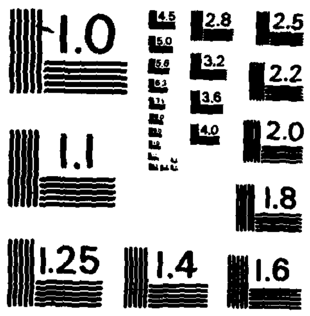

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ada 950 489

(12)

TECHNICAL REPORT RT-CR-83-2

CENTRALIZED MONITORING AND CONTROL OF ENVIRONMENTAL CHAMBERS BY DIGITAL COMPUTER

Stephen J. Dow
Department of Mathematics
The University of Alabama in Huntsville

August 1983

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898

Approved for public release; distribution unlimited

84 02 23 007

UNCLASSIFIED

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

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RT-CR-83-2	2. GOVT ACCESSION NO. AD-A138 278	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* CENTRALIZED MONITORING AND CONTORL OF ENVIRONMENTAL CHAMBERS BY DIGITAL COMPUTER		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Stephen J. Dow		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Mathetics The University of Alabama in Huntsville Huntsville, Alabama 36104		10. PROGRAM ELEMENT, PROJECT, TASK [redacted] DAAG29-81-D-0100
11. CONTROLLING OFFICE NAME AND ADDRESS Commander US Army Missile Command/ATTN: DRSMI-RPT Redstone Arsenal, Alabama 35898		12. August 1983 13. NUMBER OF PAGES 37
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Commander US Army Missile Command/ATTN: DRSMI-RT Redstone Arsenal, Alabama 35898		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

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

Environmental Chamber
Controlled Environment

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

This report documents the system which is being developed to monitor and control environmental chambers operated by the US Army Missile Laboratory at Redstone Arsenal, Alabama. It also touches on the control theory needed and describes in some detail the software at its present stage of development. (author)

TABLE OF CONTENTS

Page No.

I. INTRODUCTION 3

II. THE EXISTING SYSTEM. 3

III. BASIC ORGANIZATION OF THE NEW SYSTEM 4

IV. SOME CONCEPTS FROM CONTROL THEORY. 5

V. THE CONTROLLED VARIABLES 7

VI. SOFTWARE 7

VII. THE CONTROL EQUATION: ADJUSTMENT OF PARAMETERS 12

APPENDIX - THE SOURCE CONTROL. 14

ACKNOWLEDGMENT

The author would like to thank James Sullivan and Larry Hurst of Test and Evaluation Directorate, US Army Missile Laboratory, US Army Missile Command, for their aid in writing the software that contributed to the completion of this report.

This report constitutes part of the research objectives of the author under the Summer Faculty Research Program sponsored by the US Army Research Office and Battelle Columbus Laboratories.

I. INTRODUCTION

The Environmental and Metrology Branch, Components Test Division of the Army Missile Command's Test and Evaluation Directorate at Redstone Arsenal houses a number of environmental chambers in which missiles and components are subjected to various controlled environmental conditions. These chambers have long been controlled by means of individual mechanical controllers. Currently a system is being developed to monitor and control the chambers by computer. This report overviews the new system and the progress made toward implementing it. It also touches on the control theory needed and describes in some detail the software at its present stage of development. The source code programs are included in the appendix.

II. THE EXISTING SYSTEM

The existing system for controlling temperature in a chamber is described in this section. The controller is only one part of the overall system. The chamber has heating/cooling equipment and a thermocouple to sense the actual temperature in the chamber. The desired temperature is called the setpoint. The controller is connected to the heating/cooling equipment and to the probe; it is the job of the controller to monitor the temperature and respond through the heating/cooling equipment to differences between the temperature reading and the setpoint. If desired, the controller can change the setpoint automatically according to a predetermined schedule called a test program or profile.

The controller is connected to the heating/cooling equipment by an air line. The pressure in this line determines the level of heating or cooling administered to the chamber. Heating or cooling occurs when the pressure is above or below a certain deadband pressure (usually 10 psi). Greater heating or cooling occurs the further the pressure is from deadband. The controller controls the pressure by bleeding off air from the 20 psi supply line.

The controller monitors temperature by a gauge equippped with a pen behind which a paper temperature chart rotates. Thus a record of the temperature over time is produced. The setpoint may be given to the controller either manually or automatically. Manual setpoints are given by positioning a setpoint needle on the temperature chart. Automatic setpoint control is achieved by installing a cam on the program controller. This cam is custom cut for the particular test program. A setpoint arm rides on the edge of the rotating cam. The relative positions of the temperature gauge and the setpoint needle or arm determine the controlled air pressure by a system of gears, springs, etc. within the controller.

Figure 1 illustrates the components of the existing system and their interconnections. It should be noted that separate controllers exist for each chamber but that a central air supply is connected to all controllers.

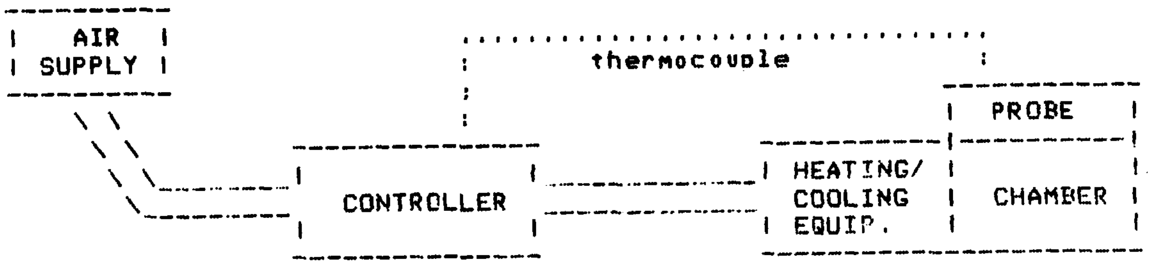

Figure 1. Components of the existing system and their interconnections.

III. BASIC ORGANIZATION OF THE NEW SYSTEM

The hardware for the new system consists of an HP2250 measurement and control processor, an HP1000 computer, and peripheral devices for the HP1000 (keyboard, display terminal, printer, and mass storage device). Also, a current to pneumatic (I/P) convertor and a 3-way valve are connected to the air lines of each chamber as shown in Figure 2.

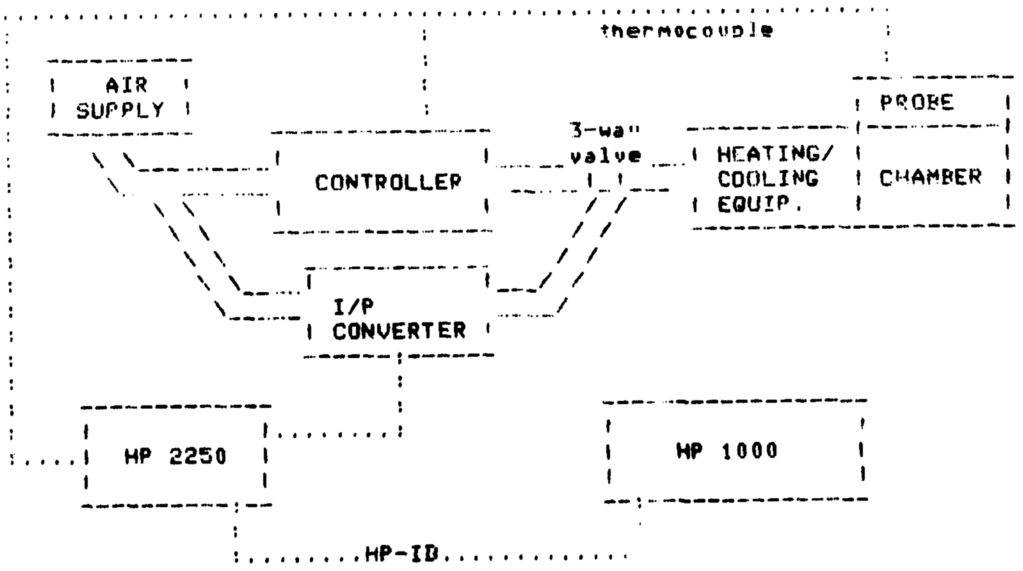

Figure 2. I/P converter and 3-way valve connected to air lines of each chamber.

The 3-way valve is the point at which the control is shifted from the existing controller to the new system. Depending on the 3-way valve's setting, the air pressure leading to the heating/cooling equipment is controlled by either the existing controller or the I/P convertor. The I/P convertor translates a 4-20 milliamp current input from the HP2250 into a 0-20 psi air pressure output. The current sent by the HP2250 to the I/P convertor is determined by the control programs within the HP1000. It should be noted that a single computer system (HP2250 and HP1000) controls all chambers. Thus the HP2250 has separate output lines to each chamber's I/P convertor and separate inputs for each chamber's thermocouple. Because the 3-way valves may be set individually, any combination of chambers may be shifted to computer control. The new system may monitor temperature (without controlling it) regardless of the setting of the 3-way valves.

