THE PRODUCTION PLANNING PROBLEMS OF FLEXIBLE MANUFACTURING SYSTEMS WITH HIGH TOOL VARIETY

Vol. 2

User Manual
(including 4 program diskettes)

by
D T S Perera

THESIS
presented to the
Department of Design, Manufacture and Engineering Management
of the
University of Strathclyde
for the Degree of
Doctor of Philosophy

UNIVERSITY OF STRATHCLYDE
GLASGOW
1988

ABSTRACT

A suite of computer simulation software which mimics the operations of a complex flexible manufacturing system was developed to support the research programme. This user manual provides instructions for the program installation and execution. It also describes all program modules and their associated data files in detail.

The simulation software system consists of three major modules, a part flow simulator, a graphical post-processor and a tool post- processor. A sample data set extracted from a real FMS has been included.

A variety of experimental environments can be created by the user by altering data elements in input data files and/or appending new data to input files.

CONTENT

CHAPTER 1 INTRODUCTION TO THE SIMULATION SYSTEM 1

1.1 Introduction 1

1.2 The Development History 1

1.3 Major Modules of the simulator 2

1.4 The Organization of the simulation system 4

1.5 The Organization of the user manual 4

CHAPTER 2 SYSTEM REQUIREMENTS AND INSTRUCTIONS FOR PROGRAM INSTALLATIONS AND EXECUTION 6

2.1 Introduction 6

2.2 Hardware and software requirements 6

2.3 Supplied software 6

2.4 Notation for user interactive commands 8

2.5 Program installation 8

2.6 Program execution 8

CHAPTER 3 THE PART FLOW SIMULATOR 11

3.1 Introduction 11

3.2 User options 11

3.3 Input data file structures 14

3.3.1 ROUT.DAT input file 17

3.3.2 FIXT.DAT input file 19

3.3.3 SCHD.DAT input file 23

3.3.4 BGIN.DAT input file 27

3.3.5 GASP.DAT input file 30

3.3.6 CTRL.DAT input file 32

3.4 System data files 36

3.4.1 Part route files 38

3.4.2 Tooling required files 43

3.4.3 Tool database file 45

3.5 Output files 46

3.6 Program routine description 54

CHAPTER 4 GRAPHICAL POST-PROCESSOR 64

4.1 Introduction 64

4.2 Graphical Kernel System 64

4.3 Design considerations 65

4.4 Input Data requirements 68

4.5 Data file structures 70

4.5.1 LOCT.DAT input data file 72

4.5.2 CLOK.DAT input data file 74

4.6 Program routines 75

CHAPTER 5 THE TOOL POST-PROCESSOR 79

5.1 Introduction 79

5.2 Organization of the tool post-processor software system 79

5.3 User Options 81

5.4 GNTR.EXE program module 83

5.5 GNDF.BAT program module 87

5.6 GNTT.EXE program module 93

5.7 GNTS.EXE program module 98

5.8 GNTC.EXE program module 102

5.9 ASTM.EXE program module 106

5.10 TPOS.EXE program module 114

5.11 GNRP.EXE program module 123

CHAPTER 6 APPLICATIONS, LIMITATIONS AND FURTHER ENHANCEMENTS 126

6.1 Application areas for general users 126

6.2 Limitations of the simulation system 126

6.3 Further enhancements 127

6.4 Conclusion 128

REFERENCES 129

CHAPTER 1

INTRODUCTION TO THE SIMULATION SYSTEM

1.1 INTRODUCTION

When the production planning and control problems of flexible manufacturing systems are solved, a model which can mimic the essential features of the FMS is an invaluable tool. During the course of this research work a comprehensive FMS simulator, a graphical post-processor and a tool post-processor were developed.

This user manual provides a detailed description of all modules of the simulation system. It also include instructions for program installation and execution.

1.2 THE DEVELOPMENT HISTORY

One of the major drawbacks of existing simulation software is that they cannot handle a large amount of data effectively and efficiently. For example, when the tool flow of a manufacturing system is modelled, the management of tool data is an important task. The existing simulation systems fail to meet this requirement.

The simulation routine libraries such as GASP [Pritsker, 1975], provide a set of basic routines required for any simulation project. The system logic is modelled by the analyst using a general purpose programming language (for example FORTRAN for GASP). This use of general a purpose programming language provides a higher flexibility to the analyst and efficient program routines can be constructed to handle a large amount of data.

The first FMS simulator of this series was developed on VAX 11/780 minicomputer under VAX/VMS operating environment using FORTRAN 77 and modified GASP routines. This model was extensively used for the research programme and it was extended to include tool flow modelling routines and the part selection algorithm routines.

The second version of the simulator was developed on PDP 11/40 minicomputer for an industrial application [Perera 1986, Perera and Carrie 1987]. This suite of software modules include a data pre-processor and several software interfaces to the FMS database. At present this simulator is used by a collaborative company to support the management decision making process.

More recently, the major program modules were successfully transferred to a microcomputer system (IBM PC/AT or compatible). Modular programming techniques were used to reduce the overall program size and a few more new features were added to the simulation system.

The main objective of this user manual is to explain the operation of the microcomputer version of the simulator and to provide instructions to the user.

1.3 MAJOR MODULES OF THE SIMULATOR

The simulation system consists of three major modules;

a. The Part Flow Simulator

This is the kernel of the simulation system. The model mimic the operations of a complex flexible manufacturing system. The user can set up different experimental environments by altering data in input data files. The simulator also generates the data required for the tool post-processor and the graphical post-processor. A number of user selective output reports have been built into the system.

b. The Graphical Post-processor

This module links the event file generated by the part flow simulator with other data files to animate the movements of entities. The user can select the overall animation speed and the pallet transfer speed. The second option provides an opportunity to trace the pallet movements at a low speed.

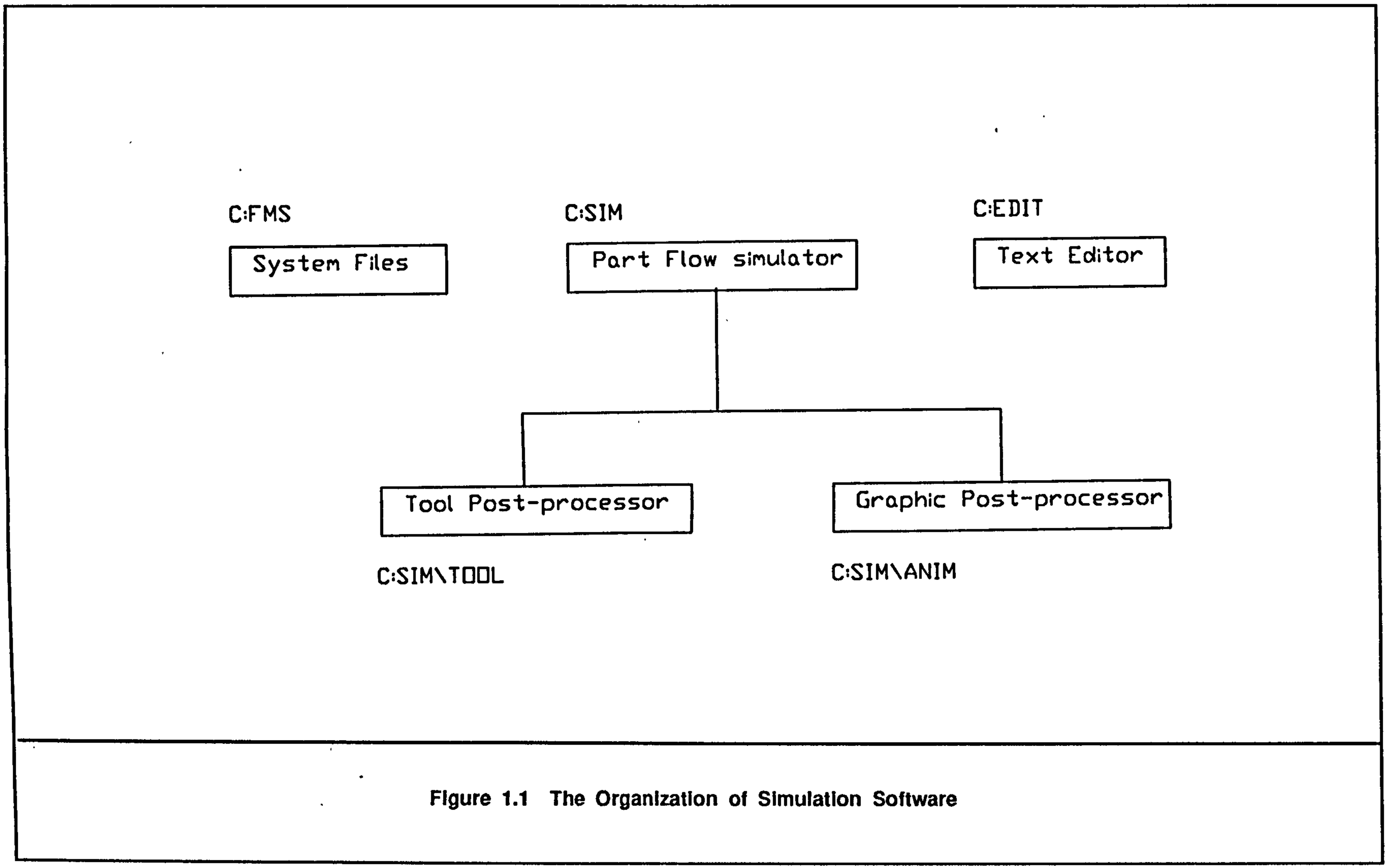


Figure 1.1 The Organization of Simulation Software

c. The Tool Post-processor

The part flow simulator generates a data file containing the information related to all cutting operation carried out within the simulated setting. The tool post-processor blends this data with data extracted from a large tool database to estimate the certain parameters of the tool management system (see Vol. 1 chapter 7 for more information about the tool management parameters). It also provides information for tool requirement planning.

1.4 THE ORGANIZATION OF THE SIMULATION SYSTEM

The Fig. 1.1 depicts the organization of the simulation system in MS-DOS operating environment. The C:SIM sub- directory contains the part flow simulator and its associated data files. The graphical post-processor and the tool post-processor and their related data files reside in the C:\SIM\ANIM and C:\SIM\TOOL sub-directories respectively.

The user can install an editor (or a word-processor) in the C:EDIT sub-directory. A number of batch command files have been provided to facilitate the data file exchange between sub-directories.

The C:FMS sub-directory consists of data files associated with a real FMS. The part requirement files, the tooling required files and a simplified version of the tool data base can be found in this directory. The program modules extract data from this system database.

1.5 THE ORGANIZATION OF THE USER MANUAL

Chapter 2 outlines the hardware and software requirements for the simulation system. It also provides the instructions to install the system and to execute all major program modules. Chapter 3 contains a detailed description of the part flow simulator and chapters 4 and 5 describe the operation of the graphical post-processor and the tool

post- processor respectively. In chapter 6, applications and limitations of the simulation system are discussed and further enhancements are suggested.

CHAPTER 2

SYSTEM REQUIREMENTS AND INSTRUCTIONS FOR PROGRAM INSTALLATION AND EXECUTION

2.1 INTRODUCTION

This chapter describes the hardware and software requirements for the simulaton software and provides instructions for the program installation and execution.

2.2 HARDWARE AND SOFTWARE REQUIREMENTS

The following system of hardware is required for the simulation software;

a. IBM PC/AT or compatible with 512 K bytes of memory.

b. 80287 Intel Maths Co-processor.

c. EGA or compatible display and adopter (for the animation module only).

d. Hard disk drive with minimum of 1.5M bytes free disk space.

The MS-DOS 2.2 (or above) operating system is required.

2.3 SUPPLIED SOFTWARE

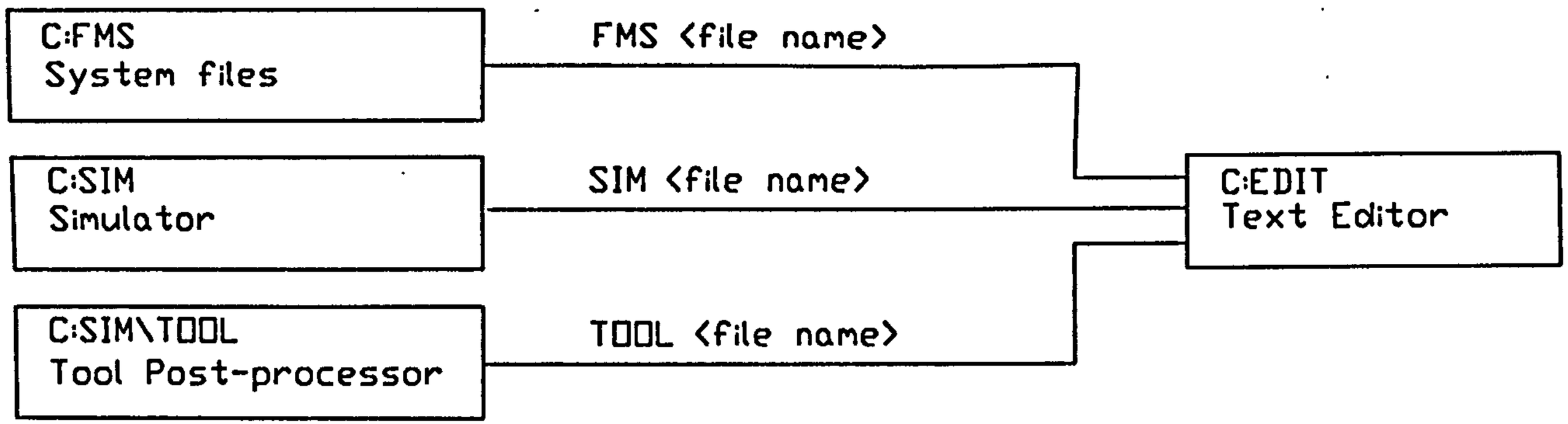
The simulation system is supplied in four 360 K bytes (5 1/4 inches) diskettes. The content of each diskette is as follows;

DISK-01 : The installation program.
System initialization program.
System data files.

DISK-02 : The part flow simulator and a sample data set.

DISK-03 : The tool post-processor and a sample data set.

DISK-04 : The graphical post-processor and a sample data set.



a. To edit a text file type EDIT <file name>. This command automaticaaly copies the file into C:EDIT directory and invoke the text editor.

Ex: To edit ROUT.DAT (in sub-directory C:SIM) type;

EDIT ROUTE.DAT

b. To send edited file back to the previous directory, type the appropriate command shown above.

Ex: To send ROUT.DAT back to the original directory type;

SIM ROUT.DAT

Figure 2.1 Use of Batch Command File

2.4 NOTATION FOR USER INTERACTIVE COMMANDS

The prompts from the operating system and/or the simulation system are shown in italic an the user responses are shown in **bold** characters.

Ex: To execute the simulator, type the following command and press RETURN.

C:>**sims**

2.5 PROGRAM INSTALLATION

A program which guides the user through the installation procedure has been provied.

Installation Procedure

Place the installation diskette in the drive A and type the following command;

a:intall

Simply follow the instructions which appear on the screen.

Installing an Editor

The user can install a text editor in C:EDIT sub- directory. A number of batch files have been included to facilitate file transfer between the C:EDIT directory and other directories. The use of batch files are shown in Fig. 2.1.

Note: These batch files assume that WordStar (WS to invoke it) has been intalled as the text editor. If the user wishes to install a different type of text editor, WS command in all batch files must be replaced by the appropriate invoking command.

2.6 PROGRAM EXECUTION

A sample data set has been provided with the software system and the user can use these files to understand the basic operational features of the simulation system. It is

reminded here, however that the user can setup different experimental conditions by changing a variety of input variables (see chapters 3 to 5 for more information).

The program execution procedure is as follows;

1. Initialising (booting-up) the computer system

a. Place the system initialization disk (DISK-01) in the drive A.

b. Switch on the computer (if it is already switched on press Alt and Ctrl keys together and followed by **Del** key). Once the DOS prompts (date and time) are provided with appropriate values, the following prompt must appear on the screen;

c:

2. Running the Part Flow Simulator

a. Change the directory.

***c:*cd\sim**

b. Execute the SIMS.EXE program by typing;

***c:*sims**

A welcome message will appear on the screen.. No user interaction is required, during the execution.

The following output files can be listed or printed;

WRPT.OUT - Weekly performance reports.

FLOW.OUT - Throughput time of individual parts.

BTCH.OUT - Throughput time of individual production orders.

TRAC.OUT - Entity tracing data.

3. Running the the Graphical Post-processor

a. Change the directory.

c:>\cd\sim\anim

b. Run the animation program.

c:>anim

The user will be prompted for animation speed values. The animation speed depend on the type of the central processing unit(CPU) and bus architecture used in the computer system. Therefore in very first execution of the animation module, the user is advised to set these values to 1.0. Having observed the animation if the user wishes, a lower value can be specified.

4. Running the Tool Post-processor

a. Change the directory.

c:>cd\sim\tool

b. Run the tool post-processor.

c:>tool

c. The output data are stored in the following files;

WRTP.OUT - Weekly performance reports.

