NASA Technical Memorandum 101988

Laboratory Process Control Using Natural Language Commands From a Personal Computer

Herbert A. Will and Michael A. Mackin
Lewis Research Center
Cleveland, Ohio

(NASA-TM-101988) LABORATORY PROCESS CONTROL USING NATURAL LANGUAGE COMMANDS FROM A PERSONAL COMPUTER (NASA. Lewis Research Center) 14 p CSCL 09B

N89-24055

Unclas
G3/61 0204415

April 1989

E-4690

LABORATORY PROCESS CONTROL USING NATURAL LANGUAGE COMMANDS FROM A PERSONAL COMPUTER

Herbert A. Will and Michael A. Mackin
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

PC software is described which provides flexible natural language process control capability with an IBM PC or compatible machine. Hardware requirements include the PC, and suitable hardware interfaces to all controlled devices. Software required includes the MS-DOS operating system, a PC-based FORTRAN-77 compiler, and user-written device drivers. Instructions for use of the software are given as well as a description of an application of the system.

INTRODUCTION

Process control flexibility is extremely important in the typical complex research laboratory environment. Process control schedules require changes frequently, often several times per day. These changes may include adding, deleting, and rearranging steps in a process. Unattended process control is also frequently required when process duration exceeds 24 hr. There is frequently the additional requirement that the system be usable by technicians and others with limited programming skills.

A software system for an IBM PC or compatible has been developed to satisfy the requirements mentioned above. Once set up, this system requires only an input file containing natural language command lines telling the system what to do and when to do it. Set up includes writing device driver routines for all controlled devices.

The following text includes an overview of the software system, an instruction section on system setup and operation, and a discussion of one application at the NASA Lewis Research Center.

SOFTWARE SYSTEM OVERVIEW

The software system includes three programs. Two of the programs, SETUP and ACDA, are written in FORTRAN-77. These programs are used to take data and control a research process. The third program is written in Pascal and must be used to generate subroutines used by the other two FORTRAN programs. (These programs can be obtained from COSMIC, The University of Georgia, Athens, GA 30602, they are identified as LEW-14907.) A fourth element in the software is the input data set SGxxxx.DAT generated by the user and contains the natural language commands that are to be executed by the computer.

SETUP, the first of the two FORTRAN programs, is used to parse the input data set (see flow diagram in fig. 1). SETUP scans the input data for syntax

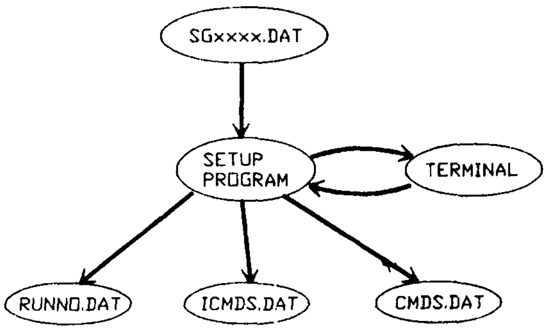


FIGURE 1. - DATA FLOW DIAGRAM FOR SETUP PROGRAM.

errors and generates three intermediate data sets ICMDS.DAT, CMDS.DAT, and RUNNO.DAT. These three data sets are used as input to the main time-sequencing program ACDA (see flow diagram in fig. 2). RUNNO.DAT contains the run number information; ICMDS.DAT contains the initialization commands; and CMDS.DAT contains the time-sequencing commands.

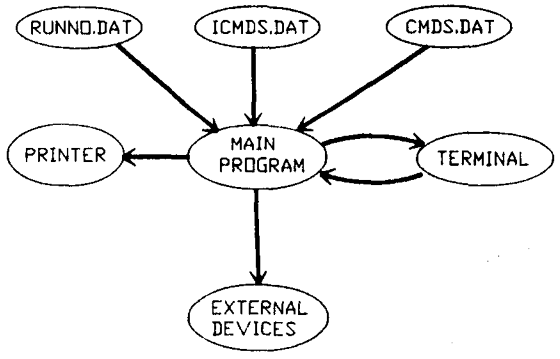


FIGURE 2. - DATA FLOW DIAGRAM FOR ACDA PROGRAM.

ACDA, the main time-sequencing program executes each command at the correct time using the intermediate data sets RUNNO.DAT, ICMDS.DAT, and CMDS.DAT as input. The program communicates with external equipment using custom subroutines which must be written by the user and linked with the ACDA program. The ACDA program contains two parts; a section for initializing the external equipment, and a section for operating the external equipment at specific times of the day. The maximum process duration is 100 hr.

