SIMULA

Digital Controller Simulation Package

by

L. K. Lauderbaugh
and
A. G. Ulsoy

Technical Report No. UM-MEAM-84-22

Department of Mechanical Engineering
and Applied Mechanics
University of Michigan
Ann Arbor, MI 48109-2125

March 1984

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This document describes the SIMULA digital controller simulation package. This program simulates the "Direct Digital Control" (DDC) of a continuous time plant. The package is written in FORTRAN and runs under the RT-11 operating system on an LSI-11/23 minicomputer system.

This document is organized as follows, included in this chapter is an introduction to the simulation package and a brief introduction to DDC. Chapter two contains detailed information on the problem formulation. Chapter three explains how to put the correctly formulated problem into the user supplied subroutines. Chapter four is a presentation of more advanced options available in the user supplied routines. Once the problem is formulated and the user supplied routines prepared, you will be ready to link and run your simulation. Chapter five contains the RT-11 commands required to run the program. Also included in chapter five is information on the data input. Chapter six contains an example of a controller design for a first order system. Appendix A contains installation notes. Appendix B contains listings of the user supplied routines for the example in Chapter Six, and Appendix C contains listings of the source code.

Direct digital control is the standard way a digital computer is used as a controller. Here the computer replaces the analog control circuit of conventional controllers. Figure one shows a diagram of a typical sample data type DDC system. The digital computer samples the process output variables, Y, by

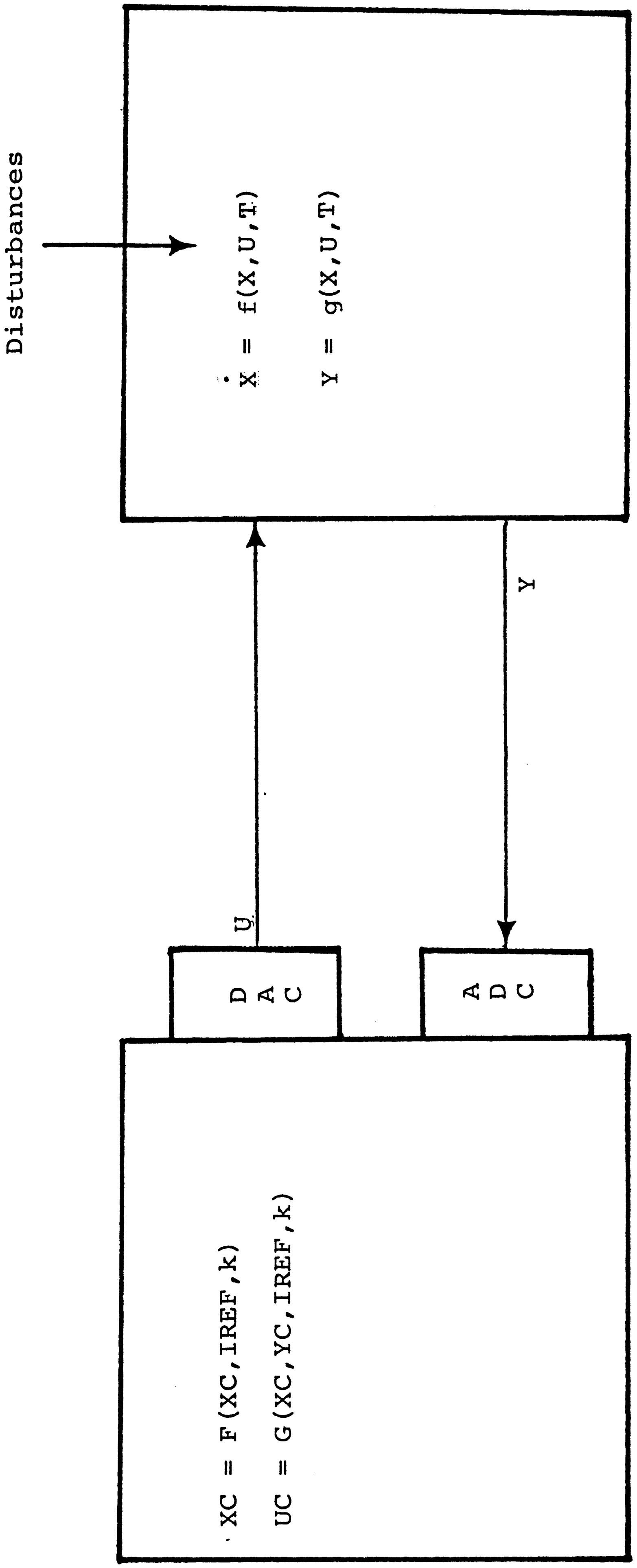
Disturbances
Ẋ = f(X,U,T)
Y = g(X,U,T)
U
Y
DAC
ADC
XC = F(XC,IREF,k)
UC = G(XC,YC,IREF,k)

Figure 1

means of an analog to digital converter (ADC). These converted values, IYC, can then be used to produce a control action, IUC. This command is converted to a continuous signal, U, and applied to the plant. The simulation package presented here provides a tool for evaluating such systems.

The effects of sampling rate, saturation, control algorithms, round off errors, and integer overflows are all of interest to the engineer designing DDC controllers. These effects may readily be evaluated using digital simulations with this package.

Figure two shows a simple flow chart for the simulation program and Appendix C contains a listing of the source code. The program first assigns zero to all the initial values of the plant and controller state variables. These are displayed and the user may change any of the initial values. The program then displays the default values of the timing constants, these include the sampling rate, delays etc., and requests changes. The program will then display the constants for the simulation of the ADC and the DAC interfaces. When the user is finished with the data entry, the program begins the simulation. During the simulation the message "CALCULATING" is displayed. These calculations may easily involve hundreds of thousands of operations, so this may take a few minutes. The exact time depends on the complexity of the plant and controller equations and the time constants. When this section is complete the program will ask you to select variables for printing or plotting. Plots may be made on the graphics display or hard

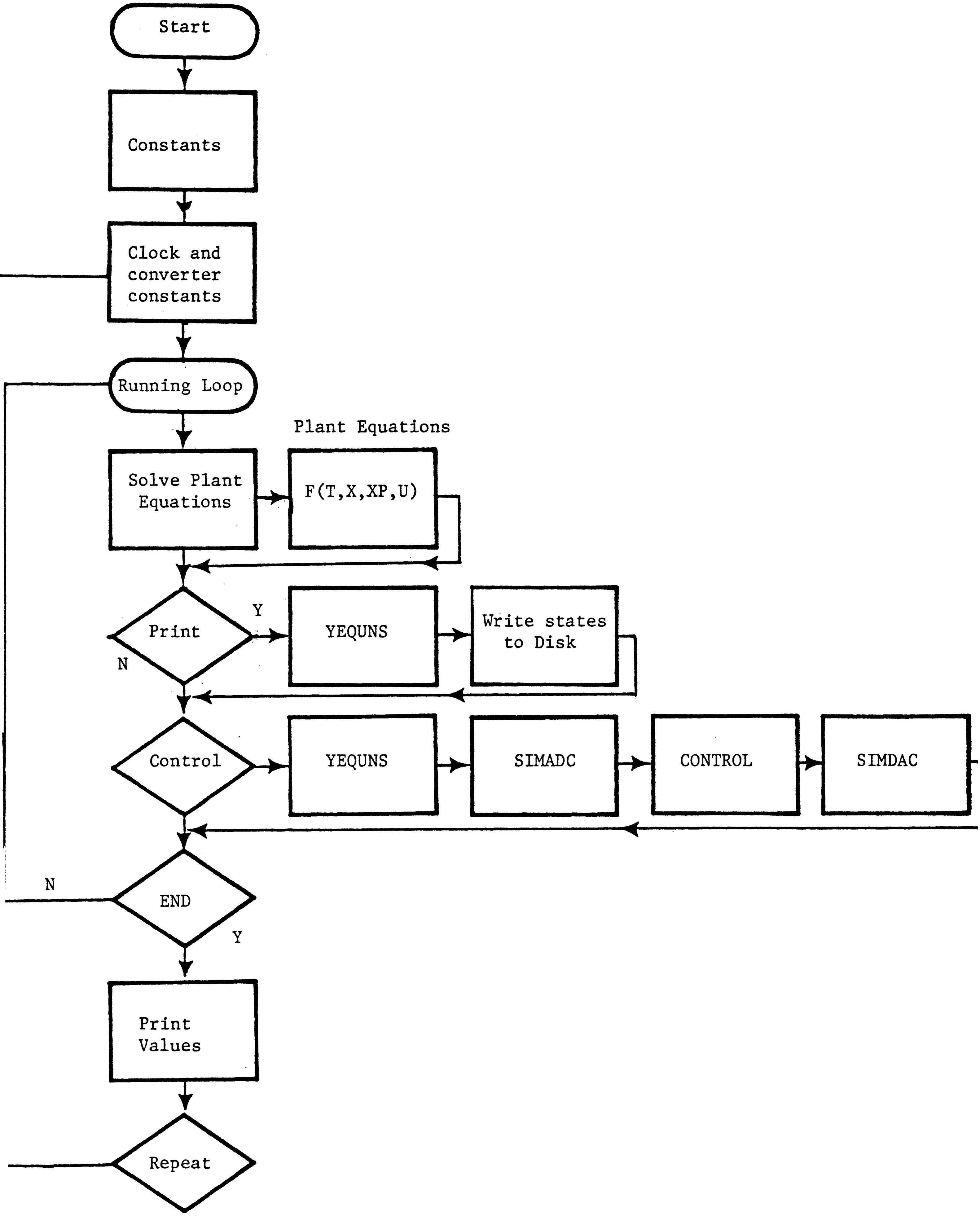
Start
Constants
Clock and converter constants
Running Loop
Plant Equations
Solve Plant Equations
F(T,X,XP,U)
Print
Y
N
YEQUNS
Write states to Disk
Control
YEQUNS
SIMADC
CONTROL
SIMDAC
N
END
Y
Print Values
Repeat

Figure 2

copies may be made on the X-Y plotter.

In the analysis of DDC systems it is sometimes desirable to look at the effects of disturbance inputs and variation of the constants in the plant equations. Chapter four describes the required modifications to the user supplied routines to evaluate these disturbances.

CHAPTER 2

PROBLEM FORMULATION

Since the simulation program is written to be a general program with a user supplied control algorithm and plant equations, the problem must be formulated in a standard way. The following is the generally accepted "modern control theory" or "state space" formulation. Also included in this chapter is a description of the behavior of the DAC and the ADC.

1. Process state equations The process (plant) being analyzed is assumed to be continuous and nonlinear the state equations may also be nonstationary. The equations are expressed in the form:

   $$dX/dT=f(X,U;T)$$
   $$Y = g(X,U,T)$$

   where,

   X is the process state vector of dimension NEQN

   U is the mainipulated process input vector of dimension NIN

   Y is the process output vector of dimension NOUT

   The functions f and g are in general nonlinear vector functions. These plant equations will be put in the user supplied subroutine F (see Chapter 3).

2. Controller state equations: The controller equations are discrete time, nonlinear, integer equations. Real arithmetic may be used but the input to the controller, IYC, and the controller output, IUC, are integer arrays. This will require the use of IFIX and FLOAT commands to change type. The controller equations are expressed in state variable form.

   $$IXC(k+1)=F(IXC,IYC,IREF,k)$$

   $$IUC(k)=G(IXC,IYC,IREF,k)$$

   IXC is the controller state equation of dimension NCEQN

   IYC is the converted value of the plant output, of dimension NOUT

   IUC is the controller output, of dimension NIN

IREF is the vector of reference values of dimension NOUT

k is the sampling interval

The functions F and G are in general nonlinear vector functions. The controller equations appear in the user supplied subroutine CONTRL (see Chapter 3).

3. <u>Digital to analog conversion</u>: The digital control signal calculated in the control algorithm is converted to continuous time and sent to the plant via the DAC. This device takes the integer values IUC and first converts them to a real value then multiplies them by a gain and performs a zero order hold. The DAC can can saturate and the maximum values (saturation) can be selected.

4. <u>Analog to digital converter</u>: The plant outputs, Y, are converted to digital values by the ADC. The converter is simulated by multiplying the continuous plant outputs by a gain then these values are converted to an integer value. Saturation is also accounted for.

NOTE

> In both the ADC and DAC operations the integer numbers have a gain and an offset associated with them. This must be taken into account then selecting reference values and when calculating controller outputs.

CHAPTER 3

USER SUPPLIED ROUTINES

Chapter two described the state space formulation. Once the problem has been formulated you are ready to begin writing the user supplied routines. This is actually much easier than it sounds. There is a file called SIMAIN.FOR which you edit to match your specific problem. Most of the work has already been done. All locations where editing is required are marked with,

```
C********** INSTRUCTIONS **********
```

Where INSTRUCTIONS tell what changes to make. The first thing you should do is copy this file to a file to edit. This can be accomplished using the command,

```
COPY SIMAIN.FOR YOURFL.FOR
```

Where YOURFL is the file name you select. The name must be six characters or less and the suffix .FOR must be used. The file SIMAIN contains the main calling program and three subroutines. The main program is used to dimension arrays used in the simulation, to set some constants that depend on the specific problem being solved, and to call the simulation routine. The three subroutines are, F which contains the plant state equations, YEQUNS which contains the static output equations, and CNTROL which contains the controller equations.

Before we get into the specifics of these routines some general comments need to be made.

The program uses "run time dimensioning" for all the arrays the user comes into contact with. This means that all data arrays are dimensioned in the main program and all subsequent dimension statements (found in each subroutine) are dimensioned one. Then when the program is compiled the compiler, in effect, changes all the unity dimensions to those specified in MAIN. This means that you will have to set the correct dimenions on each of the arrays in the main program but not in any of the subroutines.

