APPENDIX A

USER'S GUIDE TO MSA

Introduction To The User's Guide

This appendix contains a user's guide to the Manufacturing, Simulation and Analysis (MSA) suite of programs. It is intended as a stand-a-lone document and makes no reference to the thesis hence some duplication has occurred. The user of the software is directed to access the thesis for the technical parameters of the software.

Contents

List Of Figures

List Of Tables

A.1 Introduction

To be generically applicable was the goal behind the development and implementation of the MSA methodology. The MSA program suite has been created to enable the modeling, scheduling and subsequent simulation of production schedules through manufacture. The user does not need any programming expertise but a familiarity of simulation technology is advised.

A.2 Concepts

Modeling

A "Universal Transfer" is used to model the work centres contained in the facility of interest. Each of the Universal Transfers consists of two input queues, a work centre and one output queue. Each UT is defined individually with no reference to the jobs it will process hence a fixed network model of a facility never exists inside the software.

No Network Model

Entities or jobs "self carry" their route through the defined work centres. Entities *enter* a UT or node, are processed and passed onto the succeeding node. Adding a new node then becomes a simple process of just defining another UT, it does not have to be linked into a network. This approach achieves three goals:

1. Models are relatively quick to modify;
2. The modeling philosophy is generic in so far that it may be used to model a variety of manufacturing facilities; and
3. Re-routing of jobs does not require a new network to be constructed, only a modification to the Sequencing Database.

Unless a facility is in "perfect balance" a bottleneck or Critical Resource will be present. The scheduling focus of the MSA methodology is to use the Critical Resource as the key to generating schedules. The scheduling methodology is therefore generically applicable.

Scheduling

The Scheduling of jobs through the modeled facility using the Critical Resource as a focus follows a four stage process:

1. Determine which work centre is to be labelled the Critical Resource.
2. Generate the "Criticality List" which is a list of all jobs to be processed by the Critical Resource.
3. Order the Criticality List with respect to the selected dispatching rule.
4. Use the Criticality list to assign priorities to the order jobs are launch to the facility as well as the order they are processed by work centres.

Jobs are launched to the facility being modeled at the beginning of each week being simulated. In the present version of the software it is not possible to introduce jobs mid-week.

project

A project is defined as a complete set of files (16 in total) which contain all the data pertaining to a facility of interest. Fifteen individual projects are allowed for in the present version of the software and from within each project 999 separate simulation runs (versions) may be executed.

A.3 The Computer System

The files and programs described here were designed for use on an IBM PC, PS/2 or compatible computer. The programs are compiled to .EXE files and require only DOS 2.0 or greater. Hard disk requirements are approximately one and a half megabytes for the program code and up to one megabyte for a project.

A common directory for all programs and data files is required. The directory name is optional and has no effect on the operation of the software.

A.4 Activating and Using the Software

To activate the MSA software suite from the DOS prompt ensure the default directory contains all the programs, then type MSA *enter*. The title page is displayed for approximately five seconds; pressing the *esc* key bypasses this delay and displays the second screen, the copyright and licensing information. Pressing the space bar causes seven system files to be restored into memory and the Main Menu to be displayed. In general menu choices are either in white or grey, menu choices in grey are not active at that particular time, selecting one will cause an error message to be displayed giving the reason why it is inactive.

In normal operation the software makes two sounds: a high pitch single "beep" which indicates that user keyboard input is required and a lower pitch double "beep" which indicates the last input was invalid.

A.5 Project Data File Structure

A project is a set of sixteen files which define the manufacturing facility of interest and consists of: four primary files which the user has read/write access to, defining the manufacturing facility:

1. Jobs Pending;
2. Work Centre Definitions (Universal Transfers);
3. Integrated Bill of Materials; and
4. Calendar;

and twelve secondary files used by the software to maintain the status of the project. The file naming convention follows:

AAA~BOM.INF

where AAA is the project code, BOM is the file type (in this case the Bill of Materials file) and INF indicates that it is a root file.

The root maintains the project state from which the majority of simulation runs under a particular project are activated. Each simulation run off the root files is given a version number. The naming convention for such files, for example Version one, is:

AAA~BOM.001.

The number of versions off the root is limited to 999, sufficient for any normal simulation project.

If, for example, thirty periods are simulated from the root of project AAA, all data for the Job Pending file are stored under:

AAA~BOM.001.

The facility has been provided to select and load any of the thirty periods into memory and make this the active project-version-period combination. At any point in the operation of the software a project-version-period combination may be copied to the root thus defining a new start point for all subsequent simulations under that project but the user is warned that overwriting the root erases all existing versions (simulation runs) under that project. The overwrite option is provided such that a version of a project may be simulated up to a "steady state" and then written to the root. All subsequent simulation runs will then be launched already in a steady state, saving time when doing extensive "what-if?" analysis.

When loading a model the user is prompted for a project code, version number and the period of interest. Only one project, version, period combination may be loaded into memory at any one time. The right hand corner of the screen header (top three lines on the screen) indicates the currently active combination.

A.6 Creating A Project

The menu options and required input to create a project is displayed in Figure A.1. After selecting "Project Data Management" off the Main Menu and "Create" off the subsequent menu, the user is prompted for a three digit project code and a twenty five character project descriptor. The project descriptor has no effect on the operation of the software and is for the user's information only and both the code and the descriptor are displayed in the

screen header whenever the project is active. File initialization creates and stores default values in the sixteen project files.

After file initialization the Project Menu is again displayed with option "D. Input" no longer inactive. To enter the Data Input Menu this option must be selected. This Menu has a column adjacent to the input options A through E which indicates whether the files have been defined. If an input option is selected more than once the new data is appended to the existing file.

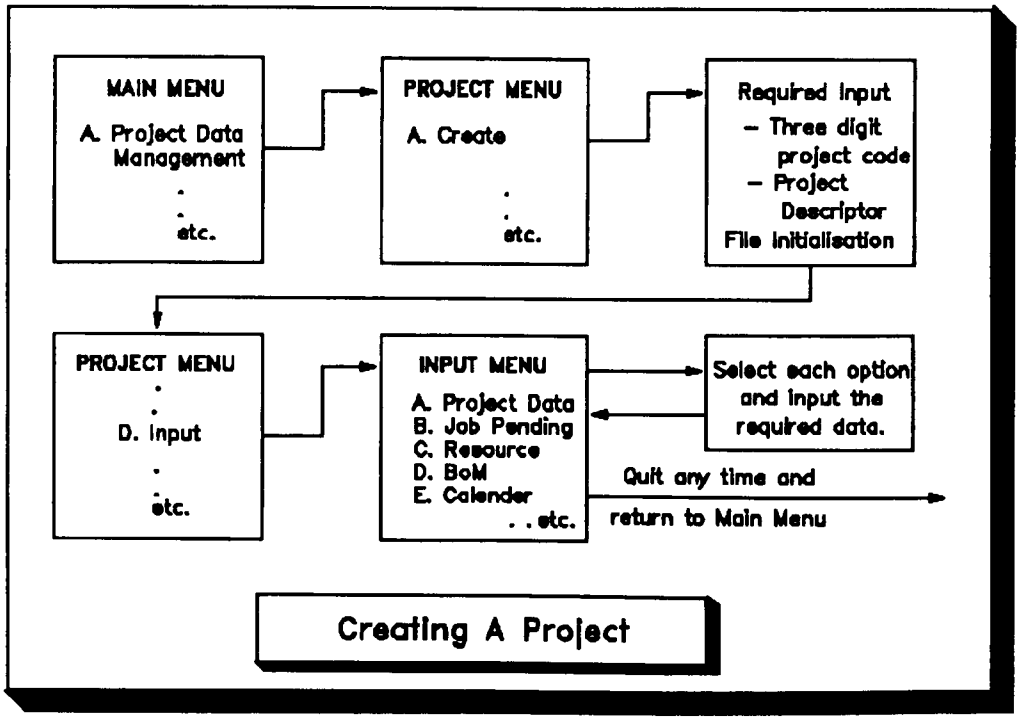


Figure A.1: Creating A Project Using The MSA Software

Options A through E do not have to be selected in any particular order and the menu may be exited at any time although it is recommended however, that the Resource Profile option be executed before the BoM Definition option.

If a previous project exists the user, when selecting one of the input options is prompted to copy the file from a previous project or create a new file. Option "I. Copy Existing Project" on the Data Input Menu allows the user to copy a complete set of data files to the new project. When copying individual files or a complete set the user is prompted for project name, version and specific period.

The following five subsections describe the required input for each of the input options A through E on the Data Input Menu:

A.6.1 PROJECT DATA

Project Data is a miscellaneous category and requires only two inputs:

Global Set Up Time

The set up time to be used when all defined work centres are to have the same set up time regardless of jobs being processed.

Global Batch Size

This is the value used when the transfer batch size is to be the same for all jobs being simulated, irrespective of the process batch size.

A.6.2 JOBS PENDING FILE

Six data fields describe a job to the pending file:

Product Name

The name of the end item product as described to the BoM file.

Job Number

Automatically assigned by the software and follows a simple chronological order. The same job number is used for all sub assemblies and components of an end item throughout the process of simulating the part through the manufacturing facility. No entry is therefore required at this point.

Quantity

How many of the end item (products) are required, the process batch size.

Transfer Batch Size

The value entered in this field is used if the simulator is to be run with each job having an individual transfer batch size. If a number is entered which is greater than the quantity required, the transfer batch size defaults to the lower value.

Due Week

The user is being asked to input the week the job is due. Production weeks are counted sequentially from the beginning of the simulation, starting with week one.

Due Day

By the end of which working day is the job required, day 1 is Monday.

A.6.3 RESOURCE PROFILE (UT Definition)

Selecting the Resource Profile Definition option from the Input Menu, the user is first asked whether the input is to be "global" or "specific". Global allows for one work centre to be defined and duplicated as many times as required. "Specific" input implies the definition is only applicable to one work centre.

Whether global or specific input is chosen each resource defined to the MSA database has nine attributes and the user is given the choice of accepting the default settings or initialising them individually. Following is a description of the nine attributes, the first two at being for information only and have no effect upon the operation of the software:

Resource Name

The "familiar" name by which the work centre is commonly identified.

Resource Group

If the resource belongs to a cell then this field may be used to identify this fact.

Simultaneous Working

How many products may be worked on by the work centre at any one time. This data field may be used to identify one cell containing a number of identical work centres. Several work centres are then defined by using only one work centre definition. For example, setting the Simultaneous Working variable to five means that five identical milling machines may be represented by one work centre definition.

Performance Factor

A value of 100% in this field implies that the resource is meeting standard times for the processes it is carrying out. Inputting a value of 60%, for example, multiplies the standard processing time by 1.4.

Operational

A boolean field indicating whether the work centre is available to process jobs or not.

Launch Job Logic

Each work centre is serviced by two input queues: component and product. When a work centre becomes idle these queues are accessed to determine the next job to be loaded. This attribute gives the queues a priority rating so the software knows which one to access first.

Unload Logic

When a job has finished processing this attribute is accessed to determine whether to move the job into the work centre's output queue or directly to the input queue of the next downstream work centre.

Queue Lengths

Each defined work centre has three queues, the length of which may be individually set.

Loading Rule

Which dispatching rule to use to select a job from those available in a work centres input queues.

A.6.4 BILL OF MATERIALS (BoM) FILE

A BoM file definition is made up of an end item (level 0), sub assemblies and components. A product definition is input a level at a time beginning at level 0 and ends when the last level input contains only components. It a sub assembly is input on level 3 for example it will generate requests for sub assembly and component input on level 4. A component is regarded as an end node and generates no further options at lower levels. The user will have to be familiar with BoM conventions in order to complete thus section for any

given project. The following is a list of required input and a description of the type of data required.

Name

This can apply to end products, sub assemblies and components. There is no control over what characters are used.

Product Set Up

Some simulations may be run with a "product" set up time, that is, all work centres used in the production of components and sub assemblies of an end item will have the same set up time. This field maintains this value and is only requested during level 0 input.

Machine Number

This refers to the number given to a work centre by the software when the resource file was created. It is recommended that the resource file be created before the BoM file.

If the user is unsure of which number was given to which work centre a pop-up menu is available by pressing *esc* then *enter*. The pop-up list displays the work centre number and the user input work centre name. To select a work centre the arrow keys are used to move the cursor adjacent to the desired item and selected by pressing *enter*.

Set Up Time

If the simulation is being run with a "part-based" set up time methodology the data in this field is used. "Part-based" refers to individual part/work centre dependent set up times.

Processing Time

The required input is the standard time to process one item.

By Inputting a "Z" into the Machine Number Field will end input of that particular part. After all data has been input for a product the user is prompted whether further input is required. Selecting "yes" enables the BoM of another BoM to be defined, selecting "no" returns control back to the Data Input Menu.

A.6.5 CALENDAR

The required inputs are the standard hours for a seven day week. After inputting the standard hours the option is available to edit individual weeks to account for the facility being shutdown or working short weeks.

A.7 Loading A Project

Figure A.2 illustrates the process of loading a project. The user must first select off the Main Menu, "Project Data Management" and "Load" off the subsequent menu. The next three screens are used to select the desired project, version, period combination. The screens each operate in the same manner: the arrow keys are used to move the cursor to the desired item which is selected by pressing *enter*. Selecting Period "002", for example, loads the state of the project at the end of period two/beginning of period three.

The root file set stores the initial state of all versions (simulation runs) carried out under a particular project. The only available period when loading the root files is "000". If the root is not fully defined and the user requests it to be

loaded then it will become the active file set. If the root is fully defined and is requested to be loaded then a new version is created which becomes the active file set.

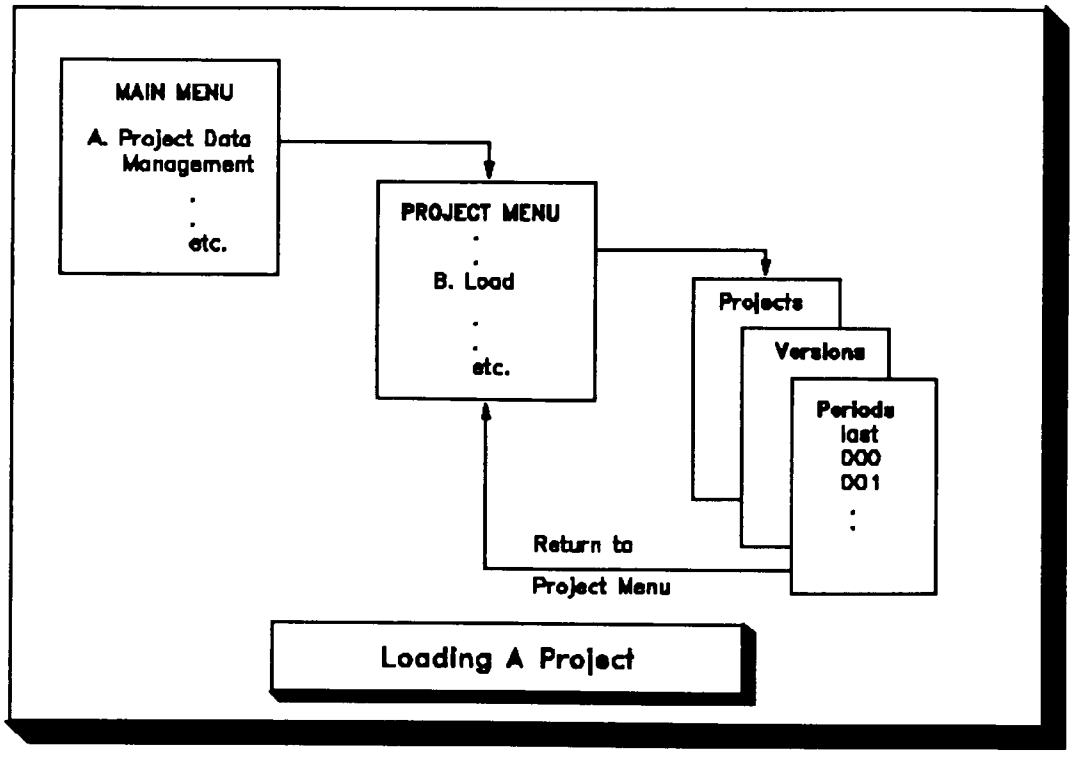


Figure A.2: Loading A Project Using The MSA Software

A.8 Editing and Viewing Data Files

Option "E. Edit" off the Data Input Menu displays the Project Editing Menu from which utilities may be selected to edit all of the four user project definition files.

For the experienced user a utility has been provided to view all of the project data files. Selecting "B. Utilities" off the Main Menu and "V. View Project Files"

off the Utilities menu displays a list of all the project files. Inputting the corresponding file number causes the contents of the file to be displayed on the screen.

A.9 Logic Check

It is possible to check the consistency of an active model by selecting "Logic Check" off the Project Data Management menu. The user is advised to perform a logic check prior to launching a simulation run. Inconsistent data may lead to spurious results and in extreme cases cause the software to terminate abnormally. The logic check performs the following three checks:

Resource Database

Checks to determine if definitions exist for all the work centres required by jobs in the pending file.

Sequencing Database 1

Determines if all jobs in the pending file have a defined Bill of Materials and Sequencing Database.

Sequencing Database 2

This check is failed if two products defined to the Bill of Materials database have the same name.

If any of the checks fail the edit option mentioned above may be used to correct the error.

A.10 Advance Simulation

Figure A.3 illustrates the process of launching a simulation run. After selecting the "Advance Simulation" option off the Main Menu, a "Welcome to the MSA Simulator" screen is displayed for ten seconds. The *esc* key may be pressed to bypass the delay.

The Advance Simulation Menu which is then displayed lists on the left hand side of the screen the eleven simulation parameters along with their present settings. The parameters may be edited before running the simulator by inputting the associated letter and pressing *enter*. The parameters are grouped into: master data, logic and output and are listed in Table A.1 along with the settings available. The next fourteen subsections describe each parameter and the available settings.

When all parameters have been set to the desired values the simulator is launched by selecting "S" off the menu on the right hand side of the screen.

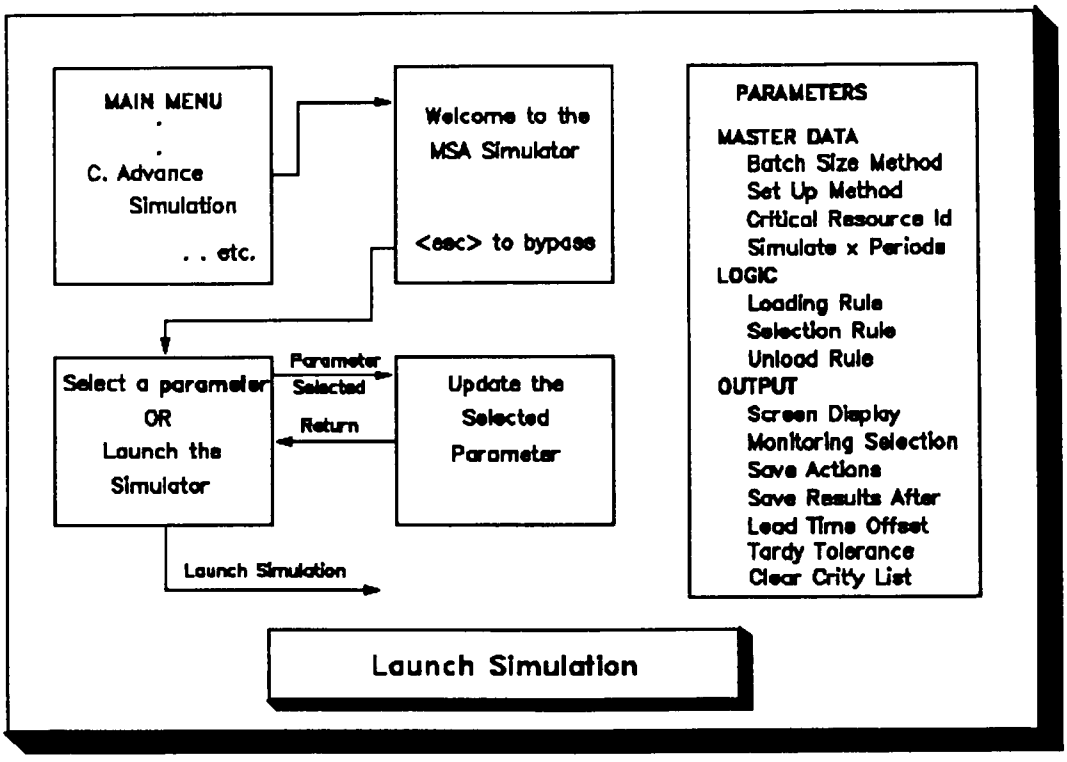


Figure A.3 : Initialising and Launching a Simulation Run

PARAMETERS	OPTIONS
Master Data	
A. Batch Size Method	None Global Product
B. Set Up Method	No Set Up Global Resource Product Part
C. Critical Resource Identification Method	Manual Random Loading Model 1 Loading Model 2 Queuing Model 1
D. Simulate Periods	Input a value (range 1-200)
Logic	
E. Loading Rule	Part then component queue Component then part queue Both together
F. Selection Rule	Random FCFS SPT EDD

Table A.1: Simulation Control Parameters

PARAMETERS	OPTIONS
Logic (cont.)	
G. Unload Logic	To Output Queue To next Input Queue
Output	
H. Screen Display	Screen 0 Screen 1 Screen 2
I. Monitoring Selection	Action Work Centre Part Product All
J. Simulation Actions	To screen To file and screen
K. Save Results After (periods)	Input a value (range 1-200)
L. Lead Time Offset	Input a value
M. Tardiness Tolerance	Input a value
N. Clear Criticality List	Yes/No

Table A.1: (Continued)

A.10.1 BATCH SIZE METHOD

The size of the transfer batch does not necessarily have to equal or divide exactly into that of the process batch. The complete process batch must be finished and unloaded from a work centre before any other job may be loaded onto it. The available batch size methods are:

None

Selecting none sets the size of the transfer batch equal to that of the process batch.

Global

One transfer batch size is defined for all end items, sub assemblies and components.

Product

Each individual product has a defined transfer batch size is held in the Job Pending File. It is applicable to all components, sub assemblies and the end item throughout the manufacturing process of a particular product and is activated by selecting this batch sizing option.

A.10.2 SET UP METHOD

Five available setting are available to define work centre set up times, from no set up time through to an individual part/centre dependent set up times. The available settings are:

No Set Up

No set up times are used.

Global

One set up time for all parts on all work centres.

Resource

Each work centre has an individual set up time which is used regardless of what part is to be processed.

Product

Each sub assembly and component of a product has the same set up time on

all work centres. Different products may have different set up times.

Part

Each part will have a specific set up time for each work centre visited, a part/work centre dependent set up time.

A.10.3 CRITICAL RESOURCE IDENTIFICATION METHOD

Five methods are available to identify the location of the Critical Resource. The first three methods are relatively quick to execute compared to the last two (CR-LM2 and CR-QM1). The last two methods require the current period to be simulated to proactively locate the Critical Resource before the actual simulation is run. The five Critical Resource selection models are:

Manual

The user may choose which of the defined work centres is to be labelled the Critical Resource and the choice is maintained throughout the number of periods which the user wishes to simulate. For example, if the simulation is to be run for five periods the same manual choice is maintained for all five.

Random

The Critical Resource is randomly chosen by the software at the beginning of each period simulated.

Loading Model 1

The loading which the required jobs will put onto the facility is calculated and the most overloaded resource is defined as the Critical Resource. This model also accounts for work-in-progress within the facility.

Loading Model 2

The simulator is run for the current period with no defined critical resource were jobs are pushed through the facility using the designated dispatching rule. The resource with the highest loading is labelled the critical resource.

Queuing Model 1

As in Loading Model 2 the simulator has to simulate through one period to identify the location of the Critical Resource. The model uses the values of the "time average queue processing time" and the "time average queue length" to locate the position of the critical resource in the manufacturing facility.

A.10.4 SIMULATE PERIODS

The required input is a figure between 1 and 200 and instructs the simulation driver on how many periods to simulate without a break. It is possible to simulate one period at a time enabling parameters to be reset on a period by period basis.

A.10.5 LOADING RULE

Each work centre defined to the MSA software has two input queues and one output queue. The two input queues are split between one holding components and the other sub assemblies. The loading rule dictates the order to access the input queues when a new job is required by a work centre. Three options are selectable:

Part Then Component Queue

The part queue is accessed first to ascertain whether any jobs exist for loading onto the idle work centre. If no eligible jobs exist the component queue is accessed to determine if a full set of components exist to load a job

onto the work centre.

Component Then Part Queue

Similar to the previous case but the component queue is accessed first followed by the part queue, to determine the jobs available for loading.

Both Together

Both queues are accessed in parallel to generate a list of jobs eligible for loading.

A.10.6 DISPATCHING RULE

When a resource completes a job and becomes idle, the above Loading Rule is used to generate a list of possible jobs. If no critical resource is selected hence the Criticality List is empty the dispatching rule selects a job from those eligible. Possible choices of the selection rule in the present version of the software are:

Random

A job is randomly selected from the list of possible jobs.

First Come First Served

The first job in the generated list is the one selected.

Shortest Processing Time

Of all jobs eligible to be loaded onto a work centre the one which can be processed in the shortest time is the one selected.

Earliest Due Date

The job with the earliest due date is selected for loading. The due date is that of the end item which the component or the sub assembly is to become part of.

A.10.7 UNLOAD RULE

This rule controls into which queue a component or sub assembly will be moved to after processing at a work centre. If there is insufficient room in the receiving queue chosen the job remains at the work centre which is then "blocked" and unable to process any other work until it is removed. Two settings exist:

To Output Queue

This moves the finished part to the output queue connected to the defined work centre.

Next Input Queue

This setting bypasses the output queue and moves a completed job directly to the input queue of the down stream work centre.

A.10.8 SCREEN DISPLAY

When the simulation is being executed three user information screens are available.

Screen Zero

Displays a message indicating that the simulation is running along with the percentage complete. This option speeds up a simulation run since screens do not have to be updated.

Screen One

The top half of the screen exhibits the simulation steps or actions as they happen. An example of an action is, "loading a job from an input queue to a work centre". The bottom half of the screen displays performance statistics for completed jobs and the performance of the Critical Resource.

Screen Two

The full screen is used to display the simulation actions as they occur.

A.10.9 MONITORING SELECTION

Screen one and two detailed above display actions of the simulator as they happen. Using the "Monitoring Selection" settings it is possible to display only those actions of interest. Five possible setting are available:

Action

Fifteen different actions are defined, for example, "move job from input queue to resource". Using this option it is possible to display one particular action on the screen and have it written to the run file, an archive copy of the actions.

Work Centre

Only those actions pertinent to a selected work centre are displayed. A possible use of this may be to display actions that occur to the identified Critical Resource.

Part

This may be used to display actions concerned with a particular component or sub assembly.

Product

Displays actions which occur to any component or sub assembly of a selected end item making it possible to track one particular product of interest.

All

All actions as they occur, are displayed on the simulation screens.

A.10.10 SIMULATION ACTIONS

The user is given the option of archiving the simulation actions. Two options are available.

To Screen

Selected actions are only sent to the screen if screen one or two was chosen under the screen display option mentioned above.

To File And Screen

Selecting this option simulation actions are displayed to the screen and archived to the hard disk.

A.10.11 SAVE RESULTS AFTER?

Results may not be required until after several periods have been simulated. In order to speed up the process of simulating up to this point the user is given the option of archiving results only after a number of periods have been simulated.

A.11 Results Output

Before using the report feature the project-version-period combination of interest must be the active problem under investigation. To access the "Report Print Menu" option F must be selected off the "Project Data Management" menu. Seven different reports may be generated and these are briefly described below. Option I on the Report Print Menu toggles between displaying the desired report on the screen or sending it to the printer, the default setting is to the screen.

When the report is displayed on the screen, the arrow keys may be used to move up and down it and by pressing the *esc* key control is passed back to Report Print Menu. The seven reports are briefly discussed below followed by example print outs of the reports (Figure A.4 to A.6):

Operating Parameters

Parameters such as batch size and set-up time method used are listed in this report.

Aggregate Data

This report contains data that is macro in nature, for example, number of jobs graduated in the active period or the number of work centres defined. The results are split into the four performance categories:

1. customer satisfaction;
2. effectiveness;
3. efficiency; and
4. WIP inventory.

Summary: Jobs

For each job graduated in the active period , regardless of when it was launched, a report is generated containing data pertinent to the particular job. For example, product name, job number, when launched and when completed.

Summary: Resources

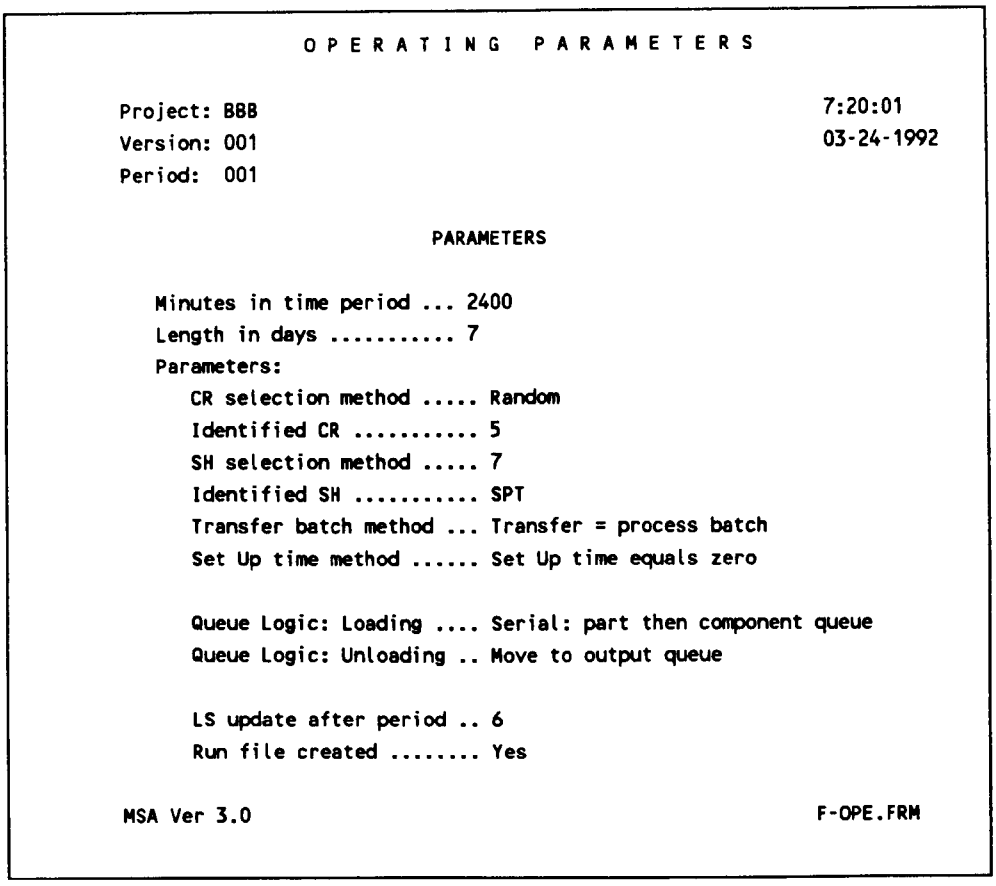
For all resources defined to the project a report is generated detailing performance and usage statistics.

Learning System Aggregates

For the active project, version, period combination this report lists the four categories of the learning system results and the overall performance figure for all the combinations of critical resource selection method and sequencing heuristic.

Print Run File

If the simulation actions where written to an action/run file this option will view/print the file. A second level menu provides the functionality to print the run file for all periods simulated, the last five or the current one.



```
                    O P E R A T I N G   P A R A M E T E R S

Project: BBB                                                    7:20:01
Version: 001                                                    03-24-1992
Period:  001

                                PARAMETERS

   Minutes in time period ... 2400
   Length in days ........... 7
   Parameters:
      CR selection method ..... Random
      Identified CR ........... 5
      SH selection method ..... 7
      Identified SH ........... SPT
      Transfer batch method ... Transfer = process batch
      Set Up time method ...... Set Up time equals zero

      Queue Logic: Loading .... Serial: part then component queue
      Queue Logic: Unloading .. Move to output queue

      LS update after period .. 6
      Run file created ........ Yes

MSA Ver 3.0                                                     F-OPE.FRM
```

Figure A.4 : Operating Parameter Report

S U M M A R Y D A T A

Project: BBB 17:20:04
Version: 001 03-24-1992
Period: 001

CUSTOMER SATISFACTION

No. of jobs completed . 0	Total early time 0
Early jobs 0	Total late time 0
Late jobs 0	Average early 0
Jobs on-time 0	Average late 0
	Total delivery error . 0

Frequency of tardy jobs ..... 0 %

EFFECTIVENESS

Total theoretical flow time .. 0	Mean flow time 0
Total actual flow time 0	Maximum flow time .... No jobs
Total manufacturing error 0	Minimum flow time .... No jobs
Total flow time (static) 0	

EFFICIENCY

Number of resources ... 12
Total time available:
Facility ..... 480 hours
All resources .... 480 hours

	Set Up	Wkg	Idle	Blkd	Wait	Util 1	Util 2
Percent:	0	79	21	0	0	79	79

WIP INVENTORY

PROCESSING TIME DATA:
Time average processing time 172

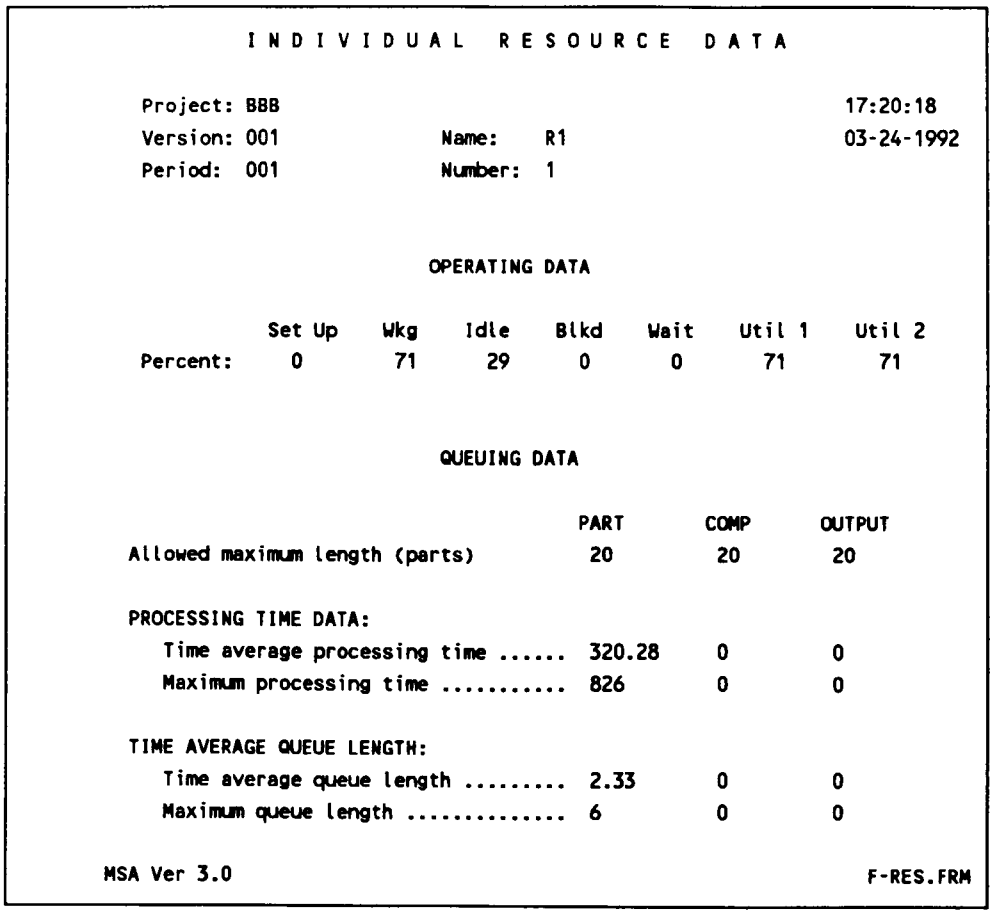
TIME AVERAGE QUEUE LENGTH:
Time average queue length 1

OVERALL SUMMARY

Measures:	Percentiles	Weighting:
Customer Performance	0	0.3333
Effectiveness	0	0.3333
Efficiency	78.5	0.3333
Overall Performance	26.16	

MSA Ver 3.0 F-SUM.FRM

Figure A.5 : Summary Data Report



INDIVIDUAL RESOURCE DATA

Project: BBB 17:20:18
Version: 001 Name: R1 03-24-1992
Period: 001 Number: 1

OPERATING DATA

	Set Up	Wkg	Idle	Blkd	Wait	Util 1	Util 2
Percent:	0	71	29	0	0	71	71

QUEUING DATA

	PART	COMP	OUTPUT
Allowed maximum length (parts)	20	20	20
PROCESSING TIME DATA:			
Time average processing time	320.28	0	0
Maximum processing time	826	0	0
TIME AVERAGE QUEUE LENGTH:			
Time average queue length	2.33	0	0
Maximum queue length	6	0	0

MSA Ver 3.0 F-RES.FRM

Figure A.6 : Individual Resource Data Report

```
"001 PERIOD                                                    "
"      Time              Res       Action       Part Prod  JobNo"
1     "W001 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  1     "
2     "W001 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  2     "
3     "W001 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  3     "
4     "W001 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  4     "
5     "W001 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  5     "
6     "W001 D01 H00 M00 1     PQ > Re  Load   PR1  PR1  1     "
7     "W001 D01 H00 M00 1     Begin Operating PR1  PR1  1     "
8     "W001 D01 H06 M00 1     Re > OQ  Unload PR1  PR1  1     "
9     "W001 D01 H06 M00 1     Move to Stock   PR1  PR1  1     "
10    "W001 D01 H06 M00 1     PQ > Re  Load   PR2  PR2  2     "
11    "W001 D01 H06 M00 1     Begin Operating PR2  PR2  2     "
12    "W001 D02 H03 M40 1     Re > OQ  Unload PR2  PR2  2     "
13    "W001 D02 H03 M40 1     Move to Stock   PR2  PR2  2     "
14    "W001 D02 H03 M40 1     PQ > Re  Load   PR3  PR3  3     "
15    "W001 D02 H03 M40 1     Begin Operating PR3  PR3  3     "
16    "W001 D03 H01 M00 1     Re > OQ  Unload PR3  PR3  3     "
17    "W001 D03 H01 M00 1     Move to Stock   PR3  PR3  3     "
18    "W001 D03 H01 M00 1     PQ > Re  Load   PR4  PR4  4     "
19    "W001 D03 H01 M00 1     Begin Operating PR4  PR4  4     "
20    "W001 D03 H07 M00 1     Re > OQ  Unload PR4  PR4  4     "
21    "W001 D03 H07 M00 1     Move to Stock   PR4  PR4  4     "
22    "W001 D03 H07 M00 1     PQ > Re  Load   PR5  PR5  5     "
23    "W001 D03 H07 M00 1     Begin Operating PR5  PR5  5     "
24    "W001 D04 H06 M00 1     Re > OQ  Unload PR5  PR5  5     "
25    "W001 D04 H06 M00 1     Move to Stock   PR5  PR5  5     "
26    "002 PERIOD                                              "
27    "Time             Res       Action       Part Prod  JobNo"
28    "W002 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  6     "
29    "W002 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  7     "
30    "W002 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  8     "
31    "W002 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  9     "
32    "W002 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  10    "
33    "W002 D01 H00 M00 1     PQ > Re  Load   PR1  PR1  6     "
34    "W002 D01 H00 M00 1     Begin Operating PR1  PR1  6     "
35    "W002 D02 H01 M40 1     Re > OQ  Unload PR1  PR1  6     "
36    "W002 D02 H01 M40 1     Move to Stock   PR1  PR1  6     "
37    "W002 D02 H01 M40 1     PQ > Re  Load   PR2  PR2  7     "
38    "W002 D02 H01 M40 1     Begin Operating PR2  PR2  7     "
39    "W002 D03 H02 M00 1     Re > OQ  Unload PR2  PR2  7     "
40    "W002 D03 H02 M00 1     Move to Stock   PR2  PR2  7     "
```

Figure A.7 : A Section of a Run File

APPENDIX B

MSA COMPUTER PROGRAMS AND DATABASES

B.1 Introduction

This appendix contains

- a glossary of the major variables used; and
- descriptions of procedures and function contained within each program.

A complete listing of the programs is included in Appendix F. For information on how to use the software successfully the reader is directed to read the User's Guide contained in Appendix A.

B.2 Glossary Of Major Variables

A variable name follows by a double parentheses indicates that it is a 2-D array. A variable name followed by a $ is a string variable and a % indicates an integer variable.

B.2.1 GENERAL AND OPERATION CONTROL

ac$()	user access control
ctl$()	software control
g$()	colour database (names)
g%()	colour database (parameters)
pjt$()	attributes of individual project
simu$()	simulation control
u1$()	user's permanent record
u2$()	user's dynamic record
wkgr	current row of on screen scrolling
wkg$()	holds the current events displayed on the screen during simulation action scrolling
Z	master control variable used when chaining

B.2.2 PROJECT DATA VARIABLES

bom$()	BoM file
bomho$()	temporary holding place for BoM file
cal$()	calendar
cr$()	critical resource selection history
job$()	jobs pending file
jobnum	maximum length of both the jobs pending file and current work file
mac$()	Universal Transfer definitions
ls$()	learning system database
pdt$()	project control data
resources	maximum number of Universal Transfers which can be defined
rltg$()	global results
rltj$()	results for completed jobs
rltm$()	results for Universal Transfers
sh$()	Dispatching Rule selection history

B.2.3 SCHEDULING DATA FILES

bn$()	loading of each job destined for the CR
bnsch$()	the Criticality List
COMP	array subscript for the component queue in the queues$() array
hold1$()	used in creating master Sequencing Database entries after a BoM has been input
INN	array subscript for the input queue in the queues$() array
load&()	loading per resource per period, used in LM1 model
now&	current simulation time in minutes

O	array subscript for the output queue in the queues$() array
qh$	used to hold a queue in order that it can be manipulated without affecting the original copy
queue$	to pass lists of jobs to procedure Rules and return the selected job
queues$()	holds the contents of all the queues for all defined Universal Transfers
R	array subscript for the Resource in the queues$() array
sq$()	run-time Sequencing Database
sqnum	maximum length of the run-time sequencing database
work$()	current work file

B.2.4 RESOURCE DATABASE FIELD NAMES

To access the resource database a record number must be passed to the sub procedure *GetRes*. The function of the procedure is to place the fifteen fields which comprise a record into the following variables names. The reason for this manipulation if to make the code more logical to write and easier to debug. The opposite subroutine is *PutRes* which writes the variables back into the resource database.

1.	nam$	name of resource
2.	recod$	resource code
3.	grup$	resource group
4.	simu$	number of jobs worked on simultaneously
5.	perf$	performance factor
6.	oper$	operational status
7.	setup$	set up time
8.	sre$	load job logic
9.	fir$	end processing logic

10.	sqo$	finish in output queue logic
11.	fqo$	output to input queue move logic
12.	outt$	maximum length of output queue
13.	inn$	maximum length of input queue
14.	comp$	maximum length of component queue
15.	rule$	dispatching rule to use

B.2.4 SEQUENCING DATABASE FIELD NAMES

To access the Sequencing Database a record number must be passed to the sub procedure *GetQue*. The function of the procedure is to place all the fields from the requested record into the following twenty one variables. For writing changes to the variables back to the Sequencing Database the *PutQue* sub procedure is used.

1.	num$	database record number
2.	sno$	record number of next process
3.	pro$	name of final product
4.	ini$	1 = gateway part
5.	mac$	resource to use
6.	spr$	succeeding part name
7.	srt$	start of holding time in output queue
8.	dur$	duration on resource in minutes to produce one of the item
9.	fin$	finish time of holding in the output queue
10.	suc$	resource to move the completed part to
11.	par$	the name of the part
12.	req$	how many different types of components have to be present to produce one of this part
13.	obt$	how of the required component are available in the input queues
14.	jno$	the job number

15	tpr$	how many of the total process batch are available
16.	tba$	total number of parts still to be produced
17.	tqu$	transfer batch quantity for send-aheads
18.	tti$	completion time of current transfer batch
19	cpt$	how many of the current part are required to produce one of the end item
20.	su1$	set up time on this resource
21.	su2$	global set up time

B.3 Group: Main

The programs in this group boot the software, restore seven system files and display to the user the Main Menu and the second level menus. This section discusses each program in the group Control. The discussion of a program covers the procedures and functions it contains and the parameters which are passed to and from the subroutines. Figure B.1 schematically displayed the hierarchy of the four programs contained in this group.

B.3.1 MSA

This is the only executable (.EXE) program in the MSA program suite and contains the opening graphics screens and the various settings for the operating environment. The program is only executed once during a session. Using the library program DIMEN the all the arrays are dimensioned before chaining to M-LIC.

B.3.3 M-LIC

Used to display the licence and copyright screens for the MSA software.

Licence (procedure)

purpose: when called the licence screen is displayed and pressing the <space bar> returns control to the calling function.

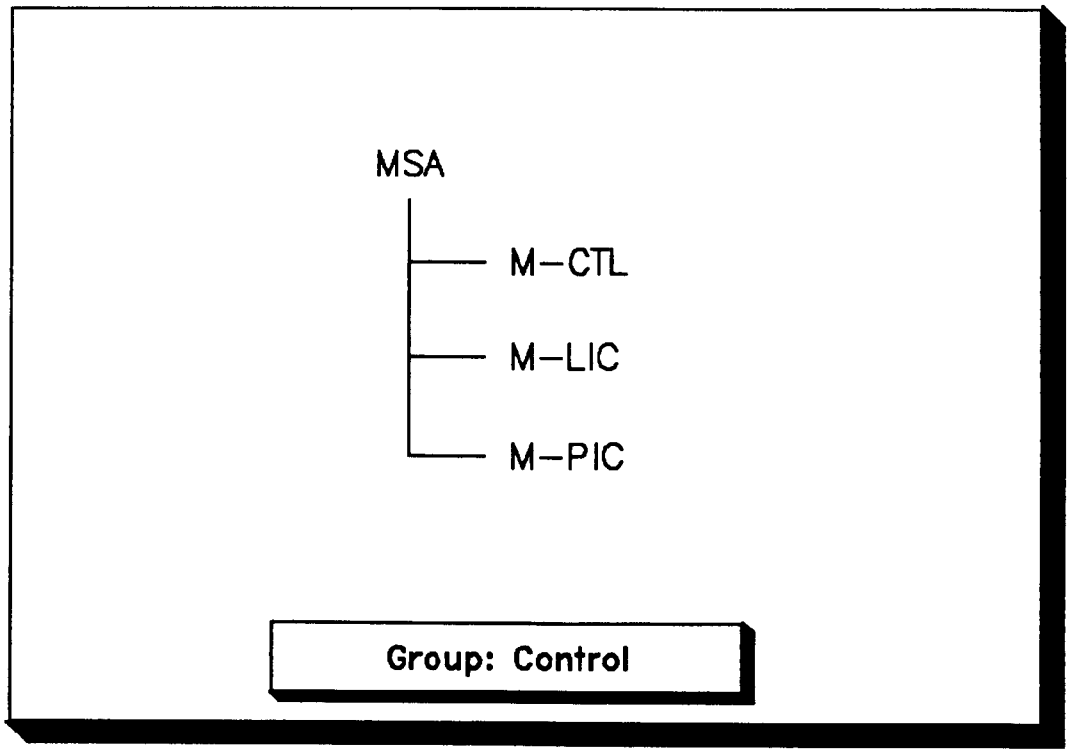


Figure B.1: Program Hierarchy in the Control Group

B.3.2 M-PIC

This program also does not contain any procedures, it simply restores the required software control files and passes control to program M-LIC.

B.3.4 M-CTL

This program achieves the primary control which is the main function of this program group. It contains the Main Menu, and menu levels 1 and 2.

go (procedure)

purpose: if the control variable Z = 1 then call MenuSel; if Z = 3 then call PjtProMenu.

InitSimData (procedure)

purpose: initialise the project control files back to the default values, in effect removing all defined projects. Before this procedure is executed the user is warned of the consequences and given the opportunity to back out.

HelpLogic (procedure)

purpose: supports the LogicCheck procedure. After the LogicCheck procedure has been executed the user is given the opportunity to view a help screen to explain the results of the checks. This procedure displays this help screen.

LogicCheck (procedure)

purpose: carries out three logic checks on the currently active model. The three checks are: all resources to be used are defined, all jobs to be launched have entries in the Sequencing Database and no duplicate definitions exist in the Sequencing Database.

MenuSel (procedure)

purpose: to display the Main Menu on the screen. From the Main Menu the user can select: Project Data Management, Advance Simulation or Learning System Update. When one of the options is selected the procedure is responsible for passing control to the required subroutine.

OverWrite (procedure)

purpose: a unique feature of the MSA methodology is the ability to overwrite the original root data set so as to define a new state at the beginning of the project. This function takes the currently active project-version-period combination and writes it to the root file set. A consequence of this is that all the other simulation runs are then not applicable and thus deleted. Before deleting the files the user is warned that data will be destroyed and is given the option to back out if required.

PjtKill (procedure)

purpose: a supporting procedure to wipe out all the data files of a project that the user has identified for deletion.

PjtProMenu (procedure)

purpose: to display and provide the functionality for the Project Data Management Menu. When the user selects an option from this menu this procedure is responsible for passing control to the required program. Project processing options available from this menu include: Create, Load, Delete, Edit, Logic Check, View/Print Reports and Overwrite Root.

Utilities (procedure)

purpose: to display the utilities menu and pass control to the desired operation once an option has been chosen. The options available off this menu include: Initialisation, View Licence Screen, View Development Team and View Project Files.

WipeEm (procedure)

purpose: initialises all the project files back to null values and is usually called

before loading a project so as to avoid spurious data being carried over from a previously active one.

B.4 Group: Project Management

This group contains seven programs to create, manipulate and delete the project data files. This section discusses the subroutines contained in each program and the parameters which must be passed to them. Figure B.2 depicts the hierarchy of the programs in this group: four primary and six support programs are defined.

The current version (3.0) of the software allows for fifteen individual projects to be defined. A project is a complete set of data, defining all required aspects of the manufacturing facility and work programme.

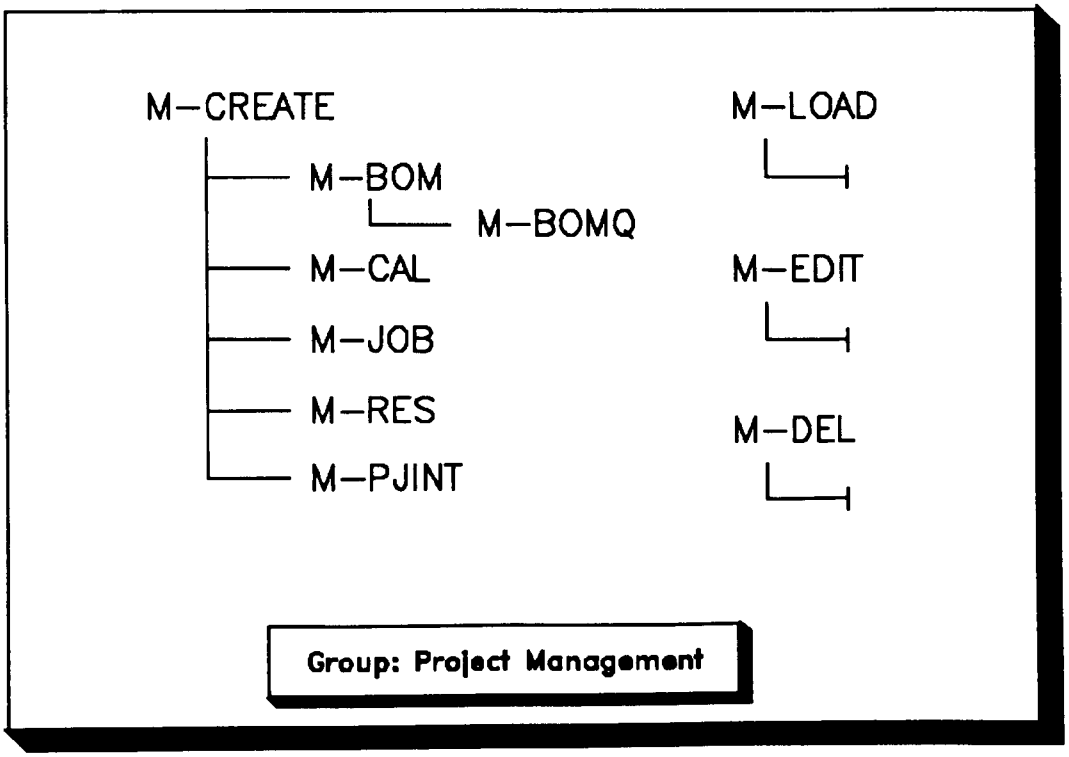


Figure B.2: Program Hierarchy in Project Management Group

B.4.1 M-CREATE

The functions of this program include creating the environment for naming a project (ensuring no duplication), creating the project files and presenting the user with the Data Input Menu. From this Data Input Menu a user can create a project.

go (procedure)

purpose: Z = 1: call NewPjtSetUp and PjtName. Z = 2: call CopyMenu.

CopyMenu (procedure)

purpose: to display the Data Input Menu. Input options off the menu include: Project Data, Jobs Pending, Resource Profiles, Bill of Materials, Calendar and Copy Existing Project. On the right hand side of the menu there is a user information section indicating whether the files have been created or not. The procedure controls the chaining to the separate programs when one of the input options is chosen. Some processing of the chosen function is carried out within this procedure.

FNattribs (function)

purpose: to print a project status box on the right hand side of the screen. The box displays the number of jobs, resources, products and weeks defined.

FNcopy (function)

purpose: to inform the user that previously defined project files exist and to prompt for whether these are to be used or a new version to be input. The previously defined files are files belonging to other projects.

NewPjtSetUp (procedure)

purpose: to update the control variables concerned with tracking the number of defined projects and the currently active one.

PjtName (procedure)

purpose: to prompt the user to input a unique three character project identifier which is to be used as the first three characters of all project files. This procedure displays a list of currently used identifiers and also provides error checking functionality. Once an identifier is accepted the user is prompted for a twenty five character description of the project, this is for user information only.

B.4.2 M-BOM

This program is primarily concerned with creating an environment for the user to input a BoM file in a logical manner.

go (procedure)

purpose: Z = 1: call DbBom; Z = 2: call DbBomExt; Z = 3: call DbBom; Z = 4: DbBomExt; Z = 5: call RunBom.

CreateBom (procedure)

purpose: to present the user with the screens to allow the logical input of a multi level, multi component BoM tree. The screens contain information in order that the user may orientate themselves Input to the BoM is carried out a level at a time as opposed to a branch at a time.

CreateBom1 (procedure)

purpose: to support CreateBom in the logical input of a BoM file.

DbBom (procedure)

purpose: to call off procedures to allow the user to input a BoM file.

DbBomExt (procedure)

purpose: to prompt the user if another BoM is to be input or return control to the calling program.

DbDel (procedure)

purpose: to delete a selected BoM file. The user is first prompted as to the product name of the BoM to be deleted and a last-chance is given to back out of the procedure before the BoM definition is deleted.

DbView (procedure)

purpose: to view and edit a BoM file.

FNInmac (function)

purpose: when the user is being prompted for a resource number the option is available to have a pop-up list of previously defined resources. This function creates and displays this list.

function identifier: returns the value of the selected resource.

RunBom (procedure)

purpose: to display the BoM Processing menu, options include: Input, View and Delete a BoM file.

B.4.3 M-BOMQ

A support program which creates from the user input BoM file, the entries in the Sequencing Database.

go (procedure)

purpose: both Z = 1: call QueueFromBom then set Z = 2 and chain to M-BOM; Z = 3: call QueueFromBom then set Z = 4 and chain to M-BOM.

FNcritPath (function)

purpose: to calculate the critical path through the BoM's and store the data in the Sequencing Database.

function identifier: returns the time to produce the Critical Path through the BoM.

QueueFromBom (procedure)

purpose: takes the BoM file and create the Sequencing Database entry and archive it.

B.4.4 M-CAL

Creates the environment for the user to input the number of hours the manufacturing facility is to be working in a standard week After input the functionality is provided to edit the file and in this way enter weeks where the facility is shutdown, or working a short week.

go (procedure)

purpose: Z = 1: call InitCal; Z = 2: call ReCal.

InitCal (procedure)

purpose: to provide the user interface for the input of a standard week, in hours.

ReCal (procedure)

purpose: the eighth column of the calendar file is the total number of minutes available in a week. Once the file has been edited this procedure recalculates the number of minutes to work per week.

B.4.5 M-JOB

This program creates the environment for the user to input the Jobs Pending file. In the present version the user can input a maximum of 200 jobs. This limit is set by the "JobNum" variable in the program DIMEN and by changing this value the maximum length of the jobs pending file may be increased.

go (procedure)

purpose: Z = 1: call NewJob; Z = 2: call JobMenu; Z = 3: call NewJob. The reasoning behind two values of Z being used to point to the same procedure is that the calls come from different programs and alternate exit parameters may be set.

EditJob (procedure)

purpose: to provide the interface and functionality to Select and Edit existing jobs.

JobMenu (procedure)

purpose: display the Job Processing menu. This menu includes Edit Job File and Input New Jobs. From this menu all other procedures in this

program are called.

NewJob (procedure)

purpose: present the user with the interface to input required data about jobs pending. After all the jobs have been input the data is written to the relevant file.

PrnCol1 (procedure)

purpose: to support procedure NewJob when creating the user interface for the input of new jobs.

B.4.6 M-RES

Provides the functions of inputting, editing, viewing and deleting of the data related to the Universal Transfers.

Two modes of input are implemented: specific and global. In specific mode the user works with only one UT at a time whereas in global mode any changes the user makes are applied to all the UT's defined.

go (procedure)

purpose: Z = 1: call InputRes; Z = 2: GlobalInput; Z = 3: call ResMenu.

ChangeRes2 (procedure)

purpose: prompts the user for answers to specific questions about the configuration of the Universal Transfer. The procedure consists of ten program units contained within a case statement with each unit responsible for displaying the options available and prompting the user to input the data.

DefaultRes (procedure)

purpose: to display on the screen the default values for the parameters which make up a Universal Transfer. It also prompts the user to either accept these defaults or go through a series of questions to specifically tailor the Universal Transfers to requirements.

EditRes (procedure)

purpose: provide the functionality to select and edit a Universal transfer from those defined to the currently active project. Error checking of the user input is also performed inside this procedure.

GlobalInput (procedure)

purpose: to prompt the user to input the names of the resources which are to be included in the current global input. Once this has been achieved and duplication of names checked the ChangeRes2 procedure is called to obtain the required input.

InputRes (procedure)

purpose: when only one Universal Transfer is being defined at the current time then this procedure is used. It again prompts the user to input a name for the Universal Transfer before passing control to ChangeRes2.

ResMenu (procedure)

purpose: displays the resource (or Universal Transfer) processing Menu on the screen. Options from the menu include: Input a specific Universal Transfer, Global Input (multiple input of a Universal Transfer) and Edit a Universal Transfer.

B.4.7 M-PJINT

As mentioned above an individual project consists of 12 project control files and 4 user configured data files. The function of M-PJINT is to initialise the 12 simulation control files to their default values.

go (procedure)

purpose: Z = 1: call PjtInit.

PjtInit (procedure)

purpose: to initialise to their default values all the arrays which are used to hold all the data about a project.

B.4.8 M-DEL

This program is used to delete the root files of a project and all version files off that root.

go (procedure)

purpose: Z = 1 and 2: call DelAttrib.

DelAttrib (procedure)

purpose: to list the existing projects on the screen and prompt the user to input the three character project code of the project that is to be deleted. The user is given one last opportunity to back out of deleting the file.

B.4.9 M-EDIT

This program uses segments of the six programs under M-CREATE, (reference Figure B.2) to edit the four user project files.

go (procedure)

purpose: Z = 1: call editmenu.

EditMenu (procedure)

purpose: to display a menu of files available for editing, including: Project Data, Jobs Pending, Resources, BoM and Calendar Files. The procedure chains to the control file to achieve whichever option is chosen.

B.4.10 M-LOAD

From the Main Menu the user can opt to load a previously defined project. This program performs this function.

go (procedure)

purpose: Z = 2: ExistPjtSetUp; Z = 7 through 14: read the required file from the chosen file set and copy it to the one that is currently in the process of being defined.

ExistPjtSetUp (procedure)

purpose: to read the desired file set into the software arrays. If the one loaded is a fully defined root file set then a new "version" is created and this becomes the active file set. In this way a user does not inadvertently edit a project root file set.

FilePickBox (procedure)

purpose: to display the instructions on how to load a project file set. This procedure is called in conjunction with Projects which provides the functionality to select and load a file set.

Projects (procedure)

purpose: provides the interface for the user to select a project-version-period combination to load. Firstly, all the names of all the available projects are displayed on the screen. To select a project the user has to move the cursor adjacent to the one of interest and press <enter>. The process is the same for selecting the version and the period to load.

B.5 Group: Simulation

The program group Simulation contains fourteen programs, seven of which implement the *sim* modules (sim 1 to 7: reference Figure 5.2), the remaining eight playing a control and support role. This section discusses the subroutines contained in each program and the parameters which must be passed to them. Figure B.3 contains illustrates the control hierarchy of the programs in this group.

B.5.1 ADV

In Chapter five the operation of the simulation module was outlined. It detailed how before any simulation model is run the user is given the option of editing any of 16 control parameters. Program ADV creates the environment for the user to edit these parameters before control is passed to M-SEQ.

go (procedure)

purpose: Z = 0: call Welcome, call Border, call Advance; Z = 1: call Na; Z = 2: call Advance.

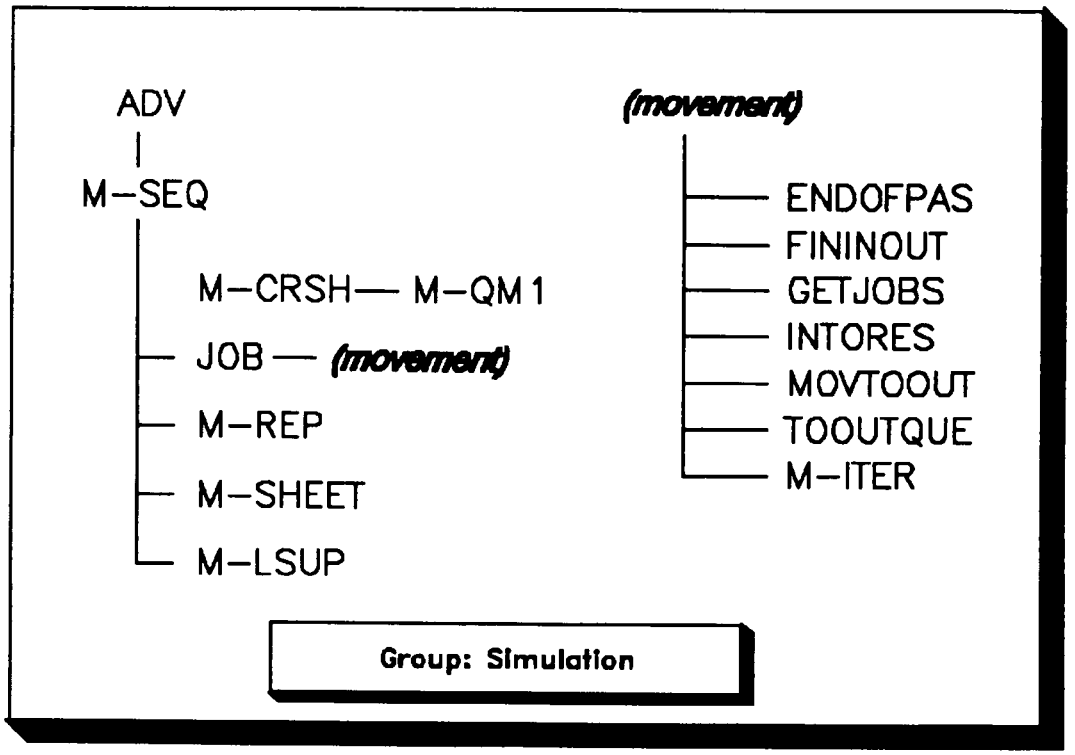


Figure 6.6: Program Hierarchy in Simulation Group

Advance (procedure)

purpose: when the Advance Simulation option is chosen off the Main Menu this procedure is fired. It provides the user interface to edit the default values for the simulation control parameters with error checking of the user inputs carried out inside the procedure. From the menu displayed the following options are available: Select Parameter to Edit, Launch Simulation, Reset Parameters.

SimVar (procedure)

purpose: when a simulation is being run the user may opt to track a variable. This procedure is called by Advance to provide the screens necessary to allow a user to select a variables to track.

Welcome (procedure)

purpose: displays a "welcome to the MSA simulator" screen. It is displayed for ten seconds which may be bypassed by pressing the <esc> key.

B.5.2 M-SEQ

The role of M-SEQ is to configure the simulation database with respect to the parameter values set by the user in program ADV.

go (procedure)

purpose: Z = 1: call MdUpdate, FinWhen; Z = 2: call PdtToMac, MinCount, ResultCfg, GetLsd, GetWork, GetSq, CrSel, PeriodAdd, GetBnSch and OpenRun. When Z = 3 the same procedures are called as for Z = 2 but the chaining parameters are different.

FinWhen (procedure)

purpose: it is possible to simulate more than one period or week. From the specific inputs made in the Advance procedure this procedure determines after which simulated week to start archiving the results and after when to stop the simulation.

GetLsd (procedure)

purpose: by using the value for the Critical Path through the BoM's this procedure steps back from the due date to find the Latest Start Date of a job. The assumption made is that the Critical Path is valid if the job was scheduled through a facility which was empty and using the currently selected batch sizing rule.

GetWork (procedure)

purpose: the Jobs Pending file holds a complete list of jobs for all jobs requiring processing by the facility in the coming weeks. If the Latest Start Date of a job falls within the week to be simulated then this procedure moves the job to the Work File.

