NISTIR 89-4107

A COMPUTER-CONTROLLED TEST SYSTEM FOR OPERATING DIFFERENT WEAR TEST MACHINES

Eric P. Whitenton
A. W. Ruff

U.S. Department of Commerce
National Institute of Standards and Technology
Institute for Materials Science and Engineering
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Raymond G. Kammer, Acting Director

NISTIR 89-4107

A COMPUTER-CONTROLLED TEST SYSTEM FOR OPERATING DIFFERENT WEAR TEST MACHINES

Eric P. Whitenton
A. W. Ruff

U.S. Department of Commerce
National Institute of Standards
and Technology
Institute for Materials Science
and Engineering
Gaithersburg, MD 20899

NOTE: Brand names are mentioned in this document only for completeness and do not constitute an endorsement of any product.

June 1989

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Raymond G. Kammer, Acting Director

TABLE OF CONTENTS

OVERVIEW

Tribology laboratories generally have more than one type of wear test machine. There is a trend toward retro-fitting these machines with dedicated computers to facilitate machine operation, data acquisition, and reporting of the test results. This can greatly enhance the through-put, flexibility, ease of use, and accuracy of the test systems. However, this necessitates the operator learning how to use the software which runs each machine. If every machine has unique software, the user has much to learn. Also, an improvement in one program often means that the programmer has to make a similar improvement in the other programs, tying up the programers time doing essentially redundant work. One solution to this problem is to group functionally similar tests together and have the same software program run all those tests.

This report discusses such a system, where the same computer and software runs three different wear test machines, a Falex crossed-cylinder, a Falex block-on-ring, and an in-house designed controlled atmosphere tribometer. The computer hardware will be described first. Then, the interface to the wear test machines and the aspects that make these machines "functionally similar" will be examined. Finally, the program itself, its use, and the data file structure will be explored. Appendix I lists the program. Appendix II lists a utility program for aiding in plotting and analysis of the results. Appendix III shows a sample printout.

COMPUTER SYSTEM HARDWARE

A schematic diagram of the computer system is shown in figure 1. The computer is effectively an IBM AT clone with a Hewlett Packard Basic Language Processor board (also known as a "Viper" board) installed. The Viper board gives the clone the ability to function as either an IBM or a Hewlett Packard (HP) machine. The board comes with its own processor, memory, and IEEE 488 interface buss to "talk" to the instrumentation. In the HP mode, a dialect of basic, called "Rocky Mountain Basic", is used. This structured language was adopted primarily because it provides very good support for real time instrumentation control. It also provides many other features such as extensive graphics and matrix commands. There is also the capability to "talk" to the IBM DOS system as though it were simply another I/O device. This allows the IBM peripherals to be used by the Viper board. IBM compatible disk files can be generated as well as HP type files. The Viper board can also independently run a test in its "stand alone" mode while the user performs other tasks, such as word processing, using the IBM part of the machine.

The computer we have used has a high density 3 1/2" floppy drive A, a double-sided, double-density 5 1/4" floppy drive B, and a 10 Mbyte hard drive C. The older style drive was chosen for the B drive to provide the capability of producing disk files for users who need such disks. Drive C has, among others, two directories. One directory is called "HPW" and contains the software to run the Viper board. The other is called "DATA" and is used for storing data should the floppy drives fail to function properly.

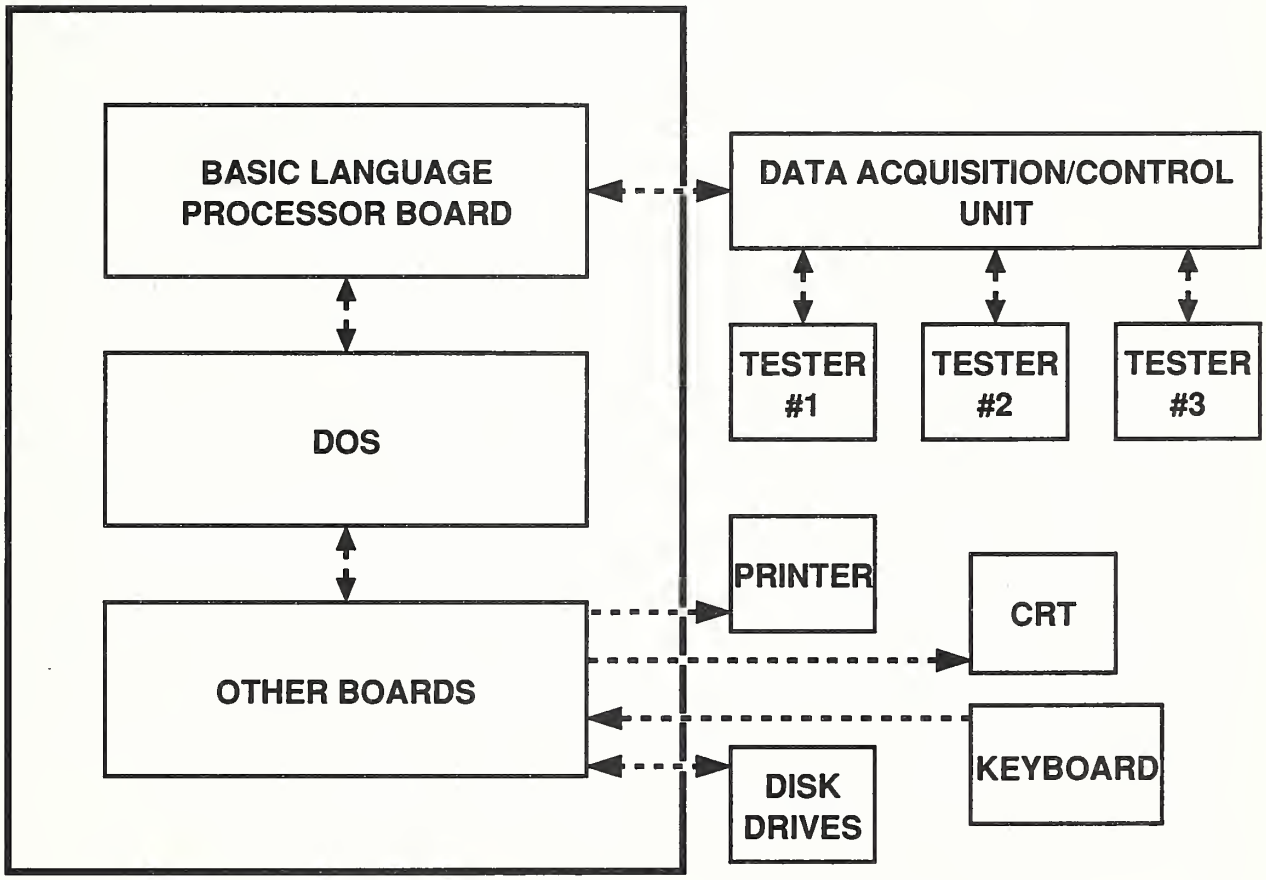


Figure 1. Schematic Diagram of Computer System.

The monitor is monochrome. Color is supported by the Viper board, but was not necessary. If a color system were used, then the program could have "PEN" statements used through-out to take advantage of this.

The printer is an HP Think Jet printer connected to the "LPT1:" port. This printer was chosen because it supports screen dumps from the Viper board. If screen dumps while in the HP mode are not needed, then almost any printer will do. The program uses screen dumps and "escape" codes for this printer, however, and would need to be modified. Connecting the printer to the "LPT1:" port allows both the Viper board and the IBM to use it. An IEEE 488 printer could be used but then only the Viper board would be able to access it.

The HP3421A data acquisition and control unit (DACU) is an IEEE 488 buss device equipped with two boards. The first is a multiplexer board in slot 0. This allows the voltmeter to measure up to 10 voltages. The second is an optically isolated digital I/O board in slot 2. This provides the unit with 8 digital input and 8 digital output lines. Slot 1 is currently left empty but provides "room" for future expansion. The voltmeter is a floating input type with auto-ranging and auto-zeroing capabilities.

WEAR TESTER INTERFACE

Many wear testers are nearly identical from the point of view of computer control. First, a motor is turned on. Then, friction an/or displacement is measured while two or more specimens rub against each other. Finally, after some number of revolutions or seconds, the motor is

turned off. The computer can then analyze the data or run another test. If the interface to the wear test machine is standardized, then the same software can run other machines.

The interface we are using in this system consists of three analog voltage measurement lines, one digital output line, and one digital input line. The first analog line measures the friction force. The second is optional and measures displacement. The third serves two functions. Each tester provides a different non-zero DC voltage on this line. This both identifies the tester and informs the computer that the tester is turned on. The digital output line controls the motor, turning it on and off. The digital input line enables the computer to know when each event occurs. If the computer is to "log" a value every revolution, then the tester must toggle that line (change its state from zero to one or one to zero) every revolution. This could be done with an optical or magnetic sensor on the motor shaft and a divide-by-two counter. If time is to be used, the tester might toggle that line every 10 seconds using a timer circuit.

USING THE PROGRAM

When running the program it first "introduces" itself with a few help screens. Next, the user enters which tester will be used. The DACU itself is checked. The appropriate voltage line is checked to insure that the tester is turned on and returning the proper voltage. The DACU is initialized.

Scaling factors for the tester are then displayed. These factors scale voltage to friction force, voltage to displacement, and events to the

correct units. Note that when the factors are later printed out, the friction force factors will have been divided by the applied load. This makes the output a coefficient of friction instead of a friction force. The coefficient will be displayed, printed, and saved in the disk file.

The disk drive and file name to ultimately store the data in are now requested. The disk is checked for potential errors, such as a duplicate or invalid file name. This requires the Viper board to perform DOS commands. The screen will (unfortunately) go blank while in DOS but return when back in the HP mode. The applied load is now entered.

For each of the measured parameters; events, friction, and/or displacement, the program will now request upper limit values. If test termination is not desired on a given parameter, simply enter an extremely high number. For example, the user is prompted "terminate test after ? distance in meters". If the slider is to have slid 50 meters before stopping, then enter 50. When prompted "terminate test after ? friction in coef.", the user could enter something like 9999. Since the friction coefficient would never reach this value, the test should never terminate for this reason. Suppose the friction coefficient during the test was known to keep increasing from 0.2 to 0.7 and the user wanted to study the changes in the surfaces as a function of how far along in this process the system was. The user could perform, say, five wear tests. One wear test would terminate at friction coefficient 0.3. Another test would terminate at friction coefficient 0.4. The others would terminate at 0.5, 0.6, and 0.7. These five sets of samples could then be examined and compared. Another use for the terminate function is to protect the equipment. If the load cell measuring friction, for example, could be damaged if the force

went over some level, then the appropriate terminate value for the friction coefficient would stop the test if the friction became to high.

