AN INTERACTIVE LANGUAGE FOR DISCUTE

DIGITAL SIMULATION

Dr. G. K. Hutchinson
University of Wisconsin
Milwaukee, Wisconsin

Introduction

Although difficult to prove, many feel that a major deterant to the use of simulation as a practical problem solving tool is the difficulty and length of time involved in designing, programming and verifying simulation models. This paper discusses CAPS, Computer Aided Programming for simulation, a simulation system developed at the University of Birmingham by Dr. A. T. Clementson. CAPS uses the principles of computer aided design to assist a user in describing his problem through an interactive dialog. CAPS then writes the user's model in ECSL, Extended Control Simulation Language. The model generated is guaranteed to be logically consistant and the program will execute on the first run. The result is a substantial reduction from the time of problem definition to the point where simulation output is available for decision making.

The CAPS/ECSL simulation package is based upon the decomposition of the system under study using activity cycles, rather than events processes or work flow. Activity cycles have been used as a basis for systems analysis for some time in England and were used as the foundation upon which Mr. R. Hills built the HOCUS simulation language. There are many indications that ECSL is the most popular simulation language in England, even before the availability of CAPS. The purpose of this paper is to show the ease with which simulations can be performed using the CAPS/ECSL system.

The first section contains an introduction to activity cycles and discusses the CAPS dialogs with an example. The appendix contains the actual CAPS dialog for generating a model of the system, a listing of the ECSL code generated by CAPS, and the results of executing the code. Interested readers should be able to use CAPS after reading the paper.

An Introduction to Activity Cycles

Before simulation activities can begin, some method must be chosen for breaking down a complex system, such as a store or factory, into smaller and simpler subsystems for ease of manipulation and understanding. This process is labeled decomposition and activity cycles is one of the methods used. For the purposes of this paper, a system is considered to be composed of entities, things which we wish to talk about and whose behavior we wish to describe as time advances. In a factory, the entities might be men, machines and jobs. In a store, they might be customers, clerks, and helpers. These entities may have attributes which distinguish and describe them. Customers might have budgets and number of items. Clerks might have check-out rates and skill levels. Helpers could be described by pay and performance rates.

The basic step in decomposing a system under study is to identify the entities of interest and group them into classes having similar or identical behavior patterns. These patterns are determined by observing that entities have two possible states, active and idle. Conceptually, it is useful to think of the entities as alternating between states of activeness and idleness, even if the time spent, or duration, of one of the states is of zero length. For instance, a customer might have the activity cycle shown in Figure 1, where active states are shown as oblongs and idle states, or Queues, as circles.

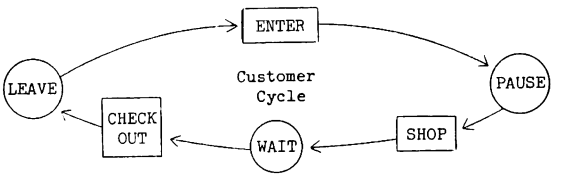

Figure 1. Customer Cycle

The queue, PAUSE, might have zero duration, as the customer might start the activity, SHOP, immediately begin upon the completion of the activity, ENTER. It appears that the customer will immediately re-enter the store upon completion of the activity, CHECKOUT. Actually this merely reflects the fact that the diagram must be drawn so as to close the cycle for each entity, a requirement of CAPS. The implications of this will be discussed.

In most systems of interest, entities that are of importance will usually spend some time in the queues, because the activity for which they are queued requires more than one entity before it can be undertaken. These activities are known as cooperative activities. For instance, the activity CHECK OUT might require a clerk. Assuming the only activity for the clerk is CHECK OUT, the activity cycle would be as shown in Figure 2.

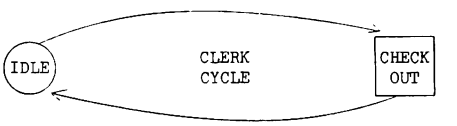

Figure 2. Clerk Cycle

The basic rule for cooperative activities is that each of the entities required by the activity must be in its immediate predecessor queue before the activity can begin. Thus, if a clerk is in IDLE and no customer is in WAIT, the clerk will spend at least a unit of time in queue.