MinCount (procedure)

purpose: the calendar for the weeks to simulate is in days and the basic unit for the simulation is in minutes. From the menu provided by Advance the user could input the number of weeks that the simulation is to be run for, this procedure calculates how many minutes to simulate to complete the next week.

OrderWork (procedure)

purpose: to order the work file with respect to the Criticality List. The effect of this is to split the work file into two lists. The first or top portion is a list of jobs that have components which are destined to go through the Critical Resource. The second portion of the list are jobs whose components are not going to be going through the Critical Resource.

PdtToMac (procedure)

purpose: the user configured simulation parameter settings which

correspond to the defined resources are applied to the database of resources by this procedure.

PeriodAdd (procedure)

purpose: increment the period or week counter.

ResultCfg (procedure)

purpose: at the end of a simulated period all the jobs that where active in the period are stored as part of the jobs results file. This procedure is run before the next period is begun to remove from the file all jobs which where completed in the previous period.

ResultInit (procedure)

purpose: the timer counting the minutes a simulation has been running begins counting at zero at the beginning of each period. Certain state variables need to be initialised to take account of this, this procedure performs this action.

B.5.3 M-CRSH

This program is a support program and does not then have a subroutine *go*. Procedures within it are simply called off by the primary program in which it is included as a library.

CrSel (p1) (procedure)

purpose: when the less processing intensive Critical Resource Identification models are used (RND, MAN and LM1) then the processing is accomplished by this procedure. The more complex models (QM1 and LM2) are evaluated partially by this procedure and partially by the

program M-QM1.

p1: a value to be used as a case statement control variable indicating which identification method to use.

B.5.4 M-QM1

The Manual, Random and Loading Model 1 methods of identifying the Critical Resource do not require that the period of interest be simulated. Loading Model 2 (LM2) and Queuing model 1 (QM1) require the period to be simulated necessitating further processing which is the function of this program.

go (procedure)

purpose: Z = 1: restore files and call M-SEQ to simulate them through the model; Z = 3: restore files and chain to the Learning System Update program; Z = 4: store the results of the execution of the QM1 and the LM2 models and restore the original version of the project.

FNlocationLM2 (function)

purpose: to determine from the results files the location of the Critical Resource when the LM2 model is used.

function identifier: the function identifier returns the value of the resource number of the Critical Resource selected when the LM2 model was used.

FNlocationQM1 (function)

purpose: to determine from the result files the location of the Critical Resource when the QM1 model is used.

function identifier: the function identifier returns the value of the resource

number of the Critical Resource selected when the QM1 model was used.

B.5.5 JOB

JOB is the simulation driver which calls off the five simulation *movement* (moving jobs between queues) programs and the two "housekeeping" programs.

When the simulation is running it gives the appearance of iterating around the five *movement* programs and when the simulation stopping conditions are met passing control to *m-iter*. After each *movement* program is executed however control is passed back to program JOB. The reason behind this strategy of operation is to have tighter control of the order of execution of the programs and simplify the process of adding programs, removing movement programs, or changing the order of execution. In certain cases the simulation driver detects that a *movement* program will achieve nothing since no jobs exist in the relevant queues hence it is bypassed cutting execution times.

go (procedure)

purpose: Z = 1: call JobLoadUp; Z = 2: call JobLoadUp2.

JobLoadUp (procedure)

purpose: to create the environment for the simulation to run in, that is, setting up the screen displays.

JobLoadUp2 (procedure)

purpose: this procedure is in effect the simulation driver whose function is to call off the movement procedures in a particular order.

The following six programs are the *movement* programs. Programming Model 2 (ref: Chapter 6) was used in the design and coding of the programs. The *movement* programs have only two procedures: go and move, the go procedure simply applies the move procedure to all the defined UT's. Each move procedure is basically a series of program units enclosed in a Case statement and by calling the units in differing orders a variety of queue processing logics may be achieved.

B.5.6 ENDOFPAS

go (procedure)

purpose: this program is run at the end of the fathoming of a particular event time. After calling the procedure EndOfPas, a comparison is made to determine if the end of the current period has been reached. Depending on the result the program flow is either directed back to the simulation driver or to M-ITER to "clean-up" at the end of a simulation.

EndOfPas (procedure)

purpose: to first determine the next event time. If the next event time is within the current period being simulated then global results are evaluated and the on-screen status displays updated. Although disabled in the current version the interrupt handling code is included in this procedure. All other procedures in this program provide support for this procedure.

FNproTime (p1, p2) (procedure)

purpose: to determine the processing time of the jobs in a queue.

function identifier: the processing time of the queue of interest.

p1: the identification number of the resource of interest.

p2: the queue of interest, either input or component.

PerLine (procedure)

purpose: to print the current simulation time on the status line displayed at the bottom of the screen when the simulation is running. This procedure is only called if screen 1 or screen 2 are currently active.

B.5.7 FININOUT

FinInOut (procedure)

purpose: to calculate and store the finish time for holding the job in the output queue. This function is provided for future researchers to use, in the present version the holding time is set to zero hence only the current time is stored.

B.5.8 GETJOBS

go (procedure)

purpose: to call off the GetJobs procedure and chain back to the simulation driver.

GetJobs (procedure)

purpose: to determine if it is possible to load a job from the Work list onto the required gateway resources. If one of the required input queues is blocked then the status of the job in the Work file is changed to "blocked". If a job is launched successfully then a record for the job is created in the jobs results file.

B.5.9 INTORES

go (procedure)

purpose: to apply the IntoRes procedure to each of the defined resources.

IntoRes (procedure)

purpose: to load a job from the input queues onto an idle resource. Twelve program units are included with a Case statement within the procedure.

case 0: to end the set up process and start processing of the designated job.

case 1: check to determine if the input queues are empty.

case 2: clear the qh$ (queue hold) variable.

case 3: clear the queue$ variable.

case 4: load the input queue data into the qh$ variable. If this queue is empty load the component queue.

case 5: load the component queue into the qh$ variable. If this queue is empty load the input queue.

case 6: load both the input and the component queue into the qh$ variable. The input queue is loaded first which is significant when the First-Come-First-Served rule is used.

case 7: considering the contents of the qh$ variable, determine which jobs have a full set of components and hence are eligible for consideration in the selection and loading process. Store the ten digit identifier of those jobs which are eligible in the queue$ variable.

case 8: if queue$ is non empty the call the Rules procedure with queue$ passed as an argument. Queue$ is also used by the Rules procedure to return the ten digit code of the job selected.

case 9: process the loading of the selected job onto the resource. This involves adding the job to the section of the queues$() array and also

removing the components the newly loaded job uses from the respective input queues. The second function of this program unit is to update state variables: with reference to the job loaded the completion time for the first transfer batch is generated and stored in the Sequencing Database.

case 10: exit routine.

case 14: if the output queue is full then the status of the resource is set to "blocked" hence no other job can be loaded.

B.5.10 TOOUTQUE

go (procedure)

purpose: to apply the ToOutQue procedure to all of the defined resources.

ToOutQue (p1) (procedure)

purpose: to determine if a transfer batch of a job has completed processing and if this is the case move the job to the next output queue. If the process batch is complete the status of the resource is set to "idle".

p1: the identification number of the resource of interest.

case 1: move a completed transfer batch to the output queue. If the process batch is complete then the UT is given a status of "idle".

case 2: move a completed transfer batch directly to the respective input queue of the downstream resource bypassing the output queue and again if the process batch is complete the status of the UT is set to "idle".

case 3: If not all transfer batches have been completed then parts are removed from the input queues and a new transfer batch completion time is generated. If an insufficient number of parts are available to produce another transfer batch then the status of the UT is set to "No

Work".

B.5.11 MOVTOOUT

go (procedure)

purpose: to apply the MovToOut procedure to all the resources in sequential order.

MovToOut (p1) (procedure)

purpose: to move jobs from the output queues, either to the next input queue or to stock if processing is complete.

p1: the identification number of the resource to apply the procedure to.

case 1: if the resource is the last on the route and the process batch is complete then the job is moved from the output queue to stock and the job completion statistics are updated.

case 2: move the job from the output queue to the relevant input queue of the downstream resource.

case 3: if there is insufficient room in the output queues then set the status of the job to "blocked" by placing a "B" in the srt$ and fin$ fields of the Sequencing Database.

FNmins (p1, p2) (function)

purpose: with reference to the defined calendar, determine the number of minutes between *p1* and *p2*.

function identifier: return the number of minutes.

p1: the first of the nine digit identifiers of a particular point in time. The break down is: 3 digits for the week, 2 for the day, 2 for the hour and 2 for the minute count.

p2: the second 9 digit time identifier.

B.5.12 M-ITER

go (procedure)

purpose: Z = 1: call EndBucket, IterDump, PeriodCount; Z = 2 and 3: call GlobalCalc; Z = 4: call OverRun, EndBucket; Z = 5: if any more periods to simulate then Z = 2 else it equals 3.

OverRun (procedure)

purpose: at the beginning of a subsequent period the current time is reset to zero. The completion times for the transfer batches needs to be reset with reference to this value.

EndBucket (procedure)

purpose: to "clean-up" the databases at the end of a simulation run. For example, removing completed jobs from the Sequencing Database, Work and Job File and Criticality List. Other "clean-up" functions include configuring the timing mechanism so that it reflects the beginning of the next period and not the end of the current one.

IterDump (procedure)

purpose: archive the simulation files, which then reflect the ending state of the period as well as the opening state of the next period.

GlobalCalc (procedure)

purpose: calculate those global results which it is only logical to evaluate at the end of a simulated period, for example, average processing time of all jobs completed in the current period.

B.5.12 M-LSUP

Called directly from the main menu M-LSUP is responsible for coordinating the collection of the data for the Learning System Database.

go (procedure)

purpose: Z = 2: configure the environment; Z = 3: restore the project data files; Z = 4: store the Learning System Database files; Z = 5: execute QM1 and LM2, and step through the CR identification methods and Dispatching Rules. The majority of the processing is carried out inside the Go procedure calling off procedures from other support programs.

project (procedure)

purpose: to prompt the user to select the period to begin the Learning System update at.

LoadingOn (procedure)

purpose: To display a message indicating that files are being loaded.

LoadingOff (procedure)

purpose: To remove the massage displayed by the LoadingOn procedure.

B.6 Group: Library Programs

Library programs are collections of subroutines which are grouped by functionality. Unlike primary and support programs the library programs are never chained to hence do not take control of the operation of the software, the routines are called off by the primary and the support programs The seven libraries are briefly discussed below:

B.6.1 COMMON

Each primary or support program requires that the program COMMON be called on the first line. The program COMMON contains only one command *common* followed by a list of sixty five variables. This command enables the values of the defined variables to be passed between *chained* programs.

B.6.2 DIMEN

This program is also only executed once on system start up by program MSA and contains thirty five dimension statements to initialise all the arrays which are used.

B.6.3 SHARED

Global variables are not allowed for in PowerBASIC as they are in IBM BASICA or high level languages such as C or PASCAL. For this reason the first line in each sub-procedure or Function requiring access to the complete set of variables must begin with the statement:

```
$include "shared.pro"
```

This statement enables the values of all the variables defined in the "shared.pro" library to be *carried into* a subroutine. Not including this line at the beginning of a subroutine definition will cause variables to be treated as local and be initialised to zero or null string.

B.6.4 M-COM

ToUpper (p1) (procedure)

purpose: changes the string supplied via the parameter *p1* to uppercase.

p1: string to be changed. Used to pass the variable in and to return it to the

calling routine.

FNletter (p1) (procedure)

purpose: returns a 1 if the string *p1* contains no numbers, all other ASCII characters are accepted.

p1: the string to test

Border (procedure)

purpose: to draw the three line screen header at the top of the screen. The block is displayed the majority of the time the software is running and contains information concerning the currently active project, the name of the software and the copyright holders.

WaitOn (procedure)

purpose: to display a flashing "WAIT" at the bottom right hand corner of the principle output.

WaitOff (procedure)

purpose: to remove the "WAIT" sign displayed by the previous procedure.

ScrClear (procedure)

purpose: clear the screen to color 0,3 (light blue).

InSound (procedure)

purpose: sound one beep to prompt the user that keyboard input is required.

FailSound (procedure)

purpose: a lower pitched two beep sound to indicate to the user that incorrect data has been input.

Fill (p1, p2, p3, p4, p5, p6) (procedure)

purpose: to display a solid box on the screen.

p1: foreground colour.

p2: background colour.

p3: distance down the screen of the top left hand corner of the box to be drawn.

p4: distance from left hand side of screen of the top left hand corner of the box.

p5: length of box on the x axis.

p6: length of box on the y axis.

Shadow (p1, p2, p3, p4, p5, p6) (procedure)

purpose: to give a box drawn by the previous procedure a shadowed or 3-D effect.

parameters: the parameters are the same as those for procedure Fill and the same values should be supplied to this procedure except two color parameters, p1 and p2.

Box (p1, p2, p3, p4, p5, p6) (procedure)

purpose: to draw a non-filled box with a double line edge.

parameters: same as the procedure Fill, above. To produce a filled box first call Fill and then use this procedure to draw the outline of the box. The parameters are the same as for the Fill and Shadow procedures.

BoxSing (p1, p2, p3, p4, p5, p6) (procedure)

purpose: to draw a non-filled box with a single line edge.

parameters: same as the procedure Fill above.

FNinData (p1, p2, p3) (function)

purpose: to obtain input from the user via the keyboard. The parameters are used to control where on the screen the user typed input is displayed and the number of characters allowed.

function identifier: returns the user typed input after the <enter> key is pressed.

p1: distance from the top of the screen to display the prompt for the user to input data.

p2: distance from the left hand side of the screen of the displayed prompt.

p3: the maximum length of the string allowed to be input.

FNinNum (p1, p2, p3) (function)

purpose: to test if a number is within a certain range.

function identifier: returns a -1 if the value not within the range (inclusive), 0 if within.

p1: the lower bound of the range.

p2: the upper bound of the range.

p3: the number to be checked.

FNtrun (p1) (function)

purpose: to remove the leading and trailing blanks from a string variable.

function identifier: returns the string with blanks stripped.

p1: the string variable that the blanks are to be stripped from.

Menu (p1 - p13) (procedure)

purpose: generic menu generator.

p1: the title to be displayed at the top of the menu.

p2-p13: the options to be displayed upon the menu.

Menu2 (p1 - p13) (procedure)

purpose: generic second level menu generator. When displayed on the screen it is two characters down and two characters to the right of the menu generated by procedure Menu.

parameters: The parameters supplied are exactly the same as to procedure: Menu..

EdArray (p1, p2, p3, p4) (procedure)

purpose: to emulate a simple spread sheet to allow the user to edit and/or view 2-D string arrays. The user can only view 9 rows by 6 columns. To move around the spread sheet the directional arrows, home, end, Page up and Page Down keys may be used.

p1: the name of the two dimensional array to be viewed.

p2: the number of rows in the arrays.

p3: the number of columns in the array.

p4: a return flag which if it equals 1 indicates that the array was edited and hence archiving may need to be carried out.

Put5 (p1) (procedure)

purpose: to add blank spaces to a string variable so the overall length is five characters. If the variable is, for example, 8 character long it is lengthened to 10.

p1: the string variable to be lengthened.

Put10 (p1) (procedure)

purpose: to lengthen a string variable to 10 characters long by adding blank spaces to the end of it.

p1: the string variable to be lengthened.

FNrndInt (p1) (function)

purpose: a function to return a random number on the interval U[1,n].

function identifier: returns the random integer.

p1: the upper limit for the number.

Spce (procedure)

purpose: to lock out all other keys and wait for the <space bar> to be pressed.

Waiting (procedure)

purpose: to keep calling the time procedure until a key is pressed. Used whilst waiting for the user to input data.

Time (procedure)

purpose: prints the current time, if it has changed, at the bottom left hand corner.

RBlock (procedure)

purpose: prints the information block in the top right of the screen. Information contained is the names of the currently active project, version and period.

Blocks (procedure)

purpose: print the left hand top corner information block. Information contained is the name of the software and the copyright owner.

B.6.5 M-FILE

By the nature of data stored in certain files and the storage requirements

different procedures exist for reading and writing files to disk.

DumpHo (p1, p2, p3, p4) (procedure)

purpose: to write a string array to a sequential file.

p1: the file to write the array into.

p2: used to pass the array to the procedure.

p3: the number of rows in the array.

p4: the number of columns in the array.

RestorHo (p1, p2, p3, p4) (procedure)

purpose: to restore files archived using the procedure DumpHo.

parameters: the parameters are the same as for the procedure DumpHo.

DumpBom (p1) (procedure)

purpose: to archive the currently active BoM file to disk.

p1: the name of the file the BoM file is to be archived to.

RestoreBom (p1) (procedure)

purpose: to restore a BoM file which was archived using the DumpBom procedure.

p1: the name of the file the BoM file is to be read from.

DumpRes (p1) (procedure)

purpose: to archive the resource database to disk.

p1: the name of the file to archive the resource database to.

RestoreRes (p1) (procedure)

purpose: to restore to memory a resource database file which was archived using the DumpRes procedure.

p1: the name of the file to read the resource database from.

DumpSq (p1) (procedure)

purpose: used for archiving to disk the Sequencing Database.

p1: the name of the output file.

RestSqdb (p1) (procedure)

purpose: to restore the Sequencing Database archived using the DumpSq procedure.

p1: the name of the input file.

Dump (p1, p2, p3) (procedure)

purpose: to call off the PjtDump procedure and archive the complete active project file set.

p1: the three digit project code.

p2: the file extension to use. ".inf" for a root file or a number, for example "002", for a file belonging to a version.

p3: the period being dumped.

RestoreFiles (p1, p2, p3) (procedure)

purpose: to restore a complete project file set from disk into the software arrays using the procedure PjtRest.

parameters: the same as for the procedure Dump.

PjtDump (p1,p2, p3, p4, p5) (procedure)

purpose: to write the desired array to disk. Data files are stored sequentially, for example, the jobs pending file for all periods for a particular project-version are stored in the one file. It is then conceivable that data for period 002 may need to be "slotted-in", this procedure performs this

function.

p1: the name of the file the array is to be written to.

p2: the parameter for passing the array into this procedure.

p3: the number of rows in the array.

p4: the number of columns in the array.

p5: the period.

PjtRest (p1,p2, p3) (procedure)

purpose: to restore from a file an array which was archived using the PjtDump procedure. The procedure is capable of extracting from a file the data pertinent to a period, even when it is not the last period archived.

p1: the name of the file the array is to be restored from.

p2: the parameter to pass the array into and out of the procedure.

p3: the desired period to be loaded.

ArrayWipe (procedure)

purpose: to clear all arrays which are used to hold the data about the active project. This procedure is normally called before loading a project to ensure that no spurious data is carried over from a previously active project.

B.6.6 M-SCR

ScrBoxes (procedure)

purpose: to call off the procedures to draw and update the user information screens when the simulation is running.

The following procedures each have the same functions hence a generic set

is described. The *XX* indicates a number. The numbers used are 0, 1 and 2.

ScrXXBox (procedure)

purpose: to draw on the screen outline for the user information screen, number *XX*.

ScrXXNum (procedure)

purpose: to update the information on user information screen, number *XX*.

B.6.7 M-SIMCOM

Working (procedure)

purpose: the user can indicate which events to track, anything from a single action such as "move-to-stock" to all events and corresponding actions. This procedure decides whether the action that has just occurred is one that the user is interested in and if the result of this test is positive then the working1 procedure is called.

Working1 (procedure)

purpose: to update the Wkg$() array which is used on the scrolling of the screen number 1 and 2. It also controls the output to the Run file which is a file listing all the actions as they occurred.

Util (p1, p2) (procedure)

purpose: the resource results file tracks how long each resource was in various states (e.g. idle or working). Every time the status of a resource changes this procedure is called to record how long it was in the previous state, record the new state and the time of the state change.

p1: the identification number of the resource that the statistics are to be updated for.

p2: the new status.

GetQue (p1) (procedure)

purpose: to read a record from the Sequencing Database and load it into a series of twenty one string variables detailed in Section B.2.3.

p1: the row number in the Sequencing Database of the record to be read.

PutQue (p1) (procedure)

purpose: the opposite of GetQue, this procedure writes the record back to the Sequencing Database.

p1: the row number in the Sequencing Database of the record to be written.

GetRes (p1) (procedure)

purpose: to read a record from the Resource Database into the series of string variables discussed in Section B.2.3.

StatusLine (procedure)

purpose: to draw and maintain the simulation status line displayed at the bottom of the screen when a simulation is running.

UpTo (procedure)

purpose: the simulation driver uses minutes as the standard time unit but thee user is interested in weeks, days, hours and minutes. This procedure take the output from ActTime and prints in the top right of the screen.

ActTime (procedure)

purpose: to convert the minute count used by the simulation driver and store the week, day, hour, minute values in the Pdt$() array.

remstr (p1, p2) (procedure)

purpose: ten character strings are used by the simulator to represent jobs within the facility. A queue is then made up of a series of ten digit blocks and it is feasible that a job may have to be removed from the middle of a queue, which this procedure carries out.

p1: the string to remove the ten characters from.

p2: the location of the block to remove in ten digit blocks.

GetSq (procedure)

purpose: to generate the run-time Sequencing Database by calling off the series of procedures: GetSim and GetSetUp.

GetSim (p1) (procedure)

purpose: takes a job in the Work file and generates the entries for the run-time Sequencing Database. This involves copying from the master database as well as initialising blank fields, for example, total batch size and set up time.

p1: the row, in the Work file, of the job that is of interest.

GetSetUp (procedure)

purpose: depending on the user configured simulation parameters the set up time of resources will come from different databases. This procedure decides what the length of the set up time is and stores the value in field twenty of the Sequencing Database.

GetBnJobs (p1, p2) (procedure)

purpose: create a list of individual components and sub assemblies which are to be processed by the designated Critical Resource.

p1: the resource identification number of the designated Critical Resource.

p2: the column number in the BnSch$() array in which the list of jobs will be stored.

GetBnSch (procedure)

purpose: to create the Criticality List and store it in BnSch$() array. The procedure GetBnJobs is called to generate the list and this procedure orders them with respect to the selected Dispatching Rule.

Rules (p1) (procedure)

purpose: to take a list of jobs passed to it via p1, and using the selected Dispatching Rule, select one of them. This one job, represented by a ten digit code is passed back to the calling routine via p1.

p1: the list of jobs from which one is to be chosen.

B.7 Database Structure

The database contains two sets of files: Systems and Project. This section presents the file name and outlines the data it contains.

B.7.1 SYSTEM FILES

Six system files are used to control the operation of the software from the colour of the screens and menus to attributes of projects which have been created.

f-ctl.inf

Maintains the current status of the software when it is operating, for example which project, version, period combination is active, the current user etc.

f-pjt.inf

Fifteen individual records are contained in this file, each one pertaining to currently created projects. Examples of information held are the project code, descriptor, how many versions exist and whether all the root files have been created.

f-gra.inf

Holds user interface parameters, for example, screen and menu colours, and the settings for the variety of sounds made by the software, for example, incorrect input sound.

f-uin.inf

A dynamic user information file, recording all the actions of the each user, for example time logged in and time logged out of the software, how many times logged in.

f-usr.inf

A static user information set: name, address, password and other non-changing data.

f-acc.inf

Contains a list of all possible actions contained in the software suite and

whether a specific user can access it.

B.7.2 PROJECT FILES

Each project has a set of root files which contain the base starting condition for any simulation run executed under that particular project. In this section:

xyz represents the three letter project code.

abc represents the DOS extension to the file name. A file extension of *inf* indicates that the file is a root file, a numeric extension, (e.g. *001*) indicates it is a simulation run 1 file

The first four files in the following list hold the user input information which to the user defines the facility, the remaining files are generated by the use of MSA software functions.

xyz~bom.abc
Bill of Materials, routing and processing times.

xyz~cal.abc
200 week calendar file.

xyz~mac.abc
All the Universal Transfer definitions.

xyz~job.abc
Jobs Pending.

xyz~wrk.abc
Holds the Work file created by M-SEQ from xyz~job.abc.

xyz~sqd.abc

The static version of the Sequencing Database.

xyz~sqr.abc

The run time version of the Sequencing Database created by M-SEQ using the static version of the database and the generated Work file.

xyz~bns.abc

Stores the Criticality List.

xyz~pdt.abc

Records parameter values set by the user in program ADV.

xyz~crh.abc

In a standard simulation run this file would record the Critical Resource Identification methods used and the Critical Resource selected. When the Learning System is being updated it also records the Critical Resource chosen by each of the identification methods.

xyz~lsy.abc

When the Learning System is updated the three learning system percentile values are stored in this file for all combinations of Critical Resource Identification method and Sequencing Heuristic.

xyz~que.abc

Records the contents of the queues and the work centre. The ten digit record number from the first two fields in the Sequencing Database is used to indicate a job.

xyz~rsg.abc

Performance results are either global, job or resource based, this file records global results such as number of jobs graduated and total number of

resources used.

xyz~rsj.abc

For every job which completes processing in the period of interest a record is created in this file.

xyz~rsm.abc

For every UT defined to the software a record exists in this file. It is used to record usage statistics and the length of time that it spent in various states during the period simulated.

APPENDIX C

VERIFICATION OF THE MSA

SOFTWARE SUITE

C.1 Introduction

Verification of a computer program is the process of confirming the correct function of program components.

Three consecutive methods were used in the verification process of the MSA software developed in this work. Firstly, the code was created in a structured modular fashion in order that sections could be removed and extensive tests performed in isolation.

Tests at this level consisted of the insertion of extensive numbers of additional print statements which detailed the operation of the software. These statements were then removed and the modular segments returned to the main code. With the ability to determine the accuracy of routines on the spot extensive recording of results was not employed.

The second approach was to trace the actions of the software with deterministic test cases using the built in reporting functions of the software to check results matched expectations.

One test case was used at this stage, test case VM1. With one machine, five different products, deterministic processing times and a range of loading levels the full trace actions was produced and manually confirmed. The full test organisation described is more fully later in this Appendix.

Having confirmed numeric accuracy of the software, the software was verified analytically by comparing results to two known theorems using SPT and EDD dispatching rules.

C.2 Verification Models

Five single resource models, detailed in Table C.1, were created to verify the software . The processing times of the products are all set at 20 minutes with the variability of the model being introduced by sampling the demand from Normal distributions.

Model Name	Processing Time	Quantity Demanded	Approx Loading
VM1	20	deterministic	50-150%
VM2	20	12	n/a
VM3	N(20,4)	N(18,3.6)	n/a
VM4	N(20,4)	N(30,6)	n/a
VM5	N(20,4)	N(48,9.6)	n/a

Table C.1: Single Resource Verification Models

Model VM1 was run for five 40 hour weeks with five jobs introduced to the facility at the beginning of each week, producing a loading of 50, 125, 125, 5 and 0 percent respectively in each week. This resulted in situations of underload and overload.

For analytical verification of the software, models VM2 through VM5 emulate static type models often mentioned in the literature. The static type model is achieved by providing sufficient time in the first week for all processing to be completed in that week.

The settings of the simulation control parameters used in all the VM models are listed in Table C.2 below:

GROUP	PARAMETERS	VALUE
MASTER	Batch Size Method:	PB = TB
	Set up:	none
	Critical Resource Id:	manual
	Simulate x Periods:	1 and 5
LOGIC	Loading Rule:	parallel
	Dispatching Rule:	all four
	Unload Rule	to output
OUTPUT	Lead Time Offset:	0
	Tardiness Tolerance:	0
	Clear Criticality List:	no

Table C.2: Control Parameter Settings

C.3 Simulation Model Trace

Using the trace technique the dispatching rules performance results and operation of the Criticality List are verified.

The Tables containing the data files of model VM1 are included at the end of this appendix: Table C.5 is the Jobs Pending File, Table C.6 is the Master Sequencing Database. Several sections of "Run Files" are included at the end of this Appendix; Run Files are chronologically ordered lists of simulation actions and the times at which they occurred.

C.3.1 Dispatching Rules

Random

Table C.7 is a section of the Run File for the VM1 model under the random dispatching rule. Lines 1 through 5 are the launch actions of the simulation at the beginning of period 1. Launching is simply the moving of jobs from the Job Pending List into the input queues of the relevant resources, resource number one in this single machine case. Comparing the order of the jobs in Jobs Pending File (Table C.5) to the order the jobs are launched (lines 1-5 of Table C.7) it is clear that they are randomly selected. Lines 28 to 32 are the launch actions of jobs for period 2, and again referring to the Jobs Pending file (Table C.5) it is possible to determine that the jobs are again randomly launched.

FCFS

Table C.8 displays the run file for the VM1 model using the FCFS dispatching rule to sequence jobs through the one resource. Lines 1 to 5 is the launching at the beginning of the period and with reference to the Job Pending File (Table C.5) it is clear they are in FCFS order.

When all the jobs have been launched they are selected from the input queue in the dispatching rule order, demonstrated by lines 6, 10 14, 18 and 22 in Table C.8. Line 14 for example of Table C.8:

"W001 D02 H03 M40 1 PQ > Re Load PR3 PR3 3"

indicates: move product PR3 from the part input queue of resource 1 to the work centre to begin processing. This simulation action occurred at 40 minutes into hour 3 of day 2 in week 1.

SPT

Table C.9 is a section of the Run File for the VM1 model under the SPT dispatching rule. The variability in the processing time for use with the SPT rule comes from the quantity demanded since the processing time for all the jobs on the single resource is twenty minutes (Column 8 of the Sequencing Database (Table C.6). The processing time used in the SPT rule is the product of the processing time and the quantity demanded. Column 4, headed "Quan", of the job file (Table C.5) lists the quantities required of the individual products. The order the jobs are launched (first five lines of the Run File Table C.9) is job numbers 3, 2, 1, 4 then 5. Jobs numbers 1 and 4 are required in the same quantities, that is, 18 units which results in the processing times of these two jobs being 18 * 20 = 360 minutes. To break the tie the FCFS rule is used, which explains why product PR1 (job number: 1) is chosen for launching before product PR4 (job number: 4). With reference to lines 6, 10, 14, 18 and 22 of the Run File it is possible to determine that the jobs are loaded onto the resource from the input queue in the same SPT order.

EDD

A section of the Run File for VM1 under the EDD dispatching rule is included in Table C.10. The due dates of the jobs are included in the Jobs Pending File, columns 6 to 9 and with reference to line 1 through 5 of Table C.10 the jobs where launched in job number order 2, 4, 3, 1 then 5.. Minutes were included in the calculations of the due date to generate accurate specific results and would not be used in an actual manufacturing environment. Jobs 2 and 4 have the same due date and again the FCFS rule was successfully employed to decide between them.

C.3.2 Performance Results

All the examples used to illustrate the verification of the performance results are from running the VM1 model for five periods using the SPT dispatching rule.

Flow Time (1)

Flow Time (1) is a measure of the time a job is on the shop floor, from release until processing complete. Since the input queue never became full it was possible to release all jobs at the beginning of each week hence the flow time (1) measure begins counting from time t = 0. Table C.11 is the Individual Graduated Data forms output by the MSA software providing information on completed jobs.

To verify the flowtime measure compare the "Launched" and "Completed" times on the individual data forms, the difference yields the value of the flow time (1) measure. For illustration purposes consider jobs number 3:

From the Job Data form Table C.11:

	week	day	hour	minute
Launch Time was:	001	01	00	00
Completion Time:	001	01	05	20
Total Flow Time:	000	00	05	20

= 320 minutes

The completion time of job number 3 may be verified by referencing Table C.9 (the Run File) line 9.

The Mean Flow Time (1) measure is included in the EFFECTIVENESS section

on the Summary Data form Table C.11. Total flowtime may be verified by checking the job flowtime values on the Tables with the Graduated Job Data forms, Table C.11. Dividing this figure by 5 (the number of jobs completed) gives the Mean Flow Time (1).

Early Time

This value is checked in a similar manner as the value of flow time. The due dates and completion times are contained in the Graduated Job Data Forms (Table C.11). The early times of jobs can be validated by subtracting the value of "Due Date" from "Completed". Considering work days are eight hours long then a value of 1780 for the earliness of job PR3 is correct.

By summing all the earliness times from the Graduated Job Data Forms one arrives at a figure of 5380 for the total earliness time. The Mean Early Time of 1076 for period 1 is correct and is included in the CUSTOMER SATISFACTION section of the Summary Data Form, Table C.12.

Manufacturing Error

The Manufacturing Error is the difference between the actual flowtime (2) measure and the theoretical flow time. Actual flow time (2) begins counting when the product is loaded onto the first resource of its route through the facility. Since there is only one resource the Manufacturing Error is necessarily zero if the software is working correctly. The product specific theoretical and actual flowtimes and the manufacturing error are contained on the Graduated Job Data forms, Table C.11. Referring to column 3 of the Job Pending File (Table C.5) and using a processing time of 20 minutes it is possible to determine that the values for the flowtimes and error are correct.

Deliver Error

The delivery error is the difference between the time a job is completed and the due date. Since all jobs were early it is equal to the earliness value demonstrated above to be correct. A separate piece of code was used to verify that the delivery error calculations are correct. The value for the total delivery error is included in the CUSTOMER SATISFACTION section of the Summary Data form, Table C.12.

Utilisation Measures

The total time available per week was 40 hours or 2400 minutes, the total flowtime (2) was 1800 minutes indicating that the working and idle statistics of 75 and 25% respectively included in the EFFICIENCY section of the Summary Data from (Table C.12) are therefore correct.

The above demonstrated that the primary measures used in the analysis of the methodology are being calculated correctly. If the performance values are being correctly generated then the Critical Resource Identificaton methods which use these values must also be working correctly.

C.3.3 Operation of the Criticality List

In the single resource model all jobs are going to appear on the Criticality List. If a job was not loaded onto a resource by the end of the week in which it was launched then it has to compete for processing time with the new jobs launched in the subsequent week. Initial trials indicated that when the SPT rule was being used this could result in some jobs never being completed because new jobs move to the front of the queue. To avoid this the Criticality List is sectioned such that no job launched in week 4 for example, may be placed ahead of any job launched in week 3. To validate the operation of this

methodology consider Table C.13 and the Run File for period 4 of model VM1 when run using the SPT dispatching rule. Lines 1 to 5 indicate that jobs numbered 16 to 20 were launched at the beginning of the period, however, when the resource becomes idle (line 6) it is job number 15 which is loaded (line 8). The processing times of job numbers 16 to 20 is 20 minutes each and the processing time of job number 15 is 600 minutes but is loaded first. This indicates the correct operation of the partitioning of the Criticality list since job number 16 was originally launched to the facility in period 3.

C.3.4 Generation of Random Numbers

The random number generator used was the one available in the PowerBASIC software package, p 244 of the PowerBASIC Reference Guide (1990).

Generating Normally distributed random numbers was achieved using the Box-Jenkins method, reference Law and Kelton (1990). To verify that the number generated were in fact normally distributed the Univariate procedure in the SAS statistical package was used. The result was positive hence the procedure was implemented in the MSA software.

C.4 Analytical Verification

For the static single machine shop two theorems have been proven:

1. To minimise the mean flow time, the shortest processing time dispatching rule is optimal.
2. For minimising the maximum job lateness the earliest due date sequencing rule is optimal.

The proofs of these theorems may be found in K.R. Baker's book

"Introduction to Sequencing and Scheduling" (1974).

Each of the models VM2 through VM5 was ran for each of the four dispatching rules, Table C.3 is a summary of the flow time results:

Model Name	Dispatching Rules RANDOM	FCFS	SPT	EDD
VM2	2520	2520	2520	2520
VM3	3476	3503	2997	3319
VM4	68011	69211	61146	67863
VM5	117826	118571	99504	119741

Table C.3: Average Flow Time In Single Resource Model

In agreement with the first theory above the SPT rule produced the lowest average flow time. In model VM2 each job had the same processing time at the one resource. As a consequence of this the results for flowtime are equal irrespective of the dispatching rule used therefore not reported in the Table.

Table C.4 displays the values for maximum job lateness for models VM3 through VM4.

Model Name	Dispatching Rules Random	FCFS	SPT	EDD
VM3	333	555	414	304
VM4	6939	7019	6919	6759
VM5	15915	16015	15875	15724

Table C.4: Maximum Job Tardiness In Single Resource Model

Job tardiness is a measure of positive lateness and no job in model VM2 was late therefore no results are reported for this model. The results in Table C.4 agree with the theory that the EDD rule minimises the maximum job lateness which in turn verifies the operation of the software.

"000 PERIOD										"
"	Prod	Sch'd	Quan	JobNo	Week	Day	Hour	Min	Prio'y	TB
"1	PR1	No	18	1	001	05	04	00	1	18
"2	PR2	No	17	2	001	05	02	00	1	17
"3	PR3	No	16	3	001	05	03	00	1	16
"4	PR4	No	18	4	001	05	02	00	1	18
"5	PR5	No	21	5	001	05	05	00	1	21
"6	PR1	No	29	6	002	05	03	00	1	24
"7	PR2	No	25	7	002	05	02	00	1	20
"8	PR3	No	25	8	002	05	01	00	1	20
"9	PR4	No	29	9	002	05	00	00	1	24
"10	PR5	No	33	10	002	05	01	00	1	28
"11	PR1	No	25	11	003	05	01	00	1	20
"12	PR2	No	28	12	003	05	02	00	1	28
"13	PR3	No	28	13	003	05	03	00	1	23
"14	PR4	No	29	14	003	05	04	00	1	24
"15	PR5	No	30	15	003	05	05	00	1	25
"16	PR1	No	1	16	004	05	06	00	1	1
"17	PR2	No	1	17	004	05	03	00	1	1
"18	PR3	No	1	18	004	05	02	00	1	1
"19	PR4	No	1	19	004	05	00	00	1	1
"20	PR5	No	1	20	004	05	01	00	1	1

Table C.5 : Jobs Pending File For Model VM2

```
"1   2     3    4   5    6    7    8   9   10   11  12   13   14   15   16   17   18   19
20   21
"1    FinPr PR1  1   1    PR1  0    20  0   Fin  PR1 0    0    #    0    0    0    #    1
1    1"
"2    FinPr PR2  1   1    PR2  0    20  0   Fin  PR2 0    0    #    0    0    0    #    1
1    1"
"3    FinPr PR3  1   1    PR3  0    20  0   Fin  PR3 0    0    #    0    0    0    #    1
1    1"
"4    FinPr PR4  1   1    PR4  0    20  0   Fin  PR4 0    0    #    0    0    0    #    1
1    1"
"5    FinPr PR5  1   1    PR5  0    20  0   Fin  PR5 0    0    #    0    0    0    #    1
1    1"
```

Key to column numbers:

1. record number
2. succeeding record
3. product name
4. gateway part if equal to 1
5. work centre number
6. component name
7. start time for holding
8. processing time
9. finish time for holding
10. number of next work centre
11. succeeding part name
12. quantity still to be produced
13. quantity on-hand
14. job number
15. process batch remaining
16. transfer batch
17. transfer quantity
18. time to finish transfer batch
19. succeeding work centre
20. set up time one
21. set up time two

Table C.6 : Master Sequencing Database for Model VM2

```
"001 PERIOD                                                    "
"     Time            Res       Action      Part Prod  Job No"
1     "W001 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  2     "
2     "W001 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  5     "
3     "W001 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  1     "
4     "W001 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  3     "
5     "W001 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  4     "
6     "W001 D01 H00 M00 1     PQ > Re  Load   PR2  PR2  2     "
7     "W001 D01 H00 M00 1     Begin Operating PR2  PR2  2     "
8     "W001 D01 H05 M40 1     Re > OQ  Unload PR2  PR2  2     "
9     "W001 D01 H05 M40 1     Move to Stock   PR2  PR2  2     "
10    "W001 D01 H05 M40 1     PQ > Re  Load   PR5  PR5  5     "
11    "W001 D01 H05 M40 1     Begin Operating PR5  PR5  5     "
12    "W001 D02 H04 M40 1     Re > OQ  Unload PR5  PR5  5     "
13    "W001 D02 H04 M40 1     Move to Stock   PR5  PR5  5     "
14    "W001 D02 H04 M40 1     PQ > Re  Load   PR1  PR1  1     "
15    "W001 D02 H04 M40 1     Begin Operating PR1  PR1  1     "
16    "W001 D03 H02 M40 1     Re > OQ  Unload PR1  PR1  1     "
17    "W001 D03 H02 M40 1     Move to Stock   PR1  PR1  1     "
18    "W001 D03 H02 M40 1     PQ > Re  Load   PR3  PR3  3     "
19    "W001 D03 H02 M40 1     Begin Operating PR3  PR3  3     "
20    "W001 D04 H00 M00 1     Re > OQ  Unload PR3  PR3  3     "
21    "W001 D04 H00 M00 1     Move to Stock   PR3  PR3  3     "
22    "W001 D04 H00 M00 1     PQ > Re  Load   PR4  PR4  4     "
23    "W001 D04 H00 M00 1     Begin Operating PR4  PR4  4     "
24    "W001 D04 H06 M00 1     Re > OQ  Unload PR4  PR4  4     "
25    "W001 D04 H06 M00 1     Move to Stock   PR4  PR4  4     "
26    "002 PERIOD                                              "
27    "Time             Res     Action      Part Prod  JobNo"
28    "W002 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  6     "
29    "W002 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  9     "
30    "W002 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  7     "
31    "W002 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  10    "
32    "W002 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  8     "
33    "W002 D01 H00 M00 1     PQ > Re  Load   PR1  PR1  6     "
34    "W002 D01 H00 M00 1     Begin Operating PR1  PR1  6     "
35    "W002 D02 H01 M40 1     Re > OQ  Unload PR1  PR1  6     "
36    "W002 D02 H01 M40 1     Move to Stock   PR1  PR1  6     "
37    "W002 D02 H01 M40 1     PQ > Re  Load   PR4  PR4  9     "
38    "W002 D02 H01 M40 1     Begin Operating PR4  PR4  9     "
39    "W002 D03 H03 M20 1     Re > OQ  Unload PR4  PR4  9     "
40    "W002 D03 H03 M20 1     Move to Stock   PR4  PR4  9     "
```

Table C.7: A Section of the Run File For Model VM1 Using The Random Dispatching Rule

```
"001 PERIOD                                                    "
"      Time            Res      Action      Part Prod  JobNo"
1     "W001 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  1     "
2     "W001 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  2     "
3     "W001 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  3     "
4     "W001 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  4     "
5     "W001 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  5     "
6     "W001 D01 H00 M00 1     PQ > Re  Load   PR1  PR1  1     "
7     "W001 D01 H00 M00 1     Begin Operating PR1  PR1  1     "
8     "W001 D01 H06 M00 1     Re > OQ  Unload PR1  PR1  1     "
9     "W001 D01 H06 M00 1     Move to Stock   PR1  PR1  1     "
10    "W001 D01 H06 M00 1     PQ > Re  Load   PR2  PR2  2     "
11    "W001 D01 H06 M00 1     Begin Operating PR2  PR2  2     "
12    "W001 D02 H03 M40 1     Re > OQ  Unload PR2  PR2  2     "
13    "W001 D02 H03 M40 1     Move to Stock   PR2  PR2  2     "
14    "W001 D02 H03 M40 1     PQ > Re  Load   PR3  PR3  3     "
15    "W001 D02 H03 M40 1     Begin Operating PR3  PR3  3     "
16    "W001 D03 H01 M00 1     Re > OQ  Unload PR3  PR3  3     "
17    "W001 D03 H01 M00 1     Move to Stock   PR3  PR3  3     "
18    "W001 D03 H01 M00 1     PQ > Re  Load   PR4  PR4  4     "
19    "W001 D03 H01 M00 1     Begin Operating PR4  PR4  4     "
20    "W001 D03 H07 M00 1     Re > OQ  Unload PR4  PR4  4     "
21    "W001 D03 H07 M00 1     Move to Stock   PR4  PR4  4     "
22    "W001 D03 H07 M00 1     PQ > Re  Load   PR5  PR5  5     "
23    "W001 D03 H07 M00 1     Begin Operating PR5  PR5  5     "
24    "W001 D04 H06 M00 1     Re > OQ  Unload PR5  PR5  5     "
25    "W001 D04 H06 M00 1     Move to Stock   PR5  PR5  5     "
26    "002 PERIOD                                              "
27    "Time            Res      Action      Part Prod  JobNo"
28    "W002 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  6     "
29    "W002 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  7     "
30    "W002 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  8     "
31    "W002 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  9     "
32    "W002 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  10    "
33    "W002 D01 H00 M00 1     PQ > Re  Load   PR1  PR1  6     "
34    "W002 D01 H00 M00 1     Begin Operating PR1  PR1  6     "
35    "W002 D02 H01 M40 1     Re > OQ  Unload PR1  PR1  6     "
36    "W002 D02 H01 M40 1     Move to Stock   PR1  PR1  6     "
37    "W002 D02 H01 M40 1     PQ > Re  Load   PR2  PR2  7     "
38    "W002 D02 H01 M40 1     Begin Operating PR2  PR2  7     "
39    "W002 D03 H02 M00 1     Re > OQ  Unload PR2  PR2  7     "
40    "W002 D03 H02 M00 1     Move to Stock   PR2  PR2  7     "
```

Table C.8: A Section of the Run File For Model VM1 Using The FCFS Dispatching Rule

"001 PERIOD											"
"	Time			Res		Action		Part	Prod	Job	No"
1	"W001	D01	H00	M00	1	Jo > PQ	Launch	PR3	PR3	3	"
2	"W001	D01	H00	M00	1	Jo > PQ	Launch	PR2	PR2	2	"
3	"W001	D01	H00	M00	1	Jo > PQ	Launch	PR1	PR1	1	"
4	"W001	D01	H00	M00	1	Jo > PQ	Launch	PR4	PR4	4	"
5	"W001	D01	H00	M00	1	Jo > PQ	Launch	PR5	PR5	5	"
6	"W001	D01	H00	M00	1	PQ > Re	Load	PR3	PR3	3	"
7	"W001	D01	H00	M00	1	Begin Operating		PR3	PR3	3	"
8	"W001	D01	H05	M20	1	Re > OQ	Unload	PR3	PR3	3	"
9	"W001	D01	H05	M20	1	Move to Stock		PR3	PR3	3	"
10	"W001	D01	H05	M20	1	PQ > Re	Load	PR2	PR2	2	"
11	"W001	D01	H05	M20	1	Begin Operating		PR2	PR2	2	"
12	"W001	D02	H03	M00	1	Re > OQ	Unload	PR2	PR2	2	"
13	"W001	D02	H03	M00	1	Move to Stock		PR2	PR2	2	"
14	"W001	D02	H03	M00	1	PQ > Re	Load	PR1	PR1	1	"
15	"W001	D02	H03	M00	1	Begin Operating		PR1	PR1	1	"
16	"W001	D03	H01	M00	1	Re > OQ	Unload	PR1	PR1	1	"
17	"W001	D03	H01	M00	1	Move to Stock		PR1	PR1	1	"
18	"W001	D03	H01	M00	1	PQ > Re	Load	PR4	PR4	4	"
19	"W001	D03	H01	M00	1	Begin Operating		PR4	PR4	4	"
20	"W001	D03	H07	M00	1	Re > OQ	Unload	PR4	PR4	4	"
21	"W001	D03	H07	M00	1	Move to Stock		PR4	PR4	4	"
22	"W001	D03	H07	M00	1	PQ > Re	Load	PR5	PR5	5	"
23	"W001	D03	H07	M00	1	Begin Operating		PR5	PR5	5	"
24	"W001	D04	H06	M00	1	Re > OQ	Unload	PR5	PR5	5	"
25	"W001	D04	H06	M00	1	Move to Stock		PR5	PR5	5	"
26	"002 PERIOD										"
27	"Time			Res		Action		Part	Prod	JobNo"	
28	"W002	D01	H00	M00	1	Jo > PQ	Launch	PR2	PR2	7	"
29	"W002	D01	H00	M00	1	Jo > PQ	Launch	PR3	PR3	8	"
30	"W002	D01	H00	M00	1	Jo > PQ	Launch	PR1	PR1	6	"
31	"W002	D01	H00	M00	1	Jo > PQ	Launch	PR4	PR4	9	"
32	"W002	D01	H00	M00	1	Jo > PQ	Launch	PR5	PR5	10	"
33	"W002	D01	H00	M00	1	PQ > Re	Load	PR2	PR2	7	"
34	"W002	D01	H00	M00	1	Begin Operating		PR2	PR2	7	"
35	"W002	D02	H00	M20	1	Re > OQ	Unload	PR2	PR2	7	"
36	"W002	D02	H00	M20	1	Move to Stock		PR2	PR2	7	"
37	"W002	D02	H00	M20	1	PQ > Re	Load	PR3	PR3	8	"
38	"W002	D02	H00	M20	1	Begin Operating		PR3	PR3	8	"
39	"W002	D03	H00	M40	1	Re > OQ	Unload	PR3	PR3	8	"
40	"W002	D03	H00	M40	1	Move to Stock		PR3	PR3	8	"

Table C.9: A Section of the Run File For Model VM1 Using The SPT Dispatching Rule

"001 PERIOD										"
"	Time			Res	Action		Part	Prod	JobNo	"
1	"W001	D01	H00	M00 1	Jo > PQ	Launch	PR2	PR2	2	"
2	"W001	D01	H00	M00 1	Jo > PQ	Launch	PR4	PR4	4	"
3	"W001	D01	H00	M00 1	Jo > PQ	Launch	PR3	PR3	3	"
4	"W001	D01	H00	M00 1	Jo > PQ	Launch	PR1	PR1	1	"
5	"W001	D01	H00	M00 1	Jo > PQ	Launch	PR5	PR5	5	"
6	"W001	D01	H00	M00 1	PQ > Re	Load	PR2	PR2	2	"
7	"W001	D01	H00	M00 1	Begin Operating		PR2	PR2	2	"
8	"W001	D01	H05	M40 1	Re > OQ	Unload	PR2	PR2	2	"
9	"W001	D01	H05	M40 1	Move to Stock		PR2	PR2	2	"
10	"W001	D01	H05	M40 1	PQ > Re	Load	PR4	PR4	4	"
11	"W001	D01	H05	M40 1	Begin Operating		PR4	PR4	4	"
12	"W001	D02	H03	M40 1	Re > OQ	Unload	PR4	PR4	4	"
13	"W001	D02	H03	M40 1	Move to Stock		PR4	PR4	4	"
14	"W001	D02	H03	M40 1	PQ > Re	Load	PR3	PR3	3	"
15	"W001	D02	H03	M40 1	Begin Operating		PR3	PR3	3	"
16	"W001	D03	H01	M00 1	Re > OQ	Unload	PR3	PR3	3	"
17	"W001	D03	H01	M00 1	Move to Stock		PR3	PR3	3	"
18	"W001	D03	H01	M00 1	PQ > Re	Load	PR1	PR1	1	"
19	"W001	D03	H01	M00 1	Begin Operating		PR1	PR1	1	"
20	"W001	D03	H07	M00 1	Re > OQ	Unload	PR1	PR1	1	"
21	"W001	D03	H07	M00 1	Move to Stock		PR1	PR1	1	"
22	"W001	D03	H07	M00 1	PQ > Re	Load	PR5	PR5	5	"
23	"W001	D03	H07	M00 1	Begin Operating		PR5	PR5	5	"
24	"W001	D04	H06	M00 1	Re > OQ	Unload	PR5	PR5	5	"
25	"W001	D04	H06	M00 1	Move to Stock		PR5	PR5	5	"
26	"002 PERIOD									"
27	"Time			Res	Action		Part	Prod	JobNo	"
28	"W002	D01	H00	M00 1	Jo > PQ	Launch	PR4	PR4	9	"
29	"W002	D01	H00	M00 1	Jo > PQ	Launch	PR3	PR3	8	"
30	"W002	D01	H00	M00 1	Jo > PQ	Launch	PR5	PR5	10	"
31	"W002	D01	H00	M00 1	Jo > PQ	Launch	PR2	PR2	7	"
32	"W002	D01	H00	M00 1	Jo > PQ	Launch	PR1	PR1	6	"
33	"W002	D01	H00	M00 1	PQ > Re	Load	PR4	PR4	9	"
34	"W002	D01	H00	M00 1	Begin Operating		PR4	PR4	9	"
35	"W002	D02	H01	M40 1	Re > OQ	Unload	PR4	PR4	9	"
36	"W002	D02	H01	M40 1	Move to Stock		PR4	PR4	9	"
37	"W002	D02	H01	M40 1	PQ > Re	Load	PR3	PR3	8	"
38	"W002	D02	H01	M40 1	Begin Operating		PR3	PR3	8	"
39	"W002	D03	H02	M00 1	Re > OQ	Unload	PR3	PR3	8	"
40	"W002	D03	H02	M00 1	Move to Stock		PR3	PR3	8	"

Table C.10: A Section of the Run File For Model VM1 Using The EDD Dispatching Rule

```
        I N D I V I D U A L   G R A D U A T E D   J O B   D A T A

    Project: VM1                                              15:02:27
    Version: 003          Product name: PR3                   03-06-1992
    Period:  001          Job number:   3

                                                 Wk   Dy   Hr   Min
    Quantity required ..... 16     Launched .... 001  01   00   00
    Transfer batch size ... 16     Completed ... 001  01   05   20
    Ave Set Up Time (min) . 0      Due Date .... 001  05   03   00

    Actual flowtime ....... 320    Tardiness ... early by 1780 minutes
    Theoretical flowtime .. 320
    Manufacturing error ... 0

MSA Ver 3.0                                                   F-JOB.FRM

        I N D I V I D U A L   G R A D U A T E D   J O B   D A T A

    Project: VM1                                              15:02:27
    Version: 003          Product name: PR2                   03-06-1992
    Period:  001          Job number:   2

                                                 Wk   Dy   Hr   Min
    Quantity required ..... 17     Launched .... 001  01   05   20
    Transfer batch size ... 17     Completed ... 001  02   03   00
    Ave Set Up Time (min) . 0      Due Date .... 001  05   02   00

    Actual flowtime ....... 340    Tardiness ... early by 1380 minutes
    Theoretical flowtime .. 340
    Manufacturing error ... 0

MSA Ver 3.0                                                   F-JOB.FRM

        I N D I V I D U A L   G R A D U A T E D   J O B   D A T A

    Project: VM1                                              15:02:27
    Version: 003          Product name: PR1                   03-06-1992
    Period:  001          Job number:   1

                                                 Wk   Dy   Hr   Min
    Quantity required ..... 18     Launched .... 001  02   03   00
    Transfer batch size ... 18     Completed ... 001  03   01   00
    Ave Set Up Time (min) . 0      Due Date .... 001  05   04   00

    Actual flowtime ....... 360    Tardiness ... early by 1140 minutes
    Theoretical flowtime .. 360
    Manufacturing error ... 0

MSA Ver 3.0                                                   F-JOB.FRM
```

Table C.11: Graduated Job Data For Model VM1 Under The SPT Dispatching Rule

```
        I N D I V I D U A L   G R A D U A T E D   J O B   D A T A

    Project: VM1                                              15:02:27
    Version: 003          Product name: PR4                   03-06-1992
    Period:  001          Job number:   4

                                                 Wk    Dy    Hr    Min
    Quantity required ..... 18     Launched .... 001   03    01    00
    Transfer batch size ... 18     Completed ... 001   03    07    00
    Ave Set Up Time (min) . 0      Due Date .... 001   05    02    00

    Actual flowtime ....... 360    Tardiness ... early by 660 minutes
    Theoretical flowtime .. 360
    Manufacturing error ... 0

MSA Ver 3.0                                                   F-JOB.FRM


        I N D I V I D U A L   G R A D U A T E D   J O B   D A T A

    Project: VM1                                              15:02:28
    Version: 003          Product name: PR5                   03-06-1992
    Period:  001          Job number:   5

                                                 Wk    Dy    Hr    Min
    Quantity required ..... 21     Launched .... 001   03    07    00
    Transfer batch size ... 21     Completed ... 001   04    06    00
    Ave Set Up Time (min) . 0      Due Date .... 001   05    05    00

    Actual flowtime ....... 420    Tardiness ... early by 420 minutes
    Theoretical flowtime .. 420
    Manufacturing error ... 0

MSA Ver 3.0                                                   F-JOB.FRM
```

Table C.11 (continued)

S U M M A R Y D A T A

Project: VM1 15:31:37
Version: 003 03-06-1992
Period: 001

CUSTOMER SATISFACTION

No. of jobs completed . 5	Total early time ..... -5380
Early jobs 5	Total late time 0
Late jobs 0	Average early ..... -1076
Jobs on-time 0	Average late 0
	Total delivery error . 5380

Frequency of tardy jobs ..... 0 %

EFFECTIVENESS

Total theoretical flow time .. 1800	Mean flow time 0
Total actual flow time 1800	Maximum flow time 420
Total manufacturing error 0	Minimum flow time 320
Total flow time (1) 5180	Mean Flow Time (1) 1036

EFFICIENCY

Number of resources ... 1
Total time available:
Facility ..... 40 hours
All resources .... 40 hours

	Set Up	Wkg	Idle	Blkd	Wait	Util 1	Util 2
Percent:	0	75	25	0	0	75	75

WIP INVENTORY

PROCESSING TIME DATA:
Time average processing time 0

TIME AVERAGE QUEUE LENGTH:
Time average queue length 0

OVERALL SUMMARY

Measures:	Percentiles	Weighting:
Customer Performance	-198.	0.3333
Effectiveness	100	0.3333
Efficiency	75	0.3333
Overall Performance	-7.67	

MSA Ver 3.0 F-SUM.FRM

Table C.12: Summary Data Form From Model VM1 Under The SPT Dispatching Rule

```
                                "004 PERIOD"
   "Time              Res       Action      Part Prod  Job No"
1      "W004 D01 H00 M00 1     Jo > PQ  Launch PR1  PR1  16   "
2      "W004 D01 H00 M00 1     Jo > PQ  Launch PR2  PR2  17   "
3      "W004 D01 H00 M00 1     Jo > PQ  Launch PR3  PR3  18   "
4      "W004 D01 H00 M00 1     Jo > PQ  Launch PR4  PR4  19   "
5      "W004 D01 H00 M00 1     Jo > PQ  Launch PR5  PR5  20   "
6      "W004 D01 H03 M40 1     Re > OQ  Unload PR4  PR4  14   "
7      "W004 D01 H03 M40 1     Move to Stock   PR4  PR4  14   "
8      "W004 D01 H03 M40 1     PQ > Re  Load   PR5  PR5  15   "
9      "W004 D01 H03 M40 1     Begin Operating PR5  PR5  15   "
10     "W004 D02 H05 M40 1     Re > OQ  Unload PR5  PR5  15   "
11     "W004 D02 H05 M40 1     Move to Stock   PR5  PR5  15   "
12     "W004 D02 H05 M40 1     PQ > Re  Load   PR1  PR1  16   "
13     "W004 D02 H05 M40 1     Begin Operating PR1  PR1  16   "
14     "W004 D02 H06 M00 1     Re > OQ  Unload PR1  PR1  16   "
15     "W004 D02 H06 M00 1     Move to Stock   PR1  PR1  16   "
16     "W004 D02 H06 M00 1     PQ > Re  Load   PR2  PR2  17   "
17     "W004 D02 H06 M00 1     Begin Operating PR2  PR2  17   "
18     "W004 D02 H06 M20 1     Re > OQ  Unload PR2  PR2  17   "
19     "W004 D02 H06 M20 1     Move to Stock   PR2  PR2  17   "
20     "W004 D02 H06 M20 1     PQ > Re  Load   PR3  PR3  18   "
21     "W004 D02 H06 M20 1     Begin Operating PR3  PR3  18   "
22     "W004 D02 H06 M40 1     Re > OQ  Unload PR3  PR3  18   "
23     "W004 D02 H06 M40 1     Move to Stock   PR3  PR3  18   "
24     "W004 D02 H06 M40 1     PQ > Re  Load   PR4  PR4  19   "
25     "W004 D02 H06 M40 1     Begin Operating PR4  PR4  19   "
26     "W004 D02 H07 M00 1     Re > OQ  Unload PR4  PR4  19   "
27     "W004 D02 H07 M00 1     Move to Stock   PR4  PR4  19   "
28     "W004 D02 H07 M00 1     PQ > Re  Load   PR5  PR5  20   "
29     "W004 D02 H07 M00 1     Begin Operating PR5  PR5  20   "
30     "W004 D02 H07 M20 1     Re > OQ  Unload PR5  PR5  20   "
31     "W004 D02 H07 M20 1     Move to Stock   PR5  PR5  20   "
```

Table C.13: A Section of the Run File For Model VM1 Using The SPT Dispatching Rule

APPENDIX D

TEST CASE RESULTS

Tables

	Ranking from random			
	Loading		Criteria	
LM1	-14	(2)	-13	(3)
LM2	-9	(1)	-8	(1)
QM1	-42	(3)	-37	(2)

	Percent Improvement From Random			
	Loading		Criteria	
LM1	-19	(3)	-20	(3)
LM2	+112	(1)	+112	(1)
QM1	+36	(2)	+31	(2)

Table D.1: Effectiveness of Critical

Resource Models Against Random

	Ranking from random			
	Loading		Criteria	
FCFS	+25	(2)	+25	(3)
SPT	+71	(1)	+66	(1)
EDD	+32	(3)	+32	(2)

	Percent Improvement From Random			
	Loading		Criteria	
FCFS	+34	(3)	+49	(2)
SPT	+80	(1)	+79	(1)
EDD	+40	(2)	+40	(3)

Table D.2: Effectiveness of Dispatching Rule

Against Random

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	20	3	20	3	19	3	14	2
FCFS	15	2	21	4	13	2	21	3
SPT	7	1	12	2	9	1	7	1
EDD	15	2	11	1	23	4	24	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	14	3	16	2	16	2	10	1
LM1	12	2	18	3	9	1	13	2
LM2	10	1	14	1	18	3	18	3
QM1	21	4	18	4	24	4	25	4

	100%		200%		300%		Totals	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	24	4	25	4	22	3	144	4
FCFS	16	3	19	3	14	2=	119	3
SPT	12	1	13	2	13	1=	73	1
EDD	15	2	11	1	13	1=	112	2
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	17	3	15	1	12	1	100	1
LM1	24	4	15	1	23	4	114	3
LM2	16	2	16	2	17	2	109	2
QM1	13	1	23	3	18	3	142	4

Table D.3: Aggregation of The Ranking Values Of All The Performance Measures For Each Level Of Loading

	mean flow time(1)		mean flow time(2)		mean early		mean manuf'g error	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	24	4	22	4	17	2	24	4
FCFS	20	3	18	3	17	2	20	2
SPT	9	1	11	1	8	1	9	1
EDD	17	2	17	2	25	3	17	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	16	2	15	2	19	3	15	2=
LM1	14	1	19	3	22	4	14	1
LM2	17	3	14	1	11	1	15	2=
QM1	21	4	22	4	16	2	24	3

	mean tardy time(1)		percentage jobs tardy		mean delivery error		Totals	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	22	4	14	3	22	4	145	4
FCFS	16	3	13	2	16	2	120	3
SPT	14	2	9	1	19	3	79	1
EDD	12	1	15	4	10	1	113	2
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	10	1	15	2	11	1	101	1
LM1	16	2	17	3	12	2	114	3
LM2	17	3	13	1=	22	4	109	2
QM1	22	4	13	1=	20	3	138	4

Table D.4: Aggregation of The Ranking Values Of Each Inidividual Performance Measures At All Levels Of Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	18	2=	0	3	2	2	-3	2
R-SPT	19	1	16	2	11	1	13	1
R-EDD	18	2=	19	1	-3	3	-11	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
LM1	0	1=	-7	2	3	2	-4	2
LM2	0	1=	-1	1	0	3	-1	1
QM1	-1	2	-17	3	6	1	-6	3

	100%		200%		300%		Totals	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	4	2	6	3	7	1	34	3
R-SPT	6	1	13	1	2	3	80	1
R-EDD	5	3	9	2	3	2	40	2
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	-11	3	9	3	-9	3	19	3
R-LM2	-3	2	33	1	84	1	112	1
R-QM1	2	1	16	2	36	2	36	2

Table D.5: The Percentage Difference of Of All The Performance Measures From the Random Value Averaged For Each Level Of Loading

	mean flow time(1)		mean flow time(2)		mean early		mean manuf'g error	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	2	2=	1	3	2	2	20	1
R-SPT	4	1	4	1	15	1	7	2
R-EDD	2	2=	2	2	-17	3	3	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	-1	1	-2	2=	4	3	-1	1
R-LM2	-2	2	-1	1	129	1	-2	2
R-QM1	-3	3	-2	2=	63	2	-4	3

	mean tardy time(1)		percentage jobs tardy		mean delivery error		total	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
R-FCFS	20	2	0	3	4	3	49	2
R-SPT	33	1=	11	1	5	2	79	1
R-EDD	33	1=	8	2	9	1	40	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
R-LM1	-10	2	-7	3	-3	1	-20	3
R-LM2	-4	1	2	2	-10	2=	112	1
R-QM1	-18	3	5	1	-10	2=	31	2

Table D.6: The Percentage Difference From Random of Each Performance Measures Averaged Over All the Levels of Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	1207	4	1599	3	2100	3	3235	2
FCFS	1193	2	1601	4	2091	2	3253	3
SPT	1165	1	1553	1	2023	1	3008	1
EDD	1196	3	1594	2	2135	4	3295	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	1190	2=	1591	4	2089	3	3176	2
LM1	1188	1	1587	3	2061	1	3155	1
LM2	1190	2=	1585	2	2072	2	3217	3
QM1	1193	3	1584	1	2126	4	3243	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	6329	4	9697	4	12628	4
FCFS	6242	3	9244	3	12144	3
SPT	6089	1	9067	2	12699	2
EDD	6132	2	8738	1	11959	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	6157	3	8934	1	11497	1
LM1	6382	4	8944	2	12489	2
LM2	6152	2	9229	3	12687	3
QM1	6101	1	9640	4	12757	4

