Dynamic Model Processor User's Manual:
A Programming Aid for Easier Simulation Modeling

by

William J. Overholtz and John W. Hauser

National Marine Fisheries Service
Northeast Fisheries Center
Woods Hole Laboratory
Woods Hole, Massachusetts 02543

January 1984
dynamic model processor
user's manual
a programming aid for easier simulation modeling
by
william j overholtz
john w hauser
**TABLE OF CONTENTS**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>DMP System Philosophy</td>
<td>3</td>
</tr>
<tr>
<td>Features</td>
<td>4</td>
</tr>
<tr>
<td>Procedures for Writing a DMP Model</td>
<td>6</td>
</tr>
<tr>
<td>Variable Disc File</td>
<td>6</td>
</tr>
<tr>
<td>Command File</td>
<td>8</td>
</tr>
<tr>
<td>Model Construction</td>
<td>8</td>
</tr>
<tr>
<td>DMP Interactive Commands</td>
<td>13</td>
</tr>
<tr>
<td>DMP Commands and Their Uses</td>
<td>14</td>
</tr>
<tr>
<td>Backup</td>
<td>14</td>
</tr>
<tr>
<td>Graph</td>
<td>15</td>
</tr>
<tr>
<td>Macro</td>
<td>16</td>
</tr>
<tr>
<td>Run</td>
<td>16</td>
</tr>
<tr>
<td>Set</td>
<td>17</td>
</tr>
<tr>
<td>Show</td>
<td>18</td>
</tr>
<tr>
<td>Sub</td>
<td>18</td>
</tr>
<tr>
<td>Table</td>
<td>19</td>
</tr>
<tr>
<td>An Example of an Actual DMP Run</td>
<td>20</td>
</tr>
<tr>
<td>Adapting Other Models</td>
<td>26</td>
</tr>
</tbody>
</table>
The education of a man is never completed until he dies.  
--Robert E. Lee

Truth in science can be defined as the working hypothesis best suited to open the way to the next better one.  
--Konrad Lorenz

Foreword:

This guide to the Dynamic Model Processor (DMP) is designed as an aid for modelers, to use the system with a minimum of effort. The system is programmed in FORTRAN 77 and operates on the Woods Hole Oceanographic VAX 11/780 computer. The guide can be read by persons who have little familiarity with Fortran, but anyone who wishes to use the processor as a simulation tool must have a working knowledge of the language.
Introduction

In the past several years we have witnessed a quantum jump in the use of high speed digital computers for solving problems in almost every pursuit of science. Astrophysics, agriculture, chemistry, nuclear engineering, medical research, and paleontology are but a few of the fields where computers have an ever increasing role. The directions that many of these disciplines are taking are being determined by the computer. The new interest in using computers to solve problems that were previously too tedious or time limited has spawned a whole new set of ideas and methods for solving complicated problems. Computer languages, such as Basic, Cobol, Fortran, APL, and Pascal have become available for use by people from every walk of life. New professional societies, such as the Society for Computer Simulation, have been formed. Simulation modeling has made it possible for researchers to enter into new realms of prediction and experimentation that were once impossible.

Even the biological sciences' venerable bastions of Baconian logic and method are becoming inextricably tied to the computer for their future. Ecology, in particular, has benefitted from the array of new analyses, models, and investigatory methods that are now available because of the computer. The role of simulation modeling in this field has grown in leaps and bounds in just a few short years and has taken a place of importance alongside the traditional methods. New professional societies, such as the International Society for Ecological Modeling, and journals,
such as Ecological Modeling, are now prominent features of the science. Numerous textbooks are available on ecological modeling and almost every major university has courses and at least one faculty member who considers himself or herself a "modeler."