The other major type of activity is the bound activity, which requires a single entity. For example, ENTER and SHOP are bound activities as shown in Figure 1. A customer finishing CHECK OUT passes through LEAVE without the passage of time and begins ENTER, an unrealistic situation (under most circumstances) which will be corrected.

It is usually useful to integrate the activity cycles for the various entities, for the store--the activity cycle diagram is shown in Figure 3.

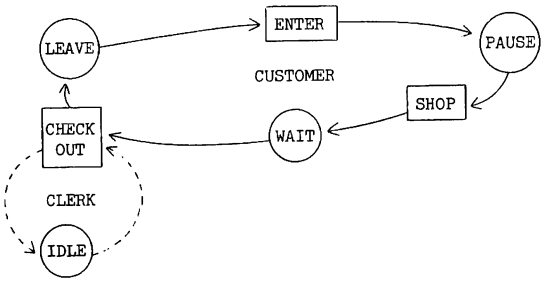

Figure 3. Combined Clerk and Customer Cycles

It is important to note that this diagram contains the complete logic of the store system as we now know it, based on two entities types, customers and clerks. The activity cycle diagram is independent of the number of customers or clerks. Thus, the diagram is equally applicable to a supermarket or a corner shop. The complexities associated with quantities of entities interacting disappear and the analyst can concentrate on the behavior of classes of entities.

The unrealism of customers immediately returning to shop after leaving was previously noted. It is easiest to visualize the queue LEAVE as being a pool of customers, serving as both a source and sink while fulfilling the requirement of closing the activity cycle. This presents the problem of restraining the customers from immediately entering the store after departing. This can easily be done by introducing a logical entity, ARRIVES, which has the cycle given in Figure 4.

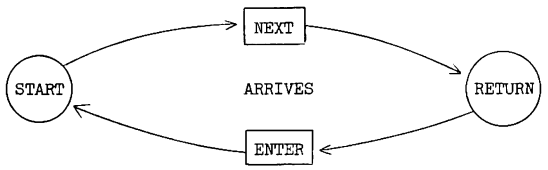

Figure 4. Logical Cycle Limiting Customer Entry

Now ENTER is a cooperative activity that can't begin until an arrives is in START and a customer is in LEAVE. The logical entity, arrives, serves as a metering device for customers coming into the store, the rate being determined by the cycle duration of arrives. If ENTER is given a duration of 0, then the cycle time for arrives is the duration of NEXT. Choosing the distribution of NEXT sets the pattern of customer arrivals or, in queuing terms, the birth rate.

CAPS Dialogs

The basic input to CAPS is the logic of the activity cycle for each entity. These are specified, upon request by CAPS, by giving, for each entity, the alternating queues--proceeded by a Q--and activities--preceeded by an A. CAPS performs many logic and consistency tests as the user supplies these cycles, pointing out the consequences of the user's model and inconsistencies (see Reference 4 for details). In fact, CAPS will not allow a user to proceed until a logically consistant model has been specified. Users are often surprised by CAPS's ability to point out shortcomings in their models, such as: "SHOP is a bound activity (it will start immediately upon completion of the preceding activity)", "No more than 3 of the 10 CUSTOMERS can be active at one time", or most devastating "Your problem does not require simulation, the static solution is...".

To complete the information needed for simulation purposes the following categories of user input must be given:

1. The duration of each activity
2. The queuing disciplines followed by each queue
3. The starting conditions
4. The system recording functions.