For each of the measured parameters, the user is next asked to enter the full scale value for the screen graph which will appear during the test run. The defaults are the values entered for test termination, which may or may not be the best values for the graph. If the test is to terminate after 10 meters, then 10 meters full scale on the X axis makes sense. If the test is to terminate on a friction coefficient of 2.5 in order to protect the load cell, but the friction is only expected to reach 0.2, then something like 0.5 full scale for the friction coefficient is more reasonable.

The user is next asked various questions which are simply text to the computer. They are put in both the disk file and the printout to document the test but the program never actually uses the information. This includes items such as the specimen materials, lubrication, and relative humidity. Something must be entered for each question. If the answer is not known, then enter a "?" or some other symbol to signify that fact.

Next, the user is asked whether or not a printout during the test is desired. If one is desired, then the user is asked "Print out 1 of every how many lines of data?". For example, if a two is entered, then one of every two lines will be printed out. This only effects the printout and has no effect on the graph or the disk file. The purpose of this is to keep the printed record down to a manageable size. The default is calculated to give three (8 1/2" by 11") pages.

A zero level will now be set for the friction and/or displacement signals. First, the system will repeatedly monitor the friction voltage

and display an average of the last 10 readings. When a key is pressed the last average displayed will be used as the friction coefficient zero level. If displacement is to be monitored, then the same is done for displacement. This allows the user to establish an appropriate zero level before the test.

Next, a temporary disk file for storing the data is created. This is discussed next in the data format section. If a printout was requested, the header for that printout is printed. The axes for the screen plot are then displayed. Upon pressing the return key, the test will now begin.

While the test is running the function keys are defined by the program. Function key F8 will terminate the test and save the data on the disk when pressed TWICE IN SUCCESSION. Thus, the user can terminate the test early if required. Function key F1 increases the full scale range for the friction plot. Function key F2 decreases it. If displacement is being monitored, then function key F3 increases and function key F4 decreases its' full scale. These effect the screen plot only and have no effect on the printout or the disk file. Note that the screen plot emulates a strip chart recorder. When the scale is changed, the data already drawn does not change, only the data to be drawn. Since the data are stored on disk and can be replotted later, there is no reason to have the computer spending time redrawing the entire screen plot.

The number displayed near the lower left hand side of the screen is the number of readings averaged together to make one data point. This is discussed more in the disk file format section next.

When the test is done, the motor shuts off. The printout finishes with a screen dump of the screen plot. The data are then put on the disk.

If there is a problem with the disk, then the user can enter another drive and file name to try. The program then ends.

DISK FILE FORMAT

If only friction is to be monitored, then each data point represents the average of the number of readings the system acquires in one event, i.e., one per event. If both friction and displacement are monitored, then friction is monitored for one event and displacement is monitored in the next event. This results in one data value for each for every two events.

There are two main sections to the disk file. The first is referred to as a "header" and contains information such as the date of the test, the test materials, etc. The second section is the column(s) of data. The first line of the header is the number of lines of header information. The first line of the data section is the number of columns of data. The second line of the data section is the column label(s).

The disk file is an ASCII type file which can be loaded into LOTUS 1-2-3 by using the "/ File Import Numbers" command. The data will be placed wherever the curser on the spreadsheet is located. More than one file can be imported onto the same spreadsheet by importing the first file, moving the cursor to a new area, and then importing the next file. The sliding distance is not included in the file to save disk space since that can easily be calculated by 1-2-3 using the information in the header. The end of the file name, the "extension", should be ".PRN" for 1-2-3 to read it. With a properly set up spreadsheet, many operations can be performed on the

data such as high and low pass filtering, moving averages and standard deviations, and plotting multiple tests on one graph.

If a test was long with many points, then the user may acquire a very large amount of data. This may be required if high resolution is needed to accurately capture the shapes of transients. If many tests are placed on the same spreadsheet, however, the computer may simply run out of memory. The program in appendix II is designed to help get around this problem. Written in Quick Basic, it can be run on any IBM. It is meant to be a "preprocessor" program, that is, to get the data ready for further processing. The preprocessor asks the user for the drive and file name of the data, the number of rows to average, and the drive and file name for the new "compacted" file. The preprocessor then reads in the original data file and creates a new data file. If the user entered in 10 rows to average, then each row of data in the new file will be the average of 10 rows in the old file.

FUTURE EXPANSION

Many of the variables used in this program are in array form. This allows for easy expansion to monitor more parameters should that be required. An earlier version of this program had a function key that would allow the user to pause and re-start the test when ever desired. Some users were afraid of accidental misuse of this key and so it was removed. A future version might not only reinstate this capability, but automatically document its' use on the disk file and the printout.

During the test the data is stored in a "RAM disk drive". This acts like a normal HP disk file only it is stored in memory instead of a floppy disk. In the event that the program itself "crashed" before saving the data, the user could use the COPY command to do so. With only a few changes in the software, a floppy or hard drive could be used for storage during the actual test instead. While this is slower than using memory, if there should be a power failure during a test the data would not be lost. This could be very useful for tests which are longer than a few hours. A future version might give the user the option to use either the RAM, floppy, or hard drive.