IV. SOME CONCEPTS FROM CONTROL THEORY

In any control system, the controller must have some means of determining what response to give to a difference between the setpoint and the measurements of the controlled variable (in sections 2 and 3, temperature). The method used predominantly is PID control, which stands for preportional-integral-derivative. The most fundamental of the three modes is proportional control. Here the controller output is given by

OUTPUT = BIAS + GAIN*ERROR,

where ERROR = SP - CV, SP = setpoint, and CV = controlled variable. BIAS and GAIN are constants that are adjusted to make the system give the best results. As can be seen from the above equations, the controller output will equal BIAS when the controlled variable is at setpoint. Also, the amount by which the output varies from BIAS is proportional to how much the controlled variable is off setpoint. GAIN is the constant of proportionality.

The integral and derivative modes of control may be regarded as refinements of proportional control. The output for PID control is given by

OUTPUT = BIAS + GAIN*(ERROR + K1*I(E) + K2*D(E)),

where I(E) is the time integral of ERROR, D(E) is the derivative of ERROR with respect to time, and K1 and K2 are constants used to adjust the integral and derivative terms. If K1 and K2 are chosen to be zero, then the equation reduces to that for proportional control.

Consider the effect of making K1 a nonzero (usually positive) constant. If ERROR is zero and has been zero since starting control, then OUTPUT will equal BIAS. Suppose that ERROR becomes nonzero for a time and then returns to zero and remains there; i.e., the controlled variable gets off setpoint and then returns to setpoint and holds steady. During the time ERROR is nonzero, I(E) will accumulate; then I(E) will become constant, but nonzero, after the controlled variable returns to setpoint. Thus OUTPUT will take on a new value unequal to BIAS when the controlled variable returns to setpoint.

This is called reset action; the integral term in the control equation is sometimes called the reset term. It is especially useful when the controlled variable is part of a process in which changing conditions cause a need for the BIAS to be reset.

We now turn to the derivative mode of control. Recall that D(E) is the derivative of ERROR with respect to time; i.e., the rate of change of ERROR. When ERROR is constant, D(E) = 0, so this term has no effect on OUTPUT. When ERROR is increasing, D(E) is positive. When ERROR is decreasing, D(E) is negative. In the following discussion the integral mode will be disregarded in order to concentrate on the effect of the derivative mode. Suppose that temperature is the controlled variable and that it is below set point. Then ERROR = SP - CV will be positive. Suppose that K2 = 0. Then OUTPUT will be greater than BIAS, so that heat is administered. As the temperature increases and approaches setpoint, ERROR decreases. ERROR remains positive until the temperature reaches setpoint, so that heat continues to be administered in decreasing amounts. If there is any lag between the call for heat from the controller and the time the temperature actually rises, then leaving the heat on up to the time temperature reaches setpoint causes the temperature to rise beyond setpoint. This is called overshoot.

Introducing derivative control by making K2 positive can help lessen overshoot. Reconsider the events described above assuming K2 > 0. Again ERROR starts out positive. Assume also that ERROR is not changing at the outset, so that D(E) = 0. OUTPUT is again above BIAS and heat is administered. As the temperature increases toward setpoint, ERROR decreases and D(E) < 0. K2*D(E) < 0, so OUTPUT is decreased by the derivative mode. As long as ERROR is large, ERROR + K2*D(E) will be positive and the heat remains on. Eventually however, when the temperature is close to setpoint, ERROR + K2*D(E) will become negative and the heat is turned off. This occurs before ERROR itself becomes negative, i.e., before the temperature reaches setpoint. The derivative mode anticipates overshoot and corrects for it. For this reason it is sometimes called preact control.

It should be noted that the mechanical controller does not compute output by an equation, but rather by physical components whose action gives the same results. In contrast, the computer does calculate output by a control equation similar to that given above. For the mechanical controller OUTPUT represents the air pressure output in psi. For the computer system OUTPUT represents the current output in milliamps. This distinction is not important though, since the I/P convertor performs a linear conversion from milliamps to psi. A more significant distinction is that the mechanical controller performs continuous control whereas the computer system performs digital control. This is inherent in the use of a digital computer. It means that temperature readings are received by the control programs at discrete time intervals rather than continuously. Thus ERROR is a sampled-data variable. The HP2250 sends current continuously to the I/P convertor but the current level sent changes only at discrete time intervals. With each new ERROR it receives, the computer calculates OUTPUT from the control equation, which then becomes the new milliamp level.

Because ERROR is a discrete rather than a continuous variable, I(E) and D(E) must be replaced by digital approximations to the integral and derivative. Here I(E) is replaced by the sum of the sequence of values of ERROR and D(E) is replaced by the difference NEW ERROR - OLD ERROR. More will be said later about the control equation and adjustment of its constants in Section VII.

V. THE CONTROLLED VARIABLES

Building 7290 of the Environmental and Metrology Branch contains nine environmental chambers. These chambers along with those in other buildings are described in a brochure entitled Test and Evaluation Directorate, Facilities and Capabilities. It is anticipated that computer control and monitoring of environmental chambers will be extended to the other buildings, but implementation of this extension has not yet begun.

Each of the nine chambers has provision for (dry bulb) temperature control. Some of the chambers can also be controlled for humidity and/or altitude tests. Humidity tests are performed by controlling wet bulb temperature in conjunction with dry bulb temperature. For altitude tests the air pressure in the chamber is controlled along with dry bulb temperature. We speak of altitude rather than air pressure as being the controlled variable, with the measurement units being feet. Altogether there are fourteen controlled variables for the nine chambers; there exist fourteen individual mechanical controllers for these variables. The wet bulb and altitude controllers function in much the same way as the dry bulb controllers described in the previous sections.

Other environmental conditions (such as sunshine, sand and dust, etc.) can be simulated in the chambers, but these conditions have manual on/off control for which the computer system is not expected to be used.

VI. SOFTWARE

A. Introduction

This section describes the software system developed for use in the environmental test site under discussion. The specific operational procedures will not be stressed here, but rather the emphasis will be on overall program organization and flow of program control. This section is intended not only for operators of the system at hand but also for those interested in digital control in general.

B. The Programs and Their Functions

The programs written for use in controlling the environmental chambers fall into four groups as follows:

1. CMCP (centralized monitoring and control program)

2. Control Programs

(a) CNTRL

(b) GETSP

3. Utility Programs

(a) PREP

(b) ALTER

(c) DISPL

(d) CHECK

(e) TABLE

4. MAC (measurement and control)

The programs in the first three groups have been written in the FORTRAN language, compiled, loaded and readied for execution in the HP1000. The remaining program, MAC, was written in Measurement and Control Language (MCL) of the HP2250.

CMCP serves as the driver program: The control and utility programs are started by calls within CMCP. CNTRL communicates with the HP2250 to control the chambers and GETSP implements automatic setpoint control. CNTRL and GETSP do not read from or write to the CRT screen and so do not provide for communication with the operator. This is the function of the utility programs.

C. Flow of Control

The start-up procedure includes instructions which load program MAC into the HP2250 and start execution of CMCP in the HP1000. At the beginning of the execution the calls which start programs CNTRL and GETSP occur. CNTRL and GETSP then run continuously as long as any chambers are being controlled. CMCP then writes a message to the screen requesting the operator to choose one utility program. CMCP execution is suspended while the utility program chosen runs. When the utility program has finished running, control returns to CMCP and the same request is repeated.

There are three programs running at any one time: CNTRL, GETSP, and either CMCP or one of its utility programs. These programs do not actually run simultaneously since the computer can have only one program running at a given instant. However, through time slicing the computer shifts rapidly from execution of one program to another, producing the effect of simultaneous execution.

D. The Test File

An environmental test consists of subjecting a test item to specified environmental conditions on a specified time schedule. For each test to be run by computer control a test file is created and consist of sequential records.

This file is stored on an external storage device by program PREP. The sequence alternates between setpoint records and time records, with the first and last records being the setpoint. A setpoint record contains a setpoint and allowable high and low deviations from setpoint. Two setpoint records together with the time record between them provide the information for one interval of the test. The two setpoints are the beginning and ending setpoints for the interval, and the time record contains the length of that interval. If the beginning and ending setpoints are the same the interval is a "soak"; if they are different the interval is a "ramp."

If the test is an altitude or humidity test each setpoint record contains two setpoints, one for the dry bulb temperature and another for the second controlled variable.

E. Communication Between the Programs

All communication between the HP2250 and the Fortran programs running in the HP1000 occurs through program CNTRL. This program contains WRITE statements to the HP2250 in which MCL statements are sent as literals. In some cases the MCL statement is a START command which causes a group of MCL statements (called a task) in program MAC to be executed.

Other than the creation and reading of test files all communication between the Fortran programs is via system common, a region of memory accessible to all programs. Two arrays are stored in system common: a two-dimensional array called HOST and a one-dimensional array called BUFF.

A particular location in HOST is indicated by HOST(K,L), where K is an integer between 1 and 14 and L is an integer between 1 and 9. When visuallizing HOST as a table we think of K as representing the row number and L the column number. The rows of this table correspond to the fourteen controlled variables described in Section V. The columns correspond to the various parameters needed for each controlled variable; these include the current measurement, setpoint, and status, as well as the constants used in the control equation.

