MICROSIM: A BASIC BASED, DISCRETE EVENT SIMULATION LANGUAGE FOR MICROCOMPUTERS

Mark L. Spearman
Superior Oil Company
Houston, Texas

Abstract

MicroSim is a simulation language developed specifically for use on microcomputers. The package is written in Basic, a language commonly found on microcomputers, and is a generalized discrete event simulator. The MicroSim language provides the analyst with all of the necessary computational features such as random number generation, time keeping, etc. that are used in simulation. This language is event oriented and allows the user to write subroutines in Basic to describe discrete events. The language also allows for limited continuous modeling by the use of linear difference equations.

INTRODUCTION

Since the first "personal computer" appeared in the market place some five years ago, the capabilities of these machines has increased greatly. No longer considered a toy, these devices are now being used by corporations for computational tasks such as planning, control, engineering analysis, and simulation. The need for a simulation tool that will run on a microcomputer has arisen from the increased presence of the small computers performing other tasks in both industry and academia.

Developing simulation models on a microcomputer using a general programming language such as Basic, FORTRAN or Pascal can be very tedious and will usually result in a model that is difficult to revise. A high level simulation language specifically designed for microcomputer use is needed. This language should provide the user with the advanced computational features such as random deviate generators, time and variable tracking, trace controls, queuing, and other features found in simulation models. MicroSim is a collection of Basic routines that will provide the simulation analyst with these features. These routines will be discussed individually later. Because most microcomputers have a maximum storage capacity of 64K bytes, the MicroSim package has been kept small in size to allow for the data requirements of the model. The structure of MicroSim results in an organized approach to modeling by requiring the user to model the simulation activities in terms of discrete "events." As in the FORTRAN based language, GASP IV, the user writes event subroutines to describe the modeled system. The world view embodied in the discrete event orientation consists of modeling a system by describing the changes that occur in the system at discrete points in time. The state of the system is defined by a set of variables called the system image. The system image in a MicroSim model is composed of two types of variables: user defined variables and MicroSim variables. An isolated point in time where the state of the system can change is called an event time and the associated logic for processing the changes to the system image is called an event. A discrete event model of a system is constructed by defining the events where changes in the system image can occur and then modeling the logic associated with each event type. A dynamic portrayal of the system is produced by causing system image changes according to the logic of each event in a time ordered sequence.

The two types of events that are found in MicroSim models are scheduled events and crossing events. Scheduled events are generated by event entities which are filed (scheduled) in an event file. Each event entity has a set of at least two attributes. The first attribute is always the event time. This is defined either in the model initialization step or within the logic of an event before the event entity is filed in the event file. The second attribute is called the event code. This code is used to reference the appropriate event logic which the user has written in a Basic subroutine. Other attributes are available for the user to define to further specify the event. Through judicious use of these successive attributes the user can create general event subroutines which can function for a number of events within the simulation. The simulation progresses when the MicroSim Executive routine removes the next chronological event entity and

Proceedings of the 1982
Winter Simulation Conference
Highland * Chao * Madrigal, Editors

82CH1844-0/82/0000-0103 $00.75 © 1982 IEEE

executes the appropriate event logic as specified by the event code. These instructions represent a change to the system by altering variables in the system image. Basically, scheduled simulation using MicroSim proceeds as follows:

```
ENTITIES
     having
          ATTRIBUTES
               cause
                    EVENTS
                         which alter the
                              SYSTEM IMAGE.
```

As an example, consider a simulation to determine the availability of a given system which is subject to failures. An event entity which represents the transition from an unfailed state to a failed state (i.e. a failure event) will have attributes for the failure time, an event code (to reference the failure event logic) and perhaps an equipment identifier. By using the third attribute as an equipment identifier, the user can write a general "failure" event for more than one component. This event will cause the failure of the specific equipment by changing the discrete variable in the system image which represents the status of the equipment. This example will be explored more fully with an actual MicroSim model.