TPLN.OUT - Tool requirement planning information.

CHAPTER 3

THE PART FLOW SIMULATOR

3.1 INTRODUCTION

This is the kernel of the simulation software system. It allows the user to perform experiments under variety of different input conditions. It also generates the data required for the post- processors. The data file organization is depicted in Fig. 3.1.

In this chapter, the input/output data file structures and the major program routines are explained. When the input file structures are described, the options available to the user to alter experimental conditions are also outlined.

The sample data provided in the software system were extracted from a real FMS. The structure of input/output data files are explained with reference to this system. The layout of the system is shown in Fig. 3.2.

3.2 USER OPTIONS

User can create different experimental environments by altering data in input data files and or adding new data to them. In the following the options available to the user are outlined (for quick cross reference, section numbers are given);

a. Changing FMS database (section 3.4).

The data file containing the part route information, tool requirements, etc. reside in the C:FMS directory. The user can change these data and/or new parts and tools can be added to the system database.

b. Changing Fixture Assignment Data (section 3.3.2)

- Changing the number of assignments of a given fixture.

- Adding new fixture types.

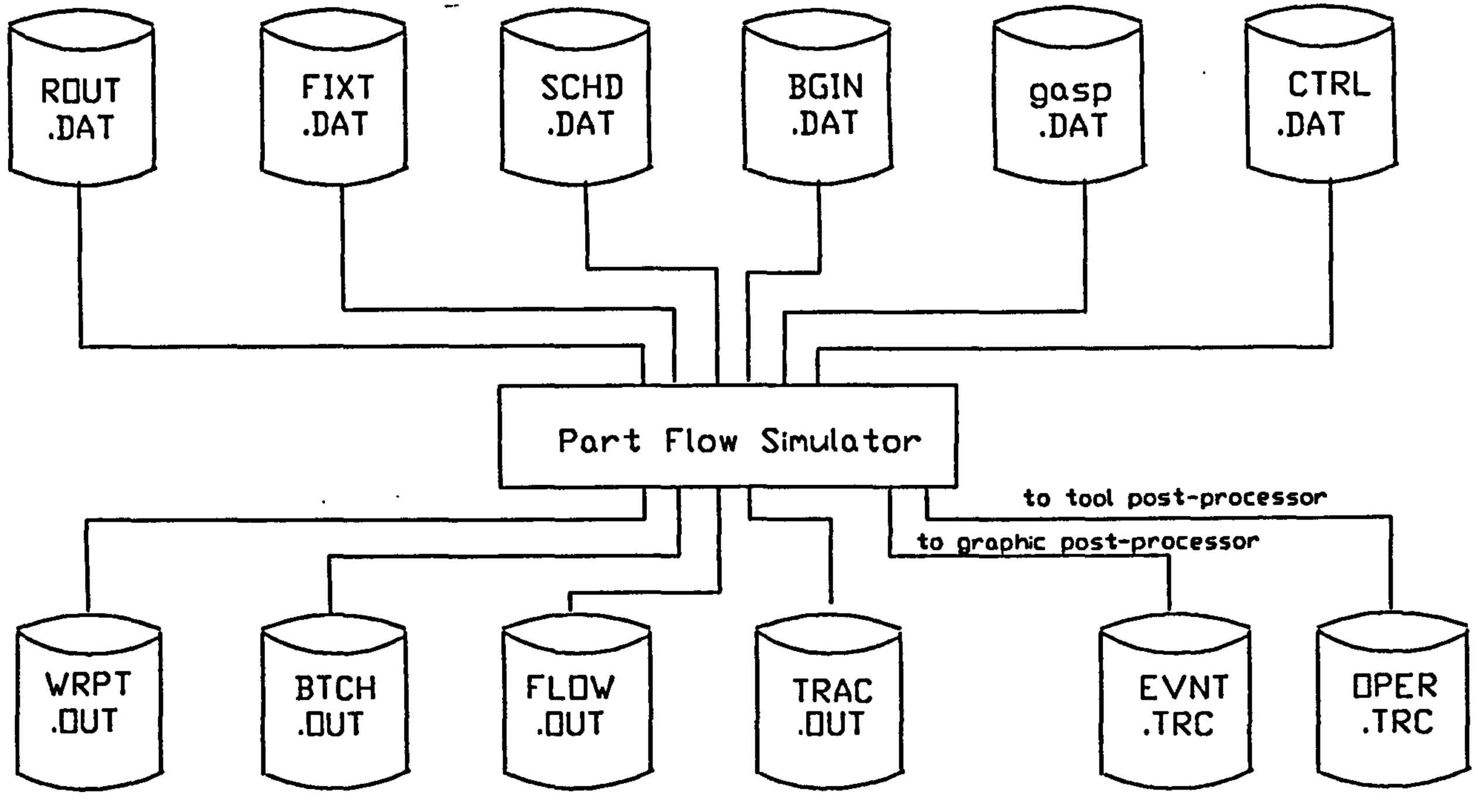


Figure 3.1 The Organization of Input/Output Files

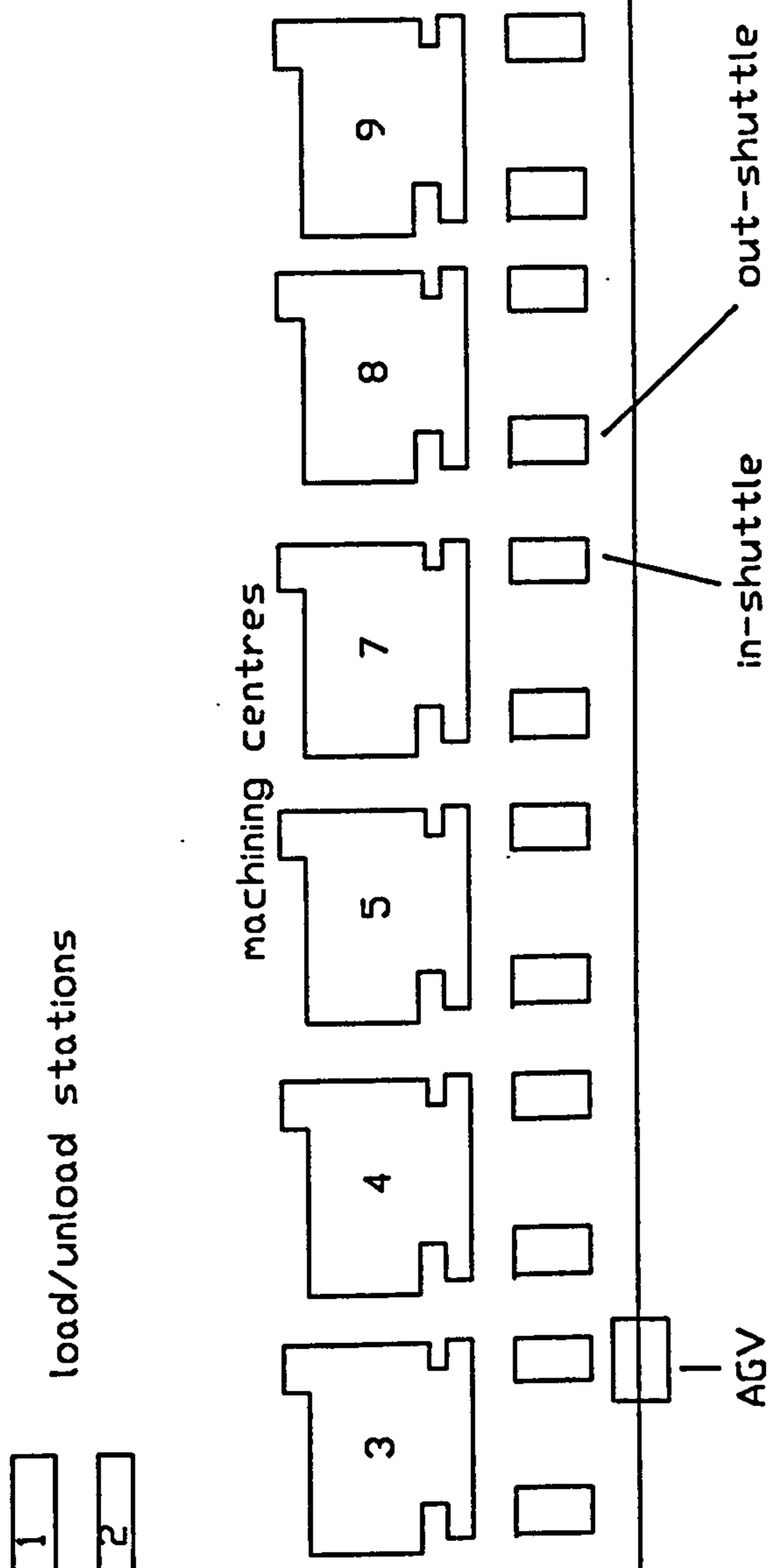


Figure 3.2 The Flexible Manufacturing System

- Deleting existing fixture types.

c. Production Requirements Data (section 3.3.3)

- Changing production requirements.(quantities, due date, raw part availability etc.)

- Adding new production orders.

- Deleting existing production orders.

d. Initial Condition and System Configuration (section 3.3.4)

- Defining a busy initial condition.

- Altering the number of stations.

- Altering the number of pallets.

e. Program Control Data (section 3.3.5)

- changing activity prioirty order.

- selecting output report options.

3.3 INPUT DATA FILE STRUCTURES

The simulator uses the following input data files.

a. ROUT.DAT (section 3.3.1) - Part route data.

b. FIXT.DAT (section 3.3.2) - Fixture assignment data.

c SCHD.DAT (section 3.3.3) - Production requirement data.

d. BGIN.DAT (section 3.3.4) - Initial condition data.

e. GASP.DAT (section 3.3.5) - Values of GASP variables.

f. CTRL.DAT (section 3.3.5) - Program control switches.

Each of these files are explained in the section shown next to data file name.

```
:
: Part route data.
:
12                                  <- Number of part types
:
        1,18                        <- Part type identification number
:                                       Number of operations
        1,1,2,1,20.00,2,20.00       <- Operation sequence number
        2,1,2,4,81.19,5,81.19          Fixture stage number
        3,1,1,8,48.56                  Number of alternative stations
        4,1,2,1,20.00,2,20.00          First alternative station number
        :                              Operation time at the first alternative station
        5,2,2,1,20.00,2,20.00          Second alternative station number
        6,2,2,4,343.46,5,343.46        Operation time at the second alternative station
        7,2,2,1,20.00,2,20.00
        8,2,1,3,28.25
        9,2,2,1,20.00,2,20.00
:
        10,3,2,1,20.00,2,20.00
        11,3,1,8,87.84
        12,3,2,1,20.00,2,20.00
:
        13,4,2,1,20.00,2,20.00
        14,4,1,8,148.35
        15,4,1,9,69.23
        16,4,2,7,76.40,8,76.40
        17,4,1,3,106.60
        18,4,2,1,20.00,2,20.00
    :
    2,17
    :
        1,1,2,1,30.0,2,30.0
        2,1,2,4,168.52,5,168.52
        3,1,2,1,20.00,2,20.00
    :
        4,2,2,1,20.00,2,20.00
        5,2,2,4,190.60,5,190.60
        6,2,2,1,20.00,2,20.00
        7,2,1,7,54.42
        .. .. .. .. ..
```

Figure 3.3 ROUT.DAT Input Data File

3.3.1 ROUT.DAT Input File

Description : This file contains the part route data for each part type in the system.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.3.

User Options :

This file is automatically created by GRPR.EXE program. Please see section 3.5 for information about changing part route data.

```
:
: Fixture assignment data.
:
34 <-- Number of fixtures
      1,1,6,1   <-- Fixture number
      2,1,6,2   <-- Number of assignments
      3,1,12,1  <-- Part type ID of the 1st assignment
      4,1,12,2  <-- Fixture stage of the 1st assignment
      5,1,9,1
      6,1,9,2
      7,1,11,1
      8,1,11,2
      9,1,4,1
      10,1,4,2
      11,1,4,3
      12,1,2,1
      13,1,2,2
      14,1,2,3
      15,1,3,1
      16,1,3,2
      17,1,3,3
      18,1,5,1
       ..  ..
```

Figure 3.4 FIXT.DAT Input Data File

3.3.2 FIXT.DAT Input File

Description : This file contains the fixture assignments data.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.4.

User Options:

a. Changing the number of assignments for a given fixture.

A fixture can be assigned up to a maximum of 4 different fixture stages (same part type or different part types). Simply change the number of assignments for the fixture and add the new list of assignments.

Ex: At present the fixture no. 5 is only used in 1st fixture stage of part type 9. It has been found this fixture can be used for 2nd fixture stage of the same part type as well as 4th fixture stage of part typ

The current fixture assignments data;

34
1,1,6,1
2,1,6,2
3,1,12,1
4,1,12,2
5,1,9,1 <----
6,1,9,2
.. ..

The new fixture assignments data;

34
1,1,6,1
2,1,6,2
3,1,12,1
4,1,12,2
5,3,9,1,9,2,1,4
6,1,9,2
.. ..

b. Adding new fixtures.

Up to 100 fixtures can be used. Add the new record containing fixture assignment data and increase the number of fixtures.

Ex: A new fixture (fixture no. 35) has been designed. This fixture can be used in 1st fixture stage of part type 4 and 2nd fixture stage of part type 10.

The current fixture assignment data;

34
1,1,6,1
2,1,6,2
3,1,12,1
.. ..
.. ..
33,1,1,3
34,1,1,4

The new fixture assignment data;

35

1,1,6,1
2,1,6,2
3,1,12,1
.. ..
.. ..
33,1,1,3
34,1,1,4
35,2,4,1,10,2

c. Deleting fixtures.

Simply delete the corresponding record and adjust the number of fixtures.

: Production orders.
:
1,2 ← Order processing start week number, The number of orders
1,8,10,4 ← Order identification code, Part type identifiaction code, Order quantity, Due week number
2,2,8,4
3,10,4,3
:
3,5
4,4,4,8
5,5,9,6
6,9,3,5
7,11,2,4
8,7,3,4

Figure 3.6 SCHD.DAT Input Data File

3.3.3 SCHD.DAT Input File

Description : This file contains production requirement data.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig 3.5.

User Options:

a. Adding new production orders.

Up to 50 production orders can be included in this file. Simply add the production order record under the correct starting week number.

Ex: A new production order, 12 parts of type 1 has to be added to the production order list. However raw casting will not be available until the beginning of the 3rd week. The production order no. is 10 and due week no. is 8.

The current production requirements;

```
: Production orders.
:
1,2
    1,1,10,4
    4,12,5,3
:
4,3
    8,4,4,7
    7,3,10,9
    5,4,2,10
```

The new production requirements;

: Production orders.
:
1,2
    1,1,10,4
    4,12,5,3
:
4,3
    8,4,4,7
    7,3,10,9
    5,4,2,10
:
3,1
    10,1,12,8

b. Deleting production orders.

Delete the production order record(s) and adjust the number of of orders in that group.

Ex: The production order no. 11 has to be deleted from the current production requirement list.

The current production requirements;

    .. ..
4,2
    10,3,4,8
    15,2,10,10
6,2
    11,1,10,12 <-----
    8,5,6,8

The new production requirements;

```
..  ..
4,2
    10,3,4,8
    15,2,10,10
6,1
    8,5,6,8
```

```
:
: Initial configuration.
:
: Load/unload stations.
2
  1,2
        1,0,0,0,0,0,1
  2,2
        2,0,0,0,0,0,2
:
: Machining centres.
6
  3,1,1,1
        3,0,0,0,0,0,3
        3,0,0,0,0,0,0
        3,0,0,0,0,0,4
  4,1,1,1
        4,0,0,0,0,0,0
        4,0,0,0,0,0,0
        4,0,0,0,0,0,5
  5,1,1,1
        5,0,0,0,0,0,6
        5,0,0,0,0,0,0
        5,0,0,0,0,0,7
  7,1,1,1
        7,0,0,0,0,0,8
        7,0,0,0,0,0,0
        7,0,0,0,0,0,9
  8,1,1,1
        8,0,0,0,0,0,10
        8,0,0,0,0,0,0
        8,0,0,0,0,0,11
  9,1,1,1
        9,0,0,0,0,0,12c
        9,0,0,0,0,0,0
        9,0,0,0,0,0,13
:
: Parts waiting outside.
0
```

Number of load/unload stations.

Load/unload station number.
Status index(si) of the load/unload station.
- Empty load/unload stations : si=1
- Load/unload station with an empty pallet : si=2
- Load/unload station with a busy pallet
  - waiting for stations : si=3
  - returned from stations : si=4

Number of stations.

Station number
Status index(si) of the in-shuttle
- Empty in-shuttle or in-shuttle with an empty pallet : si=1
- In-shuttle with a busy pallet : si=2

Status index of the station
- Idle station : si=1
- Busy station : si=2

Status index of the off-shuttle
- (same as in the case of in-shuttle)

Pallet records:
- station number
- associated order number
- part type
- urrent operation number
- priority index (not used)
- fixture number
- pallet number

Number of parts waiting outside

Note: When there are parts add records similar to pallet records.