Table D.7: Average Mean Flow Time (1) in Minutes

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	1264	3	1676	3	2224	3	3527	2
FCFS	1252	2=	1680	4	2208	2	3552	3
SPT	1224	1	1632	1	2138	1	3271	1
EDD	1252	2=	1669	2	2257	4	3577	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	1246	2	1664	2	2201	3	3430	1
LM1	1248	3	1669	4	2183	1	3466	2
LM2	1245	1	1659	1	2194	2	3502	3
QM1	1253	4	1665	3	2248	4	3530	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	6984	4	10597	4	13904	3
FCFS	6858	2	10169	3	13472	2
SPT	6705	1	9907	2	14002	4
EDD	6872	3	9831	1	13447	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	6846	3	10090	3	13137	1
LM1	7175	4	10019	1	14033	4
LM2	6719	2	10066	2	13864	3
QM1	6679	1	10329	4	13793	2

Table D.8: Average Mean Flow Time (2) in Minutes

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	908	2	500	2=	157	3	11	2
FCFS	912	3=	495	3	161	2	9	3
SPT	939	1	536	1	179	1	13	1
EDD	912	3=	500	2=	129	4	4	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	918	2	504	3	160	1	10	2
LM1	916	4	506	2=	158	2	8	3=
LM2	920	1	506	2=	153	4	11	1
QM1	917	3	511	1	156	3	8	3=

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	970	3	357	2	122	3
FCFS	988	2	352	3	158	1
SPT	1088	1	468	1	145	2
EDD	819	4	311	4	107	4
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	891	3	170	4	41	3
LM1	708	4	294	3	39	4
LM2	1149	1	588	1	295	1
QM1	1118	2	436	2	157	2

Table D.9: Average Mean Early Time in Minutes

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	484	4	634	3	894	3	1787	2
FCFS	470	2	635	4	882	2	1795	3
SPT	442	1	598	1	819	1	1547	1
EDD	473	3	628	2	927	4	1821	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	467	3	629	3	882	3	1715	2
LM1	464	1	620	2=	854	1	1693	1
LM2	466	2	620	2=	873	2	1765	3
QM1	472	4	617	1	914	4	1777	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	4640	4	8008	4	10941	4
FCFS	4499	3	7448	3	10576	3
SPT	4422	2	7340	2	10000	1
EDD	4420	1	6971	1	10189	2
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	4424	2	7166	1	9912	1
LM1	4673	4	7239	2	10931	3
LM2	4487	1	7459	3	10916	2
QM1	4398	3	7902	4	10948	4

Table D.10: Mean Manufacturing Error in Minutes

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	8	2	11	3=	216	3	1377	2
FCFS	0	1=	11	3=	203	2	1386	3=
SPT	0	1=	5	2	156	1	1096	1
EDD	0	1=	3	1	220	4	1386	3=
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	0	1=	7	2	195	3	1261	1
LM1	0	1=	9	3	175	1	1294	2
LM2	0	1=	5	1	191	2	1339	3
QM1	8	2	10	4	234	4	1352	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	1028	4	4031	4	7108	3
FCFS	862	2	3480	3	6854	2
SPT	891	3	3409	2	7233	4
EDD	740	1	3129	1	6547	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	763	1	3248	1	6379	1
LM1	935	3	3375	2	7315	4
LM2	966	4	3637	3	7148	3
QM1	856	2	3790	4	6900	2

Table D.11: Mean Tardiness in Minutes

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	0	1=	7	3=	47	2=	94	2=
FCFS	0	1=	7	3=	47	2=	94	2=
SPT	0	1=	4	2	42	1	92	1
EDD	0	1=	3	1	47	2=	98	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	0	1=	5	1	46	1=	93	1
LM1	0	1=	6	2	47	2	94	2=
LM2	0	1=	7	3	48	3	94	2=
QM1	0	1=	9	4	46	1=	94	2=

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	41	1=	70	3	88	2=
FCFS	42	2	69	2=	87	1
SPT	41	1=	64	1	88	2=
EDD	44	3	69	2=	89	3
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	41	3	81	4	95	4
LM1	49	4	77	3	97	3
LM2	40	2	56	1	74	1
QM1	34	1	65	2	84	2

Table D.12: Percentage Number of Jobs Tardy

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	373	4	511	3	917	2	1388	3
FCFS	365	3	506	2	912	1=	1395	4
SPT	335	1	542	4	939	3	1110	1
EDD	350	2	503	1	912	1=	1390	2
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	355	3	511	1	918	2	1271	1
LM1	333	1	515	2	916	1	1302	2
LM2	344	2	516	3	920	3	1350	3
QM1	389	4	520	4	925	4	1359	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	1998	4	4389	4	7230	3
FCFS	1850	2	3832	2	7012	2
SPT	1979	3	3877	3	7378	4
EDD	1559	1	3440	1	6654	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	1654	2	3418	1	6420	1
LM1	1643	1	3669	2	7355	3
LM2	2115	4	4225	3=	7443	4
QM1	1974	3	4225	3=	7056	2

Table D.13: Mean Delivery Error in Minutes

for 40 - 300% Loading

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	1199	1253	911	475
2	FCFS	1186	1244	921	463
3	SPT	1176	1234	930	452
4	EDD	1200	1254	910	477
	LM1				
5	random	1203	1259	905	479
6	FCFS	1195	1253	911	471
7	SPT	1162	1233	931	438
8	EDD	1192	1247	917	468
	LM2				
9	random	1201	1258	906	478
10	FCFS	1207	1264	900	483
11	SPT	1155	1203	961	431
12	EDD	1198	1253	911	474
	QM1				
13	random	1227	1286	912	503
14	FCFS	1184	1247	914	463
15	SPT	1168	1227	934	447
16	EDD	1196	1253	909	475

Table D.14: Measures of Inventory and Effectiveness in 40% Loading Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	0	0	911
2	FCFS	0	0	921
3	SPT	0	0	930
4	EDD	0	0	910
	LM1			
5	random	0	0	905
6	FCFS	0	0	911
7	SPT	0	0	931
8	EDD	0	0	917
	LM2			
9	random	0	0	906
10	FCFS	0	0	900
11	SPT	0	0	961
12	EDD	0	0	911
	QM1			
13	random	0	0	945
14	FCFS	0	0	914
15	SPT	0	0	934
16	EDD	0	0	909

Table D.15: Customer Satisfaction Performance in 40% Loading Model

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	1605	1678	495	639
2	FCFS	1599	1672	497	637
3	SPT	1564	1646	526	603
4	EDD	1597	1668	496	635
	LM1				
5	random	1603	1679	496	638
6	FCFS	1602	1681	497	633
7	SPT	1549	1644	531	584
8	EDD	1595	1671	500	626
	LM2				
9	random	1603	1680	494	638
10	FCFS	1614	1691	481	649
11	SPT	1540	1604	560	575
12	EDD	1585	1660	508	620
	QM1				
13	random	1584	1665	515	619
14	FCFS	1590	1678	506	621
15	SPT	1561	1641	528	596
16	EDD	1600	1678	495	631

Table D.16: Measures of Inventory and Effectiveness in 55% Loading Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	9	5	503
2	FCFS	9	6	507
3	SPT	6	5	532
4	EDD	4	4	500
	LM1			
5	random	11	6	507
6	FCFS	10	6	507
7	SPT	11	7	542
8	EDD	3	3	503
	LM2			
9	random	11	7	505
10	FCFS	8	6	489
11	SPT	0	0	560
12	EDD	3	3	511
	QM1			
13	random	16	9	530
14	FCFS	15	9	521
15	SPT	4	4	532
16	EDD	3	3	498

Table D.17: Customer Satisfaction Performance in 55% Loading Model

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	2098	2209	155	891
2	FCFS	2085	2192	170	874
3	SPT	2040	2151	180	834
4	EDD	2134	2251	134	928
	LM1				
5	random	2073	2192	155	867
6	FCFS	2036	2155	169	834
7	SPT	2024	2144	180	818
8	EDD	2113	2241	128	897
	LM2				
9	random	2104	2246	156	898
10	FCFS	2102	2223	146	891
11	SPT	2004	2112	178	812
12	EDD	2079	2196	133	892
	QM1				
13	random	2127	2248	161	921
14	FCFS	2139	2263	161	928
15	SPT	2132	2153	188	830
16	EDD	2214	2338	122	992

Table D.18: Measures of Inventory and Effectiveness in 70% Loading Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	201	46	356
2	FCFS	194	48	364
3	SPT	166	44	345
4	EDD	221	45	355
	LM1			
5	random	182	47	337
6	FCFS	164	45	333
7	SPT	160	45	340
8	EDD	195	47	332
	LM2			
9	random	237	48	393
10	FCFS	199	48	345
11	SPT	142	42	320
12	EDD	186	46	318
	QM1			
13	random	224	46	405
14	FCFS	255	49	415
15	SPT	155	39	334
16	EDD	281	51	403

Table D.19: Customer Satisfaction Performance in 70% Loading Model

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	3178	3445	14	1731
2	FCFS	3224	3494	11	1762
3	SPT	2986	3216	13	1632
4	EDD	3316	3506	4	1838
	LM1				
5	random	3169	3511	10	1742
6	FCFS	3186	3506	7	1716
7	SPT	3055	3370	11	1578
8	EDD	3189	3477	5	1734
	LM2				
9	random	3328	3605	13	1879
10	FCFS	3259	3574	11	1835
11	SPT	2963	3211	20	1514
12	EDD	3318	3619	2	1832
	QM1				
13	random	3245	3548	8	1798
14	FCFS	3345	3636	8	1868
15	SPT	3026	3286	10	1565
16	EDD	3358	3649	4	1879

Table D.20: Measures of Inventory and Effectiveness with 85% Loading Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	1297	93	1311
2	FCFS	1325	93	1336
3	SPT	1047	93	1059
4	EDD	1374	98	1378
	LM1			
5	random	1358	94	1368
6	FCFS	1325	95	1332
7	SPT	1185	93	1196
8	EDD	1309	97	1314
	LM2			
9	random	1456	94	1469
10	FCFS	1446	94	1457
11	SPT	1034	90	1054
12	EDD	1418	98	1420
	QM1			
13	random	1395	94	1403
14	FCFS	1448	94	1456
15	SPT	1120	93	1130
16	EDD	1444	94	1448

Table D.21: Customer Satisfaction Performance in 85% Loading Model

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	6265	6990	866	4575
2	FCFS	6319	6995	852	4569
3	SPT	5952	6618	1018	4220
4	EDD	6094	6821	828	4332
	LM1				
5	random	6423	7243	693	4736
6	FCFS	6437	7136	785	4637
7	SPT	6247	7055	737	4668
8	EDD	6420	7264	615	4652
	LM2				
9	random	6395	7000	1098	4707
10	FCFS	6030	6556	1261	4356
11	SPT	6218	6662	1332	4606
12	EDD	5965	6659	904	4272
	QM1				
13	random	6233	6701	1224	4543
14	FCFS	6182	6784	1055	4433
15	SPT	5941	6487	1264	4197
16	EDD	6048	6746	1264	4197

Table D.22: Measures of Inventory and Effectiveness in 100% Loading State Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	931	45	1797
2	FCFS	815	44	1667
3	SPT	662	44	1680
4	EDD	644	44	1671
	LM1			
5	random	1011	49	1704
6	FCFS	877	46	1662
7	SPT	991	47	1728
8	EDD	863	51	1578
	LM2			
9	random	1171	40	2269
10	FCFS	907	40	2168
11	SPT	1148	41	2481
12	EDD	637	40	1541
	QM1			
13	random	997	40	2221
14	FCFS	1055	38	1904
15	SPT	1264	37	2027
16	EDD	930	34	1745

Table D.23: Customer Satisfaction Performance in 100% Loading Model

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	9206	10332	184	7514
2	FCFS	9105	10199	185	7328
3	SPT	8950	10025	184	7094
4	EDD	8477	9804	155	6726
	LM1				
5	random	9354	10398	237	7664
6	FCFS	9340	10398	177	7629
7	SPT	8482	9483	417	6734
8	EDD	8598	9741	345	6990
	LM2				
9	random	9643	10462	599	7955
10	FCFS	9019	9885	576	7056
11	SPT	9475	10165	768	7882
12	EDD	8777	9754	409	6983
	QM1				
13	random	10585	11195	409	8895
14	FCFS	9514	10140	497	7780
15	SPT	9356	9956	503	7649
16	EDD	9101	10026	334	7285

Table D.24: Measures of Inventory and Effectiveness in 200% Loading Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	3592	81	3776
2	FCFS	3335	83	3493
3	SPT	3101	81	3285
4	EDD	2964	82	3119
	LM1			
5	random	3713	77	3951
6	FCFS	3683	83	3860
7	SPT	2915	65	3332
8	EDD	3190	67	3535
	LM2			
9	random	4137	56	4735
10	FCFS	3234	55	4735
11	SPT	4106	52	4874
12	EDD	3071	61	3480
	QM1			
13	random	4683	65	5092
14	FCFS	3669	55	4166
15	SPT	3514	59	4016
16	EDD	3293	67	3627

Table D.25: Customer Satisfaction Performance in 200% Loading Model

Model No	CR id and DR	mean flow time(1)	mean flow time(2)	mean early	mean man'g error
	Random				
1	random	12405	13912	38	10714
2	FCFS	11395	12964	48	10043
3	SPT	11233	12931	39	9653
4	EDD	10953	12739	38	9237
	LM1				
5	random	13367	14788	15	11680
6	FCFS	12177	13726	34	10758
7	SPT	12164	13775	23	10903
8	EDD	12247	13842	85	10385
	LM2				
9	random	12112	13396	262	10428
10	FCFS	14664	15599	392	12457
11	SPT	11995	13116	348	10370
12	EDD	11978	13344	178	10408
	QM1				
13	random	12627	13519	175	10942
14	FCFS	13008	14083	151	11135
15	SPT	12734	13705	175	10986
16	EDD	12658	13864	127	10729

Table D.26: Measures of Inventory and Effectiveness in 300% Loading Model

Model No	CR id and DR	mean tardy	% of all jobs tardy	mean delivery error
	Random			
1	random	7036	95	7074
2	FCFS	6479	96	6526
3	SPT	6175	96	6214
4	EDD	5827	94	5864
	LM1			
5	random	7886	97	7901
6	FCFS	7160	96	7195
7	SPT	7394	96	7416
8	EDD	6821	91	6907
	LM2			
9	random	6731	74	6992
10	FCFS	6661	77	7059
11	SPT	8468	69	8810
12	EDD	6733	82	6911
	QM1			
13	random	6779	84	6954
14	FCFS	7115	87	7266
15	SPT	6896	83	7071
16	EDD	6808	87	6934

Table D.27: Customer Satisfaction Performance in 300% loading Model

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	0.38	2=	0.38	2	0.49	3	0.96	2
FCFS	0.38	2=	0.39	3	0.48	2	0.97	3
SPT	0.37	1=	0.37	1=	0.45	1	0.83	1
EDD	0.37	1=	0.37	1=	0.51	4	0.99	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	0.38	2	0.39	2=	0.48	2=	0.93	2
LM1	0.37	1=	0.37	1=	0.47	1	0.92	1
LM2	0.39	3	0.39	2=	0.48	2=	0.95	3
QM1	0.37	1=	0.37	1=	0.50	3	0.96	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	2.58	4	4.07	4	4.78	2=
FCFS	2.50	3	3.75	2	4.78	2=
SPT	2.47	2	3.78	3	5.01	3
EDD	2.46	1	3.20	1	4.45	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	2.46	2	3.90	4	4.87	4
LM1	2.60	4	3.51	1	4.50	2
LM2	2.50	3	3.76	3	4.65	3
QM1	2.45	1	3.56	2	3.99	1

Table D.28: Mean Queue Length

for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	0.25	2=	0.25	2=	0.30	2=	0.68	2=
FCFS	0.25	2=	0.25	2=	0.30	2=	0.68	2=
SPT	0.23	1	0.23	1	0.25	1	0.50	1
EDD	0.25	2=	0.25	2=	0.32	2=	0.72	3
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	0.24	1	0.24	1	0.29	2=	0.59	1
LM1	0.25	2	0.25	2=	0.28	1	0.60	2
LM2	0.25	2	0.24	1	0.29	2=	0.67	3
QM1	0.24	1	0.24	1	0.32	3	0.74	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	2.34	4	4.05	3	4.55	3
FCFS	2.23	1	3.78	2	4.47	2
SPT	2.24	2	4.11	4	4.74	4
EDD	2.27	3	3.04	1	4.28	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	1.83	1	3.58	1	4.33	3
LM1	2.62	4	3.60	2	4.16	2
LM2	2.07	2	3.62	3	5.53	4
QM1	2.55	3	4.18	4	4.02	1

Table D.29: Standard Deviation for the Time Avergae Queue Length Variable for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	0.79	1	0.79	1	0.86	2	1.30	2
FCFS	0.80	2	0.82	2	0.95	4	1.36	3
SPT	0.84	4	0.84	3	0.88	1	1.14	1
EDD	0.81	3	0.79	1	0.91	3	1.40	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	0.63	1	0.72	2	0.86	4	1.06	2
LM1	0.64	2	0.64	1	0.72	1	1.03	1
LM2	0.65	3	0.64	1	0.75	2	1.25	4
QM1	0.80	4	0.80	3	0.77	3	1.20	3

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	6.58	4	11.26	2	12.58	2
FCFS	6.41	2	11.35	3	13.54	3
SPT	6.43	3	13.27	4	13.54	3
EDD	5.86	1	8.70	1	11.81	1
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	4.11	2	7.84	1	10.00	2
LM1	4.98	3	9.01	3	9.60	1
LM2	4.05	1	8.65	2	12.60	4
QM1	8.09	4	12.01	4	10.21	3

Table D.30: Maximum of the Averages of the Time Average Queue Length for 40 - 300% Loading

	40%		55%		70%		85%	
	value	rank	value	rank	value	rank	value	rank
DISPATCHING RULE								
Random	66	3=	66	3	62	3	71	3
FCFS	64	2	65	2	63	3	70	2
SPT	61	1	61	1	56	1	61	1
EDD	66	3	68	4	63	3	73	4
CRITICAL RESOURCE IDENTIFICATION METHOD								
Random	62	1	64	2	60	1	64	1
LM1	67	4	67	4	60	1	65	2
LM2	64	2	63	1	61	2	70	3
QM1	66	3	66	3	63	3	77	4

	100%		200%		300%	
	value	rank	value	rank	value	rank
DISPATCHING RULE						
Random	90	2	100	2	95	2
FCFS	89	1	101	3	94	1
SPT	90	2	109	4	95	2
EDD	93	3	95	1	96	3
CRITICAL RESOURCE IDENTIFICATION METHOD						
Random	74	1	90	1	89	1
LM1	101	3	103	3	93	3
LM2	83	2	96	2	98	2
QM1	104	4	117	4	101	4

Table D.31: Coefficient of Variation of the Time Average Queue Length for 40 - 300% Loading

Model No	CR id and DR	mean queue length	standard deviation	max. ave queue length	coefficient of variation
	Random				
1	random	0.39	0.25	0.82	64
2	FCFS	0.39	0.24	0.78	62
3	SPT	0.38	0.22	0.78	58
4	EDD	0.38	0.23	0.75	61
	LM1				
5	random	0.38	0.25	0.75	66
6	FCFS	0.39	0.24	0.77	62
7	SPT	0.34	0.23	0.82	68
8	EDD	0.37	0.27	0.86	73
	LM2				
9	random	0.39	0.24	0.71	62
10	FCFS	0.40	0.24	0.72	60
11	SPT	0.39	0.27	0.94	69
12	EDD	0.37	0.24	0.78	65
	QM1				
13	random	0.37	0.27	0.91	73
14	FCFS	0.36	0.26	0.96	72
15	SPT	0.37	0.19	0.81	51
16	EDD	0.37	0.26	0.86	70

Table D.32: Queueing Statisitcs for the 40% Loading Model

Model No	CR id and DR	mean queue length	standard deviation	max. ave queue length	coefficient of variation
	Random				
1	random	0.39	0.25	0.83	64
2	FCFS	0.39	0.26	0.85	67
3	SPT	0.38	0.22	0.79	58
4	EDD	0.38	0.25	0.72	66
	LM1				
5	random	0.38	0.25	0.74	67
6	FCFS	0.39	0.24	0.78	62
7	SPT	0.34	0.23	0.82	68
8	EDD	0.38	0.27	0.86	71
	LM2				
9	random	0.39	0.23	0.69	59
10	FCFS	0.40	0.24	0.70	60
11	SPT	0.39	0.27	0.94	70
12	EDD	0.36	0.23	0.75	64
	QM1				
13	random	0.37	0.27	0.89	73
14	FCFS	0.37	0.26	0.95	70
15	SPT	0.37	0.19	0.81	51
16	EDD	0.37	0.26	0.85	70

Table D.33: Queueing Statisitcs for the 55% Loading Model

APPENDIX E

MSA SOURCE CODE

MSA.PRO

```
'     program MSA.PRO

'     CONTENTS - main boot up program

'     Edition 1.0   Version 1.0   6/4/88

'     Manufacturing Simulation and Analysis

'     Copyright  S Hurley  &  Dr J Driscoll

$compile exe
$include "common.pro"
$include "dimen.pro"
$include "m-com.pro"

screen 0,1,1,1
color 0,7
cls
color 1,7
locate 13,23 : print "MANUFACTURING SIMULATION AND ANALYSIS" : call insound
color 0,7 : delay 1
for i = 0 to 11
  call boxsing(8,7,1+I,2+I,78-(2*I),25-(2*I)) : delay 0.4
next i

width 80 : screen 0,1,1,1 : key off              '*** SET UP SCREEN ETC.

for k = 1 to 2
  for i = 1 to 11
    call box(8,7,1+i,2+i,78-(2*i),25-(2*i))
    j = i-1
    call boxsing(8,7,1+j,2+j,78-(2*j),25-(2*j)) : delay 0.1
  next i
  l = 11
  call boxsing(8,7,1+l,2+l,78-(2*l),25-(2*l))
next k
call box(8,7,18,59,5,3)
for i = 6 to 1 step -1
  locate 19,60 : print  i : delay 0.8
next i

z = 1
locate 4,1 : print "m-lic.pr"
chain "m-lic.pbc"                        '*** PICKS UP THE LICENCE SCREEN
```

M-CTL.PRO

```
'     program M-CTL.PRO

'     OBJECTIVES - main control program

'     Edition 2.0   Version 2.0   20/7/88

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"
$SEGMENT
call go

sub go
  locate 4,1 : print "m-ctl.pr"
  $include "shared.pro"
  if z = 1 then
    call MenuSel
  elseif z = 3 then
    call pjtpromenu
    z = 1 : call menusel
  elseif z = 5 then          '*** COMING BACK FROM M-MGT.TBC
    call utilities
    z = 1 : call menusel
  else
    call failsound
    call scrclear
    locate 9,1 : print "Z.........";z;" ???"
    locate 10,10 :print "WHOOPS!!! did not set z when chaining back to m-ctl"
    fg$=input$(9)
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub MenuSel
  shared dep = 20
  $include "shared.pro"
  call scrclear
  call border
  b = 0
  do
    select case b
    case = 0
      call menu(" MAIN MENU","","  A.  Project Data Management",_
    "","  B.  Utilities",_
    "","  C.  Advance Simulation",_
        "",_
    "  L.  Learning System Update","",_
    " "," Z.  Exit to DOS","")
      if ctl$(15,1) = "" then
    color 7,1
    locate 14,7 : print "  C.  Advance Simulation"
    locate 16,7 : print "  L.  Learning System Update"
```

```
  end if
  if val(pdt$(6,1)) = 0 then
color 7,1
locate 16,7 : print " L.  Learning System Update"
  end if
  call shadow(0,3,6,43,34,9)
  call fill(15,1,6,43,34,9)
  color 14,1
  locate 7,47 : print "PROJECT STATUS"
  color 15,1
  locate 10,47  : print "    Projects Defined:";
  color 14,1
  print " ";ctl$(13,1);" "
  color 15,1
  color 15,4 : locate 23,8 : print " Select an item: "
  call insound
  b$ = FNindata$(23,29,1)
  call toupper(b$)
  b = asc(b$)
case = 65
  call pjtpromenu
  b = 0
case = 66                          '*** IN SYS MENUS
  call utilities
  b = 0
case = 67
  if ctl$(15,1) = "" then
b = 100
  else
flag = 0
x = val(ctl$(11,1))
for i = 4 to 8
  if val(pjt$(x,i)) = 0 then
    flag = 1
  end if
next i
if flag = 1 then
  b = 101
else
  z = 0
  chain "adv.pbc"
end if
  end if
case = 76
  if ctl$(15,1) = "" then
b = 100
  elseif val(pdt$(6,1)) = 0 then
b = 102
  else
z = 2
chain "m-lsup.pbc"
  end if
case = 84
  call pjtrest("test.inf",test$(),999)
  ctl$(40,1) = "11"      '***
  test$(0,1) = "1"
  test$(0,0) = "0"   '0  '*** it gets incremented in test.pro, start=0
  z = 2
```

```
  chain "test.pbc"
case = 90
  b = 0
  color 15,3 : locate 23,8 : print "                          "
  call shadow(0,3,16,43,34,5)
  call fill(15,4,16,43,34,5)
  color 14,4
  locate 17,52 : print "Confirm Exit Y/N" : call insound
  x$ = FNindata$(19,59,3) : call toupper(x$) : x$ = left$(x$,1)
  do until x$ = "Y" or x$ = "N"
call failsound : x$ = left$(FNindata$(19,59,3),1) : call toupper(x$)
  loop
  if x$ = "N" then
    call fill(0,3,16,43,35,6)
    b = 0
  else
    call dumpho("f-ctl.inf",ctl$(),50,1)
    call dumpho("f-pjt.inf",pjt$(),15,15)
    color 0,0
    exit sub
  end if
case = 100
  b = 0
  color 0,3 : locate 23,8 : print "                        "
  call shadow(0,3,6,43,35,9)
  call fill(15,4,6,43,35,9)
  call box(15,4,6,44,33,9)
  color 14,4
  locate 8,45  : print "   NO PROJECT IS ACTIVE USE    "
  locate 10,45 : print "    PROJECT DATA MANAGEMENT    "
  locate 12,45 : print " PRESS <space bar> TO CONTINUE "
  color 15,1
  call failsound : call spce
  call fill(0,3,6,43,36,10)
case = 101
  b = 0
  color 0,3 : locate 23,8 : print "                        "
  call shadow(0,3,6,43,34,9)
  call fill(15,4,6,43,34,9)
  call box(15,4,6,44,32,9)
  color 14,4
  locate 8,45  : print "      NOT ALL DATA FILES      "
  locate 10,45 : print "       ARE INITIALISED        "
  locate 12,45 : print "   PRESS <space> TO CONTINUE  "
  color 15,1
  call failsound : call spce
  call fill(0,3,6,43,35,10)
case = 102
  b = 0
  color 0,3 : locate 23,8 : print "                        "
  call shadow(0,3,6,43,35,9)
  call fill(15,4,6,43,35,9)
  call box(15,4,6,44,33,9)
  color 14,4
  locate 8,45  : print "  At least one period must be  "
  locate 9,45  : print "   simulated before running    "
  locate 10,45 : print "    Learning System Update     "
  locate 12,45 : print " Press <space bar> to continue "
```

```
      color 15,1
      call failsound : call spce
      call fill(0,3,6,43,36,10)
    case else
      color 0,3 : locate 23,8 : print "                          "
      color 15,1
      call shadow(0,3,6,43,34,9)
      call fill(15,4,6,43,34,9)
      call box(15,4,6,44,32,9)
      color 14,4
      locate 8,45  : print "        INVALID INPUT      "
      locate 10,45 : print "      PRESS <space bar>    "
      locate 12,45 : print "         TO CONTINUE       "
      color 15,1
      call failsound : call spce
      call fill(0,3,6,43,35,10)
      color 15,1
      b = 0
    end select
  loop until abc = 90
end sub


sub pjtpromenu
  $include "shared.pro"
  b = -1
  do
    select case b
    case = -1
      call fill(0,3,4,1,80,22)
      call menu("  PROJECT DATA MANAGEMENT ","","    A.  Create",_
    "    B.  Load","    C.  Delete","",_
    "    D.  Input",_
    "    E.  Edit","    F.  Logic Check","    G.  View/Print Reports",_
    "    H.  Overwrite Root","",_
    "    P.  Previous Menu")
      call fill(0,1,20,5,32,2)
      call shadow(0,3,20,5,32,2)
      locate 20,11 : print "P.  Previous Menu"
      color 14,1
      locate 9,9  : print "Projects"
      locate 13,9 : print "Project data"
      color 15,1
      b = 0
    case = 0
      color 7,1
      if val(ctl$(13,1)) = 0 then
    locate 11,9 : print "  B.  Load"
    locate 12,9 : print "  C.  Delete"
    locate 14,9 : print "  D.  Input"
    locate 15,9 : print "  E.  Edit"
    locate 16,9 : print "  F.  Logic Check"
    locate 17,9 : print "  G.  View/Print Reports"
    locate 18,9 : print "  H.  Overwrite Root
      end if
      if val(ctl$(11,1)) = 0 then
    locate 14,9 : print "  D.  Input"
    locate 15,9 : print "  E.  Edit"
```

```
 locate 16,9 : print "  F.  Logic Check"
 locate 17,9 : print "  G.  View/Print Reports"
 locate 18,9 : print "  H.  Overwrite Root"
   end if
   if val(ctl$(13,1)) = 15 then
 locate 10,9 : print "  A.  Create"
   end if
   color 15,4 : locate 23,8 : print " Select an item: "
   call insound
   b$ = FNindata$(23,29,1)
   call toupper(b$)
   b = asc(b$)
 case = 65
   b = 0
   if val(ctl$(13,1)) = 15 then
     call shadow(0,3,6,43,34,9)
     call fill(15,4,6,43,34,9)
 call box(15,4,6,44,32,9)
     locate 8,45  : print "  Maximum number of projects"
     locate 10,45 : print "    defined - use 'DELETE'  "
     locate 12,45 : print "      press ";
 color 14,4 : print "<space bar>   "

 call failsound : call spce
     call fill(0,3,6,43,35,10)
     color 15,1
   else
 call WipeEm
 z = 1
 color 0,3 : locate 23,8 : print "                          "
 chain "m-create.pbc"
   end if
 case = 66
   b = 0
   if val(ctl$(13,1)) = 0 then
 b = 101
   else
     z = 2
     chain "m-load.pbc"
   end if
 case = 67
   b = 0
   if val(ctl$(13,1)) = 0 then
 b = 101
   else
     call fill(0,3,4,1,80,21)
     z = 1
 chain "m-del.pbc"
   end if
 case = 68
   b = 0
   if val(ctl$(13,1)) = 0 then
 b = 101
   elseif ctl$(15,1) = "" then
 b = 100
   else
     call fill(0,3,4,1,80,21)
 z = 2
```

```
    chain "m-create.pbc"
      end if
    case = 69
      b = 0
      if val(ctl$(13,1)) = 0 then
    b = 101
      elseif ctl$(15,1) = "" then
    b = 100
      else
    z = 1
    call fill(0,3,4,1,80,21)
    chain "m-edit.pbc"
      end if
    case = 70
      b = 0
      if val(ctl$(13,1)) = 0 then
    b = 101
      elseif ctl$(15,1) = "" then
    b = 100
      else
    call logiccheck
    b = -1
      end if
    case = 71
      b = 0
      if val(ctl$(13,1)) = 0 then
    b = 101
      elseif ctl$(15,1) = "" then
    b = 100
      else
        call fill(0,3,4,1,80,21)
    z = 1
    chain "m-rep.pbc"
    b = -1
      end if
    case = 72
      b = 0
      if val(ctl$(13,1)) = 0 then
    b = 101
      elseif ctl$(15,1) = "" then
    b = 100
      else
        call fill(0,3,4,1,80,21)
    z = 1
    call overwrite
    b = -1
      end if
  case = 81
     if val(ctl$(14,1)) = 0 then
   b = 100
     else
   z = 1
   chain "m-sheet.pbc"
     end if
  case = 170
     if val(ctl$(14,1)) <> 0 then
       do
         flag = 0
```

```
      x = val(ctl$(11,1))
      for i = 3 to 6
        if left$(pjt$(x,i),1) = "N" then flag = 1
      next i
      if flag = 1 then
        call shadow(0,3,6,43,34,9)
        call fill(15,4,6,43,34,9)
        call box(15,4,6,44,32,9)
        locate 8,45  : print " NOT ALL FILES ARE DEFINED"
        locate 10,45 : print "     PRESS <space bar>    "
        locate 12,45 : print "    TO ENTER INPUT MENU   "
    call insound : call spce
        call fill(0,3,6,43,35,10)
        x$ = input$(1)
        z = 2
    locate 4,1 : print "m-create"
    chain "m-create.pbc"
      else
    locate 4,1 : print "que.pro "
    chain "que.pbc"
      end if
      b = 0
    loop until a = 123
  else
b = 100
  end if
case = 80
  b = 0
  call fill(0,3,6,5,33,18)
  exit loop
case = 100
  b = 0
  call shadow(0,3,6,43,34,9)
  call fill(15,4,6,43,34,9)
  call box(15,4,6,44,32,9)
  color 14,4
  locate 8,45  : print "     NO PROJECT IS ACTIVE "
  locate 10,45 : print "    USE 'LOAD' OR 'CREATE'"
  locate 12,45 : print "      PRESS <space bar>   "
  color 15,1
  call failsound : call spce
  call fill(0,3,6,43,35,10)
case = 101
  call shadow(0,3,6,43,34,9)
  call fill(15,4,6,43,34,9)
  call box(15,4,6,44,32,9)
  color 14,4
  locate 8,45  : print "     NO EXISTING PROJECTS "
  locate 10,45 : print "         USE 'CREATE'     "
  locate 12,45 : print "      PRESS <space bar>   "
  color 15,1
  call failsound : call spce
  call fill(0,3,6,43,35,10)
  b = 0
case else
  color 0,3 : locate 23,8 : print "                        "
  color 15,1
  call shadow(0,3,6,43,34,9)
```

```
      call fill(15,4,6,43,34,9)
      call box(15,4,6,44,32,9)
      color 14,4
      locate 8,45  : print "        INVALID INPUT      "
      locate 10,45 : print "      PRESS <space bar>    "
      locate 12,45 : print "         TO CONTINUE       "
      color 15,1
      call failsound : call spce
      call fill(0,3,6,43,35,10)
      color 15,1
      b = 0
    end select
  loop until abc = 90
end sub


sub utilities
  $include "shared.pro"

  call scrclear
  call border
  b = 0
  do
    select case b
    case = 0
      call menu("     UTILITIES","  A.  Initialisation",_
        "","  B.  Licence Screen",_
    "","  C.  Development Team",_
    "",_
    "  V.  View Project Files","","","  P.  Previous Menu","","")
      if ctl$(15,1) = "" then
    color 7,1 : locate 15,7
    print "  V.  View Project Files"
      end if
      color 15,4 : locate 23,8 : print " Select an item: "
      call insound
      b$ = FNindata$(23,29,1)
      call toupper(b$)

      b = asc(b$)
    case = 65
      color 15,3 : locate 23,8 : print "                      "
      call fill(0,4,16,20,48,7)
      call shadow(0,3,16,20,48,7)
      color 31,4
      locate 17,24 : print "                WARNING !"
      color 14,4
      locate 19,24 : print "This Will Erase All Simulations Projects"
      color 15,4
      locate 21,24 : print "          Continue (Y/N):"
      x$ = FNindata$(21,51,3) : call toupper(x$) : x$ = left$(x$,1)
      do until x$ = "N" or x$ = "Y"
    call failsound : x$ = FNindata$(21,51,3)
    call toupper(x$) : x$ = left$(x$,1)
      loop
      if x$ = "Y" then
    call InitSimData
      end if
```

```
        call fill(0,3,16,20,49,8)
        b = 0
      case = 66
        z = 3
        chain "m-lic.pbc"
        b = 0
      case = 67                                  '*** IN SYS MENUS
        call na
        b = 0
      case = 80
        b = 0
        exit loop
      case = 86
        if ctl$(15,1) = "" then
      b = 101
        else
      b = 0
      z = 1
      chain "m-view.pbc"
        end if
      case else

        color 0,3 : locate 23,8 : print "                          "
        color 15,1
        call shadow(0,3,6,43,34,9)
        call fill(15,4,6,43,34,9)
        call box(15,4,6,44,32,9)
        color 14,4
        locate 8,45  : print "        INVALID INPUT      "
        locate 10,45 : print "        PRESS ANY KEY    "
        locate 12,45 : print "         TO CONTINUE       "
        color 15,1
        call failsound : x$ = input$(1)
        call fill(0,3,6,43,35,10)
        color 15,1
        b = 0
      end select
    loop until abc = 91
end sub

sub WipeEm
  shared job$(), queues$(), cal$(), mac$(), bomho$()

  for i = 1 to jobnum                        '***INITIALISING JOB FILE
    for j = 1 to 15                                    '***BACK TO BLANKS
      job$(i,j) = ""
    next j
  next i
  for i = 1 to 4
    for j = 1 to resources
      queues$(j,i) = ""
    next j
  next i
  for i = 1 to 104
    for j = 1 to 8
      cal$(i,j) = ""
    next j
  next i
```

```
  for i = 1 to 15
    for j = 1 to resources
      mac$(j,i) = ""
    next j
  next i
  for i = 1 to 20
    for j = 1 to 64
      bomho$(i,j) = ""
    next j
  next i
end sub


Sub InitSimData                          '*** INITIALISE THE MASTER
  shared simu$(), pjt$(), cal$(), ctl$()              '*** SIMULATION DATA FILE

  call WaitOn
   ctl$(13,1) = "0"
   ctl$(11,1) = "0"
   ctl$(10,1) = "0"
   ctl$(14,1) = "0"

  for i = 1 to 15
    for j = 1 to 15
      pjt$(i,j) = ""
    next j
  next i
  pjt$(0,1) = "Pjt Code" : pjt$(0,2) = "Sim Runs" : pjt$(0,3) = "Pjt Name"
  pjt$(0,4) = "pjt data" : pjt$(0,5) = "Jobs"     : pjt$(0,6) = "Res's"
  pjt$(0,7) = "BoM"      : pjt$(0,8) = "Calr"     : pjt$(0,9) = "LS up"
  for i = 1 to 15
    pjt$(i,0) = "Pjt"+str$(i)
  next i

  '*** sim$() is a transcient software control
  '*** for controlling the simulation, mainly for the LS update.

  simu$(1,0) = ""
  simu$(2,0) = ""
  simu$(3,0) = ""
  simu$(4,0) = ""
  simu$(5,0) = ""
  simu$(5,1) = "CR sel LS"
  simu$(6,0) = "SH LS"
  simu$(6,1) = ""
  simu$(7,0) = ""
  simu$(7,1) = ""
  simu$(8,0) = ""
  simu$(9,0) = ""
  simu$(10,0) = ""
  simu$(11,0) = ""
  simu$(12,0) = ""
  simu$(13,0) = ""
  simu$(14,0) = "Bk"
  simu$(15,0) = "Bk"
  simu$(16,0) = "Bk"
  simu$(17,0) = "Bk"
  simu$(18,0) = "Bk"
```

```
  simu$(19,0) = "Bk"
  simu$(20,0) = "Bk"
  call dumpho("f-pjt.inf",pjt$(),15,15)
  call dumpho("f-ctl.inf",ctl$(),50,1)
  call waitoff
end sub

sub logiccheck
  $include "shared.pro"
  logic$ = ""

'*** CHECK THAT ALL RESOURCES USED ARE DEFINED
  file$ = ctl$(15,1)+"~sqd.inf"
  open file$ for input as #4
  totalmac = 0
  do until eof(4)
    input #4, x$
    if val(mid$(x$,21,5)) > totalmac then
      totalmac = val(mid$(x$,21,5))
    end if
  loop
  mac = 1
  do until mac$(mac,1) = ""
    incr mac
  loop
  decr mac
  if mac < totalmac then
    logic$ = "F"
  else
    logic$ = "P"
  end if
  close #4

'*** CHECKS THAT ALL JOBS IN JOB$() ARE IN THE SEQUENCING DATABASE
  row = 0
  do until job$(row+1,1) = ""
    incr row
    if row = 20 then exit loop
  loop
  file$ = ctl$(15,1)+"~sqd.inf"
  open file$ for input as #4
  holdjob$ = "" : holdbom$ = "" : flag = 0
  do until eof(4)
    input #4, x$
    x$ = mid$(x$,11,5)
    x$ = FNtrun(x$)
    if x$ <> holdjob$ then
      if instr(holdbom$,x$) then flag = 1
      holdbom$ = holdbom$ + x$
      holdjob$ = x$
      for i = 1 to jobnum
    if job$(i,1) = "" then exit for
    if x$ = FNtrun$(job$(i,1)) then
      decr row
      if row = 0 then exit for
    end if
      next i
    end if
```

```
  loop
  close #4
  if row > 0 then
    logic$ = logic$ + "F"
  else
    logic$ = logic$ + "P"
  end if
  if flag = 1 then
    logic$ = logic$ + "F"
  else
    logic$ = logic$ + "P"
  end if

'*** DUPLICATE BoM DEFINITION

'*** INFORM THE USER OF THE RESULT
  call shadow(0,3,6,43,34,16)
  call fill(14,1,6,43,34,16)
  color 14,1
  locate 7,46  : print "PROJECT STATUS"
  color 15,1
  locate 9,46  : print "Project name ..... ";
  color 4,7 : print " "; : print using "\ \";ctl$(15,1);
    print " " : color 15,1
  locate 10,46 : print "Project version..... ";
  x$ = ctl$(16,1) : call toupper(x$)
  if left$(x$,1) = "I" then x$ = "Root"
    color 4,7 : print " "; : print using "\  \";x$; : color 15,1
  locate 13,46
  color 14,1 : print "LOGIC TEST RESULTS"
  color 15,1
  locate 15,46 : print "Resource Database - ";
  locate 15,69 : color 4,7
  if left$(logic$,1) = "F" then
    print " FAIL "
  else
    print " PASS "
  end if
  color 15,1
  locate 17,46 : print "Sequence Database 1 - "
  locate 17,69 : color 4,7
  if mid$(logic$,2,1) = "F" then
    print " FAIL "
  else
    print " PASS "
  end if
  color 15,1
  locate 19,46 : print "Sequence Database 2 - "
  locate 19,69 : color 4,7
  if mid$(logic$,3,1) = "F" then
    print " FAIL "
  else
    print " PASS "
  end if

  color 15,3 : locate 23,8 : print "                    "
  color 15,4
  locate 23,20 : print " <space bar> = continue, <esc> for Help "
```

```
    call insound
    x$ = input$(1)

    do until asc(x$) = 32 or asc(x$) = 27
      call failsound : x$ = input$(1)
    loop
    if asc(x$) = 27 then
      color 15,3
      locate 23,20 : print "                                                "
      call helplogic
    end if
    call fill(0,3,6,43,35,16)
    color 15,3
    locate 23,20 : print "                                                "
  end sub      '*** statusbox

  sub helplogic
    call shadow(0,3,7,6,32,15)
    call fill(4,7,7,6,32,15)
    color 4,7
    locate 8,7  : print "      L O G I C  H E L P"
    color 14,7
    locate 10,7 : print "      Resource Database"
    color 4,7
    locate 11,7 : print "  FAIL - Not all resources"
    locate 12,7 : print "  required are defined."
    color 14,7
    locate 14,7 : print "     Sequence Database 1"
    color 4,7
    locate 15,7 : print "  FAIL - Not all jobs to be"
    locate 16,7 : print "  launched have a defined BoM."
    color 14,7
    locate 18,7 : print "     Sequence Database 2"
    color 4,7
    locate 19,7 : print "  FAIL - Duplicate BoM"
    locate 20,7 : print "  definition.  Edit BoM file."
    color 15,4
    locate 24,7 : print " Press <space bar> to continue ";
    call insound
    call spce
    color 15,3 : locate 24,7 : print "                                    ";
    call fill(4,3,7,6,33,16)
  end sub

  sub overwrite
    $include "shared.pro"

    call shadow(0,3,8,23,34,10)
    call fill(4,7,8,23,34,10)
    color 14,7
    locate 9,26  : print "     **** WARNING ****"
    color 4,7
    locate 11,26 : print "Overwriting the ROOT causes"
    locate 12,26 : print " all simulation runs under "
    locate 14,26 : print "       to be erased."
    color 15,7
    locate 13,37 : print " ";ctl$(15,1);" "
    color 4,7
```

```
color 14,7
locate 16,26
print "  O";
color 4,7 : print " = Overwrite, ";
color 14,7 : print "Q";
color 4,7 : print "= Quit  "
color 15,1
locate 19,29 : print " Select O or Q: "
x$ = FNindata$(19,47,1) : call toupper(x$) : x$ = left$(FNtrun$(x$),1)
do until x$ = "O" or x$ = "Q"
  call failsound : x$ = FNindata$(19,47,1) : call toupper(x$)
loop
if x$ = "Q" then
  exit sub
else
  color 0,3 : locate 19,29
  print "                "
  call fill(7,4,15,40,26,5)
  call shadow(0,3,15,40,26,5)
  color 30,4
  locate 16,40 : print "     **** WARNING ****"
  color 15,4
  locate 18,40 : print "    Are you sure (Y/N)?"
  locate 21,43 : print " Input Y or N : "
  x$ = FNindata$(21,61,1) : call toupper(x$)
  do until x$ = "Y" or x$ = "N"
    call failsound : x$ = FNindata$(21,61,1) : call toupper(x$)
  loop
end if
if x$ = "N" then
  exit sub
else
  color 0,3
  locate 21,43 : print "    "
  call fill(4,7,16,45,31,6)
  call shadow(0,3,16,45,31,6)
  call waiton
  color 14,7
  locate 18,50 : print "OVERWRITING ROOT FILE"
  call pjtkill(ctl$(15,1),"inf")
  y$ = ctl$(15,1) : z$ = "inf"
  call dump(y$,z$,val(pdt$(6,1)))
  locate 20,48 : print "DELETING SIMULATION FILES
  x$ = pjt$(val(ctl$(11,1)),2)
  if x$ = "#" then
    exit sub
  else
    for i = 1 to val(x$)
  if i < 10 then
    y$ = "00"+FNtrun$(str$(i))
  else
    y$ = "0"+FNtrun$(str$(i))
  end if
  call pjtkill(ctl$(15,1),y$)
    next i
  end if
  pjt$(val(ctl$(11,1)),2) = "#"
end if
```

```
  ctl$(16,1) = "INF"
  ctl$(17,1) = "INF"
  call rblock
  call waitoff
end sub

sub pjtkill(file1$,file2$)
  $include "shared.pro"
  f$ = file1$+"~pdt."+file2$ : kill f$
  f$ = file1$+"~bom."+file2$ : kill f$
  f$ = file1$+"~cal."+file2$ : kill f$
  f$ = file1$+"~job."+file2$ : kill f$
  f$ = file1$+"~mac."+file2$ : kill f$
  f$ = file1$+"~que."+file2$ : kill f$
  f$ = file1$+"~rsj."+file2$ : kill f$
  f$ = file1$+"~rsm."+file2$ : kill f$
  f$ = file1$+"~rsg."+file2$ : kill f$
  f$ = file1$+"~crh."+file2$ : kill f$
  f$ = file1$+"~shh."+file2$ : kill f$
  f$ = file1$+"~lsy."+file2$ : kill f$
  f$ = file1$+"~wrk."+file2$ : kill f$
  f$ = file1$+"~bns."+file2$ : kill f$
  f$ = file1$+"~sqr."+file2$ : kill f$
  call dumpho("f-pjt.inf",pjt$(),15,15)
end sub
```

M-LIC.PRO

```
'     program M-LIC.PRO

'     OBJECTIVES - creates the licence screen

'     Edition 1.0   Version 1.0   4/3/88

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

' ***** COMMON SUBROUTINE BLOCK *****

$include "common.pro"
$include "m-com.pro"

call go

sub go
  locate 4,1 : print "m-lic.pr"
  $include "shared.pro"
  if z = 1 then
    call licence
    chain "m-pic.pbc"
  elseif z = 2 then
    call licence
    z = 1
    chain "m-ctl.pbc"
  elseif z = 3 then
    call licence
    z = 5
    chain "m-ctl.pbc"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub licence
  $include "shared.pro"
  A = 2
  call scrclear
  call fill(15,1,2,44,32,7)
  call shadow(0,3,2,44,32,7)
  color 14,1
  locate 3,46 : print"  Manufacturing Simulation  "
  locate 4,46 : print"        and Analysis        "
  locate 5,46 : print"           v 2.0            "
  locate 6,46 : print"       Copyright 1991       "
  locate 7,46 : print"    S Hurley  J Driscoll    "

  call fill(15,1,2,4,32,7)
  call shadow(0,3,2,4,32,7)
  color 15,1
  locate 3,6 : print "     LICENCE CONDITIONS"
  color 14,1
  locate 5,6 : print " USER(S)   :"
  locate 6,6 : print " LOCATION  :"
  locate 7,6 : print " PERIOD    :"
  color 15,1
```

```
  locate 5,19 : print "Dr J Driscoll"
  locate 6,19 : print "unlimited    "
  locate 7,19 : print "31.12.99     "
  call fill(15,12,10,4,72,13)
  call shadow(0,3,10,4,72,13)
  color 15,4
  d = 10
  incr d : locate d,6
  print "No software, documentation or charts provided under this licence may" ;
  incr d : locate d,6
  print "be copied or ammended in any form by any means, including electronic"
  incr d : locate d,6
  print "storage, without the prior written consent of the copyright holders."
  incr d : incr d : locate d,6
  print "Use of all software, documentation and charts is limited to persons,"
  incr d : locate d,6
  print "locations, licence period and conditions shown.   Unauthorised third"
  incr d : locate d,6
  print "party or licencee use without prior written consent is prohibited."
  incr d : incr d : locate d,6
  print "This product is licenced without warrenty of any kind,  expressed or"
  incr d : locate d,6
  print "implied, as to fitness for use.    The author shall not be liable to"
  incr d : locate d,6
  print "any persons with respect to any loss or damage caused by product use."

  locate 24,24
  color 15,1
  print "  Press <space bar> to continue  "; : call insound
  x$ = input$(1)
  do until asc(x$) = 32
    call failsound
    x$ = input$(1)
  loop

  locate 24,24
  color 15,3
  print "                                        ";
end sub
```

M-PIC.PRO

```
'     program M-PIC.PRO

'     OBJECTIVES - restores all the initial values into the arrays

'     Edition 1.0   Version 1.0   17/11/89

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  $include "shared.pro"
  call restfiles
  z = 1
  chain "m-ctl.pbc"
end sub

' ***** SUBROUTINE BLOCK *****

sub restfiles
  $include "shared.pro"

  call WaitOn
  count = 0
  call fill(15,1,13,55,17,11)
  call shadow(0,3,13,55,17,11)
  color 14,1
  locate 14,57 : print "LOADING FILES"
  color 15,1
  co = 1
  call restorho("f-ctl.inf",ctl$(),50,1)      '*** TO RESTORE ALL THE FILES
  x = val(ctl$(25,1))
  locate 15+co,56 : print co;" f-ctl.inf" : incr co : delay x
  call restorho("f-pjt.inf",pjt$(),15,15)
  locate 15+co,56 : print co;" f-pjt.inf" : incr co : delay x
  'call restorS2("f-sim.inf",sim$(),20,1)
  locate 15+co,56 : print co;" f-sim.inf" : incr co : delay x
  'call restorS2("f-sys.inf",sy$(),20,1)
  locate 15+co,56 : print co;" f-sys.inf" : incr co : delay x

  'call restorS2("f-gra.per",g$(),20,2)
  locate 15+co,56 : print co;" f-gra.per" : incr co : delay x
  'call restorS2("f-uin.per",u2$(),10,14)
  locate 15+co,56 : print co;" f-uin.per" : incr co : delay x
  'call restorS2("f-usr.per",u1$(),10,6)
  locate 15+co,56 : print co;" f-usr.per" : incr co : delay x
  delay 1.5
  call WaitOff
  ctl$(10,1) = "0"
  ctl$(11,1) = ""
  ctl$(12,1) = "0"
```

```
    ctl$(14,1) = "0"
    ctl$(15,1) = ""
    ctl$(16,1) = ""
    ctl$(17,1) = ""
    ctl$(30,1) = ""
    ctl$(31,1) = ""
    ctl$(32,1) = ""
    ctl$(40,1) = ""
end sub
```

M-MGT.PRO

```
'     program M-MGT.PRO

'     OBJECTIVES - main system management program

'     Edition 1.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-init.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-mgt.pr"
  if z = 1 then
    call mgtmenu
    z = 1
    chain "m-ctl.pbc"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub mgtmenu
  shared ma$(), g$(), g%(), u1$(), u2$()
  shared ac$(), sy$(), ctl$()
  dep = 20

  d = 0
  do
    select case d
      case = 0
        call Menu("SYSTEMS MANAGEMENT","A. Initialise Files",_
          "B. Edit User Data","C. System Interegation",_
          "D. View/Edit","","","P. Previous Menu","","","","","")
        color 15,4
        locate 22,7 : print "Select an item: " : call insound
        d$ = FNindata$(22,26,1)
        call toupper(d$)
        d = asc(d$)
      case = 65                    '*** SYSTEM INITIALISATION
        call fill(0,3,4,1,80,20)
        call initialise
        call SysMgt("Y")
        d = 0
      case = 66                    '*** EDIT THE USER FILES
        call na
        d = 0
      case = 67                    '*** SYSTEM INTEREGATION
        call SysMgt("N")
        d = 0
      case = 68                    '*** VIEW/EDIT SYSTEM PARAMETERS
        call SysMgt("N")               '*** MIGHT HAVE PEOPLE VIEW BUT NO EDIT HENCE
```

```
        d = 0                    '*** SYSTEM INTERROGATION FUNCTION
      case = 80
        d = 0
      case else
        call failsound
        color 15,4 : locate 22,57 : print " INVALID CHOICE "
        delay 1
        color 15,3 : locate 22,57 : print "                "
        d = 0
    end select
  loop until d = 80
end sub

sub SysMgt(init$)
  shared dep = 20
  shared ctl$()
  shared ma$(), g$(), g%()
  shared u1$(), u2$(), m$()
  shared hold$(), s1$(), ac$()
  shared mn$(), pn$(), sn$(), cn$(), p1%()
  shared sy$()
  shared event$()
  do
    casex = 0
    if init$ = "N" then    ' ******************

      call fill(15,3,5,1,80,19)
      call fill(15,1,5,20,40,3)
      call box (15,1,5,21,38,3)
      color 14,9
      locate 6,24 : print "S Y S T E M  M A N A G E M E N T"
      color 15,1
      call fill(15,3,8,1,80,18)
      call fill(15,1,9,3,22,15)
      call boxsing(15,1,9,3,22,15)

      color 14,9
      locate 9,7 : print " System Files "
      locate 23,8 : print " 0 to exit "
      color 15,1
      locate 9,10
      for i = 1 to 10
        locate 11+i,5
        print i
        locate 11+i,8
        print sy$(i,1)
      next i

      call fill(15,1,15,35,40,5)
      call box (15,1,15,35,40,5)
      locate 16,38 : print "    Please Select a data file:" : call insound
      x$=FNindata(18,50,3)
      x = val(x$)
      call fill(0,3,4,1,80,20)

    else      '******************
      incr x
      if x = 1 then
```

```
        call fill(15,1,20,5,40,3)
        call boxsing(15,1,20,5,38,3)
        color 14,9
        locate 21,7 : print "I N T I A L I S A T I O N"
      end if
    end if
IF X = 2 THEN EXIT LOOP
    select case x
      case = 1
    if init$ = "N" then call restorho("f-ctl.inf",ctl$(),50,1)
        call SysInter(ctl$(),50,1,flag)
        if flag = 1 or LEFT$(init$,1) ="Y" then
          if init$ = "Y" then
            locate 21,50 : print " Initialise (Y/N)? "
            x$ = FNindata$(21,70,4)
            color 0,3 : locate 21,50 : print "                                    "
            color 15,1
            if left$(x$,1) = "N" then exit select
          end if
      call dumpho("f-ctl.inf",ctl$(),50,1)
        end if
      case = 2
        if init$ = "N" then call restorS2("f-gra.per",g$(),20,2)
        call SysInter(g$(),20,2,flag)
        if flag = 1 or init$ ="Y" then
          if init$ = "Y" then
            locate 21,50 : print " Initialise (Y/N)? "
            x$ = FNindata$(21,70,4)
            color 0,3 : locate 21,50 : print "                                    "
            color 15,1
            if left$(x$,1) = "N" then exit select
          end if
          call dumpS2("f-gra.per",g$(),20,2)
        end if
      case = 3
        if flag = 1 or init$ ="Y" then
          if init$ = "Y" then
            locate 21,50 : print " Initialise (Y/N)? "
            x$ = FNindata$(21,70,4)
            color 0,3 : locate 21,50 : print "                                    "
            color 15,1
            if left$(x$,1) = "N" then exit select
          end if
        end if
      case = 4
        if init$ = "N" then call restorS2("f-mac.inf",m$(),20,4)
        call SysInter(m$(),20,4,flag)
        if flag = 1 or init$ ="Y" then
          if init$ = "Y" then
            locate 21,50 : print " Initialise (Y/N)? "
            x$ = FNindata$(21,70,4)
            color 0,3 : locate 21,50 : print "                                    "
            color 15,1
            if left$(x$,1) = "N" then exit select
          end if
          call dumpS2("f-mac.inf",m$(),20,4)
        end if
      case = 5
```

```
  if flag = 1 or init$ ="Y" then
    if init$ = "Y" then
      locate 21,50 : print " Initialise (Y/N)? "
      x$ = FNindata$(21,70,4)
      color 0,3 : locate 21,50 : print "                    "
      color 15,1
      if left$(x$,1) = "N" then exit select
    end if
  end if
case = 6
  if flag = 1 or init$ ="Y" then
    if init$ = "Y" then
      locate 21,50 : print " Initialise (Y/N)? "
      x$ = FNindata$(21,70,4)
      color 0,3 : locate 21,50 : print "                    "
      color 15,1
      if left$(x$,1) = "N" then exit select
    end if
  end if
case = 7
  if flag = 1 or init$ ="Y" then
    if init$ = "Y" then
      locate 21,50 : print " Initialise (Y/N)? "
      x$ = FNindata$(21,70,4)
      color 0,3 : locate 21,50 : print "                    "
      color 15,1
      if left$(x$,1) = "N" then exit select
    end if
  end if
  if flag = 1 or init$ ="Y" then
    if init$ = "Y" then
      locate 21,50 : print " Initialise (Y/N)? "
      x$ = FNindata$(21,70,4)
      color 0,3 : locate 21,50 : print "                    "
      color 15,1
      if left$(x$,1) = "N" then exit select
    end if
  end if
case = 9
  if flag = 1 or init$ ="Y" then
    if init$ = "Y" then
      locate 21,50 : print " Initialise (Y/N)? "
      x$ = FNindata$(21,70,4)
      color 0,3 : locate 21,50 : print "                    "
      color 15,1
      if left$(x$,1) = "N" then exit select
    end if
  end if
case = 10
  if flag = 1 or init$ ="Y" then
    if init$ = "Y" then
      locate 21,50 : print " Initialise (Y/N)? "
      x$ = FNindata$(21,70,4)
      color 0,3 : locate 21,50 : print "                    "
      color 15,1
      if left$(x$,1) = "N" then exit select
    end if
  end if
```

```
      case else
        x = 0
    end select
  loop until x = 0
end sub

sub SysInter(file$(2),DIM1,DIM2,flag)
  shared ma$(), ctl$()
  flag = 0
  c = 2
  d = 8      '*** depth down screen
  t = 0
  iconrh = 0
  e = 15                '*** bottom bit
  icon = 1
  call fill(15,1,d-2,1,80,13)
  call box(15,1,d-2,1,80,13)
  color 14,9
  locate d-2,25 : print " "file$(0,0)" "
  locate d+10,7 : print " "chr$(24)chr$(25)" and PgUp, PgDn to move icon: <ENTER> to select:
<Esc> to quit "
  color 15,1
  locate d,2
  print "--------------------------------------------------------------------------------"

10
    color 31,9 : locate icon+d,c*10-2 : print chr$(4) : color 15,1
    if icon <> 1 then
      locate icon+d-1,18+(c-2)*10 : print " "
    end if
    if icon <> 9 then
      locate icon+d+1,18+(c-2)*10 : print " "
    end if
    if dim2 < 6 then
      x = dim2
    else
      x = 6
    end if
    for cols = 1 to x                    ' *** PRINTING THE TITLES
      locate d-1,(cols+1)*10-1 : print using "\       \"; file$(0,cols+iconrh)
    next cols
    if dim1 < 9 then
      y = dim1
    else
      y = 9
    end if
    for lines = 1 to y
      locate d+lines,4 : print lines+t
      locate d+lines,9 : print using "\      \"; file$(lines+t,0)
      for cols = 1 to x
        locate d+lines,(cols+1)*10-1 : print using "\      \";file$(lines+t,cols+iconrh)
      next cols
    next lines


  do                     ' *** TO READ THE KEYBOARD ARROWS
    x$=inkey$
  loop until len(x$)>0
```

```
do until asc(mid$(x$,1,1)) = 13              ' *** RETURN TO EXIT EDIT
  if len(x$) = 2 then                        ' *** ALL CTL CHARS ARE TWO CHARS
    x = asc(mid$(x$,2,1))
    select case x                          ' *** TO PICK REQ ACTION
      case = 72                            ' *** UP ARROW
        if icon > 1 then
          decr icon
        elseif t <> 0 then
          decr t
        end if
      case = 80                                                  ' *** DOWN ARROW
        if (icon < 9) and (icon < dim1) then
          incr icon
        elseif t <> dim1-9 and icon < dim1 then
          incr t
        end if
      case = 71                                                  ' *** HOME
        t = 0
      case = 75                                                  '*** LEFT ARROW
        if c > 2 then
          locate icon+d,c*10-2 : print " "
          decr c
        elseif iconrh > 0 then
          decr iconrh

        end if

      case = 77                          '*** RIGHT ARROW
        if (c < 7) and (c-1 < dim2) then
          locate icon+d,c*10-2 : print " "
          incr c
        elseif dim2 > iconrh+c-1 then
          incr iconrh
        end if
      case = 79                                                  ' *** END
        if dim1 > 9 then
          t = dim1-9
        end if
     case = 73                                                  ' *** PgUp
        if t > 9 then
          t = t - 8
        else
          t = 0
        end if
      case = 81                                                  ' *** PgDn
        if t < dim1-16  then
          t = t + 8
        elseif dim1 > 9 then
          t = dim1-9
        end if
      case else
        call failsound
    end select
    goto 10
  end if
  if len(x$) = 1 and asc(x$) = 27 then          ' 27 IS <Esc>
```

```
      exit loop
    else
      goto 10                              ' *** TO COPE WITH ORDINARY
    end if
    goto 10                              ' *** LETTERS BEING PRESSED

  loop

  if ma$(12,1) = "68" then
    if asc(mid$(x$,1,1)) = 13 then
      call fill (15,1,e+5,50,27,4)              ' *** TO CREATE THE EDITOR WINDOW
      call boxsing (15,1,e+5,50,27,4)
      color 14,9
      flag = 1
      locate e+5,60 : print " EDITOR "
      locate e+5,55 : print " <Enter> to exit "
      color 15,4
      locate d+icon,c*10-1 : print using "\      \";file$(icon+t,iconrh+c-1)
      color 15,1
      locate e+7,52 : print file$(0,iconrh+c-1) : file$(icon+t,iconrh+c-1) =
FNindata$(e+7,65,9)
      call fill(0,3,e+5,50,27,4)
      goto 10
    end if
  end if
  if file$(1,0) = "User" then
    file$(11,1) = "0"
    file$(12,1) = "0"
  end if
end sub
```