This proliferation of models, methods, and strategies is viewed by many as an impenetrable jungle, and rightly so, for the inexperienced programmer or the intermittent user. Fortunately, a number of simulation systems or model processors have become available for the serious and even part-time modeler to use. These systems strive to simplify the modeling process so that more time can be spent on the results and relevant output rather than the programming aspects. Several systems or processors, notably SIMCON, FLEX, DYNAMO, GASP, and SIMSCRIPT, were designed to make life much easier on the simulation modeler; these can be interfaced with the basic models to allow control of simulation inputs and outputs. These systems typically include a command language specific to the particular processor that allows the modeler to run one or several iterations of a model and to produce many kinds of outputs, such as graphic displays.

This document discusses and highlights a new model processor developed at the Northeast Fisheries Center. This system, known as Dynamic Model Processor (DMP), was developed to simplify the modeling process for users of the Woods Hole Oceanographic VAX 11/780 computer. This users manual is designed to explain the system and how it works. It should be of interest to people from different areas of expertise who use the WHOI computer system. For more details and documentation, consult the DMP manual prepared by John W. Hauser, 12 September 1983.
DMP System Philosophy

The goal of the DMP system is to simplify the task of computer modeling. It is a difficult process to design a computer program that not only contains the equations and code for the rules of update for the model but also contains all the input and output (I/O) and variable interactions. Anyone who has constructed a simulation model knows how laborious it can be to construct a model with all the associated I/O. Any change that needs to be made usually involves major alterations of the model code, and constructing a new model means starting all over with a new program. This process can be greatly simplified by standardizing the structure of the model and also separating it from the actual I/O and interaction procedures.

The basic design philosophy of the DMP system is as follows. The user builds a FORTRAN model according to a specified structure that can be interfaced with the processor (Figure 1). It is unnecessary to include most READ, WRITE, and FORMAT statements in the model because DMP is designed to handle I/O for the user. Programming tabular output, graphics, time sequencing and all the other real time activities that are used in simulations are handled by DMP. Much of the programming time and design necessary for interactive simulation is spent in dealing with these activities. The DMP system is such that simple commands are used to accomplish the real time interactive changes in variable and graphical output, etc., that are a necessary part of the process.
The model processor is separate from the model and as such makes the simulation process much easier (Figure 1). The system is most directly applicable in modeling exercises where discrete time is used. A time step for DMP is one complete cycle through the user's program. Resolution for the time step could be on a daily, monthly, or yearly basis, depending on the particular model of interest.

DMP has several features which are useful to the modeler in building and running a simulation model. These are as follows:

1. A set of easily used and self-explanatory commands for running the system.
2. Variables can be viewed via the GRAPH or TABLE commands after a simulation run.
3. System states at any time step can be restored using the BACKUP command.
4. Batch files can be used to run the simulation for the modeler or DMP can be run in an interactive mode.
5. MACRO's or sets of DMP commands can be constructed and used to make simulation runs. In addition, in cases where multiple runs are desired, iterations of a MACRO can be easily performed.
6. The user can intervene during any time step to change variable assignments via the SET command.
7. There are 10 separate subroutine slots available for use by the programmer to do extra calculations or work using output from DMP. These subroutines are accessed with SUB1...SUB10 commands at the interactive level.
8. DMP is interfaced with the VAX graphic library during the run initialization so that referral of graphics plots to the CALCOMP or other plotter is easily accomplished.
Figure 1. Structure of a model built to run on DMP with the necessary linkages to the system and the VAX plot library (from Hauser 1983).
Procedures for writing a DMP model

As previously mentioned, it is necessary for the DMP user to construct a FORTRAN model according to a standard design in order to access the DMP system. Three basic operations are necessary to interface your model with the DMP; these are as follows:

1. Build a disc file with all the state variables that you want DMP to recognize and record information about over the simulation time intervals you choose.

2. Build a command file that accesses the variable file, provides output work space for DMP and defines other necessary information that DMP needs to recognize and run your model.

3. Construct your model according to the same structure shown in Figure 2. DMP requires a main program loop that calls the DMP processor, a subroutine called MYSTEP that contains the model of interest and another subroutine called MYINIT which sets the initial values for the model.