The second general comment is that in more complicated problems it may be desirable to use matrix methods in the control and plant equations. This can be done by using the MATRIX library supported on the "Manufacturing systems modeling and control lab.'s" LSI-11/23 computer system. There is a separate document describing the use of the library which is included as Appendix E. The only change in the procedure outlined in this document is that the name of the matrix library must be included in the link list (see Chapter 5).

The final general comment regards the use of common blocks. The program uses several common blocks. For a detailed description see the source listings in Appendix C. The required common blocks appear in the SIMAIN file. THESE COMMON BLOCKS SHOULD NEVER BE TAMPERED WITH.

Now for the details. First, the required changes in the main program. There are two changes required in MAIN. You must put the correct dimensions in and change some of the constants. The constants you change refer to the number of plant equations, NEQN, the number of plant inputs, NIN, the number of plant outputs, NOUT, and the number of controller equations, NCEQN. The dimensions to be changed are the arrays that contain the plant and controller states, outputs and inputs. So, the dimensions are all functions of the constants mentioned in Chapter 2. See the SIMAIN listing for the exact changes required.

When you are done making changes in MAIN move on to the subroutine F which contains the plant equations. Here you enter the set of first order differential equations, where DX(1)/dT=XP(1). For example, the equation,

dX(1)/dT=2.0*X(1)+3.0,

becomes,

```
XP(1)=2.0*X(1)+3.0
```

The routine F is called many times by the routine used to solve this set of differential equations. This means the equations in this routine have a large effect on how long the simulation takes. To speed up the calculations, the equations found here should be as simple as possible. In the above example a better equation would be,

```
XP(1)=X(1)+X(1)+3.0
```

Here the multiplication by two has been replaced by one addition. Additions are much faster than multiplications.

You should now be ready to edit the CONTRL routine. Here you enter your control equations. For example, the equation,

IXC(1)(k+1)=2*IXC(1)(K)+3 (1)

and

IUC(1)(k+1)=IUC(1)(K) + 2* IXC(1)(K+1)

becomes,

```
IXC(1)=2*IXC(1)+3 (3)
```

and

IUC(1)=IUC(1)+2*IXC(1) (4)

Here the subscripts gets a little confusing. Equation 1 is interpreted as: the first controller state variable is found by multiplying the old value by two and adding three. This produces equation 3. Equation two is similar the new value of IUC is the old one plus two times the new controller state. This produces equation four. This is important so make sure you know how to go from equation one and two to three and four.

Finally the plant output equations are entered in the subroutine YEQUNS.

This completes the changes required to produce the user supplied routines. SIMAIN.FOR contains the equations used with the example discussed in chapter 6.

CHAPTER 4

ADVANCED OPTIONS

Chapter three discribes all of the basic requirements to write the user supplied routines. This will allow you to evaluate the performance of complicated digital control systems. There are, however, a few advanced options available in the simulation package. In the design of a digital control system it may be important to know the behavior of the system under the influence of external disturbances and possibly the effects of variation in the constants of the plant equations. Both of these can be examined by modifying the user supplied routines.

An external disturbance may be added to the plant equations as follows: One of the arguments transmitted to the F routine is the current value of the time, T. To add a disturbance, one simply adds a conditional statement based on the time. For example, if the state equation is,

```
XP(1)=X(1)+U(1)
```

Replace this equation with the three statements,

```
DIS=0.
IF(T.GE.TSTART)DIS=C1
XP(1)=X(1)+U(1)+DIS
```

Where TSTART is the time when the disturbance is to begin, and C1 is the value of the disturbance. C1 may be a constant or some function of time, the states, or the input to the plant.

A similar strategy can be used for variations in plant parameters. For example, if the state equation is,

```
XP(1)=A*XP(1)+U(1)
```

Where A is some function of time, A(T). This would be programmed as

```
A=A(T)
XP(1)=A*XP(1)+U(1)
```

Since you can only select plant inputs, outputs, or states or controller states for display, it would not be possible to print or plot the values of A or DIS. If you wish to be able to display these values you have to add an additional state to the plant. This will change the value of NEQN. Also remember that the routine F solves NEQN differential equations. This means that you must also add the statement

```
XP(J)=0
```

which is equivalent to

dX(J)/dT=0 or X(J) is a constant

where J is the index of the added state variable. For more details see the example in Ch 6.

CHAPTER 5

RUNNING SIMULA

This chapter describes the steps required to run your simulation. When you are done making changes to YOURFL.FOR you are ready to run the compiler. Use the following command,

```
FORT/LIST:DLO:YOURFL
```

When the compiler is done, if there are errors the file YOURFL.LST (created by the compiler) contains the compiler listing of YOURFL, correct the errors in YOURFL.FOR and move on to linking. To link YOURFL with the simulation program and the graphics routines that SIMULA uses, enter the following command

```
LINK YOURFL,SIMULA,LEALPT
```

When this is done type

```
RUN YOURFL
```

and you are under way.

Now the description of the actual running of your simulation. First make sure the graphic display is turned on and the X-Y plotter is turned OFF. The program will tell you when to turn on the plotter.

The program will first clear the screen and print the initial values of the state vector. If you want to put in any initial conditions enter yes or y. The program will ask for the index of the value to be changed. Enter the integer value followed by a comma.

NOTE

> All numerical input should be followed by a comma. If you want to enter a zero you may just hit return.

The program will now print the new values. This process is repeated until you enter no or n.

The same process will then be repeated for the state variables for the controller.

NOTE

> The units used may be whatever you wish as long as you are consistent. Time may be in seconds, hours or fortnights as long as all time units are the same and the units in the state equations agree.

The program will now display several time constants.

1. Delta T for RK4: This is the time step used for the Runge Kutta routine to solve the D.E.'s. The value should be at most 1/10th of the Delta T Control (described below). Smaller delta T will generally produce a more accurate simulation (within limits) but will require longer execution times.
2. Ending Time: This is the amount of time the simulation runs.
3. Delta T Control: This is the sampling interval.
4. Control Delay: This is the length of time between the sampling and the sending of the control. This option is currently not installed. Leave this at 0.0.

When all these values are correct for your particular case, hit return to continue.

The program will now display the DAC and ADC simulation variables. These are,

1. ADC Gain: The gain of the ADC converter. This is found by dividing the digital range by the voltage range of the converter being simulated.

2. ADC Offset: the offset of the digital value.

3. Max ADC: The maximum digital value of the ADC.

4. DAC Gain: The gain of the DAC. This is found by dividing the voltage range by the digital range. For example, if the voltage range is -5.12 volts to 5.12 volts and the digital range is 0 to 4095 (12 bit converter) then the gain is 10.24/4096=.0025 volts/count.

5. DAC offset: This is the offset of the digital values. In the above example the offset is 2048. This is the digital value that corresponds to a zero voltage.

6. Min DAC: This is the minimum voltage output. In our example this is -5.12

7. Max DAC: This is the maximum voltage output. In our example this is 5.12

The initial values that are displayed are the values used on a typical control configuration. Unless you are working on a specific system which uses different values use the default values.

The program will now perform the simulation and display the message "CALCULATING".

When the simulation is complete the program will go into the graphics selection mode. There are two stages to this selection process. The first selects which variables are available for display. This is necessary because of memory constraints. You are restricted to ten variables at a time. If you desire more than this you must run the simulation more than once.

The first variable to be selected is the independent variable. Independent and dependent refer to plotting. The horizontal axis being the independent variables. You may choose time to be the indep. variable by entering 0. Or you may select one of the plant states by entering the index of that state.

You will then be asked to enter up to ten dependent variables. These will be a series of questions asking if you wnat any plant states, inputs, outputs etc. In each case enter the desired index or 0 if you don't want any of that particular variable. The question will be repeated until you enter 0. This allows you to select as many of each variable as you wish.

In the second stage of the variable selection you may select from this set of variables for plotting, but now the program only knows these variables in the order you have selected them so it is important to remember which variables you have selected and in what order.

The program will then ask you if you want to print the results. If you respond yes a hard copy will be produced. The results will be printed at 200 equal time steps.

The program will then ask if you wish to plot the results. If you respond yes you will then be asked to enter the index of the dependent variable to be plotted. The program then finds the maximum and minimum of the data and asks you for any changes. Here there are several options. If you respond no to the changes question, the data will be plotted so that the maximum of the absolute values of the data will be the extreme value of the plot, i.e. the data is scaled so that it fills the entire plot. This may not be desirable for several reasons. This type of scaling makes the maximum value of the axis set the same as the maximum of the data this may not be a "nice" value. The other problem is that the data will be plotted on an axis set appropriate for that data (the program chooses between quadrant

I, II and IV, or all four). This is fine for each separate plot but if you are plotting more than one plot on the same page you will probably want the data to be plotted on the same axis set.

It is possible to alter this plotting procedure, but this requires some understanding of how the plotting routines work.

As mentioned above, the program first finds the maximum and minimum values of the independent and the dependent variables. These are all that is used in finding the scale factors and selecting the axis type.

For this discussion lets say you were plotting time as the independent variable and the DAC output, U, as the dependent variable. Further assume that the min value of the time is 0 and the max is 10. The minimum voltage is -4.0 the maximum is 3.0. The plotting routine first will scale the data by finding the absolute maximums. Here that would be 10 for the time and 4.0 for the voltages (the absolute value of -4). The data will be scaled so that a voltage of 4 and a time of 10 will be the full scale values. If, for some reason, you wish to have 8.0 volts be the maximum value of the plot simply respond yes to the changes prompt (printed after listing the max and min.). The program will request new values of DVMAX,DVMIN,RIDVMX,RIDVMN. These are the maximum and the minimum of the dependent and independent variables. To produce the desired changes enter the following,

```
DVMAX=8.0
DVMIN=-4.0
RIDUMX=10.0
RIDVMN=0.0
```

For scaling the program uses the absolute maximums so you could also have entered the following,

```
DVMAX=3.0
DVMIN=-8.0
RIDVMX=10.0
RIDVMN=0.0
```

This would produce the same plot as the first set of maximums and minimums.

The other possibility is to change the axis set on which the data is plotted. This is determined by the sign of the maximum and minimum. For example in our case the data would be plotted in quadrants I and IV because there are only positive values of the time and positive and negative values of the dependent variables. If you wanted the data to be plotted in all 4 quadrants you would enter a negative value for the minimum of the independent variable. If you do not want this change to affect the values used for scaling remember that only the absolute maximum determines the scale factor. In our example if one wants to plot the data on all four axes with no changes in scale the following could be used,

```
DVMAX=3.0
DVMIN=-4.0
RIDVMX=10.0
RIDVMN=-1.0
```

If you wanted all four quadrants with the maximum time of 20 the following could be used.

```
DVMAX=3.0
DVMIN=-4.0
```

RIDVMX=10.0

RIDVMN=-20.0

Here the -20 changes the axes type and specifies the new scale factors. The following will produce the same plot.

DVMAX=8.0

DVMIN=-4.0

RIDVMX=-1.0

RIDUMN = 20.0

When this selection and scaling is complete the program will ask if you want to make a hard copy. If you respond yes the program will instruct you in the use of the plotter. If you respond no then the program will make the plot on the screen. When the plotting is complete if you wish to do more plotting you will be asked if you want to erase the screen. If you respond "no" then the next plot will be on top of the previous one. This is a useful function when examining the effect of disturbances or tracking an input. You may replot the same variable as many times as you wish. This makes it possible to make a plot on the screen and adjust the scale factors etc. Until the plot is what you want then make a hard copy.

The best way to learn how to use the features of the program is to run it and play. Ch. 6 contains a brief description of an example. Run this example, try some different things, change the sampling rate, try some of the plotting options. By following this example you should be able to quickly learn how the program works, design your controller and evaluate its performance.

CHAPTER 6

EXAMPLE

This chapter discusses a simple example demonstrating the features of the SIMULA simulation package. The user supplied routines in SIMAIN.FOR are the routines used for this example. If you want to run this example copy this file to YOURFL and compile and link it as is. The plant is a first order system described by the differential equation.

dX/dT+(K/C)*X=Fd/C+U(T)/C (1)

where K varies with time. This can be thought of as a mass-spring-damper where the mass is negligable. The controller is designed for K=1.0, C is constant and equal to 0.1. Also it is assumed that there is no disturbance (Fd) acting. Equation one then becomes,

dX/dT=-10.*K*X+10*U (2)

For this example a simpl

algorithm for a P contro

UC(k+1)=KP*e(k) (3)

Where e(k)=REF(k)-Y(k)

REF(k) is the value of the reference input. Here we let the reference input be a sine wave with a period of $2\pi/3$ seconds. From digital control theory with K=1.0 and a sampling interval of 0.05 seconds, the maximum value of KP for stability is 4. In this simulation a KP of 1 is used. This will produce a nonoscillatory, stable response. We are interested in the effect of the variation of K around the design value of 1. This is evaluated by letting K change from 1 to 2 to 4 at times of .5 sec and 1.0 seconds respectively.

Appendix B contains a listing if the SIMULA.FOR program implementing this example. The subroutine F contains the plant equations. Lines 3 to 7 determine the value of K, while 9 evaluates the differential equation. The controller equations are found in the CONTRL subroutines. Line 5 determines the reference input while 6 through 9 determine the controller output and stores the various control states for printing. Note that in lines 5 and 10 the 2048 offset is added, this is required because the DAC has an offset of 2048, i.e. the digital value of 2048 corresponds to a voltage of 0.0 volts.

The subroutine YEQUNS is very simple. The output of the plant is simply the state X(1).

The results of these simulation are shown in figures 3 and 4. Figure 3 shows 3 plots on the same axes set. The time runs from 0 to 2 sec. The dependent variables plotted are the state X (the position of the mass in our mass spring damper system), the reference signal that the controller is trying to follow, and the spring constant K. Here you can see that the performance is quite good until the spring constant changes from the value of 1 used in the design of the controller. After the spring constant changes to 4 the performance is quite bad. The response actually flattens out when the reference signal goes negative. This is due to the saturation of the DAC. Figure 4 shows this quite clearly. Here the plant state is replotted with the control signal U, which is the output of the DAC. You can see that the control signal reaches its largest negative value of -5.12 and can go no further.

