AN ABSTRACT OF THE THESIS OF

Ramesh Pendakur for the degree of Master of Science in Industrial and Manufacturing Engineering presented on September 29, 1989.

Title: A Microcomputer-Based Information Management and Production Planning System - An Application in Discrete Parts Manufacturing

Abstract Approved:

Sabah U. Randhawa

The focus of this research is to design and develop a microcomputer-based information management and production planning system for a discrete parts manufacturing facility. The information management module is based on relational database concepts. This information is used by the scheduling module to generate initial production schedules based on manufacturing due dates. Appropriate user interfaces are provided to manage the databases and to help the user to refine the schedule to satisfy specific preferences. Additionally, automatic heuristic procedures are developed for work load balancing and schedule refinement.

An integrated system is developed for a single-stage production process with significant fluctuations in demand caused by the inherent randomness of the product orders. The information management module is implemented using the database software package, dBASE III PLUS and the work load balancing procedures are implemented in Microsoft C. The heuristic procedures are analyzed to evaluate their relative performance. The system has been implemented for A-dec Inc., Newberg, Oregon, a major dental clinical equipment manufacturer.
A Microcomputer-Based Information Management and Production Planning System - An Application in Discrete Parts Manufacturing

by

Ramesh Pendakur

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed September 29, 1989
Commencement June 1990
ACKNOWLEDGEMENTS

I am thankful to many individuals that have made my stay at Oregon State University a fruitful learning experience. I am most thankful to my two supervisors, Dr. Sabah Randhawa and Dr. Kenneth Funk. Without their continuous support, guidance and encouragement this thesis would not have been possible.

From Adec, I would like to thank Charles Cook, Tom Welch and Cheryl Gaibler for their continuous involvement in the developmental stages of this project. Their constructive criticisms and suggestions have been of much use in making the system useful and user-friendly.

I am thankful to my parents for their love and support. Without the support and understanding of my family members, I would not have been able to achieve my goals. I am also thankful to my uncle Setty Pendakur for his encouragement, interesting conversations, timely help and support during my stay here.

Finally, I must thank Rebecca for her patience, love and understanding.
# TABLE OF CONTENTS

1. INTRODUCTION .......................................................................................... 1
   A. General Problem Background ............................................................. 1
   B. Role of Production Planning and Control ........................................... 3
   C. Information Requirements for Effective Job Shop Control ................. 4
   D. Need for an Integrated Job Shop Control System ............................... 5
   E. Brief review of Available Tools .......................................................... 6
   F. Objectives, Methods and Approach .................................................... 10

2. SYSTEM OVERVIEW .................................................................................. 13
   A. Information Management Module .................................................... 15
      1. Database Module ........................................................................... 17
      2. Scheduling Module ....................................................................... 17
   B. Automatic Work Load Balancing Module ........................................... 18
   C. System Characteristics ..................................................................... 18
   D. System Inputs .................................................................................. 19
   E. System Outputs ................................................................................ 20
   F. Development Tools .......................................................................... 20

3. INFORMATION MANAGEMENT SYSTEM .............................................. 22
   A. Main Menu ....................................................................................... 22
   B. Database Module ............................................................................. 24
      1. Product Structure Management ..................................................... 24
      2. Sales Orders Management ............................................................. 28
      3. Report Generation ....................................................................... 31
      4. Design of the Databases ................................................................. 31
   C. Scheduling Module ......................................................................... 35
      1. Schedule Generation ..................................................................... 36
         a. Report Generation .................................................................... 37
         b. Graphical Scheduling Information Reports ............................... 38
      2. Interactive Load Balancing ............................................................ 40

