECT FILE =   DK2:WALT UTIL1.OBJ-0
00001.000    1 1 01AC
00002.000    2 1 0000          $CONTROL          NAME=UTIL1
00003.000    3 1 0000          $CONTROL          LIST,INNERLIST,TABLE,CROSS
00004.000    4 1 0000          !$CONTROL          NOLIST
00004.010    5 1 0000
00004.020    6 1 0000          ENTRY   BIN/TO/BCD,    INT/TO/FP,     FP/OUT/FMT,
00004.030    7 1 0000                  FP/TO/INT,     BUFFER/IN;
00004.040    8 1 0000
00004.050    9 1 0000          GLOBAL  TEMP2(9);
00004.060   10 1 0000
00004.070   11 1 0000          EXTERNAL  ZERO/MEM,    TRANS/DATA,    LEFT/JUST,
00004.080   12 1 0000                    ERROR/OUT,     TRANS/DATA/LEFT,
00004.081   13 1 0000                    INPUT/BUFF,  BUFF/COUNT;
00004.090   14 1 0000
00004.100   15 1 0000          EQUATE  POS/TRU/POLARY=0,  ALL/DEC/PTS=%.001111111;
00004.110   16 1 0000          EQUATE  NEG/SIGN=0,        POS/SIGN=%.10000000;
00004.120   17 1 0000
00005.000   18 1 0000          <<
00006.000   19 1 0000                  REF INTEL USER'S LIBRARY PROGRAM MANUAL,
00007.000   20 1 0000                  APRIL 1977, PROGRAM BB18 BY GUNDESTRUP,
00007.100   21 1 0000                  NIELS O.
00008.000   22 1 0000
00009.000   23 1 0000                  MODIFIED FOR USE AS A BSAL SUBROUTINE
00010.000   24 1 0000
00011.000   25 1 0000          ********    BINARY TO BCD CONVERSION     *********
00014.000   26 1 0000
00015.000   27 1 0000                  INPUTS: UNSIGNED BINARY NUMBER IN D,E
00017.000   28 1 0000                          POINTER TO LOWEST BUFFER LOC IN HL
00018.000   29 1 0000
00019.000   30 1 0000                  OUTPUT: 5 BCD DIGITS, TWO DIGITS / MEM LOC
00020.000   31 1 0000                          HL POINT TO LSD IN LOWEST LOCATION
00021.000   32 1 0000
00022.000   33 1 0000          >>
00025.000   34 1 0000
00025.100   35 1 0000          BIN TO BCD:
00026.000   36 1 0000 F5              TOS=AFLAGS;
00027.000   37 1 0001 C5              TOS=BC;
00028.000   38 1 0002 D5              TOS=DE;
00029.000   39 1 0003 E5              TOS=HL;
00030.000   40 1 0004 EB              DE<==>HL;     ! BIN # IN HL, ADDRESS IN DE
00031.000   41 1 0005 1313            DE=DE+1;DE=DE+1;  ! POINT TO MSB
00032.000   42 1 0007 01F0D8          BC=-10000;
00033.000   43 1 000A CD3600          CALL DECIMAL/NUMB;  ! FIND MOST SIG DIGIT
00034.000   44 1 000D 12              MEM(DE)=A;          ! STORE DIGIT
00035.000   45 1 000E 1B              DE=DE-1;            ! DECREMENT POINTER
00036.000   46 1 000F 0118FC          BC=-1000;
00037.000   47 1 0012 CD3600          CALL DECIMAL/NUMB;  ! FIND SECOND MSD
00038.000   48 1 0015 070707          RLC;RLC;RLC;RLC;
            1 0018 07
00039.000   49 1 0019 12              MEM(DE)=A;
00040.000   50 1 001A 019CFF          BC=-100;
```

```
00041.000   51 1 001D CD3600P          CALL DECIMAL'NUMB;  ! FIND THIRD MSD
00042.000   52 1 0020 EB               DE<==>HL;
00043.000   53 1 0021 B6               A=A OR MEM(HL);
00044.000   54 1 0022 EB               DE<==>HL;
00045.000   55 1 0023 12               MEM(DE)=A;          ! STORE SECOND BYTE
00046.000   56 1 0024 1B               DE=DE-1;            ! DECREMENT POINTER
00047.000   57 1 0025 01F6FF           BC=-10;
00048.000   58 1 0028 CD3600P          CALL DECIMAL'NUMB;  ! FIND FOURTH MSD
00049.000   59 1 002B 070707           RLC;RLC;RLC;RLC;
            1 002E 07
00050.000   60 1 002F B5               A=A OR L;           ! MSD & LSD
00051.000   61 1 0030 12               MEM(DE)=A;          ! STORE LAST BYTE
00052.000   62 1 0031 E1               HL=TOS;
00053.000   63 1 0032 D1               DE=TOS;
00054.000   64 1 0033 C1               BC=TOS;
00055.000   65 1 0034 F1               AFLAGS=TOS;
00056.000   66 1 0035 C9               RETURN;
00057.000   67 1 0036
00057.100   68 1 0036          DECIMAL'NUMB:
00058.000   69 1 0036 AF               A=A XOR A;  ! ZERO ACCUMULATOR
00059.000   70 1 0037 D5               TOS=DE;     ! SAVE NEXT DIGIT ADDRESS
00060.000   71 1 0038 5D               E=L;
00061.000   72 1 0039 54               D=H;
00062.000   73 1 003A 3C               A=A+1;      ! INCREMENT DIGIT
00063.000   74 1 003B 09               HL=HL+BC;   ! SUBTRACT 10**X. (X=1,2,...5)
00064.000   75 1 003C DA3800P          IF CARRY=1 THEN GOTO DECIMAL'NUMB(2);
00065.000   76 1 003F                              ! RESULT IS NEGATIVE
00066.000   77 1 003F 3D               A=A-1;      ! YES. RESTORE DIGIT COUNT
00067.000   78 1 0040 6B               L=E;        ! BINARY NUMBER
00068.000   79 1 0041 62               H=D;
00069.000   80 1 0042 D1               DE=TOS;     ! SAVED ADDRESS
00070.000   81 1 0043 C9               RETURN;
00071.000   82 1 0044
00072.000   83 1 0044          INT'TO'FP:
00072.100   84 1 0044          <<
00072.200   85 1 0044          ********    INTEGER TO FLOATING POINT    *********
00072.300   86 1 0044
00073.000   87 1 0044                    CONVERTS A POSITIVE INTEGER (2 DIGITS PER
00074.000   88 1 0044                     BYTE) TO A FLOATING POINT NUMBER.
00075.000   89 1 0044
00076.000   90 1 0044                      INPUT   B   BYTES IN INTEGER
00077.000   91 1 0044                              DE  POINTER TO INTEGER
00078.000   92 1 0044                              HL  DESTINATION REG(9 BYTES)
00079.000   93 1 0044
00080.000   94 1 0044                      OUTPUT  HL  POINTER TO OUTPUT FLOATING
00081.000   95 1 0044                                    POINT NUMBER
00082.000   96 1 0044                                                          >>
00083.000   97 1 0044 AF               A=A XOR A;      ! 0
00084.000   98 1 0045 C5               TOS=BC;
00085.000   99 1 0046 E5               TOS=HL;
00086.000  100 1 0047 0609             B=9;
00087.000  101 1 0049 CD0000X          CALL ZERO MEM;    ! ZERO DESTINATION REG
00088.000  102 1 004C E1               HL=TOS;
```