M-INIT.PRO

```
'     program M-INIT.PRO

'     OBJECTIVES - initialise all control files

'     Edition 1.0   Version 1.0   9/2/89

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

' ***** SUBROUTINE BLOCK *****

sub Initialise
  shared dep = 20
  shared ctl$()
  shared ma$(), g$(), g%()
  shared u1$(), u2$(), m$()
  shared hold$(), s1$(), ac$()
  shared sy$(), job$()

  call ctlinit(ctl$())
  call MenuSysMgt(sy$())
  call fill(15,3,4,1,80,20)
end sub

sub ctlinit(ctl$(2))

  ctl$(0,0) = "Primary Control File"
  ctl$(1,0) = "User name"          '*** user data
  ctl$(2,0) = "Sec Lev"
  ctl$(3,0) = "Co. Name"
  ctl$(4,0) = "Bk"
  ctl$(5,0) = "Bk"
  ctl$(6,0) = "Bk"
  ctl$(7,0) = "Bk"
  ctl$(8,0) = "Bk"
  ctl$(9,0) = "Bk"
  ctl$(10,0) = "New project"       '*** project data
  ctl$(11,0) = "Project loaded"        '*** row down the F-PJT$() file
  ctl$(12,0) = "Data changed"
  ctl$(13,0) = "Total No. Pjts"
  ctl$(14,0) = "Restore"
  ctl$(15,0) = "Pjt Name"          '*** project name, 3 digits.
  ctl$(16,0) = "Pjt Extension"         '*** project extension, 3 digits.
  ctl$(17,0) = "Sim Number"        '*** simulation run
  ctl$(18,0) = "Bk"
  ctl$(19,0) = "Bk"
  ctl$(20,0) = "Menu Level"
  ctl$(21,0) = "Bk"
  ctl$(22,0) = "Bk"
  ctl$(23,0) = "Bk"
  ctl$(24,0) = "Bk"
  ctl$(25,0) = "Delay"
  ctl$(26,0) = "Bk"
  ctl$(27,0) = "Bk"
  ctl$(28,0) = "Bk"
  ctl$(29,0) = "Bk"
  ctl$(30,0) = "D. mean" '*** used when going for steady state.
```

```
  ctl$(31,0) = "D. Var"       '*** ditto.
  ctl$(32,0) = "Bk"
  ctl$(33,0) = "Bk"
  ctl$(34,0) = "Bk"
  ctl$(35,0) = "Bk"
  ctl$(36,0) = "Bk"
  ctl$(37,0) = "Bk"
  ctl$(38,0) = "Bk"
  ctl$(39,0) = "Bk"
  ctl$(40,0) = "Steady State"
  ctl$(41,0) = "Bk"
  ctl$(42,0) = "Bk"
  ctl$(43,0) = "Bk"
  ctl$(44,0) = "Bk"
  ctl$(45,0) = "Bk"
  ctl$(46,0) = "Bk"
  ctl$(47,0) = "Bk"
  ctl$(48,0) = "Bk"
  ctl$(49,0) = "Bk"
  ctl$(50,0) = "Bk"

  for i = 1 to 50
    ctl$(i,1) = ""
  next i
  ctl$(25,1) = "0.01"
end sub

sub graphic(g$(2))
  for i = 0 to 20
    for j = 0 to 2
      g$(i,j) =""
    next j
  next i
  g$(0,0)="Graphics and Snd Database"                                    ' ***
SCREEN COLOURS
  g$(1,0)="Title  F"  :  g$(2,0)="Title  B"  :  g$(3,0)="Title  S"
  g$(4,0)="Screen F"  :  g$(5,0)="Screen B"  :  g$(6,0)="Screen S"
  g$(7,0)="Menu   F"  :  g$(8,0)="Menu   B"  :  g$(9,0)="Menu   S"
  g$(10,0)="Ques'n F" :  g$(11,0)="Ques'n B" :  g$(12,0)="Ques'n S"
  g$(13,0)="Bk   " :  g$(14,0)="Bk   " :  g$(15,0)="Bk"

  g$(1,1)  = "14"     :  g$(2,1)  = "9"      :  g$(3,1)  = "8"
  g$(4,1)  = "15"     :  g$(5,1)  = "3"      :  g$(6,1)  = "8"
  g$(7,1)  = "15"     :  g$(8,1)  = "1"      :  g$(9,1)  = "8"
  g$(10,1) = "15"     :  g$(11,1) = "4"      :  g$(12,1) = "8"
  g$(13,1) = "00"     :  g$(14,1) = "00"     :  g$(15,1) = "00"

  g$(16,0)="In Snd  " :  g$(16,1)="1157"
  g$(17,0)="Fail Snd" :  g$(17,1)="100"
  g$(18,0)="Snd 1=on" :  g$(18,1)="1"

  g$(19,0)="Blank   " :  g$(19,1)="00"
  g$(20,0)="Blank   " :  g$(20,1)="00"
end sub

sub user(u1$(2))
  u1$(0,0) = "Permanent User Data File"                                  '*** CREATES AND
DUMPS THE USER FILE
```

```
  u1$(0,1) = "Usr No"
  u1$(0,2) = "Name"
  u1$(0,3) = "Pswd"
  u1$(0,4) = "Not Used"
  u1$(0,5) = "Not Used"
  u1$(0,6) = "Not Used"
  for i = 1 to 10
    for j = 1 to 6
      u1$(i,j) = "####"
    next j
  next i
end sub

sub access1(ac$(2))
  ac$(0,0) = "Access Levels to Subs and Files"                    '*** ON ACCESS TO SUBS
  ac$(0,1) = "Subs"                                               '*** AND FILES
  ac$(0,2) = "Sec Lev"
  ac$(0,3) = "Files"
  ac$(0,4) = "Sec Lev"
  for i = 1 to 40
    ac$(i,1) = "####"
    ac$(i,2) = "####"
    ac$(i,3) = "####"
    ac$(i,4) = "####"
  next i
end sub

sub userinf(u2$(2))
  u2$(0,0) = "Working User Information File"                      '*** ON LATEST ACCESS TO
  u2$(0,1) = "Usr Name"                                           '*** SOFTWARE AND FILES
  u2$(0,2) = "Entries"
  u2$(0,3) = "Date 1"  :  u2$(0,4) = "Time In"  :  u2$(0,5) = "Time Out"
  u2$(0,6) = "Date 2"  :  u2$(0,7) = "Time In"  :  u2$(0,8) = "Time Out"
  u2$(0,9) = "Date 3"  :  u2$(0,10) = "Time In" :  u2$(0,11) = "Time Out"
  u2$(0,12) = "Date 4" :  u2$(0,13) = "Time In" :  u2$(0,14) = "Time Out"
  for i = 1 to 10
    for j = 1 to 50
      u2$(i,j) = "####"
    next j
  next i
end sub

sub MenuSysMgt(sy$(2))
  sy$(0,0) = "System Files"                          '*** NAMES OF ALL SYSTEM FILES
  sy$(1,1) = "Primary Control"                               '*** SOFTWARE AND FILES
  sy$(2,1) = "Graphics"
  sy$(3,1) = "Sec Data
  sy$(4,1) = "M/c Info"
  sy$(5,1) = "Access Levels"
  sy$(6,1) = "User Data"
  sy$(7,1) = "M/c Nickname"
  sy$(8,1) = "Product Nicknme"
  sy$(9,1) = "Sub Ass Nicknme"
  sy$(10,1) = "Comp. Nicknme"
  for i = 11 to 20
    sy$(i,1) = "####"
  next i
  open "f-sys.inf" for output as #1
```

```
  for i = 0 to 20
    write #1, sy$(i,0)
    write #1, sy$(i,1)
  next i
  close #1
end sub

sub JobInit(job$(2))
  open "f-job.inf" for output as #1
  for i = 0 to 20
    for j = 0 to 6
      job$(i,j) = ""
      write #1, job$(i,j)
    next j
  next i
  close #1
end sub

sub swapPS(s1$(2),p1%(2))
  shared dep
  for i = 1 to dep
    s1$(i,1) = str$(p1%(i,1))
  next i
end sub

sub swapSP(s1$(2),p1%(2))
  shared dep
  for i = 1 to dep
    p1%(i,1) = val(s1$(i,1))
  next i
end sub
```

M-EUSR.PRO

M-VIEW.PRO

```
'     program M-VIEW.PRO

'     OBJECTIVES - for user to view the currently active project files

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"

call go

sub go
  locate 4,1 : print "m-view.p"
  $include "shared.pro"
  if z = 1 then
    call viewfilebox
    z = 5
    chain "m-ctl.pbc"
  else
    call failsound
    call scrclear
    locate 10,10 :print "WHOOPS!!! did not set z when chaining to m-sheet"
    fg$=input$(9)
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub viewfilebox
  $include "shared.pro"

  do until vieww$ = "Z"
    call fill(0,3,4,1,80,20)
    call shadow(0,3,5,45,32,7)
    call fill(14,1,5,45,32,7)
    color 14,1
    locate 6,47  : print "PROJECT INFORMATION"
    color 15,1
    locate 9,47  : print "Project name ..... ";
    color 4,7 : print " "; : print using "\ \";ctl$(15,1);
    print " " : color 15,1
    locate 10,47 : print "Project number ..... ";
    x$ = ctl$(16,1) : call toupper(x$)
    if left$(x$,1) = "I" then x$ = "Root"
    color 4,7 : print " "; : print using "\  \";x$; : color 15,1
    call shadow(0,3,5,5,32,19)
    call fill(14,1,5,5,32,19)
    y$ = ctl$(15,1)
    z$ = ctl$(16,1)
    color 14,1
    locate 6,10  : print "  PROJECT FILES  "
    d = 5
    color 15,1
    x$ = "1.  "+y$+"-pdt."+z$
```

```
      locate d+3,12  : print x$
    x$ = "2.  "+y$+"~bom."+z$
      locate d+4,12  : print x$
    x$ = "3.  "+y$+"~cal."+z$
      locate d+5,12  : print x$
    x$ = "4.  "+y$+"~job."+z$
      locate d+6,12  : print x$
    x$ = "5.  "+y$+"~mac."+z$
      locate d+7,12  : print x$
    x$ = "6.  "+y$+"~que."+z$
      locate d+8,12  : print x$
    x$ = "7.  "+y$+"~rsj."+z$
      locate d+9,12  : print x$
    x$ = "8.  "+y$+"~rsm."+z$
      locate d+10,12  : print x$
    x$ = "9.  "+y$+"~rsg."+z$
      locate d+11,12  : print x$
    x$ = "10. "+y$+"~crh."+z$
      locate d+12,12  : print x$
    x$ = "11. "+y$+"~shh."+z$
      locate d+13,12  : print x$
    x$ = "12. "+y$+"~lsy."+z$
      locate d+14,12  : print x$
    x$ = "13. "+y$+"~wrk."+z$
      locate d+15,12  : print x$
    x$ = "14. "+y$+"~lm1."+z$
      locate d+16,12  : print x$
    x$ = "15. "+y$+"~qm1."+z$
      locate d+17,12  : print x$

    color 14,1
    for i = 1 to 15
      locate d+2+i,11 : print str$(i)+"."
    next i
    color 15,1

    call shadow(0,3,15,45,32,7)
    call fill(14,1,15,45,32,7)
    locate 16,47 : print "Select a file to view"
    locate 17,47 : print "Input a number (z=exit):"
    call insound
    vieww$ = FNindata$(20,61,2) : call toupper(vieww$)
    vieww$ = FNtrun$(vieww$)
    do until (vieww$ = "Z") or (val(vieww$) > 0 and val(vieww$) < 16)
      call failsound : vieww$ = FNindata$(20,61,2) : call toupper(vieww$)
    loop
    if vieww$ = "Z" then
      exit loop
    else
      x = val(vieww$)
      call viewfile(x)
    end if
  loop
end sub

sub viewfile(x)
  $include "shared.pro"
```

```
    file1$ = ctl$(15,1)
    file2$ = ctl$(17,1)
    call fill(0,3,5,1,80,20)
    select case x
      case = 1
        call edarray(pdt$(),50,1,kl)
      case = 2
      case = 3
        call edarray(cal$(),104,8,flag)
        if flag = 1 then call recal
      case = 4
        call edarray(job$(),jobnum,15,kl)
      case = 5
      case = 6
        call edarray(queues$(),resources,4,kl)
      case = 7
        call edarray(rltj$(),30,22,kl)
      case = 8
        call edarray(rltm$(),49,resources,kl)
      case = 9
        call edarray(rltg$(),40,1,kl)
      case = 10
        call edarray(cr$(),8,5,kl)
      case = 11
        call edarray(sh$(),8,5,kl)
      case = 12
        call edarray(ls$(),36,25,kl)
      case = 13
        call edarray(work$(),jobnum,15,kl)
    end select
  end sub      '*** ctlsheetout
```

M-CREATE.PRO

```
'     program M-CREATE.PRO

'     OBJECTIVES - main control program

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include  "m-file.pro"

call go

sub go
  locate 4,1 : print "m-create"
  $include "shared.pro"
  if z = 1 then
    call newpjtsetup
    call pjtname
    call dumpho("f-ctl.inf",ctl$(),50,1)
    call dumpho("f-pjt.inf",pjt$(),15,15)

    z = 1
    chain "m-pjint.pbc"
  elseif z = 2 then
    call copymenu
    z = 1
    chain "m-ctl.pbc"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub NewPjtSetUp
  $include "shared.pro"

  ctl$(3,1) = "1"
  ctl$(13,1) = str$((val(ctl$(13,1))+1))
  ctl$(13,1) = FnTrun$(ctl$(13,1))
  ctl$(16,1) = "inf"
  ctl$(17,1) = "inf"
  call fill(0,3,4,1,80,22)
end sub

sub pjtname
  $include "shared.pro"

  call shadow(0,3,6,7,40,7)
  call fill(15,1,6,7,40,7)
  call shadow(0,3,6,51,21,19)
  call fill(15,1,6,51,21,19)
  color 14,1
  locate 7,53 : print "EXISTING PROJECTS"
  color 15,1
```

```
 locate 9,53
 if pjt$(1,1) = "" then print "      none"
 color 14,1

 for i = 1 to 15
   if pjt$(i,1) = "" then exit for
   locate 8+i,57 : print using "##";i
   color 15,1
   locate 8+i,62 : print pjt$(i,1)
   color 14,1
 next i
 color 14,1
 locate 7,19 : print "NEW PROJECT CODE"

 color 15,1
 locate 9,15 : Print " Input New Project code"
 locate 10,15 : print "       (Z = exit) "
 x = val(ctl$(13,1))

 do
   call insound
   z$ = FNindata$(11,25,3)
   call toupper(z$)
   if FNletter(z$) = 0 then
     call fill(0,4,17,12,31,4)
     call shadow(0,3,17,12,31,4)
     color 15,4
     locate 18,15 : print " LETTERS AND NUMBERS ONLY"
     locate 19,15 : print "    press <space bar>"
     call failsound : call spce : flag = 1
     call fill(15,3,17,12,34,5)
   elseif z$ <> "Z" then
     flag = 0
     for i = 1 to 15
       if pjt$(i,1) = "" then exit for
       if z$ = pjt$(i,1) then flag = 1
     next i
     if flag = 1 then
       call fill(0,4,17,15,25,4)
       call shadow(0,3,17,15,25,4)
       color 15,4
   locate 18,18 : print " CODE ALREADY USED"
       locate 19,18 : print " press <space bar>"
   call failsound : call spce
       call fill(15,3,17,14,27,5)
     end if
   elseif z$ = "Z" then
     flag = 0
   end if
 loop until flag = 0

 if z$ <> "Z" then
   pjt$(x,1) = z$
   ctl$(11,1) = FNtrun$(str$(x))
   ctl$(15,1) = z$
   call rblock
   pjt$(x,2) = "#"
   for i = 4 to 7
```

```
    pjt$(x,i) = "0"
  next i

  ctl$(10,1) = "0"          '*** NO LONGER A NEW PROJECT
  x$ = ctl$(11,1)
  x = val(x$)
  ctl$(11,1) = FnTrun$(ctl$(11,1))
  call WaitOn
  y$ = pjt$(x,1)

  color 14,1
  locate 7,19 : print "NEW PROJECT NAME"
  color 15,1
  locate 9,15 : Print " Input New Project name  " ;
  locate 8,42 : color 14,1 : print pjt$(x,1) : color 15,1
  locate 10,15 : print "                    "
  x$ = FNindata$(11,16,24) : pjt$(x,3) = x$

  call rblock

  d = 14
  call shadow(0,3,14,7,40,11)
  call fill(15,1,14,7,40,11)
  color 14,1
  locate d,18 : print " CREATING FILES"
  x = val(ctl$(25,1))
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~pdt.inf";
  open y$+"~pdt.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~pdt.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~bom.inf";
  open y$+"~bom.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~bom.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~cal.inf";
  open y$+"~cal.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~cal.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~job.inf";
  open y$+"~job.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~job.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~mac.inf";
  open y$+"~mac.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~mac.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~que.inf";
  open y$+"~que.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~que.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~rsj.inf";
  open y$+"~rsj.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~rsj.inf";
    incr d : delay x
  color  4,7 : locate d,12 : print y$+"~rsm.inf";
  open y$+"~rsm.inf" for output as #1 : close #1
  color 15,1 : locate d,12 : print y$+"~rsm.inf";
```

```
        incr d : delay x
      color  4,7 : locate d,12 : print y$+"~bns.inf";
      open y$+"~bns.inf" for output as #1 : close #1
      color 15,1 : locate d,12 : print y$+"~bns.inf";

      d = 14
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~rsg.inf";
      open y$+"~rsg.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~rsg.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~crh.inf";
      open y$+"~crh.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~crh.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~shh.inf";
      open y$+"~shh.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~shh.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~lsy.inf";
      open y$+"~lsy.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~lsy.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~wrk.inf";
      open y$+"~wrk.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~wrk.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~lm1.inf";
      open y$+"~lm1.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~lm1.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~qm1.inf";
      open y$+"~qm1.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~qm1.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~sqd.inf";
      open y$+"~sqd.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~sqd.inf";
        incr d : delay x
      color  4,7 : locate d,30 : print y$+"~sqr.inf";
      open y$+"~sqr.inf" for output as #1 : close #1
      color 15,1 : locate d,30 : print y$+"~sqr.inf";
      call rblock
      delay 0.5

    else
      ctl$(10,1) = "0"
      ctl$(13,1) = str$((val(ctl$(13,1))-1))
      ctl$(13,1) = FnTrun$(ctl$(13,1))
      ctl$(14,1) = "0"
      z = 3
      chain "m-ctl.pbc"
    end if
  end sub


  sub CopyMenu
    $include "shared.pro"
```

```
x = val(ctl$(11,1))

if val(ctl$(13,1))-1 > 0  then none = 1
c = 0
do
  select case c
    case = 0
      call fill(0,3,4,1,80,22)
      call attribs
  call menu("DATA INPUT MENU","A.  Project Data","B.  Jobs Pending",_
    "C.  Resource Profile",_
    "D.  Bill of Materials","E.  Calendar","",_
    "I.  Copy Existing Project",_
    "","","P.  Return to Previous Menu","","")
   x = val(ctl$(11,1))
   if none = 0 then
     locate 15,7 : color 7,1
     print "I.  Copy Exisiting Project"
   end if
      color 14,1 : locate 8,28 : print "defined" : color 15,1
      color 4,7
  locate  9,29 : print " No  "
  locate 10,29 : print " No  "
  locate 11,29 : print " No  "
  locate 12,29 : print " No  "
  locate 13,29 : print " No  "
  locate 9,29  : if val(pjt$(x,4)) > 0 then print " Yes "
  locate 10,29 : if val(pjt$(x,5)) > 0 then print " Yes "
  locate 11,29 : if val(pjt$(x,6)) > 0 then print " Yes "
  locate 12,29 : if val(pjt$(x,7)) > 0 then print " Yes "
  locate 13,29 : if val(pjt$(x,8)) > 0 then print " Yes "
      color 15,4
      locate 23,7 : print " Select an Item: " : call insound
      c$ = FNindata$(23,26,1)
      call toupper(c$)
      c = asc(c$)
  color 0,3 : locate 23,7 : print "                          "
    case = 65
  c = 0
  if none = 1 then
    x$ = FNcopy$
  end if
  if none = 0 or x$ = "B" then
    pjt$(x,4) = "1"
    call fill(15,3,5,1,80,21)
    call shadow(0,3,5,15,48,19)
    call fill(15,1,5,15,48,19)
    call box(15,1,5,16,46,19)
    color 14,1
    locate 5,29 : print " PROJECT DATA INPUT "
    color 15,1
    d = 5

    locate d+2,19 : print "Global set up time:"
    locate d+4,19 : print "Global batch Size:"
    x$ = Fnindata$(d+2,45,3)
    do until val(x$) > 0 or x$ = "0"
      call failsound : x$ = Fnindata$(d+2,45,3)
```

```
    loop
    pdt$(1,1) = x$

    x$ = Fnindata$(d+4,45,3)
    do until val(x$) > 0
      call failsound : x$ = Fnindata$(d+4,45,3)
    loop
    pdt$(15,1) = x$
  elseif x$ = "A" then
    z = 9
    pjt$(x,4) = "1"
    chain "m-load.pbc"
  else
    c = 0
  end if
    case = 66
      c = 0
      if none = 0 then
        call fill(0,3,4,1,80,21)
    z = 1
    pjt$(x,5) = "1"
    chain "m-job.pbc"
  else
    x$ = FNcopy$
    if x$ = "A" then
      pjt$(x,5) = "1"
      z = 12
      chain "m-load.pbc"
    elseif x$ = "B" then
      call fill(0,3,4,1,80,21)
      pjt$(x,5) = "1"
      z = 1
      chain "m-job.pbc"
    else
      c = 0
    end if
  end if
    case = 67
  flag = 0
  if none = 0 then
    flag = 1
  else
    x$ = FNcopy$
    if x$ = "A" then
      flag = 2
    elseif x$ = "B" then
      flag = 1
    else
      flag = 3
    end if
  end if

  if flag = 1 then
    call shadow(0,3,14,24,37,9)
    call fill(15,7,14,24,37,9)
    color 14,7
    locate 15,30 : print "    RESOURCE INPUT "
    color 4,7
```

```
    locate 17,27 : print "A. Global Input of Resource"
    locate 18,27 : print "B. Individual Input of Resource"
    locate 19,27 : print "Z. Return to Input Menu"
    locate 21,27 : print "     Select an option: "
    call insound
    x$ = FNtrun$(FnIndata$(21,50,1)) : call toupper(x$)
    do until (x$ = "A" or x$ = "B") or X$ = "Z"
      call failsound
      x$ = FNtrun$(FnIndata$(21,50,1)) : call toupper(x$)
    loop
    if x$ = "A" then
      pjt$(x,6) = "1"
      call fill(0,3,4,1,80,21)
      z = 2
      locate 4,1 : print "m-res.pr"
      chain "m-res.pbc"
      call fill(0,3,4,1,80,21)
    elseif x$ = "B" then
      pjt$(x,6) = "1"
      call fill(0,3,4,1,80,21)
      z = 1
      chain "m-res.pbc"
    else
      c = 0
      exit select
    end if
  elseif flag = 2 then
    z = 13
    pjt$(x,6) = "1"
    chain "m-load.pbc"
  elseif flag = 3 then
    c = 0
    exit select
  end if

    case = 68
      c = 0
      if none = 0 then
        call fill(0,3,4,1,80,21)
    pjt$(x,7) = "1"
    z = 3
    chain "m-bom.pbc"
  else
    x$ = FNcopy$
    if x$ = "A" then
      pjt$(x,7) = "1"
      z = 10
      locate 4,1 : print "m-load.p"
      chain "m-load.pbc"
    elseif x$ = "B" then
      call fill(0,3,4,1,80,21)
      pjt$(x,7) = "1"
      z = 3
      locate 4,1 : print "m-bom.pr"
      chain "m-bom.pbc"
    else
      c = 0
      exit select
```

```
      end if
    end if
      case = 69
    c = 0
    if none = 0 then
      pjt$(x,8) = "1"
      call fill(0,3,4,1,80,21)
      z = 1
      chain "m-cal.pbc"
    else
      x$ = FNcopy$
      if x$ = "A" then
        z = 11
        pjt$(x,8) = "1"
        chain "m-load.pbc"
      elseif x$ = "B" then
        call fill(0,3,4,1,80,21)
        z = 1
        pjt$(x,8) = "1"
        chain "m-cal.pbc"
      else
        c = 0
        exit select
      end if
    end if
      case = 73
    if none = 0 then
      c = 101
    else
      for i = 4 to 8
        pjt$(x,i) = "1"
      next i
      z = 8
      chain "m-load.pbc"
    end if
      case = 80
          x = val(ctl$(11,1))
    y$ = pjt$(x,1)
    call waiton
    call dumpho("f-pjt.inf",pjt$(),15,15)
    call waitoff
    z = 3
    chain "m-ctl.pbc"
      case else
        call failsound
    call shadow(0,3,15,40,38,8)
    call fill(15,4,15,40,38,8)
    call box(15,4,15,41,36,8)
    locate 17,42 : print "       INVALID CHOICE     "
    locate 19,42 : print " PRESS <space bar> TO CONTINUE"
    call spce
'       call fill(0,3,5,1,80,20)
      color 15,1
      c = 0
  end select
 loop until abc = 123
 exit sub
end sub      '*** END OF COPY MENU
```

```
def FNcopy$
  call shadow(0,3,11,21,37,11)
  call fill(15,4,11,20,38,11)
  call box(15,4,11,21,36,11)
  color 14,4
  locate 13,24 : print "PREVIOUSLY DEFINED FILES EXIST"
  color 14,4
  locate 15,23
  print "  A. "; : color 15,4 : print "Copy from another project"
  locate 16,23
  color 14,4
  print "  B. "; : color 15,4 : print "Use existing file"
  locate 17,23
  color 14,4
  print "  Z. "; : color 15,4 : print "Return to input menu"
  locate 19,23 :
  print "        Select ";
  color 14,4 : print "A,B ";
  color 15,4 : print "or ";
  color 14,4 : print "Z"
  x$ = FnIndata$(19,47,1) : call toupper(x$)
  do until (left$(x$,1) = "A" or left$(x$,1) = "B") or (left$(x$,1) = "Z")
    call failsound
    x$ = FnIndata$(19,47,1) : call toupper(x$)
  loop
  FNcopy$ = left$(x$,1)
end def


sub attribs
  shared job$(), sim$(), jobnum, mac$(), bom$(), resources, pjt$(), cal$()
  shared ctl$()

  a = 40
  call shadow(0,3,6,5+a,32,15)
  call fill(14,1,6,5+a,32,15)
  for j = 0 to jobnum-1
    if job$(j+1,1) = "" then exit for
  next j
  for m = 0 to resources-1
    if mac$(m+1,1) = "" then exit for
  next m
  c = calnum
  p = 0

  x = val(ctl$(11,1))
  color 14,1
  locate 7,7+a  : print "CURRENT PROJECT STATUS"
  color 15,1
  locate 10,7+a : print "No. of Jobs .........";j
  locate 11,7+a : print "No. of Resources ....";m
  locate 12,7+a : print "No. of Products .....";  : print val(pjt$(x,10))
  locate 13,7+a : print "Calender (weeks) ....";c
  locate 15,7+a : print "No. of sim. runs ....";val(pjt$(val(ctl$(11,1)),2))
end sub
```

M-BOM.PRO

```
'     program M-BOM.PRO

'     CONTENTS - Main Control Prog For BoM Processing

'     Edition 1.0   Version 1.0   24/5/89

'     Manufacturing Simulation and Analysis

'     Copyright  S Hurley  and  Dr J Driscoll

$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"
$SEGMENT

call go

sub go
  locate 4,1 : print "m-bom.pr"
  $include "shared.pro"

  if z = 1 then
    DoBom = 1
    call dbbom
  elseif z = 2 then
    call dbbomext
    z = 1
    chain "m-edit.pbc"
  elseif z = 3 then
    DoBom = 1
    call dbbom
  elseif z = 4 then
    call dbbomext
    z = 2
    chain "m-create.pbc"
  elseif z = 5 then
    call runbom
    z = 1
    chain "m-edit.pbc"
  else
    call scrclear: locate 10,10
    print "WHOOPS with z going in m-bom"
    cv$=input$(99)
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub RunBom
  shared BomHo$(), mac$(), bomfile$(), prod$(), hold1$(), pjt$(), ctl$()
  do
    call fill(0,3,4,1,80,22)
    call fill(15,1,9,24,32,11)
    call box(15,1,9,25,30,11)
    call shadow(0,3,9,24,32,11)
    locate 9,32 : color 14,1 : print " BoM Processing " : color 15,1
    locate 11,34 : print "I = Input BoM"
    locate 12,34 : print "V = View BoM"
```

```
    locate 13,34 : print "D = Delete BoM"
    locate 15,28 : print "Input choice (z = return)"
    do
    DoBom$ = FNindata$(17,39,1) : call toupper(dobom$)

    if left$(DoBom$,1) = "I" then
      DoBom = 1
      call DbBom
    elseif left$(DoBom$,1) = "V" then
      DoBom = 1
      call DbView
      call fill(0,3,4,1,80,21)
    elseif left$(DoBom$,1) = "D" then
      DoBom = 1
      call DbDel
      erase bomfile
      erase prod
    elseif left$(DoBom$,1) = "Z" then
      exit loop
    else
      call failsound
    end if
    loop until DoBom = 1
  loop until left$(DoBom$,1) = "Z"
end sub


sub DbBom
  shared Bomho$(), hold1$(), mac$(), bom$(), bom, pjt$(), ctl$()

  for i = 1 to 20
    for j = 1 to 64
      bomho$(i,j) = ""
    next j
  next i
  bom = 1
  do until bom$(count,1) = ""
    incr bom
  loop

  call CreateBom
  for k = 1 to 20
    if BomHo$(k,1) = "" then exit for
    for j = 1 to 64
      que$ = que$ + bomho$(k,j)
      call put5(que$)
    next j
    bom$(bom,1) = que$
    incr bom
    que$ = ""
  next k
  prod$ = bomho$(1,1)
  call WaitOn
  if z = 1 then z = 1
  if z = 3 then z = 3
  chain "m-bomq.pbc"
end sub
```

```
sub CreateBom
  shared BomHo$(), icon, down, level, bom
  shared count, count1, BomR, Higher$, mac$(), pjt$(), ctl$()
  icon = 1
  down = 1
  level = 0

  call fill(15,3,4,1,80,22)
  for i = 0 to 20
    for j = 0 to 64
      BomHo$(i,j) = ""
    next j
  next i

  bomho$(0,1) = "Higher"
  bomho$(0,2) = "Lower"
  bomho$(0,3) = "Quantity"
  bomho$(0,64) = "Address"
  for i = 4 to 61 step 3
    bomho$(0,i)   = "M/c"+str$(i\3)
    bomho$(0,i+1) = "Set U"+str$(i\3)
    bomho$(0,i+2) = "Pro T"+str$(i\3)
  next i

  if left$(x$,1) = "Z" then exit sub

  call fill(15,1,7,3,36,18)
  call box(15,1,7,4,34,18)
  call shadow(0,3,7,4,35,18)
  locate 8,12 : color 14,1 : print "PRODUCT DATA INPUT" : color 15,1
  locate 10,7 : print "Level: "; : color 14,1 : print "0" : color 15,1
  locate 11,7 : print "Product name?: ";
  locate 12,7 : print "(z = exit)"

  x$ = FNindata$(11,23,10) : call toupper(x$)
  if x$ = "Z" then exit sub
  x = val(ctl$(11,1))
  pjt$(x,7) = FNtrun$(str$(val(pjt$(x,7)) + 1))
  locate 11,22 : print "             "
  color 4,7
  locate 11,22 : print " ";x$;" "
  color 15,1
  locate 12,7 : print "          "
  locate 13,7 : print "Product Set Up"
  y$ = FNindata$(13,25,3)
  do until val(y$) > 0 or y$ = "0"
    call failsound
    y$ = FNindata$(13,25,3)
  loop
  locate 13,24 : color 15,1 : print "     "
  locate 13,26 : color 4,7  : print " ";y$;" "
  color 15,1
  bomho$(0,0) = y$

  BomHo$(1,1) = x$
  BomHo$(1,2) = "0"
  BomHo$(1,3) = "1"
```

```
count = 2
count1 = 1
BomR = 2
Higher$ = BomHo$(1,1)

color 14,1
locate 21,9 : print "press <ese> then <enter>
locate 22,9 : print "  to view machine list"
locate 23,9 : print "       (z = exit) "
color 15,1

for k = 1 to 20
  l = (k*3) -2
  if k >= 3 then
    l = 4
    locate 15,7  : print FNtrun$(str$(k-1));FNk$(k-1);" machine #:"
    color 4,7
    locate 15,25 : print " ";BomHo$(1,(k-1)*3+1);" "
    locate 16,25 : print " ";BomHo$(1,(k-1)*3+2);" "
    locate 17,25 : print " ";BomHo$(1,(k-1)*3+3);" "

    call fill(15,1,19,7,24,2)
  end if

  locate 14+l,7  : print FNtrun$(str$(k));FNk$(k);" machine #:"

  y$= FNindata$(14+l,26,4)              '*** MACHINE NUMBER
  do until (val(y$) > 0) or (y$ = "z" or y$ = "Z")
    if asc(y$) = 27 then
  color 14,1 : locate 14+l,25 : print " VIEW " : color 15,1
  call boxsing(15,1,7,4,34,18)
  call box(15,1,11,43,28,13)
  y$ = FNinmac$
  call boxsing(15,1,11,43,28,13)
  call box(15,1,7,4,34,18)
    else
  call failsound
  y$ = FNindata$(14+l,26,4)
    end if
  loop
  locate 14+l,25 : print "      "
  color 4,7 : locate 14+l,25 : print " ";y$;" " : color 15,1
  call toupper(y$)
  if y$ = "Z" then exit for
  locate 15+l,7 : print "Set Up Time?     "
  z$ = FNindata$(14+l+1,26,4)
  do until val(z$) > 0 or z$ = "0"
    call failsound
    z$ = FNindata$(14+l+1,26,4)
  loop
  call toupper(z$)
  locate 14+l+1,25 : print "      "
  color 4,7 : locate 15+l,25 : print " ";z$;" "
  color 15,1
  locate 16+l,7 : print "Processing time? "
  z1$ = FNindata$(16+l,26,4)
  do until val(z1$) > 0
    call failsound
```

```
      z1$ = FNindata$(16+l,26,4)
    loop
    call toupper(z1$)
    locate 16+l,25 : print "      "
    color 4,7 : locate 16+l,25 : print " ";z1$;" "
    color 15,1
    BomHo$(1,k*3+1) = y$
    BomHo$(1,k*3+2) = z$
    BomHo$(1,k*3+3) = z1$
  next k

  incr level
  call CreateBom1
  incr count1
  incr level
  do
    if bomho$(count,0) = "SA" then
      higher$ = bomho$(count,2)
      call CreateBom1
      incr count
    else
      incr count
      incr level
    end if
  loop until left$(bomho$(count,2),1) = ""
end sub

sub CreateBom1
  shared BomHo$(), level
  shared count, BomR, Higher$, count1, pjt$(), ctl$()

  call fill(0,3,4,1,80,22)
  call fill(15,1,5,3,36,20)
  call box(15,1,5,4,34,20)
  call shadow(0,3,5,3,36,20)

  locate 6,7 : print "Product name: ";
  color 14,1 : print BomHo$(1,1) : color 15,1
  locate 7,7 : print "Level: "; : color 14,1 : print level : color 15,1
  locate 8,7 : print "Higher Level Part: "; : color 14,1 : print higher$
  locate 10,12 : print "SUB ASSEMBLY INPUT"
  color 15,1

  for j = 1 to 19
    color 14,1
    locate 21,9 : print "press <ese> then <enter>
    locate 22,9 : print "  to view machine list"
    locate 23,9 : print "      (z = exit)"
    color 15,1
    locate 11,7 : print "Part name :
    locate 12,7 : print "(z = none)"
    x$ = FNindata$(11,22,10) : call toupper(x$)
    locate 11,21 : print "            "
    color 4,7 : locate 11,22 : print " ";x$;" " : color 15,1
    if x$ = "Z" then exit for
    locate 13,7 : print "Quantity : "
    y$ = FNindata$(13,24,4)
    do until val(y$) > 0
```

```
  call failsound
  y$ = FNindata$(13,24,4)
loop
BomHo$(BomR,3) = FNtrun$(y$)
locate 13,21 : print "            "
color 4,7 : locate 13,22 : print " ";BomHo$(BomR,3);" " : color 15,1
BomHo$(BomR,1) = Higher$
BomHo$(BomR,2) = x$
BomHo$(BomR,0) = "SA"

for k = 1 to 20
  l = (k*3) -2
  if k >= 3 then
l = 4
locate 15,7  : print FNtrun$(str$(k-1));FNk$(k-1);" machine #:"
    color 4,7
locate 15,25 : print " ";BomHo$(BomR,(k-1)*3+1);" "
locate 16,25 : print " ";BomHo$(BomR,(k-1)*3+2);" "
locate 17,25 : print " ";BomHo$(BomR,(k-1)*3+3);" "
color 15,1
call fill(15,1,19,7,24,2)
  end if

  locate 14+l,7  : print FNtrun$(str$(k));FNk$(k);" machine #:"

  y$= FNindata$(14+l,26,4)                '*** MACHINE NUMBER
  do until (val(y$) > 0) or (y$ = "z" or y$ = "Z")
if asc(y$) = 27 then
  color 14,1 : locate 14+l,25 : print " VIEW " : color 15,1
  call boxsing(15,1,5,4,34,19)
  call box(15,1,11,43,28,13)
  y$ = FNinmac$
  call boxsing(15,1,11,43,28,13)
  call box(15,1,5,4,34,19)
else
  call failsound
  y$ = FNindata$(14+l,26,4)
end if
  loop
  locate 14+l,25 : print "      "
  color 4,7 : locate 14+l,25 : print " ";y$;" " : color 15,1

  call toupper(y$)
  if y$ = "Z" then exit for

  locate 15+l,7 : print "Set Up Time?     "
  z$ = FNindata$(14+l+1,26,4)
  do until val(z$) > 0 or z$ = "0"
call failsound
z$ = FNindata$(14+l+1,26,4)
  loop
  call toupper(z$)
  locate 14+l+1,25 : print "      "
  color 4,7 : locate 15+l,25 : print " ";z$;" "
  color 15,1

  locate 16+l,7 : print "Processing time? "
```

```
    z1$ = FNindata$(16+l,26,4)
    do until val(z1$) > 0
   call failsound
   z1$ = FNindata$(16+l,26,4)
    loop

    call toupper(z1$)
    locate 16+l,25 : print "      "
    color 4,7 : locate 16+l,25 : print " ";z1$;" "
    color 15,1
    BomHo$(BomR,k*3+1) = y$
    BomHo$(BomR,k*3+2) = z$
    BomHo$(BomR,k*3+3) = z1$
  next k

  i = 0
  do                          '*** PUTS IN ADDRESS
    incr i
    x$ = BomHo$(i,count1)
  loop until x$ = higher$

  BomHo$(BomR,64) = BomHo$(i,4)
  incr BomR
  call box(15,1,5,4,34,20)
  call fill(15,1,11,6,31,12)
next j

color 14,1
locate 10,8 : print "COMPONANT INPUT (end node)"
color 15,1

for j = 1 to 19
  locate 11,7 : print "Part name :
  locate 12,7 : print "(z = none)"
  x$ = FNindata$(11,22,10) : call toupper(x$)
  locate 11,21 : print "           "
  call toupper(x$)
  if x$ = "Z" then
    locate 11,21
    print "         "
    exit for
  end if
  color 4,7 : locate 11,22 : print " ";x$;" " : color 15,1
  locate 13,7 : print "Quantity : "
  y$ = FNindata$(13,24,4)
  do until val(y$) > 0
    call failsound
    y$ = FNindata$(13,24,4)
  loop
  BomHo$(BomR,3) = FNtrun$(y$)
  locate 13,21 : print "           "
  color 4,7 : locate 13,22 : print " ";BomHo$(BomR,3);" " : color 15,1
  BomHo$(BomR,1) = Higher$
  BomHo$(BomR,2) = x$

  color 14,1
  locate 21,9 : print "press <ese> then <enter>
  locate 22,9 : print "  to view machine list"
```

```
locate 23,9 : print "     (z = exit)"
color 15,1

for k = 1 to 20
  l = (k*3) -2
  if k >= 3 then
l = 4
locate 15,7  : print FNtrun$(str$(k-1));FNk$(k-1);" machine #:"
    color 4,7
locate 15,25 : print " ";BomHo$(BomR,(k-1)*3+1);" "
locate 16,25 : print " ";BomHo$(BomR,(k-1)*3+2);" "
locate 17,25 : print " ";BomHo$(BomR,(k-1)*3+3);" "
    call fill(15,1,19,7,24,2)
  end if

  locate 14+l,7  : print FNtrun$(str$(k));FNk$(k);" machine #:"

  y$ = FNindata$(14+l,26,4)                '*** MACHINE NUMBER
  do until (val(y$) > 0) or (y$ = "z" or y$ = "Z")
if asc(y$) = 27 then
  color 14,1 : locate 14+l,25 : print " VIEW " : color 15,1
  call boxsing(15,1,5,4,34,19)
  call box(15,1,11,43,28,13)
  y$ = FNinmac$
  call boxsing(15,1,11,43,28,13)
  call box(15,1,5,4,34,19)
else
  call failsound
  y$ = FNindata$(14+l,26,4)
end if
  loop
  locate 14+l,25 : print "      "
  call toupper(y$)
  if y$ = "Z" then
locate 14+l,25 : print "   "
exit for
  end if
  color 4,7 : locate 14+l,25 : print " ";y$;" " : color 15,1

  locate 15+l,7 : print "Set Up Time?     "
   z$ = FNindata$(14+l+1,26,4)
  do until val(z$) > 0 or z$ = "0"
call failsound
z$ = FNindata$(14+l+1,26,4)
  loop
  call toupper(z$)
  locate 14+l+1,25 : print "      "
  color 4,7 : locate 15+l,25 : print " ";z$;" "
  color 15,1

  locate 16+l,7 : print "Processing time? "

  z1$ = FNindata$(16+l,26,4)
  do until val(z1$) > 0
call failsound
z1$ = FNindata$(16+l,26,4)
  loop
```

```
      call toupper(z1$)
      locate 16+l,25 : print "      "
      color 4,7 : locate 16+l,25 : print " ";z1$;" "
      color 15,1

      BomHo$(BomR,k*3+1) = y$
      BomHo$(BomR,k*3+2) = z$
      BomHo$(BomR,k*3+3) = z1$
    next k

    i = 0
    do
      incr i
      x$ = BomHo$(i,count1)
    loop until x$ = higher$
    BomHo$(BomR,64) = BomHo$(i,4)
    incr BomR
    call box(15,1,5,4,34,20)
    call fill(15,1,11,6,31,12)
  next j
end sub

def FNinmac$
  $include "shared.pro"
  shared icon, down
  if icon <> 1 then goto 123
  maxrow = 1
  do until mac$(maxrow,1) = ""
    if mac$(maxrow,1) <> "" then incr maxrow
    if maxrow = resources then
      incr maxrow
      exit loop
    end if
  loop
  decr maxrow

  call fill(15,1,11,42,30,13)
  call box(15,1,11,43,28,13)
  call shadow(0,3,11,42,30,13)
  color 14,1
  locate 12,48 : print "AVAILABLE MACHINES"
  locate 13,46 : print "mc #"
  locate 13,56 : print "machine name"
  locate 22,44 : print " Use ";chr$(24);chr$(25);" <enter> to select "
  color 15,1

  color 31,1 : locate 15+icon-down,44 : print chr$(4) : color 15,1
  do
    for j = 1 to 7
      locate 14+j,46 : print down+j-1
      locate 14+j,57 : print mac$(down+j-1,1)
    next j

123:
    do
      x$ = inkey$
    loop until len(x$) > 0
    if asc(mid$(x$,1,1)) = 13 then
```

```
      exit loop
    end if
    if len(x$) = 2 then
      x = asc(mid$(x$,2,1))
      select case x
        case = 79
          exit loop
        case = 80              '*** DOWN ARROW
          incr icon
          if icon = maxrow + 1 then
            decr icon
            call failsound
            exit select
          end if
          if icon < 8 then
            color 15,1
            color 31,1 : locate 14+icon,44 : print chr$(4)
            color 15,1 : locate 13+icon,44 : print " "
          elseif icon >= 8 then
            incr down
            locate 13+10,44 : print chr$(23)
          end if
        case = 72              '*** UP ARROW
          decr icon
          if icon > 0 and down = 1 then
            color 15,1
        color 31,1 : locate 14+icon,44 : print chr$(4)
            color 15,1 : locate 15+icon,44 : print " "
          elseif down > 1 then
            decr down
          elseif icon = 0 then
            incr icon
            call failsound
          end if
        case else
          call failsound
      end select
    end if
    x$ = ""
  loop until X$ = "Z" or x$ = "z"
  FNinmac$ = FNtrun$(str$(icon))
end def

def FNk$(no)
  select case no
    case 1
      FNk$ = "st"
    case 2
      FNk$ = "nd"
    case 3
      FNk$ = "rd"
    case else
      FNk$ = "th"
  end select
end def

sub DbView                              '*** LETS ONE VIEW A BoM
  shared BomHo$(), high$, prod$(), mac$(), bomfile$(), ctl$(), pjt$()
```

```
icon = 1
down = 1
call fill(0,3,4,1,80,20)
maxrow = 1
count = 1
for i = 1 to 200
  if bomfile$(i,1) = "" then exit for
  if mid$(bomfile$(i,1),6,1) = "0" then
    prod$(count,1) = left$(bomfile$(i,1),5)
    incr count
  end if
next i

do until prod$(maxrow,1) = ""
  if prod$(maxrow,1) <> "" then incr maxrow
  if maxrow = resources then
    incr maxrow
    exit loop
  end if
loop
decr maxrow                    '*** NO OF BOM's

if prod$(1,1) = "" then
  call fill(15,4,10,21,38,7)
  call box(15,4,10,22,36,7)
  call shadow(0,3,10,21,38,7)
  color 15,4
  locate 12,25 : print "      No BoM's defined"
  locate 13,25 : print "press <space bar> to continue"
  x$ = input$(1)
  exit sub
end if

call fill(0,3,4,1,80,22)
call shadow(0,3,6,7,40,8)
call fill(15,1,6,7,40,8)
call shadow(0,3,6,51,21,19)
call fill(15,1,6,51,21,19)
color 14,1
locate 7,54 : print "EXISTING BoM's"
color 15,1
locate 9,53
if prod$(1,1) = "" then print "      none"
color 14,1
do
  for i = 1 to 13
    if prod$(i,1) = "" then exit for
    color 14,1
    locate 8+i,57 : print using "##";i-1+down
    color 15,1
    locate 8+i,62 : print prod$(i-1+down,1)
    color 14,1
  next i
  color 31,1 : locate 9+icon-down,54 : print chr$(4) : color 15,1

  color 14,1
  locate 7,24 : print "Bom View"
```

```
color 15,1
locate 9,15
color 15,1 : print " Use ";
color 14,1 : print chr$(24);chr$(25);
color 15,1 : print " to move cursor"
locate 10,15
color 15,1 : print "Press ";
color 14,1 : print "<enter> ";
color 15,1 : print "to select"
locate 12,15
color 14,1 : print "     <esc>";
color 15,1 : print " to quit"

do
  x$ = inkey$
loop until len(x$) > 0
x$ = FNtrun$(x$)
if asc(x$) = 27 then
  exit loop
elseif asc(x$) = 13 then
  chosen$ = prod$(icon,1)

  for i = 1 to 200
    bomfile$(i,1) = ""
  next i
  count = 0
  open "bom.db" for input as #1
  input #1,x$
  do until left$(x$,5) = chosen$
    y = eof(1)
    if y = -1 then exit loop
    input #1, x$
  loop
  count = 1
  bomfile$(count,1) = x$
  incr count
  do until left$(x$,5) <> chosen$
    y = eof(1)
    if y = -1 then exit loop
    input #1,x$
    if mid$(x$,6,1) <> "0" then
      bomfile$(count,1) = x$
      incr count
    end if
  loop
  close #1
  for i = 1 to 20
for j = 1 to 64
      x$ = mid$(bomfile$(i,1),(j*5)-4,5)
      if x$ = "" then exit for
      bomho$(i,j) = x$
    next j
    if bomfile$(i+1,1) = "" then exit for
  next i
  call fill(0,3,4,1,80,22)
  call edarray(bomho$(),20,64,jk)
  for i = 1 to 20
```

```
    for j = 1 to 64
          bomho$(i,j) = ""
        next j
      next i
      exit sub
    end if

    if len(x$) = 2 then
      x = asc(mid$(x$,2,1))
      select case x
        case = 79
          exit loop
        case = 80              '*** DOWN ARROW
          incr icon
          if icon = maxrow + 1 then
            decr icon
            call failsound
            exit select
          end if
          if icon < 14 then
            color 15,1
            color 31,1 : locate 8+icon,54 : print chr$(4)
            color 15,1 : locate 7+icon,54 : print " "
          elseif icon >= 14 then
            incr down

'           locate 13+10,44 : print chr$(23)
          end if
        case = 72              '*** UP ARROW
          decr icon
          if icon > 0 and down = 1 then
            color 15,1
            color 31,1 : locate 8+icon,54 : print chr$(4)
            color 15,1 : locate 9+icon,54 : print " "
          elseif down > 1 then
            decr down
          elseif icon = 0 then
            incr icon
            call failsound
          end if
        case else
          call failsound
      end select
    end if
    x$ = ""
  loop until X$ = "Z" or x$ = "z"
  for i = 1 to 200
    bomfile$(i,1) = ""
  next i
  for i = 1 to 40
    prod$(i,1) = ""
  next i
  close #1
end sub

sub DbDel
  shared BomHo$(), high$, prod$(), mac$(), bomfile$(), pjt$(), ctl$()
```

```
open "Bom.db" for append as #1
close #1

open "bom.db" for input as #1
for i = 1 to 200
  y = eof(1)
  if y = -1 then exit for
  input #1, bomfile$(i,1)
next i
close #1

icon = 1
down = 1
call fill(0,3,4,1,80,20)
maxrow = 1
count = 1
for i = 1 to 200
  if bomfile$(i,1) = "" then exit for
  if mid$(bomfile$(i,1),6,1) = "0" then
    prod$(count,1) = left$(bomfile$(i,1),5)
    incr count
  end if
next i

do until prod$(maxrow,1) = ""
  if prod$(maxrow,1) <> "" then incr maxrow
  if maxrow = resources then
    incr maxrow
    exit loop
  end if
loop
decr maxrow                    '*** NO OF BOM's

if prod$(1,1) = "" then
  call fill(15,4,10,21,38,7)
  call box(15,4,10,22,36,7)
  call shadow(0,3,10,21,38,7)
  color 15,4
  locate 12,25 : print "     No BoM's to delete"
  locate 13,25 : print "press <space bar> to continue"
  x$ = input$(1)
  exit sub
end if

call fill(0,3,4,1,80,22)
call shadow(0,3,6,7,40,8)
call fill(15,1,6,7,40,8)
call shadow(0,3,6,51,21,19)
call fill(15,1,6,51,21,19)
color 14,1
locate 7,54 : print "EXISTING BoM's"
color 15,1
locate 9,53
if prod$(1,1) = "" then print "      none"
color 14,1
do
  for i = 1 to 13
    if prod$(i,1) = "" then exit for
```

```
  color 14,1
  locate 8+i,57 : print using "##";i-1+down
  color 15,1
  locate 8+i,62 : print prod$(i-1+down,1)
  color 14,1
next i
color 31,1 : locate 9+icon-down,54 : print chr$(4) : color 15,1

color 14,1
locate 7,22 : print "Bom Delete"

color 15,1
locate 9,15
color 15,1 : print " Use ";
color 14,1 : print chr$(24);chr$(25);
color 15,1 : print " to move cursor"
locate 10,15
color 15,1 : print "Press ";
color 14,1 : print "<enter> ";
color 15,1 : print "to select"
locate 12,15
color 14,1 : print "     <esc>";
color 15,1 : print " to quit"

do
  x$ = inkey$
loop until len(x$) > 0
x$ = FNtrun$(x$)
if asc(x$) = 27 then
  exit loop
elseif asc(x$) = 13 then
  call shadow(0,3,16,7,40,8)
  call fill(0,4,16,7,40,8)
  color 15,4
  locate 17,12 : print "ARE YOU SURE YOU WISH TO DELETE"
  locate 18,14 : print "THE PRODUCT DEFINITION FILE"
  x$ = FNtrun$(prod$(icon,1)) : y = len(x$) : color 4,7
  locate 19,25-(y2) : print " ";x$;" "
  color 15,4
  locate 21,21 : print "YES/NO?"
  x$ = FNindata$(21,29,3) : x$ = FNtrun$(x$) : call toupper(x$)
  do until left$(x$,1) = "Y" or left$(x$,1) = "N"
    call failsound : x$ = FNindata$(21,29,3) : call toupper(x$)
  loop
  if x$ = "N" then
    exit loop
  else
    open "bom.db" for output as #1
    count = 1
    do until left$(bomfile$(count,1),5) = prod$(icon,1)
      incr count
    loop
    if count <> 1 then
      for i = 1 to count-1
        write #1,bomfile$(icon,1)
      next i
    end if
    incr count
```

```
        do until mid$(bomfile$(count,1),6,1) = "0"
          if bomfile$(count,1) = "" then exit loop
          incr count
        loop
        for i = count to 200
          if bomfile$(i,1) = "" then exit for
          write #1, bomfile$(i,1)
        next i
        exit loop
      end if
    end if
    if len(x$) = 2 then
      x = asc(mid$(x$,2,1))
      select case x
        case = 79
          exit loop
        case = 80              '*** DOWN ARROW
          incr icon
          if icon = maxrow + 1 then
            decr icon
            call failsound
            exit select
          end if
          if icon < 14 then
            color 15,1
            color 31,1 : locate 8+icon,54 : print chr$(4)
            color 15,1 : locate 7+icon,54 : print " "
          elseif icon >= 14 then
            incr down
          end if
        case = 72              '*** UP ARROW
          decr icon
          if icon > 0 and down = 1 then
            color 15,1
            color 31,1 : locate 8+icon,54 : print chr$(4)
            color 15,1 : locate 9+icon,54 : print " "
          elseif down > 1 then
            decr down
          elseif icon = 0 then
            incr icon
            call failsound
          end if
        case else
          call failsound
      end select
    end if
    x$ = ""
  loop until X$ = "Z" or x$ = "z"
  for i = 1 to 200
    bomfile$(i,1) = ""
  next i
  for i = 1 to 40
    prod$(i,1) = ""
  next i
  close #1
end sub

sub dbbomext
```

```
  shared Bomho$(), hold1$(), mac$(), bom$(), bom, Z, pjt$(), ctl$()
    call WaitOff
    if BomHo$(1,1) = "" then
      call boxsing(15,1,7,4,34,17)
    else
      call boxsing(15,1,5,4,34,20)
    end if
    call fill(15,4,18,22,37,7)
    call box(15,4,18,23,35,7)
    call shadow(0,3,18,22,37,7)
    color 15,4
    locate 19,27 : print "Do you wish to input another"
    locate 20,27 : print " Product definition (Y/N)?   "
    color 15,1
    call insound
    ExitCBom$ = FnIndata$(22,38,3) : call toupper(ExitCBom$)
    do until left$(ExitCBom$,1) = "Y" or left$(ExitCBom$,1) = "N"
      call failsound
      ExitCBom$ = FnIndata$(22,38,3) : call toupper(ExitCBom$)
    loop
   if left$(ExitCBom$,1) = "Y" then
     z = 3
     chain "m-bom.pbc"
   else
     exit sub
   end if
end sub
```

M-BOMQ.PRO

```
'     program M-BOMQ.PRO

'     CONTENTS - creates the working queue

'     Edition 1.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright  S Hurley  and  Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-bomq.p"
  $include "shared.pro"
  if z = 1 then
    if bomho$(1,1) <> "" then
      call QueueFromBom(bomho$(1,1))
    end if
    z = 2

    chain "m-bom.pbc"
  elseif z = 3 then
    if bomho$(1,1) <> "" then
      call QueueFromBom(bomho$(1,1))
    end if
    z = 4
    chain "m-bom.pbc"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub QueueFromBom(prod$)                          '*** LETS ONE VIEW A BoM
  shared BomHo$(), que$, hold1$(), ctl$(), pjt$()

  file$ = ctl$(15,1)+"~sqd."+ctl$(16,1)
  open file$ for input as #2
  recds = 1
  do until eof(2)
    incr recds
    input #2,recds$
  loop
  close #2

  Q = 1
  rows = 0
  for i = 1 to 20                  '***FOR GETTING 2nd
    if BomHo$(i,1) = "" then exit for
    for j = 4 to 61 step 3
      if BomHo$(i,j) = "" then exit for
      x$ = FNtrun$(str$(recds))
      hold1$(Q,1) = x$               '***SETTING THE ROW NUMBER
```

```
      incr recds
      hold1$(Q,3) = BomHo$(1,1)          '***SET PRO$
      flag$ = "1"                '*** problem if two parts
      if j = 4 then              '*** visit the same machine
        for k = 2 to 20
      if BomHo$(k,64) = "" then exit for
      if i = 1 then
        flag$ = "0"
        exit for
      end if
      if BomHo$(i,2) = BomHo$(k,1) then
           flag$ = "0"                       '***SET INI$
           exit for
         end if
        next k
      else
        flag$ = "0"
      end if
      hold1$(Q,4) = flag$               '***IF 1 THEN A GATEWAY
      hold1$(Q,5) = BomHo$(i,j)             '***SET RES$
      if BomHo$(i,j+3) = "" or i = 1 then     '***SET SPr$
        hold1$(Q,6) = BomHo$(i,1)
      else
        hold1$(Q,6) = BomHo$(i,2)
      end if
      hold1$(Q,7) = "#"                       '***SET SRT$
      hold1$(Q,8) = str$(val(BomHo$(i,j+2)))              '***SET DUR$
'*** THE DURATION IS TO PRODUCE ONE UNIT

      hold1$(Q,9) = "#"                       '***SET FIN$

      if BomHo$(i,j+3) = "" then              '***SET SUC$
        if i = 1 then
          hold1$(Q,10) = "FinPr"
          hold1$(Q,2)  = "FinPr"
        else
      hold1$(Q,10) = BomHo$(i,64)
          hold1$(Q,2)  = ""
       end if
      else
        if i = 1 then
      hold1$(Q,10) = BomHo$(i,j+3)
          hold1$(Q,2)  = FNtrun$(str$(Recds))
        else
      hold1$(Q,10) = BomHo$(i,j+3)
          hold1$(Q,2)  = ""
        end if
      end if
      if i = 1 then
        hold1$(Q,11) = BomHo$(1,1)
      else
    hold1$(Q,11) = BomHo$(i,2)                  '***SET PAR$
      end if

      bomho$(1,2) = bomho$(1,1)
      count = 0
      if hold1$(q,4) = "1" then
    count = 0
```

```
      else
    if j > 4 then
      count = 1
    else
      for k = 1 to 20
        if bomho$(k,1) = "" then exit for
        if ((val(bomho$(i,j)) = val(bomho$(k,64)) and i <> k) and_
                              (fntrun$(bomho$(i,2)) = fntrun$(bomho$(k,1)))) then
          incr count
        end if
      next k
    end if
      end if
      bomho$(1,2) = "0"
      x$ = FNtrun$(str$(count))
      hold1$(Q,12) = x$
      hold1$(Q,13) = "0"
      hold1$(Q,14) = "####"
      hold1$(Q,15) = "0"
      hold1$(Q,16) = "0"
      hold1$(Q,17) = "0"
      hold1$(Q,18) = "#"
      if i = 1 then
        hold1$(Q,19) = "1"
      else
        hold1$(Q,19) = BomHo$(i,3)
      end if
      hold1$(Q,20) = bomho$(i,j+1)
      hold1$(Q,21) = bomho$(0,0)
      incr Q
  next j
next i

 hold1$(0,0) = hold1$(1,3)
 hold1$(0,1) = "Num$"
 hold1$(0,2) = "SNo$"
 hold1$(0,3) = "Pro$"
 hold1$(0,4) = "Ini$"
 hold1$(0,5) = "Res$"
 hold1$(0,6) = "SPr$"
 hold1$(0,7) = "Srt$"
 hold1$(0,8) = "Dur$"
 hold1$(0,9) = "Fin$"
 hold1$(0,10) = "Suc$"
 hold1$(0,11) = "Par$"
 hold1$(0,12) = "Req$"
 hold1$(0,13) = "Obt$"
 hold1$(0,14) = "JNo$"
 hold1$(0,15) = "TPr$"
 hold1$(0,16) = "TBa$"
 hold1$(0,17) = "TQu$"
 hold1$(0,18) = "TTi$"
 hold1$(0,19) = "Cpt$"
 hold1$(0,20) = "SU1$"      /*** individual ones
 hold1$(0,21) = "SU2$"      /*** same for all

Q = 1 : AAA = 3
do until hold1$(Q,1) = ""
```

```
    if hold1$(Q,2) <> "" then
      incr Q
    else
      Q1 = 1
      if q1 = q then incr q1
      do until hold1$(Q1,1) = ""
    if ( hold1$(Q1,5) = hold1$(Q,10) and hold1$(Q1,11) = hold1$(Q,6) )_
        and ( hold1$(Q1,3) = hold1$(Q,3) and (AAA = 3) ) then
          hold1$(Q,2) = hold1$(Q1,1)
          exit loop
        end if
        incr Q1
    IF Q1 = Q THEN INCR Q1
      loop
       incr Q
    end if
  loop

  file$ = ctl$(15,1)+"~sqd."+ctl$(16,1)
  open file$ for append as #2
  row = 1
  do until left$(hold1$(row,10),1) = "F"
    incr row
  loop
  x% = FNcritpath%
  hold1$(row,10) = "F"+FNtrun$(str$(x%))

  for k = 1 to 35
    que$ = ""
    if hold1$(k,1) = "" then exit for
    for j = 1 to 21
      que$ = que$ + hold1$(k,j)
      call put5(que$)
    next j
    write #2, que$
  next k

  close #2
  file$ = ctl$(15,1)+"~bom."+ctl$(16,1)

  call dumpbom(file$)
  call dumpho("f-pjt.inf",pjt$(),15,15)

end sub


def FNcritpath%
  shared hold1$()

  if hold1$(1,2) = "FinPr" and val(hold1$(1,4)) = 1 then  '*** one line bom
    FNcritpath% = val(hold1$(1,8)) * val(hold1$(1,19))
    exit def
  end if

  row = 1
  for i = 1 to 35                '*** load up the initial ones and
    if hold1$(i,1) = "" then exit for '*** do the first test
    if val(hold1$(i,4)) = 1 then
```

```
      cp%(row,1) = val(hold1$(i,2))
      cp%(row,2) = val(hold1$(i,8)) * val(hold1$(i,19))
      cp%(row,3) = 1
      incr row
    end if
  next i

  do until a = b+1
    for i = 1 to 9            '*** clearing duplication
      up% = cp%(i,1)
      for j = i+1 to 10
    if cp%(i,1) = cp%(j,1) then
      if cp%(i,2) < cp%(j,2) then
        cp%(i,2) = cp%(j,2)
      end if
      cp%(i,3) = cp%(i,3) + cp%(j,3)
      cp%(j,1) = 0
      cp%(j,2) = 0
      cp%(j,3) = 0
    end if
      next j
    next i

    for i = 1 to 10                                    '*** step to next one
      up% = cp%(i,1) : ti% = cp%(i,2) : te% = cp%(i,3)
      row = 1
      do until cp%(i,1) = val(hold1$(row,1))
    incr row
      loop
      if te% = val(hold1$(row,12)) then
    cprow = i
    if hold1$(row,2) = "FinPr" then
      FNcritpath% = cp%(cprow,2) + (val(hold1$(row,8)) * val(hold1$(i,19)))
      exit def
    else
      cp%(cprow,1) = val(hold1$(row,2))
      cp%(cprow,2) = cp%(cprow,2)+ (val(hold1$(row,8)) * val(hold1$(i,19)))
      cp%(cprow,3) = 1
    end if
      end if
    next i
  loop
end def
```

M-CAL.PRO

```
'     program M-CAL.PRO

'     OBJECTIVES - calander input and processing program

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-ctl.pr"
  $include "shared.pro"

  if z = 1 then
    call initcal
    y$ = ctl$(15,1) : z$ = ctl$(17,1)
    call dumpfile(y$+"~cal."+z$,cal$(),calnum,8)
    call dumpho("f-pjt.inf",pjt$(),15,15)
    z = 2
    chain "m-create.pbc"
  elseif z = 2 then
    call fill(0,3,4,1,80,21)
    call edarray(cal$(),calnum,7,j)
    call recal
    y$ = ctl$(15,1) : z$ = ctl$(17,1)
    call dumpfile(y$+"~cal."+z$,cal$(),calnum,8)
    z = 1
    chain "m-edit.pbc"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub InitCal
  shared cal$(), calnum
  do
    for i = 1 to calnum
      cal$(i,8) = ""
    next i

    call fill(0,3,4,1,80,21)
    call fill(15,1,6,20,40,17)
    call shadow(0,3,6,20,40,17)
    x = 7
    color 14,1
    locate 7,22 : print " Input standard hours available on: "
    color 15,1
    locate 9,29  : print "MONDAY"
    locate 11,29 : print "TUESDAY"
    locate 13,29 : print "WEDNESDAY"
```

```
  locate 15,29 : print "THURSDAY"
  locate 17,29 : print "FRIDAY"
  locate 19,29 : print "SATURDAY"
  locate 21,29 : print "SUNDAY"
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,1) = cal$
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,2) = cal$
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,3) = cal$
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,4) = cal$
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,5) = cal$
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,6) = cal$
  incr x,2 : cal$ = FNindata$(x,40,3)
  gosub CalInput : cal$(1,7) = cal$
  for i = 1 to 7
    cal$(1,8) = str$(val(cal$(1,8)) + (val(cal$(1,i))*60))
  next i


  for i = 2 to calnum
    cal$(i,1) = cal$(1,1) : cal$(i,2) = cal$(1,2)
    cal$(i,3) = cal$(1,3) : cal$(i,4) = cal$(1,4)
    cal$(i,5) = cal$(1,5) : cal$(i,6) = cal$(1,6)
    cal$(i,7) = cal$(1,7) : cal$(i,8) = cal$(1,8)
  next i

  call fill(0,3,19,50,25,5)
  call fill(15,4,19,50,25,5)
  call shadow(0,3,19,50,25,5)
  color 15,4
  locate 20,53 : print "Satisfied (Y/N): "
  Sat$ = FnIndata$(20,71,3)
  sat$ = FNtrun$(sat$)
  sat$ = left$(sat$,1) : call toupper(sat$)
  do until sat$ = "Y" or sat$ = "N"
    call failsound
    Sat$ = left$(FnIndata$(20,71,3),1)
    sat$ = FNtrun$(sat$) : call toupper(sat$)
  loop
loop until left$(Sat$,1) = "Y"
color 15,4
locate 22,53 : print "Edit calender:"
Ed$ = left$(FnIndata$(22,71,3),1)
ed$ = FNtrun$(ed$) : call toupper(ed$)
do until sat$ = "Y" or sat$ = "N"
  call failsound
  ed$ = left$(FnIndata$(22,71,3),1)
  ed$ = FNtrun$(ed$) : call toupper(ed$)
loop

if left$(Ed$,1) = "Y" then
  call fill(0,3,4,1,80,21)
  call edarray$(cal$(),calnum,7,flag)
  if flag = 1 then call recal
```

```
  end if
  exit sub

CalInput:
  do until FnInNum%(0,24,cal$) = 0
    call failsound
    locate x,45 : print "0 to 24"
    cal$ = FNindata$(x,40,3)
    locate x,45 : print "       "
  loop
return

end sub
```