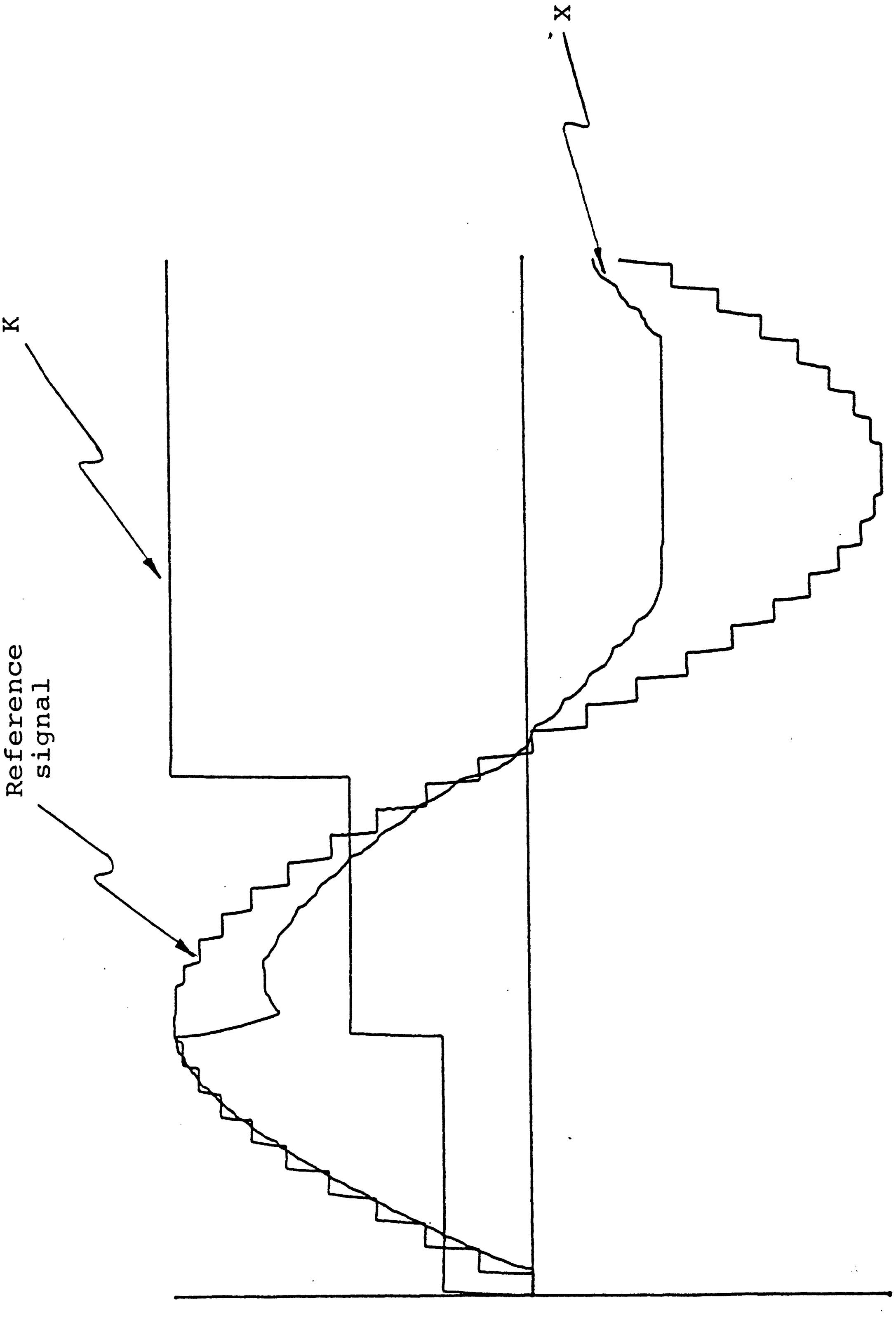

Figure 3

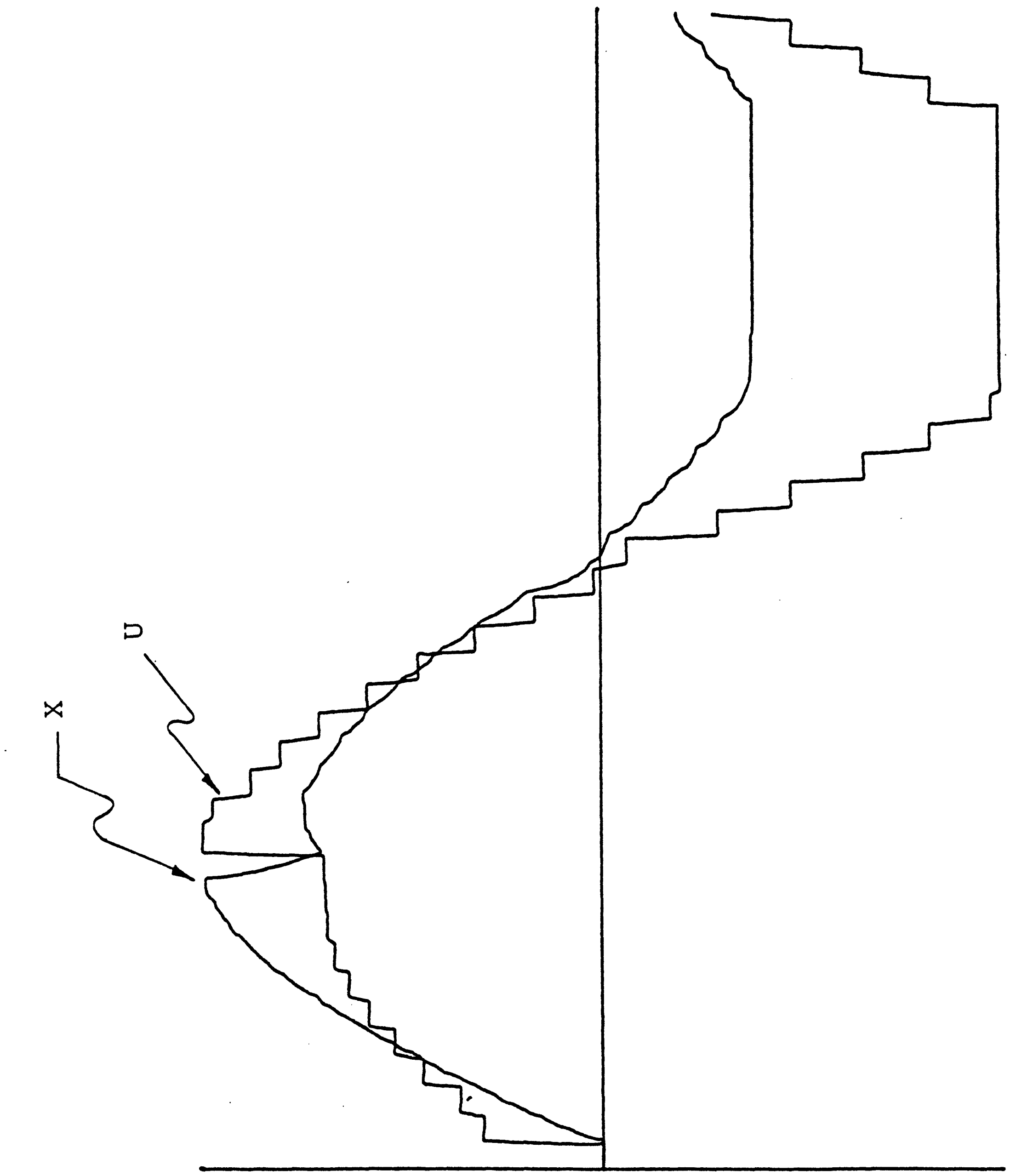

Figure 4

From this simple example many of the characteristics of DDC controllers can be observed. We have seen the effects of parameter variation, tracking performance, and saturation. With a simple modification to the control and plant equations, you can easily evaluate the response to disturbance inputs and try different control algorithms.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the work of Paul Becker who made an initial attempt at developing this package in Pascal. We would also like to acknowledge the financial support of the National Science Foundation, Grant MEAM-8112629 and we would like to thank Mrs. Teresa Combs for helping to prepare the manuscript.

APPENDIX A

INSTALLATION NOTES

The SIMULA simulation package was written to run on the "Manufacturing Systems Modeling and Control Labs" LSI-11/23 system. Some of the software is machine dependent and will require modification to run on other systems.

First the program assigns devices 5 and 6 to the TT: (the console terminal) if your system will not allow such an assignment this will require modification.

The remaining machine dependent features pertain to the graphics output. The program uses the graphics library LEAL PLOT. This library has some special installation requirements described in the manual for the graphics library. This document is included in Appendix D. If this software is being installed on a system supporting PLOT 10 the commands are very similiar except LEAL PLOT has no provisions for windowing, all graphics are performed in screen coordinates. Thus the conversion to PLOT 10 should be relatively easy.

The algorithm used for integrating the plant equations is a fourth order Runge Kutta routine. The execution time can be decreased by using a single pass algorithm, such as the fourth order Adams Bashforth algorithm. However, this algorithm is not as robust and the selection of the time step becomes much more critical, so this type of algorithm is not recommended for a general purpose program such as this. The user could, however, use his/her favorite numerical integration technique be replacing or modifying the subroutine RK4 shown in Appendix C.

APPENDIX B

LISTINGS OF USER SUPPLIED ROUTINES

```
  General Description: This is a driver routine for the simulation
               program

  External References: SIMULA

              by Leal Lauderbaugh


clarations

  LOGICAL*1 ANSW,NO,YES

******** CHANGE DIMENSIONS HERE *********

E DIMENSION OF X AND XP =NEQN
E DIMENSION OF IXC =NCEQN
E DIMENSION OF Y,IYC AND IREF =NOUT
E DIMENSION OF U AND IUC =NIN
E DIMENSION OF IUC  =NOUT
E DIMENSION OF WORK = 5*NEQN

  REAL*4 U(1)
  DIMENSION X(2),XP(2),Y(1),IXC(3),IYC(1),IUC(1),IREF(1)
  DIMENSION WORK(10)

mmon blocks

  COMMON/INOUT/IN,IOUT
  COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
  COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
  COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY
  COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV
  COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN

ta statments
  CALL ASSIGN(5,'TT:')
  CALL ASSIGN(6,'TT:')
  DATA IN/5/,IOUT/6/


et up constants for simulation

  NO='N'
  YES='Y'

******** CHANGE SIZES HERE **********

N=# OF PLANT INPUTS

  NIN=1

UT=# OF PLANT OUTPUTS

  NOUT=1
```

```
QUN= # OF PLANT STATE EQUATIONS

  NEQN=2

EQN= # OF CONTROLLER STATE EQUATIONS

  NCEQN=3

******** EDITING IN MAIN COMPLETE *********

  CALL SIMULA(X,XP,Y,U,IXC,IYC,IUC,IREF,WORK)

eck for a repeat

  WRITE(IOUT,9991)
  FORMAT(' Do you wish to repeat the calculations? (y/n)')
  READ(IN,9992)ANSW
  FORMAT(A1)
  IF(ANSW.EQ.YES)GO TO 10
  IF(ANSW.NE.NO)GO TO 9990
  STOP
  END
```

```
 SUBROUTINE F(T,X,XP,U)
******* DO NOT CHANGE THESE DIMENSIONS *********
 REAL*4 T,X(1),XP(1),U(1)
******* DELETE THIS SET OF PLANT EQUNS. AND ENTER YOUR'S ****
 IF(T.LT.0.5)X(2)=1.
 IF(T.GT.0.5.AND.T.LT.1.0)X(2)=2.
 IF(T.GT.1.0)X(2)=4.
 XP(1)=-10.*X(2)*X(1)+10.*U(1)
 XP(2)=0.
 RETURN
 END
```

```
 SUBROUTINE CONTRL(T,IXC,IYC,IUC,IREF)
 COMMON/INOUT/IN,IOUT
******* DO NOT CHANGE THESE DIMENSIONS *********
 DIMENSION IXC(1),IYC(1),IUC(1),IREF(1)
******* DELETE THIS SET OF CONTROL EQUNS. AND ENTER YOUR'S ****
 KP=2
 IREF(1)=2048+2048*SIN(3*T)
C ERROR
 IE=IREF(1)-IYC(1)
 IXC(3)=IREF(1)
 IXC(1)=KP*IE
 IXC(2)=IE
 OFFSET
 IUC(1)=IXC(1)+2048
 RETURN
 END
```

```
 SUBROUTINE YEQUNS(X,Y,U)
******* DO NOT CHANGE THESE DIMENSIONS *********
 DIMENSION X(1),Y(1),U(1)
******* DELETE THIS SET OF EQUNS. AND ENTER YOUR'S ****
 Y(1)=X(1)
 RETURN
 END
```

```
 SUBROUTINE YEQUNS(X,Y,U)
******* DO NOT CHANGE THESE DIMENSIONS *********
 DIMENSION X(1),Y(1),U(1)
******* DELETE THIS SET OF EQUNS. AND ENTER YOUR'S ****
 Y(1)=X(1)
 RETURN
 END
```