Figure 3.6 BGIN.DAT Input Data File

3.3.4 BGIN.DAT Input File

Descritption : This file consists of data related to the initial condition of the system.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.6.

User Options :

a. Altering the number of stations.

Up to 15 workstations can be added to the system. Add data records containing the information related to the station and increase the number of stations in the system.

Ex: A new station no. 10 has to be added to the system. At the start of the simulation no pallets reside on shuttle positions of this workstation.

The current initial condition file;

```
:
: Machining centres.
6
    3,1,1,1
        3,0,0,0,0,0,3
        3,0,0,0,0,0,0
        3,0,0,0,0,0,4
        ..  ..  ..
    9,1,1,1
        9,0,0,0,0,0,12
        9,0,0,0,0,0,0
        9,0,0,0,0,0,13
:
:Parts waiting outside.
0
```

The new initial condition file;

```
:
: Machining centres.
7
    3,1,1,1
        3,0,0,0,0,0,3
        3,0,0,0,0,0,0
        3,0,0,0,0,0,4
        .. .. ..
    9,1,1,1
        9,0,0,0,0,0,12
        9,0,0,0,0,0,0
        9,0,0,0,0,0,13
    10,1,1,1
        10,0,0,0,0,0,0
        10,0,0,0,0,0,0
        10,0,0,0,0,0,0
:
:Parts waiting outside.
0
```

If it is required to delete a workstation, simply remove the data records corresponding to that station and update the number of workstations in the system.

b. Altering the number of pallets.

The number of pallets in the system can be changed by adding new pallets or deleting existing pallets. When new pallets are added to the system, specify its current location and the pallets can be taken out by changing the pallet number to zero.

c. Adding new parts waiting for the system.

Any number of parts which are waiting for the system at the beginning of the simulation can be appended to the file.

```
17                                  <-- Number of queues
1,1,1                               <-- Queue number
2,10,2                                  Ranking attribute
3,0,3                                   Ranking method
4,10,2
5,0,3
6,0,3
7,0,3
8,0,3
9,2,2
10,2,2
11,0,3
12,0,3
13,0,3
14,0,3
15,0,3
16,0,3
17,0,3
:
: Maximum no. of entries and the size of nset()/qset() array.
800,10000
:
: Output data file number for error messages.
2
```

Figure 3.7 GASP.DAT Input Data File

3.3.4 GASP.DAT Input File

Description : This file consists of queue discipline data.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.7.

User Options : None

```
;-------------------------------
: Current week no., total no. of weeks and
: system up time (mins./week)
1,10,3600

  :
;-------------------------------
: Decision rule selection.
1,1,1,1,1,1,1,1,1,1

  :
;-------------------------------
: Priority order for crane based activities.
1,2,3,4,5,6,7,8,9,10
:
;-------------------------------
: Priority order for AGV based activities.
1,2,3,4,5,6,7,8,9,10
:
;-------------------------------
: Ouput data options.

  :
: Read input files (unit=1).
1
: Write batch flow data (unit=2).
0
: Write weekly reports (unit=3)
0
: Write throughput time data (unit=4)
0
: Not used (unit=5)
0
:  Not used (unit=6)
0
: Write data for the tool post-processor (unit=7)
0
: Write data for the graphical post-processor (unit=8)
1
: Write entity tracing data (unit=9)
0
: Write queue data (unit=10)
0
:
;-------------------------------
: Output report selection (wrpt.out file)
:
: Production output
0
: Workstations utilization and AGV/crane utilization.
0
: Pallet utlization
0
: Fixture Utilization
0
: Activity counts
0
;-------------------------------
: Entity tracing start/end time
0.0,100.0
:
;-------------------------------
: Queue tracing start/end times and queue number.
: (Queue number = 0 -- trace all queues)
0.0,0.0,0
:
;-------------------------------
: Fixturing (or de-fixturing) time, average travel time and
: average pallet interchange time at pallet stands (all in mins.).
20.0,4.0,1.0
```

Figure 3.8 CTRL.DAT Input Data File

3.3.5 CTRL.DAT Input File

Description : This file contains the values of program control switches.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.8

User Options :

a. Changing the system operational time.

The system operational time defines the total number of minutes for which the system is available for production per week. A default value of 3600 mins/week is used. This value can be changed by the user.

b. Selecting decision rules.

At present the user cannot select any alternative decision rules. This facility has been incoporated for future enhancements.

c. Changing the activity priority for the crane.

The default activity order is as follows;

Activity No	Activity Description.
1	Fixture new parts.
2	Fixture partially finished parts.
3	Unload parts.
4	Reset parts (at load/unload stations).
5	Load Parts.
6	Remove fixtures.
7	Unload blocked parts.
8	Not used.
9	Not used.
10	Not used.

The default activity scanning order is;

1,2,3,4,5,6,7,8,9,10

For example, if the user wants to set a higher priority for loading over unloading, the new sequence is;

1,2,5,4,3,6,7,8,9,10

d. Changing activity priority for the AGV.

The default activity scanning order is as follows;

Activity No.	Description
1	Move a busy pallet from a station to a load/unload station.
2	Move a busy pallet from a station to another station.
3	Move a busy pallet from a load/unload station to a station.
4	Move blocked busy pallets.
5	Move an empty pallet away from a load/unload station.
6	Move an empty pallet to a load/unlaod station.
7	Move an empty pallet away from an in- shuttle.
8	Move an empty pallet away from an off-shuttle
9	Not used.
10	Not used.

As explained in the section c above, the priority order can be changed.

e. Output data options.

This switches set the type of output data generated in a simulation experiment. For example, if the user wishes to use the graphical post-processor, the corresponding switch must be set to 1.

f. Output report options.

If the weekly report option is on (see the section e above), the simulator will generate reports containing the system performance data such as machine utilization etc. The user can control the type reports by setting the appropriate switches to 1.

g. Queue content tracing facility.

The user is advised not to use this facility since it generates a massive amount of information. This option was used during the program debugging

3.4 SYSTEM DATA FILES

The simulation software is provided with a sample data set which was extracted from a real FMS. Part route files, tooling required files and a simplified version of the tool database reside on the C:FMS directory. The user is allowed to alter any of these files to create new experimental conditions. In the following these options are explained in detail.

1	1	1		20.00
1	1	2		20.00
1	2	4	0102.TLR	81.19
1	2	5	0102.TLR	81.19
1	3	8	0103.TLR	48.56
1	4	1		20.00
1	4	2		20.00
2	5	1		20.00
2	5	2		20.00
2	6	4	0106.TLR	343.46
2	6	5	0106.TLR	343.46
..	..	..	.. ..	
..	..	..	.. .. ..	
4	13	1		20.00
4	13	2		20.00
4	14	8	0114.TLR	148.35
4	15	9	0115.TLR	69.23
4	16	7	0116.TLR	76.40
4	16	8	0116.TLR	76.40
4	17	3		106.60
4	18	1		20.00
4	18	2		20.00

Fixture stage number.
Operation sequence number.
Station number.
Tool requirement file name.
Operation time.

Figure 3.9 A Part Route File (01.PRF in the sample data set)

3.4.1 PART ROUTE FILES

File Name Format : nn.PRF

nn = part type ID (01,02,.... 20)

Description : These files contain part route data.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.9.

User Options :

a. Changing station number and/or processing time for a given operation.

Change the current values to new values.

Ex: For the part type 1, operation no. 3 is carried out at station 8. It is required to re-assign this operation to station no. 7 with a new processing time of 78.45.

The current part route;

1	1	1		20.00	
1	1	2		20.00	
1	2	4	0102.TLR	81.19	
1	2	5	0102.TLR	81.19	
1	3	8	0103.TLR	48.56	<-----
1	4	1		20.00	
1	4	2		20.00	
..		..	..	..	

The new part route;

1	1	1		20.00
1	1	2		20.00
1	2	4	0102.TLR	81.19
1	2	5	0102.TLR	81.19
1	**3**	**7**	**0103.TLR**	**78.45**
1	4	1		20.00
1	42			20.00

b. Changing the number of alternative stations.

Insert the new operation record.

Ex: At present the operation no. 3 (part type 1) is carried out only at the station 8. However it has been found that this operation can be performed at station no. 7 as well.

The current part route;

1	1	1		20.00	
1	1	2		20.00	
1	2	4	0102.TLR	81.19	
1	2	5	0102.TLR	81.19	
1	3	8	0103.TLR	48.56	<------
1	4	1		20.00	
1	4			220.00	
..	..		.. ..	..	..

The new part route;

1	1	1		20.00
1	1	2		20.00
1	2	4	0102.TLR	81.19
1	2	5	0102.TLR	81.19
1	3	8	0103.TLR	48.56
1	**3**	**7**	**0103.TLR**	**48.56**
1	4	1		20.00
1	4	2		20.00
..	.	.	..	.. .. ..

In the above example it is assumed that the station 7 would use the same of set of tools (hence identical tooling required file names). If the user wishes to assign new a set of tools, a new tooling required files must be created and its name must be included in the part route file.

c. Changing the process sequence.

Swap the required operation records and re-arrange the process sequence numbers.

Ex: At present fixture stage 1 for part type 1 involves two operations which are carried out at station 4 or 5 and then at station 8. It is required to reverse this sequence.

The current part route data;

1	1	1		20.00	
1	1	2		20.00	
1	2	4	0102.TLR	81.19	<----
1	2	5	0102.TLR	81.19	<----
1	3	8	0103.TLR	48.56	<----
1	4	1		20.00	
1	4	2		20.00	

The new part route data;

1	1	1		20.00
1	1	2		20.00
1	**2**	**8**	**0103.TLR**	**48.56**
1	**3**	**4**	**0102.TLR**	**81.19**
1	**3**	**5**	**0102.TLR**	**81.19**
1	4	1		20.00
1	4	2		20.00

d. Adding new operations.

Up to 25 operations can be defined for a part type. Follow the instructions given in section b above.

e. Adding new parts to the system

Up to 30 different part types can be defined. Use the format shown in Fig. 3.9 for any part route files. If the user wishes to use the tool post- processor to investigate effect these new parts on tooling aspects, the tooling required files must be created for each new operation.

f. Deleting existing part types

Simply delete the corresponding part route file from the C:FMS directory.

2	14.6711
3	20.0611
4	7.9211
5	16.3011
6	0.4011
7	0.4511
8	0.0911
9	0.0311
10	1.3411
11	0.0011
12	0.9711
13	0.3711
14	0.3711

Tool identification code.
Tool wear increment.

Figure 3.10 A Tool Requirement File (0102.TLR in the sample data set)

3.4.2 TOOLING REQUIRED FILES

File name format : nnmm.tlr

nn = part type ID (01,02,.... 20)

mm= operation ID (01,02,.... 25)

Description : These files contain tools required for individual cutting operations.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 3.10.

User Options :

Changing data in these file are quite simple. The user can change tool identification code and/or tool wear increment. If new tools are added make sure that the new tool has been included in the tool database file.

```
1       1       99.990
2       3       95.000
3       1       100.000
4       1       50.000
5       1       60.000
6       1       15.000
7       2       15.000
8       1       50.000
9       1       20.000
10      1       5.000
11      1       30.000
12      2       30.000
..      ..
```

Tool identification number.
Tool class number.
Tool life (mins).

Figure 3.11 TOOL.DBS Input Data File (simplified version of the tool data base)

3.4.3 TOOL DATABASE FILE

File Name Format : TOOL.DBS

Description : This file contains information about all tool types used in the system.

File Type : ASCII

File Access : Sequential

Data Organization : As explained in Fig. 3.11.

User Options :

The user is allowed to change the attributes of individual tools. Simply replace the current value with the new value.

3.5 OUTPUT FILES

3.5.1 INTRODUCTION

When the weekly output report option is on, the user can select a number of different output reports containing the system performance. This modular output report selection has been provided so that only relevant output reports are generated for a specified experimental environment.

3.5.2 OUTPUT REPORT OPTIONS (Weekly Performance Report)

All output reports are written into WRPT.OUT file which can be listed or printed using DOS commands. The following options are available to the user;

Output Report Option No.	Description
1	Production output data.
2	Stations, the crane and the AGV utilization.
3	Pallet utlization.
4	Fixture utilization.
5	Activity counts.

In next few sections the content of each report module are explained with the aid of sample output reports.

Output Report Name:: WRPT.OUT

Output Report Option: 1 (see Fig. 3.8)

Description: Weekly production output reports.

**** Week 3**** **** Production Output Rates ****

Order Code	Part Type	Parts Required	Parts Strated		Parts Completed	
			(c)	(d)	(c)	(d)
1	1	10	10	0	7	3
2	2	8	8	0	6	3
3	4	10	10	3	6	3
4	11	8	8	0	4	1
Total		36	36	3	23	10

(c) Cumulative value
(d) Discrete value (current week only)

Output Report Option 1 : Weekly Production Output

Output Report Name : WRPT.OUT

Output Report Option : 2 (see Fig. 3.8)

Description : Weekly utilization figures of stations, the crane and the AGV

** Week 3 ** ** Workstation Utilization **

Station no.	Station	In-shuttle	Out-shuttle
1	0.2981	0.0000	0.0000
2	0.3375	0.0000	0.0000
3	0.8114	0.3472	0.0233
4	0.7964	0.3242	0.0153
5	0.6769	0.2347	0.0156
6	0.0000	0.0000	0.0000
7	0.3253	0.0806	0.0478
8	0.5564	0.1386	0.0278
9	0.2942	0.0300	0.0142

** AGV utilization : 0.3000
** Crane utilization : 0.7806

Output Report Option 2 : Weekly Utilization Figures of Stations, the cRane and the AGV

Output Report Name : WRPT.OUT

Output Report Option : 3 (see Fig. 3.8)

Description : Weekly utilization figures of pallets.

**** Week 3 ** ** Pallet Utilization ****

pallet no	1	2	3	4	5
utilizatn	0.3608	0.0000	0.6361	0.2219	0.2969
pallet no	6	7	8	91	0
utilizatn	0.8297	0.2803	0.9667	1.0000	0.4294
pallet no	11	12	13		
utilizatn	0.0000	0.4000	0.1589		

Output Report Option 3 : Weekly Utilization Figures of Pallets

Output Report Name : WRPT.OUT

Output Report Option : 4 (see Fig. 3.8)

Description : Weekly utilization figures of fixtures.

**** Week 3 **** Fixture Utilization ****

Fixt. no	1	2	3	4	5
utilizatn	0.0000	0.0000	0.0000	0.0000	0.0000
Fixt. no	6	7	8	9	10
utilizatn	0.0000	0.2089	1.0000	0.7503	1.0000
Fixt. no	11	12	13	14	15
utilizatn	1.0000	0.0000	1.0000	0.4675	0.0000
Fixt. no	16	17	18	19	20
utilizatn	0.0000	0.0000	0.0000	0.0000	0.0000
Fixt. no	21	22	23	24	25
utilizatn	0.0000	0.0000	0.0000	0.0000	0.0000
Fixt. no	26	27	28	29	30
utilizatn	0.0000	0.0000	0.0000	0.0000	0.0000
Fixt. no	31	32	33	34	
utilizatn	0.1061	1.0000	0.7378	0.9772	

Output Report Option 4 : Weekly Utilization Figures of Fixtures

Output Report Name : WRPT.OUT

Output Report Option : 5 (see Fig. 3.8)

Description : Activity counts.

**** Week 3 ** ** Activity Counts ****

**** Crane based activities ****

Fixture new parts	: 3
Fixture partially finished parts	: 26
Load parts	: 30
Reset parts	: 16
Unload Parts	: 29
Remove fixtures	: 31
Unload blocked parts	: 0
**** Total crane assignments ****	: 135

**** AGV based activities ****

**** Busy pallets ****

Load/unload stn. to workstation	: 45
Workstation to workstation	: 48
Workstation to load/unload station	: 45
Blocked pallets	: 0
**** Total ****	: 138

**** Empty pallets ****

From in-shuttle	: 31
From out-shuttle	: 95
To load/unload stn.	: 0
**** Total ****	: 132
**** Total AGV assignments ****	: 270

**** Station based activities ****

**** Process parts ****

Station-3	: 20
Station-4	: 12
Station-5	: 11
Station-6	: 0
Station-7	: 15
Station-8	: 22
Station-9	: 13
**** Total ****	: 93

Output Report Option 5 : Activity Counts

Output Report Name: FLOW.OUT

Description : Throughput times of individual parts.

[a]	[b]	[c]	[d]	[e]	[f]	[g]	[h]	[i]	[j]	[k]	[l]	[m]
2	2	2	80	295	198	1045	280	320	160	0	0	* 2378
1	1	1	80	195	80	613	670	150	490	489	40	* 2807
..	..	..	..	..	..	..	..	..	..	..	..	..

