AD-A101 375 FLORIDA UNIV GAINESVILLE DEPT OF INDUSTRIAL AND SYS--ETC F/G 9/2
AN INTERACTIVE PRODUCTION CONTROL TRAINING MODEL FOR A NARF SHO--ETC(U)
JUN 81 J C GILMOUR, T J HODGSON N00014-76-C-0096
UNCLASSIFIED RR-81-8 NL

END
DATE FILMED
8-81
DTIC

AN INTERACTIVE PRODUCTION CONTROL TRAINING MODEL FOR A NARF SHOP

Research Report No. 81-8

by

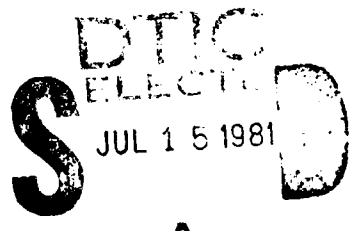
John C. Gilmour
and
Thom J. Hodgson

June, 1981

Department of Industrial and Systems Engineering
University of Florida
Gainesville, Florida 32611

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

This research was supported in part by the Office of Naval Research, under contract number N00014-76-C-0096.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 81-8	2. GOVT ACCESSION NO. AD-A101 375	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Interactive Production Control Training Model for a NARF Shop		5. TYPE OF REPORT & PERIOD COVERED Technical Reptis
		6. PERFORMING ORG. REPORT NUMBER 81-8
7. AUTHOR(s) John C. Gilmour and Thom J. Hodgson		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0096
9. PERFORMING ORGANIZATION NAME AND ADDRESS Industrial and Systems Engineering University of Florida Gainesville, Florida 32611		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 54
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217		12. REPORT DATE June, 1981 13. NUMBER OF PAGES 53
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Production Control

Training

Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

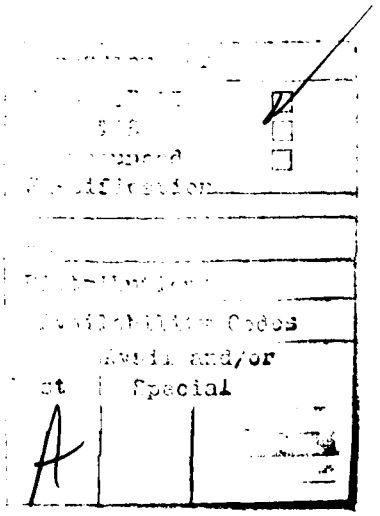
This paper is a report on an interactive production control training model for a Naval Air Rework Facility (NARF) Shop. The system is an interactive shop simulator which allows production control decisions to be made by the user. The objective is to provide a training vehicle for production control decision making. The report includes a "users' manual" and program listing.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS



ABSTRACT

This paper is a report on an interactive production control training model for a Naval Air Rework Facility (NARF) Shop. The system is an interactive shop simulator which allows production control decisions to be made by the user. The objective is to provide a training vehicle for production control decision making. The report includes a "users' manual" and program listing.

REPORT

I. INTRODUCTION

The Naval Air Rework Facility (NARF) at Jacksonville Naval Air Station is an industrial plant with some 3,000 employees. It is one of six similiar NARF's located in the United States. Top management officials are Naval officers and the remainder are civilians. The civilian work force includes engineers, planners, administrators, and mechanics representing over 40 different highly skilled trades.

The NARF mission includes maintenance engineering and heavy (depot) maintenance of military aeronautical items ranging from complete aircraft to spare components. By policy, NARF maintenance on all items is a selective process which involves diagnosis of item condition, updating with latest required changes, and limited maintenance rework to recondition the item for another period of service.

The NARF is organized by support services (Engineering, Quality Assurance, Planning, etc.)and the Production Department. The Production Department, with some 2/3 of the civilian workers, is made up of over 100 different shops. These shops, the hardware producing elements of the plant, are organized: some by product (radio shop), some by process (cleaning shop), and some by function (assembly). The shops form an interrelated network through which products flow as the maintenance process proceeds.

The NARF product workload consists of about 50% aircraft. The aircraft are examined (diagnosed), and disassembled as required to permit shop component processing. As a matter of production policy, the processed components are used to reassemble the aircraft, which is then flight tested. About 15% of the NARF workload consists of aircraft engines (which enter the plant as such) and are overhauled and returned to the Navy supply system for issue and reuse. Another 15% of the workload is made up of miscellaneous spare aeronautical components which have been in use and are sent to NARF for required maintenance. After being reconditioned by NARF, these are returned to the Navy supply system for reissue and reuse. The remaining 20% of NARF workload is of a miscellaneous nature including such unplanned items as aircraft repair (repair of damaged aircraft), customer service (on demand), and field modification.

II. THE NARF SHOPS

This paper deals with a typical NARF shop. The shops under study belong to a class of shops which have the following common characteristics:

1) The bulk of processing on jobs entering each such shop is confined to that shop. This means that a shop functions as a repair station rather than a disassembly, routing, and reassembly shop.
2) A shop of this class is manned by a substantial number of mechanics whose interchangeable skills make it possible to work on a variety of jobs in backlog.
3) Jobs in these shops have individual work content which

is small relative to the total load in the shop. Since work content is small it is also generally true that a single worker at a time processes each job.

4) Shops in this class are not highly dependent on equipment capacity as limiters.

From a production control viewpoint, the shop is composed of three basic elements: the workable backlog area, the nonworkable backlog area, and the processing area (see Figure 1). Jobs arrive at the shop and normally are entered into the workable backlog, to await processing by an available worker. If, for some reason, a job cannot be processed "as is," it is entered into the nonworkable backlog. Usually jobs will be placed in the nonworkable backlog because fo the nonavailability of a component part required for processing or technical data required for processing. Once the nonavailability is satisfied, the job returns to the workable backlog.

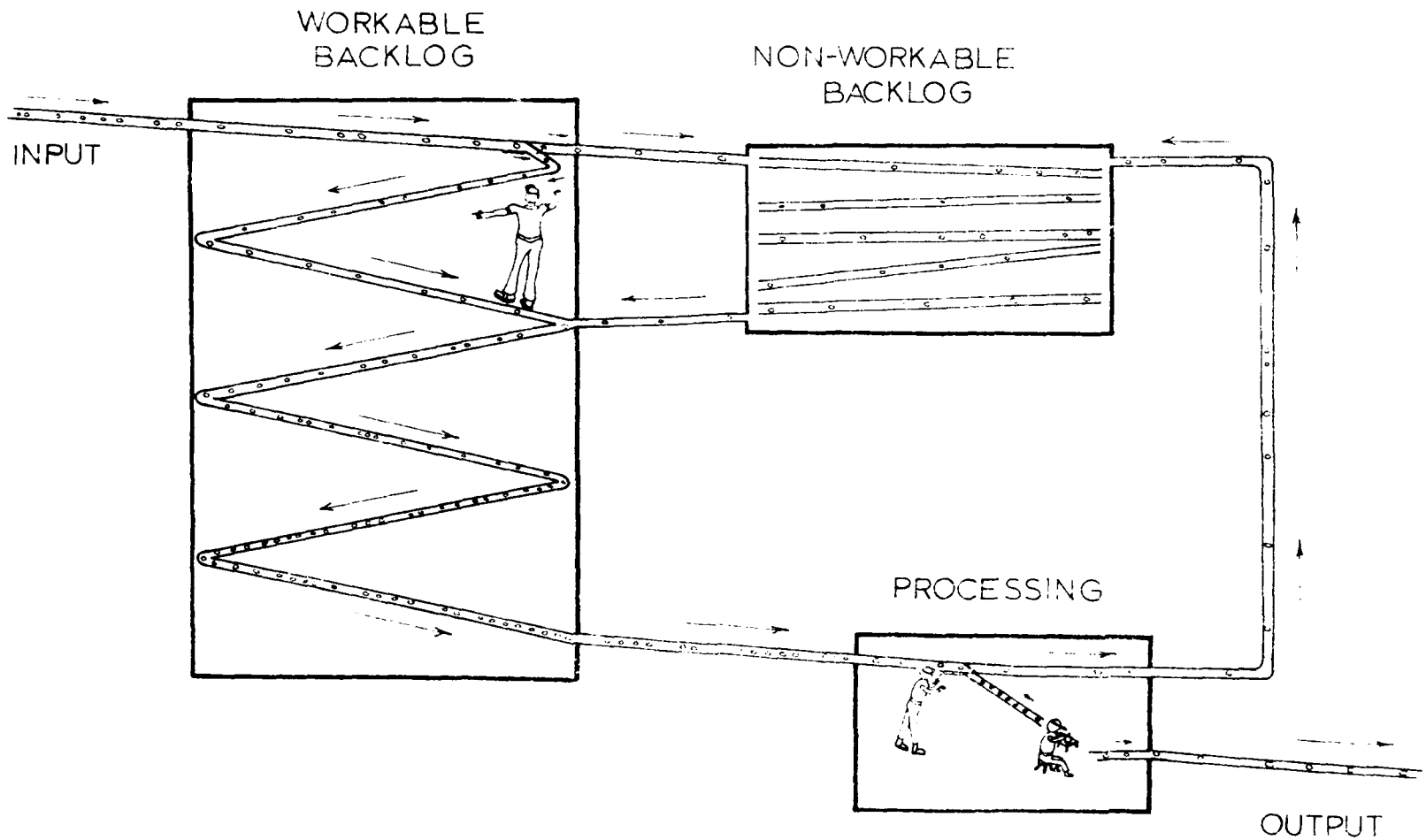
Jobs move from the workable backlog to the processing area as needed. Sometimes, during the initial disassembly of a job, it is determined that an additional component part(s) will be necessary to complete the job. If the component part(s) is not available, the job is entered into the nonworkable backlog (see Figure 1). When processing is completed on a job, it leaves the shop.

Management has at least two methods by which they can exercise control over the shop on a week to week basis. First, they can increase the shop capacity by moving qualified workers into the shop from other shops in the NARF, or they can decrease the shop capacity in the same manner. Second, they can increase shop capacity by working the existing shop manpower overtime. Management control of the shop is exercised with the objective of performing the mission of the shop, and incurring the least possible cost. Clearly, these two objectives are somewhat in conflict. Consequently, any rational control policy is formed by trading of mission performance with cost of operation.

The use of overtime increases the operating cost of the system. In addition to an increased labor rate, overtime work usually is performed at a lower effidiency rate than that of straight time work. It would seem initially that increasing the shop capacity by bringing in more workers would be preferable to overtime, but that also results temporarily in increased costs caused primarily by two factors. First, there is an administrative cost for moving workers. Second, and more important, workers moved into a shop take time to learn their jobs and, consequently, work at a reduced rate for a period immediately after assignment to the shop (assuming that the worker is qualified, this period of lowered efficiency may last one to three weeks).

The amount of work in the system is of concern to management. Low backlog levels may cause inefficient manpower utilization. If the workable backlog is depleted, the workers in the shop will be idle. Another possibility is that if the workable backlog nears depletion, the workers' efficiency will decrease, thereby resulting in an application of Parkinson's Law. In any event the effect on the system is the same. Its efficiency is reduced.

If backlogs are too large, management also has reason for concern. Large backlogs result in increased flow time for jobs



SCHEMATIC OF THE SHOP

FIGURE 1

in the system. If a job has come from an aircraft being reworked at the NARF, an excessive delay of the job in the shop will cause the processing of an aircraft to be delayed. This will result in a real cost for NARF. If a job is for the Navy supply system, an excessive delay of the job will cause a depletion of the Navy inventory which may result in the groundings of a fleet aircraft. In either case, the result is a reduction of the number of operational aircraft available to the Navy.

The shop model program was written to provide a high feedback environment for training production control personnel. Use of the model should allow beginning personnel to get a better understanding of the dynamics of the shop.

III. PROGRAM STRUCTURE

In order to discuss the internal structure of the shop model program it is first necessary to describe the different paths that a job can take through the shop.

1) The job enters the shop and is placed on the workable backlog. It is then worked on, completed, and leaves the shop.
2) The job enters the shop and is placed on the nonworkable backlog for some non-availability. Later when the non-availability is satisfied the job is placed on the workable backlog. It is then worked on, completed, and leaves the shop.
3) The job enters the shop and is placed on the workable backlog. It is then worked on but not completed due to lack of parts or skills. The job is placed on the nonworkable backlog until the parts or skills are available, when it is placed back on the workable backlog. The job is then completed and leaves the shop.
4) The job enters the shop and is placed on the nonworkable backlog. After the necessary conditions have been satisfied it is placed on the workable backlog. It is then worked on but not completed due to lack of parts or skills. The job is placed on the nonworkable backlog. Later the job is put back on the workable backlog, worked on, completed, and leaves the shop.

These paths can be seen in the shop schematic in fisure 1. For simplicity any further reference to the different paths will be by the numbers given above.