APPENDIX I - WEAR TEST PROGRAM

```
1000 ! ***** WEAR TEST, VERSION 1, ERIC P WHITENTON, NIST, FEB 1989 *****
1005 !
1010 ! ***** INITIAL SETUP *****
1015 !
1020   !ALL ARRAYS START AT ELEMENT 0, THOUGH THE 0'TH ELEMENT MAY NOT ALWAYS BE USED
1025     OPTION BASE 0
1030   !PRINTER IS THE SCREEN
1035     PRINTER IS 1
1040   !CLEAR THE SCREEN
1045     DISP "SETTING UP SCREEN, PLEASE WAIT"
1050       GCLEAR
1055       CLEAR SCREEN
1060     DISP
1065   !LOGICAL CONSTANTS
1070     COM /Logical/ INTEGER True,False
1075     False=0
1080     True=NOT False
1085   !ADDRESS OF HPIB
1090     INTEGER Buss
1095     Buss=7
1100   !ADDRESS OF DATA ACK AND CONTROL DEVICE ON HPIB
1105     INTEGER Dev
1110     Dev=709
1115   !PRINT OUT FLAG, A LOGICAL VARIABLE, USE PRINTOUT DEVICE IF TRUE
1120     INTEGER Printout_flag
1125   !DEVICE TO PRINT OUT TO
1130     COM /Print_dev/ INTEGER Printout_dev
1135     Printout_dev=26
1140   !HOW OFTEN TO PRINT OUT THE DATA
1145     INTEGER Every_other
1150   !EXPLAIN HOW PROGRAM WORKS
1155     Explain_routine
1160   !USED TO LABEL PLOT
1165     ALLOCATE Label$[50]
1170   !DISK OUTPUT STRING
1175     ALLOCATE Dsk_line$[255]
1180   !PRINTER OUTPUT STRING
1185     ALLOCATE Prt_line$[255]
1190   !PROMPT STRING
1195     ALLOCATE Prompt$[100]
1200   !REPLY TO INPUT REQUEST
1205     ALLOCATE Reply$[100]
1210     REAL Reply
1215   !LOGICAL VARIABLES
1220     INTEGER Done,Exists
1225   !COUNTER
1230     INTEGER Counter
1235   !NUMBER OF RECORDS IN HP FILES
1240     INTEGER Hp_file_size
1245     Hp_file_size=1500
```

```
1250  !RAM DRIVE TO TEMPORALLY SAVE DATA ON DURING TEST
1255    ALLOCATE Hp_ram_drive$[20]
1260    Hp_ram_drive$=":MEMORY,0"
1265  !DRIVE TO TEMPORALLY SAVE DATA ON AT END OF TEST
1270    ALLOCATE Temp_hp_drive$[20],Temp_dos_drive$[20]
1275    Temp_hp_drive$=":,1500,0"
1280    Temp_dos_drive$="C:\HPW\"
1285  !NAME OF FILE TO TEMPORALLY SAVE DATA ON DURING AND AT END OF TEST
1290    ALLOCATE Temp_hp_file$[20],Temp_dos_file$[20]
1295    Temp_hp_file$="TMP_HP"
1300    Temp_dos_file$="TMP_DOS"
1305  !DRIVE TO ULTIMATELY SAVE TRACES ON
1310    ALLOCATE Dos_drive$[20]
1315  !FILENAME OF FILE TO ULTIMATELY SAVE TRACES IN
1320    ALLOCATE Dos_file$[12]
1325  !LOGICAL VARIABLE, TELLS WHEN DONE TEST
1330    INTEGER Done_test
1335  !LOGICAL VARIABLE, TELLS IF ERROR HAS OCCURRED
1340    INTEGER Err_flag
1345  !CHANNEL COUNTER
1350    INTEGER Channel
1355  !REASON FOR ENDING TEST OR TERMINATING PROGRAM
1360    ALLOCATE Reason$[50]
1365  !NUMBER OF CHANNELS TO RECORD
1370    INTEGER Nchannels
1375  !TYPE OF TEST
1380    ALLOCATE Type_of_test$[50]
1385    CLEAR SCREEN
1390    PRINT "SELECT TEST TO RUN"
1395    PRINT " (1) for crossed cylinder test"
1400    PRINT " (2) for block on ring test"
1405    PRINT " (3) for Controlled Atmosphere Trobometer {CAT}"
1410    Prompt$="1,2 OR 3"
1415    Enter_value(Prompt$,Reply,1,3,True)
1420    SELECT Reply
1425      CASE 1
1430        Type_of_test$="CROSSED CYLINDERS"
1435        Nchannels=2
1440        Done=True
1445      CASE 2
1450        Type_of_test$="BLOCK ON RING"
1455        Nchannels=1
1460        Done=True
1465      CASE 3
1470        Err_beep
1475        End_program("CAT NOT YET FULLY SUPPORTED IN SOFTWARE")
1480    END SELECT
1485  !CYCLE COUNTERS
1490    INTEGER Cycle_count,Old_cycle_count
1495  !TELLS Cycle_counter WHEN TO INCRAMENT
```

```
1500    INTEGER Cycle_bit,Old_cycle_bit
1505  !TOTAL READINGS DURING THE TEST, USED TO COMPUTE AVERAGE READINGS/CYCLE
1510    REAL Sum_readings
1515  !FOR READING VOLTAGES
1520    INTEGER Nreadings
1525    REAL Total,Voltage
1530  !EACH TESTER HAS CORRESPONDING CHANNELS IN DATA LOGGER DEVICE
1535    ALLOCATE Channel_number$(Nchannels)[2]
1540    ALLOCATE Channel_name$(Nchannels)[10]
1545    ALLOCATE Channel_units$(Nchannels)[10]
1550    ALLOCATE Motor$[2]
1555    ALLOCATE Test_volts$[2]
1560    REAL Min_test_volts,Max_test_volts
1565    ! Channel_number$(0) NOT USED, USE Motor$ INSTEAD
1570    Channel_name$(0)="Distance"
1575    Channel_units$(0)="meters"
1580    SELECT Type_of_test$
1585      CASE "CROSSED CYLINDERS"
1590        Channel_number$(1)="2" ! FRICTION
1595        Channel_name$(1)="Friction"
1600        Channel_units$(1)="coef"
1605        Channel_number$(2)="3" ! DISPLACEMENT
1610        Channel_name$(2)="Disp"
1615        Channel_units$(2)="um"
1620        Motor$="24"            ! WHEN TALKED TO, MOTOR ON/OFF BIT
1625                               ! WHEN LISTENED TO, MOTOR REVOLUTION BIT
1630        Test_volts$="4"        ! CHANNEL WITH TEST VOLTAGE
1635        Min_test_volts=11      ! MIN VOLTAGE TO ACCEPT AS VALID TEST V
1640        Max_test_volts=13      ! MAX VOLTAGE TO ACCEPT AS VALID TEST V
1645      CASE "BLOCK ON RING"
1650        Channel_number$(1)="5" ! FRICTION
1655        Channel_name$(1)="Friction"
1660        Channel_units$(1)="coef"
1665        Motor$="25"            ! WHEN TALKED TO, MOTOR ON/OFF BIT
1670                               ! WHEN LISTENED TO, MOTOR REVOLUTION BIT
1675        Test_volts$="7"        ! CHANNEL WITH TEST VOLTAGE
1680        Min_test_volts=-11     ! MIN VOLTAGE TO ACCEPT AS VALID TEST V
1685        Max_test_volts=-9      ! MAX VOLTAGE TO ACCEPT AS VALID TEST V
1690      CASE ELSE
1695        Err_beep
1700        DISP "TESTER TYPE NOT FULLY SUPPORTED"
1705        STOP
1710    END SELECT
1715  !CALIBRATION CONSTANTS Y=A0+A1*X
1720    ALLOCATE REAL A0(Nchannels),A1(Nchannels)
1725    ALLOCATE Cal_date$[20]
1730    SELECT Type_of_test$ ! SET DEFAULT VALUES
1735      CASE "CROSSED CYLINDERS"
1740        Cal_date$="DEC 2 1988"
1745        A0(0)=0
```

```
1750          A1(0)=.0399
1755          !A0(1) SET AT BEGINNING OF TEST
1760          A1(1)=-1972
1765          !A0(2) SET AT BEGINNING OF TEST
1770          A1(2)=630
1775        CASE "BLOCK ON RING"
1780          Cal_date$="MARCH 3, 1989"
1785          A0(0)=0
1790          A1(0)=.10972
1795          !A0(1) SET AT BEGINNING OF TEST
1800          A1(1)=-93.4459
1805        CASE ELSE
1810          Err_beep
1815          DISP "TESTER TYPE NOT FULLY SUPPORTED"
1820          STOP
1825      END SELECT
1830    !VALUE OF DATA TO CAUSE TERMINATION OF TEST
1835      ALLOCATE REAL Terminate(Nchannels)
1840    !FULL SCALE VALUE FOR PLOTTING
1845      ALLOCATE REAL Plot_full_scale(Nchannels)
1850    !OFFSET FOR PLOTTING (0 TO 1)
1855      ALLOCATE REAL Plot_offset(Nchannels)
1860      Plot_offset(1)=0
1865    !TYPE OF LINE USED TO DRAW LABEL
1870      ALLOCATE Label_type(Nchannels)
1875      Label_type(1)=1
1880    !TYPE OF LINE USED TO DRAW CURVE
1885      ALLOCATE Curve_type(Nchannels)
1890      Curve_type(1)=1
1895    !PLOTTING FOR SECOND CHANNEL IF EXISTS
1900      IF Nchannels>1 THEN
1905        Plot_offset(2)=.5
1910        Label_type(2)=4
1915        Curve_type(2)=2
1920      END IF
1925    !SAVES LAST PLOTTED POINTS (Last_plotted(0)=-1 MEANS "PEN UP")
1930      ALLOCATE REAL Last_plotted(Nchannels)
1935      Last_plotted(0)=-1
1940    !USED IN PLOTTING
1945      REAL Current_x
1950    !USED FOR THE DOUBLE KEY PRESS ACTION OF SOME FUNCTION KEYS
1955      INTEGER Last_key
1960      Last_key=0
1965    !SAVES THE STARTING TIME
1970      REAL Start_time
1975    !COMMENTS TO BE SAVED ON DISK
1980      INTEGER Ncomments
1985      Ncomments=5
1990      ALLOCATE Comments$(Ncomments)[80]
1995      ALLOCATE Comment_prompts$(Ncomments)[80]
```

```
2000     Comment_prompts$(1)="SPECIMEN MATERIALS"
2005     Comment_prompts$(2)="LUBRICATION"
2010     Comment_prompts$(3)="RELATIVE HUMIDITY (%)"
2015     Comment_prompts$(4)="SPEED (RPM)"
2020     Comment_prompts$(5)="OTHER COMMENTS"
2025 !
2030 ! ***** SETUP DEVICE & MAKE SURE CORRECT CONNECTIONS TO TESTER *****
2035 !
2040   DISP "SETTING UP DEVICE & CHECKING CONNECTIONS, PLEASE WAIT"
2045     Err_flag=True
2050     ON ERROR GOTO 2140
2055     ON TIMEOUT Buss,4 GOTO 2140
2060       !RESET DEVICE
2065         OUTPUT Dev;"RS"
2070       !TEST FOR PROPER TESTER
2075         OUTPUT Dev;"DCV"&Test_volts$
2080         ENTER Dev;Voltage
2085         OUTPUT Dev;"OPN"&Test_volts$
2090         IF (Voltage<Min_test_volts) OR (Voltage>Max_test_volts) THEN
2095           Err_beep
2100           End_program("'"&Type_of_test$&"' MACHINE NOT RETURNING PROPER VOLTAGE")
2105         END IF
2110       !MISC DEVICE CODES
2115         OUTPUT Dev;"Z1"  ! AUTO ZERO
2120         OUTPUT Dev;"RA1" ! AUTO RANGE
2125         OUTPUT Dev;"F1"  ! FUNCTION 1 (DC VOLTS)
2130         OUTPUT Dev;"N3"  ! 3 1/2 DIGITS
2135       Err_flag=False
2140     OFF TIMEOUT
2145     OFF ERROR
2150   DISP
2155   IF Err_flag THEN
2160     Err_beep
2165     End_program("DEVICE '"&VAL$(Dev)&"' NOT RESPONDING CORRECTLY")
2170   END IF
2175 !
2180 ! ***** SHOW SCALING FACTORS *****
2185 !
2190   CLEAR SCREEN
2195   FOR Channel=0 TO Nchannels
2200     PRINT "SCALING FACTOR FOR ";Channel_name$(Channel);" IS ";VAL$(A1(Channel))
2205   NEXT Channel
2210   PRINT
2215   PRINT "LAST CALIBRATED ";Cal_date$
2220   PRINT
2225   PRINT
2230   PRINT "ENTER"
2235   PRINT " (1) THESE ARE OK, CONTINUE WITH TEST"
2240   PRINT " (2) THESE ARE NOT OK, END PROGRAM"
2245   Enter_value("1 or 2",Reply,1,2,True,1)
```

```
2250   IF Reply=2 THEN
2255     End_program("CALIBRATION CONSTANTS NEED TO BE UPDATED")
2260   END IF
2265 !
2270 ! ***** ENTER INITIAL PARAMETERS *****
2275 !
2280   CLEAR SCREEN
2285   Enter_dos_drive(Dos_drive$,False,"B:",True)
2290   Enter_dos_file(Dos_drive$,Dos_file$)
2295   CLEAR SCREEN
2300   Enter_value("applied load in newtons",Applied_load,MINREAL,MAXREAL,False)
2305   A1(1)=A1(1)/Applied_load ! MAKE COEF INSTEAD OF FORCE
2310   CLEAR SCREEN
2315   FOR Channel=0 TO Nchannels
2320     Prompt$="terminate test after ? "&Channel_name$(Channel)&" in "&Channel_units$(Channel)
2325     Enter_value(Prompt$,Terminate(Channel),0,MAXREAL,False)
2330   NEXT Channel
2335   CLEAR SCREEN