The other array, BUFF, is used only by programs ALTER and GETSP for starting tests. Program ALTER allows the operator to type in the row number(s) of HOST corresponding to the variable(s) to be controlled and the name of the test file to be used. This information is stored in BUFF where program GETSP has access to it. Then GETSP starts reading from the test file and gets new setpoints whenever the system clock shows that the time for the current interval has elapsed. The setpoints are communicated to program CNTRL by their storage in the HOST array.

F. Program CNTRL

All statements directing communication between the HP2250 and the HP1000 occur within program CNTRL. The program consists of initialization statements followed by the control loop. The initialization statements are executed only once. They initialize the control equation parameters and initialize the setpoints to -100.0, which is a special value used to indicate that no test is running. Once the control loop is entered, it repeats indefinitely. Each pass through the control loop causes the following actions:

1. Task 1 in program MAC is executed; this causes the HP2250 to average 200 readings over 1/60 second from each controlled variable; the averaging eliminates 60 cycle noise.

2. The averaged measurements for all controlled variables are read in from the HP 2250 and these new measurements are entered into the HOST array.

3. An output is computed for each controlled variable; if the setpoint is -100.0 this output is simply the bias for that controlled variable, otherwise the output is computed using the control equation.

4. The numerical outputs are sent to the HP 2250.

5. Task 2 in program MAC is executed; this causes the HP 2250 to send the new milliamp levels of current to the I/P convertors.

6. The status of each controlled variable is checked. The status is a number stored in the HOST array indicating if the controlled variable is out of tolerance. Depending on its value, an instruction may be sent to the HP 2250 to turn on or off an alarm bell or to shut down a chamber.

G. Program GETSP

The main functions of GETSP are to keep track of which chambers tests are being run in and for each of those chambers to implement automatic setpoint control and monitor the status of the test.

At the beginning of execution GETSP initializes all elements of arrays TOTAL and MODE2 to zero. As GETSP runs, TOTAL(K) will contain the length (in seconds) of the current interval of the test being run in the chamber whose dry bulb reference number (row number of HOST) is K, or will contain zero if that chamber is inactive. Recall that every test includes dry bulb temperature control. Array MODE2 is used to indicate whether a test involves a second controlled variable. If K is the dry bulb reference number of an active chamber and wet bulb temperature or altitude is being controlled, then MODE2(K) contains the reference number of the second controlled variable; otherwise MODE2(K) will be zero. TOTAL and MODE2 keep tra^k of which controlled variables are active as well as the current interval times.

To get tests started GETSP periodically checks BUFF(1). Recall that program ALTER stores the name of the test file and the reference number(s) of the controlled variable(s) in BUFF when a test is started. It also sets BUFF(1) equal to 1 to indicate to GETSP that such information is available. If BUFF(1) equals 1 when GETSP checks it, then GETSP reads in the test file name and reference number(s) and sets BUFF(1) back to zero. The program segment which reads from the test file is then executed. In this segment the beginning and ending setpoints and length of time for one interval are read; also the system clock is read and the interval start time stored.

The main loop in GETSP does the following steps for each controlled variable reference number K.

1. If TOTAL(K) is zero, skip to the next K.

2. If the status is 1, then the chamber is on hold; increment hold time and proceed to the next K.

3. Read the system clock and calculate the elapsed time since the start of the current interval, subtracting any accumulated hold time.

4. Check to see if the controlled variable is out of tolerance; if so the status is changed from normal (0) to alert (2) and the alarm clock is incremented. If the alarm clock shows that the chamber has been on alert for 60 seconds, then the status is changed to alarm.

5. Calculate the setpoint according to the formula:

SP = INITSP + (ELAPSE/TOTAL)*SPDIFF

where INITSP is the initial setpoint for the current interval, ELAPSE is the elapsed time since the start of the interval, TOTAL is length of the interval, and SPDIFF is difference between the initial and final setpoints for the interval. Store the setpoint in the HOST array.

6. Repeat the setpoint calculation for the second controlled variable, if applicable.

7. Compare the elapsed time with TOTAL to see if it is time to change intervals; if so the program segment which reads from the test file is entered.

H. The Utility Programs

The utility programs allow the operator to communicate with the computer while the control programs are running. Programs PREP and ALTER allow operator "keyboard" input. Programs DISPL, TABLE, and CHECK are used to display information on the screen.

1. PREP

This program creates the test file described in Section D.

2. ALTER

This program is used to start tests and to change values in the HOST array. A test file must already have been created by PREP to start a test. The operator enters the filename and the reference numbers of the controlled variables for the chamber in which the test is to be run. This information is stored in the common array BUFF, which is accessed by GETSP.

3. DISPL

This is intended to be the program which will remain running most of the time during normal operation of tests. It writes a display form on the screen and enters into this form the setpoints and measurements of all controlled variables which are currently involved in active tests (a controlled variable is active whenever its setpoint is not -100.0). DISPL is the only utility program which does not terminate itself. Instead it continually updates the setpoints and measurements on the screen until the operator breaks its execution by hitting a special key.

4. CHECK

This program allows the operator to display on the screen or on paper the contents of any test file previously created by program PREP.

5. TABLE

This program displays the current contents of all entries in the HOST array on the screen.

I. Program MAC

MAC is a special program written in MCL, the language of the HP2250. It is broken up into a number of tasks, each consisting of several individual HP2250 commands which are to be executed as a group. A given task is executed which the HP2250 receives a start instruction from program CNTRL (see Sections E and F).

VII. THE CONTROL EQUATION; ADJUSTMENT OF PARAMETERS

The control equation used in program CNTRL is

OUTPUT = BIAS + GAIN*(ERROR + INTGL*SUM + RATE*DIFF),

where SUM = sum of all errors up to the current one and DIFF = NEW ERROR - OLD ERROR; BIAS, GAIN, INTGL, and RATE are the control equation parameters discussed in section 4 (INTGL and RATE are called K1 and K2 there).

Each chamber has its own set of control parameters which must be determined in advance of using the computer system on actual test items. The BIAS for dry bulb temperature for a chamber is found by sending various milliamp levels of current to the I/P convertor and observing which level causes no heating or cooling to occur. The GAIN, INTGL, and RATE are then determined by repeated trial temperature control of the empty chamber using different values for these parameters. Experience in adjusting these parameters for two of the chambers brought out the following observations:

1. If the temperature will not keep up with setpoint changes occurring during a ramp or will not reach setpoints far from ambient, then GAIN may be too small. On the other hand, too large a GAIN may cause large overshoots during cycling of the temperature above and below setpoint.

2. Closer control can be obtained by introducing the derivative mode (making RATE positive). If RATE is too large, control breaks down. For example, the heat may be turned off by a small increase in temperature even though the temperature is well below setpoint.

3. The integral mode has not been found particularly useful so far. A possible explanation is that conditions are basically stable, making reset unnecessary. The problem encountered with the integral mode is sometimes called reset windup. This phenomenon consists of the normal cycling about setpoint being amplified by the reset term.

Trial temperature control was performed on one of the chambers, known as the ARC chamber, during July 1983. After trials with various values of the parameters, successful control was obtained with BIAS = 10700 (10.7 milliamps), GAIN = 0.8, INTGL = 0, and RATE = 0.9. The test profile consisted of a 20 minute initial interval at 100°F, followed by a 20 minute ramp from 100°F to 130°F, and finally a 1 hour soak at 130°F. After reaching the initial setpoint the temperature never varied from setpoint more than 2.5°F. During the final soak the largest error recorded was 1.7°F and the temperature stayed within one degree of setpoint during the last half of this soak.