Examination of these paths reveal many similarities. This makes it possible to write the program so that the jobs are stored together on the backlogs regardless of the route they are on.

The program stores the backlogs, event list, and the workers all on a single linked list. This requires the storage of the data. If the user wishes to input the information directly the program calls subroutine INPUT. Optionally the program calls subroutine DATA to input the information from a stored data file. In either case the program must then call subroutine PARAM. This subroutine calculates the two needed parameters of the nonworkable delay time distribution from the given mean and standard deviation.

The program achieves steady state conditions by running the

model for twenty weeks(pg. 28). These weeks use a target value determined by the number of men in the shop at the start of the simulation. After the shop has been initialized the program outputs the current shop status (See Appendix C), and prompts the user for information required for the upcoming week. The information required is the induction target value, amount of overtime, and the number of workers transferring in or out of the shop. The workers' efficiency ratings are then updated for the upcoming week. After adding or deleting workers the program is ready to simulate the week.

Jobs are generated and given exponentially distributed job sizes (pg. 48). A certain percentage of these jobs are then placed on the nonworkable backlog and the rest on the workable backlog. When a job is placed on the nonworkable backlog the program calls subroutine DELAY to calculate the delay time. The delay time distribution is a two-stage general erlang distribution. The subroutine computes the delay time so that the job's event time is not during the night. When a job is placed on the workable backlog the program calculates ten percent of the job size and stores this value. This enables the program to check the job after ten percent has been completed. This process continues until the total amount of work generated exceeds the target value.

The program places an event on the event list to mark the end of the regular work day. Idle workers are given jobs to number on which the first and last entries of each seperate list are located. Each entry must include the row number of the next entry on the list. The linked list is composed of six seperate vectors. The information stored in the vectors varies depending upon the particular list and path the job is on. For a complete description of the information stored in the vectors see pg. 24

Every operation performed by the model requires an entry to be taken off a list and then placed back on a list. This is achieved by subroutines TAKE and PUT respectively. Each of these subroutines Entries are always taken off the front of a list. The subroutine must be passed the location of the first and last entry on the list. Subroutine TAKE sets a flag when the last entry of a list has been removed. An entry may be placed on the front, back, or middle of a list. Entries placed in the middle of a list go directly in front of the entry with a lower value in vector EVENT. Subroutine PUT also requires the location of first and last entries of the list. Additionally the subroutine needs the method of entry placement. Both subroutines use an integer vector and two real variables to transfer the information to the main program.

With these two tools and others which will be discussed when needed, it is possible to begin the discussion of how the program operates. All references in this section refer to the logic flow diagram in Appendix B. The program first obtains the values of the various shop parameters, initializes the linked list, sets pointers, and then runs the shop to reach steady state conditions (pg. 18). From this point the simulation proceeds one week at a time, asking for weekly information and outputting the week's results. After the simulation is over the program calculates and outputs the final status of the shop.

Three subroutines are used to input the required shop parameters. If the user wishes to input the information directly

the program calls subroutine INPUT. Optionally the program calls subroutine DATA to input the information from a stored data file. In either case the program must then call subroutine PARAM. This subroutine calculates the needed parameters of the nonworkable delay time distribution from the given mean and standard deviation.

The model is then operated for twenty weeks to bring the shop to steady state conditions (ps. 2A). After the shop has been initialized the current shop status is outputted (See Appendix C). The program then prompts for the information required to simulate the upcoming week. The information required is the induction target value, amount of overtime, and number of workers transferring in or out of the shop. The efficiency rating of each worker is updated by one week. Transfer of workers in or out of the shop occurs at this point (ps. 3B).

Jobs are generated and given an exponentially distributed job size (ps. 3B). A certain percentage of these jobs are then placed on the nonworkable backlog and the rest are placed on the workable backlog. When a job is placed on the nonworkable backlog the program calls subroutine DELAY to compute the delay time. The delay time distribution is a two-stage erlang distribution. This delay time is added to the current clock value and adjusted so that the move time is not when the shop is closed for the night. When jobs are placed on the workable backlog the program computes ten percent of the job size and stores this value with the entry. Jobs are generated until the total amount of work generated exceeds the inputted target value.

The program begins the simulation by placing an entry on the event list to indicate the end of the day. Next the workers that are idle are given jobs from the workable backlog. Move time for these jobs are determined by adding the current clock value to the appropriate amount of the job size. This value is either ten percent or 90 percent depending on whether the job has been in process before (i.e. Job is on paths 3 or 4). The workers efficiency rating is used to adjust the move time to reflect the additional time needed by adjusting workers. When a worker takes a job his idle time is computed by subroutine IDLE. This continues until either no workers are idle or no jobs remain on the workable backlog.

Simulation proceeds by setting the clock equal to the move time of the first entry on the event list. This entry is removed and the program checks to see what the next operation is (ps. 5B). If the job is completed then the program must update the appropriate statistics and counters. Then subroutine NEWJOB is called to give the freed worker another job if one is available. When no jobs are available the worker is flagged idle until one becomes available. Otherwise the job is either going from the nonworkable to the workable backlogs or in process with ten percent completed. In the first case the job is placed on the workable backlog directly if already in process before, and if not then the ten percent value of the job size is determined and stored with the entry on the workable backlog. In the second case the program will send the job to the nonworkable backlog a given percentage of the time, while the rest continue to be worked on until completion. The last

possibility is that the end of day indicator is taken off the event list. Until this occurs the program loops back and takes the next event off the event list.

When the end of the day indicator is taken off the event list the program must do several things (pg 6B). The first thing is to put the end of day indicator for the next day onto the event list. After doing this the program then determines whether or not workers are to work overtime and how much. If no overtime is available the move times of jobs being worked on are adjusted to occur during the next day. This requires that the job first be found (It is not necessarily the first entry on the event list) and then placed back on with the adjusted move time. Subroutine RESORT is used to perform both of these operations. If overtime is available the the job is either worked on for the overtime interval and then adjusted, or completed freeing the worker. When a job is completed during overtime it is treated exactly as before and the worker is given a new job if it is available. If the worker receives a new job during the overtime period its move time is immediately adjusted to the next day. This means that a worker can not complete more than one job in any single overtime period. After all the workers have been dealt with the program starts another day. This continues until five days or a week have been completed.

When a full work week has been simulated the program outputs the contents of the backlogs. Included in this output are the worker idle time, work completed, average flow time, and number of workers in the shop (pg. 7B). This information with the exception of the average flow time is updated continuously and is available at any time. The average flow time is computed at the end of the week. The output is generated by subroutine OUTPUT. The program will then ask if another week is to be simulated.

When no more weeks are to be simulated the program computes the average and standard deviation of the flow time through the shop. Subroutine FINAL is then called and this information along with the total idle time, jobs completed, and hours completed are outputted. The subroutine also computes the shop efficiency and outputs it too. Program execution is then terminated.

APPENDIX A

Shop Schematic and
Memory Storage

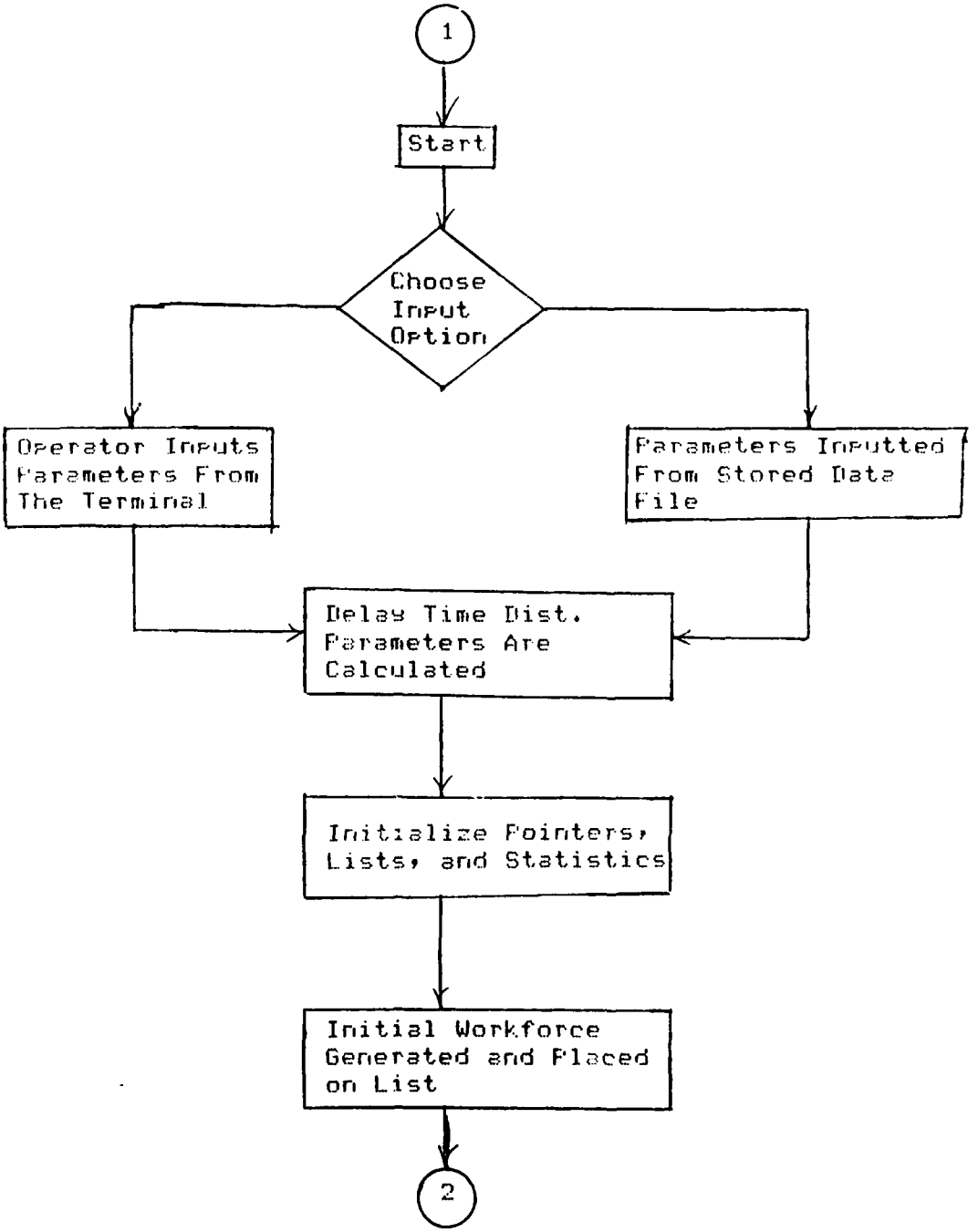
Vector	Contents for entry on workable backlog
EVENT	Ten percent of the job size
FLOW	Value of clock when job was generated
JOBSIZ	Job size
REMAIN	Ninety percent of job size
DIRECT	Row of next entry on Workable backlog
WORK	0 if not being worked on and the worker # if it is