M-JOB.PRO

```
'     program M-JOB.PRO

'     OBJECTIVES - main job processing program

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-job.pr"
  $include "shared.pro"
  if z = 1 then
    call newjob
    y$ = ctl$(15,1)
    z$ = ctl$(16,1)
    call dumpho("f-pjt.inf",pjt$(),15,15)
    call pjtdump(y$+"~job."+z$,job$(),jobnum,15,val(pdt$(6,1)))
    z = 2                '*** GET BACK TO COPYMENU
    chain "m-create.pbc"
  elseif z = 2 then
    call jobmenu
    z = 1
    chain "m-edit.pbc"
  elseif z = 3 then          '*** from m-iter
    call newjob
    y$ = ctl$(15,1)
    z$ = ctl$(17,1)
    call pjtdump(y$+"~job."+z$,job$(),jobnum,15,val(pdt$(6,1)))
    z = 3
    chain "m-seq.pbc"
  else
    call scrclear
    locate 10,10 : print "PROBLEM WITH Z GOING INTO M-JOB"
    end
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub jobmenu
  shared job$(), jobnum, ctl$(), pdt$()
  d = 0
  do
    select case d
      case = 0
        call menu("JOB PROCESSING",_
          "A. Edit Job File","","B. Input New Jobs","","","",_
          "P. Previous Menu","","","","","")
    color 15,4 : locate 23,8 : print " Select an item: "
```

```
        call insound
      d$ = FNindata$(23,29,1)  : call toupper(d$)
      d = asc(d$)
          color 0,3 : locate 23,8 : print "                          "
          color 15,1
        case = 65
      d = 0
          call fill(0,3,4,1,80,22)
          call editjob
      y$ = ctl$(15,1)
      z$ = ctl$(17,1)
      call pjtdump(y$+"~job."+z$,job$(),jobnum,15,val(pdt$(6,1)))
      call fill(0,3,4,1,80,22)
        case = 66
      d = 0
          call fill(0,3,4,1,80,22)
      call newjob
      y$ = ctl$(15,1)
      z$ = ctl$(17,1)
      call pjtdump(y$+"~job."+z$,job$(),jobnum,15,val(pdt$(6,1)))
      call fill(0,3,4,1,80,22)
        case = 80
          exit select
        case else
      color 0,3 : locate 23,8 : print "                          "
      color 15,1
      call shadow(0,3,6,43,34,9)
      call fill(15,4,6,43,34,9)
      call box(15,4,6,44,32,9)
      locate 8,45  : print "        INVALID INPUT      "
      locate 10,45 : print "      PRESS <space bar>    "
      locate 12,45 : print "         TO CONTINUE       "
      call failsound
       x$ = input$(1)
      call fill(0,3,6,43,35,10)
      color 15,1
      d = 0
      end select
    loop until d = 80
end sub


sub newjob
    row = 0
    for i = 1 to jobnum
      if job$(i,1) = "" then exit for
      incr row
    next i
    job$(0,0) = FNtrun$(str$(row))
    call fill(0,7,5,3,75,18)
    call shadow(0,3,5,3,75,18)
    call fill(15,1,6,5,71,16)
    color 14,1
    locate 7,7 : print " JOBS PENDING INPUT SCREEN "
    locate 7,45 : print "Number of jobs ......."
    color 15,1
    locate 10,7 : print " Product name .........*"
    locate 11,7 : print " Job Number............."
```

```
locate 12,7 : print "                          "
locate 13,7 : print " Quantity.............*"
locate 14,7 : print " Transfer Batch Size...*"
locate 15,7 : print "                          "
locate 16,7 : print " Due Week .............*"
locate 17,7 : print " Due Day ..............*"
color 14,1
locate 19,7 : print "*  Data input required, Z = exit"
locate 20,7 : print "   Data consistancy checked after final input"
color 15,1

color 14,4 : locate 7,68 : print " ";row;" " : color 15,1
col = row - 2
do
ac = 1
for i = col+2 to col step -1
  if i > 0 then
    color 14,1 : locate 9,(35+(10*ac)) : print "Entry";i
    for j = 1 to 8
  color 15,1 : locate 9+j,(35+(10*ac)) : print "           "
    next j
    color 15,1
    locate 10,(10*ac)+35 : print using "\       \"; job$(i,1)
    locate 11,(10*ac)+35 : print using "\       \"; job$(i,4)
    locate 13,(10*ac)+35 : print using "\       \"; job$(i,3)
    locate 14,(10*ac)+35 : print using "\       \"; job$(i,10)
    locate 16,(10*ac)+35 : print using "\       \"; job$(i,5)
    locate 17,(10*ac)+35 : print using "\       \"; job$(i,6)

    incr ac
  end if
next i

  incr col
  incr row
  color 15,1
  call insound

  job$(row,0) = FNtrun$(str$(row))

  x$ = FNindata$(10,33,9) : call toupper(x$)         '*** NAME
  do until x$ <> " " and x$ <> ""
    call failsound : job$(row,1) = FNindata$(10,33,9)
  loop
  if x$ = "Z" then exit loop
  job$(row,1) = x$
  locate 10,32:print "          ":color 4,7
  locate 10,32:print " "; : print using "\        \";job$(row,1)
  job$(0,0) = FNtrun$(str$(val(job$(0,0)) + 1))
  x$ = job$(0,0)
  jnoho = 0
  for i = 1 to jobnum
    if job$(i,1) = "" then exit for
    if val(job$(i,4)) > jnoho then
  jnoho = val(job$(i,4))
    end if
  next i
  for i = 1 to jobnum
```

```
  if work$(i,1) = "" then exit for
  if val(work$(i,4)) > jnoho then
jnoho = val(work$(i,4))
  end if
next i
incr jnoho
job$(row,4) = FNtrun$(str$(jnoho))

locate 11,32:print "  ":color 4,7
locate 11,32 : print " "; : print using "\        \";job$(row,4)
job$(row,2) = "No"
x$ = FNindata$(13,33,3) : call toupper(x$)            '*** QUANTITY
do until val(x$) > 0 or x$ = "Z"
  call failsound : x$ = FNindata$(13,33,3)
loop
if x$ = "Z" then exit loop
job$(row,3) = x$
locate 13,32:print "          ":color 4,7
locate 13,32:print " "; : print using "\        \";job$(row,3)

x$ = FNindata$(14,33,3) : call toupper(x$)        '*** TRANSFER BATCH
do until val(x$) > 0 or x$ = "Z"
  call failsound : x$ = FNindata$(14,33,3)
loop
if x$ = "Z" then exit loop
job$(row,10) = x$
locate 14,32:print "          ":color 4,7
locate 14,32:print " "; : print using "\        \";job$(row,10)


x$ = FNindata$(16,33,3) : call toupper(x$)            '*** WEEK
do until (val(x$) > 0 and val(x$) < 105) or (x$ = "Z")
  call failsound : x$ = FNindata$(16,33,3)
loop
if x$ = "Z" then exit loop
job$(row,5) = x$

if val(job$(row,5)) < 10 then
  job$(row,5) = "00"+job$(row,5)
elseif val(job$(row,5)) < 100 then
  job$(row,5) = "0" + job$(row,5)
end if
locate 16,32:print "   ":color 4,7
locate 16,32 : print " "; : print using "\        \";job$(row,5)

x$ = FNindata$(17,33,3)              '*** DAY
do until (val(x$) > 0 and val(x$) < 8) or (x$ = "Z")
  call failsound : x$ = FNindata$(17,33,3)
loop
if x$ = "Z" then exit loop
job$(row,6) = x$
if val(job$(row,6)) < 10 then job$(row,6) = "0"+job$(row,6)
locate 17,32:print "   ":color 4,7
locate 17,32 : print " "; : print using "\        \";job$(row,6)

job$(row,7) = "00"
job$(row,8) = "00"
job$(row,9) = "1"
```

```
    call fill(15,1,10,32,12,9)
    locate 20,62 : print "   "
    color 14,4 : locate 7,68 : print " ";row;" " : color 15,1
  loop until row = jobnum

  if x$ = "Z" then
    job$(0,0) = str$(row-1)
    for i = 0 to 15
      job$(row,i) = ""
    next i
  end if
end sub

sub prncol1
  shared job$(), jobnum, col

  ac = 1
  for i = col+2 to col step -1
    if i > 0 then
      color 14,1 : locate 9,(35+(10*ac)) : print "Job ";i
      for j = 1 to 8
    color 15,1 : locate 9+j,(35+(10*ac)) : print "          "
    color 15,1 : locate 9+j,(35+(10*ac))
    print using "\       \"; job$(i,j)
      next j
      incr ac
    end if
  next i

end sub

sub EditJob
  shared job$(), jobnum, row, icon

  call fill(0,3,4,1,80,22)
  row = 0
  for i = 1 to jobnum
    if job$(i,1) = "" then exit for
    incr row
  next i
  jobrow = row
  do
    call fill(0,7,5,2,78,19)
    call fill(15,1,6,3,76,17)
    call fill(0,7,10,15,50,7)
    color 14,1
    locate 7,15 : print " JOBS PENDING EDITOR "
    color 14,7
    locate 12,18 : print "Number of jobs defined .....";jobrow;" "
    locate 14,18 : print "Which job to Edit? (Z=exit) ....."
    row$ = FNindata$(14,55,5) : call toupper(row$)
    if row$ = "Z" then exit loop
    color 14,0 : locate 20,6
    print " use ";chr$(24)chr$(25);" : <enter> to edit : <end> to exit ";
    call fill(15,1,10,15,50,7)
    row = val(row$)
    color 15,1
    locate  9,5 : print " Product .....  "
```

```
    color 4,7 : locate 9,32 : print " ";job$(row,1);" " : color 15,1
    locate 10,5 : print " Scheduled ..............  ";job$(row,2)
    locate 11,5 : print " Quantity ..............  ";job$(row,3)
    locate 12,5 : print " Job Number ............  ";job$(row,4)
    locate 13,5 : print " Due Week ..............  ";job$(row,5)
    locate 14,5 : print " Due Day ...............  ";job$(row,6)
    locate 15,5 : print " Due Hour ..............  ";job$(row,7)
    locate 16,5 : print " Due Minute ............  ";job$(row,8)
    locate 17,5 : print " Priority ..............  ";job$(row,9)

    icon = 1
    do
      do
       x$ = inkey$
      loop until len(x$) > 0
      if asc(mid$(x$,1,1)) = 13 then
    call changejob
      end if
      if len(x$) = 2 then
        x = asc(mid$(x$,2,1))
        select case x
          case = 79
            exit loop
          case = 80
            incr icon
            if icon < 10 then
              color 4,7  : locate icon+8,32 : print " ";job$(row,icon);" "
              color 15,1 : locate icon+7,32 : print " ";job$(row,icon-1);" "
            else
              decr icon
              call failsound
            end if
          case = 72
            decr icon
            if icon > 0 then
              color 4,7  : locate icon+8,32 : print " ";job$(row,icon);" "
              color 15,1 : locate icon+9,32 : print " ";job$(row,icon+1);" "
            else
              incr icon
              call failsound
            end if
          case = 90

          case else
            call failsound
        end select
      end if
      x$ = ""
    loop until X$ = "Z" or x$ = "z"
  loop until a = a+1
end sub

sub changejob
  shared icon, job$(), jobnum, row

    select case icon
      case = 1
        job$(row,1) = FNindata$(9,55,10)
```

```
  locate 9,32:print "                                                    "
  color 4,7:locate 9,32:print " ";job$(row,1);" "
case = 2
  job$(row,2) = FNindata$(10,55,10)
  locate 10,32:print "                                                   "
  color 4,7 : locate 10,32 : print " ";job$(row,2);" "
case = 3
  job$(row,3) = FNindata$(11,55,10)
  locate 11,32:print "                                                   "
  color 4,7: locate 11,32 : print " ";job$(row,3);" "
case = 4
  job$(row,4) = FNindata$(12,55,3)
  locate 12,32:print "                                                   "
  color 4,7: locate 12,32 : print " ";job$(row,4);" "
case = 5
  job$(row,5) = FNindata$(13,55,4)
  locate 13,32:print "                                                   "
  color 4,7: locate 13,32 : print " ";job$(row,5);" "
case = 6
  job$(row,6) = FNindata$(14,55,4)
  locate 14,32:print "                                                   "
  color 4,7: locate 14,32 : print " ";job$(row,6);" "
case = 7
  job$(row,7) = FNindata$(15,55,5)
  locate 15,32:print "                                                   "
  color 4,7:locate 15,32:print " ";job$(row,7);" "
case = 8
  job$(row,8)  = FNindata$(16,55,20)
  locate 16,32:print "                                                   "
  color 4,7:locate 16,32:print " ";job$(row,8);" "
case = 9
  job$(row,9)  = FNindata$(17,55,10)
  locate 17,32:print "                                                   "
  color 4,7:locate 17,32:print " ";job$(row,9);" "
case = 10
  job$(row,10)  = FNindata$(18,55,10)
  locate 18,32:print "                                                   "
  color 4,7:locate 18,32:print " ";job$(row,10);" "
case = 11
  job$(row,11)  = FNindata$(19,55,10)
  locate 19,32:print "                                                   "
  color 4,7:locate 19,32:print " ";job$(row,11);" "
case = 12
 job$(row,12)  = FNindata$(20,55,10)
  locate 20,32:print "                                                   "
  color 4,7:locate 20,32:print " ";job$(row,12);" "
case = 13
 job$(row,13)  = FNindata$(21,55,10)
  locate 21,32:print "                                                   "
  color 4,7:locate 21,32:print " ";job$(row,13);" "
case = 14
  job$(row,14)  = FNindata$(22,55,10)
  locate 22,32:print "                                                   "
  color 4,7:locate 22,32:print " ";job$(row,14);" "
case = 15
  job$(row,15)  = FNindata$(23,55,10)
  locate 23,32:print "                                                   "
  color 4,7:locate 23,32:print " ";job$(row,15);" "
```

```
      case else
        for i = 1 to 2
          call insound
          call failsound
        next i
    end select
end sub

sub Dumpjob(file$)
  shared job$(), resources
  open file$ for output as #5
  for i = 1 to resources
    if job$(i,1) = "" then exit for
    OutBuf$ = ""
    for j = 0 to 15
      OutBuf$ = OutBuf$ + job$(i,j) + "\"
    next j
    write #5, OutBuf$
  next i
  close #5
end sub


sub Restorejob(file$)
  shared job$(), resources
  open file$ for input as #4
  n = 1
  for i = 1 to resources
    n = 1
    if eof(4) then exit for
    input #4, x$
    for j = 0 to 15
      y = instr(n,x$,"\")
      job$(i,j) = mid$(x$,n,y-n)
      n = y+1
    next j
  next i
  close #4
end sub
```

M-RES.PRO

```
'     program M-RES.PRO

'     CONTENTS - all the resource processing routines

'     Edition 1.0   Version 1.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley and Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

'*** NOTE:
'***  Special uses of some of the fields in the sqd record for the FinPr.
'***  srt$  =  total set up time on the critical path (cp)
'***  fin$  =  total number of machines on the cp.
'***  The tft on the F is for the tft for no set up and lot size of one.
'***  jno$  =  the longest processing time (lpt) of a resource on the cp.
'***  The formula then for calculating the cp is
'***     cp = time for lfl +  (Process batch - 1) * lpt
'***
'***
'***  These are cleared up when the sqr database is generated.
'***
call go

sub go
  $include "shared.pro"

  if z = 1 then
    call inputres
    y$ = ctl$(15,1) : z$ = ctl$(17,1)
    call dumpres(y$+"~mac."+z$)
    call dumpho("f-pjt.inf",pjt$(),15,15)
    z = 2
    chain "m-create.pbc"
  elseif z = 2 then
    call globalinput
    y$ = ctl$(15,1) : z$ = ctl$(17,1)
    call dumpres(y$+"~mac."+z$)
    z = 2
    chain "m-create.pbc"
  elseif z = 3 then
    call resmenu
    y$ = ctl$(15,1) : z$ = ctl$(17,1)
    call dumpres(y$+"~mac."+z$)
    z = 1
    chain "m-edit.pbc"
  end if
end sub

' ***** COMMON SUBROUTINE BLOCK *****

sub resmenu
  shared mac$(), resources, sim$(), row, icon
```

```
  d = 0
  do
  select case d
    case = 0
      row = 0
      for i = 1 to resources
        if mac$(i,1) = "" then exit for
        incr row
      next i
      call scrclear
      call border
      call menu("UNIVERSAL TRANSFER MENU", "A. Input a Universal Transfer",_
        "","B. Global Input","","C. Edit a Universal Transfer","",_
        "","Z. Exit to previous menu","","","","")
      color 14,1
      locate 19,9 : print row;" UT's defined"
      color 15,1
      locate 22,7 : print " Select an option: "
      x$ = FNtrun$(FNindata$(22,27,3))
      d = asc(x$)
    case = 65
      d = 0
      call fill(0,3,4,1,80,22)
      call inputres
      call fill(0,3,4,1,80,22)
    case = 66
      d = 0
      call fill(0,3,4,1,80,22)
      call globalinput
      call fill(0,3,4,1,80,22)
    case = 67
      d = 0
      call fill(0,3,4,1,80,22)
      call editres
      call fill(0,3,4,1,80,22)
    case = 80
      exit select
    case = 90
      exit loop
    case else
      color 15,4
      locate 18,22 : print "INVALID CHOICE PRESS <space bar>";
      call failsound
      x$ = input$(1)
      d = 0
  end select
  loop until d = 80
end sub



sub EditRes
  shared mn$(), mac$(), resources, row, icon

  row = 0
  for i = 1 to resources
    if mac$(i,1) = "" then exit for
```

```
    incr row
  next i
  numrow = row
  do
    call fill(15,1,6,3,76,19)
    call fill(0,7,9,15,50,13)
    color 14,1
    locate 7,15 : print " UNIVERSAL TRANSFER EDITOR .....";_
    numrow;" U.T.'s Defined"
    color 4,7
    locate 10,21 : print "O P T I O N S"
    color 14,7
    locate 12,22 : print "A.  Global editor"
    locate 13,22 : print "#.  Input Specific Resource Number"
    locate 15,22 : print "Z.  Exit editor"
    color 4,7
    locate 18,25 : print "Select an option:" : row$ = FNindata$(18,44,3)
    if row$ = "Z" then
      exit sub
    elseif row$ <> "A" then
      call fill(0,1,9,15,50,13)
      color 14,1
      locate 7,15 : print " UNIVERSAL TRANSFER EDITOR ..... editing number .... ";row$
      color 14,0 : locate 24,6
      print " use ";chr$(24)chr$(25);" : <enter> to edit : <end> to exit ";
      call fill(15,1,10,15,50,7)
      row = val(row$)
      color 15,1
      locate  9,5 : print " Resource name .....  "
      color 4,7 : locate 9,32 : print " ";mac$(row,1);" " : color 15,1
      locate  8,5 : print "
      locate 10,5 : print " Resource code .....  ";mac$(row,2)
      locate 11,5 : print " Resource group .....  ";mac$(row,3)
      locate 12,5 : print " Simultaneous working ....  ";mac$(row,4)
      locate 13,5 : print " Performance factor .....  ";mac$(row,5)
      locate 14,5 : print " Operational Y/N .....  ";mac$(row,6)
      locate 15,5 : print " Set Up time .....  ";mac$(row,7)
      locate 16,5 : print " Start Resource Logic ....  ";mac$(row,8)
  /*** input queue to resource.
      locate 17,5 : print " Finish Resource Logic ...  ";mac$(row,9)
 /*** resource to output.
      locate 18,5 : print " Start Que Out Logic .....  ";mac$(row,10)
  /*** put in output que processing time.
      locate 19,5 : print " Finish Que Out Logic ....  ";mac$(row,11)
  /*** output que to input que or remove.
      locate 20,5 : print " Out Que Length .....  ";mac$(row,12)
      locate 21,5 : print " In Que Length .....  ";mac$(row,13)
      locate 22,5 : print " Compo Que Length .....  ";mac$(row,14)
      locate 23,5 : print " Dispatching Rule .....  ";mac$(row,15)

      icon = 1
      do
        do
          x$ = inkey$
        loop until len(x$) > 0
        if asc(mid$(x$,1,1)) = 13 then
    call scrclear
    call fill(15,1,6,3,76,19)
```

```
  call fill(0,7,9,15,50,13)
call changeres2
      end if
      if len(x$) = 2 then
        x = asc(mid$(x$,2,1))
        select case x
          case = 79
            exit loop
          case = 80
            incr icon
            if icon < 16 then
              color 4,7  : locate icon+8,32 : print " ";mac$(row,icon);" "
              color 15,1 : locate icon+7,32 : print " ";mac$(row,icon-1);" "
            else
              decr icon
              call failsound
            end if
          case = 72
            decr icon
            if icon > 0 then
              color 4,7  : locate icon+8,32 : print " ";mac$(row,icon);" "
              color 15,1 : locate icon+9,32 : print " ";mac$(row,icon+1);" "
            else
              incr icon
              call failsound
            end if
          case else
            call failsound
        end select
      end if
      x$ = ""
    loop until X$ = "Z" or x$ = "z"
  else
    row = 1
    call fill(0,7,9,12,58,13)
    color 15,7
    locate 10,30 : print "G L O B A L   E D I T O R"
    color 4,7
    locate 12,15 : print "A. Resource group"
    locate 13,15 : print "B. Simultaneous working"
    locate 14,15 : print "C. Performance factor"
    locate 15,15 : print "D. Operational Y/N"
    locate 16,15 : print "E. Set Up time"
    locate 17,15 : print "F. Start Resource Logic"
    locate 18,15 : print "G. Finish Resource Logic"
    locate 12,45 : print "H. Start Que Out Logic"
    locate 13,45 : print "I. Finish Que Out Logic"
    locate 14,45 : print "J. Out Que Length"
    locate 15,45 : print "K. In Que Length"
    locate 16,45 : print "L. Compo Que Length"
    locate 17,45 : print "M. Dispatching Rule"
    color 15,7
    locate 20,20 : print "Select a variable (Z=exit):"
    do
      x$ = FNindata$(20,50,3)
      if x$ = "Z" then exit loop
      icon = asc(FNtrun$(x$)) - 62
   call changeres2
```

```
        y$ = mac$(row,icon)
        for i = 1 to resources
          if mac$(i,1) = "" then exit for
          mac$(i,icon) = y$
        next i
      loop until x$ = "Z"
    end if
  loop until a = a+1
end sub



sub globalinput
  shared mac$(), resources, sim$(), row, icon

'*** row = the number of resources already defined

  row = 0
  for i = 1 to resources
    if mac$(i,1) = "" then exit for
    incr row
  next i
  call fill(0,3,4,1,80,21)
  call fill(0,7,6,4,74,15)
  call shadow(0,3,6,4,74,15)
  call fill(15,1,7,7,68,13)
  call shadow(0,7,7,7,68,13)
  color 14,1
  locate 8,18 : print " RESOURCE DEFINITION SCREEN (global)"
  locate 18,30 : print "Number of resources defined ....."
  color 4,7 : locate 18,65 : print " ";row;" " : color 15,1
  locate 12,10 : print " How many resources to globally define (z=exit):"
  todef$ = FNindata$(12,60,2)
  do until val(todef$) > 0 or (left$(todef$,1) = "Z" or left$(todef$,1) = "z")
    call failsound
    todef$ = FNindata$(12,60,2)
  loop
  if left$(todef$,1) = "Z" or left$(todef$,1) = "z" then exit sub

  incr row
  call fill(15,1,9,7,68,7)
  locate 10,15 : print "Individual Name Input"
  e = 1
  for j = row to resources
    if j = (val(todef$) + row) then exit for
    locate 13,17 : print "Name of Resource";j;" :"
    mac$(j,0) = FNtrun$(str$(j))
    z$ = FNindata$(13,40,10) : call toupper(z$)
    do
      flag = 0
      for k = 1 to resources
    if mac$(k,1) = "" then exit for
    if z$ = mac$(k,1) then
          flag = 1 : exit for
        end if
      next k
      if flag = 1 then
        call failsound
```

```
        call fill(15,4,15,20,40,2) : color 15,4
        locate 15,23 : print "NAME ALREADY USED FOR RESOURCE ";k
        locate 16,23 : print "  press <space bar> to continue"
    call spce
        call fill(15,1,15,20,40,2)
    z$ = FNindata$(13,40,10) : call toupper(z$)
      end if
    loop until flag = 0
    mac$(j,1) = z$
  next j
  call fill(0,3,4,1,80,21)
  call fill(15,1,6,4,74,19)
  call shadow(0,3,6,4,74,19)
  flag = val(todef$)
  call defaultres(flag)
  if flag <> 0 then
    for icon = 2 to 10
      call changeres2
    next icon

    for i = row+1 to row+val(todef$)-1
      for j = 2 to 15
    mac$(i,j) = mac$(row,j)
      next j
    next i
  end if
end sub


sub InputRes
  shared mac$(), resources, sim$(), row, icon
  do
    row = 0
    for i = 1 to resources
      if mac$(i,1) = "" then exit for
      incr row
    next i
    call fill(0,3,4,1,80,21)
    call fill(15,1,6,4,74,19)
    call shadow(0,3,6,4,74,19)
    incr row
    mac$(row,0) = fntrun$(str$(row))
    call border
    icon = 1 : call changeres2
    flag = 1

'*** flag is the number of resources to globally define

    call defaultres(flag)
    if flag <> 0 then
      for icon = 2 to 10
    call changeres2
    if icon = 1 and mac$(row,1) = "Z" then
      mac$(row,1) = ""
      decr row
      exit sub
    end if
      next icon
```

```
    end if
    call fill(0,4,18,40,35,5)
    call shadow(0,3,18,40,35,5)
    color 14,4
    locate 19,43 : print "Input another resource (Y/N)?"
    x$ = left$(FNindata$(21,55,3),1) : call toupper(x$)
    do until x$ = "Y" or x$ = "N"
      call failsound
      x$ = left$(FNindata$(21,55,3),1) : call toupper(x$)
    loop
'x$ = "N"
  loop until x$ = "N" or x$ = "n"

end sub

sub defaultres(flag)
  shared mac$(), resources, row

  call fill(0,7,9,15,50,13)
  call shadow(0,1,9,15,50,13)
  locate 7,7 : print " RESOURCE DEFINITION DEFAULT VALUES >>>>>>>>>"
  color 14,7 : locate 10,21 : print "Code..................... ";
  color 4,7 : print "0"
  color 14,7 : locate 11,21 : print "Group..................... ";
  color 4,7 : print "0"
  color 14,7 : locate 12,21 : print "Simultaneous Working.... ";
  color 4,7 : print "1"
  color 14,7 : locate 13,21 : print "Performance Factor...... ";
  color 4,7 : print "100"
  color 14,7 : locate 14,21 : print "Operational.............. ";
  color 4,7 : print "Yes"
  color 14,7 : locate 15,21 : print "Set Up Time.............. ";
  color 4,7 : print "0"
  color 14,7 : locate 16,21 : print "Launch Logic............. ";
  color 4,7 : print "Parallel"
  color 14,7 : locate 17,21 : print "Unload Logic............. ";
  color 4,7 : print "To Output Queue"
  color 14,7 : locate 18,21 : print "Queue Lengths........... ";
  color 4,7 : print "20"
  color 14,7 : locate 19,21 : print "Loading Rule............. ";
  color 4,7 : print "FCFS"

  color 15,1
  locate 23,7
  print " Do you wish to accept this default resource definition (Y/N): "
  x$ = FNindata$(23,71,3) : call toupper(x$) : x$ = left$(x$,1)
  do until x$ = "Y" or x$ = "N"
    call failsound : x$ = FNindata$(23,71,3)
    call toupper(x$) : x$ = left$(x$,1)
  loop

  if x$ = "Y" then
    for i = 1 to flag
      mac$(row,2) = "0"
      mac$(row,3) = "0"
      mac$(row,4) = "1"
      mac$(row,5) = "100"
      mac$(row,6) = "Yes"
```

```
      mac$(row,7) = "0"
      mac$(row,8) = "000102030607080910"
      mac$(row,9) = "010304"
      mac$(row,12) = "20"
      mac$(row,13) = "20"
      mac$(row,14) = "20"
      mac$(row,15) = "2"

      MAC$(ROW,10) = "01"
      MAC$(ROW,11) = "010203"

      incr row

      flag = 0
    next i
  end if
end sub

sub changeres2
  shared mac$(), icon, row, resources

    call fill(0,7,9,15,50,13)
    call shadow(0,1,9,15,50,13)
    color 14,1
    locate 7,7 : print " INPUT OPTIONS FOR DEFINING RESOURES >>>>>>>>>"
    select case icon
      case = 1
        color 4,7
        locate 10,21 : print "RESOURCE NAME"
        color 14,7
        locate 12,22 : print "Input a name (z=exit)
        color 4,7
        locate 18,25 : print "Name input:" : z$ = FNindata$(18,40,10)
        call toupper(z$)
        do
          flag = 0 : k = 0
          do
            incr k
            if z$ = mac$(k,1) then flag = 1
          loop until flag = 1 or k = row
          if flag = 1 then
            call failsound
            call fill(15,4,15,20,40,2) : color 15,4
            locate 15,23 : print "NAME ALREADY USED FOR RESOURCE ";k
            locate 16,23 : print "  press <space bar> to continue"
        call spce
            call fill(15,7,15,20,40,2)
        z$ = FNindata$(18,40,10) : call toupper(z$)
          end if
        loop until flag = 0
        mac$(row,1) = z$

        color 15,1 : locate 7,55 : print "              "
      case = 2
        color 4,7
    locate 10,21 : print "RESOURCE CODE AND GROUP"
        color 14,7
    locate 12,22 : print "Input the resource code and group."
```

```
locate 13,22 : print " "
locate 14,22 : print " "
locate 15,22 : print " "
    color 4,7
locate 18,25 : print "Input the resource code:"
locate 20,25 : print "Input the resource group:"
x$ = FNindata$(18,52,4)
mac$(row,2) = FNtrun$(str$(val(x$)))
color 4,7
x$ = FNindata$(20,52,4)
    mac$(row,3) = FNtrun$(str$(val(x$)))
    color 15,1 : locate 7,55 : print "                "
  case = 3
     color 15,1 : locate 7,55 : print "Simulataneous working "
    color 4,7
    locate 10,21 : print "SIMULTANEOUS WORKING"
    color 14,7
    locate 12,22 : print "Input the number of parts that the"
    locate 13,22 : print "resource can work on simultaneously."
    locate 14,22 : print "A number between 1 and 10."
    color 4,7
    locate 18,25 : print "Select an option:" : x$ = FNindata$(18,44,2)
    do until val(x$) > 0 and val(x$) < 11
      call failsound
      x$ = FNindata$(18,44,2)
    loop
    mac$(row,4) = FNtrun$(str$(val(x$)))
    color 15,1 : locate 7,55 : print "                    "
  case = 4
    color 4,7
    locate 10,21 : print "PERFORMANCE FACTOR"
    color 14,7
locate 12,22 : print "Examples:"
locate 14,22 : print "  100%  =  Operating at normal speed"
locate 15,22 : print "  50%   =  Running at half speed"
    color 4,7
    locate 18,20 : print "Input a Figure between 0 and 100:"
    x$ = FNindata$(18,55,3)
    do until (val(x$) > 0 and val(x$) < 101) or x$ = "0"
      call failsound
      x$ = FNindata$(18,55,3)
    loop
    mac$(row,5) = FNtrun$(x$)
    color 15,1 : locate 7,55 : print "                    "
  case = 5
    color 4,7
    locate 10,21 : print "OPERATIONAL"
color 14,7
locate 12,22 : print "Resource available for manufacture"
    color 4,7

color 14,7

locate 14,22 : print "A.  No"
locate 15,22 : print "B.  Yes"
    color 4,7
    locate 18,25 : print "Select an option:" : x$ = FNindata$(18,44,1)
    do until (x$ = "A" or x$ = "a") or (x$ = "B" or x$ = "b")
```

```
      call failsound
      x$ = FNindata$(18,44,1)
    loop
    mac$(row,6) = "Yes"
    if x$ = "A" or x$ = "a" then mac$(row,6) = "No"
    color 15,1 : locate 7,55 : print "            "
  case = 6
    color 4,7
    locate 10,21 : print "SET UP TIME"
    color 14,7
    locate 12,22 : print "Input a figure in minutes"
locate 14,22 : print ""
locate 15,22 : print ""
    color 4,7
    locate 18,25 : print "Input a number:" : x$ = FNindata$(18,44,3)
    do until x$ = "0" or val(x$) > 0
      call failsound
      x$ = FNindata$(18,44,3)
    loop
    mac$(row,7) = FNtrun$(x$)
    color 15,1 : locate 7,55 : print "              "
  case = 7
    color 4,7
    locate 10,21 : print "LAUNCH JOB LOGIC"
    color 14,7
locate 12,22 : print "A.  Serial:   Part then Component queue"
locate 13,22 : print "B.  Serial:   Component then Part queue"
locate 14,22 : print "C.  Parallel: Queues Combined "
locate 15,22 : print "  "
locate 16,22 : print ""
    color 4,7
    locate 18,25 : print "Select an option:"
    x$ = FNindata$(18,44,1) : call toupper(x$)
do until (x$ = "A" or x$ = "B") or (x$ = "C")
      call failsound
      x$ = FNindata$(18,44,1) : call toupper(x$)
loop

    if x$ = "A" then
  mac$(row,8) = "000102030407080900010203050708091O"
    elseif x$ = "B" then
  mac$(row,8) = "000102030507080900010203040708091O"
elseif x$ = "C" then
  mac$(row,8) = "00010203060708091O"
    end if

color 15,1 : locate 7,55 : print "                  "
    ' screen 0,1,1,1
  case = 8
    color 4,7
    locate 10,21 : print "UNLOAD RESOURCE LOGIC"
    color 14,7
locate 12,22 : print "A.  Move to output queue"
locate 13,22 : print "B.  Move to next reource input queue"
    color 4,7
    locate 18,25 : print "Select an option:"
    x$ = FNindata$(18,44,1) : call toupper(x$)
do until asc(x$) > 64 and asc(x$) < 67
```

```
        call failsound
        x$ = FNindata$(18,44,1) : call toupper(x$)
      loop

    if x$ = "A" then mac$(row,9) = "010304"
    if x$ = "B" then mac$(row,9) = "020304"

'*** need to sort out the logic of this inside the one that moves the one from
'*** the work centre to the output queue.

      color 15,1 : locate 7,55 : print "                    "
    case = 9
      color 4,7
    locate 10,21 : print "QUEUE LENGTHS"
      color 14,7
    locate 12,22 : print "Part queue length:"
    locate 14,22 : print "Component queue length:"
    locate 16,22 : print "Output queue length:"
    x$ = FNindata$(12,48,3)
    do until val(x$) > 0 or x$ = "0"
      call failsound : x$ = FNindata$(12,48,3)
    loop
    mac$(row,13) = x$
    color 14,7
    x$ = FNindata$(14,48,3)
    do until val(x$) > 0 or x$ = "0"
      call failsound : x$ = FNindata$(14,48,3)
    loop
    mac$(row,14) = x$
    color 14,7
    x$ = FNindata$(16,48,3)
    do until val(x$) > 0 or x$ = "0"
      call failsound : x$ = FNindata$(16,48,3)
    loop
    mac$(row,12) = x$
    color 15,1 : locate 7,55 : print "                    "
      case = 10
        color 4,7
    locate 10,21 : print "LOADING RULE"
        color 14,7
    locate 12,22 : print "A.  RND (Random)"
    locate 13,22 : print "B.  FCFS (First come first served)"
    locate 14,22 : print "C.  SPT (Shortest Processing Time)"
    locate 15,22 : print "D.  EDD (Earliest Due Date)"
    locate 16,22 : print "E.  CRL (Criticality List)"
        color 4,7
        locate 19,25 : print "Select an option:"
        x$ = FNindata$(19,44,1) : call toupper(x$)
        do until asc(x$) > 64 and asc(x$) < 70
          call failsound
          x$ = FNindata$(19,44,1) : call toupper(x$)
        loop

        if x$ = "A" then
          mac$(row,15) = "1"
        elseif x$ = "B" then
          mac$(row,15) = "2"
        elseif x$ = "C" then
```

```
          mac$(row,15) = "3"
        elseif x$ = "D" then
          mac$(row,15) = "4"
        elseif x$ = "E" then
       mac$(row,15) = "10"
        end if
    color 15,1 : locate 7,55 : print "                    "
    mac$(row,10) = "01"
    mac$(row,11) = "010203"
      case else
        for i = 1 to 2
          call insound
          call failsound
        next i
    end select
end sub
```

M-PJINT.PRO

```
'     program M-PJINT.PRO

'     OBJECTIVES -  This program will inialise all the project files
'           by putting the headers or row names in for all files.

'     Edition 2.0   Version 2.0   20/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-pjint."
  $include "shared.pro"

  if z = 1 then
    call pjtinit
    x$ = ctl$(15,1)
    y$ = "inf"
    call dump(x$,y$,val(pdt$(6,1)))
    call waitoff
    z = 3
    chain "m-ctl.pbc"
  else
    call failsound
    call scrclear
    locate 10,10 :print "WHOOPS!!! did not set z when chaining back to m-pjint"
    fg$=input$(9)
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub pjtinit
  shared pjt$(), resources, JobNum
  shared pdt$(), bomho$(), cal$(), job$(), mac$(), queues$()
  shared rltj$(), rltm$(), rltg$(), lm1$(), qm1$(), work$()
  shared ls$(), cr$(), sh$(), ls%(), sq$(), bnsch$(), sqnum, bnnum, calnum

'*****************************************************************************
'*** SET UP PROJECT DATA FILE
'***
  pdt$(0,0)  = "Project Control File"
  pdt$(1,0)  = "Global set up" :pdt$(1,1) = "0" 'the set up time for all UT's
  pdt$(2,0)  = "Model Runing " :pdt$(2,1) = "1" 'which model running.
'    1-normal, 2-qm1, 3-lm2, 4-ls
  pdt$(3,0)  = "LS from to   " :pdt$(3,1) = "000000"
  pdt$(4,0)  = "Perd's to sim" :pdt$(4,1) = "1"
  pdt$(5,0)  = "No Min In Sim" :pdt$(5,1) = "0"
  pdt$(6,0)  = "Period Count " :pdt$(6,1) = "000" 'start at one
  pdt$(7,0)  = "Buc len days " :pdt$(7,1) = "7"
```

```
  pdt$(8,0)  = "End-Now&     " :pdt$(8,1) = "0"
  pdt$(9,0)  = "Over-run     " :pdt$(9,1) = "0"
  pdt$(10,0) = "No Per To Sim" :pdt$(10,1) = "1" 'stop at end of period
  pdt$(11,0) = "CR Sel Method" :pdt$(11,1) = "2"
  pdt$(12,0) = "SH Sel Method" :pdt$(12,1) = "2"
  pdt$(13,0) = "CR Number    " :pdt$(13,1) = "0"
  pdt$(14,0) = "SH Number    " :pdt$(14,1) = "0"
  pdt$(15,0) = "Global TBatch" :pdt$(15,1) = "10"
  pdt$(16,0) = "TB method    " :pdt$(16,1) = "1"
  pdt$(17,0) = "Set Up Y/N   " :pdt$(17,1) = "1"
  pdt$(18,0) = "Set Up source" :pdt$(18,1) = "1"
  pdt$(19,0) = "Loading Rule " :pdt$(19,1) = "1"
  pdt$(20,0) = "LR method    " :pdt$(20,1) = "1"
  pdt$(21,0) = "Due D Offset " :pdt$(21,1) = "0" 'measured as a % of tft
  pdt$(22,0) = "Tardy Tolerae" :pdt$(22,1) = "1" 'measured in days
  pdt$(23,0) = "Unload Rule  " :pdt$(23,1) = "1"
  pdt$(24,0) = "UR Method    " :pdt$(24,1) = "1"
  pdt$(25,0) = "Screen Disply" :pdt$(25,1) = "1"
  pdt$(26,0) = "Var Monitor 1" :pdt$(26,1) = "5" 'learning system
  pdt$(27,0) = "Var Monitor 2" :pdt$(27,1) = "ALL"
  pdt$(28,0) = "Step to file " :pdt$(28,1) = "1"
  pdt$(29,0) = "Weight 1     " :pdt$(29,1) = "0.3333"
  pdt$(30,0) = "Weight 2     " :pdt$(30,1) = "0.3333"
  pdt$(31,0) = "Weight 3     " :pdt$(31,1) = "0.3333"
  pdt$(32,0) = "Weight 4     " :pdt$(32,1) = "0.3333"
  pdt$(33,0) = "CR Method LS " :pdt$(33,1) = "" 'which cr update in LS
  pdt$(34,0) = "DR Method LS " :pdt$(34,1) = "" 'which DR using
  pdt$(35,0) = "Dmp FileAfter" :pdt$(35,1) = "1" 'write to disk after period
  pdt$(36,0) = "LS up after  " :pdt$(36,1) = "6" 'update LS after period
  pdt$(37,0) = "Dmp Run After" :pdt$(37,1) = "1" 'dump to run after period
  pdt$(38,0) = "Jobs In every" :pdt$(38,1) = "2" 'allow job input every x buc.
                          '105 switches off - in m-iter
  pdt$(39,0) = "Per then dump" :pdt$(39,1) = "1" '
  pdt$(40,0) = "Clr Bn       " :pdt$(40,1) = "0" '1 means clear Crit List.
  pdt$(41,0) = "Due date off " :pdt$(41,1) = "0" 'Due Date offset
  pdt$(42,0) = "Late Toleranc" :pdt$(42,1) = "0" 'Lateness tolerance in hours
  pdt$(43,0) = "Start Week   " :pdt$(43,1) = "001"
  pdt$(44,0) = "Start Day    " :pdt$(44,1) = "01"
  pdt$(45,0) = "Start Hour   " :pdt$(45,1) = "00"
  pdt$(46,0) = "Start Minute " :pdt$(46,1) = "00"
  pdt$(47,0) = "Current Week " :pdt$(47,1) = "001"
  pdt$(48,0) = "Current Hour " :pdt$(48,1) = "01"
  pdt$(49,0) = "Current Day  " :pdt$(49,1) = "00"
  pdt$(50,0) = "Current Min  " :pdt$(50,1) = "00"

'*****************************************************************************
'*** SET UP CALANDER FILE
'***
  cal$(0,0) = "Calendar"
  cal$(0,1) = "Monday"
  cal$(0,2) = "Tuesday"
  cal$(0,3) = "Wednesday"
  cal$(0,4) = "Thursday"
  cal$(0,5) = "Friday"
  cal$(0,6) = "Saturday"
  cal$(0,7) = "Sunday"
  cal$(0,8) = "TOTAL"
```

```
  for i = 1 to calnum
    cal$(i,0) = "W" + FnTrun$(str$(i))
  next i

  for i = 1 to 8
    for j = 1 to calnum
      cal$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP JOB FILE
'***
  job$(0,0) = "O"
  job$(0,1) = "Product"
  job$(0,2) = "Sch'd"
  job$(0,3) = "Quan'y"
  job$(0,4) = "Job No."
  job$(0,5) = "Due Week"
  job$(0,6) = "Due Day"
  job$(0,7) = "Due Hour"
  job$(0,8) = "Due Min"
  job$(0,9) = "Priority"
  job$(0,10) = "Tr Batch"
  job$(0,11) = "Bk"
  job$(0,12) = "Bk"
  job$(0,13) = "Bk"
  job$(0,14) = "Bk"
  job$(0,15) = "Bk"

  for i = 1 to 15
    for j = 1 to JobNum
      job$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP RESOURCE/MACHINE FILE
'***
  mac$(0,0) = " Resource Definition File "
  mac$(0,1) = "Name"
  mac$(0,2) = "Code"
  mac$(0,3) = "Group"
  mac$(0,4) = "Sim Wkg"
  mac$(0,5) = "Perf Fac"
  mac$(0,6) = "Operatio"
  mac$(0,7) = "Set Up"
  mac$(0,8) = "Pik Jo L"
  mac$(0,9) = "Unload L"
  mac$(0,10) = "OutHol L"
  mac$(0,11) = "ToIn L"
  mac$(0,12) = "OutQ Len"
  mac$(0,13) = "InQu Len"
  mac$(0,14) = "ComQu Le"
  mac$(0,15) = "Dis Rule"

  for i = 1 to resources
    for j = 1 to 15
```

```
      mac$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP QUEUE STATUS STATISTICS
'***
  queues$(0,0) = "Queue Status"
  queues$(0,1) = "Component"
  queues$(0,2) = "Part      "
  queues$(0,3) = "Resource  "
  queues$(0,4) = "Output    "

  for i = 1 to resources
    queues$(i,0) = FNtrun$(str$(i))
    for j = 1 to 4
      queues$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP JOB RESULTS FILE
'***
  rltj$(0,0) = " JOB RESULTS FILE "
  rltj$(0,1) = "Product"
  rltj$(0,2) = "Job Numb"
  rltj$(0,3) = "TLT"      '*** theoretical lead time
  rltj$(0,4) = "Act Flow"
  rltj$(0,5) = "Tardy"        '*** negative is early
  rltj$(0,6) = "Set Up"
  rltj$(0,7) = "Pro Time"
  rltj$(0,8) = "Que Time"
  rltj$(0,9) = "Quantity"
  rltj$(0,10) = "Srt Time"
  rltj$(0,11) = "Fin Time"
  rltj$(0,12) = "Due Date"
  rltj$(0,13) = "Act-Tlt"     '*** TFT tardyness measure. Actual Flow - TLT
  rltj$(0,14) = "TB size"
  rltj$(0,15) = "Ave Set Up"
  rltj$(0,16) = "Bk"
  rltj$(0,17) = "Srt Time2"   '*** the time at the beginning of the peiord
                '*** it was launched in.
  rltj$(0,18) = "Act Ft 2"    '*** flow time from beginning of the week.
  rltj$(0,19) = "Bk"
  rltj$(0,20) = "Bk"
  rltj$(0,21) = "Bk"
  rltj$(0,22) = "Bk"

  for i = 1 to jobnum
    for j = 1 to 22
      rltj$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP RESOURCE RESULTS FILE
'***
  rltm$(0,0) = "Inidividual Resource Statistics"
```

```
rltm$(1,0) = "Resource"    '*** 1-8 in Sub Util, in m-simcom.pro
rltm$(2,0) = "Activity"
rltm$(3,0) = "Start"
rltm$(4,0) = "Set Up"
rltm$(5,0) = "Working"
rltm$(6,0) = "Idle"
rltm$(7,0) = "Blocked"
rltm$(8,0) = "No Part" '***
rltm$(9,0) = "Bk"
rltm$(10,0) = "McTiAva"
rltm$(11,0) = "Bk"
rltm$(12,0) = "Observ" '*** number of observations, 13-18 input que
rltm$(13,0) = "tim X pt"   '*** difference X processing time
rltm$(14,0) = "aver pt"    '*** running average
rltm$(15,0) = "max pt" '*** maximum processing time of a queue
rltm$(16,0) = "tim X len"  '*** difference X length of queue
rltm$(17,0) = "aver len"   '*** running average length
rltm$(18,0) = "max len"    '*** maximum length over period
rltm$(19,0) = "tim X pt"   '*** 19-24 componant queue
rltm$(20,0) = "aver pt"            '* ** 15-34 in sub working and working1,
rltm$(21,0) = "max pt"         '*** in m-simcom.pro
rltm$(22,0) = "tim X len"
rltm$(23,0) = "ave len"
rltm$(24,0) = "max len"
rltm$(25,0) = "tim X pt"   '*** 25-30 output queue
rltm$(26,0) = "aver pt"
rltm$(27,0) = "max pt"
rltm$(28,0) = "tim X len"
rltm$(29,0) = "ave len"
rltm$(30,0) = "max len"
rltm$(31,0) = "tim X pt"   '*** 31-36 total for whole system
rltm$(32,0) = "aver pt"
rltm$(33,0) = "max pt"
rltm$(34,0) = "time X len"
rltm$(35,0) = "aver len"
rltm$(36,0) = "max len"    '*** end of queuing statistics
rltm$(37,0) = "Bk"
rltm$(38,0) = "Bk"
rltm$(39,0) = "Util 1" '*** without set up.   35-36 in sub util
rltm$(40,0) = "Max UI"
rltm$(41,0) = "Util 2" '*** with set up       in m-simcom.pro
rltm$(42,0) = "Max UII"
rltm$(43,0) = "Bk"
rltm$(44,0) = "Pro Time"   '*** total time to complete all processing
rltm$(45,0) = "NumOfJob"   '*** to be calculated at the end of a period.
rltm$(46,0) = "Bk"
rltm$(47,0) = "Bk"
rltm$(48,0) = "Bk"
rltm$(49,0) = "Bk"

for i = 1 to resources
  rltm$(1,i) = FNtrun$(str$(i))
  rltm$(2,i) = "I"
  rltm$(49,i) = ""
next i
for i = 3 to 48
  for j = 1 to resources
    rltm$(i,j) = "0"
```

```
    next j
  next i

'*****************************************************************************
'*** SET UP GLOBAL RESULTS FILE
'***

  rltg$(0,0) = " GLOBAL RESULTS FILE "
  rltg$(1,0) = "Period"
  rltg$(2,0) = "NoJobOnTime"
  rltg$(3,0) = "NoJobEarly"
  rltg$(4,0) = "NoJobLate"
  rltg$(5,0) = "TotEarTime"
  rltg$(6,0) = "TptLatTime"
  rltg$(7,0) = "DelError"
  rltg$(8,0) = "TotFlowTime"
  rltg$(9,0) = "TheoFloTime"
  rltg$(10,0) = "ManfError"
  rltg$(11,0) = "TotTimeAvail"
  rltg$(12,0) = "TotMacTiAvail"
  rltg$(13,0) = "TotSetTime"
  rltg$(14,0) = "TotProcTime"
  rltg$(15,0) = "TotIdlTime"
  rltg$(16,0) = "Bk"
  rltg$(17,0) = "Bk"
  rltg$(18,0) = "TotRealFt"       '*** total ft from beginning of week
  rltg$(19,0) = "Bk"
  rltg$(20,0) = "QuToProTime"     '*** calc at end of quarter
  rltg$(21,0) = "QuNoOfJob"
  rltg$(22,0) = "NoOfResources"
  rltg$(23,0) = "Bk"
  rltg$(24,0) = "Bk"
  rltg$(25,0) = "AveFlow"
  rltg$(26,0) = "AveQProTime"
  rltg$(27,0) = "AveQNoJob"
  rltg$(28,0) = "Util 1"
  rltg$(29,0) = "Util 2"
  rltg$(30,0) = "Ave Late"
  rltg$(31,0) = "Ave Early"
  rltg$(32,0) = "AveOnTime"
  rltg$(33,0) = "FreqTardyJob"
  rltg$(34,0) = "Bk"
  rltg$(35,0) = "Bk"
  rltg$(36,0) = "Bk"
  rltg$(37,0) = "Bk"
  rltg$(38,0) = "Bk"
  rltg$(39,0) = "Bk"
  rltg$(40,0) = "Bk"

  for j = 1 to 40
    rltg$(j,1) = "0"
  next j


'*****************************************************************************
'*** SET UP FOR CR SELECTION HISORY
'***
'*** the five periods are in the five columns
```

```
'***
  cr$(0,0) = " CR selection history"
  cr$(1,0) = "LS choic"          ' LEARNING SYSTEM CHOICE, No 1-4
  cr$(2,0) = "Selectio"
  cr$(3,0) = "CR-M"
  cr$(4,0) = "CR-R"
  cr$(5,0) = "CR-LM1"
  cr$(6,0) = "CR-LM2"
  cr$(7,0) = "CR-QM1"
  cr$(8,0) = "Bk"

  for i = 1 to 8
    cr$(i,1) = ""
  next i

'*****************************************************************************
'*** SET UP FOR SH SELECTION HISORY
'***
'*** the five periods are in the five columns
'***
  sh$(0,0) = " SH selection history"
  sh$(1,0) = "LS choic"          ' LEARNING SYSTEM CHOICE, No 1-4
  sh$(2,0) = "Selectio"
  sh$(3,0) = "SH-M"
  sh$(4,0) = "SH-R"
  sh$(5,0) = "SH-TE"
  sh$(6,0) = "Bk"
  sh$(7,0) = "Bk"
  sh$(8,0) = "Bk"

  for i = 1 to 8
    sh$(i,1) = ""
  next i

'*****************************************************************************
'*** SET UP FOR LS SELECTION HISORY
'*** the four rows correspond to the DR used.
'***
  ls$(0,0) = " IKBS "
  ls$(1,0) = "RND CR-M"    : ls$(2,0) = "RND CR-R"   : ls$(3,0) = "RND CR-LM1"
  ls$(4,0) = "RND CR-LM2"  : ls$(5,0) = "RND CR-QM1" : ls$(6,0) = "Bk"
  ls$(7,0) = "FCS CR-M"    : ls$(8,0) = "FCS CR-R"   : ls$(9,0) = "FCS CR-LM1"
  ls$(10,0) = "FCS CR-LM2" : ls$(11,0) = "FCS CR-QM1": ls$(12,0) = "Bk"
  ls$(13,0) = "SPT CR-M"   : ls$(14,0) = "SPT CR-R"  : ls$(15,0) = "SPT CR-LM1"
  ls$(16,0) = "SPT CR-LM2" : ls$(17,0) = "SPT CR-QM1": ls$(18,0) = "Bk"
  ls$(19,0) = "EDD CR-M"   : ls$(20,0) = "EDD CR-R"  : ls$(21,0) = "EDD CR-LM1"
  ls$(22,0) = "EDD CR-LM2" : ls$(23,0) = "EDD CR-QM1": ls$(24,0) = "Bk"

  ls$(25,0) = "Bk" : ls$(26,0) = "Bk" : ls$(27,0) = "Bk"
  ls$(28,0) = "Bk" : ls$(29,0) = "Bk" : ls$(30,0) = "Bk"
  ls$(31,0) = "Bk" : ls$(32,0) = "Bk" : ls$(33,0) = "Bk"
  ls$(34,0) = "Bk" : ls$(35,0) = "Bk" : ls$(36,0) = "Bk"

  ls$(37,0) = "Sim LS"

  for i = 1 to 37
    for j = 1 to 5
      ls$(i,j) = ""
```

```
    next j
  next i

'*****************************************************************************
'*** SET UP WORK FILE FOR THIS PERIOD
'***
  work$(0,0) = " The Periods Work File "
  work$(0,1) = "Product"
  work$(0,2) = "Sch'd"
  work$(0,3) = "Quan'y"
  work$(0,4) = "Job No."
  work$(0,5) = "Due Week"
  work$(0,6) = "Due Day"
  work$(0,7) = "Due Hour"
  work$(0,8) = "Due Min"
  work$(0,9) = "Priority"
  work$(0,10) = "Bk"
  work$(0,11) = "Bk"
  work$(0,12) = "Bk"
  work$(0,13) = "Bk"
  work$(0,14) = "SqNumber"    'used in getjobs
  work$(0,15) = "INI mc's"    'Initial macs.  Set in m-simcom, used getjobs

  for i = 1 to 15
    for j = 1 to JobNum
      work$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP BOM FILE
'***
  BomHo$(0,1) = "Higher"    : BomHo$(0,2) = "Lower"
  BomHo$(0,3) = "Quant'y"
  BomHo$(0,4) = "1st M/c"   : BomHo$(0,5) = "time"
  BomHo$(0,6) = "2nd M/c"   : BomHo$(0,7) = "time"
  BomHo$(0,8) = "3rd M/c"   : BomHo$(0,9) = "time"
  BomHo$(0,10) = "4th M/c"  : BomHo$(0,11) = "time"
  BomHo$(0,12) = "5th M/c"  : BomHo$(0,13) = "time"
  BomHo$(0,14) = "6th M/c"  : BomHo$(0,15) = "time"
  BomHo$(0,16) = "7th M/c"  : BomHo$(0,17) = "time"
  BomHo$(0,18) = "8th M/c"  : BomHo$(0,19) = "time"
  BomHo$(0,20) = "9th M/c"  : BomHo$(0,21) = "time"
  BomHo$(0,22) = "10th M/c" : BomHo$(0,23) = "time"
  BomHo$(0,24) = "11th M/c" : BomHo$(0,25) = "time"
  BomHo$(0,26) = "12th M/c" : BomHo$(0,27) = "time"
  BomHo$(0,28) = "13th M/c" : BomHo$(0,29) = "time"
  BomHo$(0,30) = "14th M/c" : BomHo$(0,31) = "time"
  BomHo$(0,32) = "15th M/c" : BomHo$(0,33) = "time"
  BomHo$(0,34) = "16th M/c" : BomHo$(0,35) = "time"
  BomHo$(0,36) = "17th M/c" : BomHo$(0,37) = "time"
  BomHo$(0,38) = "18th M/c" : BomHo$(0,39) = "time"
  BomHo$(0,40) = "19th M/c" : BomHo$(0,41) = "time"
  BomHo$(0,42) = "20th M/c" : BomHo$(0,43) = "time"
  BomHo$(0,44) = "Address"

  for i = 1 to 44
    for j = 1 to 20
```

```
      BomHo$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** SET UP SQRU FILE - sq run database
'***

  sq$(0,0) = "Sequencing Database"
  sq$(0,1) = "sequences"
  for i = 1 to sqnum
    sq$(i,1) = ""
  next i

'*****************************************************************************
'*** SET UP THE BN SCHEDULE FILE
'***
  bnsch$(0,1) = "The sq$()"
  bnsch$(0,2) = "load"
  bnsch$(0,3) = "product"
  bnsch$(0,4) = "job no."
  bnsch$(0,5) = ""
  bnsch$(0,0) = "Bottleneck Schedule"

end sub
```

M-LOAD.PRO

```
'     program M-LOAD.PRO

'     OBJECTIVES - to load already created project

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-load.p"
  $include "shared.pro"
  shared file1$

  copypjt = 0
  if z > 7 and z < 14 then
  '  (z = 10 or z = 11) or (z = 12 or z = 13) then
    hold$ = ctl$(11,1)
    call fill(15,3,4,1,80,22)
    call FilePickBox
    hold1$ = ctl$(15,1)
    hold2$ = ctl$(16,1)
    call projects(file1$,file2$,per$)
    if file1$ = "" then z = 20
    ctl$(15,1) = hold1$
    ctl$(16,1) = hold2$
  end if
  do until a = a+1
    select case z
      case = 2
    call fill(15,3,4,1,80,22)
    call ExistPjtSetUp

'*** CHAINING DONE WAY DOWN BELOW - NEEDS SORTING OUT.
    if file1$ <> "" then
      z = 1
      call rblock
      chain "m-ctl.pbc"
    else
      z = 3
      chain "m-ctl.pbc"
    end if
      case = 8
    copypjt = 1
    z = 9
      case = 9
    call waiton
    call dumpho("pdt.hol",pdt$(),50,1)
    call pjtrest(file1$+"-pdt."+file2$,pdt$(),val(per$))
    x$ = pdt$(1,1)
```

```
y$ = pdt$(15,1)
call restorho("pdt.hol",pdt$(),50,1)
pdt$(1,1) = x$
pdt$(15,1) = y$
call pjtdump(hold1$+"~pdt."+hold2$,pdt$(),50,1,val(pdt$(6,1)))
call waitoff
if copypjt <> 1 then
  z = 20
else
  z = 10
end if
  case = 10
call waiton
call restorebom(file1$+"~bom."+file2$)
call restsqdb(file1$+"~sqd."+file2$)
call dumpbom(hold1$+"~bom."+hold2$)
call dumpsq(hold1$+"~sqd."+hold2$)
for i = 1 to 200
  if sq$(i,1) = "" then exit for
  sq$(i,1) = ""
next i
call waitoff
if copypjt <> 1 then
  z = 20
else
  z = 11
end if
  case = 11
call waiton
call restfile(file1$+"~cal."+file2$,cal$(),calnum,8)
call dumpfile(hold1$+"~cal."+hold2$,cal$(),calnum,8)
call waitoff
if copypjt <> 1 then
  z = 20
else
  z = 12
end if
  case = 12
call waiton
call pjtrest(file1$+"~job."+file2$,job$(),val(per$))
call pjtdump(hold1$+"~job."+hold2$,job$(),jobnum,15,val(pdt$(6,1))  )
call waitoff
if copypjt <> 1 then
  z = 20
else
  z = 13
end if
  case = 13
call waiton
call restoreres(file1$+"~mac."+file2$)
call dumpres(hold1$+"~mac."+hold2$)
call waitoff
if copypjt <> 1 then
  z = 20
else
  z = 20
end if
  case = 20
```

```
    z = 2
    ctl$(11,1) = hold$
    call rblock
    chain "m-create.pbc"
      case else
    call scrclear
    locate 10,10 : print "error in going into m-load"
    locate 12,10 : print "Z value is: ";z
    beep 5
    hj$=input$(12)
    end select
  loop

end sub

' ***** SUBROUTINE BLOCK *****

sub FilePickBox
  call shadow(0,3,6,7,36,9)
  call fill(15,1,6,7,36,9)
  color 14,1
  locate 7,18: print "FILE LOAD SCREEN"
  color 15,1
  locate 9,14
  color 15,1 : print " Use ";
  color 14,1 : print chr$(24);chr$(25);
  color 15,1 : print " to move cursor"
  locate 10,14
  color 15,1 : print "Press ";
  color 14,1 : print "<enter> ";
  color 15,1 : print "to select"
  locate 11,15
  color 14,1 : print "PROJECT ";
  color 15,1 : print "then ";
  color 14,1 : print "FILE SET";
  locate 13,14
  color 14,1 : print "     <esc>";
  color 15,1 : print " to quit"
end sub


sub ExistPjtSetUp
  $include "shared.pro"

  ctl$(14,1) = "1"
  call shadow(0,3,6,7,36,9)
  call fill(15,1,6,7,36,9)
  color 14,1
  locate 7,16 : print "PROJECT LOAD SCREEN"
    color 15,1
    locate 9,14
    color 15,1 : print " Use ";
    color 14,1 : print chr$(24);chr$(25);
    color 15,1 : print " to move cursor"
    locate 10,14
    color 15,1 : print "Press ";
    color 14,1 : print "<enter> ";
    color 15,1 : print "to select"
```

```
locate 11,15
color 14,1 : print "PROJECT ";
color 15,1 : print "then ";
color 14,1 : print "FILE SET";
locate 13,14
color 14,1 : print "     <esc>";
color 15,1 : print " to quit"

call projects(file1$,file2$,per$)
if file1$ = "" then exit sub

ctl$(10,1) = "0"
y$ = file1$
z$ = file2$
d = 9
call WaitOn

call arraywipe        '*** cleans all the arrays out

call shadow(0,3,7+d,7,37,9)
call fill(15,1,7+d,7,37,9)
color 14,1
locate 7+d,18 : print " LOADING FILES"

f$ = file1$+"~pdt."+file2$
color  4,7 : locate d+8,11 : print f$
call pjtrest(f$,pdt$(),val(per$))
color 15,1 : locate d+8,11 : print f$

f$ = file1$+"~bom."+file2$
color 4,7 : locate d+9,11 : print f$
delay 0.5
color 15,1 : locate d+9,11 : print f$

f$ = file1$+"~cal."+file2$
color  4,7 : locate d+10,11 : print f$
call restfile(f$,cal$(),calnum,8)
color 15,1 : locate d+10,11 : print f$

f$ = file1$+"~job."+file2$
color  4,7 : locate d+11,11 : print f$
call pjtrest(f$,job$(),val(per$))
color 15,1 : locate d+11,11 : print f$

f$ = file1$+"~mac."+file2$
color  4,7 : locate d+12,11 : print f$
call restoreres(f$)
color 15,1 : locate d+12,11 : print f$

f$ = file1$+"~que."+file2$
color  4,7 : locate d+13,11 : print f$
call pjtrest(f$,queues$(),val(per$))
color 15,1 : locate d+13,11 : print f$

f$ = file1$+"~rsj."+file2$
color  4,7 : locate d+14,11 : print f$
call pjtrest(f$,rltj$(),val(per$))
color 15,1 : locate d+14,11 : print f$
```

```
    f$ = file1$+"~rsm."+file2$
    color  4,7 : locate d+15,11 : print f$;
    call pjtrest(f$,rltm$(),val(per$))
    color 15,1 : locate d+15,11 : print f$;

    f$ = file1$+"~rsg."+file2$
    color  4,7 : locate d+8,28 : print f$
    call pjtrest(f$,rltg$(),val(per$))
    color 15,1 : locate d+8,28 : print f$

    f$ = file1$+"~crh."+file2$
    color  4,7 : locate d+9,28 : print f$
    call pjtrest(f$,cr$(),val(per$))
    color 15,1 : locate d+9,28 : print f$

    f$ = file1$+"~lsy."+file2$
    color  4,7 : locate d+11,28 : print f$
    call pjtrest(f$,ls$(),val(per$))
    color 15,1 : locate d+11,28 : print f$

    f$ = file1$+"~wrk."+file2$
    color  4,7 : locate d+12,28 : print f$
    call pjtrest(f$,work$(),val(per$))
    color 15,1 : locate d+12,28 : print f$

    f$ = file1$+"~sqr."+file2$
    color  4,7 : locate d+15,28 : print f$;
    call pjtrest(f$,sq$(),val(per$))
    color 15,1 : locate d+15,28 : print f$;

    f$ = file1$+"~bns."+file2$
    color  4,7 : locate d+15,28 : print f$;" ";
    call pjtrest(f$,bnsch$(),val(per$))
    color 15,1 : locate d+15,28 : print f$;" ";

'************************** for dumping the new loaded file to new version
    x = val(ctl$(11,1)) : call toupper(ctl$(16,1))
    if ctl$(16,1) <> "INF" then
      ctl$(17,1) = ctl$(16,1)
    elseif ctl$(16,1) = "INF" then      'and ctl$(17,1) = "") then
      flag = 0
      for i = 4 to 8
    if val(pjt$(x,i)) = 0 then flag = 1
      next i
      if flag = 1 then
    ctl$(17,1) = "inf"
      else
    pjt$(x,2) = FNtrun$( str$( val(pjt$(x,2))+1 ) )
    if val(pjt$(x,2)) < 10 then
      pjt$(x,2) = "00" + pjt$(x,2)
    else
      pjt$(x,2) = "0" + pjt$(x,2)
    end if
    ctl$(17,1) = pjt$(x,2)
    call fill(15,7,18,9,32,5)
    call shadow(0,1,18,9,32,5)
    call waiton
```

```
    color 4,7
    locate 19,14 : print " Creating version: ";
    color 14,7 : print ctl$(17,1)
    color 4,7
    locate 21,14 : print "    from ROOT files"
    call openclose(ctl$(15,1),ctl$(17,1))
    call dump(ctl$(15,1),ctl$(17,1),val(pdt$(6,1)))
    call dumpho("f-pjt.inf",pjt$(),15,15)
    call waitoff
      end if
    end if
'************************** end of sorting out new version number and dumping

    call WaitOff
    x = val(ctl$(11,1))
    do
      flag = 0
      if flag = 1 then
    call fill(15,4,11,19,39,9)
        call box(15,4,11,20,37,9)
        color 15,4
        locate 13,23 : print "   Not all files are defined  "
        locate 15,23 : print "  press ";
        color 14,4 : print "<space bar> ";
        color 15,4 : print "to enter"
        color 15,4
        locate 17,23 : print "       Data Input Screen    "
        x$ = input$(1)
        a = 123
      else
    z = 3
    call rblock
    chain "m-ctl.pbc"
      end if
    loop until a = 123
end sub



sub projects(file1$,file2$,per$)
  shared pjt$(), ctl$(), pdt$()

level = 0
DO
  incr level
  down = 1
  icon = 1
  if level = 1 then
    call shadow(0,3,6,47,21,17)
    call fill(15,1,6,47,21,17)
    color 14,1
    locate 7,50 : print "Number"
    locate 7,58 : print "Project"
    locate 8,58 : print "Name"
    color 15,1
    PROJECT = 15
    maxrow = 1
    do until pjt$(maxrow,1) = ""
```

```
    incr maxrow
    IF MAXROW = PROJECT + 1 THEN EXIT LOOP
  loop
  decr maxrow

elseif level = 2 then
  call shadow(0,3,7,51,21,17)
  call fill(15,7,7,51,21,17)
  color 4,7
  locate 8,63 : print " ";pjt$(file1,1);" ";
  color 14,7
  locate 8,54 : print "Project: ";
  locate 9,54 : print "Available Files"
  color 15,1
  maxrow = val(pjt$(file1,2)) + 1
elseif level = 3 then
  call shadow(0,3,8,55,21,17)
  call fill(15,1,8,55,21,17)
  color 15,1
  locate 9,62 : print pjt$(file1,1)
  if file2$ = "inf" then
    x$ = "Root"
  else
    x$ = file2$
  end if
  locate 9,71 : print x$
  color 14,1
  locate 9,57 : print "Pjt:"
  locate 9,66 : print "Ver:"
  locate 10,57 : print "Periods Available"
  color 15,1
  open file1$+"~pdt."+file2$ for input as #1
  do until eof(1)
    input #1, x$
    z$ = z$ + left$(x$,3)
    for i = 1 to 51
  input #1,y$
    next i
  loop
  close #1
  maxrow = len(z$)\3 + 1
end if

do
  if level = 1 then
    for i = 1 to 13
      if pjt$(i,1) = "" then exit for
      color 14,1
      locate 8+i,52 : print using "##";i-1+down
      color 15,1
      locate 8+i,58 : print pjt$(i-1+down,1)
      color 14,1
    next i
    color 31,1 : locate 9+icon-down,49 : print chr$(4) : color 15,1
  elseif level = 2 then
    color 4,7
    locate 10,56 : print "ROOT"
    count = 1
```

```
  do until count-1 = val(pjt$(file1,2))
    if count = 13 then exit loop
locate 10+count,56 : print "Version:";
x$ = FNtrun$(str$(count-1+down))
if val(x$) < 10 then
  x$ = "00"+x$
else
  x$ = "0"+x$
end if
locate 10+count,65 : print x$
    incr count
  loop
  color 31,7 : locate 10+icon-down,53 : print chr$(4) : color 15,1
elseif level = 3 then
  color 15,1
  locate 11,60 : print "LAST"
  count = 1

  do until count-1 = len(z$)\3
if count = 13 then exit loop
x = count-1+down
x$ = mid$(z$,(x*3)-2,3)
if x$ = "" then exit loop
locate 11+count,60 : print "Period:";
locate 11+count,68 : print x$
    incr count
  loop
  color 31,1 : locate 11+icon-down,57 : print chr$(4) : color 15,1
end if

color 14,1


do
  x$ = inkey$
loop until (len(x$) > 0) '''and ((asc(x$)=13 or asc(x$)=27) or_
           '''      (asc(x$)=72 or asc(x$)=80))
x$ = FNtrun$(x$)
if asc(x$) = 27 and level = 1 then
  exit sub
elseif asc(x$) = 13 then
  if level = 1 then
color 15,1 : locate 8+icon,49 : print " ";
    locate 7,50 : print " "
    file1$ = pjt$(icon,1)
file1 = icon
ctl$(11,1) = FNtrun$(str$(icon))
ctl$(15,1) = file1$
locate 13,14
color 15,1 : print "                    "
  elseif level = 2 then
color 14,7 :
x = icon
if icon > 13 then x = 12
locate 9+x,53 : print " ";
    if icon = 1 then
      file2$ = "inf"
    else
```

```
      file2$ = FnTrun$(str$(icon-1))
      if val(file2$) < 10 then
        file2$ = "00" + file2$
      elseif val(file2$) < 100 then
        file2$ = "0" + file2$
      end if
    end if
ctl$(16,1) = file2$
  elseif level = 3 then
color 15,1
x = icon
if icon > 13 then x = 12
locate 11+x,57 : print " ";
    if icon = 1 then
  per$ = "999"
else
  icon = icon - 1
  per$ = mid$(z$,(icon*3)-2,3)
end if
  end if
  exit loop
end if

if len(x$) = 2 then
  x = asc(mid$(x$,2,1))
  select case x
    case = 79
      exit loop
    case = 80              '*** DOWN ARROW
      incr icon
      if icon = maxrow + 1 then
        decr icon
        call failsound
        exit select
      end if
      if icon < 14 then
        color 15,1
        if level = 1 then
          color 31,1 : locate 8+icon,49 : print chr$(4)
          color 15,1 : locate 7+icon,49 : print " "
    elseif level = 2 then
          color 31,7 : locate 9+icon,53 : print chr$(4)
          color 15,7 : locate 8+icon,53 : print " "
    elseif level = 3 then
      color 31,1 : locate 10+icon,57 : print chr$(4)
      color 15,1 : locate 9+icon,57 : print " "
        end if
      elseif icon >= 14 then
        incr down
      end if
    case = 72              '*** UP ARROW
      decr icon
      if icon > 0 and down = 1 then
        color 15,1
        if level = 1 then
          color 31,1 : locate 8+icon,49 : print chr$(4)
          color 15,1 : locate 9+icon,49 : print " "
    elseif level = 2 then
```

```
            color 31,7 : locate  9+icon,53 : print chr$(4)
            color 15,7 : locate 10+icon,53 : print " "
        elseif level = 3 then
          color 31,1 : locate 10+icon,57 : print chr$(4)
          color 15,1 : locate 11+icon,57 : print " "
            end if
          elseif down > 1 then
            decr down
          elseif icon = 0 then
            incr icon
            call failsound
          end if
        case else
          call failsound
      end select
    end if
    x$ = ""
  loop until X$ = "Z" or x$ = "z"
LOOP UNTIL LEVEL = 3
end sub
```

