1981 Winter Simulation Conference Proceedings
T.I. Ören, C.M. Delfosse, C.M. Shub (Eds.)

THE AUTOMATION OF SIMULATION

G. K. Hutchinson
University of Wisconsin-Milwaukee
Milwaukee, Wisconsin 53201

The principles of Computer Aided Design (CAD) have been incorporated in a simulation system to reduce the time and cost to produce simulations and expand the set of potential users to nonprogrammers. Computer Aided Programming for Simulation (CAPS) is an interactive program which queries the user and processes the responses to write a simulation program which, for models in its domain, is guaranteed to be logically correct and execute on the first run. CAPS is based upon the use of activity cycles for decomposition of the problem under study. Activity cycles, which appear to be a super set of Petrie Nets, are simply directed graphs. This technique permits one to do the classic barbership in 10 minutes at a cost of under $1.00

Activity cycles concepts will be described and the CAPS program demonstrated on an ALTOS 8000 micro computer. After the formal session is completed, a spontaneously submitted problem will be simulated on an interactive basis. CAPS writes Extended Control - Simulation Language (ECSL) which is a complete language with outstanding statistical characteristics and unique language features. Recent additions to the ECSL include *FIT, which tests distribution data against the 25 most commonly used statistical functions, and DISPLAY, which enables one to easily develop interactive simulations. CAPS/ECSL will be demonstrated on a personally owned micro computer, but will run on any system with ASCII Fortran IV, backing store, and sufficient care (16K words minimum).

1. INTRODUCTION

The analysis of complex systems generally requires that one have some method of simplifying the situation whether the intent is to better understand the system or to simulate it. There are many techniques for accomplishing this, including the flow of entities, the events that take place, the processes involved, and the behavior patterns of the entities. This latter technique, formally known as the activity cycle approach, has been the basis of much of the English activity in simulation. Dr. A. T. Clementson in teaching simulation using activity cycles at the University of Birmingham noted the similarity of structure of the code for these simulations and, applying the principles of computer aided design (CAD), developed CAPS, Computer Aided Programming for Simulation. CAPS accepts on an interactive basis user developed activity cycle diagrams, which are simply directed graphs, and produces a simulation model which, for problems within its domain, are guaranteed to be logically consistent and to execute on first submittal. The intellectual work of simulation is thus transferred from programming to the development of the activity cycle diagram. Once completed, the remainder of interactive input is essentially a clerical process.

Considerable work has been done by engineers and computer scientists on the development of the theory of Petrie Nets (1) and their use in the study of systems which exhibit conflict and concurrency of distributed control and multiple processes. Petrie Nets appear to be a subset of activity cycles and their similarities will be explored.

2. ACTIVITY CYCLES

The first step in model building is frequently the choice of a rationale for viewing the system, i.e. a world view, whether this is done explicitly or implicitly. The second step is often to simplify the system by breaking it down into smaller and more easily understood subsystems, a process often called decomposition. There are many techniques for this but the end result is always the same, a method of defining the status of the system and

changing the status over time. Probably the most common method in the U. S. is events (SIMSCRIPT) or entity flow (GPSS) but in England and Australia the activity cycle approach is most widely used.

The activity cycle approach is to first determine the entities, or things of interest, in the system. These may be physical components, such as men, machines, CPU's, or seats, or logical components, such as the conditions necessary for an activity to take place - the tide must be high for a large ship to dock. This is not unlike the approach used in other methods. The next step is the behavior patterns of the entities are determined. Basically, there are only two states in which an entity may be, active or idle. Later we will see that these correspond to the Petrie Net places and transitions. Conceptually at least, one can think of these entities as alternating between the active and idle states, either of which may have a zero duration. The implication is that an entity going directly from one active state to another passes through an idle state with no delay. In the same sense, an entity going from one idle state to another passes through an active state of zero duration.