APPENDIX: THE SOURCE CODE

```
      PROGRAM CMCP
C
C
      INTEGER READIN,TEST(3),DSP(3),ALTR(3),CHCK(3),HO(3),CNT(3),GET(3)
      INTEGER ESC,CUR,CLD,BEL
      DATA TEST/'PREP '/,DSP/'DISPL'/,ALTR/'ALTER'/,CHCK/'CHECK'/
      DATA HO/'TABLE'/,CNT/'CNTRL '/,GET/'GETSP '/
C
      ESC=27
      CUR=72
      CLD=74
      BEL=7
      WRITE(1,'(5A2)') ESC,CUR,ESC,CLD,BEL
      WRITE(1,'("Do you want to start CNTRL? (Y) (N) _")')
      READ(1,'(A2)') READIN
      IF(READIN.EQ.2HY ) CALL EXEC(10,CNT)
C
      WRITE(1,'(/"Do you want to start GETSP? (Y) (N) _")')
      READ(1,'(A2)')READIN
      IF(READIN.EQ.2HY ) CALL EXEC(10,GET)
C
  200 WRITE(1,'(4A2)') ESC,CUR,ESC,CLD
C                                                 *  CLEAR DISPLAY       *
  201 WRITE(1,'("CMCP (PR,DI,CH,AL,TA,EX to exit)  ")')
C                                                 *  PROMPT OPERATOR FOR *
C                                                 *  INPUT               *
      READ(1,'(A2)') READIN
C
        IF(READIN.EQ.2HAL) THEN
          CALL EXEC(9,ALTR)
          GO TO 200
          ENDIF
C                                                 *  START ALTER         *
        IF(READIN.EQ.2HPR) THEN
          CALL EXEC(9,TEST)
          GO TO 200
          ENDIF
C                                                 *  START PREP          *
        IF(READIN.EQ.2HDI) THEN
          CALL EXEC(9,DSP)
          GO TO 200
          ENDIF
C                                                 *  START DISPL         *
        IF(READIN.EQ.2HCH) THEN
          CALL EXEC(9,CHCK)
          GO TO 201
          ENDIF
C                                                 *  START CHECK         *
```

```
        IF(READIN.EQ.2HTA) THEN
          CALL EXEC(9,HO)
          GO TO 200                                  *  START TABLE            *
          ENDIF
C       IF(READIN.EQ.2HEX) GO TO 400                 *  EXIT                   *
C
        WRITE(1,'(//" ** INPUT ERROR **")')
        GO TO 201
C
  400 STOP
      END
      PROGRAM CNTRL
C
C
      INTEGER ITEMP(12),IOUT(14),IBUFF(100),CCODE,ERROR
      REAL ERR(14,4),RESET(14),INTGL,OUT(14)
      COMMON HOST(14,8)
C
      DO 10 K=1,14
      DO 5 L=1,8
    5 HOST(K,L)=0
      HOST(K,2)=-100.
      RESET(K)=0
      DO 10 L=1,4
   10 ERR(K,L)=0
      HOST(1,8)=9900.
      HOST(2,8)=10700.
      HOST(1,3)=1.6
      HOST(2,3)=.8
      HOST(1,5)=0.7
      HOST(2,5)=0.9
C**********************************************************************
C     Start TASK(3) in the HP2250.  This task remains running
C     constantly and serves only to keep a background task active.
C**********************************************************************
      WRITE(26,100)
  100 FORMAT("START(3)!")
C*********************************************************************
C     Begin main program loop.  The first statement starts the
C     data acquisition task, which writes the HP2250 input data
C     into PORT A.
C*********************************************************************
   20 WRITE(26,200)
  200 FORMAT("START(1)!")
C
      READ(26,IOSTAT=ERROR) CCODE
      IF(ERROR.NE.0.AND.ERROR.NE.496) CALL EROR(CCODE,ERROR)
      IF(CCODE.NE.0) CALL EROR(CCODE,ERROR)
      CALL EXEC(12,0,2,1,-1)
C**********************************************************************
C     Now read PORT A to get the data RELEASED from B1
C**********************************************************************
      READ(26:11,IOSTAT=ERROR) ITEMP
      IF(ERROR.NE.0.AND.ERROR.NE.496) CALL EROR(CCODE,ERROR)
C                                    *  Read the 12 thermocouples; the 2 *
C                                    *  altitudes will be read later on. *
      DO 80 K=1,12
C                                    *  Start the output calculations    *
      BIAS = HOST(K,8)
      TEMP = FLOAT(ITEMP(K))
      TEMP = TEMP/10.
C                                    *  The 2250 returns temps in integer*
C                                    *  format and degrees C.            *
```

```
      HOST(K,1) = 1.8*TEMP + 32.
C                                    *  Convert temp from Deg C to Deg F *
C                                    *  for display.                     *
      IF(HOST(K,2).NE.-100.) THEN
         ERR(K,4) = ERR(K,3)
         ERR(K,3) = ERR(K,2)
         ERR(K,2) = ERR(K,1)
         ERR(K,1) = HOST(K,2) - HOST(K,1)
         DIFF = ERR(K,1)+ERR(K,2)-ERR(K,3)-ERR(K,4)
         GAIN = HOST(K,3)
         INTGL = HOST(K,4)
         RATE = HOST(K,5)
         RESET(K) = INTGL*ERR(K,1)+RESET(K)
C
         OUT(K) = BIAS+GAIN*(ERR(K,1)+ERR(K,2)+RESET(K)+RATE*DIFF)*1000.
C                                    *  PID equation to calculate output *
         IOUT(K) = IFIX(OUT(K))
      ELSE
         IOUT(K) = BIAS
      ENDIF
C
      IF(HOST(K,6).NE.0.0) THEN
      WRITE(6,'( " Chamber ",I2,2X,"Diff=",F6.3,2X,"Reset=",F6.2,2X,
     * "Error=",F6.2,2X,"Out1=",F9.1)') K,DIFF,RESET(K),ERR(K,1),OUT(K)
      ENDIF
C                                    *  Print process parameters         *
      IF(OUT(K).GT.20000.) IOUT(K) = 20000
      IF(OUT(K).LT.4000.) IOUT(K) = 4000
      IF(HOST(K,7) .EQ. 2.0 .OR. HOST(K,7) .EQ. 3.0) THEN
         WRITE(26,'("DO(15,29)1!")')
C                                    *  If alert/alarm condition exists  *
C                                    *  turn on the bell.                *
         READ(26,IOSTAT=ERROR) CCODE
         IF(CCODE.NE.0) CALL EROR
      ENDIF
      IF(HOST(K,7).EQ.4.) THEN
         WRITE(26,'("DO(15,29)0!")')
         WRITE(26,'("DO(15,",I2,")0!")') K*2-1
C                                    *  If alert/alarm mode has been     *
C                                    *  disabled, turn off the bell      *
C                                    *  and allow the chamber to be      *
C                                    *  turned on.                       *
         READ(26,IOSTAT=ERROR) CCODE
         IF(CCODE.NE.0) CALL EROR
      ENDIF
      IF(HOST(K,7).EQ.3.) THEN
        WRITE(26,'("DO(15,",I2,")1!")') K*2-1
        READ(26,IOSTAT=ERROR) CCODE
        IF(CCODE.NE.0) CALL EROR
      ENDIF
   80 CONTINUE
C
```

```
C    ************************************************************
C          Now send the output to the air supply line
C          26:7 Write to variable 10 in the HP2250
C    ************************************************************
C
      WRITE(26:7,IOSTAT=ERROR) 10,12,IOUT(1),IOUT(2),IOUT(3),IOUT(4),
     * IOUT(5),IOUT(6),IOUT(7),IOUT(8),IOUT(9),IOUT(10),IOUT(11),
     * IOUT(12)
      IF(ERROR.NE.0.AND.ERROR.NE.496) CALL EROR(CCODE,ERROR)
C
      WRITE(26,25)
   25 FORMAT("START(2)!")
C
        READ(26,IOSTAT=ERROR) CCODE
        IF(ERROR.NE.0.AND.ERROR.NE.496) CALL EROR (CCODE,ERROR)
        IF(CCODE.NE.0) CALL EROR
C
      CALL EXEC(12,0,2,1,-1)
C
      GO TO 20
      END
      SUBROUTINE EROR (CCODE,ERROR)
      INTEGER CCODE,ERROR
      ITLU=6
      WRITE(ITLU,400) CCODE,ERROR
  400 FORMAT(" CONDITION CODE=",I6,2X,"FORTRAN ERROR=",I6)
      END
      PROGRAM GETSP
C
C
      INTEGER DCB(144),FILEN(14,3),REC(14),ENDREC(14),STIME(14,5)
      INTEGER NAME(3),ITIME(3),MODE2(10),REC1(2),BUFF,CLOCK(5)
      INTEGER HTIME(5),START(5)