```
00089.000  103 1 004D 23            HL=HL+1;
00090.000  104 1 004E C1            BC=TOS;
00091.000  105 1 004F E5            TOS=HL;
00092.000  106 1 0050 CD0000X       CALL TRANS'DATA;  ! MOVE INTEGER INTO FP #
00092.100  107 1 0053 E1            HL=TOS;
00093.000  108 1 0054 E5            TOS=HL;
00094.000  109 1 0055 0608          B=8;
00095.000  110 1 0057 CD0000X       CALL LEFT'JUST;  <<B=# DIGITS SHIFTED;
00095.100  111 1 005A                     IF NUMBER=0, B=#FF.          >>
00096.000  112 1 005A E1            HL=TOS;
00097.000  113 1 005B 2B            HL=HL-1;          ! SIGN & EXPON BYTE
00098.000  114 1 005C 04            B=B+1;
00099.000  115 1 005D C8            IF ZERO THEN RETURN; ! # IS 0
00100.000  116 1 005E 3E11          A=16+1;
00101.000  117 1 0060 90            A=A-B;            ! EXPONENT
00102.000  118 1 00e1 77            MEM(HL)=A;
00103.000  119 1 0062 C9            RETURN;
00104.000  120 1 00e3
00104.100  121 1 00e3               <<
00104.200  122 1 00e3               ♦♦♦♦♦   FLOATING POINT TO OUTPUT FORMAT    ♦♦♦♦♦♦♦
00104.300  123 1 00e3                                                          >>
00105.000  124 1 00e3               FP'OUT FMT:
00106.000  125 1 00e3                     <<FORMATS A FLOATING POINT NUMBER TO REAL
00107.000  126 1 00e3                     FORMAT READY FOR OUTPUT ON THE DIGITAL
00108.000  127 1 00e3                     PANEL PRINTER.  THE SIGN, POLARITY &
00109.000  128 1 00e3                     DECIMAL POINTS ARE LOADED INTO THE B REG.
00110.000  129 1 00e3                     THE FOUR PLACE NUMBER IS LOADED INTO THE
00111.000  130 1 00e3                     DE REG.  THE C REG MUST BE LOADED EXTERN-
00112.000  131 1 00e3                     ALLY FOR CHANNEL # AND I.D.  IF A NUMBER
00113.000  132 1 00e3                     IS OVER OR UNDER THE REAL NUMBER RANGE,
00114.000  133 1 00e3                     9999 AND .0001 ARE USED FOR OUTPUT,
00115.000  134 1 00e3                     RESPECTIVELY.
00116.000  135 1 00e3
00117.000  136 1 00e3                       INPUT  HL  POINTER TO FP NUMBER
00118.000  137 1 00e3                                                        >>
00119.000  138 1 00e3 7E            A=MEM(HL);
00120.000  139 1 00e4 17            RAL;
00121.000  140 1 00e5               IF CARRY=0 THEN
00122.000  141 1 00e5 DA6D00F           B=POS'TRU'POLARY OR POS'SIGN
00123.000  142 1 00e8 0630             ELSE
00124.000  143 1 00eA C36F00F           B=POS'TRU'POLARY OR NEG'SIGN;
           1 00eD 0600
00125.000  144 1 00eF
00126.000  145 1 00eF A7            << CHECK FOR OVERFLOW/UNDERFLOW >>
00127.000  146 1 0070               A=A AND A;
00128.000  147 1 0070 FA8000F       IF POSITIVE      ! POSITIVE EXPON
00128.900  148 2 0073 1F              THEN BEGIN     ! CHK FOR OVERFLOW
00130.000  149 2 0074 FE05              RAR;
00131.000  150 2 0076                   A-5;
00132.000  151 2 0076 FA7D00F           IF POSITIVE ! OVERFLOW
00133.000  152 3 0079 119999              THEN BEGIN
00134.000  153 3 007C C9                    DE=#9999;
00135.000  154 2 007D                      RETURN;
00136.000  155 2 007D                    END;
                                      END
```

```
00137.000  156 1 007D C38F00F
00138.000  157 2 0080 37
00139.000  158 2 0081 1F
00140.000  159 2 0082 FEFD
00141.000  160 2 0084
00142.000  161 2 0084 F28F00F
00143.000  162 3 0087 110000
00144.000  163 3 008A 78
00145.000  164 3 008B E608
00146.000  165 3 008D 47
00147.000  166 3 008E C9
00148.000  167 2 008F
00149.000  168 1 008F
00150.000  169 1 008F
00151.000  170 1 008F F5
00152.000  171 1 0090 110700
00153.000  172 1 0093 19
00154.000  173 1 0094 5E
00155.000  174 1 0095 23
00156.000  175 1 0096 56
00157.000  176 1 0097 F1
00158.000  177 1 0098
00159.000  178 1 0098 FE04
001e0.000  179 1 009A C29F00F
                1 009D 0E00
001e1.000  180 1 009F FE03
001e2.000  181 1 00A1 C2AE00F
                1 00A4 0E01
001e3.000  182 1 00A6 FE02
001e4.000  183 1 00A8 C2AD00F
                1 00AB 0E02
001e5.000  184 1 00AD FE01
001e6.000  185 1 00AF C2B400F
                1 00B2 0E04
001e7.000  186 1 00B4 F2B900F
                1 00B7 0E08
001e8.000  187 1 00B9 F5
001e9.000  188 1 00BA 78
00170.000  189 1 00BB E1
00171.000  190 1 00BC 47
00172.000  191 1 00BD F1
00173.000  192 1 00BE FE00
00174.000  193 1 00C0 F0
00175.000  194 1 00C1 FEFF
00176.000  195 1 00C3
00177.000  196 1 00C3 C2E000F
00178.000  197 2 00C6 7B
00179.000  198 2 00C7 E6F0
00180.000  199 2 00C9 0F0F0F
                2 00CC 0F
00181.000  200 2 00CD 5F
00182.000  201 2 00CE 7A
00183.000  202 2 00CF E60F
00184.000  203 2 00D1 070707
                2 00D4 07
```