M-EDIT.PRO

```
'     program M-EDIT.PRO

'     OBJECTIVES - edit the loaded project files

'     Edition 2.0   Version 2.0   25/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"

call go

sub go
  locate 4,1 : print "m-edit.p"
  $include "shared.pro"
  if z = 1 then
    call editmenu
    z = 3
    chain "m-ctl.pro"
  else
    call failsound
    call scrclear
    locate 10,10 :print "WHOOPS!!! did not set z when chaining back to m-edit"
    fg$=input$(9)
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub editmenu
  $include "shared.pro"
  call fill(0,3,4,1,80,20)
  b = 0
  do
    select case b
    case = 0
      call menu("PROJECT EDITING ",_
    "A.  Project Data","",_
    "B.  Jobs File","",_
    "C.  Resources File","",_
    "D.  BoM File","",_
    "E.  Calander File",_
    "",_
    "P.  Previous Menu","")
    color 15,1
    b = 1
    case = 1
      color 15,4 : locate 23,8 : print " Select an item: "
      call insound
      b$ = FNindata$(23,29,1)
      call toupper(b$)
      b = asc(b$)
    case = 65
      z = 1
      call na
```

```
      b = 0
    case = 66
      z = 2
      chain "m-job.pbc"
    case = 67
      z = 3
      chain "m-res.pbc"
    case = 68
      call na
      b = 0
    case = 69
      z = 2
      chain "m-cal.pbc"
    case = 80
      z = 3
      locate 4,1 : print "m-ctl.pr"
      chain "m-ctl.pbc"
    case else
      color 0,3 : locate 23,8 : print "                          "
      color 15,1
      call shadow(0,3,6,43,34,9)
      call fill(15,4,6,43,34,9)
      call box(15,4,6,44,32,9)
      locate 8,45  : print "        INVALID INPUT      "
      locate 10,45 : print "      PRESS <space bar>    "
      locate 12,45 : print "         TO CONTINUE       "
      call failsound : x$ = input$(1)
      call fill(0,3,6,43,35,10)
      color 15,1
      b = 1
    end select
  loop until abc = 90
end sub
```

M-DEL.PRO

```
'     program M-DEL.PRO

'     OBJECTIVES - to delete a project

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-del.pr"
  $include "shared.pro"
  if z = 1 then
    call delattrib
    z = 3
    chain "m-ctl.pbc"
  elseif z = 2 then
    call delattrib
    chain "que.pbc"
  elseif z = 5 then
    sim$(3,1) = "1"
    call delattrib
    z = 1
    chain "m-ctl.pbc"
    sim$(3,1) = "0"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub delattrib
  shared sim$(), pjt$(), util$(), simrun$(), res$(), job$(), jobnum
  shared cal$(), queues$(), resources, QueFin$(), mac$(), ctl$()

  do
    call shadow(0,3,6,7,40,8)
    call fill(15,1,6,7,40,8)
    call shadow(0,3,6,51,21,19)
    call fill(15,1,6,51,21,19)
    color 14,1
    locate 7,53 : print "EXISTING PROJECTS"
    color 15,1
    locate 9,53
    if pjt$(1,1) = "" then print "      none"
    color 14,1
    for i = 1 to 15
      if pjt$(i,1) = "" then exit for
      locate 8+i,57 : print using "##";i
      color 15,1
      locate 8+i,62 : print pjt$(i,1)
      color 14,1
```

```
 next i
 color 14,1
 locate 7,19 : print " DELETE PROJECT "

 color 15,1
 locate 9,15 :  print "   Input Project Name "
 locate 10,15 : print "       (Z = exit) "
 x = val(sim$(1,1))

 call insound
 z$ = FNindate$(12,26,3) : call toupper(z$)
 flag = 0
 do until flag = 1
   if left$(z$,1) = "Z" then exit loop
   for i = 1 to 15
 if z$ = pjt$(i,1) then
   flag = 1
   z1 = i
   exit loop
 end if
   next i

   call fill(0,4,17,15,24,5)
   call shadow(0,3,17,15,24,5)
   color 15,4
   locate 18,18 : print "  INVALID INPUT "
   locate 20,18 : print " press <space bar>"
   call failsound
   call spce : call fill(0,3,17,15,25,6)
   call insound
   z$ = FNindate$(12,26,3) : call toupper(z$)
 loop


 if flag = 1 then
   call fill(0,4,17,12,30,6)
   call shadow(0,3,17,12,30,6)
   color 15,4
   locate 18,14 : print " ARE YOU SURE YOU WISH TO"
   locate 19,14 : print "DELETE PROJECT ";
   color 4,7 : print " ";z$;" ";
   file1$ = z$
   color 15,4 : print " (Y/N)?"
   x$ = FNindate$(21,26,3) : call toupper(x$)
   do until left$(x$,1) = "Y" or left$(x$,1) = "N"
     call failsound
     x$ = FNindate$(21,26,3) : call toupper(x$)
   loop
   if left$(x$,1) = "Y" then

 y$ = pjt$(z1,1)

 if y$ = ctl$(15,1) then
   ctl$(15,1) = ""
   ctl$(16,1) = ""
   ctl$(17,1) = ""

   call rblock
```

```
 end if


 if z1 = 15 then
   for i = 1 to 15
        pjt$(15,i) = ""
      next i
    else
   for i = z1 to 14
     for j = 1 to 15
          pjt$(i,j) = pjt$(i+1,j)
        next j
      next i
   for i = 1 to 15
        pjt$(15,i) = ""
      next i
   ctl$(13,1) = FNtrun$(str$(val(ctl$(13,1)) - 1))
    end if

file2$ = "INF"
d = 9
call WaitOn
call shadow(0,3,6+d,7,40,10)
call fill(15,1,6+d,7,40,10)
color 14,1
locate 6+d,18 : print "DELETING FILES"

f$ = file1$+"-pdt."+file2$
color  4,7 : locate d+8,11 : print f$
kill f$
color 15,1 : locate d+8,11 : print f$

f$ = file1$+"-bom."+file2$
color 4,7 : locate d+9,11 : print f$
kill f$
color 15,1 : locate d+9,11 : print f$

f$ = file1$+"-cal."+file2$
color  4,7 : locate d+10,11 : print f$
kill f$
color 15,1 : locate d+10,11 : print f$

f$ = file1$+"-job."+file2$
color  4,7 : locate d+11,11 : print f$
kill f$
color 15,1 : locate d+11,11 : print f$

f$ = file1$+"-mac."+file2$
color  4,7 : locate d+12,11 : print f$
kill f$
color 15,1 : locate d+12,11 : print f$

f$ = file1$+"-que."+file2$
color  4,7 : locate d+13,11 : print f$
kill f$
color 15,1 : locate d+13,11 : print f$
```

```
    f$ = file1$+"~rsj."+file2$
    color  4,7 : locate d+14,11 : print f$
    kill f$
    color 15,1 : locate d+14,11 : print f$

    f$ = file1$+"~rsm."+file2$
    color  4,7 : locate d+15,11 : print f$;
    kill f$
    color 15,1 : locate d+15,11 : print f$;

    f$ = file1$+"~rsg."+file2$
    color  4,7 : locate d+8,28 : print f$
    kill f$
    color 15,1 : locate d+8,28 : print f$

    f$ = file1$+"~crh."+file2$
    color  4,7 : locate d+9,28 : print f$
    kill f$
    color 15,1 : locate d+9,28 : print f$

    f$ = file1$+"~shh."+file2$
    color  4,7 : locate d+10,28 : print f$
    kill f$
    color 15,1 : locate d+10,28 : print f$

    f$ = file1$+"~lsy."+file2$
    color  4,7 : locate d+11,28 : print f$
    kill f$
    color 15,1 : locate d+11,28 : print f$

    f$ = file1$+"~wrk."+file2$
    color  4,7 : locate d+12,28 : print f$
    kill f$
    color 15,1 : locate d+12,28 : print f$

    f$ = file1$+"~lm1."+file2$
    color  4,7 : locate d+13,28 : print f$
    kill f$
    color 15,1 : locate d+13,28 : print f$

    f$ = file1$+"~qm1."+file2$
    color  4,7 : locate d+14,28 : print f$
    kill f$
    color 15,1 : locate d+14,28 : print f$

    call dumpho("f-pjt.inf",pjt$(),15,15)
        sim$(4,1) = "0"
        call WaitOff
        call fill(0,3,6+d,7,41,11)
    flag = 0
    if val(ctl$(13,1)) = 0 then flag = 1
      end if              '*** YES SURE TO DELETE
    else
      flag = 1
    end if
    call fill(0,3,17,12,31,7)

loop until flag = 1
```

```
end sub
```

ADV.PRO

```
'     program ADV.PRO

'     OBJECTIVES - subroutines to advance the simulation

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"
$SEGMENT
'$include "m-crsh.pro"

call go

sub go
  locate 4,1 : print "adv.pr"
  $include "shared.pro"
  call scrclear
  do until a = a + 1
    if z = 0 then
      call welcome
      call scrclear
      call border
      call advmenu
    elseif z = 1 then
      call scrclear
      call na
      z = 0
    elseif z = 2 then
      call advance(101)
      if z = 10 then
    z = 1
'*********************************************
'*** does it do any harm if we dump this version of the pdt$()
'*** database so not really the starting point.
'***
    y$ = ctl$(15,1)
    z$ = ctl$(17,1)
    if ctl$(40,1) = "11" then     '*** so it does overwrite
      ctl$(40,1) = ""
      call pjtdump(y$+"~pdt."+z$,pdt$(),50,1,val(pdt$(6,1)))
      ctl$(40,1) = "11"
    else
      call pjtdump(y$+"~pdt."+z$,pdt$(),50,1,val(pdt$(6,1)))
    end if
    chain "m-seq.pbc"
      elseif z = 11 then
    z = 16
      else
    call scrclear : locate 10,10
    print "WHOOPS in advance where z = 2"
      end if
    elseif z = 16 then
```

```
      z = 1
      chain "m-ctl.pbc"
    else
      call failsound
      locate 11,11 : print "problems with z going into adv.pro"
      klklk$=input$(90)
    end if
  loop
end sub

' ***** SUBROUTINE BLOCK *****

sub advmenu
  $include "shared.pro"

  call shadow(0,3,6,43,34,9)
  call fill(15,1,6,43,34,9)
  color 14,1
  locate 7,47 : print "PROJECT STATUS"
  color 15,1
  locate 10,47  : print "    Project Loaded:";
  color 14,1
  print " ";ctl$(15,1);" "
  color 15,1
  call menu("  SIMULATION CONTROL",_
    "* A.  Limited Control","",_
    "  B.  Full Control","","",_
    "","",_
    "  P.  Previous Menu","","","","")
  color 15,4 : locate 23,8 : print " Select an item: "
  call insound
  x$ = FNindata$(23,29,1)
  call toupper(x$)
  do until (x$ = "A" or x$ = "B") or (x$ = "L" or x$ = "P")
    color 0,3 : locate 23,8 : print "                              "
    call shadow(0,3,16,43,34,5)
    call fill(15,4,16,43,34,5)
    color 15,4
    locate 17,47 : print "      INVALID INPUT"
    locate 19,47 : print "    Press <space bar>";
    call failsound
    call spce
    call fill(0,3,16,43,35,6)
    color 15,4 : locate 23,8 : print " Select an item: "
    call insound
    x$ = FNindata$(23,29,1) : call toupper(x$)
  loop
  locate 23,24 : color 15,3
  print "                                        "
  z = asc(x$) - 64
end sub

sub advance(b)
  shared dep = 20
  $include "shared.pro"
  call scrclear
  call border
  do until a = a + 1
```

```
 select case b
   case = 101
 call shadow(0,3,6,5,41,19)
 call fill(14,1,6,5,41,19)
 color 14,1
 locate 7,6  : print "SIMULATION                Full Control"
 locate 9,6  : print "Master"
 locate 10,6 : PRINT "Data"
 x$ = "A.B.C.D.  E.F.G.  H.I.J.K.L.M.N." : color 14,1
 for i = 1 to 16
   locate i+8,15 : print mid$(x$,(i*2)-1,2);
 next i
 color 15,1
 locate 9,18  : print "Batch Size Method...."
 locate 10,18 : print "Set Up Method........"
 locate 11,18 : print "Critical Resource Id."
 locate 12,18 : print "Simulate x Periods..."
 color 14,1
 locate 14,6 : print "Logic"
 color 15,1
 locate 14,18 : print "Loading Rule........."
 locate 15,18 : print "Selecting Rule......."
 locate 16,18 : print "Unload Rule........."
 color 14,1
 locate 18,6 : print "Output"
 color 15,1
 locate 18,18 : print "Screen Display......."
 locate 19,18 : print "Monitoring Selection."
 locate 20,18 : print "Save Actions........."
 locate 21,18 : print "Save Results After..."
 locate 22,18 : print "Lead Time Offset....."
 locate 23,18 : print "Tardiness Tolerance..";
 locate 24,18 : print "Clear Crit'y List ...";
 color 4,7
 locate 9,40  : print " ";pdt$(16,1)
 locate 10,40 : print " ";pdt$(18,1)
 locate 11,40 : print " ";pdt$(11,1)
 locate 12,40
 if len(pdt$(4,1)) = 1 then print " ";
 print pdt$(4,1)
 locate 14,40 : print " ";pdt$(20,1)
 locate 15,40 : print " ";pdt$(14,1)
 locate 16,40 : print " ";pdt$(24,1)
 locate 18,40 : print " ";pdt$(25,1)
 locate 19,40 : print " ";pdt$(26,1)
 locate 20,40 : print " ";pdt$(28,1)
 locate 21,40 : if len(pdt$(39,1)) = 1 then print " ";
 print pdt$(39,1)
 locate 22,40 : if len(pdt$(21,1)) = 1 then print " ";
 print pdt$(21,1)
 locate 23,40 : if len(pdt$(22,1)) = 1 then print " ";
 print pdt$(22,1)
 locate 24,40 : print " N";
 locate 24,40 : if val(pdt$(40,1)) = 1 then print " Y";
 pdt$(27,1) = FNtrun$(pdt$(27,1))
 locate 19,43 :
 if len(pdt$(27,1)) = 1 then
   print " ";
```

```
    print pdt$(27,1)
  else
    print left$(pdt$(27,1),2)
  end if
  color 15,1
  b = 0
    case = 0
  call fill(15,1,6,50,27,12)
  call shadow(0,3,6,50,27,12)
  color 14,1
  locate 7,55 : print "SELECTION MENU"
  color 14,1 : locate 9,52  : print "# ";
  color 15,1 : print "Modify Parameters"
  locate 10,53 : print "(input letter A-L)"
  color 14,7 : locate 12,51 : print " S. ";
  color 4,7  : print "Advance Simulation "
  color 14,1 : locate 14,52 : print "R. ";
  color 15,1 : print "Parameter Reset"
  color 14,1 : locate 16,52 : print "P. ";
  color 15,1 : print "Exit to Main Menu"
  locate 19,51 : print " Select an option: " : call insound
  x$ = FNindata$(19,74,1) : call toupper(x$)
  do
    if (asc(x$) > 64 and asc(x$) < 79) then
      flag = 1
    elseif (x$ = "S" or x$ = "R") then
      flag = 1
    elseif (x$ = "P" or x$ = "Z") then
      flag = 1
    else
      flag = 0
      call fill (15,4,19,50,27,6)
      call shadow(0,3,19,50,27,6)
      color 14,4
      locate 20,55 : print "  INVALID INPUT  "
      locate 22,55 : print "Press <space bar>"
      locate 23,55 : print "   to continue   "
      call failsound : call spce
      call fill(0,3,19,50,28,7)
      locate 19,51 : print " Select an option: " : call insound
      x$ = FNindata$(19,74,1) : call toupper(x$)
    end if
  loop until flag = 1
  call fill(15,3,6,50,28,16)
  if x$ = "P" then
    b = 90
  elseif x$ = "R" then
    b = 82
  elseif asc(x$) > 64 and asc(x$) < 79 then
    b = asc(x$) - 64
  elseif val(pdt$(14,1)) = 0 then
    call sidebox("  INITIALISATION ERROR",_
      "   Option F requires",_
      "    initialisation.",_
      "",_
      "   Press <space bar>",_
      "      to continue",_
      "",_
```

```
      "",_
      "")
      call spce
    b = 0
  elseif x$ = "S" then
    b = 83
    ctl$(40,1) = ""
  elseif x$ = "Z" then
    b = 83     '*** to run for steady state by reloading
    ctl$(40,1) = "11"      '*** the initial job file from root.
    call fill (15,4,20,50,27,5)
    call shadow(0,3,20,50,27,5)
    color 15,4
    locate 21,52 : print "Mean Demand:"
    locate 22,52 : print "Variance:"
    locate 23,52 : print "Random Seed:"
    ctl$(30,1) = FNindata$(21,65,3)
    ctl$(31,1) = FNindata$(22,65,3)
    ctl$(32,1) = FNindata$(23,65,3)
  else
    call failsound
    call scrclear : print "whoops in advance"
    fgf$=input$(90)
  end if

    case = 1
  call sidebox("  A. BATCH SIZE METHOD","",_
    "     1.  None",_
    "     2.  Global",_
    "     3.  Product",_
    "     4.  Half","",_
    "     Z.  Exit",_
    "")
  color 15,1
  locate 18,51 : print " Select an option: " : call insound
  x$ = FNindata$(18,74,1) : call toupper(x$)
  do until (val(x$) > 0 and val(x$) < 5) or (x$ = "Z")
    call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
  loop
  if asc(x$) <> 90 then
    pdt$(16,1) = x$
  end if

  if val(x$) = 2 then
    call fill (4,7,18,50,27,5)
    call shadow(0,3,18,50,27,5)
    color 4,7
    locate 19,53 : print " Current Value: ";
    color 15,1
    print " ";pdt$(15,1);" "
    color 4,7
    locate 21,53 : print " New value:     "
    x$ = FNindata$(21,70,3)
    do until (val(x$) <> 0)
      call failsound : x$ = FNindata$(21,70,3)
    loop
    pdt$(15,1) = FNtrun$(x$)
    call fill(0,3,18,50,28,6)
```

```
  end if
  color 4,7
  locate 9,40  : print " ";pdt$(16,1);
  color 15,1
  b = 0
    case = 2
  call sidebox("B. SET UP TIME SOURCE",_
    "     1.  No Set Up",_
    "     2.  Global",_
    "     3.  Resource",_
    "     4.  Product",_
    "     5.  Part",_
    "",_
    "     Z.  Exit","")
  color 15,1
  locate 18,51 : print " Select an option: " : call insound
  x$ = FNindata$(18,74,1) : call toupper(x$)
  do until (val(x$) > 0 and val(x$) < 6) or (x$ = "Z")
    call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
  loop
  if asc(x$) <> 90 then
    pdt$(18,1) = x$
  end if
  if val(x$) = 2 then
    call fill (4,7,18,50,27,5)
    call shadow(0,3,18,50,27,5)
    color 4,7
    locate 19,53 : print " Current Value: ";
    color 15,1
    print " ";pdt$(1,1);" "
    color 4,7
    locate 21,53 : print " New value:     "
    x$ = FNindata$(21,70,3)
    do until (val(x$) <> 0)
      call failsound : x$ = FNindata$(21,70,3)
    loop
    pdt$(1,1) = FNtrun$(x$)
    call fill(0,3,18,50,28,6)
  end if
  color 4,7
  locate 10,40  : print " ";pdt$(18,1)
  color 15,1
  b = 0
    case = 3
  call fill (15,4,20,50,27,5)
  call shadow(0,3,20,50,27,5)
  color 15,4
  locate 21,52 : print " The chosen method is"
  locate 22,52 : print "executed on leaving the"
  locate 23,52 : print "Simulation Control menu."
  call sidebox(" C. C.R IDENTIFICATION",_
    "     1.  Manual",_
    "     2.  Random",_
    "     3.  Loading 1",_
    "     4.  Loading 2",_
    "     5.  Queuing 1",_
    "",_
    "     Z.  Exit","")
```

```
color 15,1
locate 18,51 : print " Select an option: " : call insound
x$ = FNindata$(18,74,1) : call toupper(x$)
do until (val(x$) > 0 and val(x$) < 6) or (x$ = "Z")
  call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(11,1) = x$
end if
color 4,7
locate 11,40  : print " ";pdt$(11,1)
call fill(0,3,20,50,28,6)
color 15,1
b = 0
  case = 4
call sidebox(" D. PERIODS TO SIMULATE",_
  "",_
  "    How many periods",_
  "    to simulate ",_
  "","    Z = Exit","","","")
color 15,1
color 15,1
locate 18,51 : print " Input a number or Z: " : call insound
x$ = FNindata$(18,75,2) : call toupper(x$)
do until (val(x$) > 0 or x$ = "Z")
  call failsound : x$ = FNindata$(18,75,2) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(4,1) = FNtrun$(x$)
end if
color 4,7
locate 12,40
if len(pdt$(4,1)) = 1 then print " ";
print pdt$(4,1)
color 15,1
b = 0
  case = 5
call sidebox("E. LOADING RULE",_
  " 1. Part then Component","",_
  " 2. Component then Part","",_
  " 3. Both Together",_
  "",_
  " Z. Exit",_
  "")
color 15,1
locate 18,51 : print " Select an option: " : call insound
x$ = FNindata$(18,74,1) : call toupper(x$)
do until (val(x$) > 0 and val(x$) < 4) or (x$ = "Z")
  call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(20,1) = x$
end if
color 4,7
locate 14,40  : print " ";pdt$(20,1)';" "
color 15,1
b = 0
```

```
  case = 6
call sidebox("F. SELECTION RULE",_
  "     1.  Random",_
  "     2.  FCFS",_
  "     3.  SPT",_
  "     4.  EDD",_
  "     5.  EDD/SPT","",_
  "     Z.  Exit","")

color 15,1
locate 18,51 : print " Select an option: " : call insound
x$ = FNindata$(18,74,1) : call toupper(x$)
do until (val(x$) > 0 and val(x$) < 6) or (x$ = "Z")
  call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(14,1) = x$
end if
color 4,7
locate 15,40  : print " ";pdt$(14,1)
color 15,1
b = 0
  case = 7
call sidebox("  G. UNLOAD LOGIC",_
  "  1.  To Output Queue","",_
  "  2.  To Next Input Q",_
  "","",_
  "  Z.  Exit","","")
color 15,1
locate 18,51 : print " Select an option: " : call insound
x$ = FNindata$(18,74,1) : call toupper(x$)
do until (val(x$) > 0 and val(x$) < 3) or (x$ = "Z")
  call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(24,1) = x$
end if
color 4,7
locate 16,40  : print " ";pdt$(24,1)
color 15,1
b = 0
  case = 8
call sidebox("  H. SCREEN DISPLAY","",_
  "     1.  Display Zero",_
  "     2.  Display One",_
  "     3.  Display Two","","",_
  "     Z.  Exit","")
color 15,1
color 15,1
locate 18,51 : print " Input a number or Z: " : call insound
x$ = FNindata$(18,75,2) : call toupper(x$)
do until (val(x$) > 0 and val(x$) < 4) or (x$ = "Z")
  call failsound : x$ = FNindata$(18,75,2) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(25,1) = FNtrun$(str$(val(x$)))
end if
color 4,7
```

```
 locate 18,40  : print " ";pdt$(25,1)
 color 15,1
 b = 0
   case = 9
 call simvar
 color 4,7
 locate 19,40  : print " ";pdt$(26,1)
 locate 19,43
 if len(pdt$(27,1)) = 1 then
   print " "; : print pdt$(27,1)
 else
   print using "\\";pdt$(27,1)
 end if
 color 15,1
 b = 0
   case = 10
 call sidebox(" J. SIMULATION STEPS",_
   "",_
   " 1.  To File and Screen",_
   "",_
   " 2.  To Screen Only","",_
   " Z.  Exit","","")
 color 15,1
 color 15,1
 locate 18,51 : print " Select an option: " : call insound
 x$ = FNindata$(18,74,1) : call toupper(x$)
 do until (val(x$) > 0 and val(x$) < 3) or (x$ = "Z")
   call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
 loop
 if asc(x$) <> 90 then
   pdt$(28,1) = x$
 end if
 color 4,7
 locate 20,40  : print " ";pdt$(28,1)
 color 15,1
 b = 0
   case = 11
 call sidebox(" L. SAVE RESULTS",_
   "",_
   "    Save results after",_
   "    which period ",_
   "","    Z = Exit","","","")
color 15,1
locate 18,51 : print " Input a number or Z: " : call insound
x$ = FNindata$(18,75,2) : call toupper(x$)
do until (val(x$) > 0 or x$ = "Z")
  call failsound : x$ = FNindata$(18,75,2) : call toupper(x$)
loop
if asc(x$) <> 90 then
  pdt$(39,1) = FNtrun$(str$(val(x$)))
end if
color 4,7
locate 21,40
if len(pdt$(39,1)) = 1 then print " ";
print pdt$(39,1)
color 15,1
b = 0
  case = 12
```

```
 call sidebox("  L. LEAD TIME OFFSET",_
   "",_
   " Input a new due date",_
   " offset as percentage",_
   " of theoretical flow",_
   " flow time","","  Z = Exit","")
 color 15,1
 locate 18,51 : print " Input a number or Z: " : call insound
 x$ = FNindata$(18,75,3) : call toupper(x$)
 do until (val(x$) > 0 or x$ = "Z") or x$ = "0"
   call failsound : x$ = FNindata$(18,75,3) : call toupper(x$)
 loop
 locate 18,77: color 0,3 : print " "
 if asc(x$) <> 90 then
   pdt$(21,1) = FNtrun$(x$)
 end if
 locate 22,40 : color 15,1 : print "   "
 locate 22,40 : color 4,7
 if len(pdt$(21,1)) = 1 then print " ";
 print pdt$(21,1)
 color 15,1
 b = 0
   case = 13
 call sidebox(" M. TARDINESS TOLERANCE",_
   "",_
   " Input a new",_
   " tardiness tolerance",_
   " measured in days",_
   "","    Z = Exit","","")
 color 15,1
 locate 18,51 : print " Input a number or Z: " : call insound
 x$ = FNindata$(18,75,2) : call toupper(x$)
 do until (val(x$) > 0 or x$ = "Z") or x$ = "0"
   call failsound : x$ = FNindata$(18,75,2) : call toupper(x$)
 loop
 if asc(x$) <> 90 then
   pdt$(22,1) = FNtrun$(x$)
 end if
 color 4,7
 locate 23,40
 if len(pdt$(22,1)) = 1 then print " ";
 print pdt$(22,1)
 color 15,1
 b = 0
   case = 14
 call sidebox(" N. CLEAR CRIT'Y LIST",_
   "",_
   "  Clear the Criticality",_
   "  List in order that",_
   "  the selected CR has",_
   "  no effect.","",_
   "  Z = Exit","")
 color 15,1
 locate 18,51 : print " Y or N required: " : call insound
 x$ = FNindata$(18,75,1) : call toupper(x$)
 do until (x$ = "Y" or x$ = "N") or (x$ = "Z")
   call failsound : x$ = FNindata$(18,75,1) : call toupper(x$)
 loop
```

```
    if asc(x$) <> 90 then
      pdt$(40,1) = "0"
      if x$ = "Y" then pdt$(40,1) = "1"
    end if
    color 4,7
    locate 24,40 : print " N";
    locate 24,40 : if val(pdt$(40,1)) = 1 then print " Y";
    color 15,1
    b = 0
      case = 83
    z = 10
    exit loop
      case = 82
    y$ = ctl$(15,1)
    z$ = ctl$(17,1)
    call pjtrest(y$+"~pdt."+z$,pdt$(),999)
    b = 101
      case = 90
    call scrclear
    z = 11
    exit loop
      case = 99
    exit loop
      case else
    call scrclear
    locate 10,10 : print "WHOOPS inside advance"
    klkkl$=input$(90)
    end select
  loop
end sub


sub simvar
  shared pdt$()

  b = 0
  do until b = 90
    select case b
      case = 0
    call fill(4,7,6,50,27,11)
    call shadow(0,3,6,50,27,11)
    color 14,7
    locate  7,52 : print "1. VARIABLE TO MONITOR"
    color 4,7
    locate  9,52 : print "1. Particular Action"
    locate 10,52 : print "2. Work Centre"
    locate 11,52 : print "3. Part"
    locate 12,52 : print "4. Final Product"
    locate 13,52 : print "5. All variables"
    locate 15,52 : print "Z. Exit"
    locate 18,52 : print " Select variable: "
    call insound
    x$ = FNindata$(18,74,1) : call toupper(x$)
    do until (val(x$) > 0 and val(x$) < 6) or (x$ = "Z")
      call failsound : x$ = FNindata$(18,74,1) : call toupper(x$)
    loop
    if asc(x$) <> 90 then
      pdt$(26,1) = x$
```

```
end if
b = val(x$) : color 15,3
  case = 1
call fill(15,1,6,50,27,16)
call shadow(0,3,6,50,27,16)
color 14,1
locate 7,55 : print "PARTICULAR ACTION"
x$ = "ABCDEFGHIJKL"
j = 1
for i = 9 to 20
  locate i,54 : print mid$(x$,j,1);"."
  incr j
next i
color 15,1
locate  9,58 : print "OQ > PQ Move"
locate 10,58 : print "OQ > CQ Move"
locate 11,58 : print "CQ > Re Load"
locate 12,58 : print "PQ > Re Load"
locate 13,58 : print "Re > OQ Unload"
locate 14,58 : print "Re > CQ Unload"
locate 15,58 : print "Re > PQ Unload"
locate 16,58 : print "Begin Set Up"
locate 17,58 : print "Begin Operating"
locate 18,58 : print "No Parts"
locate 19,58 : print "Move to Stock"
locate 20,58 : print "Jo > PQ Launch"

locate 23,50 : print " Action to be Monitored: ";
call insound
x$ = FNindata$(23,77,1) : call toupper(x$)
do until asc(x$) > 64 and asc(x$) < 77
  call failsound : x$ = FNindata$(23,77,1) : call toupper(x$)
loop
pdt$(27,1) = FNtrun$(str$(asc(x$)))
b = 90
  case = 2
call fill(14,7,18,50,26,5)
call shadow(0,3,18,50,26,5)
color 4,7
locate 20,52 : print " Resource number: "
call insound
x$ = FNindata$(20,71,2) : call toupper(x$)
do until val(x$) > 0
  call failsound : x$ = FNindata$(20,71,2) : call toupper(x$)
loop
pdt$(27,1) = x$
b = 90
  case = 3
  ' simu$(10,1) = "3"
call fill(14,7,18,50,26,5)
call shadow(0,3,18,50,26,5)
color 4,7
locate 19,52 : print " Input part name: "
call insound
x$ = FNindata$(21,60,8) : call toupper(x$)
pdt$(27,1) = x$
b = 90
  case = 4
```

```
      call fill(14,7,18,50,26,5)
      call shadow(0,3,18,50,26,5)
      color 4,7
      call insound
      locate 19,52 : print " Input product name: "
      x$ = FNindata$(21,60,8) : call toupper(x$)
'       do until val(x$) > 0
      pdt$(27,1) = x$
      b = 90
        case = 5
      pdt$(26,1) = "5"
      pdt$(27,1) = "AL"
      b = 90
        case = 81
      b = 90
        case else
      call scrclear
      locate 10,10 : print "WHOOPS in simver in ADV.PRO"
      end select
    loop
    call fill(15,3,6,50,29,18)
  end sub


  sub sidebox(title$,na1$,na2$,na3$,na4$,na5$,na6$,na7$,na8$)
    call fill(4,7,6,50,27,11)
    call shadow(0,3,6,50,27,11)
    color 14,7
    locate 7,52  : print title$
    color 4,7
    locate 9,52  : print na1$
    locate 10,52 : print na2$
    locate 11,52 : print na3$
    locate 12,52 : print na4$
    locate 13,52 : print na5$
    locate 14,52 : print na6$
    locate 15,52 : print na7$
    locate 16,52 : print na8$
  end sub


  sub welcome
    call scrclear
    r = 6
    call fill(15,1,r,20,40,13)
    call box (15,1,r,21,38,13)
    call shadow(0,3,r,20,40,13)
    color 15,1
    locate r+2,27  : print "       Welcome to the      "
    color 14,1
    locate r+4,27  : print "   MSA  S I M U L A T O R  "
    locate r+6,27  : print "           v 3.0           "
    color 15,1
    locate r+8,27  : print "       Copyright 1991      "
    locate r+10,27 : print "  S Hurley  Dr. J Driscoll "
    color 15,1
    i = 1
    x = 0.0275
```

```
    for i = 1 to 80
      y = 18
      z = 58
      locate y,z : print chr$(92); : delay x
      locate y,z : print chr$(179); : delay x
      locate y,z : print chr$(47);  : delay x
      locate y,z : print chr$(196); : delay x

      while instat
        x$ = input$(1)
        if asc(x$) = 27 then exit for
      wend
    next i
  end sub

  sub logiccheck
    $include "shared.pro"

    logic$ = ""

  '*** CHECK THAT ALL RESOURCES USED ARE DEFINED
    file$ = ctl$(15,1)+"~sqdb.inf"
    open file$ for input as #4
    totalmac = 0
    do until eof(4)
      input #4, x$
      if val(mid$(x$,21,5)) > totalmac then
        totalmac = val(mid$(x$,21,5))
      end if
    loop
    mac = 1
    do until mac$(mac,1) = ""
      incr mac
    loop
    if mac < totalmac then
      logic$ = "1"
    else
      logic$ = "A"
    end if
    close #4

  '*** CHECKS THAT ALL JOBS IN JOB$() ARE IN THE SEQUENCING DATABASE
    row = 0
    do until job$(row+1,1) = ""
      incr row
      if row = 20 then exit loop
    loop
    file$ = ctl$(15,1)+"~sqdb.inf"
    open file$ for input as #4
    holdjob$ = ""
    do until eof(4)
      input #4, x$
      x$ = mid$(x$,11,5)
      x$ = FNtrun(x$)
      if x$ <> holdjob$ then
        holdjob$ = x$
        for i = 1 to jobnum
      if job$(i,1) = "" then exit for
```

```
    if x$ = FNtrun$(job$(i,1)) then
      decr row
      if row = 0 then
        exit for
      end if
    end if
      next i
    end if
  loop
  close #4
  if row <> 0 then
    logic$ = logic$ + "2"
  else
    logic$ = logic$ + "B"
  end if

'*** INFORM THE USER OF THE RESULT
  call shadow(0,3,6,43,34,15)
  call fill(14,1,6,43,34,15)
  color 14,1
  locate 7,46  : print "PROJECT STATUS"
  color 15,1
  locate 9,46  : print "Project name ..... ";
  color 4,7 : print " "; : print using "\ \";ctl$(15,1);
    print " " : color 15,1
  locate 10,46 : print "Project version..... ";
  x$ = ctl$(16,1) : call toupper(x$)
  if left$(x$,1) = "I" then x$ = "Root"
    color 4,7 : print " "; : print using "\  \";x$; : color 15,1
  locate 13,46
  color 14,1 : print "LOGIC TEST RESULTS"
  color 15,1
  locate 15,46 : print "Resource Database - ";
  locate 15,69 : color 4,7
  if mid$(logic$,1) = "1" then
    print " FAIL "
  else
    print " PASS "
  end if
  color 15,1
  locate 17,46 : print "Sequencing Database - "
  locate 17,69 : color 4,7
  if mid$(logic$,2,1) = "2" then
    print " FAIL "
  else
    print " PASS "
  end if

  color 15,3 : locate 23,8 : print "                    "
  color 15,4
  locate 23,20 : print " <space bar> = continue, <esc> for Help "
  call insound
  x$ = input$(1)

  do until asc(x$) = 32 or asc(x$) = 27
    call failsound : x$ = input$(1)
  loop
  if asc(x$) = 27 then
```

```
    color 15,3
    locate 23,20 : print "                                          "
    call helplogic
  end if
  call fill(0,3,6,43,35,16)
  color 15,3
  locate 23,20 : print "                                          "
end sub      '*** statusbox

sub helplogic
  call shadow(0,3,7,6,32,15)
  call fill(4,7,7,6,32,15)
  color 14,7
  locate 8,7  : print "          LOGIC HELP"
  locate 10,7 : print "      Resource Database"
  color 4,7
  locate 11,7 : print "  FAIL - Not all resources"
  locate 12,7 : print "  required are defined."
  locate 13,7 : print "  Input more resources."
  color 14,7
  locate 15,7 : print "     Sequencing Database"
  color 4,7
  locate 16,7 : print "  FAIL - Not all jobs to be"
  locate 17,7 : print "  launched have a defined BoM."
  color 15,4
  locate 24,7 : print " Press <space bar> to continue ";
  call insound
  call spce
  color 15,3 : locate 24,7 : print "                                ";
  call fill(4,3,7,6,33,16)

end sub
```

M-SEQ.PRO

```
'     program M-SEQ.PRO

'     OBJECTIVES - carries out all the setting up of a simulation run

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"
$SEGMENT
$include "m-crsh.pro"
$include "m-file.pro"
$SEGMENT

call go

sub go
  locate 4,1 : print "m-seq.pr"
  $include "shared.pro"
  shared rescnt
  rescnt = 1
  do until mac$(rescnt,1) = ""
    incr rescnt
    if rescnt > resources then exit loop
  loop
  decr rescnt
  do until aa = aa + 1
    if z = 1 then
      call mdupdate
    '*** if a new file is created and the user goes straight into a
    '*** a simulation then mdupdate will dump the data to 001 and
    '*** all subsequent results go to this one.
    '*** The only changes to the new created files are what the user did
    '*** inside the editor in ADV.PRO.  If the user has loaded
    '*** 00x then results are simply passed back to the 00x file.
      call finwhen
      cr$(2,1) = pdt$(11,1)
      z = 2
    elseif z = 2 then
      call pdttomac
      call mincount
      call clearbnsch
      call resultcfg
      call getlsd
      call getwork
      call getsq
      call crsel(0)
      call periodadd
      call getbnsch
locate 5,1 : print "hello 4"
      call orderwork
      call openrun
      if pdt$(2,1) = "4" then call scrlsup
```

```
      z = 1
      chain "job.pbc"
    elseif z = 3 then        '*** iteration coming from m-iter.pro
      call mincount
      x% = val(pdt$(6,1))/val(pdt$(38,1)) : x% = fix(x!)
      if x! = x% then
    z = 3
    chain "m-job.pbc"
      end if
      z = 3
    elseif z = 3 then
      call clearbnsch
      call resultcfg
      call getlsd
      call getwork
      call getsq
      call crsel(0)
      call getbnsch
      call orderwork
      call openrun
      z = 1
      chain "job.pbc"
    else
      call failsound
      locate 11,11 : print "problems with z going into m-seq.pro"
      klklk$=input$(90)
    end if
  loop
end sub

' ***** SUBROUTINE BLOCK *****

sub periodadd
  shared ctl$(), pdt$(), sqnum
  pdt$(6,1) = FNtrun$( str$(val(pdt$(6,1)) + 1) ) '*** next period
  if val(pdt$(6,1)) < 10 then
    pdt$(6,1) = "00" + FNtrun$(pdt$(6,1))
  elseif val(pdt$(6,1)) < 100 then
    pdt$(6,1) = "0" + FNtrun$(pdt$(6,1))
  else
    pdt$(6,1) = FNtrun$(pdt$(6,1))
  end if
  call rblock
end sub

$SEGMENT

sub orderwork
'*** column 0 of work$() is used to put the order numbers in and then
'*** to use this to order the rows by swapping.
'*** the idea behind all this code is that the work file is first
'*** ordered wrt the bnsch$() file but the stragglers are kept in the
'*** same FCFS order.  Which may be important.

  $include "shared.pro"
  workjobs = 1
  do                 '*** clear numbers from col one
    work$(workjobs,0) = ""
```

```
      incr workjobs
    loop until work$(workjobs,1) = ""
    row = 1
    do until bnsch$(row,1) = ""          '*** put in the numbers.
      bnjob$ = FNtrun$(bnsch$(row,5))
      bnjno = val(bnsch$(row,4))
      for i = 1 to jobnum
        if bnjob$ = FNtrun$(work$(i,1)) and bnjno = val(work$(i,4)) then
      work$(i,0) = FNtrun$(str$(row))
      exit for
        end if
        IF I = JOBNUM THEN
      CALL SCRCLEAR
      LOCATE 10,10 : PRINT "i did become equal to jobnum"
      sf$=input$(1)
        END IF
      next i
      incr row
    loop

    i = 1                '*** give the ones not in bnsch$() numbers
    do until work$(i,1) = ""
      if work$(i,0) = "" then
        work$(i,0) = FNtrun$(str$(row))
        incr row
      end if
      incr i
    loop

  'locate 21,67 : print "hello 7"

    COUNT = 1
    row = 1
    do until work$(row,1) = ""
      count = count + 1
      IF COUNT = 1000 THEN
        CALL PJTDUMP("BNSCH",BNSCH$(),BNNUM,5,VAL(PDT$(6,1)))
        CALL PJTDUMP("WORK",WORK$(),JOBNUM,15,VAL(PDT$(6,1)))
        BEEP 3
        END
      END IF
      for i = row to jobnum
        if val(work$(i,0)) = row then
      for k = 0 to 15
        swap work$(i,k), work$(row,k)
      next k
      incr row
      exit for
        end if
      next i
    loop

locate 21,67 : print "hello 8"

end sub

sub pdttomac
  $include "shared.pro"
```

```
  shared rescnt

  '*** set up loading rule
  x = val(pdt$(20,1))
  select case x
    case = 1
      x$ = "000102030407080900010203050708091O"
    case = 2
      x$ = "000102030507080900010203040708091O"
    case = 3
      x$ = "00010203060708091O"
  end select
  y = val(pdt$(14,1))
  select case y
    case = 1
      y$ = "1"
    case = 2
      y$ = "2"
    case = 3
      y$ = "3"
    case = 4
      y$ = "4"
    case = 5
      y$ = "10"
  end select
  z = val(pdt$(24,1))
  select case z
    case = 1
      z$ = "010304"
    case = 2
      z$ = "020304"
  end select
  for i = 1 to rescnt
    if mac$(i,1) = "" then exit for
    mac$(i,8)  = x$
    mac$(i,15) = "10"    '   ' ' ' ' ' ' ' ' '''      y$
    mac$(i,9)  = z$
  next i
end sub      '*** pdttomac

sub mincount
  $include "shared.pro"
  pdt$(43,1) = pdt$(47,1)
  pdt$(44,1) = pdt$(48,1)
  pdt$(45,1) = pdt$(49,1)
  pdt$(46,1) = pdt$(50,1)
  count = 0
  a = val(pdt$(7,1))
  do until a < 7
    incr count
    a = a - 7
  loop
  pdt$(47,1) = FNtrun$(str$(val(pdt$(47,1)) + count))
  pdt$(48,1) = FNtrun$(str$(val(pdt$(48,1)) + a))
  for i = 47 to 48
    if val(pdt$(i,1)) < 10 then
      pdt$(i,1) = "00"+pdt$(i,1)
    else
```

```
      pdt$(i,1) = "0"+pdt$(i,1)
    end if
  next i

  a = val(pdt$(43,1))          '*** CALCULATING THE NUMBER OF MINS
  b = val(pdt$(47,1))          '*** TO SIMULATE
  mins& = 0
  for i = a to b-1
    for j = 1 to 7
      mins& = mins& + val(cal$(i,j))
    next j
  next i
  mins& = mins& * 60
  pdt$(5,1) = FNtrun$(str$(mins&))
end sub

sub finwhen
  $include "shared.pro"
  shared rescnt

  x = val(pdt$(6,1))
  y = val(pdt$(4,1))
'  y = val(pdt$(10,1))
  z = val(pdt$(39,1))
  pdt$(10,1) = FNtrun$(str$(x+y))'*** when to stop
  pdt$(35,1) = FNtrun$(str$(x+z))'*** when to start dumping


  for i = 1 to rescnt
    rltm$(10,i) = pdt$(5,1)      '*** machine time available
  next i
end sub

sub resultcfg
  shared Wkg$(), rltm$(), rltj$(), WkgR, resources, sqnum, rltg$(), jobnum
  shared rescnt
  for i = 1 to 9
    for j = 0 to 6
      Wkg$(i,j) = ""
    next j
  next i

  WkgR = 1

'*********************************************************
'*** INITIALISE RESULTS FILES FOR JOBS, MACHINES AND GLOBAL
  for j = 1 to 40
    rltg$(j,1) = "0"
  next j

  for i = 3 to 48
    for j = 1 to rescnt
      rltm$(i,j) = "0"
    next j
  next i

  row = 1                '*** If a job has not finished
  for i = 1 to jobnum            '*** processing by the end of a
```

```
    if rltj$(i,11) = "" then      '*** bucket then all its data
      for j = 0 to 22             '*** is carried over into the next
    rltj$(row,j) = rltj$(i,j)     '*** bucket.  But it is already saved
    if i <> row then rltj$(i,j) = ""'*** in this one, ideally it should
      next j              '*** not have been so.
      incr row
    end if
  next i
end sub '*** resultcfg


sub getlsd
  $include "shared.pro"

  for jobrow = 1 to jobnum
    do until (job$(jobrow,11) = "" or job$(jobrow,1) = "")
      incr jobrow
      if jobrow = jobnum + 1 then
    call scrclear : locate 10,10 : print "in getlsd"
    call insound  : call failsound : klk$=input$(10)
      end if
    loop
    if job$(jobrow,1) = "" then exit for

    open ctl$(15,1)+"-sqd.inf" for input as #1
      cp = 0
      lpt = 0
      qua = 0
      cpsetup = 0
      for sq = 1 to sqnum
    input #1, x$
    if FNtrun$(job$(jobrow,1)) = FNtrun$(mid$(x$,11,5)) and_
                          mid$(x$,46,1) = "F" then
      setup$ = mid$(x$,31,5) : nomac$ = mid$(x$,56,5)
      set2$ = mid$(x$,101,5)
      cpsetup = FNsetuptime(setup$,nomac$,set2$)
      cp = val(mid$(x$,47,4))      '*** tft, the x in other one
      lpt = val(mid$(x$,66,5))
      tba = val(job$(jobrow,3))
      tba$ = FNtrun$(str$(tba))
      count = 0
      if val(pdt$(16,1)) = 1 then         '*** 1 transfer = process
        tqu$ = tba$
      elseif val(pdt$(16,1)) = 2 then     '*** 2 if global
        tqu$ = FNtrun$(str$(val(pdt$(15,1)) * val(mid$(x$,18*5+1,5))))
      elseif val(pdt$(16,1)) = 3 then             '*** 3 product based
        tqu$ = FNtrun$(str$(val(job$(jobrow,10)) * val(mid$(x$,91,5))) )
      elseif val(pdt$(16,1)) = 4 then     '*** 4 half of job
        tqu =  val(job$(jobrow,3))    '*** of tba
        x = 0
        if tqu mod 2 = 1 then x = 1
        tqu = (tqu \ 2) + x
        tqu$ = FNtrun$(str$(tqu * val(mid$(x$,91,5))) )
      end if
      if val(tqu$) > val(tba$) then tqu$ = tba$
      tqu = val(tqu$)
```

```
        do until tba <= 0
          tba = tba - tqu
          incr count
        loop
        tba = tba + tqu : decr count
        cp = (tba * cp) + (count * tqu * lpt)
        cp = cp + cpsetup
        cp = cp + ((val(pdt$(21,1))/100) * cp)  '*** lead time offset
        exit for
      end if
        next sq
      close #1

      flag = 0
      W = val(job$(jobrow,5))
      D = val(job$(jobrow,6))

      for k = D to 1 step -1
        cp = cp - ( val(cal$(W,k))*60 )
        if cp <= 0 then
      job$(jobrow,11) = FNtrun$(str$(W))
      job$(jobrow,12) = Fntrun$(str$(k))
      flag = 1
      exit for
        end if
      next k

      if cp > 0 and W = 1 then
        job$(jobrow,11) = "1"
        job$(jobrow,12) = "1"
        flag = 1
      end if

      if flag <> 1 then
        flag = 0
        W = W - 1
        for j = W to 1 step -1
      for k = 7 to 1 step -1
      cp = cp - ( val(cal$(j,k))*60 )
        if cp <= 0 then
          job$(jobrow,11) = FNtrun$(str$(j))
          job$(jobrow,12) = Fntrun$(str$(k))
          flag = 1
          exit for
        end if
      next k
      if flag = 1 then exit for
        next j
      end if

      if flag = 0 and cp > 0 then
        job$(jobrow,11) = "1"
        job$(jobrow,12) = "1"
        flag = 1
      end if
    next jobrow
  end sub      '*** getlsd
```

```
  sub getwork
    $include "shared.pro"
    workrow = 1
    do until work$(workrow,1) = ""
      incr workrow
      if workrow = jobnum + 1 then
        call scrclear : locate 10,10 : print "in getwork"
        call failsound: klk$=input$(10)
      end if
    loop

    for jobrow = 1 to jobnum
      if job$(jobrow,1) = "" then exit for
      if val(job$(jobrow,11)) < val(pdt$(47,1)) then  '*** because next week
        for j = 0 to 15
      work$(workrow,j) = job$(jobrow,j)

      work$(workrow,13) = pdt$(6,1)'*** week number it is to be launched
                      '*** important in ordering bnsch.

      job$(jobrow,j) = ""           '***WIPES THE ROW OUT
        next j
        incr workrow
      end if
    next jobrow

    xrow = 1                '*** SHUNTING THE JOB$() FILE
    for i = 1 to jobnum
      if job$(i,1) <> "" then
        for j = 0 to 15
      job$(xrow,j) = job$(i,j)
      if xrow <> i then job$(i,j) = ""
        next j
        incr xrow
      end if
    next i
    pdt$(47,1) = pdt$(43,1)
    pdt$(48,1) = pdt$(44,1)
    pdt$(49,1) = pdt$(45,1)
    pdt$(50,1) = pdt$(46,1)
'********************************************** end of get WORK$() file

'CALL EDARRAY(work$(),jobnum,15,kk)

end sub '*** getwork


sub resultinit
  shared rltm$(), resources, pdt$(), sqnum
  shared rescnt

  for i = 1 to rescnt
    rltm$(1,i) = FNtrun$(str$(i))
    rltm$(2,i) = "I"
  next i
  for i = 3 to 49
    for j = 1 to rescnt
```

```
        rltm$(i,j) = "0"
      next j
    next i
    for i = 1 to rescnt
      rltm$(10,i) = pdt$(5,1)
    next i
  end sub      '*** resultinit


  sub lsup
    shared simu$(), sqnum
    z = 1
    simu$(20,1) = "2"
    chain "m-lsup.pbc"
  end sub


  sub mdupdate          '*** TO ADVANCE THE SIMULATION ONE PERIOD
    $include "shared.pro"
    z = 1
    x = val(ctl$(11,1))
    call toupper(ctl$(16,1)) : call toupper(ctl$(17,1))
    if ctl$(16,1) <> "INF"  then
      ctl$(17,1) = ctl$(16,1)
    elseif ctl$(16,1) = "INF" and ctl$(17,1) = "INF" then
      pjt$(x,2) = FNtrun$( str$( val(pjt$(x,2))+1 ) )
      if val(pjt$(x,2)) < 10 then
        pjt$(x,2) = "00" + pjt$(x,2)
      else
        pjt$(x,2) = "0" + pjt$(x,2)
      end if
      ctl$(17,1) = pjt$(x,2)
      call fill(15,7,18,49,27,5)
      call shadow(0,3,18,49,27,5)
      call waiton
      color 4,7
      locate 19,52 : print "Creating version: ";
      color 14,7 : print ctl$(17,1)
      color 4,7
      locate 21,52 : print "   from ROOT files"
      call openclose(ctl$(15,1),ctl$(17,1))
      call dump(ctl$(15,1),ctl$(17,1),0)
      call dumpho("f-pjt.inf",pjt$(),15,15)
    end if
  end sub

  sub openrun
    shared ctl$(), pdt$(), sqnum
    if pdt$(28,1) = "1" and pdt$(2,1) = "1" then
      open ctl$(15,1)+"~run."+ctl$(17,1) for append as #10
      RunFileHead$ = "Time            Res       Action      Part Prod  Job No"
      x$ = pdt$(6,1)+" PERIOD"
      write #10, x$
      write #10, RunFileHead$
    end if
  end sub

  sub scrlsup
```

```
  shared simu$()
  call fill(0,7,6,45,31,5)
  call shadow(0,3,6,45,31,5)
  color 4,7
  locate 7,48 : print "CR Selection Method:   ";
  color 14,7 : print FNtrun$(simu$(5,1))
  color 4,7
  locate 9,48 : print "Dispathing Rule:       ";
  color 14,7 : print FNtrun$(simu$(6,1))
end sub

sub clearbnsch
  shared bnsch$(), bnnum
  for i = 1 to bnnum
    if bnsch$(i,0) = "" then exit for
    for j = 0 to 5
      bnsch$(i,j) = ""
    next j
  next i
end sub
```