      INTEGER*4 TOTAL(14),ELAPSE,HOLD(14),HTSAVE(14),ISEC,ALMCLK(14)
      INTEGER*4 ALRMSV(14),ALMSEC
      REAL STTEMP(14),SPDIFF(14),CURRSP(6),NEXTSP(6),SPSAVE(14)
      REAL HIDEV(14),LODEV(14)
      COMMON HOST(14,8),BUFF(10)
C
      BUFF(1)=0
      K = 0
      DO 5 K=1,14
         HOST(K,7)=0.
         HOLD(K)=0
         HTSAVE(K)=0
         TOTAL(K)=0
         DO 5 J=1,5
    5    STIME(K,J)=0
      DO 7 M=1,10
    7    MODE2(M)=0
C*****************************************************************
C     Begin main program loop.  Control remains in this loop except  *
C     to check for a new test and when it's time to change intervals.*
C*****************************************************************
   20 K=K+1
      IF(K .GT. 14) GO TO 30
      IF(TOTAL(K) .EQ. 0) GO TO 20
C                                *  TOTAL(K) is used to store the       *
C                                *  current interval's duration. It will *
C                                *  be 0 if no test is running.          *
      IF(HOST(K,7) .EQ. 1.) THEN
         CALL EXEC(11,HTIME)
         ISEC = HTIME(5)*86400 + HTIME(4)*3600 + HTIME(3)*60 + HTIME(2)
         IF(HTSAVE(K) .EQ. 0) GO TO 22
         HOLD(K) = HOLD(K) + ISEC - HTSAVE(K)
   22    HTSAVE(K) = ISEC
         GO TO 20
      ELSE
         HTSAVE(K) = 0
      ENDIF
C                                * Calculate HOLD time.                 *
      CALL EXEC(11,CLOCK)
      ELAPSE = (CLOCK(5)-STIME(K,5))*86400 + (CLOCK(4)-STIME(K,4))*3600
     *  + (CLOCK(3)-STIME(K,3))*60 + (CLOCK(2)-STIME(K,2)) - HOLD(K)
C                                *  Calculate the elapsed time from the  *
C                                *  start of the interval (STIME) to the *
C                                *  present time (CLOCK).                *
```

```
      IF(SPSAVE(K) .NE. HOST(K,2)) THEN
         SPDIFF(K) = 0.
         STTEMP(K) = HOST(K,2)
      ENDIF
C                                *  Check to see whether the setpoint    *
C                                *  has been changed by ALTER.           *
      IF((HOST(K,1).GT.(HOST(K,2)+HIDEV(K)).OR.HOST(K,1).LT.(HOST(K,2)
     * +LODEV(K))).AND.HOST(K,7).NE.4.) THEN
         HOST(K,7) = 2.
         HOST(K,6) = 1.
         ALMSEC = CLOCK(5)*86400 + CLOCK(4)*3600 + CLOCK(3)*60 +CLOCK(2)
         IF(ALRMSV(K).EQ.0) GO TO 27
         ALMCLK(K) = ALMCLK(K) + ALMSEC - ALRMSV(K)
   27    ALRMSV(K) = ALMSEC
         IF(ALMCLK(K).GE.60) HOST(K,7) = 3.
C                               *  If the alarm clock is greater than 60   *
C                               *  seconds, then the alarm condition       *
C                               *  leaves the alert mode and goes into a   *
C                               *  chamber shut-down mode. This also means *
C                               *  the operator was not around to disable  *
C                               *  the bell and take manual control.       *
      ELSE
         ALMCLK(K) = 0
         ALRMSV(K) = 0
      ENDIF
C                               *   Alarm conditions are held in           *
C                               *   HOST(K,7):     0 - normal              *
C                               *      1 - hold (manual)                   *
C                               *      2 - alert (bell)                    *
C                               *      3 - alarm (shut down)               *
C                               *      4 - disable alert/alarm bell        *
      HOST(K,2) = (FLOAT(ELAPSE)/FLOAT(TOTAL(K)))*SPDIFF(K)+STTEMP(K)
C                                *  Calculate the ramp if SPDIFF NE 0    *
      SPSAVE(K) = HOST(K,2)
C                                *  Save setpoint for next check         *
      IF(MODE2(K).NE.0) THEN
         K2 = MODE2(K)
         IF(SPSAVE(K2) .NE. HOST(K2,2)) THEN
            SPDIFF(K2) = 0.
            STTEMP(K2) = HOST(K2,2)
         ENDIF
         HOST(K2,2) = (FLOAT(ELAPSE)/FLOAT(TOTAL(K)))*SPDIFF(K2)+
     *   STTEMP(K2)
         SPSAVE(K2) = HOST(K2,2)
      ENDIF
C                                *  Repeat for second controlled variable *
      IF(ELAPSE.GE.TOTAL(K)) GO TO 55
C                                *  Is it time to change intervals?       *
      GO TO 20
C
   30 CALL EXEC(12,0,2,5,-5)
C                                *  Have finished all controllers so     *
C                                *  take a 5 second break.               *
```

```
C*********************************************************************
C     The next IF statement checks to see if the information for a   *
C     new test has been stored in BUFF(1-6) by program ALTER.         *
C*********************************************************************
      K = 0
      IF(BUFF(1) .EQ. 0) GO TO 20
C                               *  Sorry no test, start checking the  *
C                               *   chambers again at label 20        *
      K=BUFF(2)
C                               *  DB reference number                *
      MODE2(K) = BUFF(3)
C                               *  WB or ALT reference number         *
      FILEN(K,1)=BUFF(4)
      FILEN(K,2)=BUFF(5)
      FILEN(K,3)=BUFF(6)
C                               *  Name of file created by BUILD      *
C                               *  containing profile information     *
C*********************************************************************
C     The remainder of the program executes only at the start of a   *
C     test or when it is time to change intervals.                    *
C*********************************************************************
   55 NAME(1)=FILEN(K,1)
      NAME(2)=FILEN(K,2)
      NAME(3)=FILEN(K,3)
C                                * Put filename in NAME for use in     *
C                                * OPENF statement                     *
      IF(BUFF(1).NE.0) GO TO 60
C                                * If the test is now starting, skip   *
C                                * the next IF statement, which checks *
C                                * to see if it's time for the test to *
C                                * end.                                *
      IF(REC(K).GT.ENDREC(K)) THEN
         TOTAL(K)=0
         HOST(K,2) = -100.0
         IF(MODE2(K).NE.0) THEN
            K1 = MODE2(K)
            HOST(K1,2) = -100.0
            MODE2(K) = 0
            HOLD(K) = 0
         ENDIF
         WRITE(6,'(" THE ",3A2," TEST HAS ENDED, TIME: ",I3,,2X,I2,":",
     * I2,":",I2)') FILEN(K,1),FILEN(K,2),FILEN(K,3),CLOCK(5),CLOCK(4),
     * CLOCK(3),CLOCK(2)
         GO TO 20
      ENDIF
C                              *  The test has ended.                  *
   60 CALL OPENF(DCB,IERR,NAME)
      IF(IERR.LT.0) WRITE(6,'( " ERROR ON OPEN: ERROR= ",I6)')IERR
      IF(BUFF(1).EQ.0) GO TO 65
C                              *  If the test is now starting, read the *
C                              *  number of intervals from the file.    *
C                              *  Otherwise, skip down to 65.           *
```

```
      CALL READF(DCB,IERR,REC1,2)
      IF(IERR.LT.0) WRITE(6,'(" READ ERROR(REC1): ERROR= ",I6)')IERR
      ENDREC(K) = 2*REC1(2) + 2
      REC(K) = 2*(BUFF(7)) + 1
      BUFF(1) = 0
   65 NUM = REC(K)
      IR = 1
      CALL POSNT(DCB,IERR,NUM,IR)