```
  ELSE BEGIN    ! NEGATIVE EXPON
    CARRY=1;
    PAR;
    A-:-3:;
    IF NEGATIVE ! UNDERFLOW
      THEN BEGIN
        DE=#0000;
        A=B;
        A=A OR %00001000;
        B=A;
        RETURN;
      END;
  END;
<< LOAD DE REG >>
TOS=AFLAGS;     ! SAVE EXPON
DE=7;
HL=HL+DE;
E=MEM:HL:;
HL=HL+1;
D=MEM:HL:;
AFLAGS=TOS;
<< SET DECIMAL PLACE >>
A-4;
IF ZERO THEN C=%00000000;

A-3;
IF ZERO THEN C=%00000001;

A-2;
IF ZERO THEN C=%00000010;

A-1;
IF ZERO THEN C=%00000100;

IF NEGATIVE THEN C=%00001000;

TOS=AFLAGS;
A=B;
A=A OR C;  ! COMBINE SIGN+DEC PT
B=A;
AFLAGS=TOS;
A-0;
IF POSITIVE THEN RETURN;
A-:-1:;
IF ZERO
  THEN BEGIN ! SHIFT DE ONE DIGIT RT
    A=E;
    A=A AND #F0;
    RRC;RRC;RRC;RRC;

    E=A;
    A=D;
    A=A AND #0F;
    RLC;RLC;RLC;RLC;
```

```
00185.000  204 2 00D5 B3              A=A OR E;
00186.000  205 2 00D6 5F              E=A;
00187.000  206 2 00D7 7A              A=D;
00188.000  207 2 00D8 E6F0            A=A AND #F0;
00189.000  208 2 00DA 0F0F0F          RRC;RRC;RRC;RRC;
           2 00DD 0F
00190.000  209 2 00DE 57              D=A;
00191.000  210 2 00DF C9              RETURN;
00192.000  211 1 00E0                 END;
00193.000  212 1 00E0 FEFE          A=-2;
00194.000  213 1 00E2                 IF ZERO
00195.000  214 1 00E2 C2E900F         THEN BEGIN ! SHIFT DE 2 DIGITS RT
00196.000  215 2 00E5 5A                E=D;
00197.000  216 2 00E6 1600              D=0;
00198.000  217 2 00E8 C9                RETURN;
00199.000  218 1 00E9                 END;
00200.000  219 1 00E9 FEFD          A=-3;
00201.000  220 1 00EB                 IF ZERO
00202.000  221 1 00EB C2F800F         THEN BEGIN ! SHIFT DE 3 DIGITS RT
00203.000  222 2 00EE 7A                A=D;
00204.000  223 2 00EF E6F0              A=A AND #F0;
00205.000  224 2 00F1 0F0F0F            RRC;RRC;RRC;RRC;
           2 00F4 0F
00206.000  225 2 00F5 5F                E=A;
00207.000  226 2 00F6 1600              D=0;
00208.000  227 1 00F8                 END;
00209.000  228 1 00F8 C9            RETURN;
00210.000  229 1 00F9
00210.100  230 1 00F9
00210.200  231 1 00F9           ******* FLOATING POINT TO INTEGER **********
00210.300  232 1 00F9                                                    >>
00211.000  233 1 00F9           FP TO INT:
00212.000  234 1 00F9                   CONVERTS A POSITIVE FP NUMBER TO
00213.000  235 1 00F9                 INTEGER FORM
00214.000  236 1 00F9
00215.000  237 1 00F9                   INPUT:  B   # BYTES IN INTEGER
00216.000  238 1 00F9                           DE  POINTER TO FP #
00217.000  239 1 00F9                           HL  POINTER TO INTEGER LOC
00218.000  240 1 00F9
00219.000  241 1 00F9                   OUTPUT  B   # BYTES IN INTEGER
00220.000  242 1 00F9                           HL  INTEGER POINTER
00221.000  243 1 00F9
00222.000  244 1 00F9                 OUTPUTS AN ERROR MSG OF .6.6+.5.5.5.5
00223.000  245 1 00F9                 IF THERE IS AN OVERFLOW IN CONVERSION >>
00224.000  246 1 00F9
00225.000  247 1 00F9 C5              TOS=BC;
00226.000  248 1 00FA E5              TOS=HL;
00227.000  249 1 00FB D5              TOS=DE;
00228.000  250 1 00FC 48              C=B;
00229.000  251 1 00FD CD0000          CALL ZERO MEM; ! ZERO INTEGER LOC
00230.000  252 1 0100                   CHECK FOR NEGATIVE OR 0 EXPON
00231.000  253 1 0100 EB              HL == DE;
00232.000  254 1 0101 7E              A=MEM<HL>;      ! EXPON
00233.000  255 1 0102 17              RAL;
```