[a] Part identification number
[b] Order identification number
[c] Part type
[d] Waiting time [mins.) before first loading
[e] Flow time (mins.) on the first fixture
[f] Waiting time (mins) before second loading
[g] Flow time (mins.) on the second fixture

.. ..
[m] Total flow time (mins)

Output Report FLOW.OUT (Throughput times of Individual parts)

Output Report Name : BTCH.OUT

Description : Throughput times of individual orders.

[a]	[b]	[c]	[d]	[e]	[f]	[g]	[h]
2	2	1	4	4	20	12615	2595
1	1	1	4	4	0	12655	12655
3	4	1	4	5	40	13813	13773

[a] Order identification number.
[b] Part type.
[c] Order start week no.
[d] Order end week no.
[e] Order due week no.

[f] Entering time for the first part.
[g] Leaving time for the last part.
[h] Total flow time of the order.

Output Report BTCH.OUT (Throughput times of individual orders)

3.6 PROGRAM ROUTINE DESCRIPTION

The FMS simulator consists a number of program routines. The overall organization of major functional modules are shown in Fig. 3.12. In the following, the use of each program routine is briefly described.

R00 Main Program.

R0001 **program main**

This routines calls all data reading subroutines first. Having set up the required data, the simulation is initiated by calling executive routines of the crane and the AGV. Then the process of continuing the simulation is handed over to the GASP executive routine.

R01 Data Initialization

R0101 **subroutine i_gasp**

Reads and stores values related to GASP variables.

R0102 **subroutine i_rout**

Reads and stores part route data.

R0103 **subroutine i_fxpt**

Reads and stores fixture assignment data.

R0104 **subroutine i_sysm**

Reads the initial condition of the system.

R0105 **subroutine i_fram**

Reads and stores the values of various program control switches.

R02Crane Based Start Events.

R0201 **subroutine x_ascr**

This is the executive routine for the crane based activities. When the crane become available, these activities are scanned in a hierarchical order defined by the user.

If an activity can be started then the associated event start routine is called by the executive.

R0202 **subroutine s_fxnp**

When fixtures for the first fixture stage are available, a part is selected from production orders (U_SLOR routine) for fixturing.

R0203 **subroutine s_fxop**

When required fixture is free, a part is selected from partially finished parts which are waiting for fixtures.

R0204 **subroutine s_ldpt**

When a free pallet become avialable and the target station is ready to accept the part, it will be loaded.

R0205 **subroutine s_rupt**

This routine is used to reset or unload parts.

R0206 **subroutine s_rmfx**

This routines initiates the de-fixturing of parts which returned from the system.

R0207 **subroutine s_unbk**

When the parts are involved in a blockage at a load/unload station, this routine is used to unload the part temporarily in order to relieve the deadlock in material flow.

R03 AGV Based Start Events.

R0301 **subroutine x_asag**

When the AGV is free, the executive routine attempts to assign a task to the AGV, by scanning AGV based activities in a hierarchical order defined by the user.

R0302 **subroutine s_mbwl**

This routine moves a busy pallet from an off-shuttle of a workstation to a load/unload station.

R0303 **subroutine s_mbww**

A busy pallet is moved away from an out-shuttle position to an in-shuttle of the next station.

R0304 **subroutine s_mblw**

This routine moves a busy pallet from a load/unload station to a workstation.

R0305 **subroutine s_mbbk**

When busy pallets are involved in a blockage between two workstations, this routine picks up a busy pallet from a blocked pallets and moves to a free shuttle position to relieve the blockage.

R0306 **subroutine s_mefl**

If all load/unload stations are occupied by empty pallets, then a processed part cannot be delivered to load/unload stations. When this condition is detected, an empty pallet is moved away from the a load/unload station to provide space for in-coming busy pallets.

R0307 **subroutine s_metl**

Although fixtured parts available for loading and the target workstation is ready to accept the part, an empty pallet may not be available at load/unload stations. When this occurs, if available, an empty pallet is moved from a shuttle position at a workstation to a load/unload station.

R0308 **subroutine s_mefi**

It may possible that the in-shuttle of the target station is occupied by an empty pallet and therefore a busy pallet cannot be delivered to the workstation. If this condition exists the empty pallet concerned is moved away from the in-shuttle.

R0309 **subroutine s_mefo**

When the machining process is over, it is required to move the processed part to the off-shuttle, but the off-shuttle may have been blocked by an empty pallet. If this condition prevails the empty pallet is moved away from the off- shuttle.

R04 Workstation Based Start Events

R0401 **subroutine s_pros**

This routine initiates processing a part at a station.

R0402 **subroutine s_mbto**

Having processed the part, if the off-shuttle of the workstation is free, the pallet is moved to the off- shuttle.

R05 End Event Modules.

These modules update the system state at the end of each activity.

R0501 **subroutine e_evnt**

This routine calls the appropriate end event subroutine.

R0502 **subroutine e_pros**

End event of the machinig activity.

R0503 **subroutine e_mbto**

End event of moving a busy pallet to the off-shuttle.

R0504 **subroutine e_mbpl**

End event of moving busy pallets.

R0505 **subroutine e_uscr**

End event of using the crane (except for unloading parts)

R0506 **subroutine e_unld**

End event of unloading a part.

R0507 **subroutine e_mepl**

End event of moving empty pallets.

R0508 **subroutine e_rmfx**

End event of removing fixtures.

R0509 **subroutine u_rept**

Write weekly output reports.

R06 Entity handling modules.

R0601 **subroutine u_pute**

This routine adds an entity to a queue.

R0602 **subroutine u_qeni**

This routine is used, when the initial condition of the system is read.

R0603 **subroutine u_exat**

The attributes of an entity is copied into a set of global variables.

R0604 **subroutine u_schd**

This routine schedules an end event to occur after certain time period.

R07 Shuttle/pallets handling module.

R0701 **subroutine u_szsh**

This routine is used to seize the target shuttle for an AGV movement. This guarantees that the required shuttle is not be taken by any other activity, while the pallet is being moved.

R0702 **subroutine u_rlsh**

When a pallet is taken away from a shuttle position, the shuttle status is reset by this routine.

R0703 **subroutine u_rlpl**

When a part is unloaded or an empty pallet is moved, this routine reset the pallet status.

R08 Station selection module.

R0801 **function u_fdst**

This function attempts find a workstation for a given operation. The alternative stations are sought in the following order.

a. Idle workstation.

b. Busy workstation (among the alternatives) with minimum remaining processing times.

R09 Fixture selection module.

R0901 **function u_fdfx**

This routine attempts to find a fixture for a given operation. If alternative fixtures are available, the first available fixture is selected.

R10 Shuttle selection modules.

R1001 **function u_fdes**

This routine attempts to find an empty shuttle position which satisfies a given selection requirements.

R1002 **subroutine u_fdsh**

This routine uses the above function (R1001) to find an empty shuttle position for a given pallet.

R1003 **subroutine u_fdep**

This routine attempts to find an empty pallet.

R1004 **subroutine u_fdph**

This uses the above routine repetitively to find an empty pallet in a preferred order.

R1005 **subroutine u_fdeb**

This attempts to find an off-shuttle position for a blocked pallet.

R11 Order/Part selection modules.

R1101 **subroutine u_slor**

This subroutine selects a production order from which a part can be taken for fixturing. The availability of the required fixutre is also tested.

R1102 **subroutine u_slpf**

This routine selects a part from a pool of partially finished parts for fixturing.

R1103 **function u_fdpl**

This routine selects a fixtured part for loading. The availability of space at the target workstation is tested.

R12 Pallet Selection Modules

R1201 **function u_fdru**

This routine attempts to identify the next operation of a busy pallet returned from the system.

R1202 **function u_fdub**

This function tests whether busy pallets are involved in a blockage at load/unload stations.

R1203 **subroutine u_fdbm**

This routine sets the target workstation for a busy pallet waiting on an off-shuttle.

R1204 **function u_fdbs**

When busy pallets are involved in a blockage at workstations, this routine selects a pallet to take out from the circular waiting.

R1205 **function u_fdib**

This routine attempts to identify whether a target in- shuutle is blocked by an empty pallet.

R1206 **function u_fdpm**

This routine selects an empty pallet which has to be moved away from the current location.

R1207 **function u_fdel**

This function attempts to identify whether an empty pallet is required at load/unload station.

R13 Modified GASP Modules.

R1301 **subroutine x_gasp**

This the executive GASP routine which continues the simulation until the simulation end time is reached.

R1302 **subroutine u_queu**

The entity defined by the current attribute values is added to a given queue.

R1303 **subroutine u_rmov**

This removes an entity from a given queue.

R1304 **subroutine u_find**

This routine tests whether an entry with a given station exists in given queue.

R1305 **subroutine u_eror**

This routine reports GASP routine errors.

R1306 **subroutine u_pont**

This routine initializes major GASP arrays.

R14 Reporting Modules.

R1401 **subroutine u_rep1**

This reports the weekly production output reports.

R1402 **subroutine u_rep2**

This reports the utilization of workstations, the crane and the AGV.

R1403 **subroutine u_rep3**

This routine reports the utilization of pallets.

R1404 **subroutine u_rep4**

This reports the utilization of fixtures.

R1405 **subroutine u_rep5**

This outputs the activity counts.

R1406 **subroutine u_wrfl**

This routine writes the flow time components of a given part.

R15 Debugging Modules

R1501 **subroutine u_trac**

This is used to trace movements of entities.

R1502 **subroutine u_chkq**

This routine outputs the entries of the each queue.

R16 Utility Modules.

R1601 **subroutine u_adpt**

This routine adds a new part to the current part list.

R1602 **subroutine u_upfl**

This routine updates flow time components of a given part.

R1603 **subroutine u_gtpt**

This routine retrieves the processing time for a defined operation.

R1604 **subroutine u_zatb**

This routine sets all attributes of a given entity to zero.

R1605 **subroutine u_rank**

This routine ranks entries in an array.

R1606 **subroutine u_opfl**

This routine opens required output files.

R1607 **subroutine u_grps**

This routine writes data required for the graphical post processor.

CHAPTER 4

GRAPHICAL POST-PROCESSOR

4.1 INTRODUCTION

The graphical post-processor is a software module which can animate movements of entities in a simulated system. As a post- processor, it interprets data output by the base simulator to display the dynamic nature of the system being modelled. This microcomputer version of the post-processor has been constructed using Graphical Kernel System.

In this chapter, its design features, data file structures, operational aspects and hardware/software requirements are explained. Practical applications of the post-processor can be found in vol. 1 chapter 6.

4.2 GRAPHICAL KERNEL SYSTEM

Most of high-level general programming languages such as FORTRAN, do not include special language syntax for graphical applications. However there are several graphical software systems which can be linked to high level languages. Among them, Graphical Kernel System (GKS) is a generic software system which has become one of the international standards in computer graphics [Hopgood et el., 1983]. This system allows programs to support a wide variety of graphics devices and it is defined independantly of the programming language.

Before it can be used from a particular language, a *language binding* must be defined for that language [Hopgood et el., 1983]. In this graphical post-processor, IBM Personal Computer GKS is used [IBM(a),IBM(b),1984]. It consists of a linkable subroutine library supported by bindings to several programming languages. To use the Kernel System, the programmer include calls to GKS subroutines in the source code. The calling conventions and parameter sequences are described in the binding for the particular programming language.

4.3 DESIGN CONSIDERATIONS

The main objective of this exercise is to animate the dynamic nature of the system. However moving entities alone cannot depict the behaviour of the system. For example, in a manufacturing system, stationary items such as workstations must be displayed in different ways to represent its different states (such as workstation idle, workstation down etc.). In addition more information such as waiting times, queue sizes can be appended to the display to provide more comprehensive picture of the system. It was thought this graphical post-processor would include most of these features.

The graphical post-processor is driven by a data file which contains the history of the system behaviour. A text file containing required information can be easily generated by adding a few program statements to the base simulation system. The animation software processes this data and take appropriate actions (such as moving entities, changing colours etc.) to display the system state.

In order to simplify the software design process, system events can be grouped based on similar animation requirements. Then program routines can be constructed for each group to visualize these graphical changes. Each event group can be given a code number and within the computer program these code numbers can be used to fire the associated program routines. The following event group codes are used in this system.

Event Code	1	Attributes 2	3	4
1	Load/unload station no.	Load/unload colour at the end event.	-	-
2	Current station no.	Current shuttle type	Target station no.	Target shuttle type
3	Station no	Processing time.	-	-
4	Station no	-	-	-
5	-	-	-	-
6	-	-	-	-
7	Station no.	-	-	-
8	Station no.	-	-	-

Figure 4.1 Events and Their Attribute Values

[a]	[b]	[c]	[d]	[e]	[f]	[g]
245	2	6	5	1	1	2
250	6	0	0	0	0	0
270	5	0	0	0	0	0
270	2	2	2	2	5	1
270	1	6	1	2	0	0
275	6	0	0	0	0	0
275	3	2	5	201	0	0

[a] Simulation clock value
[b] Event code ID
[c] Pallet number
[d] Attribute 1
[e] Atrrbiute 2
[f] Attribute 3
[g] Attribute 4

Figure 4.2 A Segment of the Trace File

Event Code	Event Description
1	Start events of crane based activities such as loading, unloading and re
2	Start events of AGV based activities.
3	Start of processing.
4	Start of moving out to off-shuttle.
5	End events of crane based activities.
6	End events of AGV based activities.
7	End of processing.
8	End of moving to out-shuttle.

Each of these events groups requires certain amount of information to animate the system behaviour. Obviously the amount of information depends on the complexity of the display. The event time and the pallet number are essential for each event, in addition further information must be appended to each event record. A maximum of further four event attributes are defined in this system. Relationships between event codes and values of these four event attributes are shown in Figure 4.1.

A few additional program statements can be added to the base simulator to output this information. They are stored in the trace file and subsequently read by the post-processor. The data structure and a segment of the trace file is shown in Figure 4.2.

4.4 INPUT DATA REQUIREMENTS

A typical animation display will consists of a large number of elements such as workstations, shuttles etc. They also can be grouped according to their representative shapes. For example similar workstations can be represented by a specific element

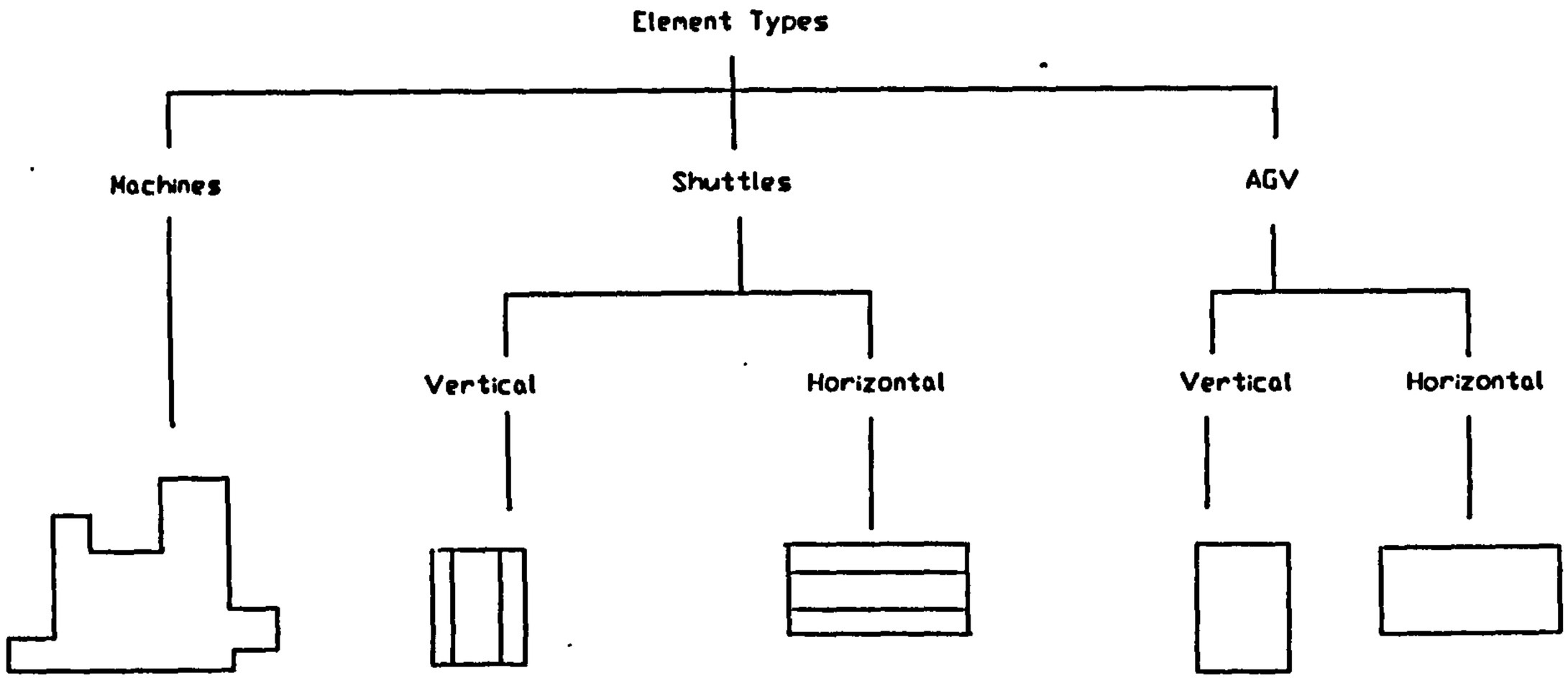


Figure 4.3 Basic Graphical Element Types

type. A number of different element types are defined in this graphical post-processor (Figure 4.3). For each element in the display, several attributes can be defined, such as location coordinates and element type etc.. For fixed elements (such as workstations) dynamic attributes (colour) and static attributes (location, element type) are stored in separate areas. Dynamic attributes are constantly updated as events are fired. For moving elements (such as AGV), most of attributes are dynamic (location, colour, element type), therefore a separate data storage area is defined for each moving element.