Variable Disc File

The DMP system operates by recording current values for specified variables at each time step of the model. This is why the variables are so easily accessed at the end of a simulation for GRAPH, TABLE, or any of the other DMP commands. It is important for the user to determine which variables should be accessible and to put these into the variable disc file (Figure 2). These variables are also put in a common block.
Figure 2. Example structure of a model that includes a main program loop, the MYINIT subroutine, the MYSTEP subroutine and the additional SUB 1-10 subroutines. This model structure is necessary for access to the DMP system (from Hauser 1983).

```
INTEGRAL (A-Z)
REAL A,B,C,D(20),E(3,4,5)
COMMON A,B,C,D,E
STOP
END

SUBROUTINE MYINIT
INTEGRAL (A-Z)
REAL A,B,C,D(20),E(3,4,5)
COMMON A,B,C,D,E
DO I=1,3
DO J=1,4
DO K=1,5
E(I,J,K)=100*I+10*J+K
END DO
END DO
RETURN
END

SUBROUTINE MYSTEP
INTEGRAL (A-Z)
REAL A,B,C,D(20),E(3,4,5)
COMMON A,B,C,D,E
A=A+1
B=B+1
C=C-1
DO I=1,20
D(I)=D(I)+1
END DO
RETURN
END

SUBROUTINE S1
INTEGRAL (A-Z)
REAL A,B,C,D(20),E(3,4,5)
COMMON A,B,C,D,E
TOTAL=A+B
WRITE(6,*,'(A,B=','TOTAL
RETURN
END

SUBROUTINE S2
RETURN
END

SUBROUTINE S3
RETURN
END

SUBROUTINE S4
RETURN
END

SUBROUTINE S5
RETURN
END

SUBROUTINE S6
RETURN
END

SUBROUTINE S7
RETURN
END

SUBROUTINE S8
RETURN
END

SUBROUTINE S9
RETURN
END

SUBROUTINE S10
RETURN
END
```

The main program calls DMP

Variables initialized in MYINIT if not previously initialized by the variable definition file.

MYSTEP contains the rule for changing variables from one timestep to the next.

Subroutine to calculate and display A+B.

Dummy subroutines
declaration statement in all the DMP subroutines that the user builds (Figure 2).

Command file

As with many complicated programs where multiple file assignments are necessary to run programs on the VAX, it is advantageous to set up a command file to make all the necessary file assignments (Figure 3). In this case the DMP system requires all the assignments in Figure 3 for execution of the user's model. It would not run, for instance, if the MACRO library $ASSIGN was not included in the command file, even if no MACRO's are to be used in the simulation.

Model Construction

As previously mentioned all programming for DMP must be in FORTRAN. The model processor is very flexible, and will work on models for discrete and continuous time applications. As such, DMP has no numerical integration subroutines or other internal capabilities for running standard continuous time integrators; these would all have to be programmed into the MYSTEP subroutine. All models and applications of DMP to present have been discrete time solutions. One loop through the MYSTEP subroutine is considered a time step in the DMP system. The level of resolution could be seconds, days, months, or years; this is entirely up to the modeler. The processor functions on number of time steps in the RUN mode, hence it only records the complete cycles through MYSTEP.
Figure 3. Linking procedure and type of command file necessary for running a linked version of a model on the DMP system. All the Fortran and other system assignments are necessary parts of the command file (from Hauser 1983).

```
LINK myprogram,[344.FISH]DMP,DISSINT/LIB,DISSPLA/LIB

Create a command file similar to the following:
$ASSIGN myvar.dat FOR001  ! Input file of variable names
$ASSIGN myrun.dat FOR002  ! Record of run, output, can be input
$ASSIGN mytable.dat FOR003  ! Tabular output
$ASSIGN mymacro.dat FOR004  ! User's macro library
$ASSIGN mygraph.dat DISSMETA  ! Deferred plot output
$ASSIGN NL: FOR009  ! Discards unneeded output
$ASSIGN SYS$COMMAND SYS$INPUT  ! Accepts replies from terminal
$RUN myprogram  ! Runs the linked model

Execute the command file.
```
To build a model and interface it with DMP the modeler must follow standard procedures. A standardized model is much easier to understand and change. The main program loop is really only a call to the DMP system (Figure 2). It contains little else but a call DMP statement.