4. AUTOMATIC WORK LOAD BALANCING MODULE .............................. 44
   A. Design of Automated Work Load Balancing Models .......................... 47
      1. Assumptions ............................................................................... 47
         a. Operational Assumptions ......................................................... 49
         b. Heuristic Mechanism Design Assumptions .............................. 49
      2. Heuristic Mechanisms .................................................................. 50
      3. Rescheduling Policies .................................................................. 55
      4. Resultant Load Balancing Models ............................................... 55
      5. Performance Measures ................................................................ 61
      6. Data For Model Comparison ....................................................... 65
      7. Results and Interpretation ............................................................. 67
      8. Analysis of the Heuristic Models ................................................... 73
         a. Two-Way ANOVA Results ...................................................... 73
         b. Summary of Results ............................................................... 75

5. CONCLUSIONS ....................................................................................... 77
   A. Summary of Accomplishments and Conclusions ............................... 77
   B. Directions for Future Research ......................................................... 78
      1. Heuristic Methods for Multi-Stage Production Process ................ 78
      2. Rescheduling Multiple Jobs per Rescheduling Action ................... 78
TABLE OF CONTENTS

3. A Graphical User Interface ................................................................. 79
4. Expert System Aid for Enhanced Decision Making .............................. 79
5. Development of an Improved Total Cost Expression ............................ 80

BIBLIOGRAPHY ................................................................................... 81

APPENDICES
APPENDIX A : Hardware and Software Requirements of the System .......... 83
APPENDIX B : dBASE III PLUS Source Code ........................................ 84
APPENDIX C : Microsoft Quick C Source Code .................................... 149
APPENDIX D : Flow Diagrams for Models 2 through Model 6 ............... 170
APPENDIX E-1 : Performance Measures Dataset .................................. 177
APPENDIX E-2 : ANOVA - Early Jobs .................................................... 178
APPENDIX E-3 : ANOVA - Late Jobs ...................................................... 179
APPENDIX E-4 : ANOVA - Total Number of Jobs Rescheduled ............. 180
APPENDIX E-5 : ANOVA - Average Distance Moved per Job ............... 181
APPENDIX E-6 : ANOVA - Variance Reduction ...................................... 182
LIST OF FIGURES

1. The Vicious Circle ................................................................. 2
2. Schematic Diagram of the System ............................................. 14
3. Hierarchy of Components of the Information Management Module .......... 16
4. Main Menu of the System ......................................................... 23
5. An Example Product Structure .................................................. 25
6. Product Structure Management Menu ......................................... 26
7. Adding a New Product Structure .............................................. 27
8. Product Structure Reports Menu ............................................... 28
9. Sales Orders Management Menu ............................................... 28
10. An Example Product Structure Report ....................................... 29
11. Adding a New Sales Order into the System .................................. 30
12. Sales Orders Report Menu ....................................................... 31
13. Product Structure Database Design ........................................... 33
14. Sales Orders Database Design .................................................. 34
15. Sales Orders Priorities Database Design ..................................... 34
16. Initial Schedule Database Design .............................................. 35
17. Scheduling Information Reports Menu ........................................ 37
18. Types of Scheduling Information Reports ..................................... 38
19. An Example Scheduling Information Report ................................... 39
20. Scheduling Information Graphics Menu ....................................... 40
21. An Example Bar Graph for the Machining Section .......................... 41
22. Flow Diagram for the Interactive Load Balancing Procedure .............. 42
23. Work Load Balancing Menu ..................................................... 46
26. Concept of Total Slack ............................................................. 52
27. Concept of a Planning Window .................................................. 54
28. Experimental Design (Resultant Models) ...................................... 56
29. Terminology used in the Models ............................................... 57
30. Flow Diagram of the Load Balancing Model 1 ................................ 58
31. Flow Diagram for Data Generation ............................................ 66
32. Automatic Work Load Balancing Procedures - Summary Output ............ 67
33 a & b. Graphical Comparison of Models 1 & 2 ............................... 69
33 c & d. Graphical Comparison of Models 3 & 4 ............................... 70
33 e & f. Graphical Comparison of Models 5 & 6 ............................... 71
1. INTRODUCTION