For stationary elements the following attributes are defined;

AttributeNo.	**Description**
1	element type ID.
2	X coordinate of element.
3	Y coordinate of element.
4	X coordinate of AGV stop point.
5	Y coordinate of AGV stop point.
6	AGV orientation (indicated byassociated element type).

Further information may be required if other features such as queue sizes and simulation clocks are added to the display.

4.5 DATA FILE STRUCTURES

In this section all input data file structures (except the trace file which is discussed above) are explained. The graphical post-processor accesses the BGIN.DAT file of the part flow simulator to obtain the initial condition.

```
2
1,1
        3,8,60,5,62,4
2,1
        3,8,54,5,56,4
6
3,3
        2,10,23,12,20,5
        1,10,30,0,0,0
        2,16,23,18,20,5
4,3
        2,25,23,27,20,5
        1,25,30,0,0,0
        2,31,23,33,20,5
5,3
        2,40,23,42,20,5
        1,40,30,0,0,0
        2,46,23,48,20,5
7,3
        2,55,23,57,20,5
        1,55,30,0,0,0
        2,61,23,63,20,5
8,3
        2,70,23,72,20,5
        1,70,30,0,0,0
        2,76,23,78,20,5
9,3
        2,85,23,87,20,5
        1,85,30,0,0,0
        2,91,23,93,20,5
```

No. of load/unload stations.

Load/unload station ID
No. of display elements

No. of workstations

Workstation no.
No. of display elements.

Element ID
X- coordinate of the element
Y- coordinate of the element
X- coordinate of the AGV stop
Y- coordinate of the AGV stop
AGV orientation (indicated by the element ID)

Figure 4.4 Content of LOCT.DAT File

4.5.1 LOCT.DAT Input File

Description :This file consists of values of attributes defined for stationary elements in the display.

File type : **ASCII**

File Access : Sequential

Data Organization : As shown in Fig. 4.4.

User Options : None

3,16,45
4,31,45
5,46,45
7,61,45
8,76,45
9,91,45

Station number
X-coordinate of the clock
Y-coordinate of the clock

Figure 4.5 Content of CLOK.DAT File

4.5.2 CLOK.DAT Input Data File

Description : The data elements in this file sets the location of clocks.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 4.5.

User Options : None

Use :

In order to provide more comprehensive information about the system, an individual clock is displayed for each workstation. When the workstation is processing a part its clock displays the remaining cutting time or the waiting time if the part remains on the machine table (off- shuttle is blocked) after processing. .pa

4.6 PROGRAM ROUTINES

The graphical post-processor embodies a number of routines coded in FORTRAN 77. In the following, the function of each routine is briefly explained.(For description of GKS based routine please see IBM(a),IBM(b),1984)

R01. **program main**

The program routine initialize the video display unit for GKS output, displays various messages and execute the animation. It also provide facilities to set global animation speed and pallet transfer speeds.

R02. **subroutine scrn**

This routine is associated with start event of crane based activities. Three arguments, pallet number, station number and the end colour of the pallet are passed to the routine. It turns the colour of the relevant workstation to magenta.

R03. **subroutine sagv**

This initiates pallet transfers to AGV. If AGV is not available at the required position, first it moves the AGV to the requesting point. Then pallet is transfered to the AGV. If the user wishes, the pallet transfer speed can be varied.

R04. **subroutine spro**

The start of processing activity is indicated by this routine. The pallet is taken from the in-shuttle and placed on the machine table. It also activiate the workstation clock.

R05. **subroutine ecrn**

At the end of crane based activities, the pallet colour is changed by this routine.

R06. subroutine eagv

The AGV is shown at the target workstation by this routine. It also transfers the pallet to the correct shuttle position.

R07. **subroutine epro**

At the end of processing the current clock is stopped and the busy pallet colour (red) is used to display the workstation. It also activates clock which indicates the waiting time.

All the above routines are used to display movements of entities and change of system state. They are supported by the following utility routines.

U01. **subroutine rdlc**

This routine reads element location data and store them in relevant data array.

U02. **subroutine begn**

This reads the initial state of the system and initialize state of the each element in the display.

U03. **subroutine inds**

This module display the system state at the initial state.

U04. **subroutine dwag**

This draws the AGV at a given location.

U05. **subroutine erag**

This erases the AGV at a given location.

U06. **subroutine mvag**

This moves the AGV to a given location.

U07.**subroutine rdck**

This routine reads positions of workstation clocks and stores them in a relevant array.

U08. **subroutine upck**

This module updates all clocks. The increment value would depend on the state of the workstation. If the workstation is processing a part, the current clock value is reduced by 1 to indicate remaining processing time. On the other hand, when

part is waiting on the machine table, the clock value is increased by 1 to show the accumulated waiting time.

U09. **subroutine dsck**

This routine displays current clock values at all workstations.

U10. **subroutine dmck**

This displays the current value of the master simulation clock.

U11. **subroutine elou**

This draws the envelope of a given element type.

U12. **subroutine elfl**

This module fills a given element type with a defined colour.

U13. **subroutine reco**

This draws a rectangle envelope.

U14. **subroutine refl**

This fills a rectangular space with a specific colour.

U15. **subroutine line**

This routine draws a line between two points.

U16. **subroutine wntx**

This routine writes a given piece of text, starting from a specific point.

U17. **subroutine wctx**

This routine writes text centered at a given point.

U18. **subroutine wrin**

This routine writes an integer number at a given point.

U19. **subroutine opwn**

This routine opens a window at a given location.

U20. subroutine dely

This routine sets the display time interval (delay time).

U21. Element shape routines

Subroutines CMCF, CSVO, CSVF, CSHO, CSHF, CAVO, CAHO, and CAGF corresponds to element shapes.

CHAPTER 5

THE TOOL POST-PROCESSOR

5.1 INTRODUCTION

The tool post-processor is an independent module which uses data generated by the simulation system, in conjunction with system tool database to estimate certain parameters associated with the tool management system.

The tool post-processor is supported by a number of program units which prepares the data required for the tool post- processor. The user can alter a variety of input data conditions to create different experimental environments.

In this chapter all these program units together with their input/output files are explained in detail.

5.2 ORGANIZATION OF THE TOOL POST-PROCESSOR SOFTWARE SYSTEM.

The overall organization of the tool post-processor software system is shown in Fig. 5.1. It has the following major program units;

a. GNTR.EXE - Concatenates tool requirement files.

b. GNDF.EXE - Generate a number of direct access files.

c. GNTT.EXE - Generate tool types data.

d. GNTS.EXE - Generate lists of tools required at stations for a given product mix.

e. GNTC.EXE - Generates the tool crib configuration.

f. ASTM.EXE - Assign tools to tool magazines.

g. TPOS.EXE - Tool post-processor.

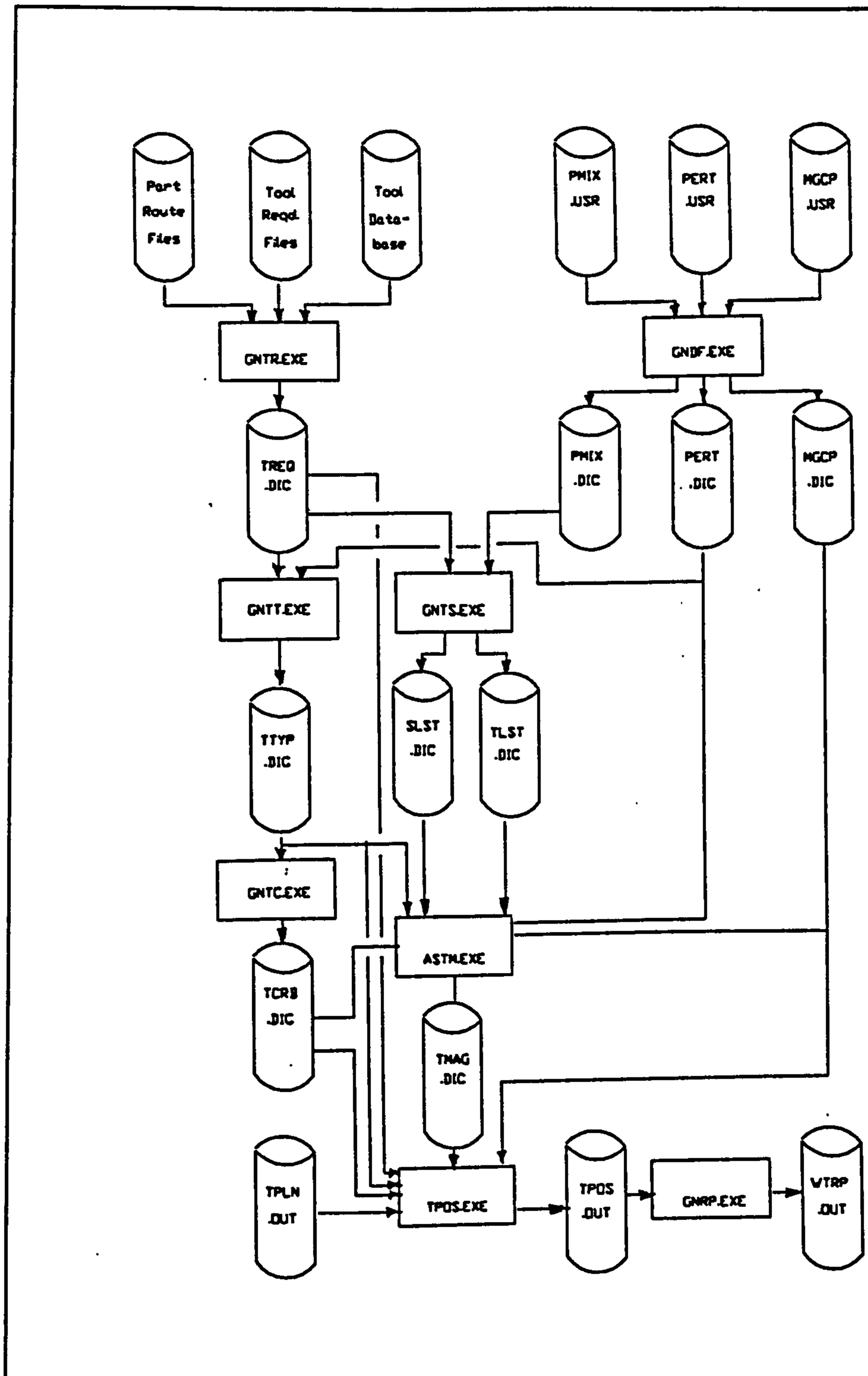


Figure 5.1 The Overall Organization of the Tool Post-processor

h. GNRP.EXE - Report generator.

In the following sections these program modules are explained in detail.

5.3 USER OPTIONS

User options available for the part flow simulator is described in section 3.4. The effects of those changes on tooling aspects can be ivestigated by executing the part flow simulator and the tool post-processor in a sequential manner. For example, the effects of changing the process sequence on tooling can be studied as follows;

a. Make necessary changes in the part route file concerned.

b. Run the part flow simulator.

c. Run the tool post-processor.

In addition to options described in section 3.4, the following alternatives are available within the tool post- processor environment;

a. Changing tool magazine capacity (section 5.5.1)

b. Changing the permanent tool list (section 5.5.2)

c. Changing the product mix used for tool configuration setup (setion 5.5.3).

d. Changing the re-grinding level of tools (section 5.6).

e. Changing the maximum number of class III tools allowed on the magazine. (see volume 1 chpater 8 for types of tool classes)

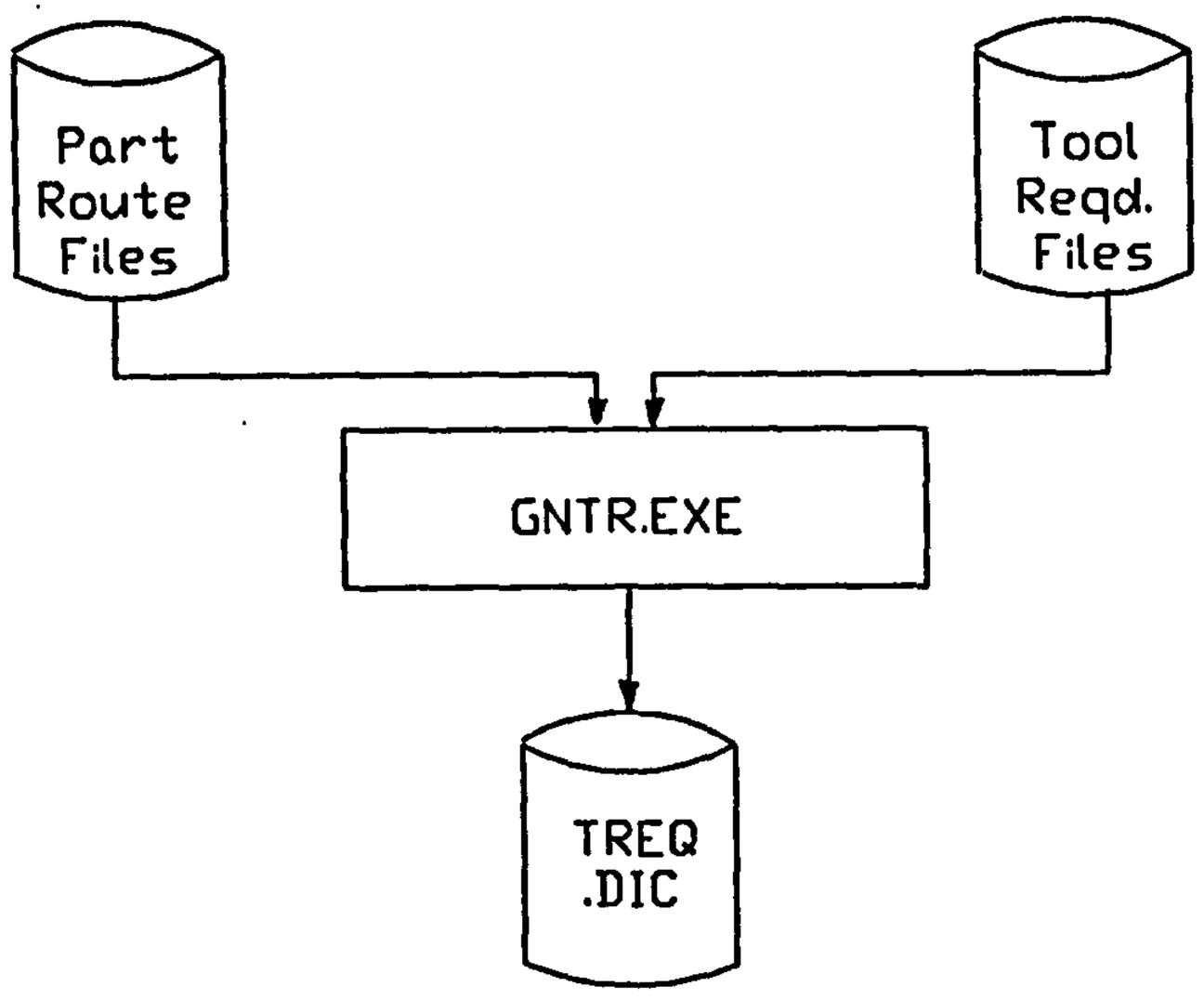


Figure 5.2 The Input/Output File Organization of GNTR.EXE Module

5.4 GNTR.EXE PROGRAM MODULE

Description : In a typical FMS a large number of tool requirements files may exist. For efficient data handling, it is necessary to concatenate all relevant data into a single file.

Input/Ouput files Organization : As shown in Fig. 5.2

Input files : All part route files and tool requirement files.

Output files : TREQ.DIC (section 5.4.1)

Logical process : The program reads all part route files and tool requirements file and stores data in a single direct access file which has three major sectors (see section 5.4.1 for more information).

1	13	30
2	31	47
..	..	..
..	..	..
11	186	200
12	201	215
13	0	0
14	216	229
15	230	247
..	..	..
212	1897	1918
213	1919	1932
214	0	0
215	0	0
216	2	1467
217	3	2006
218	4	792
219	5	1630
..	..	..
1931	4	0
1932	1	0

Part type record number.
Record number of first operation.
Record number of last operation.