2340   FOR Channel=0 TO Nchannels
2345     Enter_value("plot full scale for "&Channel_name$(Channel)&" in "&Channel_units$(Channel),Plot_full_scale(Channe
         l),0,MAXREAL,False,Terminate(Channel))
2350   NEXT Channel
2355   CLEAR SCREEN
2360   FOR Counter=1 TO Ncomments
2365     Enter_string(Comment_prompts$(Counter),Comments$(Counter),False)
2370   NEXT Counter
2375   CLEAR SCREEN
2380   PRINT "DO YOU WANT A PRINTOUT ON THE PRINTER?"
2385   PRINT " (1) YES"
2390   PRINT " (2) NO"
2395   Enter_value("1 OR 2",Reply,1,2,True,1)
2400   SELECT Reply
2405     CASE 1
2410       Printout_flag=True
2415       PRINT
2420       PRINT "PRINT OUT 1 OF EVERY HOW MANY LINES OF DATA?"
2425       Enter_value("A NUMBER",Reply,2,5000,True,MAX(2,5*INT((2.5+Terminate(0)/A1(0)/Nchannels/120)/5),0))
2430       Every_other=Reply*Nchannels
2435     CASE 2
2440       Printout_flag=False
2445   END SELECT
2450 !
2455 ! ***** MEASURE ZEROS *****
2460 !
2465   CLEAR SCREEN
2470   PRINT "MEASURING ZERO FOR PARAMETER SHOWN BELOW"
2475   PRINT
2480   PRINT "WHEN VALUE IS ACCEPTABLE, PRESS ANY KEY"
2485   FOR Channel=1 TO Nchannels
2490     LOOP
2495       OUTPUT Dev;"CLS"&Channel_number$(Channel)
```

```
2500        Total=0
2505        FOR Counter=1 TO 10
2510          OUTPUT Dev;"T2"
2515          ENTER Dev;Voltage
2520          Total=Total+Voltage
2525        NEXT Counter
2530        Voltage=Total/10
2535        DISP "VOLTAGE FOR "&Channel_name$(Channel)&" = "&VAL$(Voltage)
2540        A0(Channel)=-A1(Channel)*Voltage
2545        ON KBD ALL,1 GOTO Done_zeroing
2550      END LOOP
2555 Done_zeroing:!
2560      OFF KBD
2565      Key_chirp
2570    NEXT Channel
2575    CLEAR SCREEN
2580 !
2585 ! ***** OPEN DISK FILE, SETUP AUTO TERMINATE IF RAM DISK BECOMES FULL, & SAVE PARAMETERS *****
2590 !
2595    DISP "OPENING TEMP RAM DISK FILE, PLEASE WAIT"
2600      INITIALIZE Hp_ram_drive$,Hp_file_size+100
2605      CREATE ASCII Temp_hp_file$&Hp_ram_drive$,Hp_file_size
2610      ASSIGN @File TO Temp_hp_file$&Hp_ram_drive$
2615      ON END @File GOSUB Ram_drive_full
2620      OUTPUT @File;Ncomments+4
2625      OUTPUT @File;CHR$(34)&"DATE"&CHR$(34)&" "&CHR$(34)&DATE$(TIMEDATE)&CHR$(34)
2630      OUTPUT @File;CHR$(34)&"DISTANCE PER DATA SET ("&Channel_units$(0)&")"&CHR$(34)&" "&VAL$(A1(0)*Nchannels)
2635      OUTPUT @File;CHR$(34)&"TYPE OF TEST"&CHR$(34)&" "&CHR$(34)&Type_of_test$&CHR$(34)
2640      OUTPUT @File;CHR$(34)&"APPLIED LOAD (N)"&CHR$(34)&" "&VAL$(Applied_load)
2645      FOR Counter=1 TO Ncomments
2650        OUTPUT @File;CHR$(34)&Comment_prompts$(Counter)&CHR$(34)&" "&CHR$(34)&Comments$(Counter)&CHR$(34)
2655      NEXT Counter
2660      OUTPUT @File;Nchannels
2665      Dsk_line$=""
2670      FOR Channel=1 TO Nchannels
2675        Dsk_line$=Dsk_line$&CHR$(34)&Channel_name$(Channel)&"("&Channel_units$(Channel)&")"&CHR$(34)
2680        IF Channel<>Nchannels THEN Dsk_line$=Dsk_line$&" "
2685      NEXT Channel
2690      OUTPUT @File;Dsk_line$
2695    DISP
2700 !
2705 ! ***** HEADER OF PRINTOUT *****
2710 !
2715    IF Printout_flag THEN
2720      Done=False
2725      WHILE NOT Done
2730        DISP "PRINTING HEADER"
2735          Err_flag=True
2740          ON ERROR GOTO 2915
2745          ON TIMEOUT Printout_dev,2 GOTO 2915
```

```
2750          OUTPUT Printout_dev;CHR$(27);"&l1L"; ! SKIP PERFORATION
2755          OUTPUT Printout_dev;CHR$(27);"&s0C"; ! WRAP AROUND ON
2760          OUTPUT Printout_dev;"DATE: ";DATE$(TIMEDATE)
2765          OUTPUT Printout_dev;"CYCLES PER DATA SET: ";VAL$(Nchannels)
2770          OUTPUT Printout_dev;"1 OF EVERY ";VAL$(Every_other/Nchannels);" DATA SETS PRINTED OUT"
2775          OUTPUT Printout_dev;"TYPE OF TEST: ";Type_of_test$
2780          OUTPUT Printout_dev;"APPLIED LOAD (N): ";VAL$(Applied_load)
2785          OUTPUT Printout_dev;"ULTIMATE DISK FILE NAME: ";Dos_drive$;Dos_file$
2790          FOR Counter=1 TO Ncomments
2795            OUTPUT Printout_dev;Comment_prompts$(Counter);": ";Comments$(Counter)
2800          NEXT Counter
2805          OUTPUT Printout_dev;""
2810          OUTPUT Printout_dev;FNColumn$("CALIBRATION FACTORS:");
2815          OUTPUT Printout_dev;FNColumn$("INTERCEPT");
2820          OUTPUT Printout_dev;FNColumn$("SLOPE");
2825          OUTPUT Printout_dev;FNColumn$("UNITS")
2830          Counter=0
2835          FOR Channel=0 TO Nchannels
2840            OUTPUT Printout_dev;FNColumn$(Channel_name$(Channel));
2845            OUTPUT Printout_dev;FNColumn$(VAL$(A0(Channel)));
2850            OUTPUT Printout_dev;FNColumn$(VAL$(A1(Channel)));
2855            OUTPUT Printout_dev;FNColumn$(Channel_units$(Channel))
2860          NEXT Channel
2865          OUTPUT Printout_dev;"LAST CALIBRATED ";Cal_date$
2870          OUTPUT Printout_dev;""
2875          OUTPUT Printout_dev;""
2880          OUTPUT Printout_dev;FNColumn$("Time(h:m:s)");
2885          FOR Channel=0 TO Nchannels
2890            OUTPUT Printout_dev;FNColumn$(Channel_name$(Channel)&"("&Channel_units$(Channel)&")");
2895          NEXT Channel
2900          OUTPUT Printout_dev;""
2905          OUTPUT Printout_dev;""
2910          Err_flag=False
2915        OFF TIMEOUT
2920        OFF ERROR
2925        IF Err_flag THEN
2930            Err_beep
2935            PRINT
2940            PRINT "PRINTER NOT RESPONDING PROPERLY"
2945            PRINT
2950            PRINT "TRY AGAIN?"
2955            PRINT " (1) YES"
2960            PRINT " (2) NO"
2965            Enter_value("1 OR 2",Reply,1,2,True,2)
2970            SELECT Reply
2975              CASE 1
2980              CASE 2
2985                Printout_flag=False
2990                Done=True
2995            END SELECT
```

```
3000           ELSE
3005             Done=True
3010         END IF
3015       END WHILE
3020     DISP
3025   END IF
3030 !
3035 ! ***** SET UP PLOT *****
3040 !
3045   DISP "SETTING UP PLOT, PLEASE WAIT"
3050     CLEAR SCREEN
3055     GRAPHICS ON
3060     LINE TYPE 1
3065     VIEWPORT 10,125,28,90
3070     WINDOW 0,1,0,1
3075     GRID .05,.05,0,0,5,5
3080     FRAME
3085     CLIP OFF
3090     CSIZE 3.5,.5
3095     LORG 6
3100     FOR Counter=0 TO 10
3105       MOVE Counter/10,0
3110       LABEL VAL$(Counter*10)&"%"
3115     NEXT Counter
3120     LORG 8
3125     FOR Counter=0 TO 10
3130       MOVE 0,Counter/10
3135       LABEL VAL$(Counter*10)&"%"
3140     NEXT Counter
3145     MOVE 0,-.1
3150     AREA PEN 0 ! DEFAULT FOR REST OF TEST
3155     RECTANGLE 1,.04,FILL
3160     LORG 6
3165     MOVE .5,-.05
3170     LABEL Channel_name$(0)&" {"&VAL$(Plot_full_scale(0))&" "&Channel_units$(0)&"}"
3175     LORG 4 ! DEFAULT FOR REST OF TEST
3180     GOSUB Scaling_labels
3185   DISP
3190 !
3195 ! ***** START TEST *****
3200 !
3205   INPUT "PRESS RETURN TO START TEST",Reply$
3210   FOR Counter=1 TO 4
3215     BEEP 1000*Counter,.1
3220   NEXT Counter
3225 !
3230 ! ***** SET UP FUNCTION KEYS *****
3235 !
3240   USER 1 KEYS
3245   ON KEY 1 LABEL Channel_name$(1)&"   ^",1 GOSUB Key1
```

```
3250   ON KEY 2 LABEL Channel_name$(1)&"   v",1 GOSUB Key2
3255   IF Nchannels>1 THEN
3260       ON KEY 3 LABEL "  "&Channel_name$(2)&"     ^",1 GOSUB Key3
3265       ON KEY 4 LABEL "  "&Channel_name$(2)&"     v",1 GOSUB Key4
3270     ELSE
3275       ON KEY 3,1 CALL Err_beep
3280       ON KEY 4,1 CALL Err_beep
3285   END IF
3290   ON KEY 5,1 CALL Err_beep
3295   ON KEY 6,1 CALL Err_beep
3300   ON KEY 7,1 CALL Err_beep
3305   ON KEY 8 LABEL "ABNORMALEND TEST",1 GOSUB Key8
3310 !
3315 ! ***** START 'TIMER' *****
3320 !
3325   Start_time=TIMEDATE
3330 !
3335 ! ***** START CYCLE COUNTER & MOTOR *****
3340 !
3345   DISABLE
3350     OUTPUT Dev;"BIT"&Motor$
3355     ENTER Dev;Old_cycle_bit
3360     Sum_readings=0
3365     Cycle_count=0
3370     Old_cycle_count=0
3375     OUTPUT Dev;"CLS"&Motor$
3380   ENABLE
3385 !
3390 ! ***** RUN TEST *****
3395 !
3400   DISP "TEST RUNNING"
3405   Data(0)=A0(0)
3410   Done_test=False
3415   WHILE NOT Done_test
3420     Dsk_line$=""
3425     Prt_line$=FNColumn$(TIME$(TIMEDATE-Start_time))&FNColumn$(VAL$(PROUND(Data(0),-2)))
3430     FOR Channel=1 TO Nchannels
3435       OUTPUT Dev;"CLS"&Channel_number$(Channel)
3440       Total=0
3445       Nreadings=1
3450       OUTPUT Dev;"T2"
3455       REPEAT
3460         Nreadings=Nreadings+1
3465         ENTER Dev;Voltage
3470         OUTPUT Dev;"BIT"&Motor$
3475         Total=Total+Voltage
3480         ENTER Dev;Cycle_bit
3485         OUTPUT Dev;"T2"
3490         IF Cycle_bit<>Old_cycle_bit THEN
3495           Cycle_count=Cycle_count+1
```

```
3500          Old_cycle_bit=Cycle_bit
3505        END IF
3510      UNTIL (Cycle_count>Old_cycle_count) OR Done_test
3515      ENTER Dev;Voltage
3520      Total=Total+Voltage
3525      DISP Nreadings
3530      Sum_readings=Sum_readings+Nreadings
3535      Old_cycle_count=Cycle_count
3540      Data(Channel)=A0(Channel)+A1(Channel)*Total/Nreadings
3545      IF Data(Channel)>=Terminate(Channel) THEN
3550        IF NOT Done_test THEN
3555          Reason$="TERMINATE VALUE FOR "&Channel_name$(Channel)&" EXCEEDED"
3560          GOSUB Terminate_test
3565        END IF
3570      END IF
3575      Dsk_line$=Dsk_line$&VAL$(DROUND(Data(Channel),6))
3580      IF Channel<>Nchannels THEN Dsk_line$=Dsk_line$&" "
3585      Prt_line$=Prt_line$&FNColumn$(VAL$(PROUND(Data(Channel),-3)))
3590    NEXT Channel
3595    OUTPUT @File;Dsk_line$
3600    IF Printout_flag THEN
3605      IF Cycle_count MOD Every_other=Nchannels THEN
3610        Printout_flag=False
3615        ON ERROR GOTO 3635
3620        ON TIMEOUT Printout_dev,2 GOTO 3635
3625          OUTPUT Printout_dev;Prt_line$
3630          Printout_flag=True
3635        OFF TIMEOUT
3640        OFF ERROR
3645      END IF
3650    END IF
3655    DISABLE
3660      IF Last_plotted(0)=-1 THEN
3665          FOR Channel=0 TO Nchannels
3670            Last_plotted(Channel)=Plot_offset(Channel)+Data(Channel)/Plot_full_scale(Channel)