```
00237.000  256 1 0103 A7          A=A AND A;
00238.000  257 1 0104             IF ZERO
00239.000  258 1 0104 C20B01F       THEN BEGIN
00240.000  259 2 0107 D1              DE=TOI;
00241.000  260 2 0108 E1              HL=TOI;
00242.000  261 2 0109 C1              BC=TOI;
00243.000  262 2 010A C9              RETURN;
00244.000  263 1 010B               END;
00245.000  264 1 010B 41          B=C;     ! # BYTES IN INTEGER
00246.000  265 1 010C 1F          PAR;     ! SAVE EXPON
00247.000  266 1 010D 4F          C=A;
00248.000  267 1 010E A7          A=A AND A;       ! ZERO CARRY
00249.000  268 1 010F 78          A=B;
00250.000  269 1 0110 17          RAL; !  MULT BY 2; = DIGITS IN INTEGER
00251.000  270 1 0111 B9          A-C;
00252.000  271 1 0112             IF NEGATIVE      ! OVERFLOW
00253.000  272 1 0112 F22301F       THEN BEGIN     ! OUTPUT ERROR
00254.000  273 2 0115                 B=POS SIGN OF POS TRU POLARY
00255.000  274 2 0115 06FF                       OF ALL DEC PTS;
00256.000  275 2 0117 0E44              C=#44;
00257.000  276 2 0119 114444            DE=#4444;
00258.000  277 2 011C CD0000            CALL ERROR OUT;
00259.000  278 2 011F D1                DE=TOI;
00260.000  279 2 0120 E1                HL=TOI;
00261.000  280 2 0121 C1                BC=TOI;
00262.000  281 2 0122 C9                RETURN;
00263.000  282 1 0123               END;
00264.000  283 1 0123 23          HL=HL+1;    ! POINT TO MANTISSA OF FP #
00265.000  284 1 0124 47          B=A;        ! # DIGITS IN INTEGER
00266.000  285 1 0125 79          A=C;
00267.000  286 1 0126 1F          PAR;        ! EXPON DIV BY 2
00268.000  287 1 0127 F5          TOI=AFLAGS;
00269.000  288 1 0128             IF CARRY=1  ! EXPON IS ODD
00270.000  289 1 0128 D2340F1       THEN BEGIN
00271.000  290 2 012B EB              DE == HL;
00272.000  291 2 012C 0608            B=8;
00273.000  292 2 012E 210000G         HL=@TEMP2;
00274.000  293 2 0131 CD0000          CALL TRANS DATA LEFT;
00275.000  294 2 0134 F1              AFLAGS=TOI;
00276.000  295 2 0135 3C              A=A+1;
00277.000  296 2 0136               END
00278.000  297 1 0136 C34301F       ELSE BEGIN
00279.000  298 2 0139 EB              DE == HL;
00280.000  299 2 013A 0608            B=8;
00281.000  300 2 013C 210100G         HL=@TEMP2+1;
00282.000  301 2 013F CD0000          CALL TRANS DATA;
00283.000  302 2 0142 F1              AFLAGS=TOI;
00284.000  303 1 0143               END;
00285.000  304 1 0143 47          B=A;     ! # BYTES TO BE TRANS TO INTEGER
00286.000  305 1 0144 110400G     DE=@TEMP2+10;
00287.000  306 1 0147 EEFF        A=A XOR #FF;     ! COMPLEMENT A
00288.000  307 1 0149 4F          C=A;
00289.000  308 1 014A 26FF        H=#FF;
```

```
00290.000  309 1 0140 19          HL=HL+DE;
00291.000  310 1 014D EB          DE<==>HL;  ! INDEX @ START OF INTEGER
00292.000  311 1 014E                        ! IN TEMP2
00293.000  312 1 014E E1          HL=TOS;
00294.000  313 1 014F E1          HL=TOS;
00295.000  314 1 0150 E5          TOS=HL;
00296.000  315 1 0151 CD0000X     CALL TRANS'DATA;
00297.000  316 1 0154 E1          HL=TOS;
00298.000  317 1 0155 C1          BC=TOS;
00299.000  318 1 0156 C9          RETURN;
00300.000  319 1 0157
00300.100  320 1 0157        <<
00300.200  321 1 0157        ◆◆◆◆◆◆◆◆◆◆◆◆◆   BUFFER INPUT   ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
00300.300  322 1 0157                                                    >>
00301.000  323 1 0157        BUFFER'IN:
00302.000  324 1 0157                << CONVERTS 4 DIGITS OF INPUT DATA FROM
00303.000  325 1 0157                THE KYBD TO FLOATING PT AND STORES IT.
00304.000  326 1 0157                THE 4 DIGITS ARE UNPACKED(1/BYTE).  THE
00305.000  327 1 0157                INPUT BUFFER AND BUFF'COUNT ARE ZEROED
00306.000  328 1 0157
00307.000  329 1 0157                  INPUTS    B   # DIGITS IN BUFFER
00308.000  330 1 0157                            DE  DESTINATION REGISTER
00309.000  331 1 0157                            HL  ONE BELOW START ADDR
00310.000  332 1 0157                                  IN BUFFER
00311.000  333 1 0157                  OUTPUT    HL  @ DESTINATION FP REG
00312.000  334 1 0157                                                >>
00313.000  335 1 0157 D5          TOS=DE;
00314.000  336 1 0158 480600      C=B;B=0;
00315.000  337 1 015B 1600        D=0;
00316.000  338 1 015D 09          HL=HL+BC;  ! POINT TO LSD
00317.000  339 1 015E 7E          A=MEM(HL);
00318.000  340 1 015F EEFF        A=A XOR #FF; ! COMPLEMENT NUMBER
00319.000  341 1 0161 E60F        A=A AND #0F;
00320.000  342 1 0163 5F          E=A;
00321.000  343 1 0164 0D          C=C-1;
00322.000  344 1 0165 CA8F01F     IF ZERO THEN GOTO CHANGE'TO'FP;
00323.000  345 1 0168 2B          HL=HL-1;
00324.000  346 1 0169 7E          A=MEM(HL);
00325.000  347 1 016A EEFF        A=A XOR #FF;
00326.000  348 1 016C E60F        A=A AND #0F;
00327.000  349 1 016E 070707      RLC;RLC;RLC;RLC;
           1 0171 07
00328.000  350 1 0172 B3          A=A OR E;
00329.000  351 1 0173 5F          E=A;
00330.000  352 1 0174 0D          C=C-1;
00331.000  353 1 0175 CA8F01F     IF ZERO THEN GOTO CHANGE'TO'FP;
00332.000  354 1 0178 2B          HL=HL-1;
00333.000  355 1 0179 7E          A=MEM(HL);
00334.000  356 1 017A EEFF        A=A XOR #FF;
00335.000  357 1 017C E60F        A=A AND #0F;
00336.000  358 1 017E 57          D=A;
00337.000  359 1 017F 0D          C=C-1;
00338.000  360 1 0180 CA8F01F     IF ZERO THEN GOTO CHANGE TO'FP;
00339.000  361 1 0183 2B          HL=HL-1;
00340.000  362 1 0184 7E          A=MEM(HL);
```