The ACDA program prints the commands, as they are executed, on the printer for a permanent record while the video display shows the last twenty commands with the last command executed displayed in "reverse video." Also displayed is the time until the next command. Data can also be printed from the ACDA program.

The commands that are recognized by the SETUP program, and executed by the ACDA program, must be defined by the user. The definitions are contained in a subroutine, PCMD, and used by SETUP to recognize the input data commands and check for errors.

Since the user must write routines to control and acquire data from external equipment, SETUP and ACDA must know the names of these external routines and the number and types of parameters passed when they are executed. The subroutine PERFRM contains information about the external routine names and parameters passed and does the job of calling the subroutines which interface with external equipment. As many as six integers and six real numbers may be passed to the external equipment subroutines. A convenience program, GENFOR, is included as part of the system for the purpose of generating the PCMD and PERFRM subroutines. The program GENFOR is written in Pascal.

SYSTEM SETUP

External Drivers

The first step in the setup process is to determine all the control and data acquisition tasks required and to write drivers or subroutines to perform these activities.

The external drivers can be any subroutine which conforms to the subroutine calling convention of the FORTRAN-77 compiler used. The external drivers are called from the PERFRM subroutine using the following type of statement:

```
CALL NAME(parameters).
```

"NAME" is the name of the subroutine and "parameters" are the parameters defined in the data set COMMANDS.DAT. Data can be written to the line printer within the driver routines using FORTRAN logical unit number nine. A typical printer command would be:

```
      WRITE (9,2001)
2001  FORMAT(1X,'THIS IS A TEST OF THE PRINTER').
```

The computer programs SETUP and ACDA were written to execute commands at a specific time.

Description of the WAIT Command

There are cases where the external driver subroutines must wait for something to happen, such as a furnace heating up. Since the wait time may not be known, the user may want the program to recalculate the scheduled execution time of the remaining commands. In order to do this the external subroutine must contain the following FORTRAN commands:

```
LOGICAL*2 SCRFLAG
COMMON /SCR/SCRFLAG
SCRFLAG = .TRUE.
```

The command

```
SCRFLAG = .TRUE.
```

is the command that causes the recalculation of the scheduled execution time. This flag is then automatically reset to FALSE after the execution time has been recalculated.

Defining Commands

The first step in defining your own commands that will be recognized by the SETUP and ACDA programs is to decide which external subroutines will be called by these commands. Then you will have to decide on the form of the commands. A description of how to write these commands follows.

The program GENFOR takes as input a set of command language definitions and generates the two FORTRAN subroutines (PCMD and PERFRM). GENFOR expects the definitions to be stored in a data set with the name COMMANDS.DAT. An example of this data set is shown in figure 3.

```
INIT TEMPSCAN {ITSCAN}
INIT IEEE {IIEEE}
INIT VISHAY {IVSHAY}
INIT OVEN INTEGER REAL REAL {IOVEN}
INIT TABLE {ITABLE}
INIT DCVOLTMETER INTEGER {IDCMTR}
BELL {BELCPL}
INIT OHM4METER INTEGER {IOHMTR}
READ MULTIMETER INTEGER {RMMETER}
READ TEMPS {RTEMP}
WAIT CTEMP INTEGER REAL
WAIT HTEMP INTEGER REAL
TEMP CHANNELS INTEGER INTEGER {TCHANLS}
SET OVEN INTEGER REAL {SOVEN}
MOVE TABLEIN REAL
MOVE TABLECM REAL
READ VISHAY REAL
VISHAY BRIDGE INTEGER
CLAMP TABLE  {CTABLE}
UNCLAMP TABLE {UTABLE}
```

FIGURE 3. - LIST OF TYPICAL COMMANDS USED IN THE DATA SET "COMMANDS.DAT."

The command definition format takes the following form:

<ELEMENT><sub-element><parameters>.

For example the fourth command in figure 3 is:

INIT OVEN INTEGER REAL REAL {IOVEN}.