      IF(IERR.LT.0) WRITE(6,'( " ERROR ON POSTN: ERROR= ",I6)')IERR
      CALL READF(DCB,IERR,CURRSP,12)
      IF(IERR.LT.0) WRITE(6,'( " READ ERROR(CURSP):ERROR= ",I6)')IERR
      CALL READF(DCB,IERR,ITIME,3)
      IF(IERR.LT.0) WRITE(6,'( " READ ERROR(ITIME):ERROR= ",I6)')IERR
      CALL READF(DCB,IERR,NEXTSP,12)
      IF(IERR.LT.0) WRITE(6,'( " READ ERROR(NETSP):ERROR= ",I6)')IERR
      CALL CLOSE(DCB,IERR)
      IF(IERR.LT.0) WRITE(6,'( " ERROR ON CLOSE: ERROR= ",I6)')IERR
C
      CALL EXEC(11,START)
C
      DO 70 J=1,5
   70 STIME(K,J)=START(J)
      TOTAL(K) = ITIME(1)*3600+ITIME(2)*60+ITIME(3)
      SPDIFF(K) = NEXTSP(1)-CURRSP(1)
      STTEMP(K) = CURRSP(1)
      SPSAVE(K) = CURRSP(1)
      HOST(K,2) = CURRSP(1)
      HIDEV(K) = NEXTSP(2)
      LODEV(K) = NEXTSP(3)
      HOLD(K) = 0
C                                      *  Make initializations for the new    *
C                                      *  interval.                           *
      INTVL = (REC(K) -1)/2
      WRITE(6,'(" FILENAME: ",3A2,"  DB REF #: ",I2,"  INTERVAL: ",I3,
     * "  SP: ",F6.1,"  NEXTSP: ",F6.1,"  STARTING TIME: ",I3,2X,I2,
     * ":",I2,":",I2)')NAME,K,INTVL,CURRSP(1),NEXTSP(1),START(5),
     * START(4),START(3),START(2)
C
      IF(MODE2(K).NE.0) THEN
        K2 = MODE2(K)
        SPDIFF(K2) = NEXTSP(4)-CURRSP(4)
        STTEMP(K2) = CURRSP(4)
        SPSAVE(K2) = CURRSP(4)
        HOST(K2,2) = CURRSP(4)
        HIDEV(K2) = NEXTSP(5)
        LODEV(K2) = NEXTSP(6)
        WRITE(6,'(19X,"MODE2 REF #: ",I2,11X,"SP:",F8.1,"  NEXTSP: ",
     *   F8.1)')K2,CURRSP(4),NEXTSP(4)
      ENDIF
C
      REC(K) = REC(K) + 2
C
      GO TO 20
      END
      PROGRAM PREP
C
C
      INTEGER  DCB(144),LIST(3),IBUF(40),HOST(3),JBUF(40),BLDGN(2)
      INTEGER FILEN(3),SCND(3),THRD(3),FRTH(3),FITH(3),ITIME(5,20,3)
      INTEGER ESC,CUR,CLD,BEL,PINT,A,ALT,HUM,IT(3),NUMCYC(5),NUMINT(5)
      INTEGER INTIME(3)
C
      REAL   TEMP(5,20,6),SETPT(6),INITSP(6)
C
      DATA LIST/'MAIN'/,HOST/'B7290'/,SCND/'B7288'/,THRD/'B7863'/
      DATA FRTH/'B4500'/,FITH/'B8540'/,ALT/'NO'/,HUM/'NO'/
C
      ESC=27
      CUR=72
      CLD=74
      BEL=7
      A=17
C    *****************************************************
C    *       WRITE MAIN MENU TO SCREEN                   *
C    *****************************************************
      WRITE(1,'(5A2)')ESC,CUR,ESC,CLD,BEL
      CALL OPEN (DCB,IERR,LIST)
C
      DO 10  I=1,16
      CALL READF(DCB,IERR,IBUF,40)
C
      WRITE(1,'(40A2)')(IBUF(N),N=1,40)
   10 CONTINUE
      CALL CLOSE (DCB,IERR,0)
C
   11 WRITE(1,'(/10X,"ENTER BUILDING NUMBER _")')
      READ(1,'(I4)') BLDGN(1)
C
            IF(BLDGN.EQ.7290)  CALL PINT(HOST,A,*15)
C
            IF(BLDGN.EQ.7288)  CALL PINT(SCND,A,*15)
C
            IF(BLDGN.EQ.7863)  CALL PINT(THRD,A,*15)
C
            IF(BLDGN.EQ.4500)  CALL PINT(FRTH,A,*15)
C
            IF(BLDGN.EQ.8540)  CALL PINT(FITH,A,*15)
C
      WRITE(1,'(/"INCORRECT: TRY AGAIN")')
      GO TO 11
C*****************************************************************
C                   ENTER CHAMBER SYMBOL WITH DATA
C*****************************************************************
C
   15 WRITE(1,'(/15X,"ENTER CHAMBER SYMBOL WITH JULIAN DATE _")')
      READ(1,'(3A2)')FILEN
C
```

```
            IF(FILEN(1).EQ.2HAR .OR. FILEN(1).EQ.2HSA) THEN
              WRITE(1,'(/15X,"IS THIS AN ALTITUDE TEST _")')
              READ(1,'(A2)') ALT
              END IF
            IF(FILEN(1).EQ.2HFH .OR. FILEN(1).EQ.2HAR .OR. FILEN(1).EQ.
     *        2HSH) THEN
              WRITE(1,'(/15X,"IS THIS A HUMIDITY TEST _")')
              READ(1,'(A2)') HUM
              END IF
C*********************************************************************
C                      ENTER INITIAL SETPOINTS
C*********************************************************************
      WRITE(1,'(/20X,"INITIAL TEMPERATURE _")')
      READ(1,'(F6.1)') INITSP(1)
      INITSP(2) = 100.
      INITSP(3) = -100.
      IF (ALT .EQ. 2HNO) GO TO 20
      WRITE(1,'(1H ,10X,"ALTITUDE _")')
      READ(1,'(F8.1)') INITSP(4)
      INITSP(5) = 100000
      INITSP(6) = -100000
C
   20 IF (HUM .EQ. 2HNO) GO TO 25
        WRITE(1,'(1H ,10X,"HUMIDITY: _")')
        READ(1,'(F6.1)') INITSP(4)
        INITSP(5) = 100.
        INITSP(6) = -100.
C
   25 WRITE(1,'(/11X,"Time allowed to reach initial setpoint:")')
        WRITE(1,'(/11X,"Hours: _")')
        READ(1,'(I2)')INTIME(1)
        WRITE(1,'(/11X,"Minutes: _")')
        READ(1,'(I2)')INTIME(2)
        WRITE(1,'(/11X,"Seconds: _")')
        READ(1,'(I2)')INTIME(3)
        SETPT(1) = INITSP(1)
        SETPT(4) = INITSP(4)
C********************************************************************
C                   BEGIN LOOP,CYCLE,INTERVAL INPUT
C********************************************************************
      WRITE(1,'(/20X,"TOTAL NUMBER OF LOOPS _")')
      READ(1,'(I2)') NUMLPS
C
      DO 80 LOOP = 1,NUMLPS
      WRITE(1,'(/20X,"NUMBER OF CYCLES IN THIS LOOP _")')
      READ(1,'(I2)') NUMCYC(LOOP)
C
   30 WRITE(1,'(/20X,"NUMBER OF INTERVALS FOR THESE CYCLES _")')
      READ(1,'(I2)') NUMINT(LOOP)
C
C
         DO 80 J = 1,NUMINT(LOOP)
C
```

```
   35 WRITE(1,'(5A2)')ESC,CUR,ESC,CLD,BEL
      WRITE(1,'(25X,"BLDG NO. ",I4," CHAMBER ",3A2,)') BLDGN(1),FILEN
      WRITE(1,'(28X,"INTERVAL ",I2," OF LOOP ",I2)') J,LOOP
C
      WRITE(1,'(/11X,"BEGINNING DB TEMP:",F6.1)') SETPT(1)
      WRITE(1,'(/11X,"FINAL DB TEMP: _")')
      READ(1,'(F6.1)') TEMP(LOOP,J,1)
      WRITE(1,'(11X,"High Deviation: _")')
      READ(1,'(F3.1)') TEMP(LOOP,J,2)
      WRITE(1,'(11X,"Low Deviation: _")')
      READ(1,'(F4.1)') TEMP(LOOP,J,3)
      WRITE(1,'(/11X,"*********************"/)')
C
            IF(ALT.EQ.2HNO) GO TO 40
C
       WRITE(1,'(11X,"BEGINNING ALTITUDE: ",F8.1)') SETPT(4)
       WRITE(1,'(/11X,"FINAL ALTITUDE: _")')
       READ(1,'(F8.1)') TEMP(LOOP,J,4)
       WRITE(1,'(11X,"High Deviation: _")')
       READ(1,'(F6.1)') TEMP(LOOP,J,5)
       WRITE(1,'(11X,"Low Deviation: _")')
       READ(1,'(F6.1)') TEMP(LOOP,J,6)
C
   40       IF(HUM.EQ.2HNO) GO TO 45
C
       WRITE(1,'(11X,"BEGINNING WB TEMP: ",F6.1)') SETPT(4)
       WRITE(1,'(/11X,"FINAL WB TEMP: _")')
       READ(1,'(F6.1)') TEMP(LOOP,J,4)
       WRITE(1,'(11X,"High Deviation: _")')
       READ(1,'(F3.1)') TEMP(LOOP,J,5)
       WRITE(1,'(11X,"Low Deviation: _")')
       READ(1,'(F4.1)') TEMP(LOOP,J,6)
C
C
   45 WRITE(1,'(/11X,"Enter time for interval: Hours,Minutes,Seconds")')
      WRITE(1,'(/11X,"Hours: _")')
      READ(1,'(I2)') ITIME(LOOP,J,1)
      WRITE(1,'(11X,"Minutes: _")')
      READ(1,'(I2)') ITIME(LOOP,J,2)
      WRITE(1,'(11X,"Seconds: _")')
      READ(1,'(I2)') ITIME(LOOP,J,3)
C     ************************************************
C     *                  VERIFY                      *
C     ************************************************
C
      WRITE(1,'(" INT     TEMP    HD    LD    *    HUM       HD       LD *
     *  TIME")')
      WRITE(1,'(/2X,I2,4X,F5.1,2X,F3.1,2X,F4.1,1X,F8.1,2X,F6.1,2X,F6.1
     * ,7X,I2,X,I2,X,I2)') J,TEMP(LOOP,J,1),TEMP(LOOP,J,2),
     * TEMP(LOOP,J,3),TEMP(LOOP,J,4),TEMP(LOOP,J,5),TEMP(LOOP,J,6),
     * ITIME(LOOP,J,1),ITIME(LOOP,J,2),ITIME(LOOP,J,3)
C
```