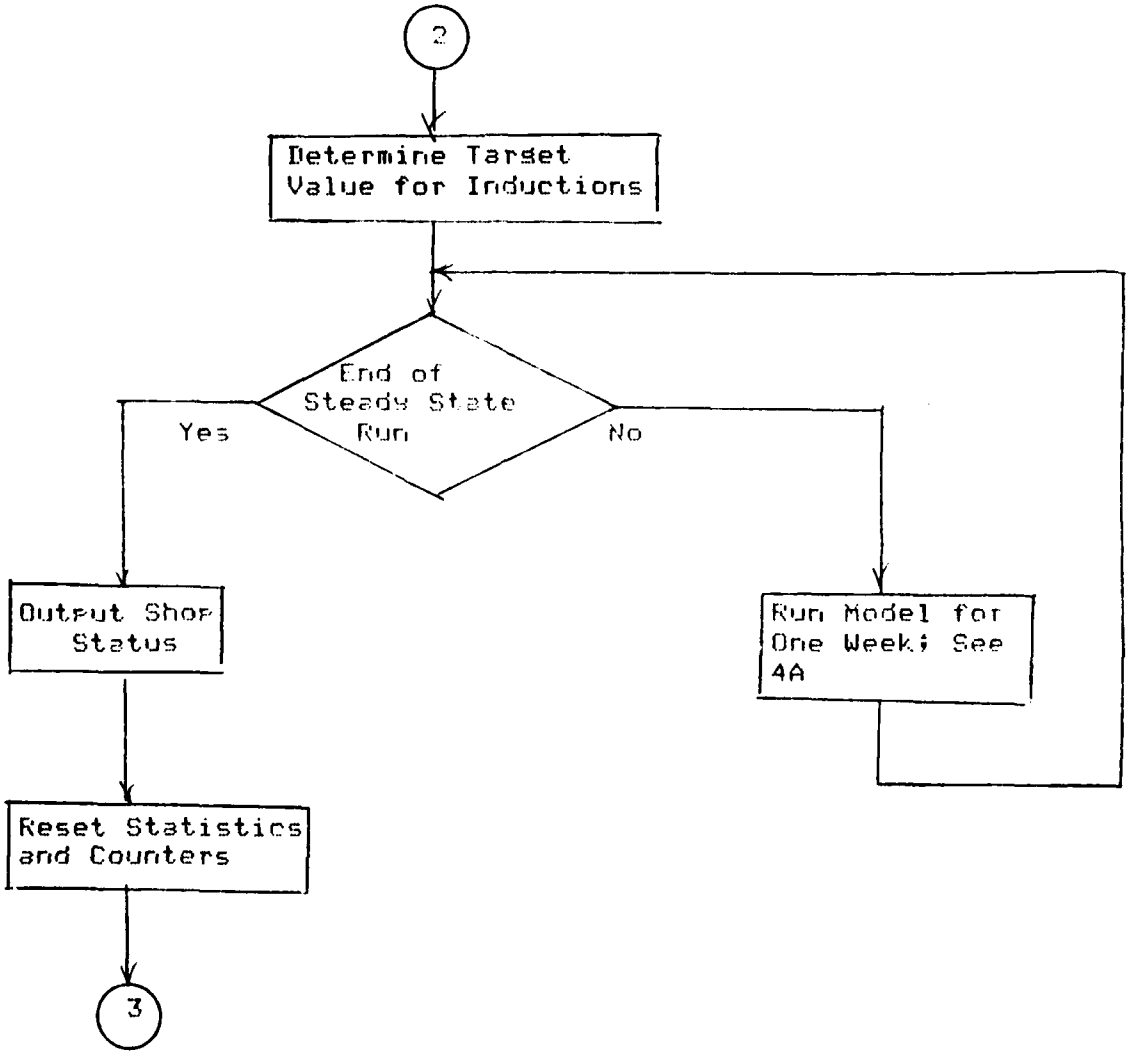
Vector	Contents for entry on Nonworkable backlog
EVENT	Time until Job moves in model
FLOW	Value of clock when Job was generated
JOBSIZ	Job size
REMAIN	----------
DIRECT	Row of next entry on Nonworkable backlog
WORK	-2 if from in process -1 if never in process

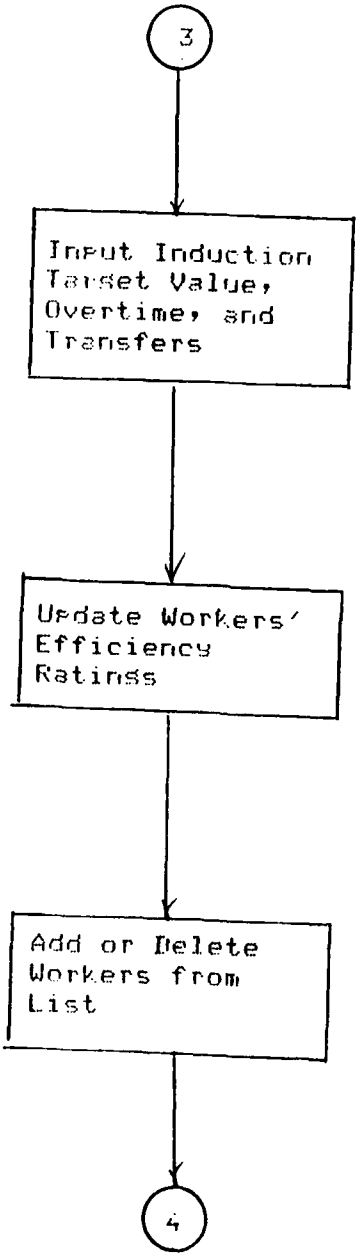
Vector	Contents for entry on Worker list
EVENT	Efficiency ratins for current week
FLOW	----------
JOBSIZ	Number of weeks worker has been in the shop
REMAIN	-1
DIRECT	Row of next entry on Worker list
WORK	----------