In this example "INIT" is the element and "OVEN" is the sub-element. This command initializes an oven. One integer and two real parameters are passed to the subroutine. The name "IOVEN" contained within the {} brackets is the name of the external subroutine that will be called by this command. If the name within the {} brackets is not included then the GENFOR program will use the "sub-element" as the name of the external subroutine. If the sub-element is missing then the "ELEMENT" is used. Note that if a "sub-element" is defined for a particular command (ELEMENT) then a second command using this same

ELEMENT must have a different (nonzero) sub-element. For example the command "INIT" with no sub-element would not be allowed. The "INTEGER REAL REAL" specifies the number and type of parameters to be used with the command.

Once the command definitions are stored in the data set COMMANDS.DAT they can be used as input for program GENFOR. This program is executed simply by typing in "GENFOR." The program will ask which drive the data set COMMANDS.DAT is located (A,B,...). If the data set is on drive "A" you would reply "A:".

The GENFOR program then generates two data sets (PCMDX.FOR and PERFRMX.FOR) on the same drive that contains the data set COMMANDS.DAT.

The data sets PCMDX.FOR and PERFRMX.FOR should be renamed PCMD.FOR and PERFRM.FOR. Old copies of PCMD.FOR and PERFRM.FOR may be renamed or deleted.

The subroutine PCMD.FOR is used to parse the input commands and check for typing errors. An example of the subroutine PCMD.FOR is shown in figure 4. The subroutine PERFRM.FOR is used to call subroutines that interface to external instruments. Note that it is up to the user to write these external subroutines. An example of the subroutine PERFRM.FOR is shown in figure 5.

At this point it is assumed that the external subroutines have been written and compiled by the user. The source code for this system was written for Microsoft FORTRAN-77 version 4.0. If a different FORTRAN compiler is used a few modifications in the source code may be necessary.

The source code SETUP.FOR and PCMD.FOR should be compiled and linked as SETUP.EXE. Then the source code ACDA.FOR, PERFRM.FOR, TIMDAT.ASM, USCROL.ASM, and "external subroutines" must be compiled and linked as ACDA.EXE. These two ".EXE" programs will be the programs used to run your equipment. Note, the ".ASM" programs are written in assembly language and the compiled code is ".OBJ."

SYSTEM APPLICATION

When the setup activity is complete the data set SGxxxx.DAT must be created to define the process events and their timing. This data set contains timing information and a series of the previously defined commands that will control the external equipment.