A. General Problem Background

A discrete parts manufacturing facility is a flexible production operation that consists of a set of versatile work stations or machine centers, and is capable of producing a wide variety of jobs. Usually these jobs are manufactured in small to medium quantities (typical of a job shop). The processing of any job consists of a sequence of tasks to be performed on the job through the various work stations. Typically, in a job shop, the job order is initiated by the customer and it is often unique. The processing requirements for the order dictates the routing of the orders through the work stations. Due to the wide variety of jobs produced in a job shop, it may be difficult to identify strong work flow pattern through the shop. Also, the job shop control problem is governed by a number of conflicting objectives such as optimizing machine and manpower utilizations, meeting due dates and minimizing work-in-process inventories. In this context, the job shop control problem is characterized by the complex on-going decision making process concerned with the timely allocation of materials, labor, machines and other resources to meet the objectives of the shop.

The flow of job orders through the manufacturing facility is a complex process. The product mix, machine center capacities and utilizations, variations in the processing times, and labor assignments are some of the factors that
contribute to the complexity of the shop floor control problem. The conflicting nature of the objectives of a shop and the resulting implications are best illustrated by Figure 1 [COPICS, 1972].

![Vicious Circle Diagram](Source: COPICS, 1972)

**Figure 1** The Vicious Circle *(Source: COPICS, 1972)*

The following scenario (drawn from Ballakur and Steudal, 1984) can be visualized to understand Figure 1. The job orders come to the shop with due date requirements usually based on customer needs and plant work load. Consider
the case of a job shop where the due dates are being missed. As a result of this, the management may decide to increase the lead times on the orders and release the orders to the shop earlier. This in turn will result in increased work-in-process inventories. The increased work load causes congestion and extends the job throughput times. This results in missing of more due dates. Thus, the emphasis on the need for an effective production control system.

B. Role of Production Planning and Control

Production planning and control deals with the decision making process about setting of production, inventory, and work force levels to satisfy product demand over a medium-term planning horizon. The need for production planning and control is emphasized by the complexity of the shop as well as the fluctuations in the product demand. The first step in this process is aggregate planning. The purpose of aggregate planning is to determine the resources such as labor, equipment and materials that should be acquired for scheduling production. When the resources are acquired and the demand is either known or accurately forecasted, scheduling is used to allocate the resources for production. Hence, the purpose of scheduling is to ensure that available resources are efficiently and effectively used to achieve the organization's goals.

An additional important shop related criteria that has been identified is the measure of work load balance at each work center over the planning horizon [Deane and Moodie, 1972]. The rules and methodologies used in scheduling try to optimize some job related performance measure such as meeting due dates. But work load balancing is important under the assumption that any shop is set up to operate at a certain production level, called the planned capacity utilization level. If there are deviations from this level during the implementation of a schedule,
some penalty costs will be incurred. If the facilities are underutilized, the costs of idle machinery, idle laborers and idle space will increase. This will be reflected in the cost of the products produced by the facility. In the case of overload, cost of overtime hours will have to be paid. These costs could be higher than the normal production costs.

In a shop with balanced work loads over a given planning horizon, the manager can do a better job of allocating resources to increase the efficiency of the shop. In labor-intensive industries, the advantages of having a balanced work load results in keeping costs of idle labor at a minimum, and would also reduce the costs associated with hiring, firing and training of employees.

C. Information Requirements for Effective Job Shop Control

To operate a production control system effectively, five basic types of information have been identified [Niland, 1970]:

1. Available shop capacities, products and their quantities to be produced,

2. Quantities of each resource required to produce a given quantity of an end-item,

3. A forecast of demand for products, a backlog of orders, or both,

4. Cost information relevant to the choices that must be made in planning, scheduling and inventory control, and

5. An estimate for uncertainties with respect to the capacities available and the demand fluctuations.
Measures of shop capacities, resources and other inputs must be standardized. Typically this is expressed in standard hours which can be man hours or machine hours or a composite of both. The forecasted demand can also be converted into standard hours. The cost information is usually used in making choices among alternative schedules or planning decisions. Cost estimates such as the cost of loss of business or goodwill as a result of missed due dates are quite difficult to quantify. Hence, guidelines should be provided to estimate such costs. It should be noted that many of the cost estimates rely heavily on the experience of the managers and should be used with caution.