As one analyzes the system, one finds that there may be many entities which have the same behavior pattern. It is usually useful to show the activity cycles in an activity cycle diagram (ACD). Idle states are shown as circles and active states as rectangles. For instance, assume that we are interested in a micro computer which is set up by an operator but which runs by itself until finished. The ACD for the computer might be as shown in Figure 1. If we were to simulate the computer, we could use a marker and move it about the ACD to show its current status. We would need to know the durations of the active states and the rules for going from one state to another. The point is that the physical location of the marker (equivalent to a Petrie Net token) gives the status of the entity.

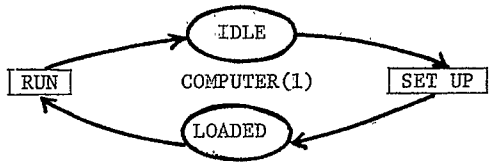


Figure 1. Computer ACD

In the same way, the ACD for the operator can be drawn, as in Figure 2. Note that SETUP occurs in both ACD's. In systems of interest, there may be many entities with the same activity in their ACD's. These activities are known as cooperative activities.

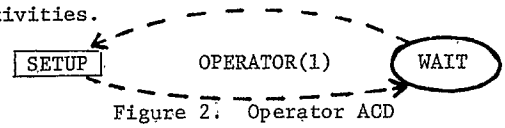


Figure 2. Operator ACD

This means that, in order for such an activity to start, each of the entities used in that activity must be in the immediate predecessor idle state. To illustrate, in Figure 3 the ACD's for the computer and operator are combined. The computer is in IDLE and the operator in WAIT, thus SETUP could begin. The duration of SETUP would be determined by the technology of the system.

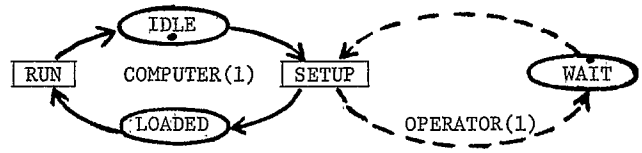


Figure 3. Combined ACD

When SETUP was completed, the markers would be moved into the successor idle states, LOADED and WAIT respectively. Note that RUN could start immediately upon completion of SETUP, as no other entity is required. Activities such as RUN are called bound activities. These occur when their immediate predecessor activity releases all of the entities needed for them to start. These entities will never spend any time in the idle states between these activities and the intervening idle states are known as dummies.

It is interesting to observe the similarities between ACD's and actual systems. In any type of system there are two types of time delays, technological and organizational. Technological delays are those which are determined by the basic technology of the system, i.e. the time that it takes the computer to RUN or to be SETUP. These delays are associated with changing the status of entities, such as changing a computer run from unprocessed to processed, or moving a workpiece from one location to another. Reducing technological delays usually must be done by physical changes to the activity, introducing a faster computer, increasing the speed of a transporter. Often these changes can be accomplished only at high costs of time and money. Organizational delays are those which occur due to a lack of proper coordination. The computer is idle because the operator isn't available. In a typical job shop, the average workpiece spends 95% of its time waiting and work-in-process inventory is the second largest expense.

ACD's highlight the two types of time delays above. Technological delays are the durations of the activities and are input to the simulation. Organizational delays are the result of the manner in which the system is operated and the variance in durations. Often the reason for the use of simulation is to determine the organizational delays, which are the idle states in the ACD's and generally known as queues. Thus in important aspects of complex, dynamic systems, ACD's closely parallel the actual system and Petrie Network Models.

There are many useful aspects of ACD's. Perhaps most important is that it gives a visual representation of the complete logic of the system, much as a PERT Network does. This enchances communications between people concerned with the system and, as will be seen, with computer programs using ACD's as input. Since the logic of the system is complete, they can be used as the basis of programming simulations, either by programmers or by automated systems such as CAPS. Note that the ACD's concentrate on the entities and their behavior. The ACD in Figure 3 could be used for a semi-automatic machine as well as for a computer. In fact it could be used for anything that followed the behavior pattern shown. The logic of ACD's is independent of the number of each type of

entity; thus the same diagram could be used for one operator and ten computers or three operators and five computers. There is no mention of the object being processed, thus the ACD in Figure 3 can be used for many systems. This is not always the case, but frequently one finds that the things being processed do not place a constraint on the system operation and need not be included. The entire focus is on the resources, i.e. the entities.