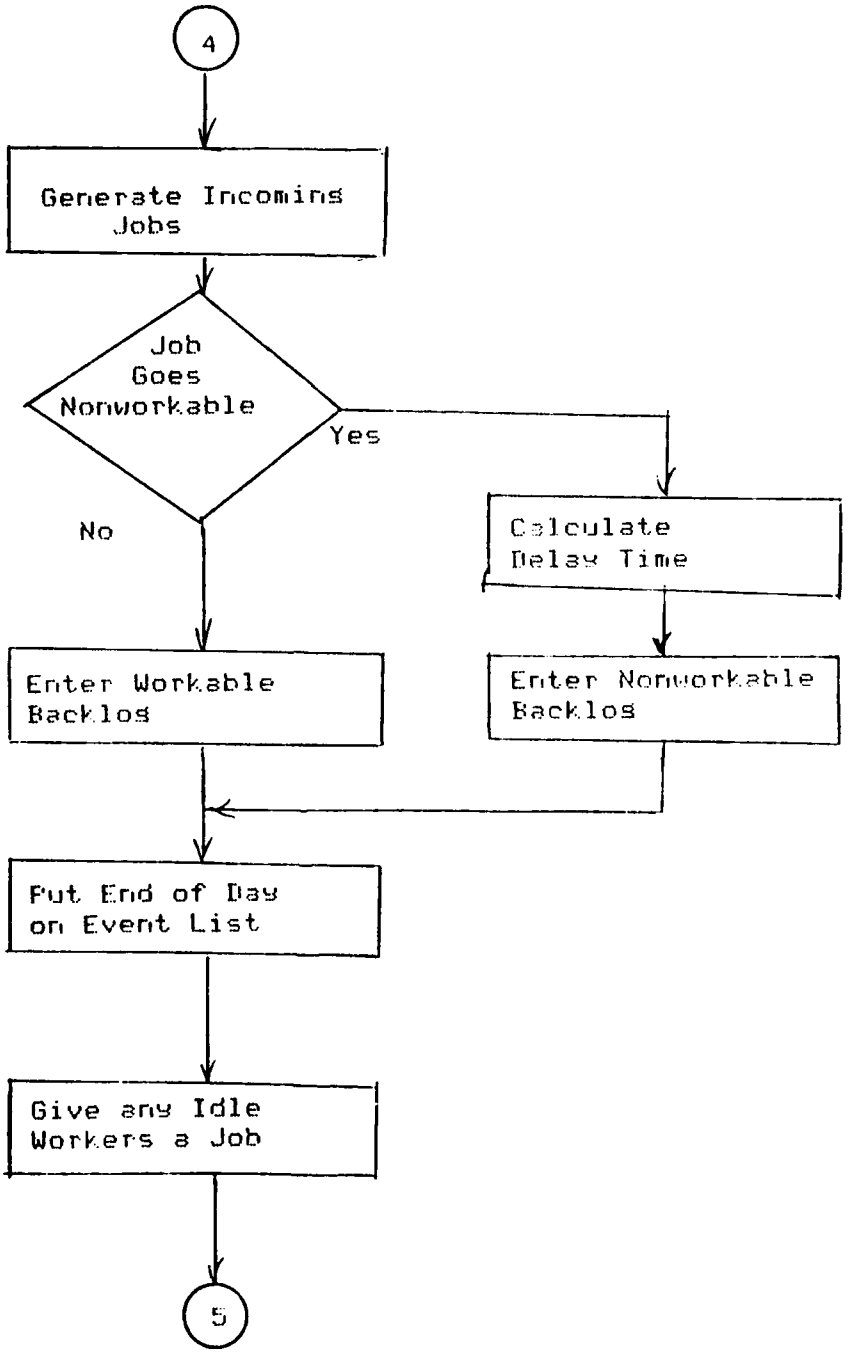
APPENDIX B

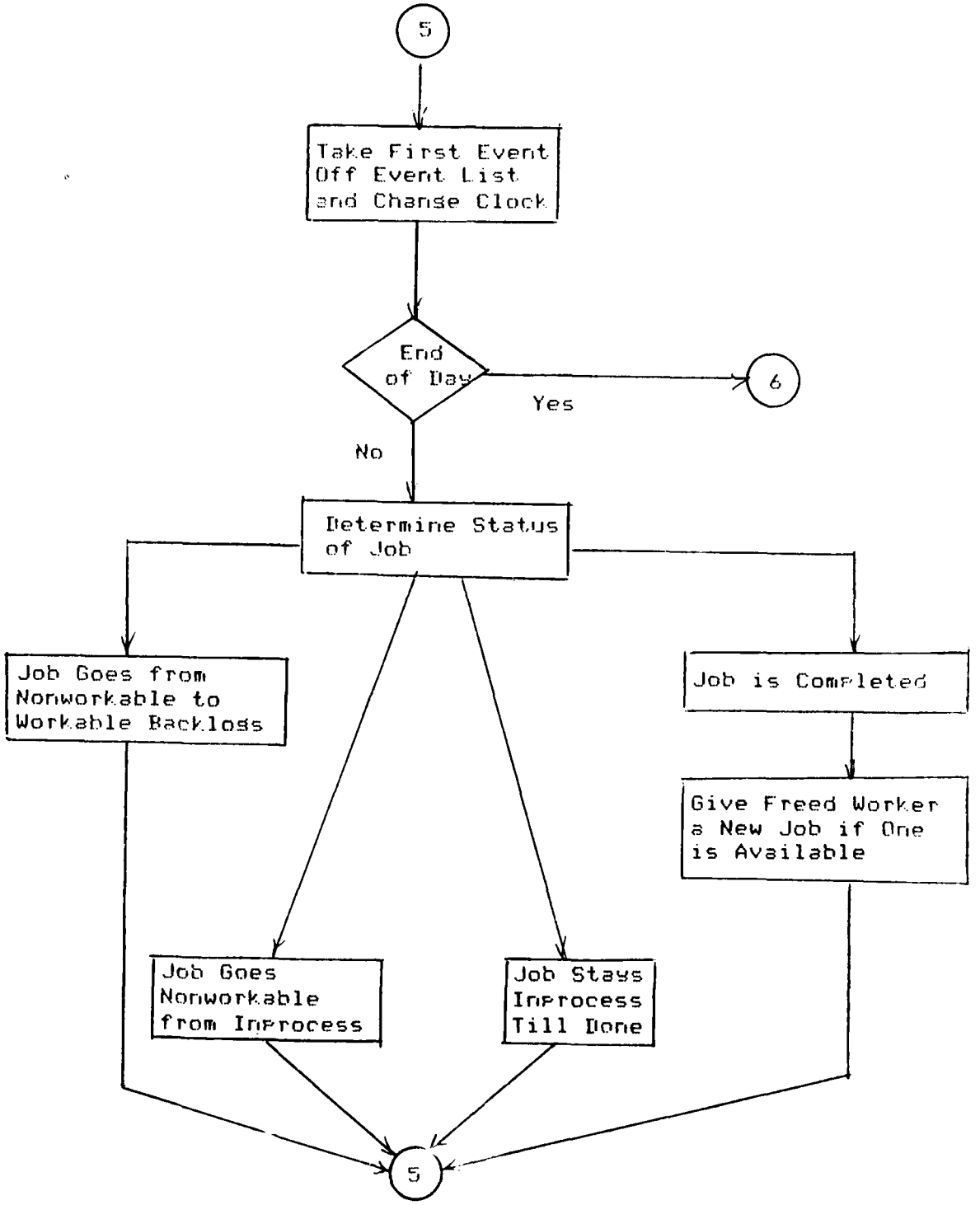
Flow Diagram

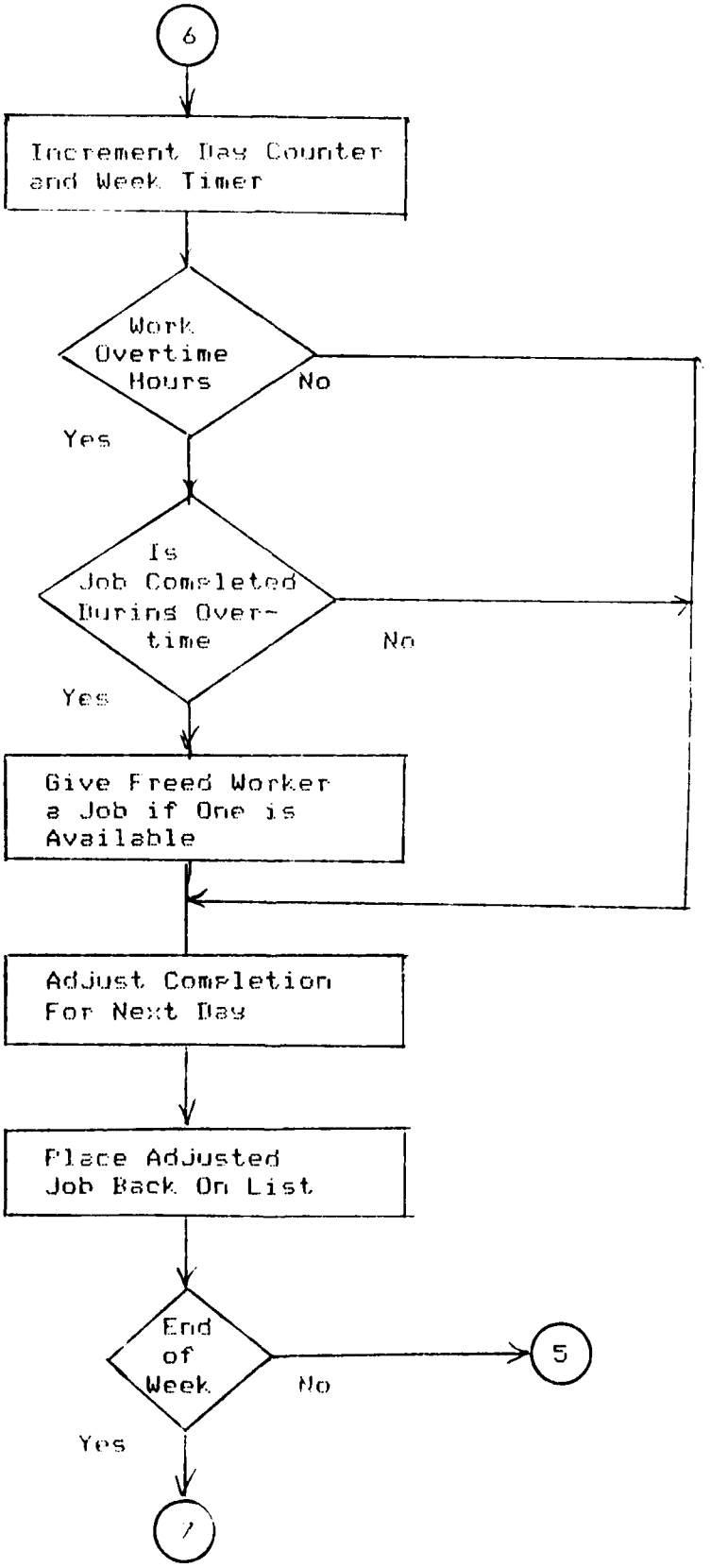

1
Start
Choose Input Option
Operator Inputs Parameters From The Terminal
Parameters Inputted From Stored Data File
Delay Time Dist. Parameters Are Calculated
Initialize Pointers, Lists, and Statistics
Initial Workforce Generated and Placed on List
2

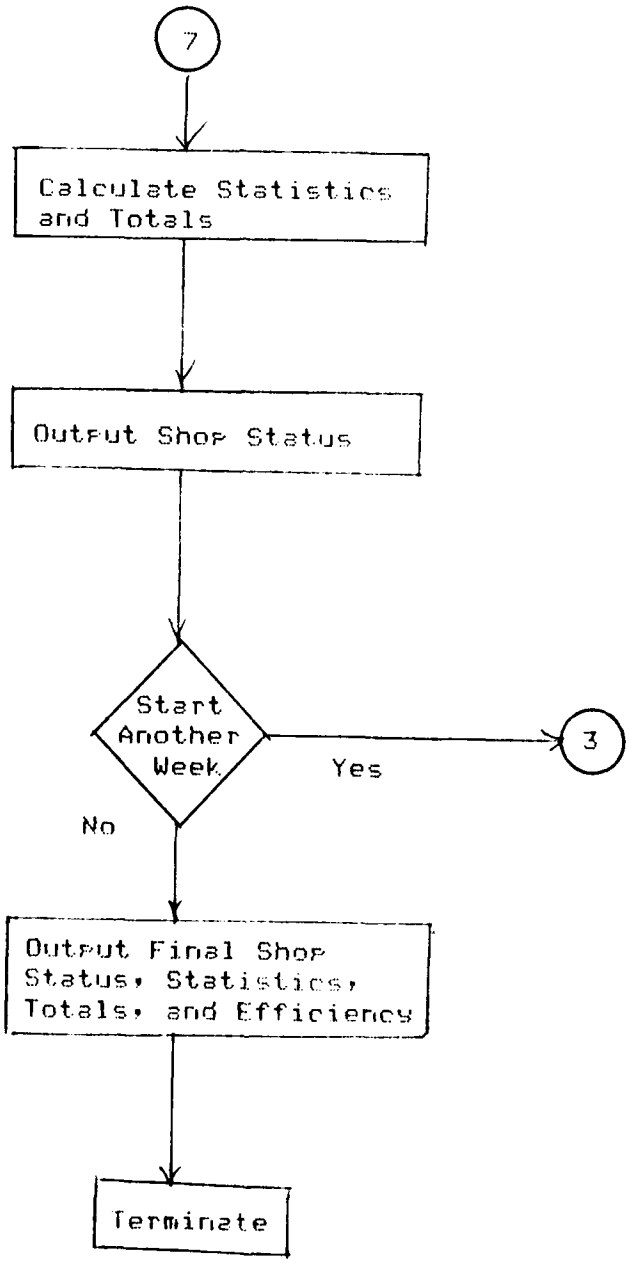

2
Determine Target Value for Inductions
End of Steady State Run
Yes
No
Output Shop Status
Run Model for One Week; See 4A
Reset Statistics and Counters
3


3
Input Induction Target Value, Overtime, and Transfers
Update Workers' Efficiency Ratings
Add or Delete Workers from List
4


4
Generate Incoming Jobs
Job Goes Nonworkable
Yes
No
Calculate Delay Time
Enter Workable Backlog
Enter Nonworkable Backlog
Put End of Day on Event List
Give any Idle Workers a Job
5


5
Take First Event Off Event List and Change Clock
End of Day
Yes
6
No
Determine Status of Job
Job Goes from Nonworkable to Workable Backloss
Job is Completed
Give Freed Worker a New Job if One is Available
Job Goes Nonworkable from Inprocess
Job Stays Inprocess Till Done
5


6
Increment Day Counter and Week Timer
Work Overtime Hours
No
Yes
Is Job Completed During Over-time
No
Yes
Give Freed Worker a Job if One is Available
Adjust Completion For Next Day
Place Adjusted Job Back On List
End of Week
No
5
Yes
7


7
Calculate Statistics and Totals
Output Shop Status
Start Another Week
Yes
3
No
Output Final Shop Status, Statistics, Totals, and Efficiency
Terminate

APPENDIX C

Users' Manual

The shop training program was written for use in training production control personnel. The main objective is to give the user a feel for how the shop operates. It is possible to see the effects of various scheduling decisions on the shop. Program SHOP was intended as a learning tool and should be treated as such.

The shop is composed of three basic elements: the workable backlog area, the nonworkable backlog area, and the processing area (see pg. 16). Jobs arrive at the shop and normally are entered into the workable backlog, to await processing by an available worker. If, for some reason, a job cannot be processed "as is," it is entered into the nonworkable backlog. Usually jobs will be placed in the nonworkable backlog because of the nonavailability of a part or technical data required for processing. Once the nonavailability is satisfied, the job returns to the workable backlog.

Jobs move from the workable backlog to the processing area as needed. Sometimes, during the initial disassembly of a job, it is determined that an additional part(s) is needed to complete the job. If the part(s) is not available, the job is entered into the nonworkable backlog (see Fisure 1). When processing is completed on a job, it leaves the shop.

This manual is a step by step discription for operating the program "SHOP". The program is a simulation model of a typical N.A.R.F. shop. To run the program type in "RUN SHOP". In the following text program instructions and displays are identified by (PROG). Input to the program is required when you see (USER).

At the beginning of the program you must choose an option for entering the initial parameters.

(PROG) DO YOU WISH TO INPUT PARAMETERS YOURSELF? [Y/N]

(USER) If you enter "N" the the parameters will be read from a stored data file.

If you choose to input the parameters yourself then the program will prompt for the information as seen in the next section. If you choose to read the values from a stored file then skip to the section after the second line of asteriks(****).

****

(PROG) INPUT PROBABILITY JOB IS DETERMINED NON-WORKABLE UPON ENTERING SHOP.

(USER) Enter the probability as a decimal number between 0.0 and 1.0.

(PROG) INPUT PROBABILITY JOB GOES NON-WORKABLE AFTER START OF PROCESSING.

(USER) Same as above.

(PROG) INPUT MEAN DELAY TIME IN DAYS.

(USER) Number must include decimal point.

(PROG) INPUT STANDARD DEVIATION OF DELAY TIME.

(USER) Same as above.

(PROG) INPUT INTEGER (ODD) SEED FOR RANDOM NUMBER GENERATOR.

(USER) Seed must be an odd integer between -99 and 999.

(PROG) INPUT AVERAGE JOB SIZE IN STANDARD MAN-HOURS.

(USER) Enter number.

(PROG) INPUT NUMBER OF WEEKS UNTIL TRANSFERED PERSONNEL ADJUSTED AND WORKING FULL EFFICIENCY.

(USER) If worker is at full efficiency during third week then enter a "3".

(PROG) INPUT EFFICIENCY RATING OF TRANSFERED WORKER FOR FIRST WEEK IN SHOP.

(USER) If the worker is expected to work at 80% efficiency then enter "0.8".

(PROG) INPUT EFFICIENCY RATING FOR NEXT WEEK.

(USER) The program will ask for efficiency ratings for all but the last week of the adjustment period. The program sees that the last week in the period is done at full efficiency, so it assigns a rating of 1.0.

****

(PROG) INPUT NUMBER OF WORKERS IN SHOP AT START OF SIMULATION.

(USER) Enter number.

Any error in the input data will result in the question being repeated. If you choose to read the parameters from the data file, the program will still prompt you for the number of men in the shop.

After the parameters are entered the program initializes the shop. This is accomplished by operating the shop for twenty weeks.

(PROG) MODEL BEING INITIALIZED

The initialization process causes a short delay after which the program displays information on the backlogs. The information is displayed in the following format.

(PROG) RESULTS FOR WEEK 0

BACKLOG	NO. OF JOBS	NO. OF HOURS OF WORK
WORKABLE	XXXX	XXXX.X
NON-WORKABLE	XXXX	XXXX.X

WORKERS WERE IDLE XXX.X HOURS.

WORK COMPLETED THIS WEEK TOTALLED XXX.X HOURS.

AVERAGE FLOW TIME XXX.X DAYS.

NUMBER OF WORKERS IN SHOP IS XXX.

The program will then prompt you for the information on the upcoming week.

(PROG) SUPPLY INDUCTIONS TO SHOP FOR UPCOMING WEEK.

(USER) Enter the target value for the week in hours.

(PROG) INPUT OVERTIME HOURS AVAILABLE THIS WEEK.

(USER) Enter the total amount of overtime hours available.

(PROG) INPUT NUMBER WORKERS TRANSFERRING TO/FROM SHOP.

(USER) Enter a positive number if transferring into the shop, and a negative number if leaving shop.

The program will then display the results for the week in the same format as above, and ask if you wish to continue.

(PROG) DO YOU WISH TO OPERATE ANOTHER WEEK? [Y/N]

(USER) Entering a "N" results in output of final statistics and termination of program.

Final results of the simulation are outputted in the following format.

(PROG) FINAL RESULTS FOR XX WEEKS

XXX JOBS WERE COMPLETED TOTALLING XXX.X HOURS OF WORK.

AVERAGE FLOW TIME WAS XX.X DAYS WITH A
STANDARD DEVIATION OF XX.X

WORKERS WERE IDLE FOR XXX.X HOURS

SHOP OPERATED AT XX.X %

There are two conditions which will cause the program to terminated prematurely.

(1) Number of jobs in shop exceeds the maximum number of jobs allowed.

(2) Last worker in shop is transferred out.

It should be noted at this point that the major factor affecting the overall operation of the shop is the status of the workable backlog. Operation of the model for a period of weeks should show that worker idle time increases when the content of this backlog is low. Any increase in idle time will lower the efficiency of the shop. At the same time it is important that the workable backlog does not become to full. This would result in a larger non-workable backlog which in turn increases the storage space needed.