The MYINIT subroutine is a variable initialization procedure. Variables can be initialized in the variable disc file also, but MYINIT overrides all such assignments. The advantages of putting all such information into one location are obvious when the user starts to build complicated models. The variables that are in this subroutine and are held in common can, of course, be changed at any time by using the SET command. This subroutine can also be used to read outside disc files that contain initial values for variables.

The final subroutine that the user needs to construct is called MYSTEP (Figure 2). This subroutine is actually the model or the rules of update that the user has constructed for the system of interest. In other words, all the relevant equations, functional relationships, mechanisms, etc., are programmed in this subroutine. To design efficient, easily tractable models it is often advantageous to use the MYSTEP subroutine as a main model loop with several FORTRAN call statements to subordinate subroutines. This makes for an efficiently designed, easily understood model that can be read and followed by most people who are familiar with FORTRAN.
Figure 4 describes the whole process in a diagram that traces the different procedures to follow for building a model, interfacing it with the DMP system, running it and leaving the processor.
Figure 4. Flow diagram describing the steps necessary for constructing a model, interfacing it with DMP, running it and exiting the DMP system.
### DMP Interactive Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP</td>
<td>Returns the model to a previous time step</td>
</tr>
<tr>
<td>EXIT</td>
<td>Terminates the DMP run sequence</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Produces a graph of variables vs time or some other specified variable</td>
</tr>
<tr>
<td>MACRO</td>
<td>Enables the user to create a set of DMP commands that can be executed during a run sequence</td>
</tr>
<tr>
<td>RUN</td>
<td>Runs the model by executing subroutine MYSTEP the requested number of times.</td>
</tr>
<tr>
<td>SET</td>
<td>Changes the current value of variables</td>
</tr>
<tr>
<td>SHOW</td>
<td>Displays the current value of a variable</td>
</tr>
<tr>
<td>SUB</td>
<td>Enables the user to execute additional user supplied subroutines</td>
</tr>
<tr>
<td>TABLE</td>
<td>Produces a table of requested variables over a specified time interval.</td>
</tr>
</tbody>
</table>
DMP Commands and Their Uses

This section summarizes the DMP commands that are available together with appropriate examples. After each command the system will, of course, prompt you for information.

BACKUP: This command will return the model state to any previous time step or restore the model to the initial conditions. This is very useful if repeated runs are needed or parameter or variable assignments need to be changed and the simulation rerun or continued.

(1.): > BACKUP
  : > 0 This command restores the model state to the starting point of the simulation run.

(2.): > BACKUP
  : > -1 This reinitializes the MYINIT subroutine and restores the model state to the starting point of the initialization run. Useful in cases where an outside function such as RAN, the VAX random number generator, is used.

(3.): > BACKUP
  : > 5 This command sets the system state back to timestep 5.
GRAPH: This command allows the user to output a graph on a graphics terminal or to a disc file for future plotting. Several options exist for producing a plot of one variable vs another or a time trajectory of a variable.

(1.) > GRAPH: Initiates GRAPH procedure

  : > Y: Output will be written on a disc file
  : > Time: X axis variable
  : > B: Y axis variable
  : > O: 1st time to be plotted
  : > 10: last time to be plotted

  This set of DMP commands will produce a graph with time as the X axis variable and B as the Y axis variable for interval 0-10.