ACD's also serve as an excellent vehicle for the analysis of systems, again because the complete logic of the system is shown. Cooperative activities become obvious. In only those queues which are immediate predecessors of cooperative activities can entities be forced to incur organizational delays. The activities for each entity are easily determined by tracing its path. Bound activities are also obvious. This is important because they cannot cause operational problems.

Because of the simplicity of the diagrams discussed to this point, queues with multiple exits have not been discussed. However, they are of major importance because they are the only points in a system where management can influence operation thru decision making. Most interesting management problems involve the assignment of scarce resources. The choice of which activity to undertake (selection of an exit path from a queue to an activity) is the assignment decision. The correct decision can reduce organizational delays, often with little or no cost. An example of a more complex system, an advanced batch machining system, is given in Figure 4. The queue TIDLE in the center of the ACD is the queue of idle transporters. An idle transporter may undertake either TWP or GETT as its next activity, a choice which is very important to system operation.

3. THE AUTOMATION OF PROGRAMMING

Since the logic of the system under study is completely given by the ACD, it is a relatively small matter to add the information necessary for simulation. For simulation purposes, the number of each type of entity must be given, often shown in parenthesis after the name. The duration for each activity must be specified either as a constant, a function of the attributes of the entities involved, a sample from a distribution, or some combination of the foregoing. Furthermore, one may wish to determine the order in which entities in queues are chosen to be something other than first-come-first serve, i.e. generally the setting of priorities. Specifiying the above provides the basic inputs for simulation. In addition, one would usually like to control starting conditions and report generation.

Robin Hills (2) showed how the writing of simulations could be automated, using ACD's as input. Alan Clementson (3) noted the similarity of strcture of simulation models and, applying the principles of computer aided design, developed Computer Aided Programming for Simulation (CAPS). CAPS is an interactive language consisting of five dialogs which query the user to assist him in specifying his model and simulation requirements. The first and most important phase is Systems Logic. The user describes the topology of his ACD by giving, for each entity, the ordered list of alternating queues and activities. The second dialog is Arithmetic which requests for each activity in the ACD a statement of the manner in which the duration is calculated. The final three dialogs, Priorities, Initial Conditions, and Reporting, give the user the opportunity to enhance his simulation run by choosing optional facilities.