3675          NEXT Channel
3680        ELSE
3685          Current_x=Data(0)/Plot_full_scale(0)
3690          FOR Channel=1 TO Nchannels
3695            PENUP
3700            LINE TYPE Curve_type(Channel)
3705            PLOT Last_plotted(0),Last_plotted(Channel)
3710            Last_plotted(Channel)=Plot_offset(Channel)+Data(Channel)/Plot_full_scale(Channel)
3715            PLOT Current_x,Last_plotted(Channel)
3720          NEXT Channel
3725          Last_plotted(0)=Current_x
3730      END IF
3735    ENABLE
3740    Data(0)=A0(0)+A1(0)*Cycle_count
3745    IF Data(0)>=Terminate(0) THEN
```

```
3750        IF NOT Done_test THEN
3755          Reason$="NORMAL TEST TERMINATION"
3760          GOSUB Terminate_test
3765        END IF
3770      END IF
3775    END WHILE
3780  !
3785  ! ***** CLEAR FUNCTION KEYS *****
3790  !
3795    OFF KEY
3800  !
3805  ! ***** TURN MOTOR OFF *****
3810  !
3815    OUTPUT Dev;"OPN"&Motor$
3820  !
3825  ! ***** FINISH PRINTOUT *****
3830  !
3835    IF Printout_flag THEN
3840      DISP "FINISHING PRINTOUT"
3845        ON ERROR GOTO 3900
3850        ON TIMEOUT Printout_dev,2 GOTO 3900
3855          IF NOT (Cycle_count MOD Every_other=Nchannels) THEN
3860            OUTPUT Printout_dev;Prt_line$
3865          END IF
3870          OUTPUT Printout_dev;""
3875          OUTPUT Printout_dev;"TEST TERMINATED (";Reason$;")"
3880          OUTPUT Printout_dev;VAL$(Cycle_count);" REVOLUTIONS"
3885          OUTPUT Printout_dev;VAL$(PROUND(Sum_readings/Cycle_count,-1));" READINGS PER CYCLE ON AVERAGE"
3890          DUMP GRAPHICS #Printout_dev
3895          OUTPUT Printout_dev;CHR$(12); ! form feed
3900        OFF TIMEOUT
3905        OFF ERROR
3910      DISP
3915    END IF
3920  !
3925  ! ***** FINISH WITH DISKING DATA *****
3930  !
3935    DISP "MAKING FINAL DISK FILE"
3940    GRAPHICS OFF
3945      !CLOSE RAM DISK DATA FILE
3950      ASSIGN @File TO *
3955      !MAKE SURE OLD INTERMEDIATE HARD DISK FILES ARE ERASED
3960      ON ERROR GOTO 3970
3965        PURGE Temp_hp_file$&Temp_hp_drive$
3970      OFF ERROR
3975      Dos_shell("DEL "&Temp_dos_drive$&Temp_dos_file$,False,Counter)
3980      !PUT RAM DISK FILE ON TO HARD DISK AND CONVERT TO DOS FILE
3985      COPY Temp_hp_file$&Hp_ram_drive$ TO Temp_hp_file$&Temp_hp_drive$
3990      Dos_shell("C:\HPW\HPWUTIL CHECKOUT "&Temp_dos_drive$&" "&Temp_hp_file$&" "&Temp_dos_drive$&Temp_dos_file$,False
          ,Counter)
3995      !PUT DOS FILE ON TO USER DISK, MAKING SURE THAT COPY WENT OK
```

```
4000     Done=False
4005     WHILE NOT Done
4010       Dos_shell("COPY "&Temp_dos_drive$&Temp_dos_file$&" "&Dos_drive$&Dos_file$,False,Counter)
4015       Dos_shell("DIR "&Dos_drive$&Dos_file$,False,Counter)
4020       IF Counter<>6 THEN
4025           Err_beep
4030           PRINT
4035           PRINT "TEMP FILE DID NOT COPY TO PERMINANT DOS FILE PROPERLY"
4040           Enter_dos_drive(Dos_drive$,True,Dos_drive$,False)
4045           Enter_dos_file(Dos_drive$,Dos_file$)
4050         ELSE
4055           Done=True
4060       END IF
4065     END WHILE
4070   CLEAR SCREEN
4075   GRAPHICS ON
4080 !
4085 ! ***** DONE PROGRAM *****
4090 !
4095   End_program(Reason$)
4100 !
4105 !
4110 !
4115 !
4120 ! ***** SCALING LABELS *****
4125 !
4130 Scaling_labels:!
4135   DISABLE
4140     MOVE 0,1.01
4145     RECTANGLE 1,.05,FILL
4150   ENABLE
4155   FOR Counter=1 TO Nchannels
4160     Label$=Channel_name$(Counter)&" {"&VAL$(Plot_full_scale(Counter))&" "&Channel_units$(Counter)&"}"
4165     DISABLE
4170       LINE TYPE Label_type(Counter)
4175       MOVE Counter/(Nchannels+1),1.01
4180       LABEL Label$
4185     ENABLE
4190   NEXT Counter
4195 RETURN
4200 !
4205 ! ***** KEY 1 *****
4210 !
4215 Key1:!
4220   Key_chirp
4225   Plot_full_scale(1)=2*Plot_full_scale(1)
4230   Last_plotted(0)=-1
4235   GOSUB Scaling_labels
4240 RETURN
4245 !
```

```
4250 ! ***** KEY 2 *****
4255 !
4260 Key2:!
4265   Key_chirp
4270   Plot_full_scale(1)=.5*Plot_full_scale(1)
4275   Last_plotted(0)=-1
4280   GOSUB Scaling_labels
4285 RETURN
4290 !
4295 ! ***** KEY 3 *****
4300 !
4305 Key3:!
4310   Key_chirp
4315   Plot_full_scale(2)=2*Plot_full_scale(2)
4320   Last_plotted(0)=-1
4325   GOSUB Scaling_labels
4330 RETURN
4335 !
4340 ! ***** KEY 4 *****
4345 !
4350 Key4:!
4355   Key_chirp
4360   Plot_full_scale(2)=.5*Plot_full_scale(2)
4365   Last_plotted(0)=-1
4370   GOSUB Scaling_labels
4375 RETURN
4380 !
4385 ! ***** KEY 8 *****
4390 !
4395 Key8:!
4400   Key_chirp
4405   IF Last_key<>8 THEN
4410       Last_key=8
4415       ON DELAY 1,1 GOSUB Key_timer_off
4420     ELSE
4425       Last_key=0
4430       Reason$="END TEST KEY PRESSED"
4435       GOSUB Terminate_test
4440   END IF
4445 RETURN
4450 !
4455 ! ***** KEY TIMER FOR SECOND KEY STROKE *****
4460 !
4465 Key_timer_off:!
4470   Last_key=0
4475 RETURN
4480 !
4485 ! ***** RAM DRIVE FULL *****
4490 !
4495 Ram_drive_full:!
```

```
4500   OFF KEY
4505   IF Done_test=False THEN
4510     Reason$="MEMORY FULL"
4515     GOSUB Terminate_test
4520   END IF
4525 RETURN
4530 !
4535 ! ***** TERMINATE TEST *****
4540 !
4545 Terminate_test:!
4550   OFF KEY
4555   Done_test=True
4560   DISP "TEST OVER ("&Reason$&")"
4565   BEEP 2500,.1
4570   BEEP 3000,.1
4575   BEEP 2500,.1
4580   BEEP 3000,.1
4585 RETURN
4590 !
4595 !
4600 END
4605 !
4610 !
4615 !
4620 SUB Err_beep
4625   BEEP 2000,.07
4630   BEEP 1500,.07
4635   BEEP 2000,.07
4640 SUBEND
4645 !
4650 !
4655 SUB Enter_value(Prompt$,Value_returned,Min_allowed,Max_allowed,INTEGER Must_be_integer,OPTIONAL Default)
4660   !
4665   COM /Logical/ INTEGER True,False
4670   INTEGER Done,Valid_number,Counter
4675   REAL Reply
4680   ALLOCATE Reply$[50]
4685   !
4690   Done=False
4695   WHILE NOT Done
4700     PRINT
4705     PRINT "ENTER ";Prompt$
4710     IF NPAR=6 THEN PRINT "  (DEFAULT is ";VAL$(Default);")"
4715     Reply$=""
4720     ENTER 2;Reply$
4725     Key_chirp
4730     Reply$=TRIM$(Reply$)
4735     IF Reply$="" THEN
4740         IF NPAR=6 THEN
4745             Value_returned=Default
```

```
4750            Done=True
4755          ELSE
4760            Err_beep
4765            PRINT "NO VALUE WAS ENTERED"
4770        END IF
4775      ELSE
4780        Valid_number=False
4785        ON ERROR GOTO 4870
4790          Counter=32
4795          WHILE Counter<=126
4800            IF POS(Reply$,CHR$(Counter))<>0 THEN CAUSE ERROR 8
4805            SELECT Counter
4810              CASE 42
4815                Counter=44
4820              CASE 44
4825                Counter=47
4830              CASE 47
4835                Counter=58
4840              CASE ELSE
4845                Counter=Counter+1
4850            END SELECT
4855          END WHILE
4860          Reply=VAL(Reply$)
4865          Valid_number=True
4870        OFF ERROR
4875        IF NOT Valid_number THEN
4880            Err_beep
4885            PRINT "'";Reply$;"' IS NOT A VALID NUMBER"
4890          ELSE
4895            IF (Must_be_integer=True) AND (Reply<>INT(Reply)) THEN
4900                Err_beep
4905                PRINT "'";VAL$(Reply);"' IS NOT AN INTEGER"
4910              ELSE
4915                IF (Reply>Max_allowed) OR (Reply<Min_allowed) THEN
4920                    Err_beep
4925                    PRINT "'";VAL$(Reply);"' IS NOT WITHIN THE RANGE OF ";VAL$(Min_allowed);" TO ";VAL$(Max_allowed
                        )
4930                  ELSE
4935                    Value_returned=Reply
4940                    Done=True
4945                END IF
4950            END IF
4955        END IF
4960    END IF
4965  END WHILE
4970  PRINT Value_returned
4975 SUBEND
4980 !
4985 !
4990 SUB Enter_string(Prompt$,String_returned$,INTEGER Capatilize,OPTIONAL Default$)
4995  !
```

```
5000   COM /Logical/ INTEGER True,False
5005   INTEGER Done
5010   ALLOCATE Reply$[200]
5015   !
5020   Done=False
5025   WHILE NOT Done
5030     PRINT
5035     PRINT "ENTER ";Prompt$
5040     PRINT "(";VAL$(MAXLEN(String_returned$));" is the maximum number of characters)"
5045     IF NPAR=4 THEN PRINT "  (DEFAULT is '";Default$;"')"
5050     Reply$=""
5055     ENTER 2;Reply$
5060     Key_chirp
5065     Reply$=TRIM$(Reply$)
5070     IF Reply$="" THEN
5075         IF NPAR=4 THEN
5080             String_returned$=Default$
5085             Done=True
5090           ELSE
5095             Err_beep
5100             PRINT "NO VALUE WAS ENTERED"
5105         END IF
5110       ELSE
5115         IF LEN(Reply$)>MAXLEN(String_returned$) THEN
5120             Err_beep
5125             PRINT "LINE TO LONG"
5130           ELSE
5135             String_returned$=Reply$
5140             Done=True
5145         END IF
5150     END IF
5155   END WHILE
5160   IF Capatilize THEN String_returned$=UPC$(String_returned$)
5165   PRINT String_returned$
5170 SUBEND
5175 !
5180 !
5185 SUB Enter_dos_file(Dos_drive$,Dos_file$,OPTIONAL Default$)
5190   !
5195   COM /Logical/ INTEGER True,False
5200   INTEGER Done,Lines_returned
5205   REAL Choise
5210   !
5215   Dos_file$=""
5220   Done=False
5225   WHILE NOT Done
5230     IF NPAR=3 THEN
5235         Enter_string("DOS disk file name to ultimately place data in",Dos_file$,True,Default$)
5240       ELSE
5245         Enter_string("DOS disk file name to ultimately place data in",Dos_file$,True)
```

```
5250     END IF
5255     IF (POS(Dos_file$,"*")<>0) OR (POS(Dos_file$,"?")<>0) THEN
5260         Err_beep
5265         PRINT
5270         PRINT "NO DOS WILD CARD CHARACTERS PLEASE ('*' OR '?')"
5275       ELSE
5280         IF (POS(Dos_file$,".")=0) OR (LEN(Dos_file$)-POS(Dos_file$,".")>3) OR (POS(Dos_file$,".")=LEN(Dos_file$)) T
             HEN
5285             Err_beep
5290             PRINT
5295             PRINT "'";Dos_file$;"' DOES NOT CONTAIN A VALID DOS EXTENSION (TRY '.PRN')"
5300           ELSE
5305             IF POS(Dos_file$,".")<=1 THEN
5310                 Err_beep
5315                 PRINT