Process Control Data Set

The process control data set SGxxxx.DAT contains two sections; an initialization section, and a time sequencing section. An example of a typical data set is shown in figure 6. The initialization section extends from the beginning of the data set to the first "END" command. The commands must be written in the same format as the commands listed in the data set "COMMANDS.DAT."

```
      SUBROUTINE PCMD(DONE,CHAIN)

      INTEGER*2 RPTR,WPTR
      CHARACTER INREC
      DIMENSION INREC(81)
      CHARACTER OUTREC
      DIMENSION OUTREC(81)
      CHARACTER SYMBOL
      DIMENSION SYMBOL(81)
      CHARACTER COMMD*81
      LOGICAL*2 CHAIN2
      COMMON /DSTR1/RPTR
      COMMON /DSTR2/WPTR
      COMMON /BLK1/INREC,OUTREC
      COMMON /BLK2/SYMBOL
      COMMON /CHN/CHAIN2
      LOGICAL*2 DONE,FLAG,CHAIN

      CHAIN = .FALSE.
      CALL DTRMN(FLAG)
      COMMD = 'END$'
      CALL COMPR(FLAG,COMMD)
      IF (FLAG) GO TO 899

      COMMD = 'CHAIN$'
      CALL COMPR(FLAG,COMMD)
      IF (FLAG) GO TO 890

100   COMMD = 'INIT$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 101
      CALL DTRMN(FLAG)
      COMMD = 'TEMPSCAN$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 102
      COMMD = '1'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

102   COMMD = 'IEEE$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 103
      COMMD = '2'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

103   COMMD = 'VISHAY$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 104
      COMMD = '3'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

104   COMMD = 'OVEN$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 105
      COMMD = '4'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = ','
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = ','
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

105   COMMD = 'TABLE$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 106
      COMMD = '5'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

106   COMMD = 'DCVOLTMETER$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 107
      COMMD = '6'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

107   COMMD = 'OHM4METER$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 108
      COMMD = '7'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

108   STOP 'ERROR ON INIT COMMAND'

101   COMMD = 'BELL$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 109
      COMMD = '8'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

109   COMMD = 'READ$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 110
      CALL DTRMN(FLAG)
      COMMD = 'MULTIMETER$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 111
      COMMD = '9'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

111   COMMD = 'TEMPS$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 112
      COMMD = '10'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

112   COMMD = 'VISHAY$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 113
      COMMD = '11'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

113   STOP 'ERROR ON READ COMMAND'

110   COMMD = 'WAIT$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 114
      CALL DTRMN(FLAG)
      COMMD = 'CTEMP$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 115
      COMMD = '12'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = ','
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

115   COMMD = 'HTEMP$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 116
      COMMD = '13'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = ','
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

116   STOP 'ERROR ON WAIT COMMAND'

114   COMMD = 'TEMP$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 117
      CALL DTRMN(FLAG)
      COMMD = 'CHANNELS$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 118
      COMMD = '14'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = ','
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

118   STOP 'ERROR ON TEMP COMMAND'

117   COMMD = 'SET$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 119
      CALL DTRMN(FLAG)
      COMMD = 'OVEN$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 120
      COMMD = '15'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = ','
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

120   STOP 'ERROR ON SET COMMAND'

119   COMMD = 'MOVE$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 121
      CALL DTRMN(FLAG)
      COMMD = 'TABLEIN$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 122
      COMMD = '16'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

122   COMMD = 'TABLECM$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 123
      COMMD = '17'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUM2
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

123   STOP 'ERROR ON MOVE COMMAND'

121   COMMD = 'VISHAY$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 124
      CALL DTRMN(FLAG)
      COMMD = 'BRIDGE$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 125
      COMMD = '18'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      CALL DTRMN(FLAG)
      CALL PNUMB
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

125   STOP 'ERROR ON VISHAY COMMAND'

124   COMMD = 'CLAMP$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 126
      CALL DTRMN(FLAG)
      COMMD = 'TABLE$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 127
      COMMD = '19'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

127   STOP 'ERROR ON CLAMP COMMAND'

126   COMMD = 'UNCLAMP$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 128
      CALL DTRMN(FLAG)
      COMMD = 'TABLE$'
      CALL COMPR(FLAG,COMMD)
      IF (.NOT.(FLAG)) GO TO 129
      COMMD = '20'
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = ',('
      CALL WQUEUE(OUTREC,COMMD)
      COMMD = '),'
      CALL WQUEUE(OUTREC,COMMD)
      GO TO 900

129   STOP 'ERROR ON UNCLAMP COMMAND'

128   STOP 'NO SUCH ELEMENT'

890   COMMD = '101,(),'
      CALL WQUEUE(OUTREC,COMMD)
      CHAIN = .TRUE.
      CHAIN2 = .TRUE.
      GO TO 900
899   DONE = .TRUE.
900   RETURN
      END
```

FIGURE 4. - EXAMPLE OF SUBROUTINE "PCMD.FOR" THAT WAS GENERATED BY THE PROGRAM "GENFOR.COM".

```
      SUBROUTINE PERFRM(I)

      INTEGER*2 TIME(100,4),TIMOLD(100,4)
      INTEGER*2 EVENT(100),IPARAM(100,6)
      REAL*4 RPARAM(100,6)
      INTEGER*2 TXLEN(100),J
      INTEGER*2 K1,K2,K3,K4,K5,K6
      REAL*4 R1,R2,R3,R4,R5,R6
      COMMON /DSTR1/TIME,TIMOLD,EVENT,IPARAM,RPARAM,TXLEN

      K1=IPARAM(I,1)
      K2=IPARAM(I,2)
      K3=IPARAM(I,3)
      K4=IPARAM(I,4)
      K5=IPARAM(I,5)
      K6=IPARAM(I,6)
      R1=RPARAM(I,1)
      R2=RPARAM(I,2)
      R3=RPARAM(I,3)
      R4=RPARAM(I,4)
      R5=RPARAM(I,5)
      R6=RPARAM(I,6)
      J=EVENT(I)
      GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
     & 16,17,18,19,20)J
      GOTO 900

 1    CALL ITSCAN
      GOTO 900
 2    CALL IIEEE
      GOTO 900
 3    CALL IVSHAY
      GOTO 900
 4    CALL IOVEN(K1,R1,R2)
      GOTO 900
 5    CALL ITABLE
      GOTO 900
 6    CALL IDCMTR(K1)
      GOTO 900
 7    CALL IOHMTR(K1)
      GOTO 900
 8    CALL BELCPL
      GOTO 900
 9    CALL RMMETER(K1)
      GOTO 900
10    CALL RTEMP
      GOTO 900
11    CALL VISHAY(R1)
      GOTO 900
12    CALL CTEMP(K1,R1)
      GOTO 900
13    CALL HTEMP(K1,R1)
      GOTO 900
14    CALL TCHANLS(K1,K2)
      GOTO 900
15    CALL SOVEN(K1,R1)
      GOTO 900
16    CALL TABLEIN(R1)
      GOTO 900
17    CALL TABLECM(R1)
      GOTO 900
18    CALL BRIDGE(K1)
      GOTO 900
19    CALL CTABLE
      GOTO 900
20    CALL UTABLE
      GOTO 900

900   RETURN
      END
```