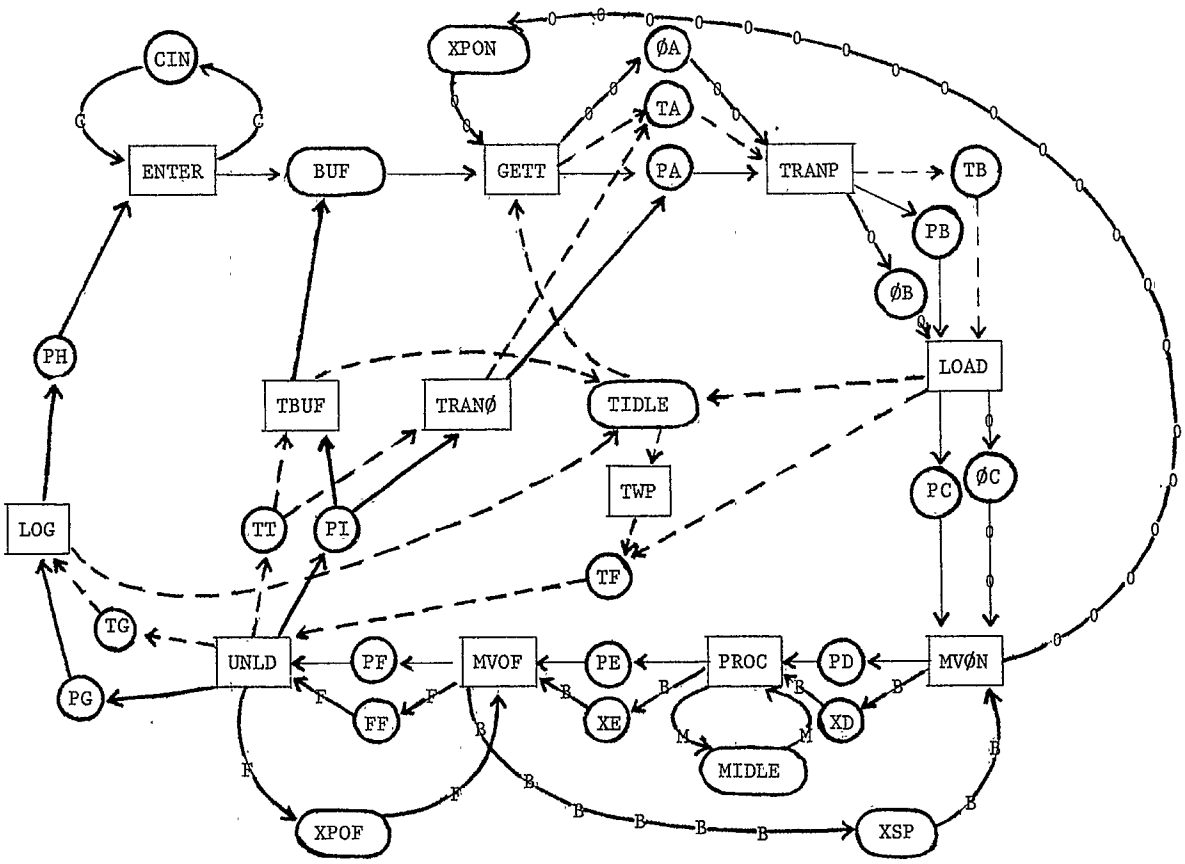
The real point is that the user has completed the intellectual work of the simulation when his ACD is done. The CAPS dialogs can be performed by a knowledgable clerk quickly and inexpensively. For instance, the classical barbershop problem was done in under ten minutes at the VDU, from the start of CAPS to simulation output, at a cost of $0.66 (see appendix). A simple model of an interactive computer network took 23 minutes and cost $3.29 (4).

When the interactive dialog is completed, CAPS writes the user's model in Extended Control and Simulation Language (ECSL). For models in its domain, CAPS guarantees to produce an ECSL (5) program that represents a logically consistent model and that will compile/execute on the first run. Essentially this rapid feedback of model results may alter the traditional procedure of development, validation, and use of a model. With rapid feedback, macro level models can quickly be developed in the traditional manner. Only the most sensitive elements indicated by the macro level model need be expanded for the final micro level model. By applying one's modeling effort in those areas with the greatest potential payoff, one can achieve a more useful model for the same level of effort or the same quality of model for less effort.

4. COMPARISON WITH PETRIE NETS

In the foregoing, several similarities of ACD's to Petrie Nets have been pointed out. Both are not only directed graphs, but bi-partite directed graphs, i.e. the arcs connecting graph elements never go between the same type of elements but, in ACD's, content queues to activities and activities to queues. The queues in ACD's correspond to places in Petrie Nets and the activities to transitions. Petrie Nets have only one type of entity, a token, which moves about the Net. ACD's have multiple types of markers, serving the same purpose, one type for each entity type. Transitions are used to change the status of the system in Petrie Nets, as activities do in ACD's. In ACD's the activities take place over time, a duration, which may be zero or greater than zero.

The conditions for a transition to take place, "fire", is that a token is in each of the immediate predecessor positions. The result of firing is to place tokens in each of the successor positions. There may be differences in the numbers of predecessor and successor positions, i.e. the transition may be a source or sink for tokens. ACD's may have multiple predecessor and



Activities

ENTER	MVOF - Move WP to Off Queue	——— Workpiece
GETT - Get Transport	UNLD - Move WP to Transporter	—C— Arrival Control
TRANP - Transport Workpiece	LOG - WP Completed	- - - -Transporter
LOAD - Move WP to On Queue	TRANØ - Transport between Groups	—O— On Queue
MVON - Move WP to Machine Bed	TBUF - Transport WP to Buffer	—M— Machine
PROC - Process WP	TWP - Pick up WP Move	—B— Machine Bed
		—F— Off Queue

Fig. 4 Activity cycle diagram for an ABMS

successor queues but the rules for choosing entities from them may be very complex. Multiple entities may be required, "batching". The entities may be placed in more than one successor queue depending on the result of the activity. In general, ACD's seem oriented to real world problems of analysis and simulation which Petrie Nets appear to be superior for analytical, theoretical investigations.