(2.) > GRAPH:

  : > N: Output will be displayed on the terminal
  : > A: X axis variable
  : > B: Y axis variable
  : > O: 1st time to be plotted
  : > 100: last time to be plotted

  This set of commands will produce a graph with A as the X axis variable and B as the Y axis variable for the set of data points that occurred between timesteps 0 and 100 for these two variables.
MACRO: This command allows the user to string together a series of DMP commands. This can be a very useful feature when a series of simulations are being run or repetitive command sequences are being used. This command does nothing itself, but allows the user to create a MACRO file that runs by typing the MACRO name.

```
(1.)
: > MACRO This phases the user with the MACRO mode
: > C Instructs the system to create a MACRO
: > GO The user gives the MACRO a name of choice, in this case GO
: > TABLE Instructs the system to produce a table
: > N The table will be displayed on the terminal
: > A Print variable A, B, C in the
: > B table
: > C
: > end table command
: > 1 1st year on table
: > 10 last year on table
: > 1 interval time
: > X ends the MACRO creation
```

This series of commands would produce a set of values for variables A, B, and C, over the interval 1-10 with every timestep printed. By typing the word GO, this whole series of commands would be executed and the table would be produced.

RUN: This command allows the user to run the model for a specified number of iterations.

```
(1.): > RUN
: > 10 Run the model for 10 timesteps
```
The user in this case has run the model of interest over 10 complete loops. Additional commands are needed to recover information from this run.

SET: This DMP command allows the user to change parameter or variable assignments at any stage of the simulation. This command allows the user to avoid changing the FORTRAN code in MYSTEP or MYINIT when parameter changes are required.

(1.): > SET allows the user to enter set mode
   : > A variable you wish to change
   : > 1.00 value you wish variable to be
   : > gets user out of the set mode

   The variable A now has the value 1.00.

(2.): > SET gets user into the set mode
   : > A change variable A
   : > 100 variable A = 100
   : > exit the set mode
   : > RUN run the model for 5 time
   : > 5 steps
   : SET enter set mode again
   : A change variable A again
   : > 50 variable A = 50
   : > exist set
   : > RUN run model for 15 more
   : > 15 iterations
This sequence of commands illustrates how the user can change a variable assignment during an actual simulation run. In this case the variable A was set to 100, 5 iterations run, set to 50 and 15 more iterations were run.

SHOW: This command allows the user to display the current value of a variable. This command is useful for following model results, debugging the model, or checking that initial values are correct.

(1.): > SHOW enter the SHOW mode
>: > A show value of variable A
  10.0000 variable A = 10.0
>: > exit show mode

SUB: This command executes a user supplied subroutine. Often additional calculations or data analyses, that are separate from the MYSTEP subroutine, are desired. DMP provides the user with a work area to write up to 10 of those separate user supplied subroutines. Any unused subroutines must be included in dummy subroutines (Figure 1). See sample user program (Figure 2) for more details.
TABLE: This command allows the user to produce tabular output for any variable in the variable file. The output can be sent to the disc or the terminal. The user can send output to the disc with or without the variable headings.

(1.): > TABLE enter the table command mode
     : > Y send table to disc
     : > N no variable headings on disc file
     : > A include variable A,B,C in file
     : > B
     : > C
     : > stop entering variables
     : > 1 1st time to be tabulated
     : > 20 last time to be tabulated
     : > 1 time interval

This series of responses produces a table of variables A,B,C, on the disc without variable headings for time intervals 1-20 with each interval printed.

(2.): > TABLE enter TABLE command mode
     : > N send output to terminal
     : > A output information for variable A
     : > stop entering variables
     : > 1 1st time to be printed
     : > 10 last time to be printed
     : > 2 time interval

This series of responses would produce a table with variable A on the terminal for time interval 1-10, but only every other timestep be printed.
An example of an actual DMP model run

Now that the structure and function of the processor have been explained, some actual examples of using the command language will help the user to better understand the whole process.