APPENDIX D

Program Code

```
      C*********************************************************************
      C**   PROGRAMMED BY: JOHN C GILMOUR                                  **
      C**                  THOM J HODGSON                                  **
      C**                  INDUSTRIAL AND SYSTEMS ENGINEERING DEPT.        **
      C**                  UNIVERSITY OF FLORIDA                           **
      C**                  GAINESVILLE, FLORIDA 32611                      **
      C**                  TELEPHONE (904)392-1464                         **
      C*********************************************************************
      C**   THE FOLLOWING PROGRAM IS A SIMULATION MODEL FOR A TYPICAL      **
      C**   N.A.R.F. SHOP.  JOBS ARE SCHEDULED BY STANDARD MANHOUR         **
      C**   CONTENT.  THE FOLLOWING IS  LIST OF THE MAJOR VARIABLES        **
      C**   AND THEIR DEFINITIONS.                                          **
      C**     P1 = PROBABILITY JOB WILL GO NON-WORKABLE UPON ENTRY TO       **
      C**          SHOP.                                                    **
      C**     P2 = PROBABILITY JOB WILL GO NON-WORKABLE AFTER PROCESS-      **
      C**          ING HAS BEGUN.                                           **
      C**     LAM1 & LAM2 = PARAMETERS FOR DELAY TIME DISTRIBUTION.         **
      C**     MNJOB = AVERAGE JOBSIZE IN MANHOURS.                          **
      C**     DMEAN = MEAN DELAY TIME.                                      **
      C**     DSTDEV = STANDARD DEVIATION OF DELAY TIME.                    **
      C**     NWK = NUMBER OF WEEKS UNTIL TRANSFERED PERSONNEL ARE          **
      C**           FULLY ADJUSTED.                                         **
      C**     NORMEN = NO. OF REGULAR WORKERS IN SHOP.                      **
      C**     ADDMEN = NO. OF WORKERS TO BE TRANSFERED IN OR OUT            **
      C**     OVRTM = HOURS OF OVER TIME AVAILABLE FOR WEEK.                **
      C**     HOUR = TARGET VALUE FOR SCHEDULING JOBS FOR WEEK.             **
      C**     EMPTY = ROW NO. OF FIRST EMPTY ENTRY.                         **
      C**     WQF = FIRST ENTRY IN WORKABLE QUEUE.                          **
      C**     WQL = LAST     "    "     "      "    .                       **
      C**     NWQF = FIRST ENTRY IN NON-WORKABLE QUEUE.                     **
      C**     NWQL = LAST    "    "     "       "   .                       **
      C**     WRKF = FIRST ENTRY IN WORKER LIST.                            **
      C**     WRKL = LAST    "    "     "    "  .                           **
      C**     I1 & I2 = SEEDS FOR UNIFORM RANDOM NUMBER GENRATOR.           **
      C**     CLOCK = TIME SINCE SIMULATION STARTED.                        **
      C**     TIMER =    "     "    WEEK STARTED.                           **
      C**     MAXLST = MAXIMUM NUMBER OF ENTRIES IN LIST                    **
      C**   FOR EACH J                                                      **
      C**     JOBSIZ(J) = JOBSIZE IN MANHOURS FOR A JOB OR EFFIECENCY       **
      C**                 RATING IF A WORKER.                               **
      C**     EVENT(J) = TIME TILL JOB MOVES IN MODEL.                      **
      C**     REMAIN(J) = TIME REMAINING IN JOB.                            **
      C**     DIRECT(J) = NEXT MEMBER IN LIST AFTER MEMBER J                **
      C**     WORK(J) = STORES WORKER NO. AND OTHER FLAGS.                  **
      C**     FLOW(J) = TIME JOB ENTERED SHOP.                              **
      C**     JOB(J) = DIFFERENT ELEMENTS OF ENTRY TO LIST.                 **
      C**     EFF(J) = EFFICIECNY OF WORKER IN WEEK J AFTER TRANSFER.       **
      C**  OTHER VARIABLES ARE USED IN VARIOUS SECTIONS OF THE PROGRAM      **
      C*********************************************************************
0001        IMPLICIT INTEGER*2(B-O,S-Z)
0002        INTEGER REMAIN,ADDMEN
0003        REAL*8 AVFLOW,SDFLOW
0004        REAL EVENT,P1,P2,MNJOB,LAM1,LAM2,CLOCK,TIME,EFF(6),IDLTM,
          *     HRWK,HRNWK,HRDONE,THRDNE,CHRDNE,TOTIDL,FLOW,
```

```
SHOP,SHOP=SHOP

     *     TOTFLO,PREFLO,SQFLOW
0005       COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
     *       DIRECT(1000),WORK(1000),JOB(3)
0006       LOGICAL*1 ANS,REPLY,YES,NO
0007       DATA YES/'Y'/,NO/'N'/,MAXLST/1000/
0008       CALL ASSIGN(6,'TI:')
0009       CALL ASSIGN(3,'DATA.DAT')
     C*************************************************************************
     C**              INPUT PARAMETERS AND CONSTRAINTS                       **
     C*************************************************************************
0010       TYPE 4
0011  4    FORMAT(10(/),20X,25('*')/20X,'*  N.A.R.F. SHOP MODEL  *'/20X,
     *       25('*')//
     *       '0   THIS PROGRAM IS A COMPUTER MODEL OF A TYPICAL'/
     *       ' N.A.R.F. SHOP.  TO INITIALIZE  MODEL CERTAIN'/
     *       ' PARAMETERS ARE NEEDED.  THE PROGRAM ALLOWS DIRECT'/
     *       ' INPUT OF  PARAMETERS OR READS FROM A STORED'/
     *       ' DATA FILE "DATA.DAT".  TO USE THE PROGRAM'/
     *       ' INPUT THE INFORMATION PROMPTED FOR.  THE FOLLOWING'/
     *       ' INFORMATION IS PROVIDED EACH WEEK: CONTENTS OF '/
     *       ' BACKLOGS, HOURS WORK COMPLETED, HOURS IDLE, AND'/
     *       ' AVERAGE FLOW TIME.  FINAL RESULTS SHOW HOURS WORK'/
     *       ' COMPLETED, HOURS IDLE, AVERAGE FLOW TIME, STANDARD'/
     *       ' DEVIATION OF FLOW TIME, AND SHOP EFFICIENCY.'/)
0012  30   TYPE 6
0013  6    FORMAT(/' WANT TO INPUT PARAMETERS YOURSELF? [Y/N] ',$)
0014       READ (6,35,ERR=30) ANS
0015  35   FORMAT(A1)
0016       IF(ANS.NE.YES) GO TO 40
0018       CALL INPUT(P1,P2,I1,I2,DMEAN,DSTDEV,NWK,MNJOB,NORMEN,EFF)
0019       GO TO 45
0020  40   IF(ANS.NE.NO) GO TO 30
0022       CALL DATA (P1,P2,I1,I2,DMEAN,DSTDEV,NWK,MNJOB,NORMEN,EFF)
0023  45   CALL PARAM(DMEAN,DSTDEV,LAM1,LAM2)
0024       TYPE 46
0025  46   FORMAT(10(/),'  MODEL BEING INITIALIZED',10(/))
     C*************************************************************************
     C**              INITIAL LIST IS EMPTY                                  **
     C*************************************************************************
0026       MXLST1 = MAXLST-1
0027       DO 50 M=1,MXLST1
0028  50   DIRECT(M) = M+1
0029       DIRECT(MAXLST) = 0
     C*************************************************************************
     C**              INITIALIZE POINTERS                                    **
     C*************************************************************************
0030       FLAG = 1
0031       IDLE = 0
0032       NOJOB = 1
0033       NWQF = 0
0034       NWQL = 0
0035       WQF = 0
0036       WQL = 0
0037       WREL = 0
```