5. SUMMARY

ACD's appear to hold much promise as a means of studying and simulating complex, dynamic systems. The automation of the generation of simulation programs has been accomplished through CAPS. CAPS has been successfully implemented throughout the world on any computer with ASCII Fortran IV, backing store, and 16K word memory. In fact CAPS is operational on the Zilog Z80 chip based ALTOS 8000 micro computer.

REFERENCES

Clementson, A.T., (1978), "ECSL CAPS, Detailed Reference Manual," University of Birmingham, Birmingham, England.

Hills, P.R. (1969), "HOCUS--USER's Manual," P.E. Consulting Group, London.

Hutchinson, G.K. (1978), "Computer Aided Simulation for Computer System Studies," 1978 Winter Simulation Conference, Vol. 1, IEEE 78, CH 1415-9.

Hutchinson, G.K. (1975), "An Introduction to Activity Cycles," Simuletter, October.

Peterson, James L. (1977), "Petrie Nets," Computing Surveys, Vol. 9, No. 3, September.

APPENDIX

```
@XQT K*KK.CAPS (Execute the CAPS program)

 EXTENDED CONTROL AND SIMULATION LANGUAGE
 ----------------------------------------
 COMPUTER AIDED PROGRAMMING SYSTEM

 DO YOU WISH TO HAVE INSTRUCTIONAL COMMENTS-
No
 PROBLEM NAME-
Barber
 DO YOU WISH TO START A NEW PROBLEM
Yes

 LOGIC
 -----

 TYPE NAME OF ONE KIND OF ENTITY
Barber,2
 TYPE A LIST OF THE STATES THROUGH WHICH THESE
 ENTITIES PASS.  PRECEDE QUEUES BY Q AND
 ACTIVITIES BY A.
Acut

 IS THIS CYCLE CORRECT
Yes

 TYPE NAME OF ONE KIND OF ENTITY
Customer,25

 TYPE LIST OF STATES AS ABOVE.
Qout, Aarrive, Qwait, Acut, Qout

 IS THIS CYCLE CORRECT
Yes

 TYPE NAME OF ONE KIND OF ENTITY.

 ARE THERE ANY (OTHER) ACTIVITIES WHICH USE MORE
 THAN ONE ENTITY OF A PARTICULAR TYPE
No

 FROM WHAT YOU SAID SO FAR, THE FOLLOWING ARE THE
 MAXIMUM NUMBER OF SIMULTANEOUS REALISATION OF THE
 ACTIVITIES.
 ACTIVITY NUMBER
 CUT       2      LIMITED BY THE NUMBER OF BARBER
 ARRIVE   25      LIMITED BY THE NUMBER OF CUSTOM
 DO YOU WISH TO APPLY ANY LIMITS WHICH ARE BELOW
 THESE
Yes
 WHICH ACTIVITY
Arrive
 WHAT IS THE LIMIT
1
 WHICH ACTIVITY
 NOT MORE THAN 3 OF THE 25 CUSTOM CAN BE ACTIVE AT
 ONE TIME.
 DO YOU WISH TO SEE A SUMMARY OF THE CYCLES.
No
 DO YOU WISH TO MAKE ANY CHANGES IN THE LOGIC
 SECTION
No

 PRIORITIES
 ----------

 ARE THERE ANY QUEUES WHOSE DISCIPLINE IS NOT
 F-I-F-O
No

 THE ORDER OF THE FOLLOWING ACTIVITIES IS UN-
 IMPORTANT
 ARRIVE
 CUT
 DO YOU WISH TO MAKE ANY CHANGES IN THE PRIORITY
 SECTION
No

 ARITHMETIC
 ----------

 AFTER EACH ACTIVITY NAME, TYPE FORMULA FOR ITS
 DURATION
 IF THE DURATION MIGHT BE ZERO, TYPE 0+....
 ARRIVE=
NEGEXP(MAT,XX)
 CUT =
NORMAL(MST,3,XX)

 IN WHICH ACTIVITY IS MAT      EVALUATED