D. Need for an Integrated Job Shop Control System

To realize the goals of an organization, there needs to be an adequate planning and control system installed in the facility. The system should not only facilitate the development of good production plans (schedules) but also ensure that the schedules are implemented correctly. To conceptualize and design such a system, the objectives of the system and the expected outputs from the system have to identified. The tasks in which the help of the system might be sought are:

1. To know what due dates to promise to the customers. These due dates could be based on both customer needs and operations considerations. The considerations could include available capacity, product characteristics, and efficiency of operations. The system could provide load distributions of the shops so that judicious choice of due dates can be made.
2. The user might need shop status information to carry out capacity planning.
   A planning and control system can answer this question by forecasting and developing work loads for individual work centers.

3. The user needs to know when to start production on an order. If the due date, processing times and order priority information is provided to the system, the system can generate production schedules for the orders.

4. In case of breakdowns or other unpredicted events, the system should be able to provide information on the planned loading levels and slack times available so that corrective actions can be taken to change the production plans.

In order for the system to be able to store and maintain relevant information to answer such questions, a computerized database seems to be a prerequisite. A database management software system would provide the capability to store data records in files on random access storage devices and automatically create and maintain all the relationships that are required between data records. User interfaces can be provided to update the information in these databases. The advantage here is that a computerized planning and control system could access the information in these databases to generate production plans.

E. Brief Review of Available Tools

In recent years, many computerized systems have been developed for production planning and inventory control. One such popular system is Materials Requirements Planning (MRP)[Riggs,1987]. There are three different types of MRP systems that are available. MRP-I is an inventory control system which
releases manufacturing and purchase orders at the right time to support the master schedule. An MRP-II system is an information system used to plan and control inventories and capacities in manufacturing. This system is called a closed-loop system and it controls both inventory and capacities. A Type III MRP is used to plan and control all the manufacturing resources. An MRP system is driven by the master schedule and the bill of materials. It requires an adequate computer system, accurate data, management support, and user knowledge through training. MRP systems have known to yield very good results when implemented with complete management support. The drawbacks of this kind of a system are the costs of equipment, cost of training personnel, and other implementation and operation costs.

Another computerized production control system that is available in the market is Optimized Production Technology (OPT). The objective of OPT is throughput optimization. This is achieved by eliminating bottlenecks in the production process [Riggs, 1987]. OPT splits the orders into number of small batches and schedules these batches consecutively to reduce throughput time. OPT is designed to produce near optimal schedules. However, OPT also needs a mid-sized computer system to support it, and is not set up for personal computers.

The Production Scheduling and Control (PSC) System was developed by Honeywell for use in discrete part manufacturing facilities. This system is fully computerized and performs the following tasks: 1) developing a capacity requirements plan, 2) scheduling and dispatching orders, and 3) shop floor control reporting [Schroeder, 1982]. The PSC system is driven by a production database
and rules and methods of production control. Essentially, PSC provides an information environment in which production planning and control is made easy.

There are many other systems that are available in the market, but most of them need mini to mid-range computer systems for implementation. Another potential problem with the systems available in the market is that they are general purpose systems. These systems have to be adapted to fit the specific manufacturing environments. The may necessitate time and cost expenditures to modify the system to fit the user environment, or the user may have to use a system that is "suboptimal" for the environment. Another important factor that may make it difficult for smaller companies to implement these systems is the costs of equipment, installation and maintenance, user training and other implementation costs. For example, the average company in the study by Schroeder et al. (1980) reported spending $375,000 on MRP system installation, with an eventual cost of $618,000 when the MRP system is fully implemented. The installation costs include people, software, hardware and training required for the development of the system. The costs vary with the size of company (Table 1). This kind of spending may be prohibitive for smaller companies. Also, the capabilities of the commercially available systems, may in some cases, be more than what the company needs.
<table>
<thead>
<tr>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost to-date $375,000</td>
<td>$600,000</td>
</tr>
<tr>
<td>Eventual Cost $618,000</td>
<td>$1,137,000</td>
</tr>
</tbody>
</table>