Activity durations may be a constant (10), a random

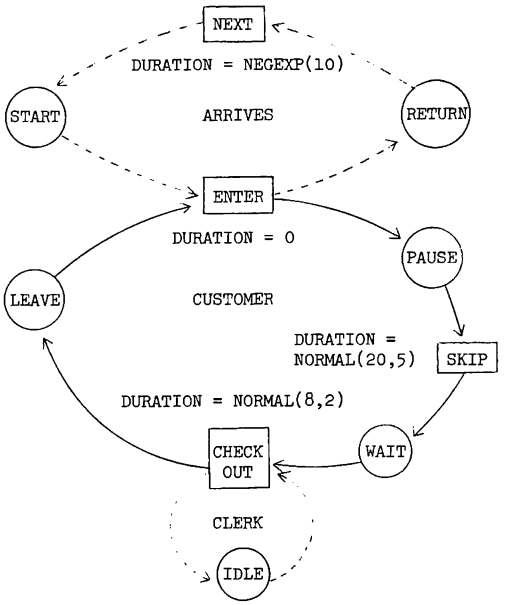

Figure 5. Store Activity Cycle Diagram

Queuing disciplines are assumed to be first-in first-out unless otherwise stated. Other disciplines, such as last-in first-out, random, or maximum of an expression, are readily available. For instance, the clerk might choose the next customer for CHECK OUT to maximize the expected tip.

The starting conditions for the simulation are usually chosen to avoid the transient conditions associated with starting conditions of "empty and idle". This is easily accomplished by indicating the activities in progress and their completion times. All entities which are not involved in activities in progress must be placed in appropriate queues and the length of the simulation stated.

Simulations run with CAPS written programs automatically provide the user with a count of the number of each activity started. In addition, the user can specify the recording of the length and wait time distributions for any queue in which entities may spend time.

The final user input is the duration of the simulation run. The example chosen is simple, by design, to illustrate the basic capabilities of CAPS. Interested readers will note that the system being simulated could be solved without simulation.

The appendix contains a listing of the actual CAPS dialog, with comments; a listing of the ECSL code written by CAPS; and the results of executing the code. The computing system used was the University of Wisconsin's Univac 1110 and the elapsed time from sign on to completion was 30 minutes. The CAPS/ECSL System is operational on a world-wide basis on any computer having a 16K core, backing store, and an ASCII Fortran IV compiler.

The generation of this code for the model in ECSL is very important, for it gives the analyst the ability to easily modify the model generated by CAPS to incorporate features inconvenient to handle in CAPS or beyond the domain of CAPS. Studies at the University of Birmingham indicate that even for such cases, CAPS can generate about 90% of the final code of a model. They have also found that it takes nearly 10 times as many Fortran, as compared to ECSL, statements for a given problem.

ECSL is a complete simulation language with power equivalent to SIMSCRIPT and SIMULA. It is the most widely used simulation language in England, perhaps because of their heavy use of activity cycles for systems decomposi-

tion. It is a completely structured language with built-in trace and debugging facilities. It has full set operators, short assignments (X+1 not X=X+1), implicit subscripts and full statistical capabilities, including independent random number streams and antithical series.

Conclusions

The ability of the CAPS/ECSL system to provide simulation output in a short time frame has been illustrated for a very simple problem. The use of activity cycles as a basis for system decomposition, for input to CAPS, and for ease of communication of the logical relationships of a system has been demonstrated. The CAPS/ECSL system appears to hold promise as a vehicle for making simulation a practical problem solving tool and as a basis for teaching the use of simulation in realistic environments.

References

1. Clementson, A. T., "Computer Aided Programming for Simulation", Univeristy of Birmingham.
2. "Extended Control and Simulation Language - User's Manual University of Birmingham.
3. Hutchinson, G. K., "An Introduction to Activity Cycles. Simuletter, October, 1975.
4. Hutchinson, G. K., "An Introduction to CAPS", Simuletter, October, 1975.