Crossing events are the second type of event found in MicroSim models. Although scheduled events could be used in their place, crossing events are more efficient in certain modeling applications. These events are generated when a continuous variable, represented by a difference equation, crosses a specified value in a specified direction. The MicroSim EXECUTIVE routine detects and calculates the time of the crossing and then calls the appropriate event logic. The event then proceeds like a scheduled event. Crossing events drive the simulation as follows:

```
CONTINUOUS VARIABLES
     represented by
          DIFFERENCE EQUATIONS
               which have crossed
                    DEFINED VALUES
                         cause EVENTS
                              which then change
                                   the SYSTEM IMAGE
```

The shutting down of an oil well because of high inventory is an example of a crossing event. Here the inventory which is represented by the continuous variable I(t), is calculated by the equation:

$$I(t) = I(t_0) + PR*(t-t_0)$$

where t is the current time, t_0 is the time of the previous update, and PR is the pumping rate. This difference equation would be incorporated into the MicroSim model in a special subroutine called DIFFEQ (GOSUB 19000) where the user codes the equation using MicroSim variables. In an initialization step the user defines the crossing events. Parameters of these events are the identifier for the equation used, the value to cross, and the direction of the crossing. After the crossing event has been detected the changes to the system image occur like a scheduled event. In this case the variable PR will most likely be set to zero which, in effect, turns the pump off.

THE MICROSIM SIMULATION LANGUAGE

Because most microcomputers have Basic interpreters available, the MicroSim language was written using Microsoft Basic and runs on a Radio Shack Model II microcomputer. To insure the portability of the package, none of the special features found in Microsoft Basic have been used in MicroSim. Using Basic does present problems that are not found in FORTRAN. Because all of the routines are in "common" a variable naming convention is required. This was solved in MicroSim by declaring that all variables starting with the letters A through N are MicroSim variables and all others are available to the user. This eliminates the possibility of the user accidently using a MicroSim variable name an event routine. A second problem is that subroutines in Basic are distinguished by line numbers. The FORTRAN statement, CALL FILE(1), is more meaningful than the equivalent in Basic, GOSUB 13000. For this reason the MicroSim documentation and this paper refer to the MicroSim subroutines by descriptive names as well as line numbers.

To use the MicroSim package the simulation analyst first loads the MicroSim Basic code into the RAM of the microcomputer. The user then writes the Basic code needed to describe the events of the model as well as other initialization, control, and display routines. Also required is a "Main" program which calls the MicroSim EXECUTIVE routine. This is done by the Basic statement, GOSUB 11000. With the model entered into the computer, the analyst can now run the model. After the Main routine calls the MicroSim EXECUTIVE the simulation proceeds as follows:

1. Initialize all "sizing variables" with defaults and call user routine, INIT1 (GOSUB 20000), to change from default. Sizing variables are used to allocate space for the arrays used in MicroSim.
2. Set up filing structure for the Event file and any other user files.
3. Call the second user initialization routine, INIT2 (GOSUB 20500), to initialize certain MicroSim variables and to file any initial events. The model can allow for user interaction at this point by allowing for parameter input from the keyboard.
4. Display the MicroSim Menu.

The MicroSim menu provides the user with features to allow for verification and debugging. The menu is displayed in figure 1.

```
MICROSIM MENU

1.  DISPLAY USER SYSTEM.
2.  FILE/REMOVE ENTITIES INTO/FROM FILES.
3.  SET TRACE PARAMETERS.
4.  DISPLAY CONTENTS OF FILES.
5.  CLEAR FILES.
6.  REVIEW/UPDATE CROSSINGS AND EQUATIONS.
7.  RESUME SIMULATION.

PRESS NUMBER OF SELECTION AND <ENTER>?
```