Operation record number.
Record number of first required tool.
Record number of last required tool.

Tool record number.
Tool type ID.
Wear increment.

Figure 5.3 TREQ.DIC Data File

5.4.1 TREQ.DIC OUTPUT DATA FILE

Description : This file contains tool requirements data of all part types.

File Access : Direct

Data Organization : The tool requirement data are stored in this file using a pointer system as shown Fig. 5.3.

User Options : None. This file is automatically created by GNTR.EXE (section 5.4) program.

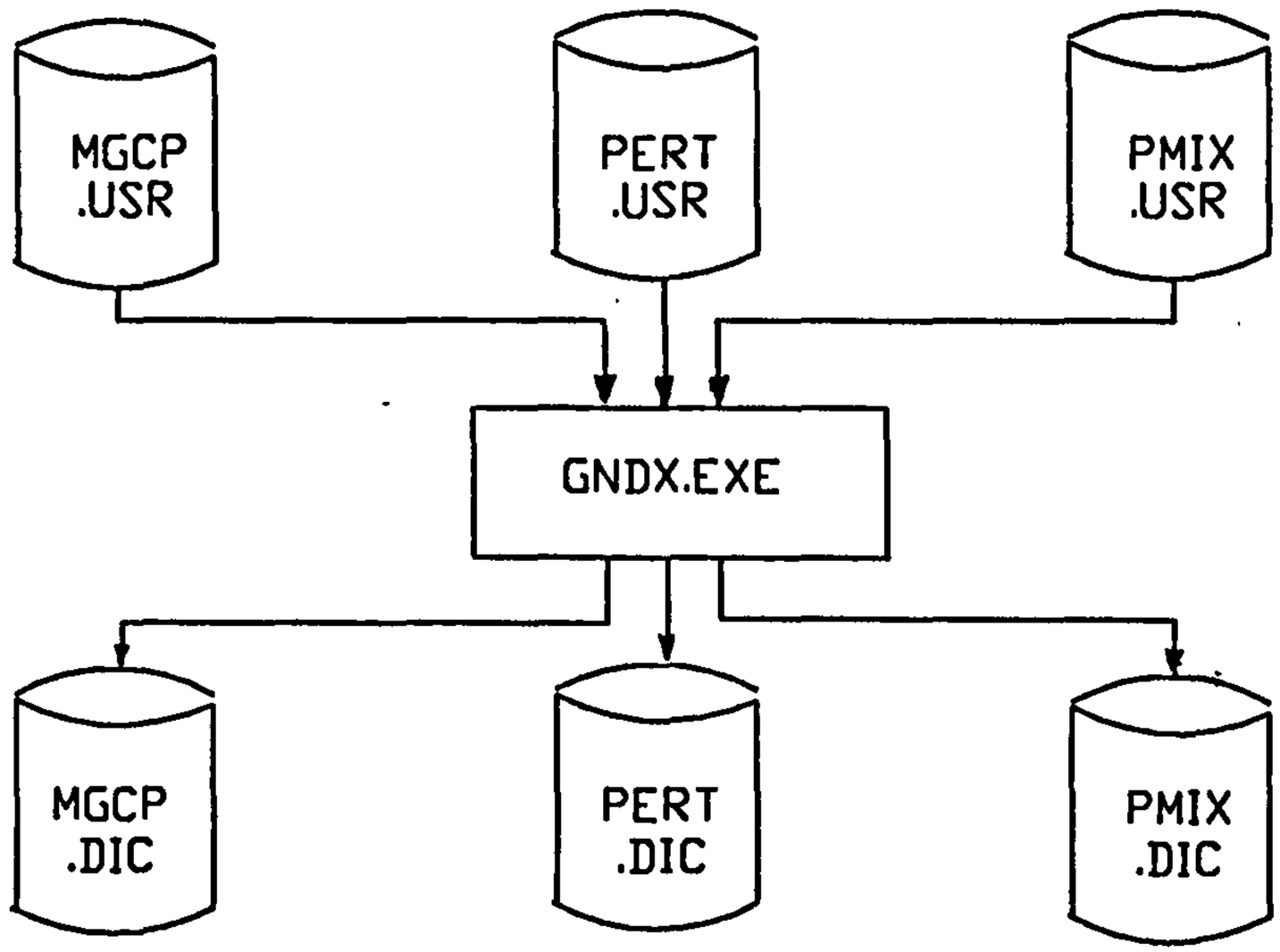


Figure 5.4 The Input/Output File Organization of GNDF.EXE Module

5.5 GNDF.BAT PROGRAM MODULE

Description : This program converts a number of sequential access files into direct access files.

Input/Output File Organization : As shown in Fig. 5.4.

Input Files : MGCP.USR (section 5.5.1)

PERT.USR (section 5.5.2)

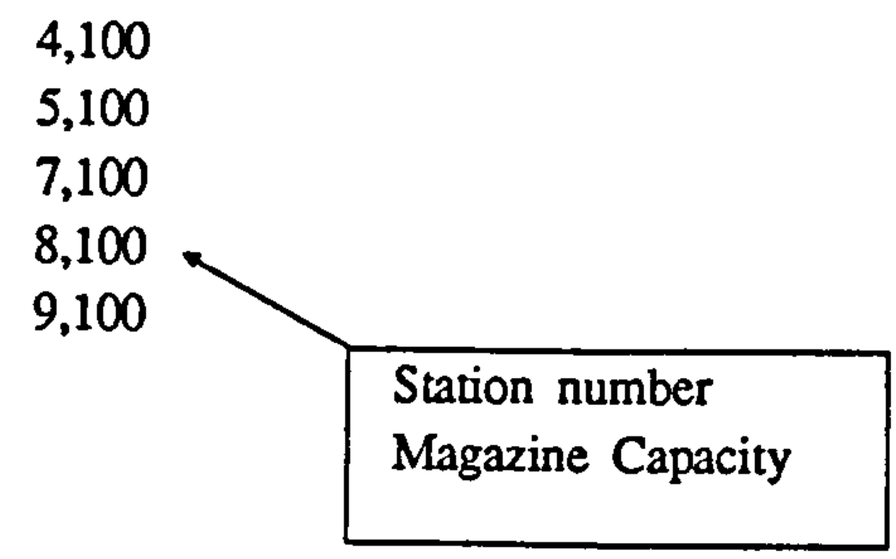
PMIX.USR (section 5.5.3)

Ouput Files : MGCP.DIC

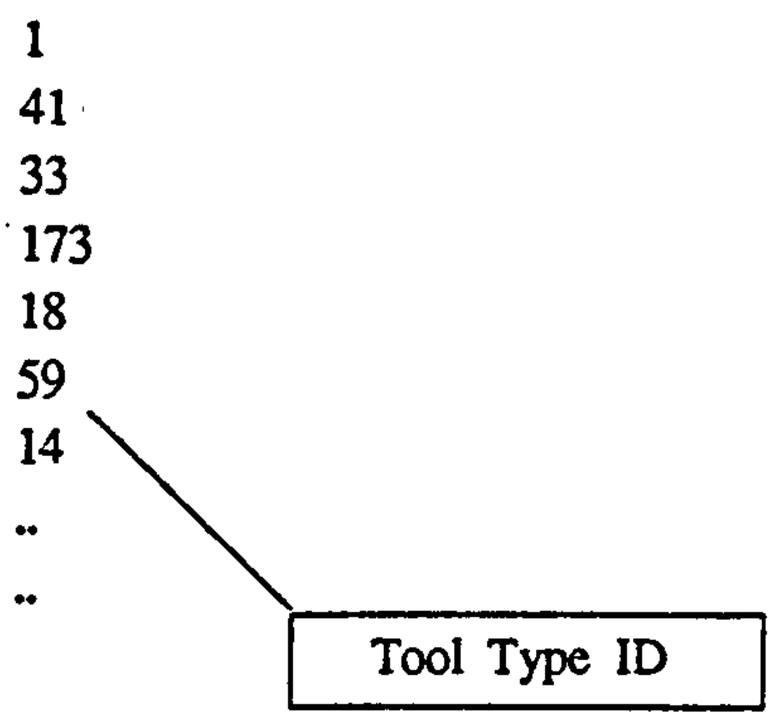
PERT.DIC

PMIX.DIC

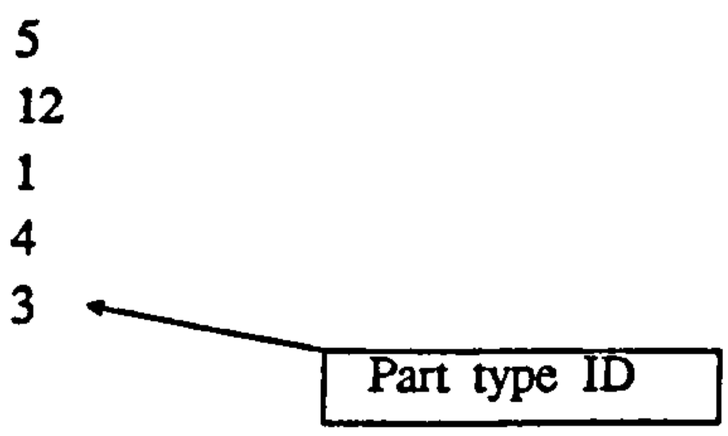
Logical Process : This program reads user defined files and converts them into direct access files.



MGCP.USR Data File



PERT.USR Data File



PMIX.USR Data File

Figure 5.5 MGCP.USR, PERT.USR and PMIX.USR Data Files

5.5.1 MGCP.USR DATA FILE

Description : This file contains the tool magazine capacities of workstation.

File Type : ASCII

Access : Sequential

Data Organization : As shown in Fig. 5.5.

User Options:

User can change the tool magazine capacity of any of the workstations up to 300 pockets. Replace the current value by the new value. When new workstations are added to the system, their tool magazine capacities must be included in this file.

5.5.2 PERT.USR DATA FILE

Description : This file contains a list of tools which are considered to be permanent tools.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 5.5.

User Options :

The user can declare any number of tools as a set of permanent tools which are loaded first when the tool configuration is setup. These tools are given a permanent location in the tool magazine. When tools are exchanged due to product variety, these tools are removed only as the last resort.

5.5.3 PMIX.USR DATA FILE

Description : This file contains a set of part type names which are used to setup the initial tool configuration.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 5.5.

User Options:

The user can set any combination of part types as the initial product mix which is used to setup the initial tool configuration. The tools associated with this part set are selected and loaded into the tool magazine. However, if the tool requirements for the given part set exceed the tool magazine capacity, all required tools cannot be loaded.

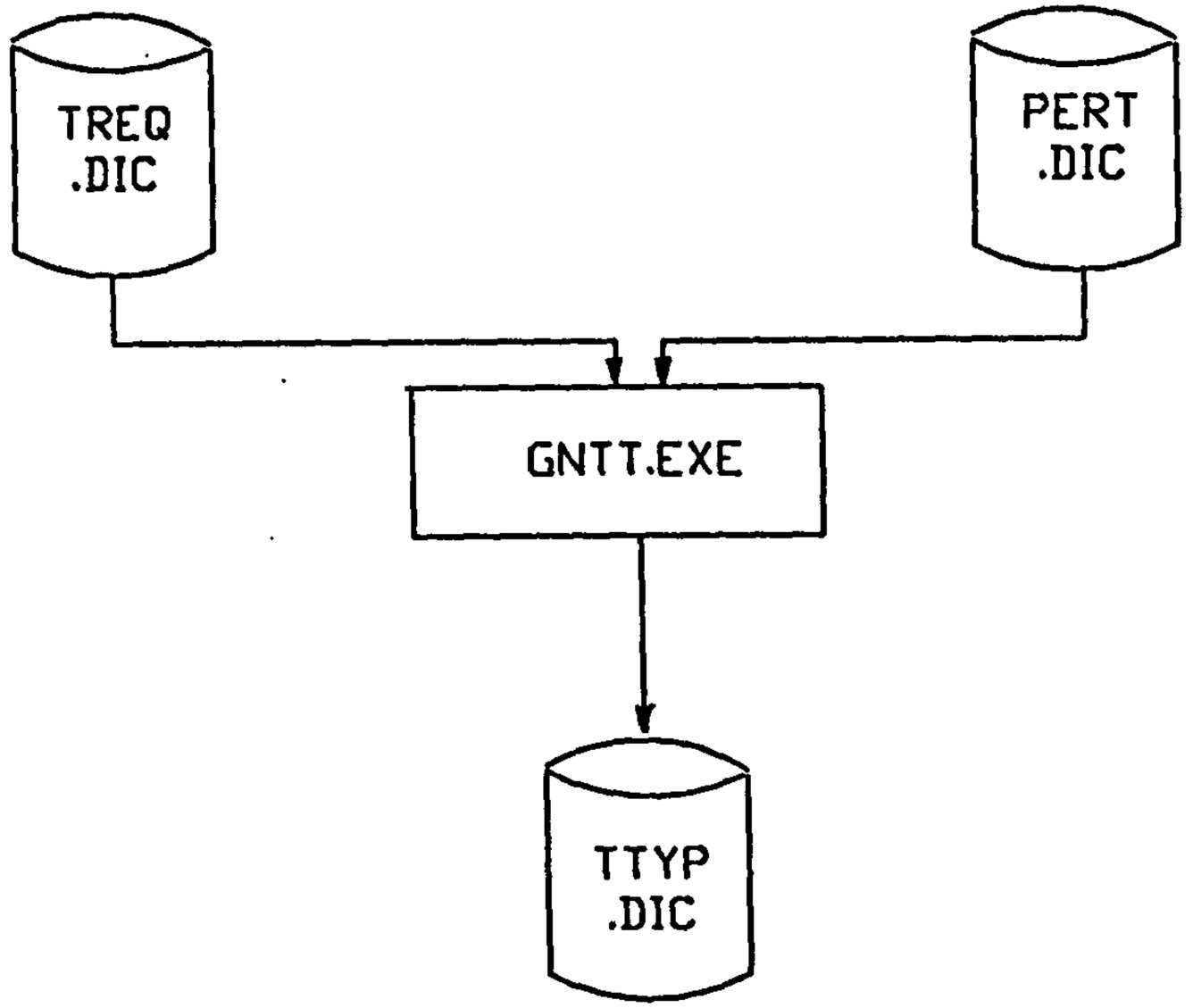


Figure 5.6 The Input/Output File Organization of GNTT.EXE Module

5.6 GNTT.EXE PROGRAM MODULE

Description : This program module generates various attributes related to each tool type used in the system.

Input/Ouput Files

Organization : As shown in Fig. 5.6.

Input Files : TREQ.DIC (section 5.4.1)

PERT.DIC (section 5.5)

Ouput Files : TTYP.DIC (section 5.6.1)

Logical Process :

This program consists of the following logical modules;

a. Establishing tool re-grinding level.

The user is prompted for the re-grinding level which is expressed as a percentage of the tool life. When the cumulative usage time passes this limit, the tool is re-ground.

b. Establishing the minimum and maximum cutting assignment for each tool type.

A tool can be assigned for more than one operation and these operations may have different cutting times. In this section the minimum and maximum cutting time assigned to each tool are established. These data may be used when tools are unloaded at intermediate stages.

c.Calculating the number of duplicates for each tool type.

In this section the number of duplicates required for each tool type is calculated in the following manner;

i. Estimate the number of station assignments of each tool type.

A particular tool type may be assigned to more than one station. The program first calculates the number of station assignments of each tool type by scanning tool requirements data.

ii. Calculate the number of duplicates.

It is assumed that at least one backup tool is made available to each workstation and one additional tool is allocated as a reserved tool.

No. of duplicates = (2* no. of station assignments) + 1

d. Marking permanent tool types

The permanent tools are marked in this section. These tools are given the highest priority when tools are loaded.

1	1	9998	6999	0	0	11	1
2	3	9500	6649	10	8331	5	0
3	1	10000	6999	195	5007	5	0
4	1	5000	3499	7	4576	9	0

Record number (=tool type ID)
Tool Class type
Tool life
Regrinding level
Minimum cutting assignment
Maximum cutting assignment
Number of duplicates
Permanent tool index [pi]
permanent tool; pi =1
non permanent too; pil = 0

Figure 5.7 TTYP.DIC Data File Structure

5.6.1 TTYP.DIC DATA FILE

Description : This file contains attributes related to each tool type used in the system.

File Access : Direct

Data Organization : As shown in Fig. 5.7.

User Options : None. This file is automatically created by GNTT.EXE (section 5.6) program module.