Appendix

I. CAPS Dialog

The following is a reproduction of the actual CAPS interactive dialog with the University of Wisconsin Univac 1110. CAPS output is in capitals and user responses in lower case. The symbol (CR) is used for carriage return, frequently used as a signal to CAPS of the end of a processing subsection. The dialog took place in 15 minutes on a 30 cps T1 terminal.

```
COMPUTER AIDED PROGRAMMING - SIMULATION***C A P S***11/21/75

 UNIVERSITY OF WISCONSIN.CONTACT G.HUTCHINSON 963-4274

 DO YOU WISH TO HAVE INSTRUCTIONAL COMMENTS-
Yes
 DURING THIS DISCUSSION YOU WILL BE ASKED FOR A NUMBER OF
  LISTS.
 WHEN A LIST IS COMPLETE A BLANK LINE SHOULD BE ENTERED.
 IF WHEN TYPING YOU MAKE ERRORS, THESE MAY BE CORRECTED
 BY BACKSPACING (ERACES LAST CHARACTER) USE THE CTRL+H KEYS
  OR ERACE FIELD (ERACES LINE) USE THE CTRL + X KEYS
 WHEN A LINE IS COMPLETE PRESS EOF.
 PLEASE NOTE-ONCE EOF HAS BEEN PRESSED
 IT MAY NOT BE POSSIBLE TO MAKE CORRECTIONS IMMEDIATELY
  THE DISCUSSION IS IN FIVE SECTIONS
 LOGIC-PRIORITIES-ARITHMETIC-RECORDING-INITIAL CONDITIONS
 AT THE END OF EACH SECTION IT IS PPOSSIBLE
 TO RETURN TO THE BEGINNING OF ANY EARLIER SECTION

 DO YOU WISH TO START A NEW PROBLEM
Yes
 PROBLEM NAME -
Shopping

  TYPE NAME OF ENTITIES
Customer
 HOW MANY
12
 TYPE A LIST OF THE STATES THROUGH WHICH THESE ENTITIES PASS.
 THIS SHOULD CONSIST OF EITHER-
    A) AN ALTERNATION OF QUEUES AND ACTIVITIES,
       STARTINGAND ENDING WITH A QUEUE
 OR B) A LIST OF ACTIVITIES ALONE
 PRECEDE QUEUES BY Q AND ACTIVITIES BY A
qleave
aenter
qpause
ashop
qwait
acheck
qleave
(cr)
 IS THIS CYCLE CORRECT
yes

  TYPE NAME OF ENTITIES
 clerk
 HOW MANY
2
 TYPE LIST OF STATES AS ABOVE
qidle
acheck
qidle
(cr)
 IS THIS CYCLE CORRECT
yes

 TYPE NAME OF ENTITIES
arrives
 HOW MANY
1
 TYPE LIST OF STATES AS ABOVE
qreturn
anext
qstart
aenter
qreturn
(cr)
 IS THIS CYCLE CORRECT
yes

  TYPE NAME OF ENTITIES
(cr)

 ARE THERE ANY 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 ACTIVI-
  TIES
 ACTIVITY NUMBER
 ENTER    1
 SHOP    12
 CHECK    2
 NEXT     1
 DO YOU WISH TO APPLY ANY LOWER LIMITS
no
 ACTIVITY SHOP   APPEARS TO BE BOUND TO ENTER
 I.E.  THE FOLLOWING QUEUES ARE DUMMIES
 PAUSE
 DO YOU AGREE
yes

 DO YOU WISH TO SEE A SUMMARY OF THE CYCLES
yes
 CUSTOM 12 QLEAVE   AENTER Q ASHOP QWAIT ACHECK  QLEAVE
 CLERK  2  QIDLE    ACHECK QIDLE
 ARRIVE 1  QRETURN  ANEXT  QSTART  AENTER  QRETURN

 ENTER  USES 1 CUSTOM 1 ARRIVE
 SHOP   USES 1 CUSTOM
 CHECK  USES 1 CUSTOM 1 CLERK
 NEXT   USES 1 ARRIVE
  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 FOLLOWING ARE BOUND ACTIVITIES
 (A BOUND ACTIVITY IS ONE WHICH WILL ALWAYS START IMMEDI-
  ATELY
 UPON THE COMPLETION OF THE PRECEDING ACTIVITY)
 SHOP

  THE ORDER OF THE FOLLOWING ACTIVITIES IN UNIMPORTANT
 CHECK
 NEXT
```

```
 ENTER
  DO YOU WISH TO MAKE ANY CHANGES IN THE PRIORITY SECTION
no

 ARITHMETIC