(1) In this first example the command file is executed, the SHOW command is illustrated and a simple table with 3 variables is produced. The model in use is an actual fisheries management model that was built to simulate the dynamics of the Georges Bank haddock stock.

```
DYNAMIC MODEL PROCESSOR BEGINS
DO YOU WANT TO REPROCESS AN OLD RUN? (Y or N): N
ENTER COMMAND : RUN
ENTER NUMBER OF Timesteps : 10
ENTER COMMAND : SHOW
ENTER VARIABLE NAME (or C/R for end of list): TCATCH
11692.15
ENTER VARIABLE NAME (or C/R for end of list): VTNUM
1.3638076E-07 1744323.2 9638865.9 3528474.9 5146131.
ENTER VARIABLE NAME (or C/R for end of list): SPSTOCK
41350.68
ENTER VARIABLE NAME (or C/R for end of list)
ENTER COMMAND : TABLE
DO YOU WISH TABLES TO BE SENT TO A DISK FILE? (Y or N): N
ENTER VARIABLE NAME (or C/R for end of list): TCATCH
ENTER VARIABLE NAME (or C/R for end of list): TSTOCK
ENTER VARIABLE NAME (or C/R for end of list): SPSTOCK
ENTER VARIABLE NAME (or C/R for end of list)
ENTER FIRST TIME TO BE TABULATED: 1
ENTER LAST TIME TO BE TABULATED: 10
ENTER TIME BETWEEN TABULATIONS: 1

<table>
<thead>
<tr>
<th>TIME</th>
<th>TCATCH</th>
<th>TSTOCK</th>
<th>SPSTOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6256.871</td>
<td>603103.958</td>
<td>57439.586</td>
</tr>
<tr>
<td>2</td>
<td>40696.044</td>
<td>111709.658</td>
<td>1243.031</td>
</tr>
<tr>
<td>3</td>
<td>61656.598</td>
<td>112861.164</td>
<td>48523.773</td>
</tr>
<tr>
<td>4</td>
<td>120339.768</td>
<td>407288.689</td>
<td>439180.153</td>
</tr>
<tr>
<td>5</td>
<td>157349.304</td>
<td>474955.059</td>
<td>447282.097</td>
</tr>
<tr>
<td>6</td>
<td>161498.309</td>
<td>614668.097</td>
<td>447282.097</td>
</tr>
<tr>
<td>7</td>
<td>149633.730</td>
<td>660817.929</td>
<td>529502.133</td>
</tr>
<tr>
<td>8</td>
<td>147263.593</td>
<td>686851.857</td>
<td>447282.097</td>
</tr>
<tr>
<td>9</td>
<td>141922.451</td>
<td>681565.326</td>
<td>41350.680</td>
</tr>
<tr>
<td>10</td>
<td>11692.151</td>
<td>44715.326</td>
<td>41350.680</td>
</tr>
</tbody>
</table>
```
(2) This next sequence of commands illustrates how to use the BACKUP and SET procedures to move the simulation back to a previous time step, change a variable value, and continue the simulation. A table is produced that is different from the previous example starting at time 6.

```
ENTER COMMAND :)SBACKUP
ENTER TIME STEP FOR BACKUP (initial condition = 0) :)5
ENTER COMMAND :)SET
ENTER VARIABLE NAME (or C/R for end of list) :)FMORT
ENTER VALUE :)0.6
ENTER VARIABLE NAME (or C/R for end of list) :)FMORT
ENTER COMMAND :)RUN
ENTER NUMBER OF TIMESTEPS :)5
ENTER COMMAND :)TABLE
DO YOU WISH TABLES TO BE SENT TO A DISK FILE? (Y or N) IF NOT IT WILL BE SENT TO YOUR TERMINAL :)N
ENTER VARIABLE NAME (or C/R for end of list) :)TCATCH
ENTER VARIABLE NAME (or C/R for end of list) :)TSTOCK
ENTER VARIABLE NAME (or C/R for end of list) :)SPSTOCK
ENTER VARIABLE NAME (or C/R for end of list) :)STOCK
ENTER FIRST TIME TO BE TABULATED :)1
ENTER LAST TIME TO BE TABULATED :)10
ENTER TIME BETWEEN TABULATIONS :)1
```