Figure 1

The first selection calls the user written routine, TRACE (GOSUB 23000), which is used to display intermediate results of the model and/or to provide for interactive input. The second selection allows the analyst to perform filing functions. The third option is used to set the trace parameters which control the frequency the EXECUTIVE routine calls the user written TRACE routine. The user can select options that call TRACE either after each event is processed, or at specified time intervals, or whenever a logic flag (LF) is set within an event. The last option allows the user to pinpoint the conditions that cause infrequent occurrences which is particularly useful in debugging and in model verification. Option four allows the user to examine the contents of any or all of the files either on the screen or dumped to a printer. The fifth option is used to remove all stored entities from all of the files. Selection six allows the user to review the current and previous status of the difference equations used in the model as well as the conditions define crossing events. The last selection is used to resume the simulation by returning control to the EXECUTIVE routine. Because of these interactive features, MicroSim provides the user with capabilities that are usually found only within large (and expensive) mainframe simulation packages. The user may halt the simulation at any time during execution by pressing the "M" key which will cause the EXECUTIVE routine to return to the Menu. At this point the user can examine all pertinent variables in the simulation. He can also set up tests by filing events directly into the event file and resuming the simulation. This allows for greater confidence in validation of the model without the use of a trace printout.

When option 7, Resume Simulation, is selected, control over the simulation is returned to the EXECUTIVE routine. The EXECUTIVE first checks the event file and records the time of the next event if one is scheduled. It then updates the difference equations by calling DIFFEQ (GOSUB 19000) and determines if a crossing has occurred before the time of the next scheduled event. If so, the simulation is updated to the time of the crossing, otherwise it is updated to the time of the next scheduled event. The EXECUTIVE also determines whether the TRACE routine should be called and when. The simulation proceeds in this manner until the current simulation time exceeds the predefined end of simulation time (Basic variable FT) or the end of simulation flag (Basic variable ES) is set to a positive value, or it is interrupted by the user. When the simulation is ended the EXECUTIVE calls the user routine, FINISH (GOSUB 24000). This routine is used to report results such as statistics collected during the run and the final conditions of the model.

To use MicroSim, the modeler will make use of three types of variables that are used within the MicroSim routines. These are sizing variables, information variables, and control variables. Sizing variables are used to dimension the MicroSim arrays. These variables are defined below:

Sizing Variables

Name	Description
NA	Number of attributes for each entity stored in the MicroSim files.
NE	Maximum number of entities to be stored in all of the files at any one time.
NF	Number of files to be used in the model including the event file.
NK	Number of crossing events defined in the model.
NQ	Number of difference equations defined in the model.
NR	Number of random number streams used in the model.

MicroSim information variables contain information that is pertinent to the status of the model. These variables may be accessed by the modeler but should not be directly changed within an event routine.

Information Variables

Name	Description
CT	Current simulation time.
LT	Time the simulation was last updated.
CK	Used to indicate which crossing event has been detected.
DV	Deviate returned from RANDOM (GOSUB 12000).
EQ(i)	Current value of the ith difference equation.
EL(i)	Value of the ith difference equation at the last simulation update.
FE(i)	Pointer to the first entity stored in file i.
LE(i)	Pointer to the last entity stored in file i.
MN(i)	Number of entities stored in file i.

The control variables are used to control the simulation and act as passing arguments for the MicroSim subroutines.

Control Variables

Name	Description
AT(i)	Attribute buffer used to store an entity before it is filed into the MicroSim files.
AR(i)	Pointer to the ranking attribute for the ith file. All MicroSim files are ranked low value first on this attribute with FIFO as tie breaker.

CR(i,j)	Array of parameters for the ith random number stream.
ES	Setting ES to a positive value in an event will terminate the simulation.
FI	Used as a passing variable for various routines to indicate a particular MicroSim file.
FT	End of simulation time. The simulation will terminate whenever the CT is greater than FT.
KK(i,j)	Array of parameters used to define the ith crossing event.
LF	Causes the EXECUTIVE to call the TRACE routine when set to a positive value in an event.
MF	Causes the EXECUTIVE to return to the MicroSim menu when set to a positive value.
N	Passing argument for filing routines to indicate the location of entity within MicroSim files.

MICROSIM SUBPROGRAMS

The principle MicroSim subprograms that are available to the modeler are listed below.