```
00341.000   363 1 0195 EEFF            A=A XOR #FF;
00342.000   364 1 0197 E60F            A=A AND #0F;
00343.000   365 1 0199 070707          RLC;RLC;RLC;RLC;
            1 019C 07
00344.000   366 1 019D B2              A=A OR D;
00345.000   367 1 019E 57              D=A;
00346.000   368 1 019F          CHANGE TO FP;
00347.000   369 1 019F EB              HL <=> DE;
00348.000   370 1 0190 220000          INPUT BUFF=HL;  ! SAVE 4 DIGIT(2 BYTE)
00349.000   371 1 0193 110000          DE=@INPUT BUFF;
00350.000   372 1 0196 0602            B=2;
00351.000   373 1 0198 E1              HL=TOS;
00352.000   374 1 0199 CD4400          CALL INT TO FP; ! HL=0 FP NUMBER
00353.000   375 1 019C E5              TOS=HL;
00354.000   376 1 019D                 << ZERO INPUT BUFF & BUFF COUNT>>
00355.000   377 1 019D 0606            B=6;
00356.000   378 1 019F 2A0000          HL=INPUT BUFF;
00357.000   379 1 01A2 CD0000          CALL ZERO MEM;
00358.000   380 1 01A5 3E00            A=0;
00359.000   381 1 01A7 320000          BUFF COUNT=A;
00360.000   382 1 01AA E1              HL=TOS;
00361.000   383 1 01AB C9              RETURN;
00362.000   384 0 01AC                 END.
            ***** 0 ERROR(S), 0 WARNING(S), SIZE 0 *****
  BCAL-80 05.2.3   78 11 27  15:27:03
ALL DEC PT          EQUATE    #007F
                      15   274
BIN TO BCD          ENTRY     #0000
                       6    35
BUFF COUNT          EXTERNAL  #0010
                      13   381
BUFFER IN           ENTRY     #0157
                       7   383
CHANGE TO FP        LABEL     #018F
                     369   344   353   360
DECIMAL NUMB        LABEL     #003E
                      68    75    43    47    51
                      58
ERROR OUT           EXTERNAL  #000D
                      12   277
FP OUT FMT          ENTRY     #00E3
                       6   124
FP TO INT           ENTRY     #00F9
                       7   233
INPUT BUFF          EXTERNAL  #000F
                      13   370   371   378
INT TO FP           ENTRY     #0044
                       6    83   374
LEFT JUST           EXTERNAL  #000C
                      11   110
NEG SIGN            EQUATE    #0000
                      16   143
POS SIGN            EQUATE    #0020
                      16   141   273
POS TPU POLARY      EQUATE    #0000
                      15   141   143   273
```

```
TEMP2                   GLOBAL    #0000
                          9   293   300   305
TRANC DATA              EXTERNAL  #000B
                          11  106   301   315
TRANC DATA LEFT         EXTERNAL  #000E
                          12  293
ZERO MEM                EXTERNAL  #000A
                          11  101   251   379
  BIAL-80 85.2.3  78 11/27  15:27:56
```

```
 DIAL-80 05.2.3   78 11/27  11:00:23
          ***** 0 ERRORS,  0 WARNING(S), SEE    0 *****
 SOURCE FILE =   DK2:WALT_UTIL2.EDT-0
 OBJECT FILE =   DK2:WALT_UTIL2.OBJ-0
00001.000    1 1 0101
00002.000    2 1 0000          $CONTROL        NAME=UTIL2
00003.000    3 1 0000          !$CONTROL        NOLIST
00004.000    4 1 0000          $CONTROL        LIST,INHEPLIST,TABLE,CROSS
00005.000    5 1 0000
00006.000    6 1 0000          ENTRY   COMPARE,   MEAN,   STD_DEV;
00007.000    7 1 0000
00008.000    8 1 0000          EXTERNAL  FP_TO_INT,  ZERO_MEM,  LARGER,
00009.000    9 1 0000                    INT_TO_FP,  DIVIDE,    MULT,
00010.000   10 1 0000                    FP_ADD,     SQRT,      FP_NUM,
00011.000   11 1 0000                    SUM,        SUM_OF_SQUARES,
00011.100   12 1 0000                    BIN_TO_BCD;
00012.000   13 1 0000
00013.000   14 1 0000          GLOBAL  SQUARE(9),
00014.000   15 1 0000                  NUM_REMAINING,
00015.000   16 1 0000                  SAMPLE_SIZE,
00017.000   17 1 0000                  @N_MINUS_1=@NUM_REMAINING,
00022.000   18 1 0000                  UP_LIM_INT(3),  LOW_LIM_INT(3);
00023.000   19 1 0000
00024.000   20 1 0000          <<
00024.100   21 1 0000          ************    COMPARISON    ********************
00024.200   22 1 0000                                                        >>
00025.000   23 1 0000          COMPARE:
00026.000   24 1 0000                  ;COMPARES READINGS WITH MEAN + -
00027.000   25 1 0000                2 STD DEVIATIONS.  IF OUTSIDE THIS
00028.000   26 1 0000                LIMIT, READINGS ARE ANNOTATED WITH
00029.000   27 1 0000                A ONE IN THE MOST SIGNIFICANT BIT.
00030.000   28 1 0000
00031.000   29 1 0000                  INPUTS:  A   BYTES PER READING
00032.000   30 1 0000                           BC  POINTER TO # OF RDGS
00033.000   31 1 0000                                 IN LIST
00034.000   32 1 0000                           DE  MEAN LOCATION
00035.000   33 1 0000                           HL  STD DEV LOCATION
00036.000   34 1 0000                                                    ;;
00037.000   35 1 0000
00038.000   36 1 0000          !       FIND THE UPPER AND LOWER LIMITS IN
00039.000   37 1 0000          !       FLOATING POINT.  CHK FOR LOWER LIM<0.
00040.000   38 1 0000 C5               TOS=BC;
00041.000   39 1 0001 F5               TOS=AFLAGS;
00042.000   40 1 0002 D5               TOS=DE;
00043.000   41 1 0003 E5               TOS=HL;
00044.000   42 1 0004 010000%          BC=@SUM; ! CALC UPPER LIMIT @SUM
00045.000   43 1 0007 CD0000x          CALL FP_ADD;  ! HL=@UPPER_LIMIT
00046.000   44 1 000A 4D44             C=L;B=H;
00047.000   45 1 000C D1               DE=TOS;       ! STD DEV
00048.000   46 1 000D D5               TOS=DE;
00049.000   47 1 000E CD0000x          CALL FP_ADD;  ! HL=@UPPER_LIMIT(2 SIGMA)
00050.000   48 1 0011 E1               HL=TOS;
00051.000   49 1 0012 D1               DE=TOS;
00052.000   50 1 0013 D5               TOS=DE;
```