FIGURE 5. - EXAMPLE OF SUBROUTINE "PERFRM.FOR" THAT WAS GENERATED BY THE PROGRAM "GENFOR.COM".

```
! TEST PROGRAM
! MEASURE RESISTANCE AND TEMPERATURE
! INITIALIZATION SECTION
INIT IEEE
INIT TEMPSCAN
TEMP CHANNELS  1 4
INIT OVEN 2  0.008192  -0.07589
INIT OHM4METER  1
INIT OHM4METER  2
END
! TIME SEQUENCING SECTION
00:00:00 READ TEMPS
00:00:10 READ MULTIMETER  1
00:00:05 READ MULTIMETER  2
!
00:00:02 SET OVEN  2  420.0
00:00:02 WAIT HTEMP  4 380.0
00:00:02 SET OVEN 2 400.0
00:15:00 READ TEMPS
00:00:10 READ MULTIMETER  1
00:00:05 READ MULTIMETER  2
!
00:00:02 SET OVEN  2  620.0
00:00:02 WAIT HTEMP  4 580.0
00:00:02 SET OVEN 2 600.0
00:15:00 READ TEMPS
00:00:10 READ MULTIMETER  1
00:00:05 READ MULTIMETER  2
00:00:02 SET OVEN 2 20.0
END
```

FIGURE 6. - EXAMPLE OF INPUT DATA SET "SGxxxx.DAT."

Initialization commands are written without any time parameter since they are executed when the program is started.

Comment lines start with an exclamation mark as the first character.

The time sequencing section is after the first "END" command in the process control data set. Each command in this section is preceded by a delay time written as

hh:mm:ss

The first two numbers represent hours, the middle two minutes, and the right two seconds. This time is the time delay between the previous command and the command on this line.

The delimiter for the time-sequencing command lines is the space. Thus there must be a space between the delay time and the actual command.

The ACDA program can handle 100 initialization commands and 100 time sequencing commands at any one time. However this number can be extended by inserting the command "CHAIN" into the data set before 100 commands are reached. This allows the user to add another 100 commands. This can be continued as needed.

The process control data set must be terminated with an "END" command.

Running the Process

The process control data set which contains the users commands should be stored on a disk in drive "A:" as SGxxxx.DAT where xxxx can be any four digit number. The user should have the programs SETUP and ACDA on the default drive. The SETUP program is run first. Figure 7 shows a copy of the CRT screen that was obtained from the SETUP program. The program will ask the user for a run number. The number xxxx from SGxxxx.DAT should be entered as a four digit number. The commands in the data set SGxxxx.DAT will be printed out as they are checked by the SETUP program (comments are not printed). The SETUP program will stop with a message if there is an error. The error will be in the last command line printed out. In that case the user should fix the error and try again.