M-CRSH.PRO

```
'     program M-CRSH.PRO

'     OBJECTIVES - cr and sh selection methosd

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

sub crsel(b)
'*** no menu so can ask where ever we want to
  $include "shared.pro"
  bhold = b
  do
    select case b
    case = 0
      escape = 0
      b = val(pdt$(11,1))
      select case b
    case = 0
      exit loop
    case = 1
      x$ = "  MANUAL"
      if val(simu$(10,1)) = 1 then
        cr$(3,1) = simu$(11,1)
        pdt$(13,1) = cr$(3,1)
        escape = 1
      else
        simu$(10,1) = "1"
      end if
    case = 2
      x$ = "  RANDOM"
    case = 3
      x$ = "LOADING 1"
    case = 4
      if pdt$(2,1) = "4" then
        pdt$(13,1) = FNtrun$(left$(simu$(8,1),5))
        x = len(simu$(8,1))
        if x = 5 then
          simu$(8,1) = ""
        else
          simu$(8,1) = mid$(simu$(8,1),6,x-5)
        end if
        cr$(6,1) = pdt$(13,1)
        escape = 1
      end if
      if pdt$(2,1) = "1" then
        pdt$(2,1) = "2"
      elseif pdt$(2,1) = "2" then
        pdt$(2,1) = "1"
        escape = 1          '*** already done once
      end if
      x$ = "LOADING 2"
    case = 5
      if pdt$(2,1) = "4" then
        pdt$(13,1) = FNtrun$(left$(simu$(7,1),5))
        x = len(simu$(7,1))
```

```
      if x = 5 then
        simu$(7,1) = ""
      else
        simu$(7,1) = mid$(simu$(7,1),6,x-5)
      end if
      cr$(7,1) = pdt$(13,1)
      escape = 1
    end if
    if pdt$(2,1) = "1" then
      pdt$(2,1) = "2"
    elseif pdt$(2,1) = "2" then
      pdt$(2,1) = "1"
      escape = 1            '*** already done once
    end if
    x$ = "QUEUING 1"
    end select

    if escape = 1 then exit loop
    if (pdt$(2,1) <> "4" and pdt$(2,1) <> "5") then
  call fill(0,3,4,1,80,22)
  call fill(0,1,7,5,30,8)
  call shadow(0,3,7,5,30,8)
  color 15,1
  locate 8,12  : print "    Executing"
  locate 9,12  : print "CRITICAL RESOURCE"
  locate 10,12 : print "    Selection"
  locate 11,12 : print "    Method: ";
  color 14,1
  print pdt$(11,1)
  locate 13,16 : print x$
  color 15,1
  delay 1
    end if
    b = val(pdt$(11,1))
  case = 1                              '*** manual
    call fill(0,3,6,45,33,16)
    color 15,1
    color 0,3 : locate 23,8 : print "                          " : color 15,1
    if pdt$(2,1) = "4" then
  call fill(0,4,13,9,28,8)
  call shadow(0,3,13,9,28,8)
  color 7,4
  locate 15,15 : print "Select a Resource"
  locate 16,15 : print " for the Learning"
  locate 17,15 : print "  System Update."
  color 14,4
  locate 19,15 : print "  MANUAL CHOICE"
  color 15,1
    end if
    if TestCaseNo = 0 then
  x$ = FNinmac$
    else
  x$ = pdt$(13,1)
    end if
    if pdt$(2,1) = "4" then
  call fill(0,3,12,9,31,10)
  call fill(0,3,6,42,31,19)
  call fill(15,1,12,17,30,9)
```

```
  call shadow(0,3,12,17,30,9)
  color 15,1
  locate 14,22 : print " Executing Learning"
  locate 15,22 : print "System update.  This"
  locate 16,22 : print "may take a while, be"
  locate 17,22 : print "      patient."
  color 14,1
  locate 19,22 : print "    Please wait."
  color 15,1
    end if

    pdt$(13,1) = x$
    simu$(11,1) = x$
    cr$(3,1) = pdt$(13,1)

    b = 80
 case = 2                        '*** random
    row1 = 0
    do until mac$(row1+1,1) = ""
 incr row1
 if row1 = resources then exit loop
    loop
    x1% = FNrndint%(row1)
    pdt$(13,1) = FNtrun$(str$(x1%))
    cr$(4,1) = pdt$(13,1)
    delay 1
    b = 80
 case = 3                                '*** loading model
    row = 1
    do until mac$(row,1) = ""
 load&(row,0) = row
 load&(row,1) = 0
 incr row
 if row = resources + 1 then exit loop
    loop

    for i = 1 to sqnum
 if sq$(i,1) = "" then exit for
 macho$ = mid$(sq$(i,1),21,5)
 durho$ = mid$(sq$(i,1),36,5)
 tbaho$ = mid$(sq$(i,1),76,5)
 su1ho$ = mid$(sq$(i,1),96,5)
 load&(val(macho$),1) = (val(durho$) * val(tbaho$)) +_
          load&(val(macho$),1) + val(su1ho$)
    next i

    maxmach% = 1
    maxload& = load&(1,1)

    for i = 1 to resources
load&(i,0) = i
if maxload& < load&(i,1) then
    maxmach% = i
    maxload& = load&(i,1)
end if
    next i
    pdt$(13,1) = FNtrun$(str$(maxmach%))
    DELAY 2
```

```
  b = 80
  cr$(5,1) = pdt$(13,1)
case = 4                    '*** loading model 2
  b = 0
  call dumpho("pdt.hol",pdt$(),50,1)
  pdt$(25,1) = "4"
  row = 1
  do until mac$(row,1) = ""
mac$(row,15) = pdt$(14,1)
incr row
if row = resources then exit loop
  loop
                    '*** set the screen variable so that there is none eventually.
  pdt$(13,1) = "1"  '*** assigning a holding value to the bn so that the
        '*** the software works, but it has no effect on the
        '*** operation of the queuing model.
  b = 80          '*** Goes on back to M-SEQ
  if pdt$(2,1) <> "5" then pdt$(2,1) = "3"
           '*** 5 if doing a QM1 inside a LSUPdate
case = 5     '*** queuing model
  call dumpho("pdt.hol",pdt$(),50,1)
  pdt$(25,1) = "4"
  row = 1
  do until mac$(row,1) = ""
mac$(row,15) = pdt$(14,1)
incr row
if row = resources then exit loop
  loop
                    '*** set the screen variable so that there is none eventually.
  pdt$(13,1) = "1"  '*** assigning a holding value to the bn so that the
        '*** the software works, but it has no effect on the
        '*** operation of the queuing model.
  b = 80          '*** Goes on back to M-SEQ
  if pdt$(2,1) <> "5" then pdt$(2,1) = "2"
        '*** 5 if doing a QM1 inside a LSUPdate
case = 80         '*** exit
  b = 0
  exit loop
case = 260
  if val(pdt$(6,1)) < val(pdt$(36,1)) and val(simu$(18,1)) = 1 then
z = 31
chain "m-iter.pbc"
  end if

  if val(pdt$(6,1)) >= val(pdt$(36,1)) then
b = 261
  else
if val(simu$(18,1)) = 1 then
  z = 31
  chain "m-iter.pbc"
end if
z = 1
exit loop
  end if

case = 261
 call scrclear
 locate 10,10 : print "In the bit in m-crsh"
```

```
      dg$=input$(2)
      do
     x = val(simu$(19,1))
     if cr$(x+2,5) = "" then
       b = x + 64
       exit loop
     else
       simu$(19,1) = FNtrun$(str$((val(simu$(19,1)) + 1)))
     end if
       loop until val(simu$(19,1)) = 5
       if val(simu$(19,1)) = 5 then
     simu$(19,1) = "0"
     pdt$(11,1) = cr$(2,5)
     pdt$(13,1) = cr$(val(cr$(2,5)),5)
     b = 80
     y$ = ctl$(15,1)
     call edarray(cr$(),8,5,hj)
     z = 1
     if val(simu$(18,1)) = 1 then
       z = 31
       chain "m-iter.pbc"
     end if
       end if
       if val(pdt$(6,1)) < val(pdt$(36,1)) and val(simu$(18,1)) = 1 then
     z = 31
     chain "m-iter.pbc"
       end if
     case else
       color 0,3 : locate 23,8 : print "                          "
       color 15,1
       call shadow(0,3,6,43,34,9)
       call fill(15,4,6,43,34,9)
       call box(15,4,6,44,32,9)
       locate 8,45  : print "        INVALID INPUT       "
       locate 10,45 : print "      PRESS <space bar>     "
       locate 12,45 : print "         TO CONTINUE        "
       call failsound
       call fill(0,3,6,43,35,10)
       call spce
       color 15,1
       b = 0
     end select
  loop until abc = 90
end sub


def FNinmac$
  shared mac$(), resources
  icon = 1 : down = 1
  maxrow = 1
  do until mac$(maxrow,1) = ""
    if mac$(maxrow,1) <> "" then incr maxrow
    if maxrow = resources then
      incr maxrow
      exit loop
    end if
  loop
  decr maxrow
```

```
  call fill(15,1,6,42,30,18)
  call box(15,1,6,43,28,18)
  call shadow(0,3,6,42,30,18)
  color 14,1
  locate 7,48 : print "AVAILABLE MACHINES"
  locate 8,46 : print "m/c #"
  locate 8,56 : print "machine name"
  locate 23,44 : print " Use ";chr$(24);chr$(25);" <enter> to select "
  color 15,1

  color 31,1 : locate 10+icon-down,44 : print chr$(4) : color 15,1
  do
    for j = 1 to 12
      locate 9+j,46 : print "                          "
      locate 9+j,46 : print down+j-1
      locate 9+j,57 : print mac$(down+j-1,1)
    next j

123:
    do
      x$ = inkey$
    loop until len(x$) > 0
    if asc(mid$(x$,1,1)) = 13 then
      exit loop
    end if
    if len(x$) = 2 then
      x = asc(mid$(x$,2,1))
      select case x
        case = 79
          exit loop
        case = 80              '*** DOWN ARROW
          incr icon
          if icon = maxrow + 1 then
            decr icon
            call failsound
            exit select
          end if
      if icon < 13 then
            color 15,1
        color 31,1 : locate 9+icon,44 : print chr$(4)
        color 15,1 : locate 8+icon,44 : print " "
      elseif icon >= 13 then
            incr down
          end if
        case = 72              '*** UP ARROW
          decr icon
          if icon > 0 and down = 1 then
            color 15,1
        color 31,1 : locate 9+icon,44 : print chr$(4)
        color 15,1 : locate 10+icon,44 : print " "
          elseif down > 1 then
            decr down
          elseif icon = 0 then
            incr icon
            call failsound
          end if
        case else
```

```
            call failsound
        end select
      end if
      x$ = ""
    loop until X$ = "Z" or x$ = "z"
    FNinmac$ = FNtrun$(str$(icon))
end def
```

```
'     program M-QM1.PRO

'     OBJECTIVES - control program for the qm1 model for
'        - determining the location of the critical resource

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-file.pro"

call go

sub go
  $include "shared.pro"
  locate 4,1 : print "m-qm1.pr"
  select case z
    case 1
      qm1$ = FNlocationqm1$
      lm2$ = FNlocationlm2$
      call fill(15,7,14,25,35,5)
      call shadow(0,3,14,25,35,5)
      color 14,7
      locate 15,29 : print "Critical Resource from QM1"
      color 4,7
      locate 17,29 : print "    resource number: ";qm1$
      delay 2
      cr$ = pdt$(13,1)
      for i = 1 to sqnum
    if sq$(i,1) = "" then exit for
    sq$(i,1) = ""
      next i
      now& = 0
      y$ = ctl$(15,1) : z$ = ctl$(17,1)
      call arraywipe
      call restorefiles(y$,z$,999)
      call restorho("pdt.hol",pdt$(),50,1)
      cr$(2,1) = pdt$(11,1)
      pdt$(13,1) = qm1$
      cr$(6,1) = lm2$
      cr$(7,1) = qm1$

      pdt$(2,1) = "1"
      pdt$(0,0) = "In QM1"
      pdt$(2,1) = "2"
      z = 2
      chain "m-seq.pbc"
    case 3
      x$ = FNlocationqm1$
      simu$(7,1) = simu$(7,1) + x$ : call put5(simu$(7,1))
      simu$(15,1) = simu$(7,1)
      x$ = FNlocationlm2$
      simu$(8,1) = simu$(8,1) + x$ : call put5(simu$(8,1))
```

```
        simu$(16,1) = simu$(8,1)
        z = 3
        chain "m-lsup.pbc"
      case 2
        x$ = FNlocationqm1$
        simu$(7,1) = simu$(7,1) + x$ : call put5(simu$(7,1))
        x$ = FNlocationlm2$
        simu$(8,1) = simu$(8,1) + x$ : call put5(simu$(8,1))
        z = 2
        chain "m-seq.pbc"
      case 4
        qm1$ = FNlocationqm1$
        lm2$ = FNlocationlm2$
        call fill(15,7,14,25,35,5)
        call shadow(0,3,14,25,35,5)
        color 14,7
        locate 15,29 : print "Critical Resource from LM2"
        color 4,7
        locate 17,29 : print "    resource number: ";lm2$
        delay 2
        cr$ = pdt$(13,1)
        for i = 1 to sqnum
      if sq$(i,1) = "" then exit for
      sq$(i,1) = ""
        next i
        now& = 0
        y$ = ctl$(15,1) : z$ = ctl$(17,1)
        call arraywipe
        call restorefiles(y$,z$,999)
        call restorho("pdt.hol",pdt$(),50,1)
        cr$(2,1) = pdt$(11,1)
        pdt$(13,1) = lm2$
        cr$(6,1) = lm2$
        cr$(7,1) = qm1$
        pdt$(0,0) = "In LM2"
        pdt$(2,1) = "2"
        z = 2
        chain "m-seq.pbc"
      case else
        locate 10,10 : print "WHOOPS with z in m-qm1.pro"
        k1$=input$(16)
  end select
end sub

' ***** SUBROUTINE BLOCK *****

def FNlocationqm1$
  shared simu$(), rltm$(), resources, pdt$(), mac$(), cr$()

  x = val(rltm$(32,1))
  y = 1
  for i = 2 to resources
    if mac$(i,1) = "" then exit for
    if x < val(rltm$(32,i)) then
      x = val(rltm$(32,i))
      y = i
    elseif x = val(rltm$(32,i)) then              '*** using util'n to
      if val(rltm$(39,y)) < val(rltm$(39,i)) then     '*** break a tie.
```

```
      x = val(rltm$(32,i))
      y = i
        end if
      end if
    next i
    FNlocationqm1$ = FNtrun$(str$(y))
'   pdt$(13,1) = FNtrun$(str$(y))
'   cr$(7,1) = pdt$(13,1)

end def

def FNlocationlm2$
  shared simu$(), rltm$(), resources, pdt$(), mac$(), cr$()

'*** Really need to put some work in here to calculate the results.

  maxmach% = 1
  maxload% = val(rltm$(39,1))

  for i = 1 to resources
    if rltm$(1,i) = "" then exit for
    if maxload% < val(rltm$(39,i)) then
      maxmach% = i
      maxload% = val(rltm$(39,i))
    end if
  next i
  b = 80
  FNlocationlm2$ = FNtrun$(str$(maxmach%))
end def


sub sq1
  shared sq$(), sqnum
  call scrclear
  for i = 1 to 6
    print sq$(i,1)
  next i
  jh$=input$(1)
end sub
```

JOB.PRO

```
'     program JOB.PRO

'     OBJECTIVES - main simulation program

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"

LOCATE 21,1: PRINT "HELLO 7"

$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  locate 4,1 : print "job.pro "
  $include "shared.pro"
  if z = 1 then
    call jobloadup
  elseif z >= 2 and z <= 7 then
    call  JobLoadUp2
  end if
end sub

sub JobLoadUp                    '***PICKS OFF
  $include "shared.pro"
  call ScrBoxes
  z = 2
  call JobLoadUp2
end sub

sub JobLoadUp2                '***CALLS OFF THE SIMULATION STEPS
  $include "shared.pro"

  select case z
    case = 2
      chain "tooutque.pbc"
    case = 3
      chain "fininout.pbc"
    case = 4
      chain "movtoout.pbc
    case = 5
      chain "getjobs.pbc"
    case = 6
      chain "intores.pbc"
    case = 7
      chain "endofpas.pbc"
    case else
      call scrclear : locate 10,10 : print "whoops in z in job.pro"
      k1$=input$(19)
  end select
end sub '*** jobloadup2
```

```
sub sq1
  shared sq$(), now&
  for i = 1 to 9
    print sq$(i,1)
  next i
  print " "
  print "now&..";now&
end sub
```

ENDOFPAS.PRO

```
'     program ENDOFPAS.PRO

'     OBJECTIVES - end of pass through sim model, do again?

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  locate 4,1 : print "endofpas"
  $include "shared.pro"
  call endofpass
  if now& >= val(pdt$(5,1)) then
    z = 100
    chain "m-iter.pbc"
  end if
  z = 2
  chain "tooutque.pbc"
end sub

'***** SUBROUTINE BLOCK *****

sub endofpass
  $include "shared.pro"

  Now.Ho& = now&
  now& = 533000

  for i = 1 to resources          '*** WHAT IF NOWT
    if queues$(i,R) <> "" then                '*** IS WORKING GET NOW& WHEN
      x$ = left$(queues$(i,R),5)      '*** PUT IN FINISH TIMES
      x = val(x$) - 10000             '*** IN THE SIMULATOR
      call getque(x)
      if val(TTi$) > Now.Ho& and val(TTi$) < Now& then Now& = val(TTi$)
    end if
    if (rltm$(49,i) <> "" and rltm$(2,i) = "S") and (val(pdt$(18,1)) <> 1) then
      x = val(mid$(rltm$(49,i),2,4))
      call getque(x)
      if val(TTi$) > Now.Ho& and val(TTi$) < Now& then
    Now& = val(TTi$)
      end if
    end if
  next i

  flag = 0

  if now& >= val(pdt$(5,1)) then      '*** past end of bucket
    if now& = 533000 then
      flag = 1
```

```
      now& = val(pdt$(5,1))                '*** now& = end of bucket
      pdt$(8,1) = FNtrun$(str$(now.ho&))  '*** storing finish time.
    else
      pdt$(9,1) = "1"
      now& = val(pdt$(5,1))
      pdt$(8,1) = pdt$(5,1)      '*** GONE OVER THE END OF  A PERIOD
    end if
    if pdt$(25,1) = "1" then call statusline
  end if

  if flag <> 1 then

'*** Not evaluated if jump to end of period because all jobs are finished
'*** so that get a figure for only the time that jobs where available.

    diff& = now& - now.ho&
    for re = 1 to resources
      rltm$(12,re) = FNtrun$(str$(val(rltm$(12,re)) + 1))
      if queues$(re,inn) = "" then
    x1& = 0
      else
    x1& = FNprotime&(re,inn)
      end if
      rltm$(13,re) = FNtrun$(str$(val(rltm$(13,re)) + (x1& * diff&)))
      x# = val(rltm$(13,re))/now& : x# = round(x#,2)
      rltm$(14,re) = FNtrun$(str$(x#))
      if x1& > val(rltm$(15,re)) then rltm$(15,re) = FNtrun$(str$(x1&))
      x1% = len(queues$(re,Inn))/10 : x1% = round(x1%,2)
      rltm$(16,re) = FNtrun$(str$(val(rltm$(16,re)) + (x1% * diff&)))
      x# = val(rltm$(16,re))/now& : x# = round(x#,2)
      rltm$(17,re) = FNtrun$(str$(x#))
      if x1% > val(rltm$(18,re)) then rltm$(18,re) = FNtrun$(str$(x1%))

      if queues$(re,COMP) = "" then
    x2& = 0
      else
    x2& = FNprotime&(re,COMP)
      end if
      rltm$(19,re) = FNtrun$(str$(val(rltm$(19,re)) + (x2& * diff&)))
      x# = val(rltm$(19,re))/now& : x# = round(x#,2)
      rltm$(20,re) = FNtrun$(str$(x#))
      if x2& > val(rltm$(21,re)) then rltm$(21,re) = FNtrun$(str$(x2&))
      x2% = len(queues$(re,COMP))/10 : x2% = round(x2%,2)
      rltm$(22,re) = FNtrun$(str$(val(rltm$(22,re)) + (x2% * diff&)))
      x# = val(rltm$(22,re))/now& : x# = round(x#,2)
      rltm$(23,re) = FNtrun$(str$(x#))
      if x2% > val(rltm$(24,re)) then rltm$(24,re) = FNtrun$(str$(x2%))

      rltm$(25,re) = "0"
      rltm$(26,re) = "0"
      rltm$(27,re) = "0"
      rltm$(28,re) = "0"
      rltm$(29,re) = "0"
      rltm$(30,re) = "0"

      rltm$(31,re) = FNtrun$(str$(val(rltm$(13,re)) + val(rltm$(19,re))))
      x# = val(rltm$(31,re))/now& : x# = round(x#,2)
      rltm$(32,re) = FNtrun$(str$(x#))
```

```
        if x1&+x2& > val(rltm$(33,re)) then rltm$(33,re) = FNtrun$(str$(x1&+x2&))
        rltm$(34,re) = FNtrun$(str$(val(rltm$(16,re)) + val(rltm$(22,re))))
        x# = val(rltm$(34,re))/now& : x# = round(x#,2)
        rltm$(35,re) = FNtrun$(str$(x#))
        if x1% + x2% > val(rltm$(36,re)) then
      rltm$(36,re) = FNtrun$(str$(x1% + x2%))
        end if
      next re
    end if

    '*********************************************************************
    '*** OBSERVING THE AMOUNT OF JOBS IN
    '*** THE QUEUES BEFORE THE SIMULATION IS UPDATED TO THE NEXT EVENT

    if pdt$(25,1) = "1" then
      call perline
      call scr1anum
    end if
  '************************************************************
  '*** INPUT MORE JOBS OPTION

  '*** maybe for the probability distribution input of jobs.
  '************************************************************

    call ActTime
    if pdt$(25,1) = "1" then call statusline
    call statusline
    Intpt = 0
    if instat then                  '***INTERUPT HANDLING
      call insound
      X$=input$(1)
      x = asc(x$)
      if x = 27 then
        select case x
          case = 0
            exit select
          case = 1
            intpt = 1
        exit sub        '*** was EXIT LOOP
          case = 2
            intpt = 2
        exit sub        '*** was EXIT LOOP
          case else
        end select
      end if
    end if

end sub


def FNprotime&(re,Q)
  shared sq$()
  qh$ = "" : queue$ = ""
  if Q = 2 then       '*** input queue
    qh$ = queues$(re,Inn)
  elseif Q = 1 then
    qh$ = queues$(re,COMP)
  else
```

```
      call scrclear : locate 10,10 : print "WHOOPS in FNprotime"
      dkl$=input$(1)
    end if
                  '*** put all viable jobs into
    m = 1             '*** the queue$
    z$ = mid$(qh$,(m*10)-4,5)
    do
      z = val(z$) - 10000
      call getque(z)
      n = 1 : got = 0
      if val(req$) = 0 then
        call remstr(qh$,m)
        queue$ = queue$ + num$ + SNo$
        decr m
      else
        do
      y = instr(n,qh$,z$)
      x = y + 4
      if x mod 10 = 0 then
        call remstr(qh$,(y+4)/10)
        if y < m*10 and m <> 0 then decr m
        incr got

      else
        n = n + 4

      end if
        loop until y = 0
        if got = val(req$) then queue$ = queue$ + num$ + SNo$
      end if
      incr m
      z$ = mid$(qh$,(m*10)-4,5)
    loop until z$ = ""
    z& = 0
    for i = 1 to len(queue$)/10          '*** calculating duration
      x$ = mid$(queue$,i*10-8,4)                    '*** to make all batch.
      x% = val(mid$(sq$(val(x$),1),36,5))
      y% = val(mid$(sq$(val(x$),1),76,5))
      z& = (x% * y%) + z&
    next i
    FNprotime& = z&
  end def

  sub perline
    shared now&, pdt$()
    x = (now&*100)/val(pdt$(5,1))
    x = round(x,2)
    x = (x/100)*52
    x = round(x,2)
    color 7,1
    for i = 1 to x
      locate 23,25+i
      print chr$(219)
    next i
  end sub
```

FININOUT.PRO

```
'     program FININOUT.PRO

'     OBJECTIVES - the finish time in the out queue

'     Edition 2.0   Version 2.0   20/7/88

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  locate 4,1 : print "fininout"
  $include "shared.pro"
  for i = 1 to resources                    '***GOING DOWN FIN IN OUT
    if queues$(i,0) <> "" then
      call FinInOut(i)
    end if
  next i
  z = 4
  if val(simu$(20,1)) = 0 then
    chain "MovToOut.pbc"
  elseif val(simu$(21,1)) = 1 then
    chain "m-qm1.pbc"
  elseif val(simu$(22,1)) = 1 then
    chain "m-lsup.pbc"
  end if

end sub

' ***** SUBROUTINE BLOCK *****

sub FinInOut(i)
  $include "shared.pro"

  y$ = queues$(i,0)
  if y$ <> "" then
    j = 1
    x$ = mid$(y$,(10*j)-9,5)
    do
      x = val(x$) - 10000
      call getque(x)
      if FNtrun$(fin$) = "#" then
        z = val(mac$)
        call getres(z)
        prty$ = FNtrun$(SQO$)
        cases = 1
        FinOut$ = mid$(prty$,(2*cases)-1,2)
        do
          FinOut = val(FinOut$)
          select case FinOut
            case = 1
              call getque(x)
```

```
            fin$ = FNtrun$(str$(now&)) : call put5(fin$)
            call putque(x)
                exit loop
              case else
            call failsound : call insound : call scrclear
            locate 1,1 : print "fail in FININOUT"
            call edarray(mac$(),resources,15,kl) : x$ = input$(30)
            end select
            incr cases
            FinOut$ = mid$(prty$,(2*cases)-1,2)
          loop until FinOut$ = "  " or finout$ = ""
        end if
        incr j
        x$ = mid$(y$,(10*j)-9,5)
      loop until x$ = ""
    end if
end sub      '*** FinInOut
```

```
'     program GETJOBS.PRO

'     OBJECTIVES - jobs from work to gateway machines

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  locate 4,1 : print "getjobs"
  $include "shared.pro"

  call getjobs
  z = 6
  if val(simu$(20,1)) = 0 then
    chain "intores.pbc"
  elseif val(simu$(21,1)) = 1 then
    chain "m-qm1.pbc"
  elseif val(simu$(22,1)) = 1 then
    chain "m-lsup.pbc"
  end if

end sub

' ***** SUBROUTINE BLOCK *****

sub GetJobs          '*** LOADS JOBS INTO QUEUES - ie/ Initi10001
  $include "shared.pro"

  for i = 1 to resources
    if mac$(i,1) = "" then exit for
    if val(mac$(i,13)) = len(queues$(i,inn))/10 then
      mac$(i,3) = "B"
    end if
  next i
  for i = 1 to jobnum
    if work$(i,1) = "" then exit for
    if left$(work$(i,2),1) = "N" or work$(i,2) = "B" then  '*** not sched'd
      jobrow = i
      job$ = FNtrun$(work$(i,1))
      jobno$ = FNtrun$(work$(i,4))
      rownum = val(work$(i,14))
      if rownum = 0 then
    for j = 1 to sqnum
      querow = j
      if FNtrun$(mid$(sq$(j,1),11,5)) = job$ and_
         FNtrun$(mid$(sq$(j,1),66,5)) = jobno$ then exit for
    next j
    work$(i,14) = str$(querow)
      else
```

```
querow = val(work$(i,14))
  end if
  QueRowHold = QueRow
  call getque(querow)
  block = 0
  do              '*** is there room in all gateways
    if val(ini$) = 1 then
  x = val(mac$)
  inn$ = mac$(x,13)
  if val(inn$) = len(queues$(val(mac$),inn))/10 then
    block = 1
        work$(i,2) = "B"
        exit loop
      end if
end if
incr obser
su1tot = su1tot + val(su1$)
    incr querow
    call getque(querow)
    if num$ = "" then exit loop
  loop until FNtrun$(pro$) <> job$


  if block <> 1 then
movement = 1
    querow = querowhold
    call getque(querow)
    l = 1
    do
  if rltj$(l,1) = "" then exit loop
      incr l
loop until l = jobnum+1
rltj$(l,1) = FNtrun$(pro$)        '*** set up results file
rltj$(l,2) = FNtrun$(jno$)
rltj$(l,6) = FNtrun$(str$(su1tot))        '*** total setup time

rltj$(l,14) = FNtrun$(tqu$)       '*** transfer batch size
rltj$(l,15) = FNtrun$(str$(su1tot/obser)) '*** ave set up
y = val(mid$(num$,2,4))
do until mid$(sq$(y,1),46,1) = "F"
  incr y
loop
m = 1
do until val(jno$) = val(work$(m,4))
  incr m
loop
dayend$ = FNtrun$(cal$(val(work$(m,5)),val(work$(m,6))))
if len(dayend$) = 1 then dayend$ = "0"+dayend$
rltj$(l,12) = work$(m,5)+work$(m,6)+dayend$+work$(m,8)   '*** due date
x& = val(mid$(sq$(y,1),47,4))
rltj$(l,3) = FNtrun$(str$(x&)) '*** theoretical flow time
rltj$(l,9) = FNtrun$(work$(m,3))
    flag = 0 : once = 0
    do
  if (val(ini$) = 1 and FNtrun$(pro$) = job$) and FNtrun$(jno$) = jobno$ then
    queues$(val(mac$),Inn) = queues$(val(mac$),Inn) + "Initi" + num$
        srt$ = "0    "
        fin$ = "0    "
```

```
            call putque(querow)
          if once = 0 then
            act = 12 : call working
               work$(i,2) = "Y"
            end if
            once = 1
          end if
        incr querow

          call getque(querow)
          if num$ = "" then exit loop
          if Job$ = FNtrun$(pro$) and FNtrun$(jno$) = jobno$ then flag = 1
        loop until (Job$ <> FNtrun$(pro$) or FNtrun$(jno$) <> jobno$) and flag = 1
      end if
    end if
  next i
end sub '*** end GetJobs
```

INTORES.PRO

```
'     program INTORES.PRO

'     OBJECTIVES - from input queues into the resource

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  locate 4,1 : print "intores."
  $include "shared.pro"

  for i = 1 to resources
    call IntoRes(i)
  next i

  z = 7
  if val(simu$(20,1)) = 0 then
    chain "endofpas.pbc"
  elseif val(simu$(21,1)) = 1 then
    chain "m-qm1.pbc"
  elseif val(simu$(22,1)) = 1 then
    chain "m-lsup.pbc"
  end if
end sub


sub IntoRes(i)
  $include "shared.pro"
  shared curres
    escape = 0
    if val(pdt$(18,1)) <> 1 then
      if (queues$(i,R) = "" and rltm$(2,i) = "S") then
    x = val(mid$(rltm$(49,i),2,4))
    call getque(x)
    if val(tti$) = now& then
      srtres$ = "09"
      queue$ = rltm$(49,i)
      rltm$(49,i) = ""
    else
      exit sub
    end if

      elseif queues$(i,R) = "" then  '  and rltm$(2,i) = "I" then  '***TRUE IF NOT WKG
    call getres(i)
    if val(res$) = 0 then exit sub                '***NO MORE RES DEFINED
    prty$ = FNtrun$(SRe$)
    cases = 1
    SrtRes$ = mid$(prty$,(cases*2)-1,2)
```

```
  else
exit sub
  end if
else
  if queues$(i,R) = "" then   'and rltm$(2,i) = "I" then   '***TRUE IF NOT WKG
call getres(i)
if val(res$) = 0 then exit sub                '***NO MORE RES DEFINED
prty$ = FNtrun$(SRe$)
cases = 1
SrtRes$ = mid$(prty$,(cases*2)-1,2)
  else
exit sub
  end if
end if
  escape = 0
  do
    srtres = val(srtres$)
    select case srtres
  case = 0
    if rltm$(49,i) <> "" then          ' AND RLTM$(2,I) <> "S" then

          qh$ = queues$(i,Inn) + queues$(i,COMP)
      z$ = mid$(rltm$(49,i),1,5)
      z = val(z$) - 10000
      call getque(z)
          n = 1 : got = 0
      if val(req$) = 0 then
    num = 1 : flag = 0
    do
      y = instr(num,qh$,z$)
      x = y + 4
      if x mod 10 <> 0 then
        num = num + 4
      elseif x mod 10 = 0 then
        flag = 1
        exit loop
      end if
    loop until y = 0

    if flag = 0 then
      escape = 1
      exit select
    end if

          else
    n = 1
    do
      y = instr(n,qh$,z$)
      if y <> 0 then
        x = y + 4
        if x mod 10 = 0 then
          incr got
          n = y + 10
        else
          n = y + 5
        end if
      end if
           loop until y = 0
```

```
    if got = val(req$) then
      queue$ = mid$(rltm$(49,i),1,10)
      rltm$(49,i) = ""
            else
              escape = 1
              exit select
            end if
          end if
          x = instr(SRe$,"09")
          cases = (x+1)/2
        else
          incr cases
        end if
  case = 1
    if queues$(i,Inn) = "" and queues$(i,COMP) = "" then exit loop
        incr cases
      case = 2
    qh$ = ""
        incr cases
      case = 3
    queue$ = ""
        incr cases
  case = 4
    if queues$(i,Inn) = "" then
      cases = cases + 4
    else
      qh$ = queues$(i,Inn)
      incr cases
    end if
  case = 5
    if queues$(i,COMP) = "" then
      cases = cases + 4
    else
      qh$ = queues$(i,COMP)
      incr cases
    end if
  case = 6
    qh$ = queues$(i,Inn) + queues$(i,COMP)
        incr cases
  case = 7                  '*** put all viable jobs into
    m = 1                   '*** the queue$
        z$ = mid$(qh$,(m*10)-4,5)
    do
          z = val(z$) - 10000
          call getque(z)
      n = 1 : got = 0
          if val(req$) = 0 then
            call remstr(qh$,m)
            queue$ = queue$ + num$ + SNo$
            decr m
          else
            do
      y = instr(n,qh$,z$)
      x = y + 4
      if x mod 10 = 0 then

                call remstr(qh$,(y+4)/10)
                if y < m*10 and m <> 0 then decr m
```

```
            incr got

    elseif x mod 10 <> 0 then
      n = n + 4
    end if
         loop until y = 0
         if got = val(req$) then queue$ = queue$ +_
                                              num$ + SNo$
       end if
       incr m
       z$ = mid$(qh$,(m*10)-4,5)
     loop until z$ = ""
  incr cases
  if queues$(i,R) <> "" then   exit loop   '***TRUE IF NOT WKG
case = 8
  if queue$ = "" then
    cases = cases + 2
  else
    curres = i
    call rules(queue$)
    incr cases
  end if
case = 9              '*** remove the job from the
  x = val(mid$(queue$,2,4))
  call getque(x)
  if (rltm$(2,i) = "I" and val(pdt$(18,1)) <> 1) AND_
                      val(su1$) <> 0 then
    act = 8 : call working
    call util(i,"S")
    rltm$(49,i) = queue$
    x = val(mid$(queue$,2,4))
    call getque(x)
    tti$ = FNtrun$(str$(now& + val(su1$) ))
    call put5(tti$)
    call putque(x)
    exit sub
  else
    remov$ = "" : got = 0 : num = 1        '*** relevent queue
    qhold$ = ""
    qhold$ = queues$(i,Inn) + queues$(i,COMP)
    queues$(i,R) = queue$
    z$ = mid$(queue$,2,4)
    call getque(val(z$))
    if val(tqu$) > val(tba$) - val(tpr$) then
  tquho$ = str$(val(tba$) - val(tpr$))
    else
  tquho$ = tqu$
    end if
    parho$ = FNtrun$(par$)
    cptho$ = FNtrun$(cpt$)

    movement = 1
    do
  y = instr(num,qhold$,left$(queue$,5))
  x = y + 4
  if x mod 10 <> 0 then
    num = num + 4
  elseif y <> 0 then
```

```
          num = y + 5 : incr got
          z$ = mid$(qhold$,y-5,5)
          if left$(z$,1) = "I" then

'*** processing to put the time it was started into the jobs results file.
            z1 = val(mid$(qhold$,y+1,4))
              jobrow = 1
              do until val(rltj$(jobrow,2)) = val(mid$(sq$(z1,1),66,5))
            incr jobrow
              loop
              if rltj$(jobrow,10) = "" then
            rltj$(jobrow,10) = pdt$(47,1)+pdt$(48,1)+_
                        pdt$(49,1)+pdt$(50,1)
              end if
'*** finish putting the time that the job started into the database

              remov$ = "Initi"+left$(queue$,5)
              exit loop
            end if
            z = val(z$) - 10000
            call getque(z)
            if FNtrun$(par$) = parho$ then
              tba$ = fntrun$(str$(val(tba$) - val(tquho$)))
              tpr$ = fntrun$(str$(val(tpr$) - val(tquho$)))
            else
              TTtqu$ = tqu$
              if val(tqu$) > val(tba$) then '  - val(tpr$) then
                TTtqu$ = str$(val(tba$)   )   '     - val(tpr$))
              end if
              tba$ = fntrun$(str$(val(tba$) - val(TTtqu$)))
              tpr$ = fntrun$(str$(val(tpr$) - val(TTtqu$)))
            end if
            call put5(tpr$) : call put5(tqu$) : call put5(tba$)
            if val(tpr$) = 0 then remov$ = remov$ + mid$(qhold$,y-5,10)
            call PutQue(z)
          end if
            loop until y = 0

            x$ = left$(queue$,5)
            x = val(x$) - 10000
            call getque(x)
            hold$ = tqu$
            if val(tqu$) >= val(tba$) - val(tpr$) then_
           hold$ = str$(val(tba$) - val(tpr$))
            tti$ = str$(now& + (val(hold$)*val(dur$)))
            tqu$ = fntrun$(tqu$)
            tti$ = FNtrun$(tti$)
            call put5(tqu$)
            call put5(tti$)
            call putque(x)
            act = 4 : call working
            act = 9 : call working
            call util(i,"W")
            l = 1
            if remov$ <> "" then
          z$ = mid$(remov$,(l*10)-9,10)
          do
            y = instr(queues$(i,Inn),z$)
```

```
          if y <> 0 then
            call remove(queues$(),i,Inn,(y+9)/10)
            incr l
          else
            y = instr(queues$(i,COMP),z$)
            if y <> 0 then
              call remove(queues$(),i,COMP,(y+9)/10)
              incr l
            end if
          end if
          z$ = mid$(remov$,(l*10)-9,10)
        loop until z$ = "" or y = 0
          end if
        end if
        exit loop
      case = 10
        exit loop
      case = 14              '*** JIT one
        if val(outt$) = len(queues$(i,0))/10 then
          call util(i,"B")
          escape = 1
          exit loop
        end if
        incr cases
      case else
        call scrclear
        locate 1,11 : print "fail intores"
        call edarray(mac$(),resources,15,kl) : x$ = input$(30)
    end select
    srtres$ = mid$(prty$,(cases*2)-1,2)
    if escape = 1 then  exit loop
      loop until (srtres$ = "" or escape = 1) or queues$(i,R) <> ""
end sub '*** end IntoRes
```

MOVTOOUT.PRO

```
'     program MOVTOOUT.PRO

'     OBJECTIVES - moving from the out queue to in or remove from system

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  $include "shared.pro"
  locate 4,1 : print "movetoou"
  for i = 1 to resources                    '***GOING DOWN FIN IN OUT
    if queues$(i,0) <> "" then
      call movetoout(i)
    end if
  next i
  z = 5
  if val(simu$(20,1)) = 0 then
    chain "getjobs.pbc"
  elseif val(simu$(21,1)) = 1 then
    chain "m-qm1.pbc"
  elseif val(simu$(22,1)) = 1 then
    chain "m-lsup.pbc"
  end if
end sub


sub MoveToOut(i)
  $include "shared.pro"
    escape = 0
    y$ = queues$(i,0)
    if y$ <> "" then
      j = 1
      x$ = mid$(y$,(10*j)-9,5)
      do
        x = val(x$) - 10000
    call getque(x)
        if val(fin$) = now& or FNtrun$(fin$) = "B" then
          z = val(mac$)
          call getres(z)
          prty$ = FNtrun$(FQO$)
          cases = 1
      moveout$ = mid$(prty$,(2*cases)-1,2)
          do
            moveout = val(moveout$)
            select case moveout
          case = 1                          '***IF FINISHED WHAT

        if val(fin$) = now& or FnTrun$(fin$) = "B" then
```

```
                if left$(suc$,1) = "F" then
            for k = 1 to jobnum
              if FnTrun$(Pro$) = rltj$(k,1) and_
                  val(FnTrun$(JNo$)) = val(rltj$(k,2)) then
'*** these three once had .Ho extensions
              Srt$ = "#    "
              Fin$ = "#    "
              Obt$ = "0    "
            call putque(x)
            act = 11 : call working
                       if val(TBa$) = val(TPr$) then
              rltj$(k,11) = pdt$(47,1)+pdt$(48,1)+_
                    pdt$(49,1)+pdt$(50,1)
              x = FNmins(rltj$(k,11),rltj$(k,12))
              rltj$(k,5) = FNtrun$(str$(x))          '*** TARDY
              x = FNmins(rltj$(k,11),rltj$(k,10))
              rltj$(k,4) = FNtrun$(str$(x))
              x& = val(rltj$(k,4)) - val(rltj$(k,3))
              rltj$(k,13) = FNtrun$(str$(x&))
              l = 1
              do until FNtrun$(pro$) = FNtrun$(work$(l,1))_
                   and val(jno$) = val(work$(l,4))
                incr l
              loop
              work$(l,2) = "Finish"
                         end if
                       exit for
                     end if
                   next k
                   call remove(queues$(),i,0,j)
                   decr j
                   escape = 1
                   exit loop
          end if
          incr cases
               end if
          case = 2
               x$ = mid$(y$,(10*j)-4,5)
               x = val(x$) - 10000
               call getque(x)
               w = val(mac$)
        call getres(w)
        if val(req$) > 1 then
          compt = 1
        else
          compt = 0
        end if
        if compt = 0 then
          if val(inn$) > len(queues$(w,Inn))/10 then
            y = instr(queues$(w,Inn),mid$(y$,(j*10)-9,10))
            if y = 0 then queues$(w,Inn) =_
                 queues$(w,Inn) + mid$(y$,(j*10)-9,10)
            act = 1 : call working
            movement = 1
          else
            incr cases       '*** BLOCKED
            exit select
          end if
```

```
        else
          if val(compo$) > len(queues$(w,COMP))/10 then
            y = instr(queues$(w,COMP),mid$(y$,(j*10)-9,10))
            if y = 0 then queues$(w,COMP) =_
                 queues$(w,COMP) + mid$(y$,(j*10)-9,10)
            act = 2 : call working
            movement = 1
          else
            incr cases       '*** BLOCKED
            exit select
          end if
        end if
        x$ = mid$(y$,(10*j)-9,5)
        x = val(x$) - 10000
        call getque(x)
        if compt = 1 then
          srt$ = "#    " : fin$ = "#    "
        else
          srt$ = "0    " : fin$ = "0    "
        end if
        call putque(x)
        call remove(queues$(),val(mac$),0,j)
        decr j
        escape = 1
        exit loop
        x$ = mid$(y$,(10*j)-9,5)
        x = val(x$) - 10000
        call getque(x)
          case = 3
        j = 1
        x$ = mid$(y$,(10*j)-9,5)
        x = val(x$) - 10000
        call getque(x)
        srt$ = "B    "
               fin$ = "B    "
        call putque(x)
               escape = 1
               exit loop
             case else
        call failsound : call insound : call scrclear
        locate 1,1 : print "WHOOPS in movtoout" : x$ = input$(30)
           end select
           moveout$ = mid$(prty$,(2*cases)-1,2)
         loop until moveout$ = ""
       end if
       incr j
       x$ = mid$(y$,(10*j)-9,5)
     loop until x$ = "" or escape = 1
   end if
end sub      '*** end MOVEOUT


def FNmins(fir$,sec$)
  shared cal$()

  if val(fir$) >= val(sec$) then
    sign = 1
    sw = val(mid$(sec$,1,3))
```

```
    sd = val(mid$(sec$,4,2))
    sh = val(mid$(sec$,6,2))
    sm = val(mid$(sec$,8,2))
    fw = val(mid$(fir$,1,3))
    fd = val(mid$(fir$,4,2))
    fh = val(mid$(fir$,6,2))
    fm = val(mid$(fir$,8,2))
  else
    sign = -1
    sw = val(mid$(fir$,1,3))
    sd = val(mid$(fir$,4,2))
    sh = val(mid$(fir$,6,2))
    sm = val(mid$(fir$,8,2))
    fw = val(mid$(sec$,1,3))
    fd = val(mid$(sec$,4,2))
    fh = val(mid$(sec$,6,2))
    fm = val(mid$(sec$,8,2))
  end if
  if sw = fw and sd = fd then'*** where days and hr
    if sh = fh then              '*** are the same
      mins = fm - sm
    else         '*** where day same
      mins = 60 - sm             '*** hour different
      sh = sh + 1
      hour = fh - sh
      mins = mins + (60*hour)
      mins = mins + fm
    end if
  else           '*** everyting different
    mins = 60 - sm
    sh = sh + 1
    hour = val(cal$(sw,sd)) - sh
    mins = (60*hour) + mins
    sd = sd + 1
    if sd = 8 then
      sw = sw + 1 : sd = 1
    end if
    do until sd = fd and sw = fw
color 6,7
locate 1,1
print "sw...";sw
print "sd...";sd
print "fir$..";fir$
print "sec$..";sec$
      mins = val(cal$(sw,sd))*60 + mins
locate 1,1
print "                "
print "m                "
print "                 "
print "                  "
      sd = sd + 1
      if sd = 8 then
    sd = 1
    sw = sw + 1
      end if
    loop
    mins = (fh*60) + fm + mins
  end if
```

```
  FNmins = sign * mins
end def
```

TOOUTQUE.PRO

```
'     program TOOUTQUE.PRO

'     OBJECTIVES - move the job form the resource to the out queue

'     Edition 2.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley & Dr J Driscoll

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"

call go

sub go
  $include "shared.pro"
  locate 4,1 : print "tooutque"
  for i = 1 to resources                    '***FIN IN mc
    if queues$(i,R) <> "" then
      call ToOutQue(i)
    end if
  next i
  z = 3
  if val(simu$(20,1)) = 0 then
    chain "FinInOut.pbc"
  elseif val(simu$(21,1)) = 1 then
    chain "m-qm1.pbc"
  elseif val(simu$(22,1)) = 1 then
    chain "m-lsup.pbc"
  end if
end sub

' ***** SUBROUTINE BLOCK *****

sub ToOutQue(i)
  $include "shared.pro"

    y$ = queues$(i,R)
    if queues$(i,R) <> "" then                          '***TRUE IF NOT WORKING
      x$ = left$(queues$(i,R),5)
      x = val(x$) - 10000
      call getque(x)
      if val(TTi$) = now& or left$(fin$,1) = "B" then     '***TRUE IF JOB FIN
        z = val(mac$)
        call getres(z)
        prty$ = FNtrun$(FiR$)
        cases = 1
    if mid$(queues$(i,R),6,1) = "F" then prty$ = "010304"
    ToOut$ = mid$(prty$,(2*cases)-1,2)
        do
          ToOut = val(ToOut$)
          select case ToOut
        case = 1                    '*** TO OUTPUT QUEUE
             if len(queues$(i,0))/10 < val(outt$) then
               call getque(x)
```

```
remov$= ""
        y = instr(queues$(i,0),queues$(i,R))
        if y = 0 then queues$(i,0) = queues$(i,0) + queues$(i,R)
if val(tqu$) > val(tba$) - val(tpr$) then
  tpr$ = tba$
else
  tpr$ = FNtrun$(str$(val(tpr$) + val(tqu$)))
end if
srt$ = FNtrun$(str$(now&)) : call put5(srt$)
call put5(tpr$) : fin$ = "#    "
call putque(x)
cases = cases + 1    '*** because only looking at 010304
  else
cases = cases + 2
  end if
case = 2             '*** TO INPUT QUEUE
  x1$ = mid$(queues$(i,R),6,5)
  x1 = val(x1$) - 10000
  call getque(x1)
  z1 = val(mac$)
  call getres(z1)

  if val(req$) = 1 then
if len(queues$(z1,Inn))/10 < val(inn$) then
  remov$= ""
  y = instr(queues$(z1,Inn),queues$(i,R))
  if y = 0 then queues$(z1,Inn) = queues$(z1,Inn)+ queues$(i,R)
  call getque(x)
  TPr$ = str$(val(TPr$) + val(TQu$)) : call put5(tpr$)
  srt$ = FNtrun$(str$(now&)) : call put5(srt$)
  fin$ = "#    "
  call putque(x)
  cases = cases + 1
else
  cases = cases + 2
end if
  else
if len(queues$(z1,COMP))/10 < val(compo$) then
  remov$= ""
  y = instr(queues$(z1,COMP),queues$(i,R))
  if y = 0 then queues$(z1,COMP) = queues$(z1,COMP) + queues$(i,R)
  call getque(x)
  TPr$ = str$(val(TPr$) + val(TQu$)) : call put5(tpr$)

  srt$ = FNtrun$(str$(now&)) : call put5(srt$)
  fin$ = "#    "
  call putque(x)
  cases = cases + 1
else
  cases = cases + 2
end if
  end if

case = 3         '*** LOAD NEXT TRANSFER BATCH
  call getque(x)
  if val(tqu$) > val(tba$) - val(tpr$) then
tquho$ = str$(val(tba$) - val(tpr$))
  else
```

```
        tquho$ = tqu$
          end if
          parho$ = FNtrun$(par$)
          cptho$ = FNtrun$(cpt$)
          if val(TPr$) = val(TBa$) then '   and val(tba$) = 0 then
        queues$(i,R) = ""
        act = 5 : call working
        call util(i,"I")
        movement = 1
        exit loop
          end if
          if val(ini$) = 1 then
        HOLD$ = TQU$
        if val(TQu$) > val(TBa$) - val(TPr$) then_
                  HOLD$ = str$(val(TBa$) - val(TPr$))
           TTi$ = fntrun$(str$(now& + val(HOLD$)*val(Dur$)))
        call put5(tqu$)
        call put5(tti$)
        call PutQue(x)
        act = 9 : call working
        call util(i,"W")
        movement = 1
        exit loop
          end if
          n = 1 : got = 0
          qh$ = queues$(i,Inn) + queues$(i,COMP)
          do
        y = instr(n,qh$,left$(queues$(i,R),5))
        x1 = y + 4
        if x1 mod 10 <> 0 then
          n = n + 4
        elseif x1 mod 10 = 0 then
          n = y + 5
          incr got
        end if
          loop until y = 0
          if got <> val(req$) then
        rltm$(49,i) = queues$(i,R)   '*** store the old set up
        queues$(i,R) = ""
        TTi$ = "0    "
        call PutQue(x)
        act = 10 : call working
        call util(i,"N")
        exit loop
          else
        call PutQue(x)
        got = 0 : n = 1
        do
          y = instr(n,qh$,left$(queues$(i,R),5))
          x1 = y + 4
          if x1 mod 10 <> 0 then
            n = n + 4
          elseif x1 mod 10 = 0 then
            n = y + 5 : incr got
            z$ = mid$(qh$,y-5,5)
            z = val(z$) - 10000
            call getque(z)
'*** at this point we are trying to get the ones removed from the
```

```
'*** input queue.
          if FNtrun$(par$) = parho$ then
            tba$ = fntrun$(str$(val(tba$) - val(tquho$)))
            tpr$ = fntrun$(str$(val(tpr$) - val(tquho$)))
          else
            TTtqu$ = tqu$
            if val(tqu$) > val(tba$) then
              TTtqu$ = tba$
            end if
            tba$ = FNtrun$(str$(val(tba$) - val(TTtqu$)))
            tpr$ = FNtrun$(str$(val(tpr$) - val(TTtqu$)))
          end if
            call put5(tpr$) : call put5(tba$)
            if val(TPr$) = 0 then remov$ = remov$ + mid$(qh$,y-5,10)
            call PutQue(z)
          end if
        loop until y = 0
          end if
          call getque(x)
          hold$ = tqu$
          if val(tqu$) >= val(tba$) - val(tpr$) then_
                  hold$ = str$(val(tba$) - val(tpr$))
          tti$ = str$(now& + val(hold$)*val(dur$))
          call put5(tqu$) : call put5(tti$)
          call PutQue(x)
          act = 5 : call working
          l = 1
          if remov$ <> "" then
        z$ = mid$(remov$,(l*10)-9,10)
        do
          y = 0
          y = instr(queues$(i,Inn),z$)
          if y <> 0 then
            call remove(queues$(),i,Inn,(y+9)/10)
            incr l
          else
            y = instr(queues$(i,COMP),z$)
            if y <> 0 then
              call remove(queues$(),i,COMP,(y+9)/10)
              incr l
            end if
          end if
          z$ = mid$(remov$,(l*10)-9,10)
        loop until z$ = "" or y = 0
          end if
          exit loop
        case = 4                    '***BLOCKS M/C
          fin$ = Fntrun$(fin$)
          fin$ = "B" + fin$ : call put5(fin$)
          call putque(x)              'IS X AGAIN CORRECT
          call util(i,"B")
          exit loop
        case = 12
             if len(queues$(i,0))/10 < val(Outt$) then
               queues$(i,0) = queues$(i,0) + queues$(i,R)
               queues$(i,R) = ""
               call getque(x)                    'IS THE X CORRECT
                 srt$ = FNtrun$(str$(now&)) : call put5(srt$)
```

```
                fin$ = "#    "
              call putque(x)
        act = 5 : call working
              movement = 1
        call util(i,"I")
              x$ = mid$(y$,6,5)
              x$ = left$(y$,5)
              x = val(x$) - 10000
              call getque(x)
              exit loop
            end if
          case else
            call failsound : call insound
          locate 1,1 : call scrclear :print "fail in TOUTQUE ...";I
          call edarray(mac$(),resources,15,kl) : x$ = input$(30)
          end select
          ToOut$ = mid$(prty$,(2*cases)-1,2)
        loop until ToOut$ = "  " or toout$ = ""
      end if
    end if
end sub '*** end ToOut
```

M-ITER.PRO

```
'     program M-ITER.PRO

'     CONTENTS - at the end of a period deals with move to next

'     Edition 1.0   Version 2.0   22/2/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley  and  Dr J Driscoll

'*************************************************************************
'***                                                                   ***
'***                                                                   ***
'***     this program                                                  ***
'***       at the and of a period this dumps all the                   ***
'***       files if required and sets up the cloack and                ***
'***       work$() file for the next period and moves on               ***
'***       to simulate this or goes back to M-ADV.PRO                  ***
'***                                                                   ***
'***                                                                   ***
'*************************************************************************

$compile chain
$include "common.pro"
$include "m-com.pro"
$include "m-simcom.pro"
$SEGMENT
$include "m-file.pro"

call go

sub go
  locate 4,1 : print "m-iter.p"
  $include "shared.pro"
  do until a = a + 1
    if z = 100 then      '*** from endofpas
      select case val(pdt$(2,1))
    case 1           '*** normal simulation
      if pdt$(9,1) = "1" then call overun
      call endbucket
      call iterdump
      if val(pdt$(6,1)) = val(pdt$(10,1)) then    '*** any more periods
        simu$(10,1) = "0"    '*** used to avoid doing manual again
        if TestCaseNo <> 0 then
          z = 3
          chain "test.pbc"
        else
          z = 1
          chain "m-ctl.pbc"
        end if
      else
        z = 2
        chain "m-seq.pbc"
      end if
    case 2           '*** QM1
      call globalcalc
      z = 1
      chain "m-qm1.pbc"
```

```
    case 3          '*** LM2
      call globalcalc
      z = 4
      chain "m-qm1.pbc"
    case 4
      if pdt$(9,1) = "1" then call overun
      call endbucket     '*** to tidy up files but no dump
      z = 4
      chain "m-lsup.pbc"
    case 5
      if pdt$(9,1) = "1" then call overun
      call endbucket
      if val(pdt$(6,1)) = val(pdt$(10,1)) then    '*** any more periods
        z = 3
      else
        z = 2
      end if
      chain "m-qm1.pbc"
    case else
      call scrclear : locate 10,10
      print "WHOOPS going into m-iter.pro" : x$ = input$(50)
      end select
    end if
  loop
end sub

' ***** SUBROUTINE BLOCK *****

sub overun                              '*** to reset the times in the sq$()
  $include "shared.pro"           '*** array to reflect new zero.
  for i = 1 to resources
    if queues$(i,R) <> "" then
      x = val(mid$(queues$(i,R),2,4))
      call getque(x)
      tti$ = FNtrun$(str$(val(tti$) - val(pdt$(5,1))))
      call put5(tti$)
      call putque(x)
    elseif rltm$(2,i) = "S" then
      x = val(mid$(rltm$(49,i),2,4))
      call getque(x)
      tti$ = FNtrun$(str$(val(tti$) - val(pdt$(5,1))))
      call put5(tti$)
      call putque(x)
    end if
  next i
  pdt$(9,1) = "0"
end sub

sub endbucket
  $include "shared.pro"
  close #10
  for i = 1 to resources
    call util(i,rltm$(2,i))
  next i
  call globalcalc                       '*** global calcs

'*****************************************************************
'*** TO REMOVE THE FINISHED JOBS FROM WORK AND FROM THE SQ$()
```

```
'*** DATABASE.
'*** NOW ALSO FROM THE BNSCH$() DATABASE

'*****************************************************************
'*** clear up sq$()
'***
  for i = 1 to jobnum
    if work$(i,1) = "" then exit for
    if left$(work$(i,2),1) = "F" then
      j = 1
      do until FNtrun$(mid$(sq$(j,1),11,5)) = FNtrun$(work$(i,1)) and_
                val(mid$(sq$(j,1),66,5)) = val(work$(i,4))
    incr j
      loop

      k = j
      do until FNtrun$(mid$(sq$(k,1),11,5)) <> FNtrun$(work$(i,1)) or_
                val(mid$(sq$(k,1),66,5)) <> val(work$(i,4))
    sq$(k,0) = ""
    sq$(k,1) = ""
    incr k
      loop

'*** SHUNT SQ$()
      row = 1
      for ii = 1 to sqnum
    if sq$(ii,1) <> "" then
      sq$(row,1) = sq$(ii,1)
      if row <> ii then
        sq$(ii,1) = ""
        sq$(ii,0) = ""
      end if
      incr row
    end if
      next ii
'*** END SHUNT

      l = k - j      '*** l is the quantity to remove
      for m = 1 to sqnum
    if sq$(m,1) = "" then exit for
    x$ = left$(sq$(m,1),5)
    y$ = mid$(sq$(m,1),6,5)
    z$ = mid$(sq$(m,1),11,95)
    if val(x$) > 10000 + k - 1 then
      x$ = str$(val(x$) - l)
      x$ = FNtrun$(x$)
    end if
    if val(y$) > 10000 + k - 1 then
      y$ = str$(val(y$) - l)
      y$ = FNtrun$(y$)
    end if
    sq$(m,1) = x$ + y$ + z$
      next m

      for m = 1 to 4
    for n = 1 to resources
      if queues$(n,m) <> "" then
        x = len(queues$(n,m))
```

```
        for p = 1 to x/5
          x$ = mid$(queues$(n,m),(p*5)-4,5)
          if val(x$) > 10000 + k - 1 then
        x$ = FNtrun$(str$(val(x$) - l))
        mid$(queues$(n,m),(p*5)-4,5) = x$
          end if
        next p
      end if
    next n
      next m
    end if
  next i
'****************** end of sorting out the sq$().
  best = 1
  for i = 1 to jobnum
    if work$(i,1) = "" then exit for
    if val(work$(i,4)) > best then
      best = val(work$(i,4))
    end if
  next i
  incr best
  if val(ctl$(40,1)) = 11 then
    call steady(best)
  end if
  for i = 1 to jobnum                    '*** clears rows in work$()
    if work$(i,1) = "" then exit for
    if left$(work$(i,2),1) = "F" then
      for j = 0 to 15
    work$(i,j) = ""
      next j
    end if
    work$(i,14) = ""
  next i

  bkrow = 0                     '*** shunts rows in work$()
  for i = 1 to jobnum
    incr bkrow
    if work$(i,1) = "" then exit for
  next i

  if bkrow < jobnum then
    row = bkrow
    for i = row+1 to jobnum
      if work$(i,1) <> "" then
    for j = 0 to 15
      work$(bkrow,j) = work$(i,j)
      work$(i,j) = ""
    next j
    incr bkrow
      end if
    next i
  end if


'*****************************************************************
'*** SET UP THE TIMING MECHANISM FOR THE NEXT PERIOD

  call acttime           '*** START AGAIN AT THE BEGINNING OF
```

```
  now& = 0          '*** THE NEXT PERIOD
  pdt$(43,1) = pdt$(47,1)
  pdt$(44,1) = pdt$(48,1)
  pdt$(45,1) = pdt$(49,1)
  pdt$(46,1) = pdt$(50,1)

end sub

sub steady(best)
  shared job$(), ctl$(), jobnum, pdt$(), work$(), jobHo$()

  count = 1
  do until job$(count,1) = ""
    incr count
  loop
  countho = count
  call pjtrest(ctl$(15,1)+"~job.inf",jobho$(),999)
  for i = 1 to 16
    if jobho$(i,1) = "" then exit for
    for j = 1 to 15
     job$(i-1+count,j) = jobho$(i,j)
    next j
  next i

  do until job$(count,1) = ""
    job$(count,4) = FNtrun$(str$(best))
    incr count
    incr best
  loop

   weeks = 0        '  WAS 2.
   call quantity(countho)
   call duedate(weeks,countho)
'*** remember that due date does this to all jobs would need sorting
'*** out if some jobs are to be left over.  Taken care of with countho.
  locate 22,1 : print "seed...";ctl$(32,1);
end sub


sub duedate(weekstoadd,countho)
  $include "shared.pro"
  randomize val(ctl$(32,1))
  ctl$(32,1) = FNtrun$(str$(val(ctl$(32,1))+1))

  x = val(pdt$(47,1))
  j = countho
  do until job$(j,1) = ""
    y = FNuniform(5,5)
    hr = FNuniform(0,7)
    mi = FNuniform(0,59)
    count = x + weekstoadd
    if count < 10 then
      job$(j,5) = "00"+FNtrun$(str$(count))
    elseif count < 100 then
      job$(j,5) = "0"+FNtrun$(str$(count))
    else
      job$(j,5) = FNtrun$(str$(count))
    end if
```

```
    job$(j,6) = "0"+FNtrun$(str$(y))
    if hr < 10 then
      job$(j,7) = "0"+FNtrun$(str$(hr))
    else
      job$(j,7) = FNtrun$(str$(hr))
    end if
    if mi < 10 then
      job$(j,8) = "0"+FNtrun$(str$(mi))
    else
      job$(j,8) = FNtrun$(str$(mi))
    end if
    incr j
  loop
end sub


sub jumble
  shared jobnum, work$(), job$(), ctl$()
  '*** to randomize the work file.
  randomize val(ctl$(32,1))
  ctl$(32,1) = FNtrun$(str$(val(ctl$(32,1))+1))

  row = 0
  do until left$(job$(row+1,2),1) = "N" and job$(row+1,11) = ""
    incr row
  loop

  endrow = row
  do until job$(endrow+1,1) = ""
    incr endrow
  loop

  if row <> endrow and endrow <> 0 then
    for i = row+1 to endrow
      uni = int(rnd * (endrow-row)) + (row+1)
      for j = 0 to 15
    swap job$(i,j), job$(uni,j)
      next j
    next i
  end if
end sub

sub iterdump
  $include "shared.pro"
  if val(pdt$(6,1)) >= val(pdt$(35,1)) or_
                  val(pdt$(6,1)) = val(pdt$(10,1)) then
    simu$ = pdt$(25,1)                              '*** TO FILE
    pdt$(25,1) = "0"
    call scrboxes
    pdt$(25,1) = simu$
    y$ = ctl$(15,1) : z$ = ctl$(17,1)
    call dump(y$,z$,val(pdt$(6,1)))
  end if
end sub         '*** endbucket

sub globalcalc
  shared ls$(), rltj$(), rltm$(), rltg$(), pdt$(), resources, jobnum, mac$()
  shared cr$(), sh$(), sqnum
```

```
  resno = 1
  do until mac$(resno,1) = ""
    incr resno
    if resno > resources then exit loop
  loop
  decr resno

'************************************************************
'*** CALCULATION OF UTILISATION RESULTS
  for mc = 1 to resno
    total = 0
    for i = 4 to 8
      total = total + val(rltm$(i,mc))
    next i
    if total <> 0 then
      S = (val(rltm$(4,mc)) * 100) / total : S = round(S,2)
      W = (val(rltm$(5,mc)) * 100) / total : W = round(W,2)
    else
      S = 0 : W = 0 : II = 0 : B = 0 : N = 0
    end if
    rltm$(39,mc) = FNtrun$(str$(W))
    rltm$(41,mc) = FNtrun$(str$(W+S))
  next mc

'*********************************************************************
'*** CALCULATE THE GLOBAL RESULTS
  for i = 1 to jobnum                  '*** jobs early/late/ontime
    if rltj$(i,1) = "" then exit for       '*** & tot time early/late
    if rltj$(i,11) <> "" then
      if val(rltj$(i,5)) < 0 then
    rltg$(3,1) = FNtrun$(str$(val(rltg$(3,1)) + 1))
    rltg$(5,1) = FNtrun$(str$(val(rltg$(5,1)) + val(rltj$(i,5))))
      elseif val(rltj$(i,5)) > 0 then
    rltg$(4,1) = FNtrun$(str$(val(rltg$(4,1)) + 1))
    rltg$(6,1) = FNtrun$(str$(val(rltg$(6,1)) + val(rltj$(i,5))))
      else
    rltg$(2,1) = FNtrun$(str$(val(rltg$(2,1)) + 1))
      end if
      rltg$(8,1) = FNtrun$(str$(val(rltg$(8,1)) + val(rltj$(i,4))))  'tot flow
      rltg$(9,1) = FNtrun$(str$(val(rltg$(9,1)) + val(rltj$(i,3))))  'theo flow
      rltg$(18,1) = FNtrun$(str$(val(rltg$(18,1)) + val(rltj$(i,18))))'flow
    end if
  next i                  '**********

  rltg$(7,1) = FNtrun$(str$(val(rltg$(6,1)) - val(rltg$(5,1))))  'manu-g err
  x = val(rltg$(7,1)) : x = abs(x) : rltg$(7,1) = FNtrun$(str$(x))
  rltg$(10,1) = FNtrun$(str$(val(rltg$(8,1)) - val(rltg$(9,1))))  'del error
  x = val(rltg$(10,1)) : x = abs(x) : rltg$(10,1) = FNtrun$(str$(x))

  x1& = 0 : x2& = 0 : x3& = 0 : x4& = 0 : X4! = 0
  x5! = 0 : x5! = 0 : x6! = 0 : x7! = 0

  for i = 1 to resno
    x1& = x1& + val(rltm$(10,i))
    x2& = x2& + val(rltm$(4,i))
    x3& = x3& + val(rltm$(5,i))
    x4& = x4& + val(rltm$(6,1))
```

```
    x4! = x4! + val(rltm$(32,i))
    x5! = x5! + val(rltm$(35,i))
    x6! = x6! + val(rltm$(39,i))
    x7! = x7! + val(rltm$(41,i))
  next i

  x4! = x4!/resno
  x5! = x5!/resno
  x6! = x6!/resno
  x7! = x7!/resno

  rltg$(11,1) = FNtrun$(str$(val(pdt$(5,1))*resno))
  rltg$(12,1) = FNtrun$(str$(x1&))    'tot mc time avail
  rltg$(13,1) = FNtrun$(str$(x2&))    'tot set up time
  rltg$(14,1) = FNtrun$(str$(x3&))    'tot working time
  rltg$(15,1) = FNtrun$(str$(x4&))    'tot idle time
  rltg$(26,1) = FNtrun$(str$(x4!))    'ave pt of queue wrt time
  rltg$(27,1) = FNtrun$(str$(x5!))    'ave queue length in jobs
  rltg$(28,1) = FNtrun$(str$(x6!))    'util I
  rltg$(29,1) = FNtrun$(str$(x7!))    'util II

  rltg$(28,1) = mid$(rltg$(28,1),1,4)
  rltg$(29,1) = mid$(rltg$(29,1),1,4)

  if val(rltg$(4,1)) = 0 then
    rltg$(30,1) = "0"
  else
    y1! = val(rltg$(6,1)) / val(rltg$(4,1)) 'ave late
    rltg$(30,1) = FNtrun$(str$(y1!))
  end if
  if val(rltg$(3,1)) = 0 then
    rltg$(31,1) = "0"
  else
    y1! = val(rltg$(5,1)) / val(rltg$(3,1)) 'ave early
    rltg$(31,1) = FNtrun$(str$(y1!))
  end if

  x = val(rltg$(2,1)) + val(rltg$(3,1)) + val(rltg$(4,1))  'freq tardy job
  y = val(rltg$(4,1))
  if x = 0 or y = 0 then
    rltg$(33,1) = "0"
    rltg$(25,1) = "0"
  else
    y1! = (y/x) * 100
    rltg$(33,1) = FNtrun$(str$(y1!))            '***
    y1! = (val(rltg$(8,1)) / x)                          'ave flow
    rltg$(25,1) = FNtrun$(str$(y1!))
  end if



  '*********************************************************************
  '*** LEARNING SYSTEM UPDATE
    f = 5
    x = val(pdt$(14,1))              '* the sequencing heuristic
    y = val(pdt$(11,1))              '* SH selection model
    if pdt$(2,1) <> "1" then
      z = (x*6) - 6 + y
```

```
    else
      z = 37              '*** similation results into row
    end if                '*** 26 of ls$()

    if val(rltg$(9,1)) = 0 then
      ls$(z,1) = "0"
      ls$(z,2) = "0"
    else
      x1! = val(rltg$(7,1))/val(rltg$(9,1))   '*** delivery error one
      x1! = (1 - x1!) * 100
      x$ = FNtrun$(str$(x1!))
      ls$(z,1) = left$(x$,5)
      x1! = val(rltg$(10,1))/val(rltg$(9,1))  '*** manufacturing error one
      x1! = (1 - x1!) * 100
      x$ = FNtrun$(str$(x1!))
      ls$(z,2) = left$(x$,5)
    end if

    x$ = left$(rltg$(28,1),5)
    ls$(z,3) = x$                 '*** utilisation
    x! = 0
    for i = 1 to 3
      x! = (val(ls$(z,i)) * val(pdt$(28+i,1))) + x!
    next i
    x! = round(x!,2)
    x$ = FNtrun$(str$(x!))
    ls$(z,5) = left$(x$,5)
end sub


sub quantity(countho)
  $include "shared.pro"
  if ctl$(40,1) = "11" then       '*** using the test role facility
    v1 = val(pdt$(12,1))
  else
    v1 = val(ctl$(30,1))
  end if
  v2 = val(ctl$(31,1))
  '*** v1 = mean, v2 = coefficient of variation
  rod = 0 '*** rounding
  call normal2(v1,v2)

  open "random.out" for input as #1
    for i = countho to jobnum
      if job$(i,1) = "" then exit for
      input #1, xx$
      if val(xx$) < 1 then xx$ = "1"
      if val(xx$) > 14 then xx$ = "14"
      job$(i,3) = xx$
    next i
  close #1
end sub

sub normal2(v1,v2)
  shared rod, scrprn, p1$(), ctl$(), jobnum, job$()

  OPEN "RANDOM.OUT" FOR OUTPUT AS #1
```

```
  randomize val(ctl$(32,1))
  ctl$(32,1) = FNtrun$(str$(val(ctl$(32,1))+1))
  sd! = (v2 * v1) /100   '***  sd = cv * mean divide by 100
  sd! = round(sd!,rod)
  p1$(2) = "  NORMAL: mean =" + str$(v1) + ", std. dev. =" + str$(sd!) +_
       ", from a CV of " + v2$ + "%."
  for i = 1 to jobnum
    if job$(i,1) = "" then exit for
    y# = FNnorm#
    y# = (y# * sd!) + v1
    y! = round(y#,rod)
    y$ = FNtrun$(str$(y!))
    if val(y$) < 1 then y$ = "1" '*** no zero demands.
    write #1, y$
  next i
  close #1
end sub

def FNnorm#
  y# = 0 : flag = 0
  do
    u1# = rnd : v1# = (2*u1#) - 1
    u2# = rnd : v2# = (2*u2#) - 1
    w# = (v1# * v1#) + (v2# * v2#)
    if w# > 1 then
      flag = 0
    else
      flag = 1
      y# = (-2) * (log(w#))
      y# = y#/w#
      y# = sqr(y#)
      x1# = v1# * y#
      x2# = v2# * y#
    end if
  loop until flag = 1
  FNnorm# = x1#
end def

def FNuniform(v1,v2)
  shared scrprn
  FNuniform = int(rnd * (v2-v1+1)) + v1
end def
```