SHOP,SHOP=SHOP

```
0038          WRKF = 0
0039          EMPTY = 1
0040          WEEK = 0
0041          JOBWK = 0
0042          HRWK = 0.0
0043          JOBNWK = 0
0044          HRNWK = 0.0
0045          CLOCK = 0.0
0046          HRDONE = 0.0
0047          THRDNE = 0.0
0048          TOTFLO = 0.0
0049          SQFLOW = 0.0
0050          SUM = NORMEN
0051          ITR = NORMEN
        C*************************************************************************
        C**                  GENERATE WORKERS                                   **
        C*************************************************************************
0052          DO 60 K=1,NORMEN
0053          JOB(1) = NWK
0054          TIME = 1.0
0055          JOB(2) = -1
0056          JOB(3) = 0
0057          PNT = EMPTY
0058          CALL PUT(WRKF,WRKL,0,TIME,RFLOW,EMPTY)
0059    60    CONTINUE
0060          HOURS = NORMEN*400
0061          OVRTM = 0
0062          ADDMEN = 0
0063    90    IF(WEEK.EQ.20) FLAG = 0
0065          WEEK = WEEK+1
0066          IF(FLAG.EQ.1) GO TO 200
0068          WEEK = 0
0069          CALL OUTPUT(JOBWK,HRWK,JOBNWK,HRNWK,CHRDNE,WEEK,SUM,
             *            IDLTM,AVFLOW)
0070          RHOUR = HRDONE
0071          JCOMP = 0
0072          JBWK = 0
0073          TOTIDL = 0.0
0074          RQFLOW = TOTFLO
0075          RSQFLO = SQFLOW
        C*************************************************************************
        C**                  INPUT DATA FOR UPCOMING WEEK                       **
        C*************************************************************************
0076    100   WEEK = WEEK+1
0077          IDLTM = 0.0
0078    115   TYPE 116
0079    116   FORMAT(/' SUPPLY INDUCTIONS TO SHOP FOR NEXT WEEK (HOURS) ',$)
0080          READ(6,117,ERR=115) HOURS
0081    117   FORMAT(I5)
0082          HOURS = HOURS*10
0083    120   TYPE 121
0084    121   FORMAT(/' INPUT OVERTIME HOURS AVAILABLE THIS WEEK. ',$)
0085          READ(6,117,ERR=120) OVRTM
0086    125   TYPE 126
```

```
FORTRAN IV      V02.2-1          FRI 12-JUN-81 05:46:07          PAGE 003
[100,6]SHOP=SHOP

      125   FORMAT('0INPUT NUMBER WORKERS TRANSFERRING TO/FROM SHOP:',I3,
     *        ' POSITIVE NUMBER IF TRANSFERRING INTO SHOP, AND '
     *        ' NEGATIVE NUMBER IF LEAVING SHOP.',/)
            READ(6,117,ERR=125) ADDMEN
      C*************************************************************
      C**             UPDATE WORKER STATUS                        **
      C*************************************************************
      140   PNT = WRKF
      150   IF(JOBSIZ(PNT).EQ.NWK) GO TO 160
            JOBSIZ(PNT) = JOBSIZ(PNT)+1
            EVENT(PNT) = EFF(JOBSIZ(PNT))
      160   PNT = DIRECT(PNT)
            IF(PNT.NE.0) GO TO 150
      C*************************************************************
      C**             ADD OR DELETE WORKERS                       **
      C*************************************************************
            SUM = SUM+ADDMEN
            IF(ADDMEN.EQ.0) GO TO 200
            IF(ADDMEN.LT.0) GO TO 175
            DO 170 K=1,ADDMEN
            JOB(1) = 1
            TIME = EFF(1)
            JOB(2) = -1
            JOB(3) = 0
            IDLE = IDLE+1
            PNT = EMPTY
            CALL PUT(WRKF,WRKL,0,TIME,RFLOW,EMPTY)
            CALL TIDLE(SUM,CLOCK,0,PNT,IDLTM,ITR,TOTIDL)
            CALL NEWJOB(PNT,WQF,WQL,NWQF,NWQL,EMPTY,JOBWK,HRWK,
     *                  CLOCK,NOJOB,IDLE,SUM,IDLTM,ITR,TOTIDL)
      170   CONTINUE
            GO TO 200
      C*************************************************************
      C**                     DELETE WORKERS                      **
      C*************************************************************
      175   ADDMEN = -ADDMEN
            DO 180 K=1,ADDMEN
            CALL DELETE(WQF,WQL,NWQF,NWQL,WRKF,WRKL,EMPTY,CLOCK,IDLE,
     *                  JOBWK,HRWK)
      180   CONTINUE
      C*************************************************************
      C**                   GENERATE INCOMING JOBS                **
      C*************************************************************
      200   HOURS = 0
            IF(HOURS.EQ.0) GO TO 240
      210   MAXJOB = JOBWK+JOBNWK+SUM+1
            IF(MAXJOB.GE.MAXLST) GO TO 810
            JOB(1) = -ALOG(RAN(I1,I2))*MNJOB*10
            IF(JOB(1).EQ.0) GO TO 210
            RFLOW = CLOCK
            IF(RAN(I1,I2).GT.P1) GO TO 220
      C*************************************************************
      C**                   ENTER NON-WORKABLE QUEUE                **
      C*************************************************************
```

```
0133          JOB(3) = 1
0134          JOBNWK = JOBNWK+1
0135          HRNWK = HRNWK+JOB(1)
0136          CALL DELAY(TIME,I1,I2,LAM1,LAM2,CLOCK)
0137          CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
0138          GO TO 230
      C*************************************************************************
      C**                      ENTER WORKABLE QUEUE                           **
      C*************************************************************************
0134  220     JOBWK = JOBWK+1
0135          HRWK = HRWK+JOB(1)
0139          NOJOB = 0
0139          DJOB = JOB(1)/10
0140          TIME = DJOB
0141          JOB(2) = JOB(1)-DJOB
0142          JOB(3) = 0
0143          CALL PUT(WQF,WQL,0,TIME,RFLOW,EMPTY)
0144  230     THOURS = THOURS+JOB(1)
0145          IF(THOURS.LE.HOURS) GO TO 210
      C*************************************************************************
      C**                    START SIMULATION                                 **
      C*************************************************************************
0147  240     DAY = 0
0148          TIMER = 0
0149          IF(CLOCK.NE.0.0) GO TO 280
      C*************************************************************************
      C**                    PUT END OF DAY INDICATOR ON EVENT CHAIN          **
      C*************************************************************************
0151          JOB(1) = -1
0152          TIME = 80.001
0153          JOB(2) = 0
0154          JOB(3) = 0
0155          CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
0156  280     PNT = WRKF
0157  290     IF(REMAIN(PNT).NE.-1) GO TO 300
0159          CALL TIDLE(SUM,CLOCK,0,PNT,IDLTM,ITR,TOTIDL)
0160          IDLE = IDLE+1
0161  300     PNT = DIRECT(PNT)
0162          IF(PNT.NE.0) GO TO 290
0164  310     PNT = WRKF
0165          DO 315 J=1,SUM
0166          IF(IDLE .EQ.0) GO TO 320
0168          IF(NOJOB.EQ.1) GO TO 320
0170          IF(REMAIN(PNT).NE.-1) GO TO 315
0172          CALL NEWJOB(PNT,WQF,WQL,NWQF,NWQL,EMPTY,JOBWK,HRWK,
             *            CLOCK,NOJOB,IDLE,SUM,IDLTM,ITR,TOTIDL)
0173  315     PNT = DIRECT(PNT)
      C*************************************************************************
      C**                    UPDATE CLOCK                                     **
      C*************************************************************************
0174  320     CLOCK = EVENT(NWQF)
      C*************************************************************************
      C**                    TAKE FIRST EVENT AND TEST FOR TYPE               **
      C*************************************************************************
```

```
C**********
C**                 JOB GOES NON-WORKABLE FROM IN PROCESS
C**********
      PNT = JOB(3)
      JOB(3) = -2
      CALL DELAY(TIME,I1,I2,LAM1,LAM2,CLOCK)
      JOBNWK = JOBNWK+1
      HRNWK = HRNWK+JOB(2)
      CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
      GO TO 350
C**********
C**                 JOB GOES FROM NON-WORKABLE TO WORKABLE
C**********
425   DJOB = JOB(1)/10
      TIME = DJOB
      JOB(2) = JOB(1)-DJOB
      JOBNWK = JOBNWK-1
      HRNWK = HRNWK-JOB(1)
      CALL PUT(WQF,WQL,1,TIME,RFLOW,EMPTY)
      NOJOB = 0
      JOBWK = JOBWK+1
      HRWK = HRWK+JOB(1)
      GO TO 310
C**********
C**                 JOB STAYS IN PROCESS UNTIL COMPLETION
C**********
450   DUMMY = JOB(2)/EVENT(JOB(3))
      TIME = CLOCK+DUMMY
      JOB(2) = 0
      RFMAIN(JOB(3)) = EMPTY
      CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
      GO TO 320
C**********
C**                   JOB IS COMPLETED
C**********
475   PNT = JOB(3)
      HRDONE = HRDONE+JOB(1)
      JCOMP = JCOMP+1
      TOTFLO = TOTFLO+CLOCK-RFLOW
      SQFLOW = SQFLOW+(CLOCK-RFLOW)**2
C**********
C**                 GIVE FREED WORKER NEW JOB
C**********
      RFMAIN(PNT) = -1
      IDLE = IDLE+1
      CALL ITIDLE(SUM,CLOCK,0,PNT,IDLTM,ITR,TOTIDL)
      CALL NEWJOB(PNT,WQF,WQL,NWQF,NWQL,EMPTY,JOBWK,HRWK,
     *            CLOCK,NOJOB,IDLE,SUM,IDLTM,ITR,TOTIDL)
```

```
                GO TO 320
      C*****************************************************************
      C**      JOB RETURNS FROM NON WORKABLE QUEUE FOR COMPLETION      **
      C*****************************************************************
  500 CALL PUT(WQF,WQL,1,TIME,RFLOW,EMPTY)
      NOJOB = 0
      JOBNWK = JOBNWK-1
      HRNWK = HRNWK-JOB(2)
      JOBWK = JOBWK+1
      HRWK = HRWK+JOB(1)
      GO TO 310
      C*****************************************************************
      C**                      SHIFT IS OVER                          **
      C*****************************************************************
      C**              DETERMINE OVERTIME HOURS                       **
      C*****************************************************************
  525 DAY = DAY+1
      TIMER = TIMER+80
      DHR = OVRTM/SUM
      ZIP = 10
      IF(DHR.GT.(6-DAY)) ZIP = 20
      TIME = CLOCK+240.0
      CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
      CLOCK = CLOCK-0.001
      C*****************************************************************
      C**                  FINISH WORK FOR THE DAY                    **
      C*****************************************************************
      PNT = WRKF
      DO 660 M=1,SUM
      RTIME = CLOCK
      IF(REMAIN(PNT).EQ.-1) GO TO 650
      IF(OVRTM.EQ.1) ZIP = 10
      IF(OVRTM.EQ.0) ZIP = 0
      RTIME = RTIME+ZIP
      WROW = REMAIN(PNT)
      RDIF = EVENT(WROW)-CLOCK-ZIP
      OVRTM = OVRTM-ZIP/10
      IF(RDIF.LE.0.0)  GO TO 550
      IF(WROW.NE.NWQF) GO TO 530
      CALL TAKE(NWQF,NWQL,TIME,RFLOW,EMPTY,STATUS)
      TIME = TIME+160-ZIP
      CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
      GO TO 660
  530 PNT1 = NWQF
  535 PNT2 = DIRECT(PNT1)
      IF(REMAIN(PNT).EQ.PNT2) GO TO 540
      PNT1 = PNT2
      GO TO 535
  540 TIME = EVENT(PNT2)+160-ZIP
      CALL RESORT(PNT,PNT1,PNT2,EMPTY,NWQF,NWQL,TIME)
      GO TO 660
      C*****************************************************************
      C**          JOB IS COMPLETED DURING OVERTIME HOURS             **
      C*****************************************************************
```

```
550    IF(WROW.NE.NWQF) GO TO 560
       CALL TAKE(NWQF,NWQL,TIME,RFLOW,EMPTY,STATUS)
       GO TO 600
560    PNT1 = NWQF
565    PNT2 = DIRECT(PNT1)
       IF(REMAIN(PNT).EQ.PNT2) GO TO 570
       PNT1 = PNT2
       GO TO 565
570    DIRECT(PNT1) = DIRECT(PNT2)
       DIRECT(PNT2) = EMPTY
       EMPTY = PNT2
600    HRDONE = HRDONE+JOBSIZ(WROW)
       JCOMP = JCOMP+1
       TOTFLO = TOTFLO+ CLOCK+ZIP-FLOW(WROW)
       SQFLOW = SQFLOW+(CLOCK+ZIP-FLOW(WROW))**2
       REMAIN(PNT) = -1
       IDLE = IDLE+1
       ATIME = CLOCK+ZIP+RDIF
       CALL TIDLE(SUM,ATIME,0,PNT,IDLTM,ITR,TOTIDL)
       CALL NEWJOB(PNT,WQF,WQL,NWQF,NWQL,EMPTY,JOBWK,HRWK,
     *             ATIME,NOJOB,IDLE,SUM,IDLTM,ITR,TOTIDL)
C**********************************************************************
C**     MODEL ASSUMES THAT THE PROBABILITY OF A JOB BEING STARTED     **
C**     AND COMPLETED IN THE SAME OVERTIME PERIOD IS 0.
C**********************************************************************
       IF(REMAIN(PNT).EQ.NWQF) GO TO 630
       IF(REMAIN(PNT).EQ.  -1) GO TO 650
       PNT1 = NWQF
610    PNT2 = DIRECT(PNT1)
       IF(REMAIN(PNT).EQ.PNT2) GO TO 620
       PNT1 = PNT2
       GO TO 610
620    TIME = EVENT(PNT2)+160.0-ZIP
       CALL RESORT(PNT,PNT1,PNT2,EMPTY,NWQF,NWQL,TIME)
       GO TO 660
630    CALL TAKE(NWQF,NWQL,TIME,RFLOW,EMPTY,STATUS)
       TIME = TIME+160.0-ZIP
       CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
       GO TO 660
650    CALL TIDLE(SUM,RTIME,1,PNT,IDLTM,ITR,TOTIDL)
       IDLE = IDLE-1
660    PNT = DIRECT(PNT)
       CLOCK = CLOCK+160.0
       IF(TIMER.EQ.1040) GO TO 700
       TIMER = TIMER+160
       GO TO 280
C**********************************************************************
C**                        WEEK IS OVER                               **
C**********************************************************************
700    CHRDNE = HRDONE-THRDNE
       THRDNE = HRDONE
       AVFLOW = 0
       IF(JCOMP.NE.JBWK) AVFLOW = (TOTFLO-PREFLO)/(JCOMP-JBWK)/240.0
       PREFLO = TOTFLO
```

```
FORTRAN IV       V02.2-1              12-JUN-80  05:46:??          PAGE ??
SHOP,CHOP=SHOP

0313            JOBWK = JCOMP
0319            IF(FLAG.EQ.1) GO TO 90
0321            CALL OUTPUT(JOBUK,HRWK,JOBNWK,HRNWK,CHKTIM,WEEK,SUM,
               *                IDLTM,AVFLOW)
0322   710      TYPE 711
0323   711      FORMAT(/' DO YOU WISH TO OPERATE ANOTHER WEEK? (Y/N) ',$)
0324            ACCEPT 727,REPLY
0325   727      FORMAT(A4)
0326            IF(REPLY.EQ.YES) GO TO 100
0328            IF(REPLY.EQ. NO) GO TO 800
0329            GO TO 710
0331   800      RHOUR = (HRDONE-RHOUR)/10.0
0332            TOTIDL = TOTIDL/10.0
0333            AVFLOW = (TOTFLO-RDFLOW)/(JCOMP*240.0)
0334            SDFLOW = SQRT((SQFLOW-RSQFLO-JCOMP*(TOTFLO-RDFLOW))/
               *             (JCOMP-1))/240.0
0335            CALL FINAL(JCOMP,RHOUR,TOTIDL,WEEK,AVFLOW,SDFLOW)
0336            STOP
0337   810      MAXJOB = JOBWK+JOBNWK
0338            WRITE(6,815) MAXJOB
0339   815      FORMAT('0MAXIMUIM JOB LIMIT EXCEEDED. ',I4,' JOBS IN SHOP.')
0340            STOP
0341            END
```