APPENDIX C

PROGRAM LISTING

```
  SUBROUTINE SIMULA(X,XP,Y,U,IXC,IYC,IUC,IREF,WORK)
  General Description:This is a driver routine for control simulation.
              version 2.1

  External References:RK4
                  LEALPT
                  SLOPE

              by Leal Lauderbaugh


clarations

  LOGICAL*1 ANSW,NO,YES
  REAL*4 U(1)
  DIMENSION X(1),XP(1),Y(1),IXC(1),IYC(1),IUC(1),IREF(1)
  DIMENSION WORK(1)

mmon blocks

  COMMON/INOUT/IN,IOUT
  COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
  COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
  COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY
  COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV
  COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN

sign devices
  CALL ASSIGN(3,'DL0:PFILE.DAT')
  CALL ASSIGN(4,'DL0:CFILE.DAT')

t up initial values and constants

  CALL START(X)

  CALL INITAL(X,IXC)
  CALL CLKPRM
  CALL CNVPRM
ear screen and print calculating messsage
  CALL CLEAR
  WRITE(IOUT,20)
  FORMAT(20X,' CALCULATING ',//,20X,'Please wait...',/////)

  running loop

  U(1)=0
  T=0
lculate control and print for t=0
  TCRIT2=0.
  TCRIT1=0.
  GOTO 210
  CALL RK4CNT(T,X,XP,Y,U,WORK)

eck for sample
```

```
  IF(T.GE.TCRIT2)GOTO 300

eck for a print

  IF(T.GE.TCRIT1)GOTO 400
ntinue loop
  GOTO 200

mple

  CALL YEQUNS(X,Y,U)
nvert Y to YC
  DO 310 I1=1,NOUT
  CALL SIMADC(Y(I1),IYC(I1))
nd control
  CALL CONTRL(T,IXC,IYC,IUC,IREF)
nvert control to continuous time
  DO 320 I1=1,NIN
  CALL SIMDAC(IUC(I1),U(I1))
nd next control time
  TCRIT2=TCRIT2+DELT
  GOTO 250

int

  IXP=IFIX(XSCALE*T+125.)
  CALL YEQUNS
nvert Y to YC
  DO 410 I1=1,NOUT
  CALL SIMADC(Y(I1),IYC(I1))
  WRITE(3)T,(X(J),J=1,NEQN),(U(J),J=1,NIN),(Y(J),J=1,NOUT)
  WRITE(4)(IXC(J),J=1,NCEQN),(IUC(J),J=1,NIN),(IYC(J),J=1,NOUT)
eck for TEND
  IF(T.GE.TEND)GOTO 500
nd next print time
  TCRIT1=TCRIT1+TPRINT
ntinue loop
  GOTO 200

d loop, rewind files and call print routine

  CALL PRNOUT(T,X,Y,U,IXC,IYC,IUC)
  CLOSE(UNIT=3)
  CLOSE(UNIT=4)
  RETURN
  END
```

```
  SUBROUTINE INITAL(X,IXC)

  General Description: This routine is used to input the initial
  values for the differential equations of the plant and the controller.


  Arguments:

       Common blocks:

               COMMON/INOUT/IN,IOUT
               IN: The input device number is 5 which
                   is assigned to TT
               IOUT: The output device number is 6 which
                     is assigned to TT


               X: the REAL*4 state vector

               COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
               NEQN: integer number of plant state equations
               NIN: number of plant inputs.
               NOUT: number of output equations.
               NCEQN: number of controller equations.

  External References:none

               by Leal Lauderbaugh    Oct 83


  LOGICAL*1 ANSW,NO,YES
  DATA NO/'N'/,YES/'Y'/
  DIMENSION X(1),IXC(1)

mmon blocks

  COMMON/INOUT/IN,IOUT
  COMMON/SIZE/NEQN,NIN,NOUT,NCEQN

ite the plant state vector and request changes.

0 CALL CLEAR
  WRITE(IOUT,150)
  FORMAT(' The initial values of the state vector are, ')
  FORMAT(/,' X(',I2,')=',G12.4)
  FORMAT(' ')
  DO 200 I=1,NEQN
  WRITE(IOUT,160)I,X(I)
  WRITE(IOUT,170)

eck for a change

  WRITE(IOUT,9991)
  FORMAT(/,' Do you wish to make any changes ? (y/n) '$)
```

```
  READ(IN,9992)ANSW
  FORMAT(A1)
  IF(ANSW.EQ.YES)GO TO 1000
  IF(ANSW.NE.NO)GO TO 9990
swer is no
  GOTO 410
  WRITE(IOUT,300)
  FORMAT(/,' Enter the index of the value to be changed '$)
  READ(IN,310)IC
  FORMAT(I2)
  WRITE(IOUT,320)IC
  FORMAT(/,' input the value of X(',I2,')= '$)
  READ(IN,330)X(IC)
  FORMAT(G12.4)
  GOTO 110

ite the controller state vector and request changes.

) CALL CLEAR
  WRITE(IOUT,450)
  FORMAT(' The initial values of the controller state vector are')
  FORMAT(/,' XC(',I2,')=',I5)
  DO 500 I=1,NCEQN
  WRITE(IOUT,460)I,IXC(I)
  WRITE(IOUT,170)

eck for a change

  WRITE(IOUT,9991)
  READ(IN,9992)ANSW
  IF(ANSW.EQ.YES)GO TO 1200
  IF(ANSW.NE.NO)GO TO 8990
  GOTO 2000
  WRITE(IOUT,300)
  READ(IN,310)IC
  WRITE(IOUT,620)IC
  FORMAT(/,' input the integer value of XC(',I2,')= '$)
  READ(IN,630)IXC(IC)
  FORMAT(I6)
  GOTO 410

  RETURN
  END
```

```
 SUBROUTINE CLKPRM

 General Description:This routine is used to input time constants.

 External References: CLEAR

              by Leal Lauderbaugh    Oct 83
 Arguments:

     formal:none

     Common:

     COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY

             H-delta T for the RK4 routine(real)
             TEND-ending time(real)
             TPRINT-printing interval(real)
             DELT-sampling interval (real)
             DELDST-interval between disturbances(real)
             DELAY-interval between sampling and control(real)


mmon blocks

 COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY
 COMMON /INOUT/ IN,IOUT
 CALL CLEAR

 WRITE(IOUT,10) H,TEND,DELT,DELDST,DELAY
 FORMAT('1',5X,'CLOCK VARIABLE DISPLAY:',//,
+10X,'1) delta T for RK4 ',F8.4,'  2) ending time      ',F8.4,/,
+10X,'3) delta T control ',F8.4,'  4) delta T dist.    ',F8.4,,/,
+10X,'5) control delay   ',F8.4,/,)

 WRITE(IOUT,11)
 FORMAT(5X,'To change any of the variables enter the number of',/
+    ,5X,'the variable. Otherwise return to continue. '$)
 READ(IN,12,ERR=9) IFLAG
 FORMAT(I1)

 IF (IFLAG.EQ.0) RETURN

 WRITE(IOUT,8)
 FORMAT(//)
 GO TO (101,102,105,106,107) IFLAG

 ASSIGN 101 TO I
 WRITE(IOUT,201)
 FORMAT(' > Enter delta T for the Runge-Kutta routine (real)= '$)
 READ(IN,202,ERR=990) H
 FORMAT(F10.4)
 GO TO 9
```

```
 ASSIGN 102 TO I
 WRITE(IOUT,203)
 FORMAT(' > Enter the ending time (real)= '$)
 READ(IN,202,ERR=990) TEND
 TPRINT=TEND/200.
 GO TO 9

 ASSIGN 105 TO I
 WRITE(IOUT,206)
 FORMAT(' > Enter the sampling interval (real)= '$)
 READ(IN,202,ERR=990) DELT
 GO TO 9

 ASSIGN 106 TO I
 WRITE(IOUT,207)
 FORMAT(' > Enter delta T for the disturbance (real)= '$)
 READ(IN,202,ERR=990) DELDST
 GO TO 9

 ASSIGN 107 TO I
 WRITE(IOUT,208)
 FORMAT(' > Enter the control delay (real)= '$)
 READ(IN,202,ERR=990) DELAY
 GO TO 9

...Error message for illegal entry

 WRITE(IOUT,991)
 FORMAT(' ***** ERROR ***** Invalid entry, try again',/,' ')
 GO TO I

 CONTINUE
 RETURN
 END
```

```
  SUBROUTINE CLEAR

  General Description: This routine is used to clear the screen

  Arguments:

      Common blocks:

              COMMON/INOUT/IN,IOUT
              IN: The input device number is 5 which
                  is assigned to TT
              IOUT: The output device number is 6 which
                    is assigned to TT


              by Leal Lauderbaugh   Oct 83

mon blocks

  COMMON/INOUT/IN,IOUT

  DO 2 I=1,30
  WRITE(IOUT,3)
  FORMAT(' ')

  RETURN
  END
```

```
  SUBROUTINE RK4CNT(T,X,XP,Y,U,WORK)

ıbroutine to carry out one step of the classical fourth order
ınge-Kutta method for integration of a system of NEQU first order
ıfferential equations, x'=f(T,X,U)

ERNAL REFERENCES

  F=F(T,X,XP,U)        User supplied subroutine to evaluate the vector
               valued function F(T,X,U).  The value of F at T,X,U
               is to be returned in XP.


               X: the REAL*4 state vector
               Y: the REAL*4 output vector
               U: The REAL*4 input vector
               T: the current value of time,T is advanced by
                  H on each call to RK4CNT

      COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
               NEQN: integer number of plant state equations
               NIN: number of plant inputs.
               NOUT: number of output equations.
               NCEQN: number of controller equations.

      COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY

               H-delta T for the RK4 routine(real)
               TEND-ending time(real)
               TPRINT-printing interval(real)
               DELT-sampling interval (real)
               DELDST-interval between disturbances(real)
               DELAY-interval between sampling and control(real)


  ADAPTED BY L.K.LAUDERBAUGH    DEC/82

  REAL*4 U(1)
  DIMENSION X(1),XP(1),Y(1)
  DIMENSION WORK(1)
  COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
  COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY

:T UP ARRAYS TO STORE VECTORS USED IN THE COMPUTATION

  I1=1
  I2=NEQN+1
  I3=2*NEQN+1
  I4=3*NEQN+1
  ISV=4*NEQN+1

)MPUTE AND SAVE AVERAGE SLOPES TO BE USED IN THE COMPUTATION.

  CALL F(T,X,WORK(I1),U)
```

```
  T=T+H/2.
  DO 200 I=1,NEQN
  K1=ISV+I-1
  K2=I1+I-1
  WORK(K1)=X(I)+(H/2.)*WORK(K2)
  CALL F(T,WORK(ISV),WORK(I2),U)
  DO 250 I=1,NEQN
  K1=ISV+I-1
  K2=I2+I-1
  WORK(K1)=X(I)+(H/2.)*WORK(K2)
  CALL F(T,WORK(ISV),WORK(I3),U)
  T=T+H/2.
  DO 300 I=1,NEQN
  K1=ISV+I-1
  K2=I3+I-1
  WORK(K1)=X(I)+H*WORK(K2)
  CALL F(T,WORK(ISV),WORK(I4),U)

MPUTE THE NEW VALUES OF X

  DO 400 I=1,NEQN
  K1=I1+I-1
  K2=I2+I-1
  K3=I3+I-1
  K4=I4+I-1
  X(I)=X(I)+(H/6.)*(WORK(K1)+2.*WORK(K2)+2.*WORK(K3)+WORK(K4))

  DONE
  RETURN
  END
```

```
 SUBROUTINE CNVFRM

 General Description: This routine is used to input the parameters
 used in the routines that simulate the ADC and the DAC.

 Arguments:

     Common blocks:

             COMMON/INOUT/IN,IOUT
             IN: The input device number is 5 which
                 is assigned to TT
             IOUT: The output device number is 6 which
                   is assigned to TT

             COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
             ADCGAN-The gain of the analog to digital
                     converter.
             IADCOF-The offset of the analog to digital
                     converter.
             IMXADC-The maximum integer value of the output
                     of the converter.
             DACGAN-The gain of the digital to analog
                     converter.
             IDACOF-the offset of the DAC
             MINDAC,MAXDAC- The maximum and minimum values
                     of the voltage output of the DAC

 External References: CLEAR

             by Leal Lauderbaugh   Oct 83


 COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
 COMMON /INOUT/ IN,IOUT
 CALL CLEAR

 WRITE(IOUT,10) ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
 FORMAT('1',5X,'DAC ADC SIMULATION VARIABLE DISPLAY:',//,
+10X,'1) ADC gain(real)  ',F8.4,'  2) ADC offset(int) ',I8,/,
+10X,'3) max ADC (int)   ',I8,'  4) DAC gain (real)   ',F8.4,/,
+10X,'5) DAC offset (int)',I8,'  6) min DAC (real)    ',F8.4,/,
+10X,'7) max DAC (real)  ',F8.4,//)

 WRITE(IOUT,11)
 FORMAT(5X,'To change any of the variables enter the number of',/
+    ,5X,'the variable. Otherwise return to continue. '$)
 READ(IN,12,ERR=9) IFLAG
 FORMAT(I1)

 IF (IFLAG.EQ.0) RETURN

 WRITE(IOUT,8)
 FORMAT(//)
```

```
 GO TO (101,102,103,105,106,107,108) IFLAG

 FORMAT(I4)
 ASSIGN 101 TO I
 WRITE(IOUT,201)
 FORMAT(' > Enter the ADC gain (real)= '$)
 READ(IN,202,ERR=990) ADCGAN
 FORMAT(F10.4)
 GO TO 9

 ASSIGN 102 TO I
 WRITE(IOUT,203)
 FORMAT(' > Enter ADC offset (int)= '$)
 READ(IN,50,ERR=990) IADCOF
 GO TO 9

 ASSIGN 103 TO I
 WRITE(IOUT,204)
 FORMAT(' > Enter max ADC value (int)= '$)
 READ(IN,50,ERR=990) IMXADC
 GO TO 9

 ASSIGN 105 TO I
 WRITE(IOUT,206)
 FORMAT(' > Enter DAC gain (real)= '$)
 READ(IN,202,ERR=990) DACGAN
 GO TO 9

 ASSIGN 106 TO I
 WRITE(IOUT,207)
 FORMAT(' > Enter DAC offset (int)= '$)
 READ(IN,50,ERR=990) IDACOF
 GO TO 9

 ASSIGN 107 TO I
 WRITE(IOUT,208)
 FORMAT(' > Enter min DAC value (real)= '$)
 READ(IN,202,ERR=990) DACMIN
 GO TO 9

 ASSIGN 108 TO I
 WRITE(IOUT,209)
 FORMAT(' > Enter max DAC value (real)= '$)
 READ(IN,202,ERR=990) DACMAX
 GO TO 9

...Error message for illegal entry

 WRITE(IOUT,991)
 FORMAT(' ***** ERROR ***** Invalid entry, try again',/,' ')
 GO TO I

 CONTINUE
 RETURN
```

```
 SUBROUTINE SIMADC(VOLTS,IADCVL)

 General Description: This routine is used to simulate the
 behavior of the analog to digital conversion process. The analog
 voltage VOLTS is multipled by thegain of the converter, ADCGAN
 then the resulting value is converted to an integer value in the
 range 0 to IMXADC. Saturation is accounted for.

 Calling Example:CALL SIMADC(5.34,IADCVL)

 Arguments:
     Formal: VOLTS- The real value representing the voltage to the
             converter.
             IADCVL-The integer converted value

     Common blocks:

             COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
             ADCGAN-The gain of the analog to digital
                     converter.
             IADCOF-The offset of the analog to digital
                     converter.
             IMXADC-The maximum integer value of the output
                     of the converter.
             DACGAN-The gain of the digital to analog
                     converter.
             IDACOF-the offset of the DAC
             MINDAC,MAXDAC- The maximum and minimum values
                     of the voltage output of the DAC
 External References: none

             by Leal Lauderbaugh    Oct/83


mon blocks

 COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX

tiply by the gain and convert to an integer
 IADCVL=IFIX(VOLTS*ADCGAN)+IADCOF

ck for saturation.

 IF(IADCVL.GE.IMXADC)IADCVL=IMXADC
 IF(IADCVL.LE.0)IADCVL=0
 RETURN
 END
```

```
 SUBROUTINE SIMDAC(IDACIN,VOLTS)

 General Description: This routine is used to simulate the
 behavior of the digital to analog conversion process.

 Calling Example:CALL SIMDAC(3024,VOLTS)

 Arguments:
     Formal: VOLTS- The real value representing the voltage output
             of the converter.
             IDACIN-The integer value to be converted

     Common blocks:

             COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
             ADCGAN-The gain of the analog to digital
                     converter.