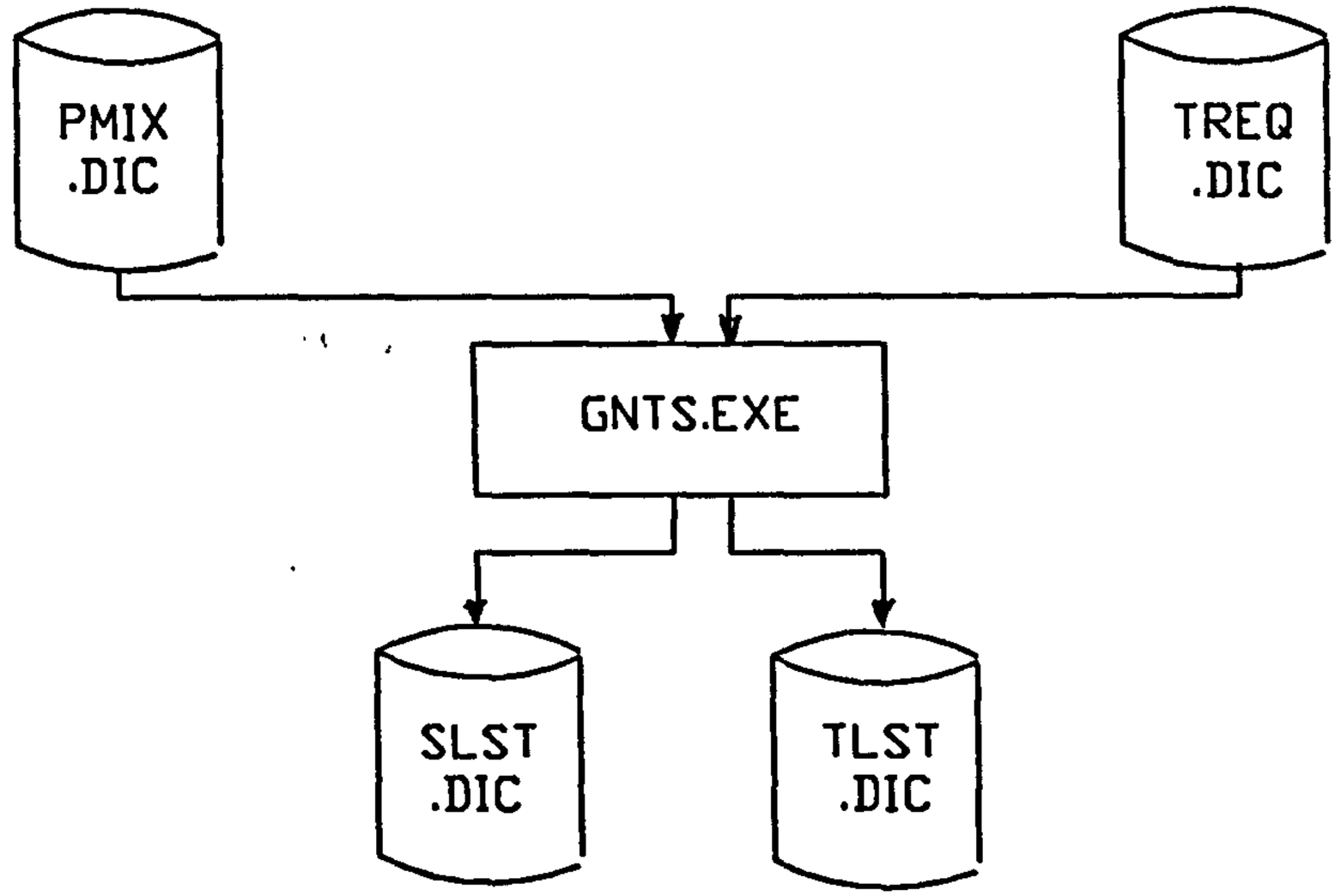


Figure 5.8 The Input/Output File Organization of GNTS.EXE Module

5.7 GNTS.EXE PROGRAM MODULE

Description :This program generates lists of tools required at each workstation for a given product mix.

Input/Ouput File Organization : As shown in Fig. 5.8

Input Files : PMIX.DIC (section 5.5)

TREQ.DIC (section 5.4.1)

Ouput Files : SLST.DIC (section 5.7.1)

TLST.DIC (section 5.7.1)

Logical Process :

This program uses the set of part types defined by the user to generate lists of tools required at each workstation.

When tools are loaded from the tool crib to tool magazines these lists are used.

4	1	117
5	118	234
7	235	335
8	336	477
9	478	591

Station number.
First tool record number.
Last tool record number.

1	2
2	3
3	4
4	5
114	334
115	335

Record number.
Tool type ID.

Figure 5.9 SLST.DIC and TLST.DIC Data File Structures

5.7.1 SLST.DIC AND TLST.DIC DATA FILES

Description : These files jointly hold the lists of tools required at each workstation.

File Access : Direct

Data Organization : As shown in Fig. 5.9.

User Options : None. These files are created by GNTS.EXE (section 5.7) program module.

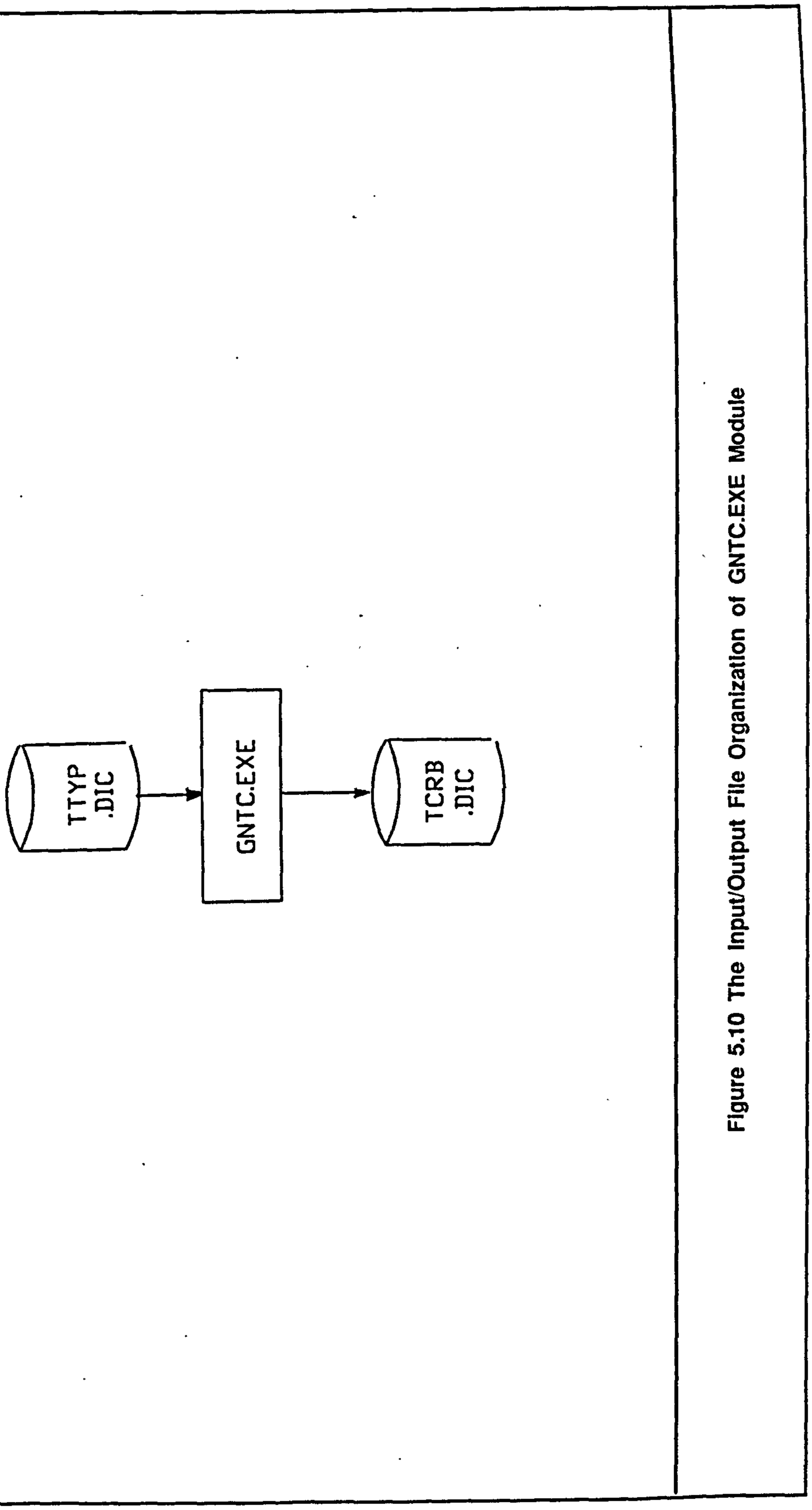


Figure 5.10 The Input/Output File Organization of GNTC.EXE Module

5.8 GNTC.EXE PROGRAM MODULE

Description : This program generates the tool crib configuration.

Input/Ouput File Organization : As shown in Fig. 5.10.

Input Files : TTYP.DIC (section 5.6.1)

Output Files : TCRB.DIC (section 5.8.1)

Logical Process :

The number of duplicates defined for each tool type are used here to set the size of the tool crib. Every individual tool is given a specific location in the tool crib. When tools are assigned to tool magazines, the tools are taken from the tool crib.

1	1	4	9998
2	1	5	9998
3	1	7	9998
4	1	8	9998
5	1	9	9998
11	1	0	9998
..	..	..	..
..	..	..	..

Tool ID.
Tool type ID.
Current location (station number).
Remainig tool life
Remaining tool life.

Figure 5.11 TCRB.DIC Data File Structure

5.8.1 TCRB.DIC DATA FILE

Description : This file contains information about the tool crib configuration.

File Access : Direct

Data Organization : As shown in Fig. 5.11.

User Options : None. This data file is automatically created by GNTC.EXE (section 5.8) program module.

Note : When the tool post-processor is executed, the data in this file is updated. Therefore it is necessary to reset the values in this file, before the next use of the tool post-processor. This automatically is carried out by TOOL.BAT batch command file.

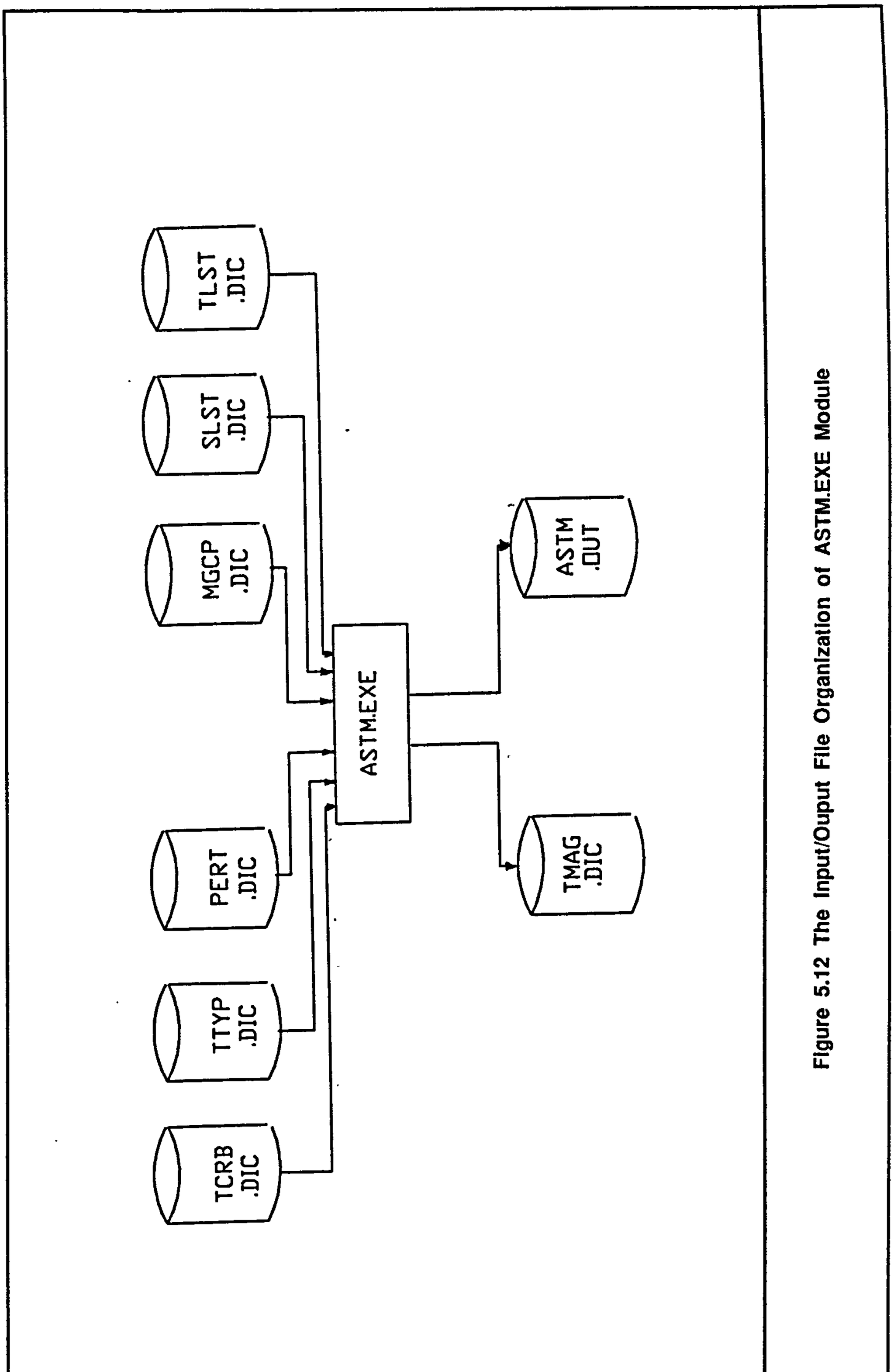


Figure 5.12 The Input/Ouput File Organization of ASTM.EXE Module

5.9 ASTM.EXE PROGRAM MODULE

Description : This program sets up the tool magazine configuration of the system.

Input/Output File Organization : As shown in Fig. 5.12

Input Files : SLST.DIC and TLST.DIC section 5.7.1)

TTYP.DIC (section 5.6.1)

PERT.DIC (section 5.5)

TCRB.DIC (section 5.5)

MGCP.DIC (section 5.5)

Ouput Files : TMAG.DIC (section 5.9.1)

ASTM.OUT (section 5.9.2)

Logical Process :

This program takes tools from the tool crib and assigns them to tool magazines. The number of class III tools allowed in the magazine is set by the user. The tools are loaded in the following order;

a. All permanent tools.

b. All Class III tools.

c. Class I and Class II tool alternately.

Program Routines

R01 program main

This main program first reads a number of input data files and then call appropriate subroutines to load tools into magazines.

R02 subroutine ldt1

This routine loads Class I type tools.

R03 subroutine ldt2

This routine loads Class II type tools.

R04 subroutine ldt3

This routine loads Class III type tools.

R05 subroutine pktl

This routine selects a tool of a given type from the tool crib.

R06 subroutine rntl

A tool which cannot be loaded due to limited space, is returned to the tool crib.

R07 subroutine ldtl

This rouitne loads tools in a pre-defined order. It also prompts for the number of class III tools allowed in the each magazine.

R08 subroutine sltl

This routine identifies the permanent tools in a given list.

R09 subroutine rktb

This routine ranks tool lists in a pre-defined order.

R10 subroutine rptr

This routine writes tool loading statistics.

Class I	1	1	1	← Record number.
Class II	2	134	2	Tool ID.
	3	-154	3	Tool class.
Class III	4	154	3	
	5	-154	3	
	6	276	2	
	7	138	1	

Figure 5.13 TMAG.DIC Data File Structure

5.9.1 TMAG.DIC DATA FILE

Description : This file contains information related to tool magazine configuration.

File Access : Direct

Data Organization : As shown in Fig. 5.13

User Options : None. This file is automatically created by ASTM.EXE program module.

Station No: 4
No.of tools to be loaded stn;
Class I : 88 Class II ; 19 Class III: 10

No. of tools not loaded stn;
Class I : 37 Class II ; 0 Class III: 0

Station No: 5
No.of tools to be loaded stn;
Class I : 88 Class II ; 19 Class III: 10

No. of tools not loaded stn;
Class I : 37 Class II ; 0 Class III: 0

Station No: 7
No.of tools to be loaded stn;
Class I : 89 Class II ; 10 Class III: 2

No. of tools not loaded stn;
Class I : 5 Class II ; 0 Class III: 0

Station No: 8
No.of tools to be loaded stn;
Class I : 121 Class II ; 12 Class III: 9

No. of tools not loaded stn;
Class I : 60 Class II ; 0 Class III: 0

Station No: 9
No.of tools to be loaded stn;
Class I : 85 Class II ; 6 Class III: 23

No. of tools not loaded stn;
Class I : 33 Class II ; 2 Class III: 1

Figure 5.14 ASTM.OUT Data File

5.9.2 ASTM.OUT DATA FILE

Description :This file contains information about tool loading statistics. It shows the number of tools loaded in each class type at each workstation.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 5.14

User Options : Inspection only.