 AFTER EACH ACTIVITY NAME, TYPE, FORMULA FOR ITS DURATION
 CHECK =
normal (8 2 xx)
 NEXT +
negexp (10 xc)
 ENTER =
o
 SHOP =
normal (20 5 xd)

 IN WHICH ACTIVITY IS XX    EVALUATED-
 (N.B. IF VARIABLE IS NOT TO BE EVALUATED BY ANY ACTIVITY,
 JUST TYPE EOF)
(cr)
 WHAT IS ITS INITIAL VALUE-
1235
 IN WHICH ACTIVITY IS XC    EVALUATED-
(cr)
 WHAT IS ITS INITIAL VALUE-
321
 IN WHICH ACTIVITY IS XD    EVALUATED-
(cr)
 WHAT IS ITS INITIAL VALUE-
987
 DO YOU WISH TO DEFINE ANY OTHER ATTRIBUTES FOR ENTITIES
no
  DO YOU WISH TO MAKE ANY CHANGES IN THE ARITHMETIC SECTION
no
 RECORDING
 TWO KINDS OF RECORDING MAY BE INCLUDED
 1) LENGTH OF QUEE
 2) LENGTH OF TIME ENTITY IS DELAYED IN QUEUE
 TYPE, AFTER THE QUEUE NAME, WHICH KIND OF RECORDING IS RE-
  QUIRED
 TYPE 0, IF NO RECORDING REQUIRED
 TYPE 3, IF BOTH KINDS ARE REQUIRED
 WAIT  =
3
 LEAVE =
0
 IDLE  =
3
 START =
0
 RETURN=
0
 FOR EACH QUEUE FOR WHICH DELAYS ARE TO BE RECORDED
 SPECIFY THE HISTOGRAM RANGE
 (THIS RANGE WILL BE DIVIDED INTO 10 EQUAL   INTERVALS)
 WAIT    RANGE=0 TO
40
 IDLE    RANGE=0 TO
20
  DO YOU WISH TO MAKE ANY CHANGES IN THE RECORDING SECTION
no

  INITIAL CONDITIONS
 ARE THERE ANY ACTIVITIES IN PROGRESS
yes
 (NOTE-TERMINATION TIMES MUST BE CONSTANTS)
 ACTIVITY -
shop
 TERMINATION TIME =
5
 TERMINATION TIME =
14
 TERMINATION TIME =
22
 TERMINATION TIME =
23
 TERMINATION TIME =
(cr)
 ACTIVITY -
check
 TERMINATION TIME =
4
 TERMINATION TIME =
(cr)
 ACTIVITY -
next
 TERMINATION TIME =
2
 ACTIVITY -
(cr)

  TYPE HOW MANY ENTITIES SHOULD BE IN EACH QUEUE LISTED
 AFTER THE QUEUE NAME
 CUSTOM - 12 ENTITIES
  5 USED BY ACTIVITIES IN PROGRESS
 WAIT   -
1
 LEAVE  -
6
 CLERK  -  2 ENTITIES
  1 USED BY ACTIVITIES IN PROGRESS
 IDLE   -
2
 ONLY  1 LEFT - TRY AGAIN
1
 ARRIVE -  1 ENTITIES
  1 USED BY ACTIVITIES IN PROGRESS
 PLEASE GIVE THE DURATION OF THE SIMULATION
200
  DO YOU WISH TO MAKE ANY CHANGES IN THE INITIAL CONDITION
   SECTION
no
 CHECK ,WHICH YOU HAVE USED AS A NAME, IS AN ECSL KEYWORD
 PLEASE GIVE A REPLACEMENT -
ocheck
  HAVE YOU FINISHED-
yes

 YOUR CAPS GENERATED PROGRAM,IN ECSL IS IN FILE H.

 CAPS AND MACC BID YOU ADIEU.


@ADD,P H*SS.RUNOL


@ADD,P H*SS.RUNOL
READY
READY
FURPUR-MACC 2.04-11/21-12:27
    11 BLOCKS COPIED
COPY   COMPLETED..
@ADD H.

E.C.S.L. SYSTEM - UNIVERSITY OF WISCONSIN



@ADD H.
ADD FILE NOT ASSIGNED OR CATALOGUED
ERRO MODE  ERR-TYPE:  02  ERR-CODE:  05
ERROR ADDRESS:  023522    BDI:  000004
USER DID AN ER EABT$
 REENT ADDR:057627  BDI:200005
@ASG,UP HH.
READY
@COPY HH.,??
@COPY H.,HH.
FURPUR-MACC 2.04-11/21-12:28

H      IS NOT CATALOGUED OR ASSIGNED
FAC STATUS:  400010000000
@PRT,T H.
H      IS NOT CATALOGUED OR ASSIGNED
FAC STATUS:  400010000000
@PR??
@EDIT, U H.
CAN'T ASSIGN INPUT FILE



@H*SS.CAPS

 COMPUTER AIDED PROGRAMMING - SIMULATION***C A P S***
  11/21/75

  UNIVERSITY OF WISCONSIN. CONTACT G. HUTCHINSON 963-4274
```