 WHAT IS ITS INITIAL VALUE
22
 IN WHICH ACTIVITY IS XX       EVALUATED
 WHAT IS ITS INITIAL VALUE
1243
 IN WHICH ACTIVITY IS MST      EVALUATED
 WHAT IS ITS INITIAL VALUE
20
 DO YOU WISH TO DEFINE ANY MORE ATTRIBUTES FOR
 ENTITIES
No
 DO YOU WISH TO MAKE ANY CHANGES IN THE ARITH-
 METIC SECTION
No

 RECORDING
 ---------

 WAIT =
5
 OUT  =
0
 DELAY HISTOGRAM RANGES
 WAIT    RANGE=0 TO
100
 WHAT LENGTH OF RUN-IN PERIOD IS REQUIRED
100
 DO YOU WISH TO MAKE ANY CHANGES IN THE RECORDING
 SECTION
No

 INITIAL CONDITIONS
 ------------------

 ARE THERE ANY ACTIVITIES IN PROGRESS
Yes
 ACTIVITY -
Arrive
 TERMINATION TIME =
12
 ACTIVITY
CY\YUT
 TERMINATION TIME =
18
 TERMINATION TIME =
8
 TYPE HOW MANY ENTITIES SHOULD BE IN EACH QUEUE
 LISTED AFTER THE QUEUE NAME
 CUSTOM - 25 ENTITIES
 3 USED BY ACTIVITIES IN PROGRESS
 WAIT -
```

```
2
 OUT -
20
 PLEASE GIVE THE DURATION OF THE SIMULATION
2100
 DO YOU WISH TO MAKE ANY CHANGES IN THE INITIAL
 CONDITION SECTION
No

 SOME ENTITIES ARE APPARENTLY SUITABLE FOR
 AGGREGATION
 I.E. THEY HAVE NO ATTRIBUTES, NO DELAY RECORDING
 AND USE ONLY FIFO Q-DISCIPLINE
 DO YOU WISH ME TO AGGREGATE BARBER
 PARDON - PLEASE ANSWER YES OR NO
Yes
 YOUR PROGRAM HAS BEEN WRITTEN HAVE YOU FINISHED
Yes
 ECSL PROGRAM WRITTEN TO FILE BARBER

****************************************

@COST
FOR Y59184 AT 18:05:38 ON 10-20-80...COST VERSION
80-01...
SPECIFY ITEM:175725
ALL
 ITEM      AMOUNT      SPENT     LIMIT
CAU          0.27      $.02        60
CARDIN        595      $.06
CONNCT          8      $.02
CORE         0.03      $.03
CORESQ       0.57      $.01
FILERQ        255      $.09
FILEWD      78679      $.01
JOB             1      $.02
PRINT           2      $.02       222
PUNCH           0      $.00       200
  TOTAL                $.40   $777.33
MORE?
@P
  $.41          (Use EDIT program to list the ECSL
 READY           program written by CAPS. (Not
 READY           required to do so.))
@EDIT,U BARBER.
EDIT 1.46-10/20/80-18:06:30
EDIT
:P 222
*COMPILE BARBER
 THERE ARE  25 CUSTOM SET    WAIT      OUT
 THERE ARE   1 ZZARRI
 FUNCTION PICTURE NORMAL    NEGEXP
 CUSTOM1  INTO    WAIT      AFTER       12
 TIME  OF  ZZARRI   =       12
 BARBER+  1 AFTER     18
 CUSTOM2  INTO    OUT       AFTER       18
 BARBER+  1 AFTER      8
 CUSTOM3  INTO    OUT       after        8
 RECYCLE
 RUNINZ=  100 AND SWITCH ADD ON AFTER RUNINZ AND
 PREVCLOCK =RUNINZ
 ACTIVITIES 2100
 DURATION= CLOCK - PREVCLOCK
 ADD      WAIT    TO HIST   ZAWAIT      DURATION
 ADD      TIME  IN  WAIT  TO HIST  ZBWAIT (11,5,10)
 ADD      DURATION* BARBER TO AZBARBER
 FOR  ZZARRI WITH    TIME    OF ZZARRI LTOADD