             IADCOF-The offset of the analog to digital
                     converter.
             IMXADC-The maximum integer value of the output
                     of the converter.
             DACGAN-The gain of the digital to analog
                     converter.
             IDACOF-the offset of the DAC
             DACMIN,DACMAX- The maximum and minimum values
                     of the voltage output of the DACC
 External References: none

             by Leal Lauderbaugh      Oct 83


mmon blocks

 COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX

vert to floating point and multiply by the gain

 VOLTS=DACGAN*FLOAT(IDACIN-IDACOF)

ck for saturation.

 IF(VOLTS.GE.DACMAX)VOLTS=DACMAX
 IF(VOLTS.LE.DACMIN)VOLTS=DACMIN
 RETURN
 END
```

```
SUBROUTINE START(X)

General Description: This routine is used to input the initial
and default values.


Arguments:
            X: the REAL*4 state vector

    Common blocks:

            COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
            NEQN: integer number of plant state equations
            NIN: number of plant inputs.
            NOUT: number of output equations.
            NCEQN: number of controller equations.

    COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY

            H-delta T for the RK4 routine(real)
            TEND-ending time(real)
            TPRINT-printing interval(real)
            DELT-sampling interval (real)
            DELDST-interval between disturbances(real)
            DELAY-interval between sampling and control(real)

            COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
            ADCGAN-The gain of the analog to digital
                    converter.
            IADCOF-The offset of the analog to digital
                    converter.
            IMXADC-The maximum integer value of the output
                    of the converter.
            DACGAN-The gain of the digital to analog
                    converter.
            IDACOF-the offset of the DAC
            MINDAC,MAXDAC- The maximum and minimum values
                    of the voltage output of the DAC


            by  Leal K Lauderbaugh


DIMENSION X(1)

imon blocks

COMMON/INOUT/IN,IOUT
COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
COMMON/CLOCK/H,TEND,TPRINT,DELT,DELDST,DELAY
COMMON/ADCDAC/ADCGAN,IADCOF,IMXADC,DACGAN,IDACOF,DACMIN,DACMAX
H=.001
TEND=2.
TPRINT=TEND/200.
```

```
 DELT=.05
 DELDST=0.
 DELAY=0.

 up defalt values for DAC and ADC

 ADCGAN=400.
 IADCOF=2048
 IMXADC=4095
 DACGAN=1./400.
 IDACOF=2048
 DACMIN=-5.12
 DACMAX=5.12

o state vector

 DO 100 I=1,NEQN
 X(I)=0.0

 RETURN
 END
```

```
 SUBROUTINE PRNOUT(T,X,Y,U,IXC,IYC,IUC)

 General Description: This subroutine is a driver routine used to
     selsct and display the simulation results.

 Calling Example:

 Arguments:
     Formal:

     Common blocks:

             COMMON/INOUT/IN,IOUT
             IN: The input device number is 5 which
                 is assigned to TT
             IOUT: The output device number is 6 which
                   is assigned to TT

 External References:

             by Leal Lauderbaugh


larations
 LOGICAL*1 ANSW,NO,YES
 REAL*4 U(1)
 DIMENSION X(1),Y(1),IXC(1),IYC(1),IUC(1)
 DIMENSION PRNT(201,11),IND(10)
 DIMENSION DV(201),RIDV(201)
 DATA NO/'N'/,YES/'Y'/

mon blocks

 COMMON/INOUT/IN,IOUT
 COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
 COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV
 COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN

ect the independent and the dependent variables.

 NREC=201
 CALL GRSELC(IND)

d the results and set up print vector

 CALL READRC(NREC,X,Y,U,T,IXC,IYC,IUC,IND,PRNT)

ck for a print
 WRITE(IOUT,100)
 FORMAT(/,' Do you wish to print the results?')
 READ(IN,92)ANSW
 FORMAT(A1)
 IF(ANSW.EQ.YES)GO TO 200
 IF(ANSW.NE.NO)GO TO 90
```

```
ck for a plot

 WRITE(IOUT,130)
 FORMAT(/,' Do you wish to plot the results?')
 READ(IN,92)ANSW
 IF(ANSW.EQ.YES)GO TO 390
 IF(ANSW.NE.NO)GO TO 120
 GOTO 1000
 CALL ASSIGN(2,'LP:')
 WRITE(2,210)
 FORMAT('     INDP.',7X,'DEP 1',7X,'DEP 2',7X,'DEP 3',7X,
+  'DEP 4',7X,'DEP 5',7X,'DEP 6',7X,'DEP 7',7X,'DEP 8',7X,
+  'DEP 9',7X,'DEP 10',//)
 DO 250 IP=1,NREC
 WRITE(2,270)PRNT(IP,11),(PRNT(IP,IK),IK=1,ICNT)
 FORMAT(11(1X,G11.4))
 GOTO 120
ar the graphics screen
 CALL PAGE
 WRITE(IOUT,410)
 FORMAT(/,' Enter the index of the dependent variable 1-10 ')
 READ(IN,450)IPLOT
 FORMAT(I2)
 up DV and RIDV
 DO 470 I=1,NREC
 DV(I)=PRNT(I,IPLOT)
 RIDV(I)=PRNT(I,11)

d the max and min

 CALL MAXMIN(NREC,DV,RIDV)

d the axis type

 CALL AXTYPE(IQUAD)

d the scale factors

 CALL SCALE(IQUAD,DVSCL,IDVOFF,RIDVSC,IIDVOF)
 WRITE(IOUT,500)DVMAX,DVMIN,RIDVMX,RIDVMN
 FORMAT(/,' The max plotted value of the dep. variable is'
+  ,G11.4,/,
+  ' The min plotted value of the dep. variable is ',G11.4,/,
+  ' The max plotted value of the indep. variable is',G11.4,/,
+  ' The min plotted value of the indep. variable is ',G11.4,/,
+  ,/,' Do you wish to make any changes? (y/n)')
 READ(IN,92)ANSW
 IF(ANSW.EQ.YES)GO TO 510
 IF(ANSW.NE.NO)GO TO 495
 GOTO 600
 WRITE(IOUT,520)
 FORMAT(' Input the new DVMAX,DVMIN,RIDVMX,RIDVMIN')
 READ(IN,530)DVMAX,DVMIN,RIDVMX,RIDVMN
```

```
 FORMAT(4(G10.4))
 GOTO 490

cK for hard copy

 WRITE(IOUT,610)
 FORMAT(/,' Would you like this plot to be a hard copy? ',/,
+  ' If no, the plot will be directed to the screen.')
 READ(IN,92)ANSW
 IF(ANSW.EQ.YES)GO TO 700
 IF(ANSW.NE.NO)GO TO 600

t on the screen

t axis set
 CALL AXES(IQUAD)
e to the first point
 IX=IFIX(RIDV(1)*RIDVSC)+IIDVOF
 IY=IFIX(DV(1)*DVSCL)+IDVOFF
 CALL MOVE(IX,IY)
le and plot data
 DO 650 I=2,NREC
 IX=IFIX(RIDV(I)*RIDVSC)+IIDVOF
 IY=IFIX(DV(I)*DVSCL)+IDVOFF
 CALL DRAW(IX,IY)
 GOTO 800

t on the plotter

tialize plotter
 CALL PLTINT
t axis set
 CALL PAXES(IQUAD)
e to the first point
 IX=IFIX(RIDV(1)*RIDVSC)+IIDVOF
 IY=IFIX(DV(1)*DVSCL)+IDVOFF
 CALL PMOVER(IX,IY)
 e and plot data
 DO 750 I=2,NREC
 IX=IFIX(RIDV(I)*RIDVSC)+IIDVOF
 IY=IFIX(DV(I)*DVSCL)+IDVOFF
 CALL PLOTER(IX,IY)
 CALL PMOVER(0,0)
 WRITE(IOUT,810)
 FORMAT(/,' Would you like to do more plotting? (y/n)')
 READ(IN,92)ANSW
 IF(ANSW.EQ.YES)GO TO 830
 IF(ANSW.NE.NO)GO TO 800
 GOTO 1000
 WRITE(IOUT,820)
 FORMAT(/,' Would you like to erase the screen? (y/n)')
 READ(IN,92)ANSW
 IF(ANSW.EQ.YES)GOTO 850
 IF(ANSW.NE.NO)GO TO 830
```

```
GOTO 400
CALL PAGE
GOTO 400
RETURN
END
```

```
 SUBROUTINE GRSELC(IND)

 General Description: This routine is used to select and read the
     variables for printing and graphics.

 Calling Example:

 Arguments:
     Formal:

     Common blocks:

             COMMON/INOUT/IN,IOUT
             IN: The input device number is 5 which
                 is assigned to TT
             IOUT: The output device number is 6 which
                   is assigned to TT

             COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV
             INCT: the number of independent variables selected
             IX: number of plant states selected
             IY: number of plant outputs selected
             IU: number of plant inputs selected
             INXC: number of controller states selected
             INYC: number of controller inputs selected
             INUC: number of controller outputs selected
             INDV: The indep. variable
                     0=time
                     I=X(I) I not = 0
             IND: index vector

 External References: CLEAR

             by Leal Lauderbaugh


larations

 LOGICAL*1 ANSW,NO,YES
 DIMENSION IND(10)

mon blocks

 COMMON/INOUT/IN,IOUT
 COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
 COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV

a statments

 DATA NO/'N'/,YES/'Y'/
ign devices

ndex of the index vector, IND.  ICNT=number of dep variables selected
 I=1
```

```
 ICNT=0
ber of each type of variable
 IX=0
 IY=0
 IU=0
 INXC=0
 INYC=0
 INUC=0
 CALL CLEAR
 WRITE(IOUT,30)
 FORMAT(20X,' PRINTING AND GRAPHICS',/,/,
+  ' You will be asked to select the independent variable, and up',
+  /,' to ten dependent variables. The dependent variables may',/,
+  /,' be selected from X,Y,U,XC,YC,UC. ',/,/)
 WRITE(IOUT,40)
 FORMAT(' Select the independent variable. If you want time to',/,
+  ' be the independent variable then enter 0. To select a plant',/,
+  ' state enter the index.')
 READ(IN,130)INDV
 ILG=NEQN
 IRET=35
 IF(INDV.GT.NEQN.OR.INDV.LT.0)GOTO 700
 WRITE(IOUT,120)
 FORMAT(' Do you wish to plot plant states (X)? ',/,
+    ' If yes, enter the index. If no, enter 0 '$)
 FORMAT(' ')
 READ(IN,130)IANS
 FORMAT(I2)
 ILG=NEQN
 IRET=115
 IF(IANS.GT.NEQN.OR.IANS.LT.0)GOTO 700
 WRITE(IOUT,125)
 IF(IANS.EQ.0)GOTO 215
 IND(I)=IANS
 I=I+1
 IX=IX+1
 ICNT=ICNT+1
 IF(ICNT.EQ.10)GOTO 1000
 GOTO 115
 WRITE(IOUT,220)
 FORMAT(' Do you wish to plot plant outputs (Y)? ',/,
+    ' If yes, enter the index. If no, enter 0 '$)
 READ(IN,130)IANS
 ILG=NOUT
 IRET=215
 IF(IANS.GT.ILG.OR.IANS.LT.0)GOTO 700
 WRITE(IOUT,125)
 IF(IANS.EQ.0)GOTO 315
 IND(I)=IANS
 I=I+1
 IY=IY+1
 ICNT=ICNT+1
 IF(ICNT.EQ.10)GOTO 1000
 GOTO 215
```

```
 WRITE(IOUT,320)
 FORMAT(' Do you wish to plot plant inputs (U)? ',/,
+     ' If yes, enter the index. If no, enter 0 '$)
 READ(IN,130)IANS
 ILG=NIN
 IRET=315
 IF(IANS.GT.ILG.OR.IANS.LT.0)GOTO 700
 WRITE(IOUT,125)
 WRITE(IOUT,125)
 IF(IANS.EQ.0)GOTO 415
 IND(I)=IANS
 I=I+1
 IU=IU+1
 ICNT=ICNT+1
 IF(ICNT.EQ.10)GOTO 1000
 GOTO 315
 WRITE(IOUT,420)
 FORMAT(' Do you wish to plot controller states (XC)? ',/,
+     ' If yes, enter the index. If no, enter 0 '$)
 READ(IN,130)IANS
 ILG=NCEQN
 IRET=415
 IF(IANS.GT.ILG.OR.IANS.LT.0)GOTO 700
 WRITE(IOUT,125)
 WRITE(IOUT,125)
 IF(IANS.EQ.0)GOTO 515
 IND(I)=IANS
 I=I+1
 INXC=INXC+1
 ICNT=ICNT+1
 IF(ICNT.EQ.10)GOTO 1000
 GOTO 415
 WRITE(IOUT,520)
 FORMAT(' Do you wish to plot controller inputs (YC)? ',/,
+     ' If yes, enter the index. If no, enter 0 '$)
 READ(IN,130)IANS
 ILG=NOUT
 IRET=515
 IF(IANS.GT.ILG.OR.IANS.LT.0)GOTO 700
 WRITE(IOUT,125)
 WRITE(IOUT,125)
 IF(IANS.EQ.0)GOTO 615
 IND(I)=IANS
 I=I+1
 INYC=INYC+1
 ICNT=ICNT+1
 IF(ICNT.EQ.10)GOTO 1000
 GOTO 515
 WRITE(IOUT,620)
 FORMAT(' Do you wish to plot controller outputs (UC)? ',/,
+     ' If yes, enter the index. If no, enter 0 '$)
 READ(IN,130)IANS
 ILG=NIN
 IRET=615
```

```
 IF(IANS.GT.ILG.OR.IANS.LT.0)GOTO 700
 WRITE(IOUT,125)
 WRITE(IOUT,125)
 IF(IANS.EQ.0)GOTO 1020
 IND(I)=IANS
 I=I+1
 INUC=INUC+1
 ICNT=ICNT+1
 IF(ICNT.EQ.10)GOTO 1000
 GOTO 615
 WRITE(IOUT,750)ILG
 FORMAT(10X,'***** ERROR *****',/,/,
+  ' The index entered was out of range. The largest alowable'
+  ,/,' index for this variable is ',I2,'. Please re-enter.')
 GOTO IRET
 WRITE(IOUT,1010)
 FORMAT(/,' Ten dependent variables have been selected.',/,
+  ' If you wish to display more than these ten variables',/,
+  ' you will have to make another run of the entire program.')
 GOTO 1030
 WRITE(IOUT,1025)
 FORMAT(//,' Variable selection complete.')
 RETURN
 END
```

```
 SUBROUTINE READRC(NREC,X,Y,U,T,IXC,IYC,IUC,IND,PRNT)

 General Description: This routine is used to read NREC from
     the disk and set up a print matrix.

 Calling Example:

 Arguments:
     Formal:

     Common blocks:

             COMMON/INOUT/IN,IOUT
             IN: The input device number is 5 which
                 is assigned to TT
             IOUT: The output device number is 6 which
                   is assigned to TT

             COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV,IND
             INCT: the number of independent variables selected
             IX: number of plant states selected
             IY: number of plant outputs selected
             IU: number of plant inputs selected
             INXC: number of controller states selected
             INYC: number of controller inputs selected
             INUC: number of controller outputs selected
             INDV: The indep. variable
                     0=time
                     I=X(I) I not = 0
 External References:

             by Leal Lauderbaugh



clarations

  LOGICAL*1 ANSW,NO,YES
  REAL*4 U(1)
  DIMENSION  X(1),Y(1),IXC(1),IYC(1),IUC(1)
  DIMENSION PRNT(201,11),IND(10)

mmon blocks

  COMMON/INOUT/IN,IOUT
  COMMON/SIZE/NEQN,NIN,NOUT,NCEQN
  COMMON/INDEX/ICNT,IX,IY,IU,INXC,INYC,INUC,INDV

  REWIND 3
  REWIND 4

op to read the records

  DO 1000 I1=1,NREC
```

```
  IOFF=0
  WRITE(IOUT,11)I1,NREC
  FORMAT(' I1=',I4,' NREC=',I4)
ad a record
  READ(3)T,(X(J),J=1,NEQN),(U(J),J=1,NIN),(Y(J),J=1,NOUT)
  WRITE(IOUT,3)T,(X(J),J=1,NEQN),(U(J),J=1,NIN),(Y(J),J=1,NOUT)
  READ(4)(IXC(J),J=1,NCEQN),(IUC(J),J=1,NIN),(IYC(J),J=1,NOUT)
  WRITE(IOUT,4)(IXC(J),J=1,NCEQN),(IUC(J),J=1,NIN),(IYC(J),J=1,NOUT)
  FORMAT(8(2X,F4.0))
  FORMAT(8(2X,I4))
sign independent variables
  IF(INDV.EQ.0)PRNT(I1,11)=T
  IF(INDV.NE.0)PRNT(I1,11)=X(INDV)

sign the indep. variables

  IF(IX.EQ.0)GOTO 200
  DO 110 I2=1,IX
  I3=I2+IOFF
  WRITE(IOUT,120)I2,I3,IOFF,IND(I3)
0 FORMAT(' 100 ',4(2X,I2))
  PRNT(I1,I3)=X(IND(I3))
  IOFF=IOFF+IX

  IF(IY.EQ.0)GOTO 300
  DO 210 I2=1,IY
  I3=I2+IOFF
  WRITE(IOUT,220)I2,I3,IOFF,IND(I3)
0 FORMAT(' 200 ',4(2X,I2))
  PRNT(I1,I3)=Y(IND(I3))
  IOFF=IOFF+IY

  IF(IU.EQ.0)GOTO 400
  DO 310 I2=1,IU
  I3=I2+IOFF
  WRITE(IOUT,320)I2,I3,IOFF,IND(I3)
0 FORMAT(' 300 ',4(2X,I2))
  PRNT(I1,I3)=U(IND(I3))
  IOFF=IOFF+IU

  IF(INXC.EQ.0)GOTO 500
  DO 410 I2=1,INXC
  I3=I2+IOFF
  WRITE(IOUT,420)I2,I3,IOFF,IND(I3)
0 FORMAT(' 400 ',4(2X,I2))
  PRNT(I1,I3)=IXC(IND(I3))
  IOFF=IOFF+INXC

  IF(INYC.EQ.0)GOTO 600
  DO 510 I2=1,INYC
  I3=I2+IOFF
  WRITE(IOUT,520)I2,I3,IOFF,IND(I3)
0 FORMAT(' 500 ',4(2X,I2))
  PRNT(I1,I3)=IYC(IND(I3))
```

```
  IOFF=IOFF+INYC

  IF(INUC.EQ.0)GOTO 1000
  DO 610 I2=1,INUC
  I3=I2+IOFF
  WRITE(IOUT,620)I2,I3,IOFF,IND(I3)
0 FORMAT(' 600 ',4(2X,I2))
  PRNT(I1,I3)=IUC(IND(I3))

0 CONTINUE
  RETURN
  END
```