5320                 PRINT "'";Dos_file$;"' IS NOT A VALID DOS FILE NAME"
5325               ELSE
5330                 Dos_shell("DIR "&Dos_drive$&Dos_file$,False,Lines_returned)
5335                 IF Lines_returned=0 THEN
5340                     Err_beep
5345                     PRINT
5350                     PRINT "'";Dos_file$;"' IS NOT A VALID DOS FILE NAME"
5355                   ELSE
5360                     IF Lines_returned=6 THEN
5365                         Err_beep
5370                         PRINT
5375                         PRINT "'";Dos_file$;"' ALREADY EXISTS"
5380                         PRINT "(1) delete the old file"
5385                         PRINT "(2) enter in a different file name"
5390                         Enter_value("1 OR 2",Choise,1,2,True,2)
5395                         IF Choise=1 THEN
5400                           Dos_shell("DEL "&Dos_drive$&Dos_file$,False,Lines_returned)
5405                           Dos_shell("DIR "&Dos_drive$&Dos_file$,False,Lines_returned)
5410                           IF Lines_returned=6 THEN
5415                               Err_beep
5420                               PRINT
5425                               PRINT "'"&Dos_file$&"' IS PROTECTED FROM DELETION"
5430                             ELSE
5435                               Done=True
5440                           END IF
5445                         END IF
5450                       ELSE
5455                         Done=True
5460                     END IF
5465                 END IF
5470             END IF
5475         END IF
5480     END IF
5485   END WHILE
5490 SUBEND
5495 !
```

```
5500 !
5505 SUB Enter_dos_drive(Dos_drive$,INTEGER C_drive_allowed,Default$,INTEGER Test_only)
5510   !
5515   COM /Logical/ INTEGER True,False
5520   INTEGER Done,Lines_returned,Ask_for_drive
5525   REAL Choise,Max_choise
5530   ALLOCATE Prompt$[20]
5535   !
5540   Dos_drive$=""
5545   IF C_drive_allowed=True THEN
5550       Max_choise=3
5555       Prompt$="1, 2, OR 3"
5560     ELSE
5565       Max_choise=2
5570       Prompt$="1 OR 2"
5575   END IF
5580   SELECT Default$
5585     CASE "A:"
5590       Choise=1
5595     CASE "B:"
5600       Choise=2
5605     CASE "C:\DATA\"
5610       Choise=3
5615     CASE ELSE
5620       Choise=0
5625   END SELECT
5630   Choise=MIN(Choise,Max_choise)
5635   Ask_for_drive=NOT Test_only
5640   Done=False
5645   WHILE NOT Done
5650     IF Ask_for_drive THEN
5655         PRINT
5660         PRINT "Which DOS drive is to be used for ultimately storing the data ?"
5665         PRINT "(1) 3 1/2 inch drive {drive 'A:'}"
5670         PRINT "(2) 5 1/4 inch drive {drive 'B:'}"
5675         IF C_drive_allowed=True THEN PRINT "(3) hard drive {C:\DATA\} IN EMERGENCIES ONLY!"
5680         IF Choise=0 THEN
5685             Enter_value(Prompt$,Choise,1,Max_choise,True)
5690           ELSE
5695             Enter_value(Prompt$,Choise,1,Max_choise,True,Choise)
5700         END IF
5705         SELECT Choise
5710           CASE 1
5715             Dos_drive$="A:"
5720           CASE 2
5725             Dos_drive$="B:"
5730           CASE 3
5735             Dos_drive$="C:\DATA\"
5740         END SELECT
5745       ELSE
```

```
5750          Dos_drive$=Default$
5755      END IF
5760      CLEAR SCREEN
5765      Dos_shell("DIR "&Dos_drive$,True,Lines_returned)
5770      IF Lines_returned>=4 THEN
5775          Done=True
5780        ELSE
5785          Err_beep
5790          PRINT
5795          PRINT "DRIVE '";Dos_drive$;"' NOT RESPONDING PROPERLY . . ."
5800          PRINT "EITHER THERE IS A PROBLEM WITH THE DISK DRIVE,"
5805          PRINT " THE DISK IS FAULTY, THE DISK IS NOT INITIALIZED,"
5810          PRINT " OR THERE IS NO DISK IN THE DRIVE"
5815          Ask_for_drive=True
5820      END IF
5825    END WHILE
5830  SUBEND
5835  !
5840  !
5845  SUB End_program(Reason$)
5850    !
5855    LOOP
5860      DISP "PROGRAM ENDED ("&Reason$&")"
5865      STOP
5870    END LOOP
5875  SUBEND
5880  !
5885  !
5890  SUB Dos_shell(Command$,INTEGER Print_flag,Lines_returned)
5895    !
5900    COM /Logical/ INTEGER True,False
5905    ALLOCATE Reply$[80]
5910    !
5915    Lines_returned=0
5920    ON TIMEOUT 19,.08 GOTO 5960
5925      OUTPUT 19;"WAIT_OFF"
5930      OUTPUT 19;Command$&" > HPW_PIPE"
5935      LOOP
5940        ENTER 19;Reply$
5945        IF Print_flag=True THEN PRINT Reply$
5950        Lines_returned=Lines_returned+1
5955      END LOOP
5960    OFF TIMEOUT
5965  SUBEND
5970  !
5975  !
5980  SUB Key_chirp
5985    BEEP 2500,.02
5990  SUBEND
5995  !
```

```
6000 !
6005 DEF FNColumn$(String$)
6010   !
6015   INTEGER Length,Max_length
6020   Max_length=20
6025   !
6030   Length=LEN(String$)
6035   IF Length>=Max_length THEN
6040       RETURN String$[1,Max_length]
6045     ELSE
6050       RETURN RPT$(" ",Max_length-Length)&String$
6055   END IF
6060 FNEND
6065 !
6070 !
6075 SUB Explain_routine
6080     !
6085     COM /Logical/ INTEGER True,False
6090     COM /Print_dev/ INTEGER Printout_dev
6095     INTEGER Done
6100     INTEGER Current_page,Max_page
6105     REAL Reply,Default_action
6110     !
6115     Current_page=1
6120     Max_page=4
6125     Done=False
6130     WHILE NOT Done
6135       CLEAR SCREEN
6140       GOSUB Select_page
6145       PRINT "------------------------------------------------------------------- PAGE";Current_page;"OF";Max_page
6150       PRINT " (1) print this page and display next page  | (2) display next page"
6155       PRINT " (3) display previous page                  | (4) continue with program";
6160       IF Current_page<>Max_page THEN
6165           Default_action=2
6170         ELSE
6175           Default_action=4
6180       END IF
6185       Enter_value("1 to 4",Reply,1,4,True,Default_action)
6190       SELECT Reply
6195         CASE 1
6200           PRINTER IS Printout_dev
6205             GOSUB Select_page
6210           PRINTER IS 1
6215           Current_page=(Current_page MOD Max_page)+1
6220         CASE 2
6225           Current_page=(Current_page MOD Max_page)+1
6230         CASE 3
6235           Current_page=((Current_page+Max_page-2) MOD Max_page)+1
6240         CASE 4
6245           Done=True
```

```
6250        END SELECT
6255      END WHILE
6260      CLEAR SCREEN
6265    SUBEXIT
6270 Select_page:!
6275      SELECT Current_page
6280        CASE 1
6285          Print_centered("ROTATIONAL WEAR TEST")
6290          Print_centered("National Institute Of Standards and Technology")
6295          Print_centered("Version 1.0 1989 by EPW")
6300          PRINT
6305          PRINT
6310          Print_centered("THIS PROGRAM WILL RUN ANY 1 OF 3 MACHINES:")
6315          PRINT
6320          Print_centered("The crossed cylinder machine.")
6325          Print_centered("The block on ring machine.")
6330          Print_centered("The controlled atmosphere tribometer (CAT).")
6335          PRINT
6340          PRINT
6345        CASE 4
6350          Print_centered("THE DISK DATA FILE")
6355          PRINT
6360          PRINT "  The disk data file generated is an ASCII type file which can be loaded into"
6365          PRINT "LOTUS 1-2-3 by using the '/ File Import Numbers' command.  The start of the"
6370          PRINT "data will be where ever the curser on the spreadsheet is located.  More than"
6375          PRINT "1 file can be imported onto the same spreadsheet by importing the first data"
6380          PRINT "set, moving the cursor to a new area, and then importing the next data set."
6385          PRINT "The first few lines of the data is a 'header' giving information about the"
6390          PRINT "test.  After that is the actual data.  The sliding distance is not included in"
6395          PRINT "the data disk file to save disk space since that can easily be calculated by"
6400          PRINT "1-2-3 using information in the header.  The end of the file name (the"
6405          PRINT "extension) should be '.PRN' for 1-2-3 to read it."
6410        CASE 2
6415          Print_centered("BEFORE ATTEMPTING TO RUN THIS PROGRAM THE FOLLOWING THINGS SHOULD BE CHECKED:")
6420          PRINT
6425          Print_centered("All equipment to be used is powered on.")
6430          Print_centered("If the printer is to be used, make sure that there is plenty of paper.")
6435          Print_centered("The data disk has been formatted and there is enough room on it for the data.")
6440          PRINT
6445          PRINT
6450          Print_centered("IF YOU MUST START OVER:")
6455          PRINT
6460          Print_centered("First press 'Shift' and 'Scroll Lock' at the same time to stop the program.")
6465          Print_centered("Then press 'F3' to run the program.")
6470          PRINT
6475        CASE 3
6480          Print_centered("WHEN ANSWERING THE QUESTIONS NOTE THAT:")
6485          PRINT
6490          Print_centered("Some questions have a default value which is displayed while others do not.")
6495          Print_centered("Defaults, if present, can be used by simply pressing 'ENTER'.")
```

```
6500          Print_centered("If no default is present, then a reply must be typed in.")
6505          Print_centered("When you enter a reply, the computer will 'chirp' to acknowledge that fact.")
6510          Print_centered("If your reply is not understood by the computer, an error message will be")
6515          Print_centered("issued along with another chance to enter your reply.")
6520          Print_centered("If you are entering a number, you can not reply with an equation, such as")
6525          Print_centered("'1/2', but must reply with the number, such as '.5'.")
6530          PRINT
6535          PRINT
6540      END SELECT
6545    RETURN
6550  SUBEND
6555  !
6560  SUB Print_centered(OPTIONAL Line$)
6565    SELECT NPAR
6570      CASE 0
6575        PRINT
6580      CASE 1
6585        PRINT RPT$(" ",40-(LEN(Line$) DIV 2));Line$
6590    END SELECT
6595  SUBEND
```