 DURATION TO BZZZARRI
 PREVCLOCK = CLOCK
 BEGIN   ARRIVE
 TIME   OF ZZARRI    LE 0
 FIND    FIRST    CUSTOM A  FROM     OUT
 DURATION=  NEGEXP( MAT     , XX     )
 ADD    1   TO ARRIVE
 CUSTOM A   FROM  OUT  INTO  WAIT AFTER  DURATION
 TIME    OF ZZARRI=   DURATION
 BEGIN   CUT
 BARBER  GE1
 FIND    FIRST   CUSTOM A FROM      WAIT
 DURATION= NORMAL( MST      , 3 , XX    )
 ADD   1 TO CUT
 BARBER- 1 AND    BARBER+ 1   AFTER DURATIOM
 CUSTOM A FROM    WAIT    INTO  OUT AFTER DURATION
 REPEAT
 FINALISATION
 PRINT ' ARRIVE WAS STARTED'ARRIVE'TIMES'
 PRINT ' CUT    WAS STARTED'CUT   'TIMES'
 PRINT'UTILIZATION OF BARBER'+4, (1.0-AZBARBER/
   ( 2.0*(CLOCK -RUNINZ)))
 PRINT'UTILIZATION OF ARRIVE'+4, (1.0-BZZZARRI/
   ( 1.0*(CLOCK -RUNINZ)))
 PRINT /'HISTOGRAM OF LENGTH OF QUEUE WAIT'
    /PICTURE(ZAWAIT )
 PRINT /'HISTOGRAM OF DELAYS AT WAIT' /PICTURE
    (ZBWAIT )
 DATA
BARBER    0
WAIT   4TO       5
OUT    6TO   *
MST      20
XX     1243
MAT      22
 END
*STOP
***TOP OF FILE***
:@COST
NOTHING CHANGED, NOTHING FILED
FOR Y59184 AT 18:07:51 ON 10-20-80...COST VERSION
80-01...
SPECIFY ITEM:
@P
  $.51 (Cost up to now)
  READY
  READY (Execute the ECSL program)
@XQT K*KK.ECSL
 TYPE NEXT CONTROL CARD NOW
@ADD BARBER.  (Add the file with your program in
               it - note period!_

      6308 SPACES AVAILABLE FOR LISTS
 TYPE NEXT CONTROL CARD NOW
 ARRIVE WAS STARTED    104 TIMES  (Execution
 CUT WAS STARTED       104 TIMES   Results)
 UTILIZATION OF BARBER  .5277
 UTILIZATION OF ARRIVE  .9665

 HISTOGRAM OF LENGTH OF QUEUE WAIT
 CELL FREQUENCY
    014191****************************************
************************************************
    1 190**********
    2 111*******
    3  39**
    4  52***
    5  55***
    6  42***
    7  16*
    8   4

HISTOGRAM OF DELAYS AT WAIT
CELL FREQUENCY
    5  60****************************************
```

```
   15  20**********************
   25   5*****
   35   5*****
   45   5*****
   55   4****
   65   5*****

TYPE NEXT CONTROL CARD NOW  (Running took 1 min.
                            14 sec. - cost $0.26;
END OF ECSL PROGRAM         ..total time to
********************        results w/o printing -
                            9 min. 27 sec. - cost
                            $0.66!)
@COST

FOR Y59184 AT 18:09:05 ON 10-20-80...COST VERSION
80-01...
SPECIFY ITEM:
@P
  $.77 - .51                (The remaining output
  READY                      is reruns of problem
  READY                      with more arrivals of
@EDIT,U BARBER.              customers)
EDIT 1.46-10/20/80-18:09:24
EDIT
:40
  PRINT/'HISTOGRAM OF LENGTH OF QUEUE WAIT'
  /PICTURE(ZAWAIT )
:50
EXECUTE
```