APPENDIX D

GRAPHICS LIBRARY DOCUMENTATION

CHAPTER D-1

General Description

This document describes the general use of the graphics language LEAL PLOT. This language is intended to be easy to use without becoming to far removed from the actual hardware. All the graphics instructions is this language use screen coordinates. These are integer values ranging from 0 to 1023 in the X direction, and 0 to 780 in the Y direction. This restriction requires the user to scale and offset vectors to the proper range. This scaling requires some additional work and code, but provides more precise control over the drawing process. The routine MAXMIN, SCALE, AXTYPE may be used for scaling if the data is to be plotted using the AXES routines.

The screen coordinates are set up with the origin at the lower left corner. The X axis is horizontal with a range of 1023. The Y axis is vertical with a range of 780. The center of the screen is located at 390,512.

The language also contains commands for plotting on an X-Y recorder. The coordinates for these routines are identical to the screen coordinates.

The language resides in the object library LEALPT.OBJ. To use these routines, link LEALPT.OBJ with your FORTRAN program. The graphics operations are initiated by a subroutine call to the appropriate routine. For example, to draw to the center of the screen from the current position:

```
IXCSRN=512
IYSCRN=390
CALL DRAW(IXSCRN,IYSCRN).
```

OR

```
CALL DRAW(512,390)
```

The software is developed on three levels. The first level is hardware dependent and is used to control the graphics terminal and the X-Y plotter. Level I comprises three subroutines, INITSL, and SEND are both assembly language subroutines used to send characters to the terminal. The third routine in Level I is a digital to analog conversion routine, used in hard copy routines.

The second level comprises subroutines used to move and draw on the terminal and the plotter. These routines can be thought of as driver routines for the Level I routines. Also included at this level are routines for switching from alpha to graphics mode.

The third level of routines are the routines that make the user's job easier. These include, routines for drawing axes, magnitude scaling, etc.

CHAPTER D-2

INSTALLATION NOTES

This software has been developed to run on an LSI-11/23 with a Tektronix 4006-1 graphics terminal. It is assumed that the serial line unit MXV-11-A, or a compatable unit, is installed at location 176510. This can be changed by changing the address in the assembly language routines SEND and INITSL. It is further assumed that an AAV-11-A digital to analog converter is installed at location 170440 and jumpered to produce +/-5.12 volts output Channel 0 should be connected to the X input and channel 1 connected to the Y input of the recorder. Both channels are to be set to .5 volts/inch.

CHAPTER D-3

LEVEL I ROUTINES

The Level I routines are never directly called by the user. These are the routines that communicate with the actual devices, thus, these are the routines that will require modification if the language is moved to another system. The three assembly language routines used are:

1. INITSL: This routine initializes the serial I/O unit.
2. SEND: This routine sends out characters to the 4006 terminal.
3. DAC: This routine performs digital to analog conversions.

CHAPTER D-4

LEVEL II ROUTINES

These routines are used as drivers for the Level I routines. These routines may be called by the user directly or by other Level II routines and by Level III routines. The Level II routines are:

1. PLOTER(IXSCRN,IYSCRN): This is a routine to draw on an X Y recorder to a position corresponding to IXSCRN,IYSCRN.

   NOTE

   Caution should be used when using this routine to make sure that IYSCRN=780. Numbers larger than 780 will be off scale.

2. PMOVER(IXSCRN,IYSCRN): This routine rapidly moves the pen on The X-Y plotter to (IXSCNR,IYSCRN).
3. PLTINT: This routine is used to initialize the plotter. It puts the pen at 0,0, and initializes the location vectors.
4. AMODE: This routine puts the terminal into the alpha mode.
5. GMODE: This routine puts the terminal into the graphics mode.
6. DRAW(IXSCRN,IYSCRN): This subroutine draws on the terminal to the screen location IXSCRN,IYSCRN.
7. MOVE(IXSCRN,IYSCRN): This routine moves the cursor to the screen location, (IXSCRN,IYSCRN).
8. BELL: This routine enables the tone from the speaker.
9. PAGE: This routine erases the screen, puts the terminal in the alpha mode and places the cursor in the home position.
10. WAIT(ITIME): This routine is used by several routines to provide delays.

CHAPTER D-5

LEVEL III ROUTINES

These routines are high level fortran routines that perform complex combinations of the basic graphics functions. These will eventually include routines for relative graphics, etc. The routines currently in the language are:

1. AXES(IQUAD): this routine draws a set of coordinate axes in the screen window (125-825),(125-625) and leaves the cursor at the origin. The routine plots one of the following:

   IQUAD=1 quadrant I

   IQUAD=2 quadrant I and IV

   IQUAD=3 all

2. PAXES: The same as AXES only plotted on the X-Y plotter.

3. MAXMIN(N,DV,RIDV): This routine finds the maximum of the dependent and the independent variables.

   N: number of arguments

   DV(N): The array containing the dependent variable.

   RIDV(N): The array containing the independent variable. RIDV and DV must have the same dimension.

   NOTE

   This routine and the following two (AXTYPE,SCALE) use the common block MXMN. This common block must appear in the calling routine.

   COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN

   All arguments are REAL*4

   These common block arguments are returned with the values of the max and min of the variable arrays.

4. AXTYPE(IQUAD): This routine uses the values returned

from MAXMIN to select the axis type IQUAD. This routine uses the common block MXMN.

NOTE

This routine should be called after calling MAXMIN or after artifically setting the max and min.

5. SCALE(IQUAD,DVSCL,IDVOFF,RIDVSC,IIDVOF): This routine calculates the scale factors and offsets. These calculations are based on the value of the arguments in the common block MXMN and on the value of IQUAD.

IQUAD: See AXES

DVSCL: scale factor for the dependent variable

IDVOFF: The offset of the axes for the dep. variable

RIDVSC: scale factor for the independent variable

IIDVOF: The offset of the axes for the independent variable.