FORTRAN IV V02.[illegible] FRI 1[illegible] JUN 81 09:4[illegible] PAGE [illegible]
[illegible].CHOP[illegible]

```
      SUBROUTINE INPUT(P1,P2,I1,I2,DMEAN,DSTDEV,NWK,MNJOB,EFF)
C*****************************************************************************
C**   SUBROUTINE INPUT ALLOWS THE OPERATOR TO INPUT THE INITIAL              *
C**   PARAMETERS AND CONSTRAINTS.                                            *
C**   FOR A DEFINITION OF THE VARIABLES SEE THE MAIN PROGRAM.                *
C*****************************************************************************
      REAL LAM1,LAM2,MNJOB
      DIMENSION EFF(6)
    5 TYPE 6
    6 FORMAT(/' INPUT PROBABILITY JOB IS DETERMINED
     *          NON-WORKABLE UPON ENTERING SHOP.  ',$)
      READ(6,10,ERR=5) P1
   10 FORMAT(F5.4)
   15 TYPE 16
   16 FORMAT(/' INPUT PROBABILITY JOB GOES NON-WORKABLE'/
     *          ' AFTER START OF PROCESSING. ',$)
      READ(6,10,ERR=15) P2
   20 TYPE 21
   21 FORMAT(/' INPUT MEAN DELAY IF JOB GOES NONWORKABLE IN DAYS.'/
     *          ' (MUST INCLUDE DECIMAL POINT (EXAMPLE "3.0")) ',$)
      READ(6,22,ERR=20) DMEAN
   22 FORMAT(F10.0)
   25 TYPE 26
   26 FORMAT(/' INPUT STANDARD DEVIATION OF DELAY TIME. (MUST
     *          ' INCLUDE DECIMAL POINT.    ',$)
      READ(6,22,ERR=25) DSTDEV
   30 TYPE 31
   31 FORMAT(/' INPUT INTEGER (ODD) SEED FOR RANDOM'/
     *          ' NUMBER GENERATOR. (I3 FORMAT) ',$)
      READ(6,32,ERR=30) I1
   32 FORMAT(I3)
      I2 = I1
   40 TYPE 41
   41 FORMAT(/' INPUT AVERAGE JOB SIZE IN STANDARD MAN-HOURS'/
     *          ' (MUST INCLUDE DECIMAL POINT) ',$)
      READ(6,45,ERR=40) MNJOB
   45 FORMAT(F6.2)
   60 TYPE 61
   61 FORMAT(/' INPUT NUMBER OF WEEKS UNTIL TRANSFERED PERSONNEL'/
     *          ' ADJUSTED AND WORKING FULL EFFICIENCY. (I3 FORMAT) ',$)
      READ(6,32,ERR=60) NWK
   65 TYPE 66
   66 FORMAT(/' INPUT EFFICIENCY RATING OF TRANSFERED WORKER'/
     *          ' FOR FIRST WEEK IN SHOP. (IF WORKER IS'/
     *          ' EXPECTED TO WORK AT 80% EFFICIENCY THEN'/
     *          ' EFFICIENCY RATING IS 0.80)  ',$)
      READ(6,10,ERR=65) EFF(1)
      M = 1
   24 IF(M.GE.NWK-1) GO TO 27
      M = M+1
   75 TYPE 76
   76 FORMAT(/' INPUT EFFICIENCY RATING FOR THE NEXT WEEK. ',$)
      READ(6,10,ERR=75) EFF(M)
      GO TO 24
```

```
      80      TYPE 81
      81      FORMAT(' INPUT EFFICIENCY RATING FOR THIS WEEK AGAIN. ',$)
              GO TO 75
              EFF(NWK) = 1.0
      50      TYPE 51
      51      FORMAT(' INPUT NUMBER OF WORKERS IN SHOP AT START',
             *       ' OF SIMULATION. (I3 FORMAT) ',$)
              READ(6,32,ERR=50) NORMEN
              RETURN
              END
```

```
FORTRAN IV      V02.2-1          FRI 12-JUN-81 05:46:31          PAGE 0001
SHOP,SHOP=SHOP

0001          SUBROUTINE DATA(P1,P2,I1,I2,DMEAN,DSTDEV,NWK,MNJOB,NORMEN,EFF)
      C*****************************************************************
      C**    SUBROUTINE DATA IS USED TO INPUT THE INITIAL PARAMETERS    **
      C**    FROM A STORED DATA FILE.                                   **
      C**     FOR A DEFINITION OF THE VARIABLES SEE THE MAIN PROGRAM.   **
      C*****************************************************************
0002          REAL MNJOB
0003          DIMENSION EFF(6)
0004          READ(3,5,ERR=25) P1,P2,DMEAN,DSTDEV,NWK,MNJOB
0005    5     FORMAT(2(2X,F5.4),14X,2(2X,F10.3),2X,I3,2X,F6.2)
0006          READ(3,10,ERR=25) (EFF(J),J=1,6)
0007    10    FORMAT(2X,6F4.2)
      C*****************************************************************
      C**      USE CLOCK TO FIND RANDOM SEED FOR RANDOM NUMBER GENERATOR  **
      C*****************************************************************
0008          I1 = SECNDS(0.)/3.0
0009          IF(MOD(I1,2).EQ.0) I1 = I1 + 1
0011          I2 = I1
0012    15    TYPE 16
0013    16    FORMAT(/' INPUT NUMBER WORKERS IN SHOP AT START OF',
             *         ' SIMULATION, ',$)
0014          READ(6,20,ERR=15) NORMEN
0015    20    FORMAT(I5)
0016          RETURN
0017    25    TYPE 26
0018    26    FORMAT(/' ERROR IN FILE "DATA.DAT", CHECK FILE.')
0019          STOP
0020          END
```

```
0001          SUBROUTINE PARAM(DMEAN,DSTDEV,LAM1,LAM2)
      C*************************************************************** *
      C**   SUBROUTINE CALCULATES THE PARAMETERS FOR THE DELAY TIME    **
      C**   DISTRIBUTION.                                              **
      C*************************************************************** *
0002          REAL*8 A,B
0003          REAL LAM1,LAM2
      C*************************************************************** *
      C**                CALCULATE COEFFICIENT OF VARIANCE            **
      C*************************************************************** *
0004          COVAR = DSTDEV/DMEAN
0005          IF(COVAR.GE.0.99) GO TO 20
0007          IF(COVAR.LE.0.71) GO TO 30
      C*************************************************************** *
      C**           FIND PARAMETERS OF GENERAL TWO-STAGE ERLANG      **
      C*************************************************************** *
0009          A = DMEAN/(DMEAN**2-DSTDEV**2)
0010          B = SQRT(2*DSTDEV**2-DMEAN**2)/(DMEAN**2-DSTDEV**2)
0011          LAM1 = A-B
0012          LAM2 = A+B
0013          RETURN
      C*************************************************************** *
      C**              FIND PARAMETERS OF EXPONENTIAL                 **
      C*************************************************************** *
0014  20      DUMI = 1.0/DMEAN
0015          LAM2 = 100.0*DUMI
0016          LAM1 = LAM2/(DMEAN*LAM2-1.0)
0017          RETURN
      C*************************************************************** *
      C**         FIND PARAMETERS OF SPECIAL TWO-STAGE ERLANG.        **
      C*************************************************************** *
0018  30      LAM1 = 2.0/DMEAN
0019          LAM2 = LAM1
0020          RETURN
0021          END
```

```
FORTRAN IV       V02.2-1          FRI 12-JUN-81 05:46:37          PAGE ...
SHOP,SHOP=SHOP

0001          SUBROUTINE DELETE(WQF,WQL,NWQF,NWQL,WRKF,WRKL,EMPTY,CLOCK,
             *                  IDLE,JOBWK,HRWK)
      C*****************************************************************
      C**   SUBROUTINE DELETES WORKERS FROM THE SHOP AND PUTS ANY WORK  **
      C**   LEFT BACK ON THE WORKABLE BACKLOG.                          **
      C*****************************************************************
0002          IMPLICIT INTEGER (A-Z)
0003          REAL EVENT,CLOCK,TIME,HRWK,FLOW,RFLOW
0004          COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
             *       DIRECT(1000),WORK(1000),JOB(3)
0005          IF(WRKF.EQ.WRKL) GO TO 50
      C*****************************************************************
      C**              FIND WORKER'S LOCATION IN LIST.                **
      C*****************************************************************
0007          PNTA = WRKF
0008    5     PNTB = DIRECT(PNTA)
0009          IF(DIRECT(PNTB).EQ.0) GO TO 10
0011          PNTA = PNTB
0012          GO TO 5
      C*****************************************************************
      C**              REMOVE WORKER AND UPDATE POINTERS.             **
      C*****************************************************************
0013    10    DIRECT(PNTA) = 0
0014          WRKL = PNTA
0015          DIRECT(PNTB) = EMPTY
0016          EMPTY = PNTB
0017          IF(REMAIN(PNTB).EQ.-1) GO TO 40
0019          PNT1 = REMAIN(PNTB)
0020          IF(PNT1.EQ.NWQF) GO TO 25
0022          PNT2 = NWQF
0023    15    PNT3 = DIRECT(PNT2)
0024          IF(PNT3.EQ.PNT1) GO TO 20
0026          PNT2 = PNT3
0027          GO TO 15
      C*****************************************************************
      C**        PUT UNFINISHED JOB BACK ON WORKABLE BACKLOG.         **
      C*****************************************************************
0028    20    JOB(1) = JOBSIZ(PNT3)
0029          JOB(2) = REMAIN(PNT3)
0030          JOB(3) = WORK(PNT3)
0031          TIME = EVENT(PNT3)
0032          RFLOW = FLOW(PNT3)
0033          DIRECT(PNT2) = DIRECT(PNT3)
0034          DIRECT(PNT3) = EMPTY
0035          EMPTY = PNT3
0036          IF(DIRECT(PNT2).EQ.0) NWQL=PNT2
0038          GO TO 30
0039    25    CALL TAKE(NWQF,NWQL,TIME,RFLOW,EMPTY,STATUS)
0040    30    TIME = TIME-CLOCK
0041          DUMMY = TIME*EVENT(JOB(3))
0042          IF(JOB(2).EQ.0) GO TO 35
0044          TIME = DUMMY
0045          JOB(3) = -2
0046          CALL PUT(WQF,WQL,1,TIME,RFLOW,EMPTY)
```

```
FORTRAN IV       V02.2-1                    UN-81 05:44:42          PAGE
SHOP,SHOP=SHOP

0047          JOBWK = JOBWK+1
0048          HRWK = HRWK+JOB(1)
0049          RETURN
0050    35    JOB(2) = DUMMY
0051          JOB(3) = -1
0052          CALL PUT(WQF,WQL,1,TIME,RFLOW,EMPTY)
0053          JOBWK = JOBWK+1
0054          HRWK = HRWK+JOB(1)
0055          RETURN
      C*************************************************************************
      C**                        WORKER WAS IDLE.                             **
      C*************************************************************************
0056    40    IDLE = IDLE-1
0057          RETURN
0058    50    TYPE 51
0059    51    FORMAT(/' ALL WORKERS DELETED.  PROGRAM TERMINATED.')
0060          STOP
0061          END
```