Name	Line	Description
EXECUTIVE	11000	Routine which controls the progress of the simulation.
RANDOM	12000	Routine that is used to generate a random deviate (Basic variable DV) which can be used in user coded events. RANDOM can generate uniform, exponential, normal, weibull, poisson, erlang, and triangular deviates. The parameters for the ith random number set are initialized in the CR(i,j) array in the INIT2 (GOSUB 20500) routine.
FILE	13000	Allows the modeler to file the contents of the attribute buffer, AT(i), into the MicroSim file defined by FI.
PAUSE	13500	Allows the analyst to program a pause within an event. This allows the user to either continue with the simulation or to return to the menu.
COPY	14000	Allows the copying of the Nth entity from the MicroSim files into the AT(i) buffer.
REMOVE	17000	Allows the modeler to remove the Nth entity from the MicroSim file, FI, without disturbing the contents of the AT(i) buffer.
LOCATE	15000	This routine returns the location, N, of a specific entity within the MicroSim files based on specified criteria. Once located the modeler may use the COPY or REMOVE routines.

USER WRITTEN SUBPROGRAMS

MicroSim requires that the analyst write a number of specific routines depending on the model. All of the routines have defaults within the MicroSim package so only those needed in the model must be written.

Name	Line	Description
INIT1	20000	Used to define all sizing variables which are used in dimensioning of MicroSim arrays. This routine must only be written if the user wishes to change from default.
INIT2	20500	Used to initialize any dimensioned MicroSim variable and all user defined variables. Can also be used to file any initial events into the event file.
DIFFEQ	19000	Used to compute the current value of any difference equations.
EVENT	22000	Is called by the EXECUTIVE and is used to call the appropriate event routine.
TRACE	23000	Used to display or print intermediate values. This routine can be made interactive to allow for user intervention and decision making during the simulation.
FINISH	24000	Used to display or print the summary report for the simulation.

EXAMPLE OF A MICROSIM MODEL

The concepts of MicroSim are best demonstrated with the use of a simple example.

The purpose of this example model will be to determine the expected availability of a component that is subject to failures. Assume that the distribution of the time to failure is distributed as a Weibull with a location parameter of 400 hours and a shape parameter of 3. This yields an expected time between failures, TBF, of 357.36 hours. If the repair time distribution is triangular with a minimum of 5, a mode of 10, and a maximum of 20, then the expected time to repair, TTR, will be 15 hours. Therefore the availability will be:

$$A = \frac{TBF}{TBF+TTR} = \frac{357.36}{357.36+15} = 0.960$$

The simulation should converge to this value after sufficient run time. For this model the run time will be 10000 hours. The user defined variable, ST, will denote the status of the component with 1 representing the unfailed state and 0 the failed state. The states of the model and their respective transition events are shown in figure 2.

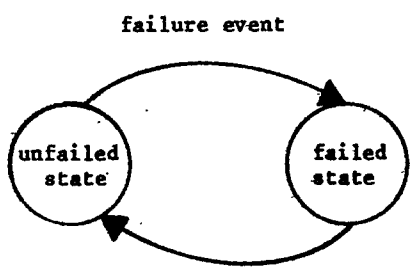

Figure 2

There will be two events defined in the model. The first one will be a failure event. This will arbitrarily have an event code of 2 (1 is reserved for crossing events). The other event will be the repair event with an event code of 3. The logic of the events is as follows:

Failure Event (Event code 2)

1. Shut down component by setting the user variable, ST, equal to zero.
2. Schedule repair event.

Repair Event (Event code 3)

1. Bring up component by setting ST equal to one.
2. Schedule next failure.

A difference equation may be used to compute the total time in the unfailed state. At the end of the simulation, EQ(1) will contain the total time that the component was in an unfailed state. This equation, using both MicroSim variables and the user defined variable, ST, will be:

EQ(1) = EL(1) + ST*(CT-LT)

Then, using CT for the total elapsed time, the simulated availability will be:

A = EQ(1)/CT for CT > 0

In the user routine, INIT2 (20500), the parameters for RANDOM must be defined. These are stored in the array CR(i,j) and are given below:

Event	I mode/mean	CR(I,1)	,2 min	,3 max	,4 shape	,5 type
Failure	1	400	0	1E20	3	4
Repair	2	10	5	20	0	7

INIT2 will also be used to file the initial failure event that starts the simulation.