To find the screen coordinates for plotting data the following equation can be used,

IY=IFIX(DVSCL*DV(I))+IDVOFF

and

IX=IFIX(RIDVSC*RIDVC(I)+IIDVOF

```
SUBROUTINE GMODE

General Description: This routine is used to put the terminal into the
    graphics mode. Note, the first vector sent to the terminal
    after the call to GMODE will be a move i.e. a dark vector.

External References:INITSL
                SEND

              by Leal Lauderbaugh     9/83

CALL INITSL
IA=29
CALL SEND(IA)
RETURN
END
```

```
 SUBROUTINE DRAW(IXSCRN,IYSCRN)

 General Description: This routine is used to translate screen
     coordinates into decimal equivalent of ASCII character to be
     sent to the terminal.  The routine also sends the character.
     While the routine is called DRAW it is also used in the move
     command.  The first call to DRAW after calling GMODE will be
     a move i.e. a dark vector, otherwise the vector will be drawn.

 Calling Example: CALL DRAW(IXSCRN,IYSCRN)

 Arguments:
     Formal:IXSCRN-INTEGER*2 X value in screen coordinates.
            IYSCRN-   "       Y    "    "    "          "

     Common blocks:
         COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
             IXSCUR,IYSCUR-X,Y screen position.
             IXPCUR,IYSCUR-X,Y plotter position.
             IXDCUR,IYDCUR-X,Y DAC values.

 External References:SEND
                  WAIT

              by Leal Lauderbaugh       9/83

 COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR

hd high bytes

 ISUMX=IXSCRN/32
 ISUMY=IYSCRN/32
 IHIX=32+ISUMX
 IHIY=32+ISUMY

hd low bytes
 ILOX=IXSCRN-32*ISUMX+64
 ILOY=IYSCRN-32*ISUMY+96
 CALL SEND(IHIY)
 CALL SEND(ILOY)
 CALL SEND(IHIX)
 CALL SEND(ILOX)
 ITIME=10
 CALL WAIT(ITIME)
 IXSCUR=IXSCRN
 IYSCUR=IYSCRN
 RETURN
 END
```

```
SUBROUTINE MOVE(IXSCRN,IYSCRN)

General Description: This routine is used to move the cursor to
    location IXSCRN,IYSCRN.

Arguments:
    Formal:IXSCRN,IYSCRN-the integer screen locations.

External References: GMODE
                     DRAW

            by Leal Lauderbaugh      9/83

CALL GMODE
CALL DRAW(IXSCRN,IYSCRN)
RETURN
END
```

```
SUBROUTINE BELL

General Description: This routine is used to enable the tone from the
    speaker.

External References:INITSL
                SEND

            by Leal Lauderbaugh     9/83

CALL INITSL
IA=7
CALL SEND(IA)
RETURN
END
```

```
SUBROUTINE PAGE

General Description: This routine is used to erase the screen, retun
    the terminal to alpha mode, and place the curser in the home
    position.

    Common blocks:
        COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
            IXSCUR,IYSCUR-X,Y screen position.
            IXPCUR,IYSCUR-X,Y plotter position.
            IXDCUR,IYDCUR-X,Y DAC values.

External References:INITSL
                SEND
                WAIT

            by Leal Lauderbaugh     9/83

COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
CALL INITSL
IA=27
IB=12
CALL SEND(IA)
CALL SEND(IB)
ITIME=250
CALL WAIT(ITIME)
IXSCUR=0
IYSCUR=0
RETURN
END
```

```
SUBROUTINE AMODE

General Description: This routine is used to put the terminal into the
    ALPHA mode.

External References:INITSL
                    SEND

            by Leal Lauderbaugh     9/83

CALL INITSL
IA=31
CALL SEND(IA)
RETURN
END
```

```
SUBROUTINE WAIT(ITIME)

General Description: This routine is used to provide delays.  The
    system is desined assuming a baud rate of 2400.  Higher
    baud rates require the addition to a few more equations to
    increase the delay.

Arguments:
    Formal:ITIME- integer number of loops

External References: none

            by Leal Lauderbaugh    9/83

DO 100 I=1,ITIME
WAIT=(5.3/ITIME)*SIN(.355)
RETURN
END
```

```
  SUBROUTINE PLOTER(IXSCRN,IYSCRN)

  General Description: This routine is used to draw on a X,Y
      recorder to a position corresponding to IXSCRN,IYSCRN.  The
      subroutine assumes that the DAC #0 is connected to the x axis
      and that #1 is connected to the Y.  both channels are to be
      set to .5 volts/in.

  Calling Example:CALL PLOTER(IXSCRN,IYSCRN)

  Arguments:
      Formal:IXSCRN,IYSCRN are the screen coordinates to move to.

      Common blocks:
          COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
              IXSCUR,IYSCUR-X,Y screen position.
              IXPCUR,IYSCUR-X,Y plotter position.
              IXDCUR,IYDCUR-X,Y DAC values.

  External References:DAC

              by Leal Lauderbaugh     9/83

  COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
  XFACT=1.7
  YFACT=XFACT
  ITIME=2
  IOFF=2048
  IXCHAN=0
  IYCHAN=1
  IDELX=1
  IDELY=1
nvert values to DAC values
  IXDAC=(IXSCRN*XFACT)+IOFF
  IYDAC=(IYSCRN*YFACT)+IOFF
nd incrememts to move.
  IXINC=IXDAC-IXDCUR
  IYINC=IYDAC-IYDCUR
  IF(IXINC.LT.0)IDELX=-1
  IF(IYINC.LT.0)IDELY=-1
  IXINC=IABS(IXINC)
  IYINC=IABS(IYINC)
  IXMOVE=IXINC
  IYMOVE=IYINC
  WRITE(5,15)IXDAC,IXMOVE,IDELX,IXDCUR
  WRITE(5,15)IYDAC,IYMOVE,IDELY,IYDCUR
5 FORMAT(4(' ',I8))
  READ(5,22)ID
  FORMAT(I1)
rst move in largest direction or X if they are equal
  IF(IXINC.GE.IYINC)GOTO 100
  IYDCUR=IYDCUR+IDELY
  CALL DAC(IYCHAN,IYDCUR)
  CALL WAIT(ITIME)
```

```
  IYINC=IYINC-1
  GOTO 150
eck for no move
  IF(IXINC.EQ.0)GOTO 500
  IXDCUR=IXDCUR+IDELX
  CALL DAC(IXCHAN,IXDCUR)
  CALL WAIT(ITIME)
  IXINC=IXINC-1
  WRITE(5,15)IXDCUR,IXINC,IYDCUR,IYINC
  READ(6,22)ID

MOVE LOOP

eck to see if there are moves remainig
  IF(IXINC.EQ.0.AND.IYINC.EQ.0)GOTO 500
nd % moves remaining
  IF(IXMOVE.EQ.0)GOTO 160
  XPERCT=FLOAT(IXINC)/FLOAT(IXMOVE)
  IF(IYMOVE.EQ.0)GOTO 170
  YPERCT=FLOAT(IYINC)/FLOAT(IYMOVE)
  GOTO 180
  XPERCT=0.
  GOTO 155
  YPERCT=0.
eck for X and Y move
  IF(XPERCT.NE.YPERCT)GOTO 200
ve X & Y
  IXDCUR=IXDCUR+IDELX
  IYDCUR=IYDCUR+IDELY
  CALL DAC(IXCHAN,IXDCUR)
  CALL DAC(IYCHAN,IYDCUR)
  CALL WAIT(ITIME)
date moves remaining
  IXINC=IXINC-1
  IYINC=IYINC-1
  WRITE(5,15)IXDCUR,IXINC,IYDCUR,IYINC
  GOTO 150

eck for X or Y move

  IF(XPERCT.LT.YPERCT)GOTO 250
move
  IXDCUR=IXDCUR+IDELX
  CALL DAC(IXCHAN,IXDCUR)
  CALL WAIT(ITIME)
  IXINC=IXINC-1
  WRITE(5,15)IXDCUR,IXINC,IYDCUR,IYINC
  GOTO 150
move
  IYDCUR=IYDCUR+IDELY
  CALL DAC(IYCHAN,IYDCUR)
  CALL WAIT(ITIME)
  IYINC=IYINC-1
  WRITE(5,15)IXDCUR,IXINC,IYDCUR,IYINC
```

```
  GOTO 150
date position
  IXPCUR=IXSCRN
  IYPCUR=IYSCRN
  RETURN
  END
```

```
 SUBROUTINE PMOVER(IXSCRN,IYSCRN)

 General Description: This routine is used to move the pen on a X,Y
     recorder to a position corresponding to IXSCRN,IYSCRN.  The
     subroutine assumes that the DAC #0 is connected to the x axis
     and that #1 is connected to the Y.  both channels are to be
     set to .5 volts/in.

 Calling Example:CALL PMOVER(IXSCRN,IYSCRN)

 Arguments:
     Formal:IXSCRN,IYSCRN are the screen coordinates to move to.

     Common:
         COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
             IXSCUR,IYSCUR-X,Y screen position.
             IXPCUR,IYSCUR-X,Y plotter position.
             IXDCUR,IYDCUR-X,Y DAC values.

 External References:DAC
                  WAIT

             by Leal Lauderbaugh     9/83

  COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
  COMMON/INOUT/IN,IOUT
  XFACT=1.7
  YFACT=XFACT
  ITIME=200
  IOFF=2048
  IXCHAN=0
  IYCHAN=1
nvert values to DAC values
  IXDAC=(IXSCRN*XFACT)+IOFF
  IYDAC=(IYSCRN*YFACT)+IOFF
nd out values
  WRITE(IOUT,10)
  READ(IN,20)IDUMMY
  FORMAT(' RAISE PEN, RETURN TO CONT.')
  FORMAT(I1)
  CALL DAC(IXCHAN,IXDAC)
  CALL DAC(IYCHAN,IYDAC)
  WRITE(IOUT,25)
  FORMAT(' LOWER PEN, RETURN TO CONT')
  READ(IN,20)IDUMMY
date position
  IXDCUR=IXDAC
  IYDCUR=IYDAC
  IXPCUR=IXSCRN
  IYPCUR=IYSCRN
  RETURN
  END
```

```
 SUBROUTINE PLTINT

 General Description: This routine is used to initialize the plotter

 Calling Example:CALL PLTINT

 Arguments:

     Common blocks:
         COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
             IXSCUR,IYSCUR-X,Y screen position.
             IXPCUR,IYSCUR-X,Y plotter position.
             IXDCUR,IYDCUR-X,Y DAC values.

 External References: DAC

             by Leal Lauderbaugh    9/83

 COMMON/INOUT/IN,IOUT
 COMMON/SPOT/IXSCUR,IYSCUR,IXPCUR,IYPCUR,IXDCUR,IYDCUR
 IXCHAN=0
 IYCHAN=1
 IOFF=2048
 CALL DAC(IXCHAN,IOFF)
 CALL DAC(IYCHAN,IOFF)
 WRITE(IOUT,10)
 FORMAT(' TURN ON THE X-Y RECORDER'
+  ,/,' SET X AND Y AXES TO .5V/IN'
+  ,/,' ZERO X AND Y AXES - RETURN TO CONT.')
 READ(IN,20)IDUMMY
 FORMAT(I1)
 IXPCUR=0
 IYPCUR=0
 IXDCUR=IOFF
 IYDCUR=IOFF
 RETURN
 END
```

```
  SUBROUTINE AXES(IQUAD)

  General Description: This routine is used to draw the axes for
      quadrants I,I&IV or all four.  The axes are plotted at
      screen coordinates (125-825),(125-625).  The routine leaves
      the cursor at the zero of the axes drawn.

  Calling Example: CALL AXES(IQUAD)

  Arguments:
      Formal: IQUAD designation of axes type
              1-quadrant I
              2-quadrants I&IV
              3-all

      Common blocks:

              COMMON/INOUT/IN,IOUT
              IN: The input device number is 5 which
                  is assigned to TT
              IOUT: The output device number is 6 which
                    is assigned to TT

  External References: LEAL PLOT

              by Leal Lauderbaugh    7/83


mmon blocks

  COMMON/INOUT/IN,IOUT

eck which quadrants are required
  ITEST=IQUAD-2
  IF(ITEST)1000,2000,3000

ot axies for quadrant 1

0 CALL MOVE(125,625)
  CALL DRAW(125,125)
  CALL DRAW(825,125)
  CALL MOVE(125,125)
  GOTO 2500

aw axies for all four quadrents

0     CALL MOVE(125,375)
  CALL DRAW(825,375)
  CALL MOVE(475,625)
  CALL DRAW(475,125)
  CALL MOVE(475,375)
  GOTO 2500

aw axies for quadrents I&IV
```

```
0 CALL MOVE(125,625)
  CALL DRAW(125,125)
  CALL MOVE(825,375)
  CALL DRAW(125,375)
0 RETURN
  END
```

```
  SUBROUTINE PAXES(IQUAD)

  General Description: This routine is used to plot the axes for
      quadrants I,I&IV or all four.  The axes are plotted at
      screen coordinates (125-825),(125-625).  The routine leaves
      the cursor at the zero of the axes drawn.

  Calling Example: CALL PAXES(IQUAD)

  Arguments:
      Formal: IQUAD designation of axes type
              1-quadrant I
              2-quadrants I&IV
              3-all

      Common blocks:

              COMMON/INOUT/IN,IOUT
              IN: The input device number is 5 which
                  is assigned to TT
              IOUT: The output device number is 6 which
                    is assigned to TT

  External References: LEAL PLOT

              by Leal Lauderbaugh    7/83


mmon blocks

  COMMON/INOUT/IN,IOUT

eck which quadrants are required
  ITEST=IQUAD-2
  IF(ITEST)1000,2000,3000

ot axies for quadrant 1

0 CALL PMOVER(125,625)
  CALL PLOTER(125,125)
  CALL PLOTER(825,125)
  CALL PMOVER(125,125)
  GOTO 2500

aw axies for all four quadrents

)     CALL PMOVER(125,375)
  CALL PLOTER(825,375)
  CALL PMOVER(475,625)
  CALL PLOTER(475,125)
  CALL PMOVER(475,375)
  GOTO 2500

aw axies for quadrents I&IV
```

```
) CALL PMOVER(125,625)
  CALL PLOTER(125,125)
  CALL PMOVER(825,375)
  CALL PLOTER(125,375)
) RETURN
  END
```

```
  SUBROUTINE MAXMIN(N,DV,RIDV)

  General Description: This subroutine is used to find the maximum and
      the minimum values of the two arrays DV and RIDV.  This
      information can then be used for scaling plots of the data in
      the arrays.

  Calling Example:DIMENSION DV(5),RIDV(5)
                  CALL MAXMIN(N,DV,RIDV)

  Arguments:
                  N= # of points in the arrays (both arrays must
                     have the same number of points.
                  DV,RIDV contain the data points.

                  COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN
                         arguments contain the maximum and minimum
                         values of the  arrays.


  External References: none

                  by Leal Lauderbaugh    7/83


  DIMENSION DV(1),RIDV(1)

mmon blocks

  COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN


t initial values to the first elements in the array.
  RIDVMX=RIDV(1)
  RIDVMN=RIDV(1)
  DVMAX=DV(1)
  DVMIN=DV(1)

op to find the max and min

  DO 100 I=2,N
  IF(RIDV(I).GT.RIDVMX) RIDVMX=RIDV(I)
  IF(RIDV(I).LT.RIDVMN) RIDVMN=RIDV(I)
  IF(DV(I).GT.DVMAX) DVMAX=DV(I)
  IF(DV(I).LT.DVMIN) DVMIN=DV(I)
  CONTINUE
  RETURN
  END
```

```
  SUBROUTINE AXTYPE(IQUAD)

  General Description: This routine is used to determine the axes
      type for plotting.

  Arguments:
      Formal:IQUAD
              =1: QUADRANT I
              =2: QUADRANT I&IV
              =3: ALL

      Common blocks:

              COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN
                      arguments contain the maximum and minimum
                      values of the  arrays.

  External References:

              by Leal Lauderbaugh


clarations

mmon blocks

  COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN

eck for all four

  IF(RIDVMN.GE.0)GOTO 100
1 four quadrants
  IQUAD=3
  GOTO 500

eck for I&IV

  IF(DVMIN.GE.0)GOTO 200
adrants I&IV
  IQUAD=2
  GOTO 500
adrant I
  IQUAD=1
  RETURN
  END
```

```
      V02.1-10   Wed 21-Mar-84 00:30:54              PAGE 001

  SUBROUTINE SCALE(IQUAD,DVSCL,IDVOFF,RIDVSC,IIDVOF)

  General Description: This routine determines the scale factors and
      offsets. For internal calcualtions X is the indep. variable
      and Y is the dep. variable.

  Arguments:
      Formal:

      Common blocks:

              COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN
                      arguments contain the maximum and minimum
                      values of the  arrays.

  External References:

              by Leal Lauderbaugh

mmon blocks

  COMMON/MXMN/DVMAX,DVMIN,RIDVMX,RIDVMN

t up offsets and ranges

  IF(IQUAD.NE.1)GOTO 100
adrant I
  XEXTNT=700.
  YEXTNT=500.
  IIDVOF=125
  IDVOFF=125
  GOTO 300
  IF(IQUAD.NE.2)GOTO 200
adrants I&IV
  XEXTNT=700.
  YEXTNT=250.
  IIDVOF=125
  IDVOFF=375
  GOTO 300
  4
  XEXTNT=350.
  YEXTNT=250.
  IIDVOF=475
  IDVOFF=375

nd the absolute max

  IF(ABS(DVMAX).GE.ABS(DVMIN))GOTO 320
  YAMAX=ABS(DVMIN)
  GOTO 350
  YAMAX=ABS(DVMAX)
  IF(ABS(RIDVMX).GE.ABS(RIDVMN))GOTO 370
  XAMAX=ABS(RIDVMN)
  GOTO 400
```

```
  XAMAX=ABS(RIDVMX)

nd scale factors, if abs. maximum =0 scalefactor=1.

  IF(XAMAX.EQ.0.)GOTO 410
  RIDVSC=XEXTNT/XAMAX
  IF(YAMAX.EQ.0.)GOTO 420
  DVSCL=YEXTNT/YAMAX
  GOTO 500
  RIDVSC=1.
  GOTO 408
  RIDVSC=1.
  RETURN
  END
```

```
    .TITLE INITSL
    .GLOBL INITSL

    GENERAL DESCRIPTION: THIS ROUTINE IS USED TO INITIALIZE THE SERIAL
            LINE UNIT STATUS REDGISTER.  IT IS ASSUMED THAT THE LINE IS
            INSTALLED AT LOCATION 176510.

    CALLING EXAMPLE: CALL INITSL

    ARGUMENTS: NONE

    EXTERNAL REFERENCES: NONE

                    BY LEAL LAUDERBAUGH     9/83

    XCSR=176514

SL: BIC #101,@#XCSR            ;NO INTERRUPTS, NO BREAK
    RTS     PC

    .TITLE SEND
    .GLOBL SEND

    GENERAL DESCRIPTION: THIS ROUTINE IS USED TO SEND AN ASCII CHARACTER
            TO THE SERIAL LINE UNIT INSTALLED AT 176510.

    CALLING EXAMPLE: CALL SEND(ICHAR)

    ARGUMENTS:

            FORMAL ARGUMENTS: ICHAR- THE INTEGER*2 DECIMAL EQUIVALENT OF
                    THE CHATACTER BEING SENT.

    EXTERNAL REFERENCES: NONE

                    BY LEAL LAUDERBAUGH     9/83


    SET UP ADRESSES

    XCSR=176514
    XBUF=176516

:   MOV     (R5)+,R0            ;GET # OF ARGUMENTS
    MOV     @(R5)+,R1           ;GET ARGUMENT

ECK FOR TRANSMIT READY

:   BIT     #200,@#XCSR
    BEQ     LOOP

ANSMIT

    MOV     R1,@#XBUF
    RTS     PC
    .TITLE  DAC
    .GLOBL  DAC
```

```
GENERAL DESCRIPTION: THIS REAL TIME ROUTINE IS USED TO INITIATE A
        DIGITAL TO ANALOG CONVERSION.  IT IS ASSUMED THAT THE
        AAV11-A BOARD IS INSTALLED AT THE STANDARD ADRESSES:

                DAC0    170440
                DAC1    170442
                DAC2    170444
                DAC3    170446

CALLING EXAMPLE:

        CALL DAC(CHANNO,DACVAL)

ARGUMENTS:
        COMMON BLOCKS: NONE
        FORMAL ARGUMENTS:
                CHANNO: AN INTIGER*2 VARIABLE CONTAINING THE DAC
                        CHANNEL NUMBER
                DACVAL: THE INTEGER*2 VALUE THAT IS TO BE CONVERTED

EXTERNAL REFERENCES: NONE

                BY LEAL LAUDERBAUGH


MOV     (R5)+,R0
MOV     @(R5)+,R0       ;GET CHANNEL #
MOV     @(R5)+,R1       ;GET VALUE
MOV     R0,R2           ;MULTIPLY THE CHANNEL NUMBER BY 2
ADD     R2,R0           ;BY ADDING IT TO ITSELF
ADD     #170440,R0      ;SET UP ADRESS FOR CONVERTER
MOV     R1,(R0)         ;MOVE THE VALUE INTO THE CONVERTER
RTS     PC
.END
```