```
00053.000   51 1 0014 E5              TOS=HL;
00054.000   52 1 0015 7E              A=MEM(HL);
00055.000   53 1 0016 17              RAL;            ! CHANGE SIGN OF STD DEV
00056.000   54 1 0017 37              CARRY=1;
00057.000   55 1 0018 1F              RAR;
00058.000   56 1 0019 77              MEM(HL)=A;      ! CALC LOWER LIMIT
00059.000   57 1 001A 010000'         BC=@SUM OF SQUARES; ! @ LOWER LIMIT
00060.000   58 1 001D CD0000'         CALL FP ADD;    ! HL=@LOWER LIMIT
00061.000   59 1 0020 D1              DE=TOS;
00062.000   60 1 0021 D5              TOS=DE;
00063.000   61 1 0022 4D44            C=L;B=H;
00064.000   62 1 0024 CD0000'         CALL FP ADD;
00065.000   63 1 0027 E1              HL=TOS;         ! RTN POS SIGN TO STD DEV
00066.000   64 1 0028 7E              A=MEM(HL);
00067.000   65 1 0029 17              RAL;
00068.000   66 1 002A A7              A=A AND A;      ! CLEAR CARRY
00069.000   67 1 002B 1F              RAR;
00069.100   68 1 002C 77              MEM(HL)=A;
00070.000   69 1 002D                 << IF LOWER LIM LESS THAN 0, SET = 0 >>
00071.000   70 1 002D 210000'         HL=@SUM OF SQUARES; ! @ LOWER LIMIT
00072.000   71 1 0030 7E              A=MEM(HL);
00073.000   72 1 0031 A7              A=A AND A;
00074.000   73 1 0032                 IF NEGATIVE
00075.000   74 1 0032 F23A00F          THEN BEGIN
00076.000   75 2 0035 0600               B=0;
00077.000   76 2 0037 CD0000'            CALL ZERO MEM;
00078.000   77 1 003A                   END;
00079.000   78 1 003A D1              DE=TOS;         ! MEAN LOCATION
00080.000   79 1 003B                 << PUT UPPER & LOWER LIMIT INTO INTEGER
00081.000   80 1 003B                    FORM                                >>
00082.000   81 1 003B F1              AFLAG=TOS;      ! BYTES PER RDG
00083.000   82 1 003C F5              TOS=AFLAG;
00084.000   83 1 003D 47              B=A;
00085.000   84 1 003E 110000'        DE=@SUM;        ! @ UPPER LIMIT
00086.000   85 1 0041 210E006        HL=@UP LIM INT;
00087.000   86 1 0044 CD0000'        CALL FP TO INT;  ! B= BYTES IN INTEGER
00088.000   87 1 0047 110000'        DE=@SUM OF SQUARES; ! @ LOWER LIMIT
00089.000   88 1 004A 210E006        HL=@LOW LIM INT;
00090.000   89 1 004D CD0000'        CALL FP TO INT;  ! B= BYTES IN INTEGER
00091.000   90 1 0050                 << SCAN LIST & COMPARE WITH UPPER AND
00092.000   91 1 0050                    LOWER LIMITS                        >>
00093.000   92 1 0050 C1              BC=TOS;         ! BYTES PER RDG
00094.000   93 1 0051 E1              HL=TOS;
00095.000   94 1 0052 7E              A=MEM(HL);
00096.000   95 1 0053 320900G         NUM REMAINING=A;
00097.000   96 1 0056 23              HL=HL+1;
00098.000   97 1 0057 480600          C=B;B=0;
00099.000   98 1 005A E5              TOS=HL;         ! POINTER TO 1ST LOC IN LIST
00100.000   99 1 005B C5              TOS=BC;         ! BYTES PER RDG IN C
00101.000  100 1 005C                 COMPARE ANOTHER:
00102.000  101 1 005C                 << COMPARE WITH UPPER LIMIT          >>
00103.000  102 1 005C 41              B=C;
00104.000  103 1 005D 110E006        DE=@UP LIM INT;
00105.000  104 1 0060 CD0000'        CALL LARGER;    ! B=1 IF NUM > UPPER LIMIT
```

```
00106.000  105 1 0063 05              B=B-1;
00107.000  106 1 0064                 IF ZERO
00108.000  107 1 0064 C27500F           THEN BEGIN
00109.000  108 2 0067           SET'FLAG:
00110.000  109 2 0067 C1                  BC=TOS;
00111.000  110 2 0068 E1                  HL=TOS;
00112.000  111 2 0069 09                  HL=HL+BC;
00113.000  112 2 006A 2B                  HL=HL-1;
00114.000  113 2 006B 3E80                A='10000000;
00115.000  114 2 006D B6                  A=A OR MEM(HL);
00116.000  115 2 006E 77                  MEM(HL)=A;
00117.000  116 2 006F 23                  HL=HL+1;
00118.000  117 2 0070 E5                  TOS=HL;
00119.000  118 2 0071 C5                  TOS=BC;
00120.000  119 2 0072 C38A00F             GOTO CHECK'FOR'END;
00121.000  120 1 0075                   END;
00122.000  121 1 0075                 << COMPARE WITH LOWER LIMIT     >>
00123.000  122 1 0075 110E006         DE=#LOW LIM INT;
00124.000  123 1 0078 C1              BC=TOS;
00125.000  124 1 0079 E1              HL=TOS;
00126.000  125 1 007A E5              TOS=HL;
00127.000  126 1 007B C5              TOS=BC;
00128.000  127 1 007C 41              B=C;
00129.000  128 1 007D CD0000          CALL LARGER;  ! B=0 IF NUM < LOWER LIM
00130.000  129 1 0080 AF              A=A XOR A;
00131.000  130 1 0081 B8              A-B;
00132.000  131 1 0082 CA5700F         IF ZERO THEN GOTO SET'FLAG;
00133.000  132 1 0085                 << NUMBER WITHIN +/- 2 STD DEV   >>
00134.000  133 1 0085 C1              BC=TOS;
00135.000  134 1 0086 E1              HL=TOS;
00136.000  135 1 0087 09              HL=HL+BC;
00137.000  136 1 0088 E5              TOS=HL;
00138.000  137 1 0089 C5              TOS=BC;
00139.000  138 1 008A           CHECK FOR END:
00140.000  139 1 008A 3A0A006         A=NUM REMAINING;
00141.000  140 1 008D 3D              A=A-1;
00142.000  141 1 008E 320A006         NUM REMAINING=A;
00143.000  142 1 0091 C2500F          IF NONZERO THEN GOTO COMPARE'ANOTHER;
00144.000  143 1 0094 E1              HL=TOS;
00145.000  144 1 0095 C1              BC=TOS;
00146.000  145 1 0096 C9              RETURN;
00147.000  146 1 0097
00148.000  147 1 0097                 <<
00148.100  148 1 0097                 ********************     MEAN     **********************
00148.200  149 1 0097                                                                        >>
00149.000  150 1 0097           MEAN:
00150.000  151 1 0097                 << CALCULATES THE MEAN VALUE OF A LIST.
00151.000  152 1 0097                 ANY ELEMENTS WHICH ARE FLAGGED IN THE
00152.000  153 1 0097                 MOST SIGNIFICANT BIT ARE DISREGARDED.
00153.000  154 1 0097
00154.000  155 1 0097                   INPUTS:  B   BYTES PER BIG
00155.000  156 1 0097                            DE  MEAN LOCATION
00156.000  157 1 0097                            HL  LOC OF START OF LIST
00157.000  158 1 0097                                SHOWING # IN LIST
```