Below are the listings with remarks of all of the Basic routines that are written by the user. Note that a colon is used as a statement delimiter.

MAIN Program

```
100 REM MAIN ROUTINE
110 GOTO11000  :REM CALL MICROSIM EXEC
```

DIFFEQ Routine

```
19000 EQ(1)=EL(1)+ST*(CT-LT)
19010 RETURN
```

INIT1 Routine

```
20000 NA=2:NE=2:NF=1:NK=0:NQ=1:NR=2
20010 RETURN
```

INIT2 Routine

```
20500 DATA 400,0,1E20,3,4 :REM SET DISTN PARMS
20510 FOR X=1TO5:READ CR(1,X):NEXT X
20520 DATA 10,5,20,0,7
20530 FOR X=1TO5:READ CR(2,X):NEXT X
20540 REM SET STATUS TO ON
20550 ST=1
20560 REM FILE FIRST FAILURE
20570 N=1:GOSUB 12000 :REM CALL RANDOM FOR FAILURE
20580 AT(1)=DV:AT(2)=2 :REM FILL AT BUFFER
20590 FI=1:GOSUB13000 :REM FILE IN EVENT FILE
20600 FT=10000 :REM SET FINISH TIME
20610 RETURN
```

EVENT Routine

```
22000 ON AT(2) GOTO 22010,22100,22170
22010 PRINT"ERROR, AT(2) = 1":STOP
22100 REM FAILURE EVENT
22110 ST=0
22120 REM CALC TIME TO REPAIR
22130 N=2:GOSUB12000 : REM CALL RANDOM
22140 AT(1)=CT+DV:AT(2)=3
22150 GOSUB13000 :REM FILE REPAIR EVENT
22160 RETURN
22170 REM REPAIR EVENT
22180 ST=1
22190 REM CALC TIME TO NEXT FAILURE
22200 N=1:GOSUB12000
22210 AT(1)=CT+DV:AT(2)=2
22220 GOSUB13000
22230 RETURN
```

TRACE Routine

```
23000 CLS: REM CLEAR SCREEN
23010 PRINT"TIME = ";CT;"COMPT = ";ST
23020 PRINT"AVAILABILITY = ";
23030 IF CT>0 THEN PRINTEQ(1)/CT ELSE PRINT 0
23050 GOSUB13500
23060 RETURN
```

FINISH Routine

```
24000 REM PRINT MESSAGE, CALL TRACE AND STOP
24010 CLS:PRINT"END OF SIMULATION":PRINT:PRINT
24020 GOSUB 23010
24030 STOP
```

RESULTS

The final output is shown in figure 3.

```
END OF SIMULATION

TIME = 10132  COMPT = 1
AVAILABILITY = .968589

HIT ENTER TO CONTINUE.
```

Figure 3

The model and the MicroSim routines used about 12K bytes of memory. The total execution time for the run was 24 seconds.

CONCLUSIONS

MicroSim provides the microcomputer user with a general simulation tool. Although MicroSim does require the user to be familiar with details of the language such as variable names and line numbers, it has features which make the modeling task very efficient. Using the interactive features the analyst can quickly develop, debug, and verify a simulation model. Models developed on microcomputers are portable and can be made very user friendly. The use of microcomputers for simulation is extremely cost effective especially when the model needed is not terribly large and does not draw upon existing data sources. As microcomputers become more abundant so will the simulation applications which utilize them.

REFERENCES

(1) Shannon, Robert E., Systems Simulation the Art and Science, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975

(2) Pritsker, A. Alan B., The GASP IV Simulation Language, John Wiley & Sons, Inc., New York, 1974

(3) Spearman, Mark L., "Dynamic Interactive System Simulation: Simulation as a Tactical Decision Making Tool", Summer Computer Simulation Conference, Seattle, Washington, Aug 1980

(4) The MicroSim User's Guide, Industrial Management Consultants, 15223 Woodhorn, Suite 200, Houston, Texas, 77062, 1981

(5) Kapur, K. C., Lamberson, L. R., Reliability in Engineering Design, John Wiley & Sons, New York, 1977