SHOP,SHOP=SHOP

```
0001          SUBROUTINE DELAY(TOTAL,I,J,PAR1,PAR2,TIME)
      C*****************************************************************
      C**     SUBROUTINE DETERMINES DELAY TIME FOR A JOB ENTERING THE    **
      C**     NON-WORKABLE QUEUE.  DELAY TIME IS ADJUSTED SO THAT NO     **
      C**     JOBS CAN COME OFF THE QUEUE AT NIGHT.                      **
      C**     VARIABLES ARE AS DEFINED IN MAIN PROGRAM.                  **
      C*****************************************************************
0002  5       INTER = (-ALOG(RAN(I,J))/PAR1-ALOG(RAN(I,J))/PAR2)*240
0003          IF(INTER.GT.4800) GO TO 5
0005          TOTAL = INTER+TIME
0006          REM = AMOD(TOTAL,240.0)
0007          IF(REM.GT.80.0) GO TO 10
0009          RETURN
0010  10      TOTAL = TOTAL+240-REM
0011          RETURN
0012          END
```

```
SHOP,SHOP=SHOP

0001          SUBROUTINE PUT(FROW,LROW,SORT,REAL,RFLOW,SPACE)
        C*********************************************************************
        C**   SUBROUTINE PUT IS USED TO PUT AN ENTRY ONTO THE LIST.  ENTRIES  **
        C**   CAN BE INSERTED ON EITHER END OF THE LIST OR INTO THE MIDDLE.   **
        C**    SORT = VARIABLE WHICH DETERMINES TYPE OF INSERTION.            **
        C**              IF 0 THEN ENTRY IS PUT ONTO BACK OF LIST.            **
        C**              IF 1    "    "    "   "    "  FRONT OF LIST.         **
        C**              IF 2 ENTRY IS SORTED INTO THE LIST SO THAT IT IS IN  **
        C**              FRONT OF THE FIRST LARGER ENTRY.                     **
        C*********************************************************************
0002          IMPLICIT INTEGER (A-Z)
0003          REAL EVENT,FLOW,REAL,RFLOW
0004          COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
             *       DIRECT(1000),WORK(1000),JOB(3)
        C*********************************************************************
        C**           DETERMINE WHERE ENTRY GOES;FIRST,LAST,OR SORTED.        **
        C*********************************************************************
0005          IF (FROW.EQ.0) GO TO 50
0007          IF (SORT.EQ.0) GO TO 40
0009          IF (SORT.EQ.1) GO TO 30
        C*********************************************************************
        C**            SORT LIST TO FIND WHERE ENTRY GOES.                    **
        C*********************************************************************
0011          PNT1 = FROW
0012          IF (EVENT(PNT1).GT.REAL) GO TO 30
0014    10    PNT2 = DIRECT(PNT1)
0015          IF (PNT2.EQ.0) GO TO 40
0017          IF (EVENT(PNT2).GT.REAL) GO TO 20
0019          PNT1 = PNT2
0020          GO TO 10
        C*********************************************************************
        C**              PUT ENTRY INTO MIDDLE OF LIST.                       **
        C*********************************************************************
0021    20    JOBSIZ(SPACE) = JOB(1)
0022          EVENT(SPACE) = REAL
0023          FLOW(SPACE) = RFLOW
0024          REMAIN(SPACE) = JOB(2)
0025          WORK(SPACE) = JOB(3)
0026          ZZ = SPACE
0027          SPACE = DIRECT(SPACE)
0028          DIRECT(PNT1) = ZZ
0029          DIRECT(ZZ) = PNT2
0030          RETURN
        C*********************************************************************
        C**              PUT ENTRY ONTO FRONT OF LIST.                        **
        C*********************************************************************
0031    30    PNT = FROW
0032          FROW = SPACE
0033          JOBSIZ(FROW) = JOB(1)
0034          EVENT(FROW) = REAL
0035          FLOW(FROW) = RFLOW
0036          REMAIN(FROW) = JOB(2)
0037          WORK(FROW) = JOB(3)
0038          SPACE = DIRECT(SPACE)
```

```
0039          DIRECT(FROW) = PNT
0040          RETURN
      C*****************************************************************
      C**              PUT ENTRY ONTO END OF LIST.                    **
      C*****************************************************************
0041    40    PNT = LROW
0042          LROW = SPACE
0043          JOBSIZ(LROW) = JOB(1)
0044          EVENT(LROW) = REAL
0045          FLOW(LROW) = RFLOW
0046          REMAIN(LROW) = JOB(2)
0047          WORK(LROW) = JOB(3)
0048          SPACE = DIRECT(SPACE)
0049          DIRECT(PNT) = LROW
0050          DIRECT(LROW) = 0
0051          RETURN
      C*****************************************************************
      C**             ENTRY IS FIRST IN LIST SETUP IS REQUIRED.       **
      C*****************************************************************
0052    50    FROW = SPACE
0053          LROW = SPACE
0054          JOBSIZ(FROW) = JOB(1)
0055          EVENT(FROW) = REAL
0056          FLOW(FROW) = RFLOW
0057          REMAIN(FROW) = JOB(2)
0058          WORK(FROW) = JOB(3)
0059          SPACE = DIRECT(SPACE)
0060          DIRECT(FROW) = 0
0061          RETURN
0062          END
```

```
0001          SUBROUTINE TAKE(FROW,LROW,AREAL,RFLOW,SPACE,STATUS)
      C*************************************************************
      C**   SUBROUTINE TAKE IS USED TO REMOVE AN ENTRY IN THE LIST,    **
      C**   VACANCY IS THEN DESIGNATED AS FIRST AVAILABLE EMPTY SPACE. **
      C**   STATUS IS A FLAG, WHEN THERE ARE NO JOBS IN THE LIST IT IS **
      C**   SET TO '0'.  OTHERWISE IT IS SET TO '1'.                   **
      C*************************************************************
0002          IMPLICIT INTEGER(D-Z)
0003          REAL EVENT,FLOW,RFLOW
0004          COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
             *       DIRECT(1000),WORK(1000),JOB(3)
0005          STATUS = 1
0006          IF(FROW.EQ.0) GO TO 15
      C***** TRANSFER INFORMATION ON ENTRY ***************************
0008          JOB(1) = JOBSIZ(FROW)
0009          AREAL = EVENT(FROW)
0010          RFLOW = FLOW(FROW)
0011          JOB(2) = REMAIN(FROW)
0012          JOB(3) = WORK(FROW)
      C****** UPDATE POINTERS ****************************************
0013          PNT = DIRECT(FROW)
0014          DIRECT(FROW) = SPACE
0015          SPACE = FROW
0016          FROW = PNT
0017          IF(FROW.EQ.0) GO TO 15
0019          RETURN
0020    15    STATUS = 0
0021          LROW = 0
0022          RETURN
0023          END
```

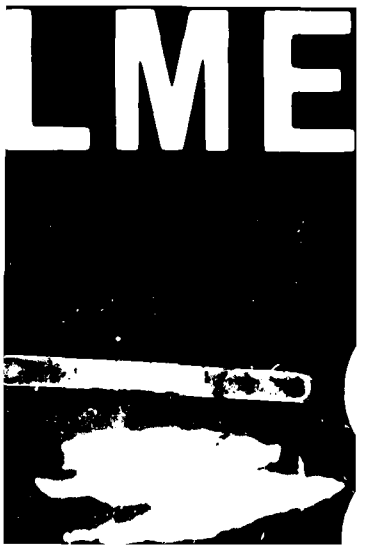
```
FORTRAN IV      V02.2-1          FRI 12-JUN-81 05:46:50          PAGE 001
SHOP.SHOF=SHOP

0001          SUBROUTINE NEWJOB(PNT,WQF,WQL,NWQF,NWQL,EMPTY,JOBWK,HRWK,
             *                  CLOCK,NOJOB,IDLE,SUM,IDLTM,ITR,TOTIDL)
      C*****************************************************************
      C**   SUBROUTINE GIVES WORKER A NEW JOB FROM THE WORKABLE QUEUE.   **
      C**   VARIABLES ARE AS DEFINED IN THE MAIN PROGRAM.                **
      C*****************************************************************
0002          IMPLICIT INTEGER(A-Z)
0003          REAL EVENT,TIME,CLOCK,IDLTM,HRWK,FLOW,RFLOW
0004          COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
             *       DIRECT(1000),WORK(1000),JOB(3)
0005          IF(NOJOB.EQ.1) RETURN
0006          CALL TIDLE(SUM,CLOCK,1,PNT,IDLTM,ITR,TOTIDL)
0008          IDLE = IDLE-1
0009          CALL TAKE(WQF,WQL,TIME,RFLOW,EMPTY,STATUS)
0010          IF(STATUS.EQ.0) NOJOB=1
0012          JOBWK = JOBWK-1
0013          HRWK = HRWK-JOB(1)
0014          IF(JOB(3).NE.-2)GO TO 20
      C*****************************************************************
      C**          JOB HAS ALREADY BEEN IN-PROCESS BEFORE             **
      C*****************************************************************
0016          DUMMY = JOB(2)/EVENT(PNT)
0017          TIME = DUMMY+CLOCK
0018          JOB(2) = 0
0019    10    JOB(3) = PNT
0020          REMAIN(PNT) = EMPTY
0021          CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
0022          RETURN
      C*****************************************************************
      C**            JOB HAS NOT BEEN IN-PROCESS BEFORE               **
      C*****************************************************************
0023    20    DUMMY = TIME/EVENT(PNT)
0024          TIME = DUMMY+CLOCK
0025          GO TO 10
0026          END
```

```
0001          SUBROUTINE IDLE(SUM,CLOCK,FLAG1,PNT,IDLTM,ITR,TOTIDL)
              C*************************************************************
              C**         SUBROUTINE DETERMINES AMOUNT OF WORKER IDLE TIME.  **
              C*************************************************************
0002          IMPLICIT INTEGER (D-Z)
0003          REAL IDLTM,EVENT,TOTIDL,FLOW
0004          COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
             *        DIRECT(1000),WORK(1000),JOB(3)
0005          DIMENSION IFLAG(100),BTIME(100)
0006          IF(SUM.GT.ITR) ITR = SUM
0008    5     IF(FLAG1.EQ.1) GO TO 20
              C*************************************************************
              C**                   WORKER STARTS IDLE PERIOD.              **
              C*************************************************************
0010          DO 10 I=1,ITR
0011          IF(IFLAG(I).EQ.0) GO TO 15
0013    10    CONTINUE
0014    15    IFLAG(I) = 1
0015          BTIME(I) = CLOCK
0016          WORK(PNT) = I
0017          RETURN
              C*************************************************************
              C**                   WORKER ENDS IDLE PERIOD                 **
              C*************************************************************
0018    20    I = WORK(PNT)
0019          IDLTM = IDLTM+CLOCK-BTIME(I)
0020          TOTIDL = TOTIDL+CLOCK-BTIME(I)
0021          IFLAG(I) = 0
0022          RETURN
0023          END
```

```
FORTRAN IV      V02.2-1          FRI 12-JUN-81 05:46:56          PAGE  1
SHOP,SHOP=SHOP

0001          SUBROUTINE RESORT(PNT,PNT1,PNT2,EMPTY,NWQF,NWQL,TIME)
      C**********************************************************************
      C**      SUBROUTINE TAKES THE ENTRY AT ROW "PNT2" , CHANGES THE        **
      C**      EVENT TIME AND THEN INSERTS IT BACK ONTO THE LIST.            **
      C**********************************************************************
0002          IMPLICIT INTEGER (A-Z)
0003          REAL EVENT,TIME,FLOW,RFLOW
0004          COMMON EVENT(1000),FLOW(1000),JOBSIZ(1000),REMAIN(1000),
             *       DIRECT(1000),WORK(1000),JOB(3)
0005          JOB(1) = JOBSIZ(PNT2)
0006          JOB(2) = REMAIN(PNT2)
0007          JOB(3) = WORK(PNT2)
0008          RFLOW = FLOW(PNT2)
0009          DIRECT(PNT1) = DIRECT(PNT2)
0010          DIRECT(PNT2) = EMPTY
0011          EMPTY = PNT2
0012          IF(PNT2.EQ.NWQL) NWQL = PNT1
0014          CALL PUT(NWQF,NWQL,2,TIME,RFLOW,EMPTY)
0015          REMAIN(PNT) = PNT2
0016          RETURN
0017          END
```

```
FORTRAN IV      V02.2-1              2-JUN-81 05:45:52          PAGE 001
SHOP,SHOP=SHOP

0001          SUBROUTINE OUTPUT(JOBWK,HRWK,JOBNWK,HRNWK,DONE,WEEK,SUM,
             *                  IDLTM,AVFLOW)
      C*****************************************************************
      C**   SUBROUTINE OUTPUTS THE STATE OF THE SYSTEM AT THE END OF THE   *
      C**   WEEK.  VARIABLES ARE AS DEFINED IN THE MAIN PROGRAM.           *
      C*****************************************************************
0002          IMPLICIT INTEGER(A-Z)
0003          REAL EVENT,CLOCK,IDLTM,HRWK,HRNWK,DONE,WDONE,HW,HNW
0004          REAL*8 AVFLOW
0005          WDONE = DONE/10.0
0006          IDLTM = IDLTM/10.0
0007          HW = HRWK/10.0
0008          HNW = HRNWK/10.0
0009          WRITE(6,10)WEEK
0010    10    FORMAT(10(/),15X,'RESULTS FOR WEEK',I4//)
0011          WRITE(6,15)
0012    15    FORMAT(5X,'BACKLOG',10X,'NO. OF JOBS     NO. OF HOURS OF WOR
0013          WRITE(6,20)
0014    20    FORMAT(5X,'=======',10X,11('='),5X,20('='))
0015          WRITE(6,25)JOBWK,HW,JOBNWK,HNW
0016    25    FORMAT(5X,'WORKABLE',I18,F19.1/5X,'NONWORKABLE',I15,F19.1/)
0017          WRITE(6,35) IDLTM
0018    35    FORMAT(5X,'WORKERS WERE IDLE ',F6.1,' HOURS'/)
0019          WRITE(6,30)WDONE
0020    30    FORMAT(5X,'WORK COMPLETED THIS WEEK TOTALLED ',F7.1,' HOURS.
0021          WRITE(6,36) AVFLOW
0022    36    FORMAT(5X,'AVERAGE FLOW TIME ',F6.1,' DAYS'/)
0023          WRITE(6,40) SUM
0024    40    FORMAT(5X,'NUMBER OF WORKERS IN SHOP IS ',I4/)
0025          RETURN
0026          END
```

```
FORTRAN IV      V02.2 1          FRI 12-JUN-81 05:47:01          PAGE  1
SHOP.SHOP=SHOP

      SUBROUTINE FINAL(JCOMP,RHOUR,TOTIDL,NWEEK,AVFLOW,SDFLOW)
C*********************************************************************
C**   SUBROUTINE OUTPUTS THE STATE OF THE SYSTEM AT THE END OF THE   **
C**   SIMULATION.  SEE MAIN PROGRAM FOR DEFINITION OF VARIABLES.     **
C*********************************************************************
      REAL*8 AVFLOW,SDFLOW
      WRITE(6,10) NWEEK
10    FORMAT(10(/),10X,'FINAL RESULTS FOR',I3,' WEEKS'//)
      WRITE(6,15) JCOMP,RHOUR
15    FORMAT(110,' JOBS WERE COMPLETED TOTALING',F7.1,
     *       ' HOURS OF WORK')
      WRITE(6,16) AVFLOW,SDFLOW
16    FORMAT(/' AVERAGE FLOW TIME WAS ',F7.1,' DAYS WITH A'/
     *       ' STANDARD DEVIATION OF ',F7.1)
      WRITE(6,20) TOTIDL
20    FORMAT(/' WORKERS WERE IDLE FOR ',F7.1,' HOURS.')
      STAT = RHOUR/(RHOUR+TOTIDL)*100.0
      WRITE(6,25) STAT
25    FORMAT(/' SHOP OPERATED AT ',F5.1,'%'////)
      RETURN
      END
```


LME