<table>
<thead>
<tr>
<th>TIME</th>
<th>TCATCH</th>
<th>TSTOCK</th>
<th>SPSTOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6655.271</td>
<td>23183.058</td>
<td>18168.646</td>
</tr>
<tr>
<td>2</td>
<td>8286.775</td>
<td>24175.089</td>
<td>19430.438</td>
</tr>
<tr>
<td>3</td>
<td>7898.644</td>
<td>21789.588</td>
<td>16794.641</td>
</tr>
<tr>
<td>4</td>
<td>5208.350</td>
<td>67658.686</td>
<td>61485.628</td>
</tr>
<tr>
<td>5</td>
<td>15394.092</td>
<td>59240.164</td>
<td>54755.641</td>
</tr>
<tr>
<td>6</td>
<td>16394.090</td>
<td>59240.164</td>
<td>54755.641</td>
</tr>
<tr>
<td>7</td>
<td>32712.904</td>
<td>33932.273</td>
<td>32940.566</td>
</tr>
<tr>
<td>8</td>
<td>26541.123</td>
<td>39390.812</td>
<td>23278.576</td>
</tr>
<tr>
<td>9</td>
<td>26392.258</td>
<td>31779.664</td>
<td>18379.631</td>
</tr>
<tr>
<td>10</td>
<td>31061.963</td>
<td>49464.129</td>
<td>18379.631</td>
</tr>
<tr>
<td></td>
<td>38457.830</td>
<td>51384.188</td>
<td>30159.031</td>
</tr>
</tbody>
</table>
This example illustrates the use of the GRAPH command. This command can be used to produce variable trajectories or plots of one variable vs another.

The first case shows a plot of a variable vs time.

```
ENTER COMMAND : GRAPH
DO YOU WISH TO DEFER GRAPHICAL OUTPUT SO THAT IT CAN BE PRODUCED LATER ON ANOTHER DEVICE? (Y or N)
IF NOT OUTPUT WILL BE TO YOUR GRAPHICS TERMINAL : N
ENTER VARIABLE FOR X AXIS : TIME
ENTER VARIABLE FOR Y AXIS : TIME
ENTER FIRST TIME TO BE PLOTTED : 0
ENTER LAST TIME TO BE PLOTTED : 10
```

![Graph of TORCH vs TIME]
This sequence of commands produces a plot of one variable vs another.

```
ENTER COMMAND
:GRAPH
DO YOU WISH TO DEFER GRAPHICAL OUTPUT SO THAT IT CAN BE PRODUCED LATER ON ANOTHER DEVICE? (Y OR N) N
ENTER VARIABLE FOR X AXIS :TCATCH
ENTER VARIABLE FOR Y AXIS :XNUM
ENTER SUBSCRIPT 1
ENTER FIRST TIME TO BE PLOTTED 1
ENTER LAST TIME TO BE PLOTTED 10
```

![Graph plot](image-url)
(4) This example illustrates the use of the MACRO command. A set of DMP commands are strung together and given the name STOP. When this MACRO name is entered, the MACRO is executed and a table of specified variables is produced.

MACRO
ENTER C IF YOU WISH TO CREATE A NEW MACRO
ENTER D IF YOU WISH TO DELETE AN OLD MACRO
ENTER Q IF YOU DO NOT WISH TO DO EITHER

ENTER NAME OF MACRO TO BE CREATED
STOP
ENTER COMMAND

:STOP
How many times do you wish to execute this macro?

:1

<table>
<thead>
<tr>
<th>TIME</th>
<th>TCATCH</th>
<th>TSTOCK</th>
<th>SPSTOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5625.271</td>
<td>31933.998</td>
<td>18195.646</td>
</tr>
<tr>
<td>2</td>
<td>5369.244</td>
<td>21176.289</td>
<td>18430.438</td>
</tr>
<tr>
<td>3</td>
<td>17938.358</td>
<td>71593.686</td>
<td>18742.641</td>
</tr>
<tr>
<td>4</td>
<td>15394.938</td>
<td>59240.154</td>
<td>16489.641</td>
</tr>
<tr>
<td>5</td>
<td>32712.904</td>
<td>33320.873</td>
<td>32940.566</td>
</tr>
<tr>
<td>6</td>
<td>26541.123</td>
<td>39390.683</td>
<td>17823.627</td>
</tr>
<tr>
<td>7</td>
<td>29392.450</td>
<td>31779.664</td>
<td>23760.576</td>
</tr>
<tr>
<td>8</td>
<td>31051.858</td>
<td>49404.139</td>
<td>30199.631</td>
</tr>
<tr>
<td>9</td>
<td>38457.838</td>
<td>51304.188</td>
<td>18375.931</td>
</tr>
</tbody>
</table>