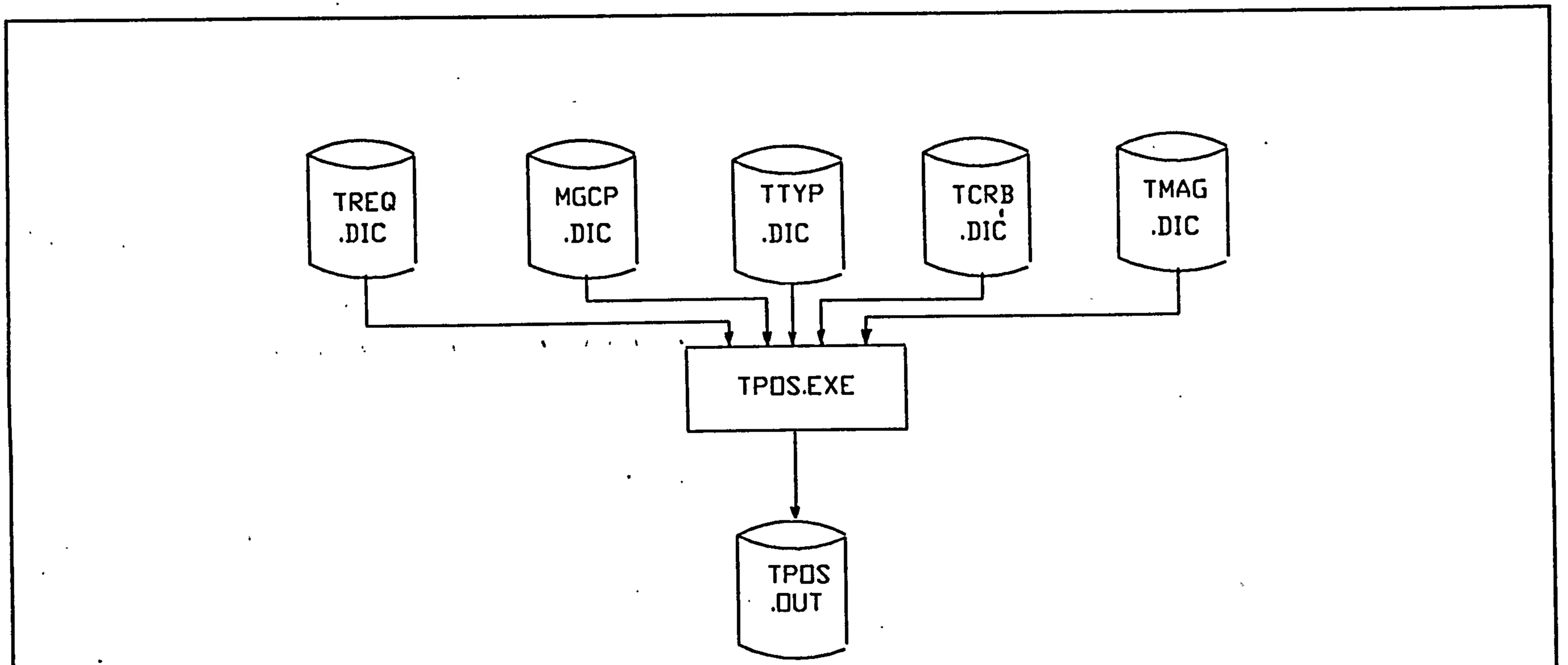


Figure 5.15 The Input/Output File Organization of TPOS.EXE Module

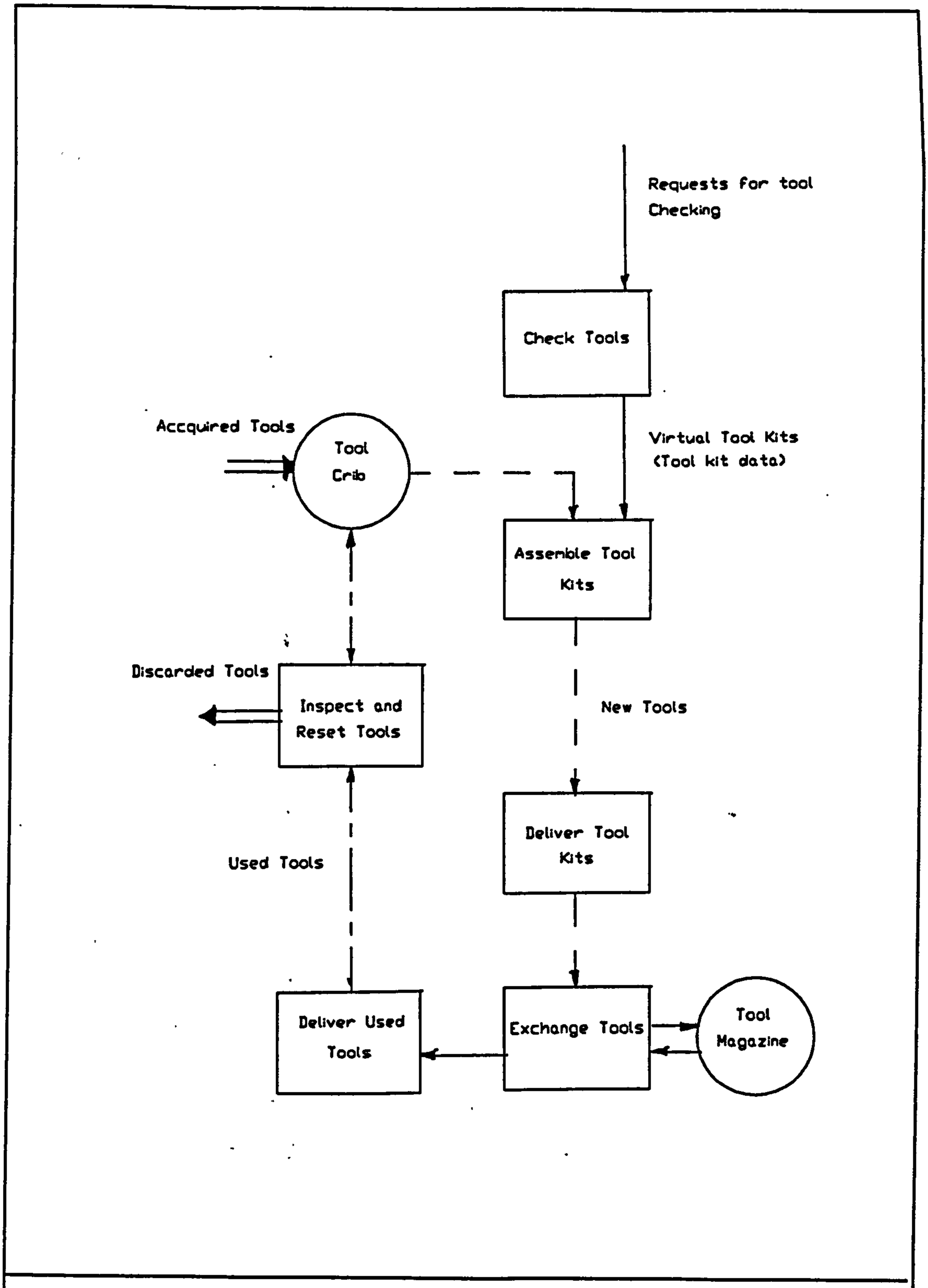


Figure 5.16 The Tool Flow in a Typical FMS

5.10 TPOS.EXE PROGRAM MODULE

Description : This is the tool post-processor. It uses the data files generated from other modules to estimate certain parameters of the tool management system.

Input/Output File Organization : As shown in Fig. 5.15.

Input Files : TREQ.DIC (section 5.4.1)

MGCP.DIC (section 5.5)

TTYP.DIC (section 5.6.1)

TCRB.DIC (section 5.8.1)

TMAG.DIC (section 5.9.1)

Output Files : TPOS.OUT (section 5.10.1)

TPLN.OUT (section 5.10.2)

Logical Process :

The tool flow in a typical flow is shown in Fig. 5.16. A number of program routines have been constructed to mimic operations at each stage.

Note : Time delays are not considered within the tool post-processor.

Program Routines

R01 program main

This is the executive program. It reads records of the operation trace file and calls appropriate program routines.

R02 subroutine inar

This routine opens required input files and reads data.

R03 subroutine tchk

This routine checks the presence of tools at a magazine for a given operation. If tool exchanges are necessary, a temporary list of required tool is created.

R04 subroutine crkt

This routine uses the above mentioned (R03) temporary tool list to create a virtual tool kit (i.e. information about required tool). The kit is assigned a unique identification number.

R05 subroutine askt

This routine picks tools from the tool crib to assemble tool types defined by a virtual tool kit.

R06 subroutine drkt

This routine delivers the assembled tool kit to the workstation.

R07 subroutine exwt

This routine exchange worn out tools. The new tools is placed in the pocket occupied by the old tool.

R08 subroutine expt

The tools are exchanged due to product variety. It is required to find an empty pocket for tools involved in this kind of exchange.

***R09* subroutine fnpk**

This routine finds a pocket for a new tools. The following preference orders are considered in each case.

In-coming tool type : Class I

Empty pocket for Class I tool

Non-permanent Class I tool

Non-permanent Class II tool

Permanent Class I tool

Permanent Class II tool

Non-permanent Class III tool

Permanent Class III tool

In-coming tool type : Class II

Empty pocket for Class II tool

Non-permanent Class II tool

Non-permanent Class I tool

Permanent Class II tool

Permanent Class I tool

Non-permanent Class III tool

Permanent Class III tool

In-coming tool type : Class III

Empty pocket for Class III tool

Non-permanent Class III tool

Permanent Class III tool

R10 **subroutine fnem**

This routine attempts to find an empty pocket for given tool class.

R11 **subroutine fntr**

This routine attempts to find a tool which is not used in the current operation.

R12 **subroutine flag**

This routine flags all tools used in the current operation.

R13 **subroutine wrkt**

This routine writes the tool kit parameters to the output file.

R14 **subroutine uplf**

This routine updates tool life data.

R15 **subroutine dlkt**

This deletes a tool kit.

R16 **subrouitne pstl**

This routine resets the tool attributes.

[a]	[b]	[c]	[d]	[e]	[f]	[g]	[h]
324.000	4	1	6	1	1	0	0
675.000	4	1	2	1	1	0	0
773.000	4	2	2	2	2	0	0

[a]. Simulation clock time
[b] Request station number
[c] Request part number
[d] Request operation number
[e] Tool kit size
[f] No. of changes due to wear
[g] No. of tools brought into the station for exchanges due to product variety
[h] No. of tools unloaded for exchanges due to product variety.

Figure 5.17 TPOS.OUT Data File Structure

5.10.1 TPOS.OUT DATA FILE

Description : This file contains information related to tool kits created within the tool post-processor.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 5.17.

User Options : Inspection only.

** Week No: 2 Day: 1 Shift: Day **

** Station no: 4 **

Tool type ID	Quantity
132	1
13	2
4	1
274	1
135	1

Figure 5.18 TPLN.OUT Data File Structure

5.10.2 TPLN.OUT DATA FILE

Description : This file contains information required for tool requirement planning.

File type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 5.18.

User Options : Inspection only.

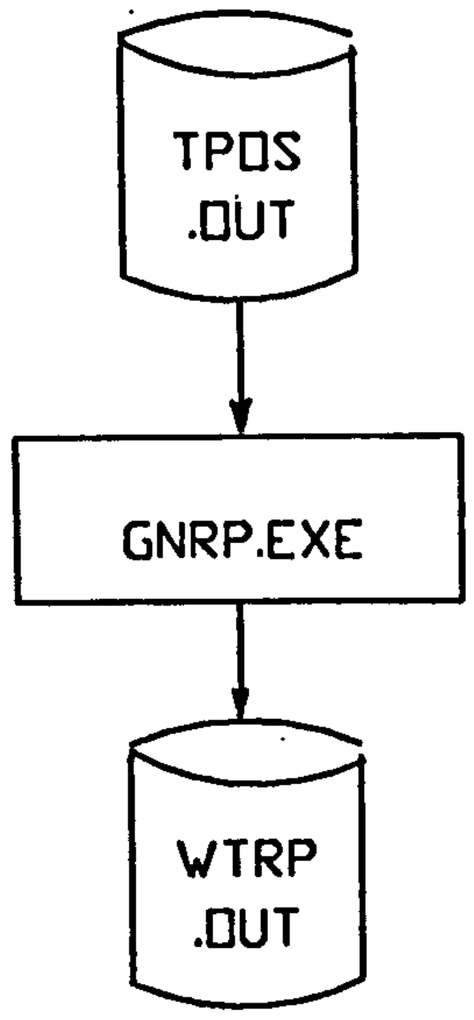


Figure 5.19 The Input/Output File Organization of GNPR.EXE Module

5.11 GNRP.EXE PROGRAM MODULE

Description : This program reads data generated by the tool post-processor and creates weekly performance reports.

Input/Output Files Organization : As shown in Fig. 5.19.

Input Files : TPOS.OUT (section 5.10.1)

Ouput Files : WTRP.OUT (section 5.11.1)

Logical Process : The TPOS.OUT file contains information about individual tool kits created within the post-process. This program unit reads this data and generates weekly performance reports.

Week No: 1

Stat. No:	No. of Ops:	No. of Kits	Mx. Kt Size	Tool Exchanges Wear In	Wear Out	Prod In	Prod Out
4	12	12	5	28	28	0	0
5	5	5	4	11	11	0	0
7	2	2	2	3	3	0	0
8	16	16	10	16	16	73	73
9	6	6	6	13	13	0	0
Total	41	41		71	71	73	73

Figure 5.20 WRTP.OUT Data File Structure

5.11.1 WTRP.OUT DATA FILE

Description : This file contains information about weekly performance of the tool management system.

File Type : ASCII

File Access : Sequential

Data Organization : As shown in Fig. 5.20.

User Options : Inspection only.

CHAPTER 6

APPLICATIONS, LIMITATIONS AND FURTHER ENHANCEMENTS

6.1 APPLICATION AREAS FOR GENERAL USERS

This simulation system mimics operations of a complex FMS, in particular, the parts have very complex flow patterns (multiple operations on different fixture types). It includes all major features of FMSs, including fixtures and secondary material handling systems such as cranes. This simulation system can be used in the following areas;

a. Understanding FMS Operating Characteristics

A variety of experimental environments can be created by the user to understand the effects of changes in input conditions. For example, the effects of the number of pallets on the system performance can be easily investigated by changing the initial conditions.

b. As a teaching tool.

This simulation system can be used as a demonstration tool in undergraduate and postgraduate classes to explain operations of a typical FMS. It can be regarded as a comprehensive teaching tool as it compromises a part flow simulator, a graphical post-processor and a tool post- processor. The data files can be used to describe the organization of FMS database.

6.2 LIMITATIONS OF THE SIMULATION SYSTEM

This simulation software system was developed within a very limited time period to support the main research programme. Therefore, the following constraints were imposed at the design stage;

a. A single AGV system.

At present it is assumed that all workstations are served by a single AGV. The exact track layout is not considered instead an average travelling time is assumed for all movements.

b. A limited number of decision making rules.

The default system logic used in the simulator, represents operations of a real FMS. Therefore at present alternative decision rules cannot be used. However facilities have been provided to integrate further decision rules in the system.

6.3 FURTHER ENHANCEMENTS

This simulation system can be further enhanced in various directions;

a. A modular simulation system

It is possible to develop further program modules to represent a variety of different features of FMSs and all modules can be stored in an object library. The program modules required for a given configuration, can be selected from the program library to generate the simulator.

Although simulation languages such as SIMAN provides better facilities for modular architecture, the execution speed for large models (written in SIMAN) appears to be long.

b. Linking a database management system.

As mentioned before, the management of data within the simulator is an important task. The performance of the simulator can be enhanced by linking the simulator to a database management system.

c. Linking an expert system.

Generally, simulation models are repetitively executed, with human intervention at each stage, to solve a given problem. It may possible that the decision making process can be automated by linking the simulator to an expert system.

6.4 CONCLUSION

A comprehensive simulation tool has been developed. It provided an opportunity to understand various facets of developing a computer simulation software. A number of operating systems, graphic systems, etc. were studied in detail.

This simulation system is used by a collaborating company to support the management decision making.

It is anticipated that a more powerful simulation tool can be generated by integrating the simulator to other software tools such as database management systems and expert systems, etc.

REFERENCES

Hopgood F R A, Due D A, Gallop J R and Sutclife D C, 1983, Introduction to Graphical Kernel System, Academic Press.

IBM Personal Computer Software (Professional Graphic Series), 1984(a), Personal Computer Graphical Kernel System, Programmers Guide.

IBM Personal Computer Software (Professional Graphic Series), 1984(b), Personal Computer Graphical Kernel System, Volume 3: Language Binding (FORTRAN)

Perera D T S, 1986, User Manual (FMS Simulator), Anderson Strathclyde Plc, Motherwell, Glasgow.

Perera D T S and Carrie A S, 1987, A Simulation Tool for Real Time Scheduling of FMS, Proceedings of the Third National Conference on Production Research, Nottingham.

Pritsker A A B, 1975, Simulation with GASP IV, Prentice Hall Inc.