```
      WRITE(1,'(//"Is the above data correct?")')
      WRITE(1,'(/"Enter GO for correct data or NO for incorrect data.
     *_")')
      READ(1,'(A2)') ICOR
      IF(ICOR.EQ.2HNO) GO TO 35
      DO 50 K=1,6
   50 SETPT(K) = TEMP(LOOP,J,K)
C
   80 CONTINUE
C*********************************************************************
C           Count intervals and store in BLDGN(2)
C*********************************************************************
      INTCNT = 1
      DO 85 LOOP = 1,NUMLPS
      DO 85 I=1,NUMCYC(LOOP)
      DO 85 J=1,NUMINT(LOOP)
   85 INTCNT = INTCNT + 1
      BLDGN(2) = INTCNT
      ISIZE = ((INTCNT*20 + 15)/128)+1
C                                    *  Compute # of blocks, each interval*
C                                    *  uses 20 words, (15 + 4) +1 extra   *
      IF(ISIZE.LT.0) THEN
      WRITE(6,'(" The file is to large, and will not be created.")')
      GO TO 99
      ENDIF
C*********************************************************************
C          Create the file and write the test profile to it.
C*********************************************************************
      CALL CREAT(DCB,IERR,FILEN,ISIZE,4)
         IF(IERR.LT.0) GO TO 150
      CALL WRITF(DCB,IERR,BLDGN,2)
         IF(IERR.LT.0) GO TO 160
      CALL WRITF(DCB,IERR,FILEN,3)
         IF(IERR.LT.0) GO TO 160
C
      CALL WRITF(DCB,IERR,INITSP,12)
         IF(IERR.LT.0) GO TO 160
      CALL WRITF(DCB,IERR,INTIME,3)
         IF(IERR.LT.0) GO TO 160
      CALL WRITF(DCB,IERR,INITSP,12)
         IF(IERR.LT.0) GO TO 160
      DO 95 LOOP= 1,NUMLPS
        DO 95 I=1,NUMCYC(LOOP)
        DO 95 J=1,NUMINT(LOOP)
          DO 90 K=1,6
   90       SETPT(K)=TEMP(LOOP,J,K)
          DO 91 K=1,3
   91       IT(K)=ITIME(LOOP,J,K)
          CALL WRITF(DCB,IERR,IT,3)
           IF(IERR.LT.0) GO TO 160
          CALL WRITF(DCB,IERR,SETPT,12)
           IF(IERR.LT.0) GO TO 160
   95   CONTINUE
```

```
C
C
      CALL CLOSE(DCB,IERR,0)
      IF (IERR.LT.0) GO TO 170
      GO TO 99
C
  100 WRITE(1,'("FMP ERROR ON OPEN-MAIN= ",I5)')IERR
  101 WRITE(1,'("FMP ERROR ON READF-MAIN= ",I5)')IERR
  110 WRITE(1,'("FMP ERROR ON OPEN= ",I5,"  ON BLDG ",3A2)')IERR,BLDGN
  120 WRITE(1,'("FMP ERROR ON READF=",I5," ON BLDG ",3A2)')IERR,BLDGN
  130 WRITE(1,'("FMP ERROR ON CLOSE= ",I5," ON BLDG ",3A2)')IERR,BLDGN
  140 WRITE(1,'("FMP ERROR ON OPEN= ",I5," ON FILE ",3A2)')IERR,FILEN
  150 WRITE(1,'("FMP ERROR ON CREAT=",I5," ON FILE ",3A2)')IERR,FILEN
  160 WRITE(1,'("FMP ERROR ON WRITF ",I5," ON FILE ",3A2)')IERR,FILEN
  170 WRITE(1,'("FMP ERROR ON CLOSE ",I5," ON FILE ",3A2)')IERR,FILEN
C
  99  STOP
      END
C
      SUBROUTINE PINT(BLDG,J,*)
C
C   * THIS ROUTINE WRITES A MENU FOR A SPECIFIED BUILDING *
      INTEGER DCB(144),JBUF(40),BLDG(3),J,ESC,CUR,CLD,BEL
C
      ESC=27
      CUR=72
      CLD=74
      BEL=7
C
      WRITE(1,'(5A2)') ESC,CUR,ESC,CLD,BEL
C
      CALL OPEN(DCB,IERR,BLDG)
      DO 10 I=1,J
      CALL READF(DCB,IERR,JBUF)
      CALL EXEC(2,1,JBUF,40)
   10 CONTINUE
      CALL CLOSE(DCB,IERR,0)
      RETURN 1
      END
      PROGRAM ALTER
C
C
      INTEGER ESC,CUR,CLD,BUFF
      COMMON HOST(14,8),BUFF(10)
C
      ESC=27
      CUR=72
      CLD=74
C
   10 WRITE(1,'(4A2)') ESC,CUR,ESC,CLD
      WRITE(1,'("Do you want to change a value in the HOST parameter tab
     *le? (Y) (N) _")')
      READ(1,'(A2)') L
      IF (L.EQ.2HN ) GO TO 20
   15 WRITE(1,'("Enter row number of HOST to be altered (1-14) _")')
      READ(1,'(I2)') K
      WRITE(1,'("Enter column number of HOST to be altered (1-8) _")')
      READ(1,'(I2)') L
      WRITE(1,'(//"The current value of that location is ",F8.2)')
     * HOST(K,L)
      WRITE(1,'(/"* Do you want to change this value? (Y) (N) * _")')
      READ(1,'(A2)') IASK
      IF(IASK.EQ.2HY ) THEN
         WRITE(1,'(//"Enter new value now _")')
         READ(1,'(F8.2)') HOST(K,L)
      ENDIF
      WRITE(1,'("Any more changes ? (Y) (N) _")')
      READ(1,'(A2)') IMORE
      IF(IMORE.EQ.2HY ) GO TO 15
      STOP
C                                        *  The above statements apply only  *
C                                        *  if a parameter in HOST is to be  *
C                                        *  changed.                         *
   20 WRITE(1,'(/"Do you want to start a test? ( WARNING! You need a fil
     *e to start a test) (Y) (N) _")')
      READ(1,'(A2)') L
      IF(L.NE.2HY ) STOP
      WRITE(1,'(/"ENTER TWO NUMBERS:"/"the first number is the dry bulb
     *reference number;"/"the second number is either the wet bulb or al
     *titude reference number"/"or (0) for dry bulb only. ")')
      WRITE(1,'(/"Enter DB reference number (1-10) _")')
      READ(1,'(I2)') BUFF(2)
      WRITE(1,'(/"Enter WB or ALT reference number (0) or (11-14)_")')
      READ(1,'(I2)') BUFF(3)
      WRITE(1,'(/"Enter six character name of file containing profile fo
     *r this test _")')
      READ(1,'(3A2)') BUFF(4),BUFF(5),BUFF(6)
      WRITE(1,'(3A2)')BUFF(4),BUFF(5),BUFF(6)
      WRITE(1,'(/"Enter the interval number at which the test is to begi
     *n (normally 1) _")')
```

```
      READ(1,'(I3)') BUFF(7)
      WRITE(1,'(/"Do you really want to start this test now? (Y) (N) _"
     * )')
      READ(1,'(A2)')IASK
      IF(IASK.NE.2HY ) STOP
      BUFF(1)=1
C                                          *  The above statements apply only  *
C                                          *  if a test is to be started. As   *
C                                          *  you will notice, the rows in HOST*
C                                          *  that apply to the chamber (to be *
C                                          *  started) will be needed along    *
C                                          *  with the file name created by BLD2.
      STOP
      END
      PROGRAM DISPL
C
C
      INTEGER DASH(3),ESC,CUR,CLD,FORM(3),IBUF(50),IDCB(144),BLANK(3)
      INTEGER AND,AAA,CNO,CEE,RNO,WYE,CNB,CNC,TBUF(15),R,C,F
      REAL HOST,ARRAY(9,6)
      COMMON HOST(14,8)
      DATA DASH/'------'/,FORM/'DP'/,CNO/'23'/,RNO/'7'/,BLANK/'      '/
C
      ESC = 27
      CUR = 72
      CLD = 74
      AND = 38
      AAA = 97
      WYE = 89
      CEE = 99
      CCC = 67
C
      DO 10 R=1,9
      DO 10 C=1,6
   10 ARRAY(R,C) = -100.0
C
      WRITE(1,'(4A2)') ESC,CUR,ESC,CLD
C     ********************************************************
C     *                 WRITE FORM (DP) TO THE SCREEN          *
C     ********************************************************
          CALL OPENF(IDCB,IERR,FORM)
            DO 20 F = 1,15
              CALL READF(IDCB,IERR,IBUF,50)
              WRITE(1,'(50A2)')(IBUF(N),N = 1,50)
   20       CONTINUE
              DO 22 F = 1,4
               CALL READF(IDCB,IERR,IBUF,40)
               WRITE(1,'(40A2)')(IBUF(N),N = 1,40)
   22         CONTINUE
          CALL CLOSE(IDCB,IERR,0)
C     ********************************************************
   99   WRITE(1,'(2A2)') ESC,CUR
C                                               *  HOME CURSOR            *
        ARRAY(1,1)=HOST(1,1)
        ARRAY(1,2)=HOST(1,2)
        ARRAY(2,1)=HOST(8,1)
        ARRAY(2,2)=HOST(8,2)
        ARRAY(3,1)=HOST(4,1)
        ARRAY(3,2)=HOST(4,2)
        ARRAY(3,3)=HOST(3,1)