```
00158.000  159 1 0097
00159.000  160 1 0097                    OUTPUT  HL  @MEAN LOCATION
00160.000  161 1 0097                            FP'NUM CONTAINS THE SAMPLE
00161.000  162 1 0097                                 SIZE
00162.000  163 1 0097                                                    >>
00163.000  164 1 0097 3E00          A=0;
00164.000  165 1 0099 320A006       SAMPLE'SIZE=A;
00165.000  166 1 009C 480600        C=B;B=0;
00166.000  167 1 009F D5            TOS=DE;
00167.000  168 1 00A0 C5            TOS=BC;
00168.000  169 1 00A1 7E            A=MEM(HL);
00169.000  170 1 00A2 3209006       NUM'REMAINING=A;
00170.000  171 1 00A5 23            HL=HL+1;
00171.000  172 1 00A6 E5            TOS=HL;
00172.000  173 1 00A7               << ZERO SUMMATION REGISTER       >>
00173.000  174 1 00A7 0609          B=9;
00174.000  175 1 00A9 210000X       HL=@SUM;
00174.100  176 1 00AC CD0000X       CALL ZERO'MEM;
00175.000  177 1 00AF E1            HL=TOS;
00176.000  178 1 00B0 C1            BC=TOS;
00177.000  179 1 00B1 C5            TOS=BC;
00178.000  180 1 00B2 E5            TOS=HL;
00179.000  181 1 00B3          NEW NUMBER:
00180.000  182 1 00B3 09            HL=HL+BC;
00181.000  183 1 00B4 2B            HL=HL-1;     ! POINT TO FLAG
00182.000  184 1 00B5 7E            A=MEM(HL);
00183.000  185 1 00B6 A7            A=A AND A;
00184.000  186 1 00B7               IF POSITIVE  ! FLAG NOT SET
00185.000  187 1 00B7 FAD500F         THEN BEGIN
00186.000  188 2 00BA                    << CONVERT OF FP NUMBER  >>
00187.000  189 2 00BA D1                DE=TOS;
00188.000  190 2 00BB C1                BC=TOS;
00189.000  191 2 00BC C5                TOS=BC;
00190.000  192 2 00BD D5                TOS=DE;
00191.000  193 2 00BE 41                B=C;
00192.000  194 2 00BF 210000X           HL=@FP'NUM;
00193.000  195 2 00C2 CD0000X           CALL INT TO FP;  ! HL=@FP'NUM
00194.000  196 2 00C5 010000X           BC=@SUM;
00195.000  197 2 00C8 5059              D=B;E=C;
00196.000  198 2 00CA CD0000X           CALL FP'ADD;
00197.000  199 2 00CD                   << ADD 1 TO SAMPLE SIZE  >>
00198.000  200 2 00CD 3A0A006           A=SAMPLE'SIZE;
00199.000  201 2 00D0 3C                A=A+1;
00199.100  202 2 00D1 27                DAA;   ! DECIMAL FORMAT
00200.000  203 2 00D2 320A006           SAMPLE'SIZE=A;
00201.000  204 1 00D5                 END;
00202.000  205 1 00D5                 << DECREMENT COUNTER FOR NUM REMAINING
00203.000  206 1 00D5                IN LIST & INCREMENT LIST POINTER    >>
00204.000  207 1 00D5 E1            HL=TOS;
00205.000  208 1 00D6 C1            BC=TOS;
00206.000  209 1 00D7 09            HL=HL+BC;
00207.000  210 1 00D8 C5            TOS=BC;
00208.000  211 1 00D9 E5            TOS=HL;
```