II. ECSL PROGRAM GENERATED

The CAPS dialog resulted in a program, written in ECSL. The listing of the program follows. Note that there are no "GO TO's", the code in modular, and logic is designated by indentation.

```
E.C.S.L. SYSTEM - UNIVERSITY OF WISCONSIN


*     COMPILE SHOP

E.C.S.L. SYSTEM UNIVERSITY OF WISCONSIN      PROGRAM - SHOP
COMPILED ON 11/21/75                           PAGE   1
  1 THERE ARE 12 CUSTOM SET WAIT LEAVE WITH TIME
  2 THERE ARE 2 CLERK SET IDLE WITH TIME
  3 THERE ARE 1 ARRIVE SET START RETURN
  4 FUNCTION PICTURE NEGEXP NORMAL
  5 HIST ZAWAIT (CUSTOM 0,1)
  6 HIST WWAIT (10, 2, 4)
  7 HIST ZBIDLE (CLERK 0,1)
  8 HIST WIDLE (10, 1, 2)
  9 DURATION= 5
 10 CHAIN
 11   CUSTOM 1 INTO WAIT AFTER DURATION
 12   TIME OF CUSTOM 1 = DURATION
 13 DURATION= 14
 14 CHAIN
 15   CUSTOM 2 INTO WAIT AFTER DURATION
 16   TIME OF CUSTOM 2 = DURATION
 17 DURATION= 22
 18 CHAIN
 19   CUSTOM 3 INTO WAIT AFTER DURATION
 20   TIME OF CUSTOM 3 = DURATION
 21 DURATION= 23
 22 CHAIN
 23   CUSTOM 4 INTO WAIT AFTER DURATION
 24   TIME OF CUSTOM 4 = DURATION
 25 DURATION= 4
 26 CHAIN
 27   CUSTOM 5 INTO LEAVE AFTER DURATION
 28   CLERK 1 INTO IDLE AFTER DURATION
 29   TIME OF CLERK 1 = DURATION
 30 DURATION= 2
 31 CHAIN
 32   ARRIVE 1 INTO START AFTER DURATION
 33 RECYCLE

 34 ACTIVITIES 200

 35 BEGIN RECORD
 36 DURATION=CLOCK-PREVCLOCK
 37 PREVCLOCK=CLOCK
 38 ADD A TO ZAWAIT ,DURATION
 39 ADD B TO ZBIDLE ,DURATION

 40 BEGIN OCHECK
 41 FIND FIRST COLUMN A IN WAIT
 42 FIND FIRST CLERK B IN IDLE
 43 DURATION=NORMAL ( 8 , 2 XX )
 44 OCHECK+1
 45 CHAIN
 46   CUSTOM A FROM WAIT INTO LEAVE AFTER DURATION
 47   ADD -TIME OF CUSTOM A TO WWAIT
 48   CLERK B FROM IDLE INTO IDLE AFTER DURATION
 49   ADD -TIME OF CLERK B TO WIDLE
 50   TIME OF CLERK B = DURATION
 51 REPEAT

 52 BEGIN NEXT

E.C.S.L. SYSTEM  UNIVERSITY OF WISCONSIN     PROGRAM - SHOP
COMPILED ON 11/21/75                           PAGE   2
 53 FIND FIRST ARRIVE A IN RETURN
 54 DURATION=NEGEXP (10 XC )
 55 NEXT +1
 56 CHAIN
 57   ARRIVE A FROM RETURN INTO START AFTER DURATION
 58 REPEAT

 59 BEGIN ENTER