ENTER COMMAND
Another feature of the MACRO command is that the sequence can be executed repeatedly if necessary. This is extremely valuable when multiple iterations of the same set of commands are necessary, for example, in a stochastic simulation.

In this case the MACRO "STOP" is executed 2 times.

STOP

MANY TIMES DO YOU WISH TO EXECUTE THIS MACRO?

:\^2

<table>
<thead>
<tr>
<th>TIME</th>
<th>TCATCH</th>
<th>TSTOCK</th>
<th>SPSTOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5625.271</td>
<td>23103.698</td>
<td>18195.646</td>
</tr>
<tr>
<td>2</td>
<td>5633.776</td>
<td>21176.289</td>
<td>18430.438</td>
</tr>
<tr>
<td>3</td>
<td>76458.244</td>
<td>71659.688</td>
<td>18742.641</td>
</tr>
<tr>
<td>4</td>
<td>76594.562</td>
<td>87659.266</td>
<td>16495.656</td>
</tr>
<tr>
<td>5</td>
<td>76594.052</td>
<td>59240.164</td>
<td>16742.641</td>
</tr>
<tr>
<td>6</td>
<td>39528.273</td>
<td>33932.273</td>
<td>17823.627</td>
</tr>
<tr>
<td>7</td>
<td>39390.602</td>
<td>31779.664</td>
<td>17823.627</td>
</tr>
<tr>
<td>8</td>
<td>31061.963</td>
<td>49404.128</td>
<td>18379.631</td>
</tr>
<tr>
<td>9</td>
<td>38457.532</td>
<td>51304.188</td>
<td>30159.631</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIME</th>
<th>TCATCH</th>
<th>TSTOCK</th>
<th>SPSTOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5625.271</td>
<td>23103.698</td>
<td>18195.646</td>
</tr>
<tr>
<td>2</td>
<td>5633.776</td>
<td>21176.289</td>
<td>18430.438</td>
</tr>
<tr>
<td>3</td>
<td>76458.244</td>
<td>71659.688</td>
<td>18742.641</td>
</tr>
<tr>
<td>4</td>
<td>76594.562</td>
<td>87659.266</td>
<td>16495.656</td>
</tr>
<tr>
<td>5</td>
<td>76594.052</td>
<td>59240.164</td>
<td>16742.641</td>
</tr>
<tr>
<td>6</td>
<td>39528.273</td>
<td>33932.273</td>
<td>17823.627</td>
</tr>
<tr>
<td>7</td>
<td>39390.602</td>
<td>31779.664</td>
<td>17823.627</td>
</tr>
<tr>
<td>8</td>
<td>31061.963</td>
<td>49404.128</td>
<td>18379.631</td>
</tr>
<tr>
<td>9</td>
<td>38457.532</td>
<td>51304.188</td>
<td>30159.631</td>
</tr>
</tbody>
</table>

ENTER COMMAND

:D:EXIT

DYNAMIC MODEL PROCESSOR ENDS

FORTRAN STOP
Adapting Other Models

Since the nature of the DMP system is very general, other models can be adapted quite easily and run on DMP. The necessary changes have already been discussed, they involve setting up the model according to the structure in Figure 2. Much of the I/O could be removed, so in this sense the model will be streamlined and much more efficient. Any system dependent programming that will not run on the VAX will, of course, not work in DMP either.