M-COM.PRO

```
'     program M-COM.PRO

'     CONTENTS - common routines

'     Edition 1.0   Version 1.0   14/9/88

'     Manufacturing Simulation and Analysis

'     Copyright Dr J Driscoll

' ***** COMMON SUBROUTINE BLOCK *****

sub toupper(y3$)
  for i = 1 to len(y3$)
    y = asc(mid$(y3$,i,1))
    if y > 90 then
      y = y - 32
      x1$ = x1$ + chr$(y)
    else
      x1$ = x1$ + mid$(y3$,i,1)
    end if
  next i
  y3$ = x1$
  x1$ = ""
end sub

def FNletter(y$)
  for i = 1 to len(y$)
    y = asc(mid$(y$,i,1))
    if ((y > 64) and (y < 91)) or ((y > 47) and (y < 58)) then
      FNletter = 1
    else
      FNletter = 0
      exit def
    end if
  next i
  FNletter = 1
end def


sub border                    ' *** SETS UP THE STANDARD TOP THREE LINES
  shared sim$(), ctl$()
  call fill(14,9,1,1,80,3)
  color 14,9
  locate 1,2 : print "Copyright:                    MANUFACTURING SIMULATION"
  locate 2,2 : print " S Hurley                           AND ANALYSIS"
  locate 3,2 : print "J Driscoll"
  call rblock
end sub

sub waiton
  color 31,4
  locate 25,77 : print "WAIT";
  color 15,1
end sub

sub waitoff
  color 0,3
```

```
  locate 25,77 : print "    ";
  color 15,1
end sub

sub scrclear          ' *** clears the screen ***
  color 0,3
  cls
end sub

sub insound       ' *** asking for input ***
  sound 1157,2
end sub

sub failsound          ' *** wrong input given ***
  sound 100,3 : delay 0.3
  sound 100,3
end sub

sub fill(a,b,y,x,x1,y1)           ' *** creates box with no border ***
  color a,b
  for i = 0 to y1-1
    for j = 0 to x1-1
      locate y+i,x+j
      print " ";
    next j
  next i
  color 15,1
end sub

sub shadow(a,b,y,x,x1,y1)                          '*** CREATES SHADOW
  color a,b
  for i = 1 to y1
    locate y+i,x+x1 : print chr$(219);
  next i
  for i = 0 to x1-1
    locate y+y1,x+1+i : print chr$(223);
  next i
  color 15,1
end sub


sub box(a,b,y,x,x1,y1)          ' *** creates box outline ***
  color a,b
  locate y,x
  print chr$(201);
  for i = 1 to x1-2
    print chr$(205);
  next i
  print chr$(187);
  for i = 1 to y1-2
    locate y+i,x      : print chr$(186);
    locate y+i,x+x1-1 : print chr$(186);
  next i
  locate y+y1-1,x
  print chr$(200);

  for i = 1 to x1-2
    print chr$(205);
```

```
  next i

  print chr$(188);
end sub

sub boxsing(a,b,y,x,x1,y1)          ' *** creates box outline ***
  color a,b
  locate y,x
  print chr$(218);
  for i = 1 to x1-2
    print chr$(196);
  next i
  print chr$(191);
  for i = 1 to y1-2
    locate y+i,x      : print chr$(179);
    locate y+i,x+x1-1 : print chr$(179);
  next i
  locate y+y1-1,x
  print chr$(192);

  for i = 1 to x1-2
    print chr$(196);
  next i

  print chr$(217);
end sub

def FNindata$(y,x,x1)
400: color 15,4
locate y,x-1 : print " "
locate y,x+x1: print " "
  locate y,x
  x$ = " "
  x3$ = chr$(3)
  for x0 = 1 to x1
    print chr$(22);
    x$ = x$ + " "
    x3$ = x3$ + chr$(3)
  next x0
  locate y,x : x4 = 0
420: x3 = 0 : x0$ = " "
  x0$ = inkey$
  x3 = len(x0$)
  if x3 > 0 then 440
  call waiting
  color 15,4
  goto 420
440: if asc(x0$) = 8 then 470
  if asc(x0$) <> 13 then 500
  goto 530
470: if x4 = 0 then 490
  mid$(x$,x4,1) = " " : x4 = x4 - 1: locate y,x+x4: print chr$(22);
  locate y,x+x4 : goto 420
490: locate y,x : print chr$(22) : locate y,x : goto 420
500: if x4 = x1 then 420
  x4 = x4+1 : mid$(x$,x4,1)=x0$ : locate y,x : x2$ = x$
  if x5 = 99 then x2$ = x3$
  locate y,x : print left$(x2$,x4) : goto 420
```

```
530: if x4>0 then x$ = left$(x$,x4) else x$ = " "
540: if x4 = 0 or left$(x$,1) <> " " then 560 else x$ = mid$(x$,2,len(x$)-1)
  x4 = x4 - 1 : goto 540
560: if x4 = 0 or right$(x$,1) <> " " then 580 else x$ = mid$(x$,1,len(x$)-1)
  x4 = x4 - 1
  goto 560
580: x5 = 0
  FNindata$ = x$
  color 15,1
  if x$ = "" or x$ = " " then
    call failsound
    goto 400
  end if
  for i = 1 to len(x$)
    if asc(mid$(x$,i,1)) = 0 then
      call failsound
      goto 400
    end if
  next i
end def

Def FnInNum%(low,high,num$)
  for i = 1 to len(num$)
    if asc(mid$(num$,i,1)) < 48 or asc(mid$(num$,i,1)) > 57 then
      FnInNum% = -1
      exit def
    end if
  next i
  if val(num$) < low or val(num$) > high then
    FnInNum% = -1
  else
    FnInNum% = 0
  end if
end def

def FNtrun$(x$)
  do
    if len(x$) = 0 then exit loop
    if left$(x$,1) = " " then x$ = mid$(x$,2,len(x$)-1)
    if len(x$) = 0 then exit loop
    if mid$(x$,len(x$),1) = " " then x$ = mid$(x$,1,len(x$)-1)
  loop until left$(x$,1) <> " " and right$(x$,1) <> " "
  FNTrun$ = x$
end def

sub na
  call fill(0,3,4,1,80,21)
  call shadow(0,3,10,21,38,10)
  call fill(15,1,10,21,38,10)
  color 14,1
  locate 14,31 : print "NOT AVAILABLE YET";
  locate 16,31 : print "press <space bar>"
  call insound
  call spce
  call fill(0,3,4,1,80,21)
end sub

sub Menu(title$,na1$,na2$,na3$,na4$,na5$,na6$,na7$,na8$,na9$,_
```

```
                                                    na10$,na11$,na12$)
  call shadow(0,3,6,5,32,15)
  call fill(14,1,6,5,32,15)
  color 14,1
  locate 7,7  : print title$
  color 15,1
  locate 9,7  : print na1$
  locate 10,7 : print na2$
  locate 11,7 : print na3$
  locate 12,7 : print na4$
  locate 13,7 : print na5$
  locate 14,7 : print na6$
  locate 15,7 : print na7$
  locate 16,7 : print na8$
  locate 17,7 : print na9$
  locate 18,7 : print na10$
  locate 19,7 : print na11$
  locate 20,7 : print na12$
end sub

sub Menu2(title$,na1$,na2$,na3$,na4$,na5$,na6$,na7$,na8$,na9$,_
                                                    na10$,na11$,na12$)
  call shadow(0,3,7,7,32,15)
  call fill(4,7,7,7,32,15)
  color 14,7
  locate 9,9  : print title$
  color 4,7
  locate 11,9  : print na1$
  locate 12,9 : print na2$
  locate 13,9 : print na3$
  locate 14,9 : print na4$
  locate 15,9 : print na5$
  locate 16,9 : print na6$
  locate 17,9 : print na7$
  locate 18,9 : print na8$
  locate 19,9 : print na9$
  locate 20,9 : print na10$
  locate 21,9 : print na11$
  locate 22,9 : print na12$
end sub


SUB EdArray(file$(2),DIM1,DIM2,flag) public
  flag = 0
  c = 2
  d = 8      '*** depth down screen
  t = 0
  iconrh = 0
  e = 15                '*** bottom bit
  icon = 1
  call fill(15,1,d-2,2,78,13)
  call box(15,1,d-2,2,78,13)
  color 14,9
  locate d-2,25 : print " "file$(0,0)" "
  locate d+10,7 : print " "chr$(24)chr$(25)" and PgUp, PgDn to move icon: <ENTER> to select:
<Esc> to quit "
  color 15,1
  locate d,3
```

```
print "------------------------------------------------------------------------"

7
    color 31,9 : locate icon+d,c*10-2 : print chr$(4) : color 15,1
    if icon <> 1 then
      locate icon+d-1,18+(c-2)*10 : print " "
    end if
    if icon <> 9 then
      locate icon+d+1,18+(c-2)*10 : print " "
    end if
    if dim2 < 6 then
      x = dim2
    else
      x = 6
    end if
    for cols = 1 to x                    ' *** PRINTING THE TITLES
      locate d-1,(cols+1)*10-1 : print using "\       \"; file$(0,cols+iconrh)
    next cols
    if dim1 < 9 then
      y = dim1
    else
      y = 9
    end if
    for lines = 1 to y
      locate d+lines,4 : print lines+t
      locate d+lines,9 : print using "\      \"; file$(lines+t,0)
      for cols = 1 to x
        locate d+lines,(cols+1)*10-1 : print using "\      \";file$(lines+t,cols+iconrh)
      next cols
    next lines


  do                     ' *** TO READ THE KEYBOARD ARROWS
    x$=inkey$
  loop until len(x$)>0


  do until asc(mid$(x$,1,1)) = 13            ' *** RETURN TO EXIT EDIT
    if len(x$) = 2 then                      ' *** ALL CTL CHARS ARE TWO CHARS
      x = asc(mid$(x$,2,1))
      select case x                        ' *** TO PICK REQ ACTION
        case = 72                          ' *** UP ARROW
          if icon > 1 then
            decr icon
          elseif t <> 0 then
            decr t
          end if
        case = 80                                                ' *** DOWN ARROW
          if (icon < 9) and (icon < dim1) then
            incr icon
          elseif t <> dim1-9 and icon < dim1 then
            incr t
          end if
        case = 71                                                ' *** HOME
          t = 0
        case = 75                                                '*** LEFT ARROW
          if c > 2 then
            locate icon+d,c*10-2 : print " "
```

```
            decr c
          elseif iconrh > 0 then
            decr iconrh
          end if
        case = 77                          '*** RIGHT ARROW
          if (c < 7) and (c-1 < dim2) then
            locate icon+d,c*10-2 : print " "
            incr c
          elseif dim2 > iconrh+c-1 then
            incr iconrh
          end if
        case = 79                                          ' *** END
          if dim1 > 9 then
            t = dim1-9
          end if
       case = 73                                          ' *** PgUp
          if t > 9 then
            t = t - 8
          else
            t = 0
          end if
        case = 81                                          ' *** PgDn
          if t < dim1-16  then
            t = t + 8
          elseif dim1 > 9 then
            t = dim1-9
          end if
        case else
          call failsound
      end select
      goto 7
    end if
    if len(x$) = 1 and asc(x$) = 27 then          ' 27 IS <Esc>
      exit loop
    else
      goto 7                          ' *** TO COPE WITH ORDINARY
    end if
    goto 7                            ' *** LETTERS BEING PRESSED
  loop

  if asc(mid$(x$,1,1)) = 13 then
    call fill (15,1,e+5,50,27,4)              ' *** TO CREATE THE EDITOR WINDOW
    call boxsing (15,1,e+5,50,27,4)
    color 14,9
    flag = 1
    locate e+5,60 : print " EDITOR "
    locate e+5,56 : print " <Esc> to quit "
    color 15,4
    locate d+icon,c*10-1 : print using "\      \";file$(icon+t,iconrh+c-1)
    color 15,1
    locate e+7,52 : print file$(0,iconrh+c-1)
    file$(icon+t,iconrh+c-1) = FNindata$(e+7,65,9)
    call fill(0,3,e+5,50,27,4)
    goto 7
  end if
end sub

sub put5(x$)
```

```
  y% = len(x$)/5
  if (y%*5) = len(x$) then exit sub
  y = len(x$) + 5
  y = y/5
  y = fix(y)
  y = y*5
  x = len(x$)
  for i = x to y-1
    x$ = x$ + " "
  next i
end sub

sub put10(x$)
  y% = len(x$)/10
  if (y%*10) = len(x$) then exit sub
  y = len(x$) + 10
  y = y/10
  y = fix(y)
  y = y*10
  x = len(x$)
  for i = x to y-1
    x$ = x$ + " "
  next i
end sub

def FNrndint%(x)
  randomize timer
  FNrndint% = int(rnd * (x-1)) + 1
end def

sub spce
  call waiting
  x$ = input$(1)
  do until asc(x$) = 32
    call failsound
    x$ = input$(1)
  loop
end sub

sub waiting          ' *** prints time when waiting for input
  color 14,9
  while not instat
    x$=time$
    while not instat
      y$ = time$
      if mid$(x$,7,2) <> mid$(y$,7,2) then
        call time
        exit loop
      end if
    wend
  wend
  color 15,1
end sub

sub time
'  x$ = left$(time$,5)
  locate 25,1 : print " ";time$;" ";
end sub
```

```
sub rblock
  shared pjt$(), ctl$(), pdt$()
  x = val(ctl$(11,1))
  if val(ctl$(13,1)) = 0 or ctl$(15,1) = "" then
    color 14,1
    locate 2,70 : print "No Project"
    locate 3,70 : print "is active"
    color 15,1
  else
    color 15,1
    locate 1,70 : print "          "
    color 14,1
    locate 1,66 : print " Project: ";
    locate 2,66 : print " Version: ";
    locate 3,66 : print " Period: ";
    color 0,1
    locate 1,76 : print "    ";
    locate 2,76 : print "    ";
    locate 3,75 : print "     ";
    color 15,1
    locate 1,76 : print pjt$(x,1);" "
    call toupper(ctl$(17,1))
    if ctl$(17,1) = "INF" then
      x$ = "Root"
    else
      x$ = ctl$(17,1)+" "
    end if
    locate 2,76 : print x$
    locate 3,75 : print " ";pdt$(6,1);" "
  end if
  locate 3,26 : color 15,1
  print "                              "
  if ctl$(15,1) <> "" then
    y = len(pjt$(x,3))
    locate 3,40-(y/2)
    print " ";pjt$(x,3);" "
  end if
end sub

sub blocks
  shared now$

  call fill(14,9,1,1,10,3)
  color 14,1
  locate 1,2 : print " M.S.A. "
  locate 2,2 : print "Co Name"
  locate 3,2 : print "User  ??"

  call fill(14,9,1,70,10,3)
  color 14,9
  call time
  locate 3,71 : print "Sim. Time "
  color 15,1
end sub
```

M-FILE.PRO

```
'     program M-FILE.PRO

'     CONTENTS - all the file handling routines

'     Edition 1.0   Version 1.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright S Hurley and Dr J Driscoll

' ***** COMMON SUBROUTINE BLOCK *****

sub dumpS2(file$,arr$(),lines,cols)
  open file$ for output as #1                    '*** WRITING
  for l = 0 to lines
    x$ = ""
    for c = 0 to cols
      if arr$(l,c) = "" then
    arr$(l,c) = " "
      else
    arr$(l,c) = left$(arr$(l,c),10)
      end if
      x$ = x$ + arr$(l,c) : call put10(x$)
    next c
    write #1, x$
  next l
  close #1
end sub

sub restorS2(file$,arr$(),lines,cols)
  open file$ for input as #1                     '*** READING
  for l = 0 to lines
    input #1, x$
    for c = 0 to cols
      arr$(l,c) = mid$(x$,(c*10)+1,10) : arr$(l,c) = FNtrun$(arr$(l,c))
    next c
    x$ = ""
  next l
  close #1
end sub

sub dumpfile(file$,arr$(),lines,cols)
  open file$ for output as #1                    '*** WRITING

  for l = 0 to lines
    x$ = ""
    for c = 0 to cols
      x$ = x$ + arr$(l,c)+","
    next c
    write #1, x$
  next l
  close #1
end sub

sub restfile(file$,arr$(),lines,cols)
  open file$ for input as #1                     '*** READING
  for l = 0 to lines
    n = 1
```

```
    if eof(1) then exit for
    input #1, x$
    for c = 0 to cols
      y = instr(n,x$,",")
      arr$(l,c) = mid$(x$,n,y-n)
      n = y + 1
    next c
  next l
  close #1
end sub


sub dumpho(file$,arr$(),lines,cols)
  open file$ for output as #1                    '*** WRITING
  for l = 0 to lines
    for c = 0 to cols
      write #1, arr$(l,c)
    next c
  next l
  close #1
end sub

sub restorho(file$,arr$(),lines,cols)
  open file$ for input as #1                     '*** READING
  for l = 0 to lines
    for c = 0 to cols
      input #1, arr$(l,c)
    next c
  next l
  close #1
end sub

sub DumpBom(file$)
  shared bom$()
  open file$ for append as #1
  for i = 1 to 200
    if bom$(i,1) = "" then exit for
    write #1,bom$(i,1)
  next i
  close #1
end sub

sub RestoreBom(file$)
  shared bom$()

  open file$ for input as #1
  for i = 1 to 200
    if eof(1) then exit for
    input #1,x$
    bom$(i,1) = x$
  next i
  close #1
end sub

sub DumpRes(file$)
  shared mac$(), resources

  open file$ for output as #5
```

```
  for i = 1 to resources
    if mac$(i,1) = "" then exit for
    OutBuf$ = ""
    for j = 0 to 15
      OutBuf$ = OutBuf$ + mac$(i,j) + "\"
    next j
    write #5, OutBuf$
  next i
  close #5
end sub


sub RestoreRes(file$)
  shared mac$(), resources
  open file$ for input as #4
  n = 1
  for i = 1 to resources
    n = 1
    if eof(4) then exit for
    input #4, x$
    for j = 0 to 15
      y = instr(n,x$,"\")
      mac$(i,j) = mid$(x$,n,y-n)
      n = y+1
    next j
  next i
  close #4
end sub

sub dumpsq(file$)
  shared sq$(), sqnum
  open file$ for output as #1
  for i = 1 to sqnum
    if sq$(i,1) = "" then exit for
    write #1,sq$(i,1)
  next i
  close #1
end sub

sub restsqdb(file$)
  shared sq$(), sqnum
  open file$ for input as #1
  for i = 1 to sqnum
    if eof(1) then exit for
    input #1,x$
    if x$ = "" then exit for
    sq$(i,1) = x$
  next i
  close #1
end sub

sub dump(y$,z$,per)
  $include "shared.pro"
  call pjtdump(y$+"~pdt."+z$,pdt$(),50,1,per)
  call pjtdump(y$+"~rsj."+z$,rltj$(),jobnum,22,per)
  call pjtdump(y$+"~rsm."+z$,rltm$(),49,resources,per)
  call pjtdump(y$+"~rsg."+z$,rltg$(),40,1,per)
  call pjtdump(y$+"~crh."+z$,cr$(),8,1,per)
```

```
  call pjtdump(y$+"~shh."+z$,sh$(),8,1,per)
  call pjtdump(y$+"~lsy."+z$,ls$(),37,5,per)
  call pjtdump(y$+"~sqr."+z$,sq$(),sqnum,1,per)
  call dumpfile(y$+"~cal."+z$,cal$(),calnum,8)
  call pjtdump(y$+"~job."+z$,job$(),jobnum,15,per)
  call dumpres(y$+"~mac."+z$)
  call pjtdump(y$+"~wrk."+z$,work$(),jobnum,15,per)
  call pjtdump(y$+"~bns."+z$,bnsch$(),bnnum,5,per)
  call pjtdump(y$+"~que."+z$,queues$(),resources,4,per)
end sub    '*** dump

sub restorefiles(y$,z$,per)
  $include "shared.pro"
  call pjtrest(y$+"~pdt."+z$,pdt$(),per)
  call restfile(y$+"~cal."+z$,cal$(),calnum,8)
  call pjtrest(y$+"~job."+z$,job$(),per)
  call restoreres(y$+"~mac."+z$)
  call pjtrest(y$+"~que."+z$,queues$(),per)
  call pjtrest(y$+"~rsj."+z$,rltj$(),per)
  call pjtrest(y$+"~rsm."+z$,rltm$(),per)
  call pjtrest(y$+"~rsg."+z$,rltg$(),per)
  call pjtrest(y$+"~crh."+z$,cr$(),per)
  call pjtrest(y$+"~wrk."+z$,work$(),per)
  call pjtrest(y$+"~sqr."+z$,sq$(),per)
  call pjtrest(y$+"~bns."+z$,bnsch$(),per)

end sub    '*** restorefiles

sub pjtdump(file$,arr$(),lines,cols,per)
  shared pdt$(), ctl$()

  if ctl$(40,1) = "11" then       '*** ie running test cases.
    open file$ for append as #1      '*** do not have to slot in.
    write #1,pdt$(6,1) + " PERIOD"
    for l = 0 to lines
      if arr$(l,1) = "" and arr$(l,0) = "" then exit for
      x$ = ""
      for c = 0 to cols
    x$ = x$ + arr$(l,c) + ", "
      next c
      write #1, x$
    next l
    close #1
    exit sub
  end if

  open file$ for append as #1
  close #1

  open file$ for input as #1
  open "dump.dmp" for output as #2
  do until (val(left$(x$,3)) = per and mid$(x$,5,6) = "PERIOD") or eof(1)
    input #1, x$
    write #2, x$
  loop
  if eof(1) then write #2,pdt$(6,1) + " PERIOD"
  for l = 0 to lines
    if arr$(l,1) = "" and arr$(l,0) = "" then exit for
```

```
    x$ = ""
    for c = 0 to cols
      x$ = x$ + arr$(l,c) + ", "
    next c
    write #2, x$
  next l
  if not eof(1) then
    input #1, x$
    do until (val(left$(x$,3)) = per+1 and mid$(x$,5,6) = "PERIOD") or eof(1)
      input #1, x$
    loop
  end if
  if not eof(1) then write #2, x$
  do until eof(1)
    input #1, x$
    write #2, x$
  loop
  close #1
  close #2
  open file$ for output as #1
  open "dump.dmp" for input as #2
  do until eof(2)
    input #2, x$
    write #1, x$
  loop
  close #1
  close #2
end sub


sub pjtrest(file$,arr$(),per)
  open file$ for input as #1        '*** READING

  if per <> 999 then
    input #1, x$
    do until (val(left$(x$,3)) = per and mid$(x$,5,6) = "PERIOD")
      input #1, x$
    loop

    l = 0
    do until eof(1)
      input #1, x$
      if mid$(x$,5,6) = "PERIOD" then
    exit loop
      else
    arr$(l,0) = x$
    incr l
      end if
    loop
    hold = l
  else
    do until eof(1)
      input #1, x$
      if mid$(x$,5,6) = "PERIOD" then
    for i = 0 to l-1
      arr$(i,0) = ""
    next i
    l = 0
```

```
      else
     arr$(l,0) = x$
     incr l
      end if
     loop
     hold = l
   end if

   l = 0 : c = 0
   do until arr$(l,0) = ""
     n = 1
     if arr$(l,0) = "" then exit loop
     x$ = arr$(l,0)
     c = 0
     do until a = a+1
       y = instr(n,x$,",")
       if y = 0 then exit loop
       arr$(l,c) = mid$(x$,n,(y)-n)
       n = y + 2
       incr c
     loop
     incr l
     if l = hold then exit loop
   loop
   close #1
end sub

sub openclose(y$,z$)
'*****************************************************************************
'*** MAKE SURE THE FILES EXIST
'***
  open y$+"~pdt."+z$ for output as #1 : close #1
  open y$+"~bom."+z$ for output as #1 : close #1
  open y$+"~cal."+z$ for output as #1 : close #1
  open y$+"~job."+z$ for output as #1 : close #1
  call dumpres(y$+"~mac."+z$)
  open y$+"~wrk."+z$ for output as #1 : close #1
  open y$+"~bns."+z$ for output as #1 : close #1
  open y$+"~que."+z$ for output as #1 : close #1
  open y$+"~rsj."+z$ for output as #1 : close #1
  open y$+"~rsm."+z$ for output as #1 : close #1
  open y$+"~rsg."+z$ for output as #1 : close #1
  open y$+"~crh."+z$ for output as #1 : close #1
  open y$+"~shh."+z$ for output as #1 : close #1
  open y$+"~lsy."+z$ for output as #1 : close #1
  open y$+"~sqr."+z$ for output as #1 : close #1

end sub

sub arraywipe
  shared pjt$(), resources, JobNum, sqnum
  shared pdt$(), bomho$(), cal$(), job$(), mac$(), queues$()
  shared rltj$(), rltm$(), rltg$(), lm1$(), qm1$(), work$()
  shared ls$(), cr$(), sh$(), ls%(), sq$(), bnsch$(), bnnum, calnum

'*****************************************************************************
'*** WIPE PROJECT DATA FILE
'***
```

```
  for i = 0 to 1
    for j = 1 to 50
      pdt$(j,i) = ""
    next j
  next i

'*****************************************************************************
'***  WIPE CALANDER FILE
'***

  for i = 0 to 8
    for j = 0 to calnum
      cal$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE JOB FILE
'***
  for i = 0 to 15
    for j = 0 to JobNum
      job$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE RESOURCE/MACHINE FILE
'***
  for i = 0 to resources
    for j = 0 to 15
      mac$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE QUEUE STATUS STATISTICS
'***
  for i = 0 to resources
    for j = 0 to 4
      queues$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE JOB RESULTS FILE
'***
  for i = 0 to jobnum
    for j = 0 to 22
      rltj$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE RESOURCE RESULTS FILE
'***
  for i = 0 to 48
    for j = 0 to resources
```

```
      rltm$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE GLOBAL RESULTS FILE
'***
  for i = 0 to 1
    for j = 0 to 40
      rltg$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE CR SELECTION HISORY
'***
'*** the five periods are in the five columns
'***
  for i = 0 to 8
    cr$(i,1) = ""
  next i

'*****************************************************************************
'*** WIPE SH SELECTION HISORY
'***
'*** the five periods are in the five columns
'***
'  for i = 0 to 8
'    sh$(i,1) = ""
'  next i

'*****************************************************************************
'*** WIPE LS SELECTION HISORY
'*** the four rows correspond to the DR used.
'***
  for i = 0 to 37
    for j = 0 to 5
      ls$(i,j) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE WORK FILE THIS PERIOD
'***
  for i = 0 to 15
    for j = 0 to JobNum
      work$(j,i) = ""
    next j
  next i

'*****************************************************************************
'*** WIPE BOM FILE
'***
  for i = 0 to 44
    for j = 0 to 20
      BomNo$(j,i) = ""
    next j
  next i
```

```
'*****************************************************************************
'*** WIPE SQRU FILE - sq run database
'***

  for i = 1 to sqnum
    sq$(i,0) = ""
    sq$(i,1) = ""
  next i

'*****************************************************************************
'*** WIPE THE BN SCHEDULE FILE
'***
  for i = 0 to 5
    for j = 0 to bnnum
      bnsch$(j,i) = ""
    next j
  next i
end sub
```

M-SIMCOM.PRO

```
'     program M-SIMCOM.PRO

'     CONTENTS - common simulation routines

'     Edition 1.0   Version 2.0   1/1/91

'     Manufacturing Simulation and Analysis

'     Copyright  S Hurley  J Driscoll

' ***** COMMON SUBROUTINE BLOCK *****

$include "m-scr.pro"

sub working
  shared simu$(), act, mac$, par$, pro$, now&, que1$, Wkg$(),WkgR, rltm$()
  shared action$(), jno$, pdt$(), sqnum

  select case val(pdt$(26,1))
    case = 1
      if act = val(pdt$(27,1)) then call working1
    case = 2
      if val(mac$) = val(pdt$(27,1)) then call working1
    case = 3
      if FnTrun$(Par$) = pdt$(27,1) then call working1
    case = 4
      if FNtrun$(Pro$) = pdt$(27,1) then call working1
    case = 5
      call working1
    case = 6
      do
    call scrclear:locate 10,10:print "Problem in WORKING gosub routine"
        call failsound: call insound
      loop until a = 10
  end select
end sub


sub working1
  shared simu$(), act, mac$, suc$, par$, pro$, now&, que1$, Wkg$(), WkgR
  shared action$(), queues$(), jno$, pdt$(), Inn, rltm$(), sqnum

  select case act
    case = 1
      act$ = "OQ > PQ  Move   "
    case = 2
      act$ = "OQ > CQ  Move   "
    case = 3
      act$ = "CQ > Re  Load   "
    case = 4
      act$ = "PQ > Re  Load   "
    case = 5
      act$ = "Re > OQ  Unload "
    case = 6
      act$ = "Re > CQ  Unload "
    case = 7
      act$ = "Re > PQ  Unload "
    case = 8
```

```
      act$ = "Begin Set Up    "
    case = 9
      act$ = "Begin Operating "  '** begin working
    case = 10
      act$ = "No Parts        "
    case = 11
      act$ = "Move to Stock   "
    case = 12
      act$ = "Jo > PQ  Launch "
    case else
      call scrclear : print "WHOOPS IN working1, in M-SIMCOM"
  end select

  if WkgR < 7 then
    Wkg$(WkgR,0) = FNtrun$(str$(WkgR))
  else
    Wkg$(WkgR,0) = FNtrun$(str$(2 + val(Wkg$(5,0))))
  end if

  Wkg$(WkgR,1) = pdt$(49,1)+"  "+pdt$(50,1)
  if act < 3 then
    x$ = FNtrun$(suc$)
    if val(x$) = 0 then x$ = "Fin"
    y$ = FNtrun$(mac$)
    Wkg$(WkgR,2) = ">"+y$
  else
    Wkg$(WkgR,2) = mac$
  end if
  Wkg$(WkgR,3) = act$
  Wkg$(WkgR,4) = pro$
  Wkg$(WkgR,5) = par$
  Wkg$(WkgR,6) = jno$

  if pdt$(28,1) = "1" and pdt$(2,1) = "1" then
    Tim$ = "W"+pdt$(47,1)+" D"+pdt$(48,1)+" H"+pdt$(49,1)+_
       " M"+pdt$(50,1)
    Sim$ = Sim$ + Tim$ + " "
'    Leg$ = Wkg$(WkgR,1) : Sim$ = Sim$ + Leg$ + " "
    Leg$ = Wkg$(WkgR,2) : Sim$ = Sim$ + Leg$ + " "
    Leg$ = Wkg$(WkgR,3) : Sim$ = Sim$ + Leg$
    Leg$ = Wkg$(WkgR,4) : Sim$ = Sim$ + Leg$
    Leg$ = Wkg$(WkgR,5) : Sim$ = Sim$ + Leg$
    Wkg$(WkgR,6) = JNo$
    Leg$ = jno$ : Sim$ = Sim$ + Leg$
    write #10, sim$
  end if

  if pdt$(25,1) = "1" then
    call Scr1num(WkgR)
  end if
  if WkgR < 7 then incr WkgR
end sub

sub Util(mc,status$)
  shared rltm$(), resources, now&, pdt$(), sqnum
  diff = now& - val(rltm$(3,mc))

  x = asc(rltm$(2,mc))
```

```
    select case x
     case = 83                          '*** (S)et up
       rltm$(4,mc) = str$(val(rltm$(4,mc))+diff)
     case = 87                          '*** (W)orking
       rltm$(5,mc) = str$(val(rltm$(5,mc))+diff)
     case = 73                                          '*** (I)dle
       rltm$(6,mc) = str$(val(rltm$(6,mc))+diff)
     case = 66                          '*** (B)locked
       rltm$(7,mc) = str$(val(rltm$(7,mc))+diff)
     case = 78                          '*** (N)o parts
       rltm$(8,mc) = str$(val(rltm$(8,mc))+diff)
     case = 69
       exit select                      '*** (E)nd
   end select                         '*** USED TO TIE
   rltm$(3,mc) = str$(now&)
   rltm$(2,mc) = status$
end sub


sub que
   shared queues$(), COMP, Inn, R, O, now&, sqnum
   for i1 = 1 to 10
     locate 1+i1+1,2  : print "C";i1;" ";queues$(i1,COMP)
     locate 1+i1+12,2 : print "I";i1;" ";queues$(i1,Inn)
     locate 1+i1+1,40 : print "R";i1;" ";queues$(i1,R)
     locate 1+i1+12,40: print "O";i1;" ";queues$(i1,O)
   next i1
   locate 24,1 : print "now& = ";now&;
end sub

sub getque(querow)
   shared num$, sno$, pro$, ini$, mac$, spr$, srt$, dur$, fin$, suc$, par$
   shared req$, obt$, jno$, tpr$, tba$, tqu$, tti$, cpt$, su1$, su2$
   shared sq$(), sqnum

   x$ = sq$(querow,1)

   num$ = mid$(x$,1,5)  : sno$ = mid$(x$,6,5)  : pro$ = mid$(x$,11,5)
   ini$ = mid$(x$,16,5) : mac$ = mid$(x$,21,5) : spr$ = mid$(x$,26,5)
   srt$ = mid$(x$,31,5) : dur$ = mid$(x$,36,5) : fin$ = mid$(x$,41,5)
   suc$ = mid$(x$,46,5) : par$ = mid$(x$,51,5) : req$ = mid$(x$,56,5)
   obt$ = mid$(x$,61,5) : jno$ = mid$(x$,66,5) : tpr$ = mid$(x$,71,5)
   tba$ = mid$(x$,76,5) : tqu$ = mid$(x$,81,5) : tti$ = mid$(x$,86,5)
   cpt$ = mid$(x$,91,5) : su1$ = mid$(x$,96,5) : su2$ = mid$(x$,101,5)
end sub


sub putque(querow)
   shared num$, sno$, pro$, ini$, mac$, spr$, srt$, dur$, fin$, suc$, par$
   shared req$, obt$, jno$, tpr$, tba$, tqu$, tti$, cpt$, su1$, su2$
   shared sq$(), now&, sqnum

   x$=num$+sno$+pro$+ini$+mac$+spr$+srt$+dur$+fin$+suc$+par$+req$+_
      obt$+jno$+tpr$+tba$+tqu$+tti$+cpt$+su1$+su2$

   sq$(querow,1) = x$
end sub
```

```
sub getres(resrow)
  shared res$, nam$, recod$, grup$, simu$, perf$, oper$, setu$
  shared sre$, fir$, sqo$, fqo$, outt$, inn$, compo$, rule$
  shared mac$(), sqnum

  res$   = mac$(resrow,0)  : nam$  = mac$(resrow,1)
  recod$ = mac$(resrow,2)  : grup$ = mac$(resrow,3)  : sim$ = mac$(resrow,4)
  perf$  = mac$(resrow,5)  : oper$ = mac$(resrow,6)  : setu$ = mac$(resrow,7)
  sre$   = mac$(resrow,8)  : fir$  = mac$(resrow,9)  : sqo$  = mac$(resrow,10)
  fqo$   = mac$(resrow,11) : outt$ = mac$(resrow,12) : inn$   = mac$(resrow,13)
  compo$ = mac$(resrow,14) : rule$ = mac$(resrow,15)
end sub


sub statusline
  shared pdt$(), now&, sqnum

  call fill(15,1,25,1,80,1)

  locate 25,3
  color 15,1 :
  print "Week:      Day:      Mins:        Stop Min:         CR:     SH:";
  color 14,1
  locate 25,9  : print pdt$(47,1);
  locate 25,19 : print pdt$(48,1);
  x$ = FNtrun$(str$(now&))
  locate 25,30 : print using "\    \";x$;
  locate 25,47 : print using "\    \";pdt$(5,1);
  locate 25,59 : print using "\\";pdt$(13,1);
  locate 25,67 : print using "\\";pdt$(14,1);
end sub

sub upto
  shared pdt$(), pjt$(), sqnum
  color 14,9
  locate 1,70 : print " ";time$;" ";
  locate 2,68 : print "          "
  locate 2,68 : print "Wk ";pdt$(47,1);" Dy ";pdt$(48,1)
  locate 3,68 : print "          "
  locate 3,68 : print "Hr ";pdt$(49,1);" Mi ";pdt$(50,1)
  color 15,1
end sub

sub remove(a$(2),i1,j1,k1)        '*** FOR REMOVING FROM QUEUES$()
  re$ = ""
  if k1 <> 1 then re$ = mid$(a$(i1,j1),1,(10*k1)-10)
  a$(i1,j1) = re$ + mid$(a$(i1,j1),(10*k1)+1,len(a$(i1,j1))-(10*k1)+1)
end sub

sub remstr(a$,stepp)
  stepp = stepp
  re$ = ""
  if stepp <> 1 then re$ = mid$(a$,1,(10*stepp)-10)
  a$ = re$ + mid$(a$,(10*stepp)+1,len(a$)-(10*stepp)+1)
end sub

sub acttime
  shared pdt$(), cal$(), now&, sqnum
```

```
  abc = 0
  nowho& = now&
  count = 0
  hours = 0
  mins = 0

  do until nowho& < 60
    nowho& = nowho& - 60
    incr hours
  loop
  mins = nowho&

  x = val(pdt$(43,1))
  y = val(pdt$(44,1))
  abc = 0

  do until hours < 0
    hours = hours - val(cal$(x,y))
    if hours >= 0 then
      incr y
      if y = 8 then
     incr x
     y = 1
      end if
    end if
  loop
  hours = hours + val(cal$(x,y))

  pdt$(47,1) = FNtrun$(str$(x))
  pdt$(48,1) = FNtrun$(str$(y))
  pdt$(49,1) = FNtrun$(str$(hours))
  pdt$(50,1) = FNtrun$(str$(mins))

  if val(pdt$(47,1)) < 10 then
    pdt$(47,1) = "00" + pdt$(47,1)
  elseif val(pdt$(47,1)) < 100 then
    pdt$(47,1) = "0" + pdt$(47,1)
  end if
  if val(pdt$(48,1)) < 10 then pdt$(48,1) = "0" + pdt$(48,1)
  if val(pdt$(49,1)) < 10 then pdt$(49,1) = "0" + pdt$(49,1)
  if val(pdt$(50,1)) < 10 then pdt$(50,1) = "0" + pdt$(50,1)
end sub

!*************************************************************************
!*** PUTS THE RIGHT VALUES IN SQ ARRAY FOR SEQUENCING
!***

sub getsim(jobrow)
  $include "shared.pro"

  call waiton
  count = 0

  for i = 1 to sqnum
    if sq$(i,1) = "" then exit for
    incr count
  next i
  sqrow = count+1
```

```
    file$ = ctl$(15,1)+"-sqd.inf"

    open file$ for input as #4
    do until eof(4)
      input #4, x$
      if FNtrun$(mid$(x$,11,5)) = FNtrun$(work$(jobrow,1)) then exit loop
    loop
    count = sqrow - val(left$(x$,5))

    jobno$ = work$(jobrow,4)

    do until FNtrun$(mid$(x$,11,5)) <> FNtrun$(work$(jobrow,1))
      if mid$(x$,46,1) = "F" then
        lptho$ = mid$(x$,66,5)
      end if
      y$ = mid$(x$,1,5)
      z$ = mid$(x$,6,5)
      jno$ = work$(jobrow,4) : call put5(jno$)
      tba$ = FNtrun$(str$(val(work$(jobrow,3)) * val(mid$(x$,18*5+1,5))))
      call put5(tba$)

      if val(pdt$(16,1)) = 1 then           '*** 1 transfer = process
        tqu$ = tba$
      elseif val(pdt$(16,1)) = 2 then       '*** 2 if global
        tqu$ = FNtrun$(str$(val(pdt$(15,1)) * val(mid$(x$,18*5+1,5))))
      elseif val(pdt$(16,1)) = 3 then             '*** 3 product based
        tqu$ = FNtrun$(str$(val(work$(jobrow,10)) * val(mid$(x$,91,5))) )
      elseif val(pdt$(16,1)) = 4 then       '*** 4 half of job
        tqu =  val(work$(jobrow,3))'*** of tba
        x = 0
        if tqu mod 2 = 1 then x = 1
        tqu = (tqu \ 2) + x
        tqu$ = FNtrun$(str$(tqu * val(mid$(x$,91,5))) )
      end if
      if val(tqu$) > val(tba$) then tqu$ = tba$
      call put5(tqu$)

      y$ = FNtrun$(str$(val(y$) + count + 10000))
      if mid$(x$,6,5) <> "FinPr" then
        z$ = FNtrun$(str$(val(z$) + count + 10000))
      end if
      hold$ = y$+z$+mid$(x$,11,55)+jno$+"0    "+tba$+tqu$+mid$(x$,17*5+1,20)
      if sqrow = sqnum then
        call scrclear
        locate 10,10
        print "Ran out of room in sq$()"
        locate 12,10
        print "reset sqnum inside DIMEN.PRO"
        locate 14,10
        print "and compile MSA"
        call failsound : ghg$=input$(2)
        call edarray(sq$(),sqnum,1,kj)
      end if
'*************************************
'*** TO PUT THE CORRECT TFT VALUE INTO THE SQ DATABASE
      if mid$(hold$,46,1) = "F" then
        setup$ = mid$(hold$,31,5) : nomac$ = mid$(hold$,56,5)
        set2$ = mid$(hold$,101,5)
```

```
      cpsetup = FNsetuptime(setup$,nomac$,set2$)
      x = val(mid$(hold$,47,4))
      y = val(lptho$)
      z = val(work$(jobrow,3)) - 1
      tba = val(tba$)
      tqu = val(tqu$)
      counter = 0
      do until tba <= 0
    tba = tba - tqu
    incr counter
      loop
      tba = tba + tqu : decr counter
      tft = (tba * x) + (counter * tqu * y)

'*** find the number of complete transfer batch sizes not including the last
'*** one and times this by the TB size and the lpt.  Then the remainder which
'*** may be as large as a whole tb is multi-d by the lot-for-lot time.
'*** All in all a lot of jumping through hoops but worth it.  Now late at
'*** night in Blacksburg in the good old US of A, wow.

      x$ = FNtrun$(str$(tft + cpsetup))
      x$ = "F" + x$ : call put5(x$)
      hold$ = mid$(hold$,1,30) + "#    " + mid$(hold$,36,5) + "#    " +_
          x$ + mid$(hold$,51,55)
    end if

    sq$(sqrow,1) =  hold$

    if val(mid$(hold$,16,5)) = 1 then
      initial$ = mid$(hold$,21,5)
      work$(jobrow,15) = work$(jobrow,15) + initial$
    end if

    sq$(sqrow,0) = FNtrun$(work$(jobrow,3))   '*** QUANTITY REQUIRED
    call put5(sq$(sqrow,0))

    incr sqrow
    if eof(4) then exit loop
    input #4, x$
  loop
  call waitoff
  close #4
end sub      '*** getsim

sub getsq
  shared bn$(), resources, sq$(), jobnum, pdt$(), bnsch$(), work$()
  shared num$, sno$, pro$, ini$, mac$, spr$, srt$, dur$, fin$, suc$, par$
  shared req$, obt$, jno$, tpr$, tba$, tqu$, tti$, cpt$, rule$, mac$()
  shared sqnum, bnnum

  count = 0
  do
    incr count
  loop until sq$(count,1) = ""
  sq$(0,0) = FNtrun$(str$(count))


'****************************************
```

```
'****** SETS UP THE SQ ARRAY FOR THE CURRENT WORK FILE
  for k = 1 to jobnum
    if work$(k,1) = "" or work$(k,1) = " " then exit for
    if left$(work$(k,2),1) = "N" then call getsim(k)
  next k

'*************************************
'****** GETS THE CORRECT SET UP TIMES AND LOADS THEM INTO
'****** THE su1$ FIELD IN THE SQ$() DATABASE

  call getsetup

'*************************************
'****** PUTS THE QUANTITY REQUIRED INTO THE LAST COLUMN
  for k = 1 to sqnum
    if sq$(k,1) = "" then exit for
    sq$(k,1) = mid$(sq$(k,1),1,100) + sq$(k,0)
  next k
end sub

sub getbnsch
  $include "shared.pro"
  shared curres

'*************************************************************
'****** PUTS THE NAME OF THE BOTTLENECK AT THE TOP OF THE
'****** BNSCH$() ARRAY

  bnsch$(0,1) = pdt$(13,1)

  if val(pdt$(40,1)) = 1 then
    exit sub '*** set in adv to clear the bnsch
    beep 3
  end if

'*************************************************************
'****** GETS ALL THE JOBS THAT ARE GOING TO GO THROUGH
'****** THE BN AND ORDERS THEM wrt THE SEQUENCING RULE

  for i = 1 to 1          '*** only one bn at the mo'
    if bnsch$(0,(i*2)-1) = "" then exit for
    call getbnjobs(bnsch$(0,(i*2)-1),(i*2)-1)   '*** gets the bn jobs and ld

    j = 1
    k = 1
    do until bnsch$(j,1) = ""
      jn = val(bnsch$(j,4))
      do until jn = val(work$(k,4))
    incr k
    if k = jobnum then
      beep
      call scrclear : locate 1,1 : print "Two ed's"
      call edarray(bnsch$(),bnnum,5,jh)
      call edarray(work$(),jobnum,15,jk)
    end if
      loop
      bnsch$(j,0) = work$(k,13)
      incr j : k = 1
```

```
    loop

    srtrow = 1
    do until bnsch$(srtrow,1) = ""
      perd = val(bnsch$(srtrow,0))
      finrow = srtrow
      do until val(bnsch$(finrow+1,0)) <> perd or bnsch$(finrow+1,1) = ""
    incr finrow
      loop
      if finrow <> srtrow then
    call orderbn(srtrow,finrow)
      end if
      srtrow = finrow + 1
    loop
  next i
end sub  '*** getbnsch

'*********************************************************
'*** to order a set of the bnschedule, from rows
'*** srtrow to finrow.

sub orderbn(srtrow,finrow)
  $include "shared.pro"

  x = val(pdt$(14,1))
  job$ = ""
  prd$ = "" : job$ = "" : lod$ = "" : no3$ = "" : no4$ = "" : no5$ = ""
  for j = srtrow to finrow
    prd$ = prd$ + bnsch$(j,0) : call put10(prd$)
    job$ = job$ + bnsch$(j,1)
    lod$ = lod$ + bnsch$(j,2) : call put10(lod$)
    no3$ = no3$ + bnsch$(j,3) : call put10(no3$)
    no4$ = no4$ + bnsch$(j,4) : call put10(no4$)
    no5$ = no5$ + bnsch$(j,5) : call put10(no5$)
  next j
  row = srtrow
  queue$ = job$
  rule$ = pdt$(14,1)
  i = 1

  do until job$ = ""
    curres = 0
    call rules(queue$)
    y = instr(job$,queue$)
    y = (y-1)/10 + 1
    call remstr(job$,y)
    bnsch$(row,(i*2)-2) = mid$(prd$,(y*10)-9,10) : call remstr(prd$,y)
    bnsch$(row,(i*2)-1) = queue$
    bnsch$(row,(i*2))   = mid$(lod$,(y*10)-9,10) : call remstr(lod$,y)
    bnsch$(row,(i*2)+1) = mid$(no3$,(y*10)-9,10) : call remstr(no3$,y)
    bnsch$(row,(i*2)+2) = mid$(no4$,(y*10)-9,10) : call remstr(no4$,y)
    bnsch$(row,(i*2)+3) = mid$(no5$,(y*10)-9,10) : call remstr(no5$,y)
    incr row
    queue$ = job$
  loop
end sub
```

```
'*****************************************************************
'*** loads all the jobs from que.db that are going through the bn
'*****************************************************************
sub getbnjobs(bn$,col)
  shared sq$(), resources, bnsch$(), sqnum, bnnum
  row = 1
  for i = 1 to sqnum
    if sq$(i,1) = "" then
      exit for
    elseif val(mid$(sq$(i,1),21,5)) = val(bn$) and_
       val(mid$(sq$(i,1),76,5)) > 0 then

'*** The second test is to see it tba$ is greater than 0 which implies
'*** that the machine has already processed the job and hence the
'*** should not be include in the CR schedule.

'locate 3,1
'print "row...";row
'print "col...";col

      bnsch$(row,col) = mid$(sq$(i,1),1,10)
      hold = val(mid$(sq$(i,1),36,5))*val(mid$(sq$(i,1),76,5))
      bnsch$(row,col+1) = FNtrun$(str$(hold))
      bnsch$(row,col+2) = FNtrun$(mid$(sq$(i,1),51,5))    '*** part name
      bnsch$(row,col+3) = FNtrun$(mid$(sq$(i,1),66,5))    '*** job num
      bnsch$(row,col+4) = FNtrun$(mid$(sq$(i,1),11,5))
'*** if the product name was stored then the whole product even if a part
'*** did not go through the bn would be given a priority over others.
'*** Having the part name only then the parts that go throught the
'*** bn have a priority and the others in the same structure do not.
      incr row
    end if
  next i
end sub

sub getsetup                  '*** puts the setup time into su1$
  $include "shared.pro"

  a = val(pdt$(18,1))

  select case a
    case = 1          '*** no set up
      for i = val(sq$(sqnum,1)) to sqnum
    if sq$(i,1) = "" then exit for
    sq$(i,1) = mid$(sq$(i,1),1,95) + "0    " + "0    "
      next i
    case = 2          '*** global set up time
      x$ = pdt$(1,1)
      call put5(x$)

      for i = val(sq$(0,0)) to sqnum
    if sq$(i,1) = "" then exit for
    sq$(i,1) = mid$(sq$(i,1),1,95) + x$ + "0    "
      next i
    case = 3
      for i = val(sq$(sqnum,1)) to sqnum
    if sq$(i,1) = "" then exit for
    x = val(mid$(sq$(i,1),21,5))
```

```
    x$ = mac$(x,7) : call put5(x$)
    sq$(i,1) = mid$(sq$(i,1),1,95) + x$ + "0    "
      next i
    case = 4          '*** product set up, copy su2$ to su1$
      for i = val(sq$(sqnum,1)) to sqnum
    if sq$(i,1) = "" then exit for
    sq$(i,1) = mid$(sq$(i,1),1,95) + mid$(sq$(i,1),101,5) + "0    "
      next i
    case = 5          '*** individual part set up
      '*** leave as is
    case else
      call scrclear
      locate 10,10 : print "WHOOPS IN getset up sub in m-simcom.pro"
      call failsound : x$ = input$(99)
  end select
end sub      '*** getsetup

def FNsetuptime(setup$,nomac$,set2$)     '*** calcs on the fly what the set up will
  $include "shared.pro"             '*** be on the critical path machines.

  a = val(pdt$(18,1))

  select case a
    case = 1          '*** no set up
      FNsetuptime = 0
    case = 2          '*** global set up time
      FNsetuptime = val(pdt$(1,1)) * val(nomac$)
    case = 3
      FNsetuptime = 0
      '*** not sure how to do this one because it requires that the route
      '*** down the critical path be stored.
    case = 4          '*** product set up, copy su2$ to su1$
      FNsetuptime = val(set2$) * val(nomac$)
    case = 5          '*** individual part set up
      FNsetuptime = val(setup$)
    case else
      call scrclear
      locate 10,10 : print "WHOOPS IN FNsetuptime in m-simcom.pro"
      call failsound : call failsound : call insound : x$ = input$(99)
  end select
end def      '*** FNsetuptime

sub statusbox
  shared ctl$(), pdt$(), sqnum

  call shadow(0,3,6,45,32,15)
  call fill(14,1,6,45,32,15)
  color 14,1
  locate 7,47  : print "PROJECT STATUS"
  color 15,1
  locate 9,47  : print "Project name ..... ";
  color 4,7 : print " "; : print using "\ \";ctl$(15,1);
    print " " : color 15,1
  locate 10,47 : print "Project number ..... ";
  x$ = ctl$(16,1) : call toupper(x$)
  if left$(x$,1) = "I" then x$ = "Root"
    color 4,7 : print " "; : print using "\  \";x$; : color 15,1
  locate 11,47 : print ""
```

```
  locate 12,47 : print "Current period ..... ";
    color 4,7 : print " "; : print using "###";val(pdt$(6,1));
    print " " : color 15,1
  locate 13,47 : print ""
  locate 14,47 : print "Time bucket length ... ";
    color 4,7 : print " "; : print using "###";val(pdt$(7,1));
    print " " : color 15,1
  locate 15,47 : print ""
  locate 16,47 : print "Critical resource .... ";
    color 4,7
      if val(pdt$(13,1)) <> 0 then
    print " "; : print using "###";val(pdt$(13,1)); : print " "
      else
    print "none"
      end if
      color 15,1
  locate 17,47 : print "Sequencing heuristic . ";
    color 4,7
      if val(pdt$(14,1)) <> 0 then
    print " "; : print using "###";val(pdt$(14,1)); : print " "
      else
    print "none"
      end if
      color 15,1
  locate 18,47 : print ""
  locate 19,47 : print ""
  locate 20,47 : print ""
end sub      '*** statusbox


sub dumpsheet(file$,arr$(),lines,cols)
  shared pdt$(), sqnum

  open file$ for output as #1
  write #1, pdt$(6,1)+" PERIOD"
  for l = 0 to lines
    x$ = ""
    for c = 0 to cols
      if arr$(l,c) = "" then arr$(l,c) = " "
      x$ = x$ + chr$(34)+arr$(l,c)+chr$(34)
    next c
    y = len(x$)
    x$ = mid$(x$,2,len(x$)-2)
    write #1, x$
  next l
  close #1
end sub

'*************************************************************************
'*** dispatching rules. queue$ = all possible jobs that can be loaded.
'*** rule$ comes from a resource definition
'*************************************************************************
sub rules(queue$)
  shared sq$(), pdt$(), work$()
  shared rule$, curres
  shared num$, sno$, pro$, ini$, mac$, spr$, srt$, dur$, fin$, suc$, par$
  shared req$, obt$, jno$, tpr$, tba$, tqu$, tti$, cpt$
  shared bnsch$(), sqnum, bnnum, jobnum
```

```
''''  shared launch$()  ' taken out to see if used

  select case val(rule$)
    case = 1                    '*** RANDOM
      x = len(queue$)/10
      y% = int(rnd*(x)) + 1
      queue$ = mid$(queue$,(10*y%)-9,10)
    case = 2                                '*** FCFS
      queue$ = left$(queue$,10)
    case = 3               '*** SPT
      durho = 10000
      for j = 1 to len(queue$)/10
    z$ = mid$(queue$,(10*j)-9,5)
        z = val(z$) - 10000
    call getque(z)
    pt = val(dur$) * val(tba$)
    if pt < durho then
      it$ = mid$(queue$,(10*j)-9,10)
      durho = pt
        end if
      next j
      queue$ = it$
    case = 4                    '*** EDD method of sequencing
      best$ = "600000000"
      it$ =   "000000000"
      for j = 1 to len(queue$)/10
    z$ = mid$(queue$,(10*j)-9,5)
        z = val(z$) - 10000
    call getque(z)
    for k = 1 to jobnum
      if val(jno$) = val(work$(k,4)) then
        dd$ = FNtrun$(work$(k,5)) + FNtrun$(work$(k,6)) +_
          FNtrun$(work$(k,7)) + FNtrun$(work$(k,8))
        exit for
      end if
    next k
    if val(dd$) < val(best$) then
      it$ = mid$(queue$,(10*j)-9,10)
      best$ = dd$
    end if
      next j
      queue$ = it$
    case = 5
      queue.ho$ = queue$
      rule$ = "4"
      call rules(queue$)
      quehold$ = quehold$ + queue$
      y = instr(queue.ho$,queue$)
      call remstr(queue.ho$,y)
      queue$ = queue.ho$
      if queue$  <> "" then
    call rules(queue$)
    quehold$ = quehold$ + queue$
    y = instr(queue.ho$,queue$)
    call remstr(queue.ho$,y)
    queue$ = queue.ho$
```

```
  end if
  if queue$  <> "" then
call rules(queue$)
quehold$ = quehold$ + queue$
y = instr(queue.ho$,queue$)
call remstr(queue.ho$,y)
queue$ = queue.ho$
  end if
  if queue$  <> "" then
call rules(queue$)
quehold$ = quehold$ + queue$
y = instr(queue.ho$,queue$)
call remstr(queue.ho$,y)
queue$ = queue.ho$
  end if
  if queue$  <> "" then
call rules(queue$)
quehold$ = quehold$ + queue$
y = instr(queue.ho$,queue$)
call remstr(queue.ho$,y)
queue$ = queue.ho$
  end if
  queue$ = quehold$
  rule$ = "5"
  call rules(queue$)
case = 10                  '*** TAKES THE BN SCHEDULE AND USES IT
  occ = 999
  for k = 1 to len(queue$)/10

z = val(mid$(queue$,(10*k)-8,4))
x$ = mid$(sq$(z,1),51,5)  '*** part name hold
x$ = FNtrun$(x$)
y = val(mid$(sq$(z,1),66,5))  '*** job number hold
    j = 1
do until bnsch$(j,1) = ""
  if x$ = FNtrun$(bnsch$(j,3)) and y = val(bnsch$(j,4)) then

    if j < occ then
         occ = j
      it$ = mid$(queue$,(10*k)-9,10)
       end if
    if j = 1 then exit for
       exit loop
      else
       incr j
      end if
loop
  next k
  if occ <> 999 then
if val(pdt$(13,1)) = curres then
  for k = occ to bnnum
    if bnsch$(k,1) = "" then exit for
    bnsch$(k,1) = bnsch$(k+1,1)
    bnsch$(k,2) = bnsch$(k+1,2)
    bnsch$(k,3) = bnsch$(k+1,3)
    bnsch$(k,4) = bnsch$(k+1,4)
    bnsch$(k,5) = bnsch$(k+1,5)
  next k
```

```
 end if
 queue$ = it$
  else
 rule$ = pdt$(14,1)
 call rules(queue$)
 rule$ = "10"
  end if
case = 40
  flag = 0
  n = 1
  do
    x$ = left$(bnsch$(n,1),5)
    x$ = str$(val(x$) - 10000)
    x$ = sq$(val(x$),1)
    bnjobno$ = mid$(x$,66,5)
    for i = 1 to 20
      if bnjobno$ = q$(i,1) then
        flag = 1
        exit loop
      end if
    next i
    incr n
  loop until bnsch$(n,1) = ""
  if flag = 0 then
    rule$ = "1"
    call rules(queue$)
    rule$ = "4"
  else
    queue$ = mid$(queue$,(n*10),9)
  end if
case = 55
  flag = 0
  x1$ = left$(queue$,5)
  x1 = val(x1$) - 10000
  mac = val(mid$(sq$(x1,1),21,5))
  launch$ = launch$(mac,1)
  for l = 1 to (len(launch$)/5)
    y$ = mid$(launch$,(5*l)-4,5)
    n = 1
    do
      y = instr(n,queue$,y$)
      if y <> 0 then
        queue$ = mid$(queue$,y,10)
        flag = 1
        exit for
      else
        n = n + 10
      end if
    loop until n > len(launch$)
  next l
  if flag = 0 then
    rule$ = "1"
    call rules(queue$)
    rule$ = "5"
  end if
case else
  call failsound
  call insound : call scrclear : locate 10,39 : print "WHOOPS - RULES"
```

```
      xx$=input$(30)
  end select
end sub

sub sqbits(row)
  $include "shared.pro"

  locate 3,1
  CALL GETQUE(ROW)
  print "num$ ";num$
  print "sno$ ";sno$
  print "pro$ ";pro$
  print "ini$ ";ini$
  print "mac$ ";mac$
  print "spr$ ";spr$
  print "srt$ ";srt$
  print "dur$ ";dur$
  print "fin$ ";fin$
  print "suc$ ";suc$
  print "par$ ";par$
  print "req$ ";req$
  print "obt$ ";obt$
  print "jno$ ";jno$
  print "tpr$ ";tpr$
  print "tba$ ";tba$
  print "tqu$ ";tqu$
  print "tti$ ";tti$
  print "cpt$ ";cpt$
  print "su1$ ";su1$
  print "su2$ ";su2$
  sd$=input$(1)
end sub
```

COMMON

```
common ctl$(2),_
  z, A%, bomho$(2),_
  event$(2), ma$(2), queues$(2),_
  p1%(2), mi$(2), sy$(2), simu$(2), pjt$(2), SimRun$(2),_
  job$(2),load$(2), hold$(2), jobho$(2),_
  res$(2), Wkg$(2),_
  cal$(2), bom$(2), load&(2),_
  pdt$(2), rltj$(2), rltm$(2), rltg$(2), cr$(2), sh$(2), ls$(2), work$(2),_
  g$(2), m$(2), file1$, file2$, hold1$(2), p1$(1), p2$(1),_
  now&, Nam$, ReCod$,Grup$, Simu$, Perf$, Oper$, SetU$,_
  SRe$, FiR$, SQO$, FQO$, Outt$, Inn$, comp$,_
  rule$, res$,_
  Num$, SNo$, Pro$, ini$, mac$, SPr$, Srt$, Dur$, Fin$,_
  Suc$, Par$, Req$, Obt$, JNo$, TPr$, TBa$, TQu$, TTi$, Cpt$,su1$,su2$,_
  sq$(2), mac$(2), turn$(1), launch$(2), bnsch$(2), bn$(2),_
  COMP, Inn, R, O, jobnum, resources, sqnum, bnnum,_
  bomfile$(2), prod$(2), QT$(2), WKGR, TestCaseNo, test$(2), calnum
```

PAGE NUMBERING AS ORIGINAL

```
dep = 20
dim g$(0:dep,0:2)              '*** COLORS DATABASE - NAMES
dim ctl$(0:50,0:1)      '*** NEW MAIN CONTROL FILE
dim g%(0:dep,0:1)              '*** COLORS DATABASE - FIGURES
dim u1$(0:10,0:6)              '*** USER PERMANENT FILE
dim u2$(0:10,0:50)             '*** USER RECORD FILE
dim m$(0:dep,0:4)              '*** MACHINE INFO FILE
dim s1$(0:dep,0:1)             '***
dim ac$(0:40,0:4)              '*** ACCESS DATA
dim p1%(0:dep,0:1)             '***
dim mi$(0:20,0:2)           '*** MISCELANEOUS CONTROL VARIABLES
dim sy$(0:20,0:1)              '*** NAMES OF ALL THE SYSTEM FILES
dim simu$(0:20,0:1)         '*** SIMULATION DATA FILE
dim pjt$(0:15,0:15)         '*** PROJECT ATTRIBUTES
dim SimRun$(0:20,0:20)  '*** HOLDS THE INFO ABOUT A PROJECTS SIMULATION RUNS

jobnum = 200
dim job$(0:jobnum,0:15)     '*** THE JOBS PENDING FILE
dim jobho$(0:100,0:15)
dim hold$(0:20,0:6)         '*** NOT SURE???
dim bomho$(0:20,0:64)          '*** USED IN CREATE BOM ROUTINE
calnum = 200
dim cal$(0:calnum,0:8)      '*** CALENDAR
dim Wkg$(0:9,0:6)              '*** FOR THE SCROLLING
resources = 20
dim queues$(0:resources,0:4)    '*** CARRIES ALL THE QUEUES
dim res$(0:resources,0:15) '*** RESOURCE STATUS
sqnum = 1500
dim sq$(0:sqnum,0:1)           '*** THE SIMQUE FILE
dim mac$(0:resources,0:15)   '*** THE RESOURCES PROFILE
dim bn$(0:resources,0:2)   '*** THE LOADING AND THE SET OF EACH MACHINE
bnnum = 150
dim bnsch$(0:bnnum,0:5)         '*** THE SCHEDULES OF THE BOTTLENECKS

COMP = 1
Inn = 2
R = 3
O = 4
dim pdt$(0:50,0:1)      '*** project data
dim rltj$(0:jobnum,0:22)   '*** results for job file
dim rltm$(0:49,0:resources)'*** results for machines
dim rltg$(0:40,0:1)         '*** global aggregates
dim cr$(0:8,0:1)        '*** critical resource selection history
dim work$(0:jobnum,0:15)   '*** work queue
dim ls$(0:37,0:5)       '*** new learning system
dim load&(0:resources,0:1) '*** loading calculation
dim p1$(66)             '*** for printing reports
dim p2$(66)             '*** for printing reports

dim test$(0:30,0:14)         '*** test case configurations

dim sh$(0:8,0:1)        '*** sequencing selection history
dim hold1$(0:35,0:21)       '*** for ceating queues
dim launch$(0:resources,0:6)
dim bomfile$(0:200,0:1)
dim prod$(0:40,0:1)
dim bom$(0:200,0:1)
dim queFin$(0:21,0:10) '*** HOLDS THE JOBS THAT HAVE FINISHED
```

SHARED.PRO

```
shared bn$(), bnsch$(), BomHo$(), bom$(), cal$(), cr$(),_
  ctl$(),_
  HOLD1$(), job$(), load&(), ls$(), res$(), pjt$(), queues$(),_
  pdt$(), rltj$(), rltm$(), rltg$(), sh$(),_
  work$(), mac$(), turn$(), resources,_
  wkg$(), launch$(), sq$(), simu$(),_
  res$, nam$, recod$, grup$, simu$, perf$, oper$, setu$,_
  sre$, fir$, sqo$, fqo$, outt$, inn$, compo$, rule$, QT$(),_
   Z, jobnum, sqnum, bnnum,_
   p1$(), p2$(), TestCaseNo, Test$(), calnum, jobho$()

shared num$, sno$, pro$, ini$, mac$, spr$, srt$, dur$, fin$, suc$, par$
shared req$, obt$, jno$, tpr$, tba$, tqu$, tti$, cpt$, su1$, su2$
shared WkgR, Now&, Now.ho&, SimNo,  Intpt
shared COMP,Inn,R,O, que1$, act, movement
```