 60 FIND FIRST COLUMN A IN LEAVE
 61 FIND FIRST ARRIVE B IN START
 62 DURATION=0
 63 ENTER +1
 64 ADURATION= DURATION+NORMAL ( 20, 5 XD )
 65 CHAIN
 66   CUSTOM A FROM LEAVE INTO WAIT AFTER ADURATION
 67   TIME OF CUSTOM A =ADURATION
 68   ARRIVE B FROM START INTO RETURN AFTER DURATION
 69 REPEAT
 70 BEGIN COUNT QUEUES
 71 COUNT A IN WAIT
 72 COUNT B IN IDLE

 73 FINALISATION
 74 PRINT/OCHECK WAS STARTED/OCHECK/ TIMES/
 75 PRINT/NEXT WAS STARTED/NEXT / TIMES/
 76 PRINT/ENTER WAS STARTED/ENTER / TIMES/
 77 PRINT//HISTOGRAM OF LENGTH OF QUEUE WAIT /
 78 PICTURE(ZAWAIT )
 79 PRINT//HISTOGRAM OF DELAYS AT WAIT /
 80 PICTURE(WWAIT )
 81 PRINT//HISTOGRAM OF LENGTH OF QUEUE IDLE /
 82 PICTURE(ZBIDLE )
 83 PRINT/HISTOGRAM OF DELAYS AT IDLE /
 84 PICTURE(WIDLE )

 85 DATA
 86 WAIT 6
 87 LEAVE 7 TO 12
 88 IDLE 2
 89 XD 987
 90 XC 321
 91 XX 1235
 92 END
```

III Program Execution

The following is the output from the execution of the program given above. The elasped time from log-in to log-off was 27 minutes. The program generated by CAPS can be saved, modified and rerun to avoid going through CAPS again. Multiple execution runs are easily accomplished to assist the user in experimental undertakings.

```
E.C.S.L.  SYSTEM - UNIVERSITY OF WISCONSIN

*      EXECUTE

E.C.S.L. SYSTEM                    PROGRAM - SHOP        EXE
CUTED ON 11/21/75                    PAGE   1
OCHECK WAS STARTED            18 TIMES
NEXT   WAS STARTED            14 TIMES
ENTER  WAS STARTED            14 TIMES

HISTOGRAM OF LENGTH OF QUEUE WAIT
  CELL  FREQUENCY
     0   187**************************************************
     1     8**
     2     5*

HISTOGRAM OF DELAYS AT WAIT
  CELL  FREQUENCY
     2    16****************
     6     1*
    10     1*
HISTOGRAM OF LENGTH OF QUEUE IDLE
  CELL  FREQUENCY
     0    49************************
     1   151**************************************************
           ************
***************

HISTOGRAM OF DELAYS AT IDLE
  CELL  FREQUENCY
     1     7*******
     3     2**
     5     0
     7     1*
     9     4****
    11     0
    13     0
```