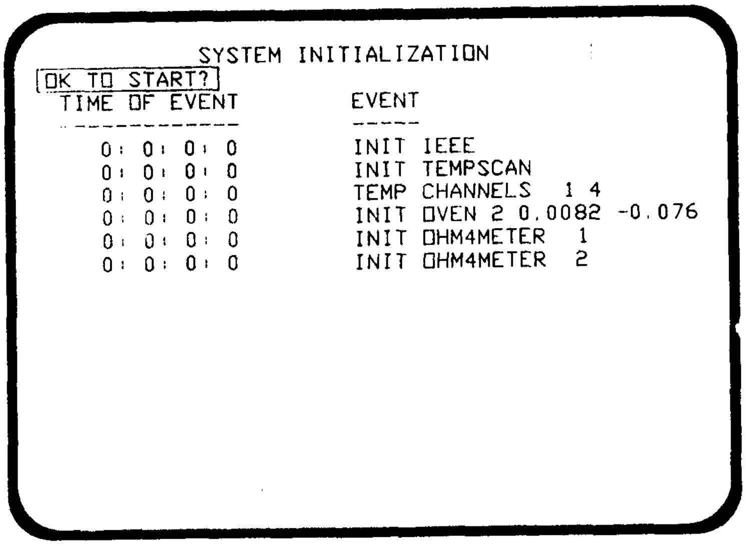
```
THIS PROGRAM CHANGES THE COMMANDS IN FILE A:SGXXXX.DAT
ENTERED BELOW.    ENTER THE RUN NUMBER IN THE FORM XXXX
ENTER RUN NUMBER =>
0008
A:SG0008.DAT
INIT IEEE
INIT TEMPSCAN
TEMP CHANNELS  1 4
INIT OVEN 2  0.0082 -0.076
INIT OHM4METER  1
INIT OHM4METER  2
END
00:00:00 READ TEMPS
00:00:10 READ MULTIMETER  1
00:00:05 READ MULTIMETER  2
00:00:02 SET OVEN  2  420
00:00:02 WAIT HTEMP  4 380.0
00:00:02 SET OVEN 2 400.0
00:15:00 READ TEMPS
00:00:10 READ MULTIMETER  1
00:00:05 READ MULTIMETER  2
00:00:02 SET OVEN  2  620.0
00:00:02 WAIT HTEMP  4 580.0
00:00:02 SET OVEN 2 600.0
00:15:00 READ TEMPS
00:00:10 READ MULTIMETER  1
00:00:05 READ MULTIMETER  2
00:00:02 SET OVEN 2 20.0

END

MAKE SURE A PRINTER IS ATTACHED TO DEVICE PRN:
BEFORE RUNNING THE "ACDA" PROGRAM
Stop - Program terminated.
```

FIGURE 7. - EXAMPLE OF CRT OUTPUT FROM "SETUP" PROGRAM.

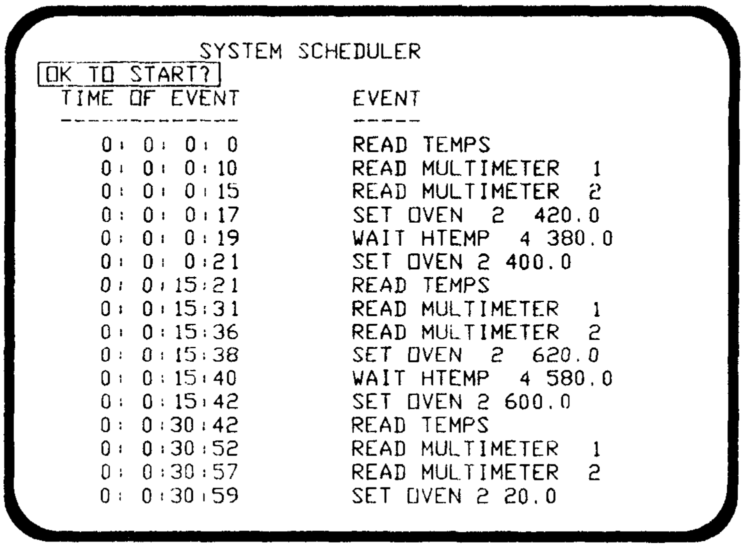
The second program that needs to be run is the ACDA program. An operating printer must be assigned as the "PRN:" device, since this program will print the commands as they are executed. The CRT output for the initialization part of the ACDA program is shown in figure 8. The words "OK TO START" will be shown in reverse video. A reply of "Y" will start the initialization. The "TIME OF EVENT" has no meaning here and will be zero. The commands or "EVENT" will be highlighted in reverse video after each is performed.



```
                   SYSTEM INITIALIZATION
[OK TO START?]
   TIME OF EVENT             EVENT
   -------------             -----
      0: 0: 0: 0             INIT IEEE
      0: 0: 0: 0             INIT TEMPSCAN
      0: 0: 0: 0             TEMP CHANNELS  1 4
      0: 0: 0: 0             INIT OVEN 2 0.0082 -0.076
      0: 0: 0: 0             INIT OHM4METER  1
      0: 0: 0: 0             INIT OHM4METER  2
```

FIGURE 8. - CRT OUTPUT FROM INITIALIZATION SECTION OF ACDA PROGRAM.