APPENDIX II - DATA PREPROCESSOR PROGRAM

```
' ***** ERIC P WHITENTON, NIST, FEB 1989 *****
OPTION BASE 1
DIM D#(200)
'
' ************************ CONSTANTS *******************
CONST False% = 0
CONST True% = -1
'
' ************************ MAIN PROGRAM ********************
PRINT
PRINT
PRINT "                              PREPROCESSOR"
PRINT "         (enter ^C during any question to terminate early)"
PRINT
PRINT "  This program will read in columns of numbers either with or without a header"
PRINT "and output a new file of the data with the rows averaged."
PRINT
'
'GET INPUT FILE DRIVE
PRINT
DoneEntry% = False%
DO
  INPUT "ENTER drive to input data from  ", InDrive$
  IF InDrive$ <> "" THEN DoneEntry% = True%
LOOP UNTIL DoneEntry%
IF INSTR(InDrive$, ":") = 0 THEN InDrive$ = InDrive$ + ":"
'
'SHOW WHAT IS ON THAT DRIVE
PRINT
ON ERROR GOTO DiskError1
  FILES InDrive$
ON ERROR GOTO 0
'
'GET INPUT FILE NAME
PRINT
DoneEntry% = False%
DO
  INPUT "ENTER input file name  ", InFile$
  IF InFile$ <> "" THEN DoneEntry% = True%
LOOP UNTIL DoneEntry%
'
'DETERMINE TYPE OF FILE (WITH OR WITHOUT HEADER), SHOW HEADER, AND GET NColumns%
ON ERROR GOTO DiskError1
  OPEN "I", #1, InDrive$ + InFile$
    LINE INPUT #1, Lyne1$
    LINE INPUT #1, Lyne2$
    IF INSTR(Lyne2$, CHR$(34)) <> 0 THEN
        Header% = True%
        NComments% = VAL(Lyne1$)
        IF NComments% <> 0 THEN
```

```
          PRINT
          PRINT "COMMENTS FOR '"; InDrive$ + InFile$; "'"
          PRINT Lyne2$
          FOR Counter% = 1 TO NComments% - 1
            LINE INPUT #1, Lyne$
            PRINT Lyne$
          NEXT Counter%
          INPUT #1, NColumns%
        ELSE
          PRINT
          PRINT "NO COMMENTS FOR '"; InDrive$ + InFile$; "'"
          NColumns% = VAL(Lyne2$)
      END IF
    ELSE
      Header% = False%
      NColumns% = 0
      Pointer% = 1
      WHILE (MID$(Lyne2$, Pointer%, 1) = " ") AND (Pointer% <= LEN(Lyne2$))
        Pointer% = Pointer% + 1
      WEND
      WHILE Pointer% <= LEN(Lyne2$)
        NColumns% = NColumns% + 1
        WHILE (MID$(Lyne2$, Pointer%, 1) <> " ") AND (Pointer% <= LEN(Lyne2$))
          Pointer% = Pointer% + 1
        WEND
        WHILE (MID$(Lyne2$, Pointer%, 1) = " ") AND (Pointer% <= LEN(Lyne2$))
          Pointer% = Pointer% + 1
        WEND
      WEND
      PRINT
      PRINT "NO HEADER FOR '"; InDrive$ + InFile$; "'"
  END IF
  PRINT
  PRINT NColumns%; "columns"
CLOSE #1
ON ERROR GOTO 0
'
'DECIDE IF THIS IS THE FILE YOU REALLY WANTED
PRINT
INPUT "ok to use this file? (ENTER 'Y')  ", Reply$
IF (Reply$ = "Y") OR (Reply$ = "y") THEN
  ON ERROR GOTO DiskError1
    '
    'GET OUTPUT FILE DRIVE
    PRINT
    DoneEntry% = False%
    DO
      INPUT "ENTER drive to output data to  ", OutDrive$
      IF OutDrive$ <> "" THEN DoneEntry% = True%
    LOOP UNTIL DoneEntry%
```

```
  IF INSTR(OutDrive$, ":") = 0 THEN OutDrive$ = OutDrive$ + ":"
  '
  'SHOW WHAT IS ON THAT DRIVE IF NEEDED
  IF (Header% = True%) OR (InDrive$ <> OutDrive$) THEN
    PRINT
    FILES OutDrive$
  END IF
  '
  'GET OUTPUT FILE NAME
  PRINT
  DoneEntry% = False%
  DO
    INPUT "ENTER output file name  ", OutFile$
    IF OutFile$ <> InFile$ THEN DoneEntry% = True%
  LOOP UNTIL DoneEntry%
ON ERROR GOTO 0
'
'ENTER ROWS TO AVERAGE
DoneEntry% = False%
PRINT
DO
  INPUT "ENTER the number of rows to average  ", RowsToAverage%
  IF RowsToAverage% > 0 THEN DoneEntry% = True%
LOOP UNTIL DoneEntry%
'
'MAKE OUTPUT FILE
ON ERROR GOTO DiskError1
  OPEN "I", #1, InDrive$ + InFile$
    OPEN "O", #2, OutDrive$ + OutFile$
      '
      'HEADER FOR OUTPUT FILE
      IF Header% THEN
          LINE INPUT #1, Lyne$
          PRINT #2, NComments% + 2
          FOR Counter% = 1 TO NComments%
            LINE INPUT #1, Lyne$
            PRINT #2, Lyne$
          NEXT Counter%
          PRINT #2, CHR$(34); "ROWS AVERAGED"; CHR$(34); " "; RowsToAverage%
          PRINT #2, CHR$(34); "ORIGINAL FILE NAME"; CHR$(34); " "; CHR$(34); InDrive$ + InFile$; CHR$(34)
          LINE INPUT #1, Lyne$
          PRINT #2, Lyne$
          LINE INPUT #1, Lyne$
          PRINT #2, Lyne$
        ELSE
          PRINT #2, 2
          PRINT #2, CHR$(34); "ROWS AVERAGED"; CHR$(34); " "; RowsToAverage%
          PRINT #2, CHR$(34); "ORIGINAL FILE NAME"; CHR$(34); " "; CHR$(34); InDrive$ + InFile$; CHR$(34)
          PRINT #2, NColumns%
      END IF
```

```
        '
        'AVERAGED ROWS FOR OUTPUT FILE
        DoneDisking% = False%
        WHILE NOT DoneDisking%
          FOR Col% = 1 TO NColumns%
            D#(Col%) = 0#
          NEXT Col%
          Rows% = 0
          DoneAveraging% = False%
          WHILE NOT DoneAveraging%
            LINE INPUT #1, Lyne$
            WHILE LEFT$(Lyne$, 1) = " "
              Lyne$ = RIGHT$(Lyne$, LEN(Lyne$) - 1)
            WEND
            FOR Col% = 1 TO NColumns%
              Pointer% = 1
              EndOfNumber% = False%
              WHILE NOT EndOfNumber%
                IF Pointer% = LEN(Lyne$) THEN EndOfNumber% = True% ELSE IF MID$(Lyne$, Pointer%, 1) = " " THEN EndOfNumb
                er% = True% ELSE Pointer% = Pointer% + 1
              WEND
              D#(Col%) = D#(Col%) + VAL(LEFT$(Lyne$, Pointer%))
              Lyne$ = RIGHT$(Lyne$, LEN(Lyne$) - Pointer%)
              WHILE LEFT$(Lyne$, 1) = " "
                Lyne$ = RIGHT$(Lyne$, LEN(Lyne$) - 1)
              WEND
            NEXT Col%
            Rows% = Rows% + 1
            IF Rows% = RowsToAverage% THEN DoneAveraging% = True%
            IF EOF(1) THEN DoneAveraging% = True%: DoneDisking% = True%
          WEND
          IF Rows% = RowsToAverage% THEN
            FOR Col% = 1 TO NColumns%
              Lyne$ = STR$(CSNG(D#(Col%) / Rows%))
              IF LEFT$(Lyne$, 1) = " " THEN Lyne$ = RIGHT$(Lyne$, LEN(Lyne$) - 1)
              IF Col% <> 1 THEN Lyne$ = " " + Lyne$
              PRINT #2, Lyne$;
            NEXT Col%
            PRINT #2,
          END IF
        WEND
      CLOSE #2
    CLOSE #1
  ON ERROR GOTO 0
END IF
PRINT
PRINT "PROGRAM DONE"
PRINT
END
 '
'
```

```
' ***************** DATA DISK ERROR *****************
DiskError1:
  RESUME NextLine
NextLine:
  ON ERROR GOTO DiskError2
    CLOSE
  ON ERROR GOTO 0
  PRINT
  PRINT "DISK ERROR !"
  PRINT
  END
DiskError2:
  RESUME NEXT
```