        ARRAY(3,4)=HOST(3,2)
        ARRAY(4,1)=HOST(2,1)
        ARRAY(4,2)=HOST(2,2)
        ARRAY(4,3)=HOST(11,1)
        ARRAY(4,4)=HOST(11,2)
        ARRAY(4,5)=HOST(14,1)
        ARRAY(4,6)=HOST(14,2)
        ARRAY(5,1)=HOST(5,1)
        ARRAY(5,2)=HOST(5,2)
        ARRAY(6,1)=HOST(6,1)
        ARRAY(6,2)=HOST(6,2)
        ARRAY(7,1)=HOST(7,1)
        ARRAY(7,2)=HOST(7,2)
        ARRAY(8,1)=HOST(9,1)
        ARRAY(8,2)=HOST(9,2)
        ARRAY(8,3)=HOST(12,1)
        ARRAY(8,4)=HOST(12,2)
        ARRAY(9,1)=HOST(10,1)
        ARRAY(9,2)=HOST(10,2)
        ARRAY(9,5)=HOST(13,1)
        ARRAY(9,6)=HOST(13,2)
C  ****************************************************
C  *     WRITE MEASUREMENTS FROM COMMON TO THE SCREEN    *
C  ****************************************************
        IF(ARRAY(1,2).NE.-100.) THEN
          WRITE(1,'(7A2,F5.1,2X,F5.1)') ESC,AND,AAA,CNO,CEE,RNO,WYE,
     *    ARRAY(1,1),ARRAY(1,2)
          ELSE
          WRITE(1,'(7A2,3A2,2X,3A2)') ESC,AND,AAA,CNO,CEE,RNO,WYE,DASH,
     *    BLANK
          ENDIF
      DO 30 I = 2,9
C                                        * CHECK 1st ELEMENT IN EACH      *
C                                        * ROW. IF THERES NOTHING THERE   *
C                                        * THE CHAMBER IS IDLE,SO WRITE   *
C                                        * DASHES IN EACH COL IN THE ROW *
        IF(ARRAY(I,2).NE.-100) THEN
C
         IF(ARRAY(I,4).NE.-100) THEN
C                                        * SINCE THERES A VALID SP        *
C                                        * IN THE FIRST COL,CKECK THE     *
C                                        * THIRD COL (HUMIDITY).          *
       WRITE(1,'(5A2,F5.1,2X,F5.1,3X,F5.1,2X,F5.1,3X,3A2,2X,3A2)')ESC,
     * AND,AAA,CNO,CCC,ARRAY(I,1),ARRAY(I,2),ARRAY(I,3),ARRAY(I,4),
     * DASH,BLANK
         GO TO 30
         ENDIF
C
         IF(ARRAY(I,6).NE.-100) THEN
          WRITE(1,'(5A2,F5.1,2X,F5.1,3X,3A2,9X,F6.1,1X,F6.1)') ESC,
     * AND,AAA,CNO,CCC,ARRAY(I,1),ARRAY(I,2),DASH,ARRAY(I,5),ARRAY(I,6)
         GO TO 30
         ENDIF
C
         WRITE(1,'(5A2,F5.1,2X,F5.1,3X,3A2,3X,3A2)') ESC,AND,AAA,CNO,
     * CCC,ARRAY(I,1),ARRAY(I,2)
C
       ELSE
         WRITE(1,'(5A2,3A2,9X,3A2,9X,3A2)')ESC,AND,AAA,CNO,CCC,
     * DASH,DASH,DASH
C
       ENDIF
```

```
C
   30 CONTINUE
C  ********************************************************
C  *                   GET TIME AND PRINT                 *
C  ********************************************************
      CALL FTIME(TBUF)
C
      WRITE(1,'(/5A2,15A2)') ESC,AND,AAA,CNO,CCC,TBUF
C  ********************************************************
C  *                    CHECK BREAK BIT                   *
C  ********************************************************
       IF(IFBRK(IDUM)) 100,99
C        **********************
100  DO 110 K=1,14
110     IF(HOST(K,7).EQ.2. .OR. HOST(K,7).EQ.3.) HOST(K,7)=4.
      STOP
      END


TN4X,L
      PROGRAM CHECK
      INTEGER DCB(144),FILEN(3),TIME(3),REC1(2),REC2(3)
      REAL SETPT(6)
C
  10  IDLU = 1
      WRITE(1,'("Filename _")')
      READ(1,'(3A2)') FILEN
C
      WRITE(1,'(/"Do you want a printout? (Y) (N) _")')
      READ(1,'(A2)')IASK
      IF(IASK.EQ.2HY ) IDLU = 6
      CALL OPEN(DCB,IERR,FILEN)
C
      CALL READF(DCB,IERR,REC1)
      WRITE(IDLU,'(/" BUILDING NUMBER: ",I4,5X,"NUMBER OF INTERVALS: ",
     * I4)') REC1(1),RFC1(2)
C
      CALL READF(DCB,IERR,REC2)
      WRITE(IDLU,'(" FILENAME: ",3A2)') REC2
      WRITE(IDLU,'(//" INTERVAL* DBSP   HIDEV   LODEV  *   M2SP     HIDE
     *V      LODEV"/)')
C
            DO 20 K=1,REC1(2)
      CALL READF(DCB,IERR,SETPT)
      WRITE(IDLU,'( 3X,I3,3X,F6.1,2X,F6.1,2X,F6.1,2X,F8.1,2X,F8.1,2X,
     * F9.1)')K,SETPT
C
      CALL READF(DCB,IERR,TIME)
      WRITE(IDLU,'( 10X,I2,":",I2,":",I2)') TIME
  20  CONTINUE
C
```

```
      CALL READF(DCB,IERR,SETPT)
      WRITE(IDLU,'( 9X,F6.1,2X,F6.1,2X,F6.1,2X,F8.1,2X,F8.1,2X,F9.1)')
     * SETPT
      CALL CLOSE(DCB,IERR,0)
      WRITE(1,'(/"ANY MORE? (Y) (N) _")')
      READ(1,'(A2)') IMORE
      IF(IMORE.EQ.2HY ) GO TO 10
      STOP
      END
      PROGRAM TABLE
C
C
      INTEGER M(14),ESC,CUR,CLD,BEL
      COMMON HOST(14,8)
      ESC = 27
      CUR = 72
      CLD = 74
      BEL = 7
    2 WRITE(1,'(5A2)')ESC,CUR,ESC,CLD,BEL
      WRITE(1,'(//"       MEAS       SP       GAIN    RESET    RATE    PRINT
     * STATUS    BIAS"/)')
      DO 5 J=1,14
    5 M(J)=J
      WRITE(1,10) (M(J),(HOST(J,K),K=1,8),J=1,14)
   10 FORMAT(I2,2X,F7.2,2X,F7.2,2X,F6.2,2X,F6.2,2X,F6.2,3X,F4.2,3X,F4.2,
     * 2X,F9.1)
      WRITE(1,'(/"RUN AGAIN? (Y) (N) _")')
      READ(1,'(A2)') IASK
      IF(IASK.EQ.2HY ) GO TO 2
      STOP
      END
```

```
* PROGRAM MAC
*
SET ECHO OFF
CLEAR
*
* here we go !!!
*
RESET!
NTASK(3)!
*                               Dimension 20 Variables,2 Buffers
DIM(40,2,400,22)!
*****************************************************************
*** Main Task To Set Up The Ranges For The ADC To Do TTEMPS From ***
*****************************************************************
*
RANGE(2,16,1) 1000
     (2,1,12) 100,100,100,100,100,100,100,100,100,100,100,100
CLB(1)
CLB(2)
AON(1)!
*****************************************************************
****** Task To Get And Send The Thermocouple Data To The Host ******
*****************************************************************
TASK(1)
*                               Task To Get Temp From Vib Cell 1
IN(B1)
REF(2,16)
*CLB(1)
CLB(2)
CPACE(0,0,083)
WPACE BLOCK TTEMP(2,1,200)
AAVERAGE(B1,200,B2(1))
REWIND(B1)
WPACE BLOCK TTEMP(2,2,200)
AAVERAGE(B1,200,B2(2))
REWIND(B1)
WPACE BLOCK TTEMP(2,3,200)
AAVERAGE(B1,200,B2(3))
REWIND(B1)
WPACE BLOCK TTEMP(2,4,200)
AAVERAGE(B1,200,B2(4))
REWIND(B1)
WPACE BLOCK TTEMP(2,5;200)
AAVERAGE(B1,200,B2(5))
REWIND(B1)
WPACE BLOCK TTEMP(2,6,200)
AAVERAGE(B1,200,B2(6))
REWIND(B1)
WPACE BLOCK TTEMP(2,7,200)
AAVERAGE(B1,200,B2(7))
REWIND(B1)
WPACE BLOCK TTEMP(2,8,200)
AAVERAGE(B1,200,B2(8))
```

```
REWIND(B1)
WPACE BLOCK TTEMP(2,9,200)
AAVERAGE(B1,200,B2(9))
REWIND(B1)
WPACE BLOCK TTEMP(2,10,200)
AAVERAGE(B1,200,B2(10))
REWIND(B1)
WPACE BLOCK TTEMP(2,11,200)
AAVERAGE(B1,200,B2(11))
REWIND(B1)
WPACE BLOCK TTEMP(2,12,200)
AAVERAGE(B1,200,B2(12))
REWIND(B1)
REWIND(B2)
SKIP(B2,12)
*                              Move The Pointer Back To 1st Element
REL(B2,A)!
*                              Release B2 To Port A ; End Task 1
*       *****************************************************
*       * Task To Do The Control Output (4-20ma) To The Valve *
*       *****************************************************
TASK(2)
CPACE(0,0,168)
*                              Task To Output CO To All Cells
OUT(V10)
WPACE ;CO(9,1,12) !
*
TASK(3,32767)
REPEAT(0)
PAUSE
NEXT!
* Thats All Folks !!!
QUIT
```

DISTIRBUTION LIST

	No of Copies
US Army Materiel Systems Analysis Activity ATTN: DRXSY-MP Aberdeen Proving Ground, Maryland 21005	1
DRSMI-R	1
DRSMI-RPR	15
DRSMI-RPT	1
DRSMI-LP, Mr. Voigt	1
DRSMI-RTC, Mr. Sullivan	4
Department of Math University of Alabama in Huntsville Huntsville, AL 35899	5
Frances E. Dean, PhD Battlelle Research Triangle Park Office 200 Park Drive P. O. Box 12297 Research Triangle Park, NC 27709	1

DATE
ILMED
8