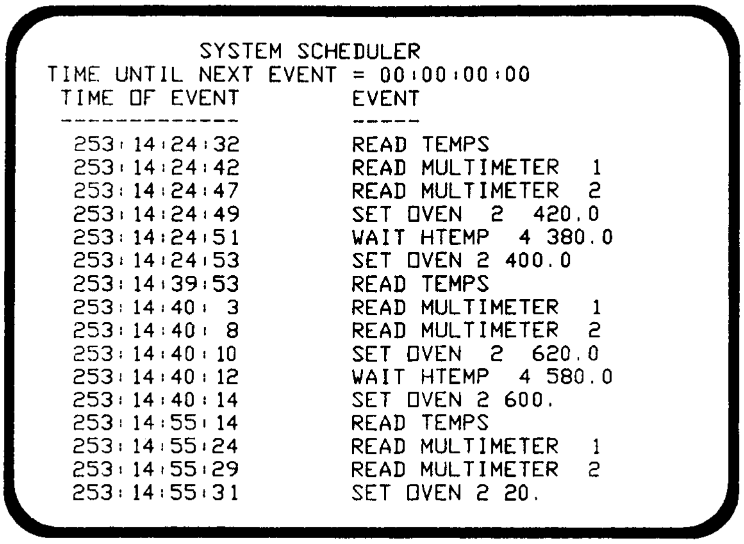
After the initialization is completed the time sequencing or "SCHEDULER" part of the ACDA program is displayed on the CRT. This display is shown in figure 9 before typing "Y" to start the SCHEDULER. The commands are displayed under the "EVENT" column. The commands will be scrolled up as each is executed if there are more than 20. The column under "TIME OF EVENT" represents the time since the SCHEDULER started. The time from left to right is days, hours, minutes, and seconds.



```
                   SYSTEM SCHEDULER
[OK TO START?]
   TIME OF EVENT             EVENT
   -------------             -----
      0: 0: 0: 0             READ TEMPS
      0: 0: 0:10             READ MULTIMETER  1
      0: 0: 0:15             READ MULTIMETER  2
      0: 0: 0:17             SET OVEN  2  420.0
      0: 0: 0:19             WAIT HTEMP  4 380.0
      0: 0: 0:21             SET OVEN 2 400.0
      0: 0:15:21             READ TEMPS
      0: 0:15:31             READ MULTIMETER  1
      0: 0:15:36             READ MULTIMETER  2
      0: 0:15:38             SET OVEN  2  620.0
      0: 0:15:40             WAIT HTEMP  4 580.0
      0: 0:15:42             SET OVEN 2 600.0
      0: 0:30:42             READ TEMPS
      0: 0:30:52             READ MULTIMETER  1
      0: 0:30:57             READ MULTIMETER  2
      0: 0:30:59             SET OVEN 2 20.0
```

FIGURE 9. - CRT OUTPUT FOR TIME SEQUENCING PART OF ACDA PROGRAM BEFORE TYPING "Y".

The display will change after a "Y" is typed in to start the program as shown in figure 10. The column under "TIME OF EVENT" is now changed to the actual time the command is scheduled to take place. The time is represented as days, hours, minutes, and seconds from the beginning of the current year. Each event is highlighted in reverse video after it is executed.



```
              SYSTEM SCHEDULER
TIME UNTIL NEXT EVENT = 00:00:00:00
 TIME OF EVENT          EVENT
 -------------          -----
   253:14:24:32         READ TEMPS
   253:14:24:42         READ MULTIMETER  1
   253:14:24:47         READ MULTIMETER  2
   253:14:24:49         SET OVEN  2  420.0
   253:14:24:51         WAIT HTEMP  4 380.0
   253:14:24:53         SET OVEN 2 400.0
   253:14:39:53         READ TEMPS
   253:14:40: 3         READ MULTIMETER  1
   253:14:40: 8         READ MULTIMETER  2
   253:14:40:10         SET OVEN  2  620.0
   253:14:40:12         WAIT HTEMP  4 580.0
   253:14:40:14         SET OVEN 2 600.
   253:14:55:14         READ TEMPS
   253:14:55:24         READ MULTIMETER  1
   253:14:55:29         READ MULTIMETER  2
   253:14:55:31         SET OVEN 2 20.
```

FIGURE 10. - CRT OUTPUT FOR TIME SEQUENCING PART OF ACDA PROGRAM AFTER TYPING "Y".