```
00209.000  212 1 00DA 3A0900G          A=NUM'REMAINING;
00210.000  213 1 00DD 3D               A=A-1;
00211.000  214 1 00DE 320900G          NUM'REMAINING=A;
00212.000  215 1 00E1 C2B300F          IF NONZERO THEN GOTO NEW'NUMBER;
00213.000  216 1 00E4                  << CALCULATE THE MEAN                >>
00218.000  217 1 00E4 0601             B=1;
00219.000  218 1 00E6 110A00G          DE=@SAMPLE'SIZE;
00220.000  219 1 00E9 210000X          HL=@FP'NUM;
00221.000  220 1 00EC CD0000X          CALL INT'TO FP; ! HL=@FP'NUM
00222.000  221 1 00EF EB               DE<==>HL;
00223.000  222 1 00F0 210000X          HL=@SUM;
00224.000  223 1 00F3 C1               BC=TOS;
00225.000  224 1 00F4 C1               BC=TOS;
00226.000  225 1 00F5 C1               BC=TOS;    ! MEAN LOCATION
00227.000  226 1 00F6 CD0000X          CALL DIVIDE;  ! HL=@MEAN LOCATION
00228.000  227 1 00F9 C9               RETURN;
00229.000  228 1 00FA
00230.000  229 1 00FA          <<
00230.100  230 1 00FA          ♦♦♦♦♦♦♦♦♦♦♦♦    STANDARD DEVIATION    ♦♦♦♦♦♦♦♦♦♦♦♦♦
00230.200  231 1 00FA                                                       >>
00231.000  232 1 00FA          STD'DEV:
00232.000  233 1 00FA                  << CALCULATES THE STANDARD DEVIATION
00233.000  234 1 00FA                  OF THE FORM
00234.000  235 1 00FA
00236.000  236 1 00FA                                  2        _2
00237.000  237 1 00FA                          SUM(X )  -  N X
00238.000  238 1 00FA                   S =  --------------------
00239.000  239 1 00FA                                N - 1
00240.000  240 1 00FA
00241.000  241 1 00FA                    INPUTS:  A   BYTES PER RDG
00242.000  242 1 00FA                             DE  STD DEV LOC(9 BYTES)
00243.000  243 1 00FA                             BC  MEAN LOC(9 BYTES)
00244.000  244 1 00FA                             HL  LOC OF NUMBER OF RDGS
00245.000  245 1 00FA                                 & START OF LIST
00246.000  246 1 00FA
00247.000  247 1 00FA                    OUTPUT:  HL  @STD DEV
00248.000  248 1 00FA                                                       >>
00249.000  249 1 00FA D5               TOS=DE;    ! LOC OF STD DEV
00250.000  250 1 00FB C5               TOS=BC;    ! LOC OF MEAN
00251.000  251 1 00FC 4F0600          C=A;B=0;   ! BYTES PER RDG
00252.000  252 1 00FF 7E               A=MEM(HL);
00253.000  253 1 0100 3209006          NUM'REMAINING=A;
00257.000  254 1 0103 23               HL=HL+1;
00258.000  255 1 0104 C5               TOS=BC;    ! BYTES PER RDG
00259.000  256 1 0105 E5               TOS=HL;    ! POINTER TO CURRENT NUMBER
00259.100  257 1 0106 16005F           D=0;E=A;
00259.200  258 1 0109 210A006          HL=@SAMPLE'SIZE;
00259.300  259 1 010C CD0000X          CALL BIN TO'BCD;  ! SAMPLE SIZE BCD
00260.000  260 1 010F                  << ZERO SUMMATION REGISTERS >>
00261.000  261 1 010F 0609             B=9;
00262.000  262 1 0111 210000           HL=@SUM;
00263.000  263 1 0114 CD0000X          CALL ZERO MEM;
00264.000  264 1 0117 0609             B=9;
00265.000  265 1 0119 210000X          HL=@SUM OF'SQUARES;
00266.000  266 1 011C CD0000X          CALL ZERO MEM;
```

```
00321.000  321 1 0180 17              RAL;
00322.000  322 1 0181 3F              CARRY=NOT CARRY;
00323.000  323 1 0182 1F              RAR;
00324.000  324 1 0183 320000X         SUM=A;  ! SIGN CHANGED TO MINUS
00325.000  325 1 0186                 << ADD SUM OF SQUARES - MEAN SQ TIMES N >>
00326.000  326 1 0186 110000X         DE=@SUM OF SQUARES;
00327.000  327 1 0189 424B            B=D;C=E;
00328.000  328 1 018B CD0000X         CALL FP$ADD;     ! NUMERATOR OF VARIANCE
00329.000  329 1 018E                 << FIND N-1 (FLOAT PT) & DIVIDE INTO
00330.000  330 1 018E                   THE NUMERATOR TO FIND THE VARIANCE >>
00331.000  331 1 018E 3A0A006         A=SAMPLE SIZE;
00332.000  332 1 0191 C699            A=A+#99;         ! SUBTRACT 1
00333.000  333 1 0193 27              DAA;
00334.000  334 1 0194 3209006         N$MINUS$1=A;
00335.000  335 1 0197 1109006         DE=@N$MINUS$1;
00336.000  336 1 019A 210000X         HL=@FP$NUM;
00337.000  337 1 019D 0E01            B=1;
00338.000  338 1 019F CD0000X         CALL INT$TO FP;
00339.000  339 1 01A2 EB              DE==HL;          ! @DIVISOR
00340.000  340 1 01A3 210000X         HL=@SUM OF SQUARES;  ! NUMERATOR
00341.000  341 1 01A6 010000X         BC=@SUM;      ! @VARIANCE
00342.000  342 1 01A9 CD0000X         CALL DIVIDE;     ! HL=@VARIANCE
00343.000  343 1 01AC                 << TAKE THE SQRT OF THE VARIANCE TO
00344.000  344 1 01AC                   FIND THE STD DEV                  >>
00345.000  345 1 01AC C1              BC=TOS;
00346.000  346 1 01AD CD0000X         CALL SQRT;
00347.000  347 1 01B0 C9              RETURN;
00348.000  348 0 01B1                 END.
            ***** 0 ERROR(S), 0 WARNING(S), SIZE 0 *****
  BCAL-80 05.2.3   78 11 27  11:14:20
BIN TO BCD         EXTERNAL  #0015
                     12   259
CHECK FOR END      LABEL     #008A
                    138   119
COMPARE            ENTRY     #0000
                      5    33
COMPARE ANOTHER    LABEL     #005C
                    100   142
DIVIDE             EXTERNAL  #000E
                      9   224   311   343
FP$ADD             EXTERNAL  #0010
                     10    43    47    58    62
                    198   279   288   328
FP$NUM             EXTERNAL  #0012
                     10   194   219   274   282
                    283   307   317   336
FP$TO INT          EXTERNAL  #000A
                      8    86    89
INT TO FP          EXTERNAL  #000D
                      9   195   220   275   308
                    333
LARGER             EXTERNAL  #000C
                      8   104   123
LOW LIM INT        GLOBAL    #000E
                     18    88   122
```

```
MEAN                ENTRY      #0097
                      6    150
MULT                EXTERNAL   #000F
                      9    285    316    319
N MINUS 1           GLOBAL     #0009
                     17    334    335
NEW NUMBER          LABEL      #0083
                    181    215
NEXT SAMPLE         LABEL      #0123
                    272    302
NUM REMAINING       GLOBAL     #0009
                     15     17     95    139    141
                    170    212    214    253    291
                    297
SAMPLE SIZE         GLOBAL     #000A
                     16    165    200    203    218
                    258    306    331
SET FLAG            LABEL      #0067
                    108    131
SORT                EXTERNAL   #0011
                     10    346
SQUARE              GLOBAL     #0000
                     14    284    314
STD DEV             ENTRY      #00FA
                      6    232
SUM                 EXTERNAL   #0013
                     11     42     64    175    196
                    223    262    277    311    318
                    320    324    341
SUM OF SQUARES      EXTERNAL   #0014
                     11     57     70     87    265
                    286    326    340
UP LIM INT          GLOBAL     #000E
                     13     85    103
ZERO MEM            EXTERNAL   #000E
                      8     76    176    263    266
  DIAL-80 05.2.3    78-11-27   11:15:47
```