APPENDIX E

MATRIX LIBRARY DOCUMENTATION

CHAPTER E-1

INTRODUCTION

This document describes the use of the MATRIX laboratory. The library contains subroutines from both the Scientific Subroutine Package, SSP, and those written by the author. The source will be denoted by the initials SSP or LKL. Additional information on the SSP routines can be found in the Digital manual AA-1101D-TC,AD-1101D-T1.

CHAPTER E-2

DESCRIPTION OF THE ROUTINES

The following list gives a brief description of the routines and a description of the calling format. Consult the SSP manual or the attached listing for more detailed information.

1. MATRIN (LKL) This is a routine to input a matrix. The routine is called as follows: CALL MATRIX(A,N,M,MSA).
2. MATPRT (LKL) This is a routine to output a matrix. The routine is called as follows: CALL MATPRT(A,N,M,MSA).
3. MADD (SSP) This routine is used to add two matrixcies.
4. MFUN (SSP) This routine is used to apply a function to each element of a matrix.
5. MPRD (SSP) This is a routine to multiply two matricies.
6. MSTR (SSP) This is a routine to change the storage made of a matrix.
7. MSUB (SSP) This routine is used to subtract one matrix from another.
8. MTRA (SSP) This routine forms the transpose of a matrix.
9. SMPY (SSP) This is a routine to multiply each element of a matrix by a scalar.
10. XCPY (SSP) This routine copies a sub matrix from a matrix.
11. MCPY (SSP) This routine copies an entire matrix.
12. LOC (SSP) This routine is used by the other subroutines.

```
  SUBROUTINE MATRIN(A,N,M,MSA)

  General Description: This subroutine is used to input an nxm matrix A
      The routine does not do any error checking.  It is assumed
      that if MSA is set to 1 or 2 that the matrix is square.

  Calling example:
   to input a general 5x7 matrix:
      N=5
      M=7
      MSA=0
      CALL MATRIN(A,N,M,MSA)

  Arguments:

      Formal:
              A-name of the input matrix
              N-number of rows in matrix A
              M-number of columns in A
              MSA-one digit number for the storage mode of A
                      0-General
                      1-Symmetric
                      2-diagonal

      Common blocks:
              IN-input device
              IOUT-output device
  External References: MATPRT
                   LOC

              by Leal Lauderbaugh  7/83


clarations

  LOGICAL*1 ANSW,NO,YES

mmon blocks

  COMMON/INOUT/IN,IOUT

ta statments

  DATA NO/'N'/,YES/'Y'/

  DIMENSION A(1)

termine the form of A

  WRITE(IOUT,5)N,M,MSA
  ITEST=MSA-1
  FORMAT(3(' ',I1))
  IF(ITEST) 100,200,300
```

```
put a general matrix by columns

  WRITE(IOUT,105)
  FORMAT(' INPUT THE MATRIX BY COLUMNS')
  DO 117 J=1,M
  DO 117 I=1,N
  WRITE(IOUT,110)I,J
  FORMAT(' INPUT THE ',I2,','I2,'TH ELEMENT')
  CALL LOC(I,J,IR,N,M,MSA)
  READ(IN,120)A(IR)
  FORMAT(G10.4)
  GOTO 400

put a symmetric matrix by columns

  WRITE(IOUT,205)
  FORMAT(' INPUT THE UPPER TRIANGLE OF THE SYMMETRIC MATRIX')
  DO 217 J=1,M
  JK=J
  DO 217 I=1,JK
  WRITE(IOUT,110)I,J
  CALL LOC(I,J,IR,N,M,MSA)
  READ(IN,120)A(IR)
  GOTO 400

put a diagonal matrix

  WRITE(IOUT,305)
  FORMAT(' INPUT THE MATRIX DIAGONAL')
  DO 315 I=1,N
  J=I
  WRITE(IOUT,110)I,I
  CALL LOC(I,J,IR,N,M,MSA)
  WRITE(IOUT,320)IR,I
  FORMAT(2(I4))
  READ(IN,120)A(IR)

int the matrix and check for changes

  CALL MATPRT(A,N,M,MSA)

eck for a change

  WRITE(IOUT,9991)
  FORMAT(' Do you wish to make any changes? (y/n)')
  READ(IN,9992)ANSW
  FORMAT(A1)
  IF(ANSW.EQ.YES)GO TO 10000
  IF(ANSW.NE.NO)GO TO 9990
  GOTO 10500
0 WRITE(IOUT,10320)
0 FORMAT(' INPUT I,J')
  READ(IN,10330)I,J
0 FORMAT(2(I3))
```

```
eck for changes in an off diagonal element of a diagonal matrix

  IF((MSA.EQ.2).AND.(I.NE.J))GOTO 10350

  check to see if I and J are in the range of the matrix

  IF((I.GT.N).OR.(J.GT.M))GOTO 10400
  WRITE(IOUT,10340)I,J
0 FORMAT(' INPUT THE NEW ',I2,',',I2,' th ENTRY')
  CALL LOC(I,J,IR,N,M,MSA)
  READ(IN,120)A(IR)
  GOTO 400
0 WRITE(IOUT,10355)
5 FORMAT(' **** ERROR ****',/,
 +     ' THE MATRIX IS DIAGONAL.  YOU ARE ONLY',/,
 +     ' ALLOWED TO CHANGE THE DIAGONAL ENTRIES.')
  GOTO 9990
0 WRITE(IOUT,10410)N,M
0 FORMAT(' **** ERROR ****',/,
 +     ' THE I OR J SPECIFIED WAS TO LARGE.',/,
 +     ' THE DIMENSION OF THE MATRIX IS,',I2,'x',I2)
  GOTO 9990
0 RETURN
  END
```

```
SUBROUTINE MATPRT(A,N,M,MSA)

General Description: this routine is used to print the nxm matrix A

Calling Example:
    N=5
    M=7
    CALL MATPRT(A,N,M)

Arguments:
    Formal:
            A - name of the input matrix
            N - number of rows in matrix A
            M - number of columns in A
            MSA-one digit number for the storage mode of A
                    0-General
                    1-Symmetric
                    2-diagonal

    Common blocks:

            COMMON/INOUT/IN,IOUT
            IN: The input device number is 5 which
                is assigned to TT
            IOUT: The output device number is 6 which
                  is assigned to TT

            by Leal Lauderbaugh 7/83



mmon blocks

  COMMON/INOUT/IN,IOUT

  DIMENSION A(1)
  DO 15 I=1,N
  DO 12 J=1,M
  CALL LOC(I,J,IR,N,M,MSA)
eck for IR=0 if true then print 0.0
cause this is an off diagonal element of a
agonal matrix.
  IF(IR.EQ.0)GOTO 50
  P=A(IR)
  GOTO 12
  P=0.0
  WRITE(IOUT,20)P
  FORMAT('$',G10.4)
  WRITE(IOUT,25)
  FORMAT(' ')
  RETURN
  END
```