Description of Applications

This computer program is used in a high temperature sensor research program at NASA Lewis. The research involves measuring various strain gage parameters at many different temperatures and for long periods of time.

The equipment consists of a 10 channel digital thermocouple for measuring temperature, a dual programable power supply for setting the temperature of two ovens, a digital multimeter for making voltage and four-wire ohmmeter tests on the strain gages, a stepping motor for bending the strain gages, and a 10 channel strain gage bridge. A PC communicates with this equipment by means of an RS-232 port and a IEEE-488 buss as shown in figure 11.

The computer program can remotely control all of the instruments without any intervention by the an operator once an experiment is set up. A typical experiment would be to heat two strain gages to various temperatures and measure the resistance of the gages as a function of temperature. A typical data set to do this is shown in figure 6. The first part of the data set (up to the first END command) initializes the IEEE buss, the temperature scanner, the oven no. 2, and the multimeter to be used as a four-wire ohmeter.

The second part of the data set (after the first END command) records the temperature, resistance values, and then changes the oven temperature. Each

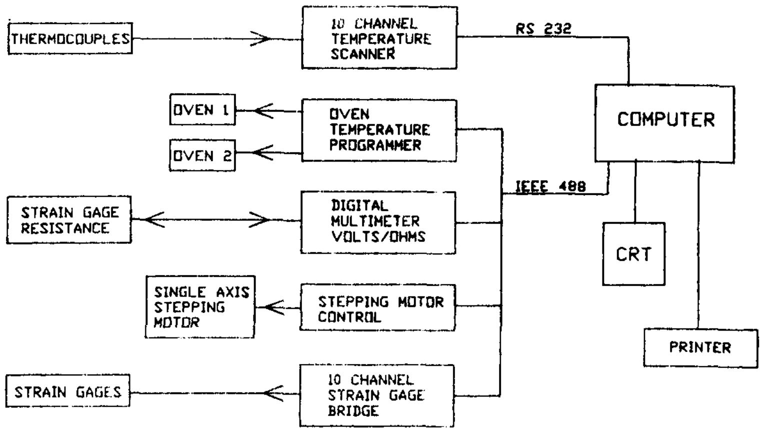

THERMOCOUPLES
10 CHANNEL TEMPERATURE SCANNER
RS 232
COMPUTER
OVEN 1
OVEN 2
OVEN TEMPERATURE PROGRAMMER
IEEE 488
STRAIN GAGE RESISTANCE
DIGITAL MULTIMETER VOLTS/OHMS
CRT
SINGLE AXIS STEPPING MOTOR
STEPPING MOTOR CONTROL
PRINTER
STRAIN GAGES
10 CHANNEL STRAIN GAGE BRIDGE

FIGURE 11. - EQUIPMENT USED FOR HIGH TEMPERATURE SENSOR RESEARCH PROGRAM.

function is done at a predetermined time. The link between the computer program and the instruments are the external driver subroutines. These subroutines control the instrument and if needed can send the data to the printer. The computer program (ACDA) also takes care of printing each command as it is executed.

CONCLUSION

In conclusion this report is intended to be used as an instruction manual for setting up and operating the SETUP and ACDA programs. The operator can make up custom commands for operating and taking data from external research equipment. The programs allow the operator to control and take data from external equipment at any time of the day or night without the operator being present.

National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-101988	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Laboratory Process Control Using Natural Language Commands From a Personal Computer		5. Report Date April 1989
		6. Performing Organization Code
7. Author(s) Herbert A. Will and Michael A. Mackin		8. Performing Organization Report No. E-4690
		10. Work Unit No. 505-62-01
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135-3191		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

The programs described in this report are available from COSMIC; they are identified as LEW-14907.

16. Abstract

PC software is described which provides flexible natural language process control capability with an IBM PC or compatible machine. Hardware requirements include the PC, and suitable hardware interfaces to all controlled devices. Software required includes the MS-DOS operating system, a PC-based FORTRAN-77 compiler, and user-written device drivers. Instructions for use of the software are given as well as a description of an application of the system.

17. Key Words (Suggested by Author(s)) Personal computer; System controller; Timed sequence; Research equipment	18. Distribution Statement Unclassified – Unlimited; Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of pages	22. Price*
Unclassified	Unclassified	14	A03

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161