APPENDIX III - SAMPLE PRINTOUT

DATE: 7 Feb 1989
CYCLES PER DATA SET: 1
1 OF EVERY 25 DATA SETS PRINTED OUT
TYPE OF TEST: BLOCK ON RING
APPLIED LOAD (N): 33.10875
ULTIMATE DISK FILE NAME: B:SLCTST61.PRN
SPECIMEN MATERIALS: RING:440C PIN:Cu SOLID 99.999% LAB AIR ATMOSPHERE
LUBRICATION: NONE
RELATIVE HUMIDITY (%): 50 (UNCORRECTED AT START OF TEST)
SPEED (RPM): 75 (estimated)
OTHER COMMENTS: MECHANICAL ZERO TAKEN WOTH NO LOAD ON LOAD CELL

CALIBRATION FACTORS:	INTERCEPT	SLOPE	UNITS
Distance	0	.10972	meters
Friction	-.0454337327746	-2.82372484615	coef

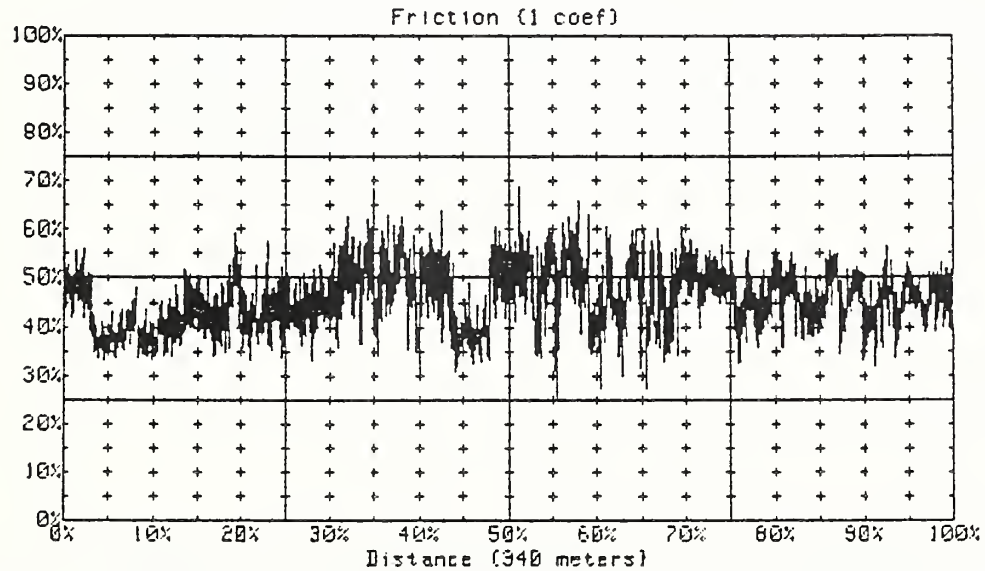
LAST CALIBRATED JAN 23 1989

Time(h:m:s)	Distance(meters)	Friction(coef)
00:00:00	0	.162
00:00:23	2.74	.503
00:00:43	5.49	.51
00:01:03	8.23	.449
00:01:24	10.97	.384
00:01:44	13.71	.384
00:02:04	16.46	.365
00:02:24	19.2	.369
00:02:44	21.94	.381
00:03:04	24.69	.423
00:03:24	27.43	.416
00:03:44	30.17	.391
00:04:04	32.92	.372
00:04:24	35.66	.392
00:04:44	38.4	.394
00:05:04	41.14	.342
00:05:24	43.89	.371
00:05:44	46.63	.515
00:06:04	49.37	.447
00:06:24	52.12	.409
00:06:45	54.86	.463
00:07:05	57.6	.451
00:07:25	60.35	.492
00:07:45	63.09	.404
00:08:05	65.83	.566
00:08:25	68.58	.436
00:08:45	71.32	.408
00:09:05	74.06	.462
00:09:25	76.8	.438
00:09:45	79.55	.38
00:10:05	82.29	.494
00:10:25	85.03	.387
00:10:45	87.78	.425
00:11:05	90.52	.415
00:11:26	93.26	.462
00:11:46	96	.439
00:12:06	98.75	.429
00:12:26	101.49	.435
00:12:46	104.23	.432
00:13:06	106.98	.463

00:13:26	109.72	.526
00:13:46	112.46	.532
00:14:06	115.21	.565
00:14:26	117.95	.516
00:14:46	120.69	.468
00:15:06	123.44	.539
00:15:26	126.18	.509
00:15:46	128.92	.625
00:16:07	131.66	.452
00:16:26	134.41	.452
00:16:47	137.15	.443
00:17:07	139.89	.54
00:17:27	142.64	.562
00:17:47	145.38	.507
00:18:07	148.12	.438
00:18:27	150.86	.349
00:18:47	153.61	.425
00:19:07	156.35	.395
00:19:27	159.09	.384
00:19:47	161.84	.388
00:20:07	164.58	.45
00:20:27	167.32	.551
00:20:47	170.07	.526
00:21:07	172.81	.559
00:21:27	175.55	.554
00:21:47	178.29	.574
00:22:07	181.04	.456
00:22:27	183.78	.51
00:22:47	186.52	.533
00:23:07	189.27	.514
00:23:27	192.01	.502
00:23:47	194.75	.599
00:24:07	197.5	.494
00:24:27	200.24	.508
00:24:47	202.98	.361
00:25:07	205.72	.526
00:25:27	208.47	.545
00:25:47	211.21	.408
00:26:07	213.95	.439
00:26:27	216.7	.542
00:26:47	219.44	.49
00:27:07	222.18	.6
00:27:27	224.93	.558
00:27:47	227.67	.47
00:28:07	230.41	.379
00:28:27	233.16	.503
00:28:47	235.9	.545
00:29:07	238.64	.482
00:29:27	241.38	.512
00:29:47	244.13	.491
00:30:07	246.87	.556
00:30:27	249.61	.477
00:30:47	252.36	.505
00:31:07	255.1	.489
00:31:27	257.84	.322
00:31:47	260.59	.462
00:32:07	263.33	.494
00:32:27	266.07	.487
00:32:47	268.81	.408
00:33:07	271.56	.535

00:33:27	274.3	.43
00:33:47	277.04	.465
00:34:07	279.79	.454
00:34:27	282.53	.374
00:34:47	285.27	.446
00:35:07	288.01	.388
00:35:27	290.76	.442
00:35:47	293.5	.513
00:36:07	296.24	.427
00:36:27	298.99	.409
00:36:47	301.73	.515
00:37:07	304.47	.505
00:37:26	307.22	.416
00:37:46	309.96	.379
00:38:06	312.7	.418
00:38:26	315.44	.44
00:38:46	318.19	.377
00:39:06	320.93	.456
00:39:26	323.67	.496
00:39:46	326.42	.48
00:40:06	329.16	.42
00:40:26	331.9	.478
00:40:46	334.65	.501
00:41:06	337.39	.503
00:41:24	339.91	.409

TEST TERMINATED (NORMAL TEST TERMINATION)
3099 REVOLUTIONS
5.3 READINGS PER CYCLE ON AVERAGE



NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 89-4107

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
July 1989

4. TITLE AND SUBTITLE

A Computer-Controlled Test System for Operating Different Wear Test Systems

5. AUTHOR(S)

Eric P. Whitenton and A. W. Ruff

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER
N00014-89-F-0021

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

Office of Naval Research
Code 1131
Arlington, VA 2217

10. SUPPLEMENTARY NOTES

[X] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

This report discusses a wear tester control system, where the same computer and software runs three different wear test machines; a commercial crossed-cylinder, a commercial block-on-ring, and an in-house designed controlled-atmosphere tribometer. The computer hardware, and the interface to the wear test machines, and the aspects that make the machines "functionally similar" are examined. The program itself, its use, and the data file structure are also explored.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

computer control system; software; test system; wear; wear testers; wear testing

13. AVAILABILITY

[XX] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[XX] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
49

15. PRICE
$12.95

ELECTRONIC FORM


AMERICAN BINDERY
2
5-94