Cost of MRP Installation (thousands of dollars)

<table>
<thead>
<tr>
<th>Annual Sales</th>
<th>Current Cost</th>
<th>Eventual Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Under $10 million</td>
<td>$93</td>
<td>$194</td>
</tr>
<tr>
<td>$11-25 million</td>
<td>210</td>
<td>385</td>
</tr>
<tr>
<td>$26-50 million</td>
<td>298</td>
<td>560</td>
</tr>
<tr>
<td>$51-100 million</td>
<td>511</td>
<td>912</td>
</tr>
<tr>
<td>$101-500 million</td>
<td>565</td>
<td>800</td>
</tr>
<tr>
<td>Over $500 million</td>
<td>1633</td>
<td>2237</td>
</tr>
</tbody>
</table>

Table 1 Estimated Cost of MRP Installation [Source: Schroeder et al., 1980]

With personal computers (PC) becoming more powerful, user-friendly and smaller in size, their use has increased tremendously in recent years. There is a definite lack of PC-based systems for production planning and control. One way of increasing the usefulness of these machines in a manufacturing environment is to build more PC-based applications that can help the managers
perform their jobs better. PC-based systems are relatively inexpensive, and affordable to most small-sized companies. The trend with larger companies is also PC-based networking systems. These systems tend to be more user friendly and easy to learn as compared to large-scale systems. Hence, the costs of equipment, installation and training are minimized. The monetary risks due to implementation failures of such systems is also comparatively lower.

F. Objectives, Methods and Approach

There are three distinct objectives underlying this research project. The first objective is to provide a mechanism to manage the information needs of a job shop operation by implementing a database system and to provide user interfaces that can aid the user to update and maintain information in these databases. The second objective is to use this information to generate a macro-level production plan for the facility. The third and final objective is to develop a set of heuristic procedures that can be used to refine the initial schedules for load balancing purposes. In order to meet the objectives of this research project, the following methods were adopted.

The initial phase of the project was to define the environment in which the end-product of this research was going to be used. This research effort was a joint project between Oregon State University and A-dec Inc., a major dental clinical equipment manufacturer located in Newberg, Oregon. The ultimate users of this project were identified as the managers and production planners at A-dec. As part of the initial phase, the needs of the users were assessed through several discussions with the managers at A-dec. Observational data from the shop floor and information from other manufacturing support people was also collected to augment the user needs.
Once the user needs were assessed, the first prototype design process began. The design of the databases and the framework for designing the user interfaces was developed through several iterations. During the initial prototype evaluation process, the design was evaluated by the managers at A-dec as well as the project team, and changes and modifications were made to the system based on this feedback. The databases and the user interfaces were finalized after a number of prototypes. The implementation tool used for the databases and the user interfaces was dBASE III PLUS.

The second phase of the project was to identify a methodology to generate the schedules by making use of the information available in these databases. After discussions with the production managers at A-dec, schedule generation using backward loading (backward loading begins with the due date for each job and loads the processing-time requirements against each work center by proceeding backward in time) from the due dates was adopted. This methodology was also coded and implemented in dBASE III PLUS. One of the major tasks that was addressed during these two phases of the project was the format and type of information reports the system was to produce. The process of deciding on the format and standardizing the output required a discussion with the end users at A-dec. The format for the graphical information reports was also discussed and finalized. The graphical information reports program was implemented using dGRAPH software package.

The third phase of the project was to develop user interfaces to accomplish interactive load balancing. The objective here was to provide the users with interactive tools to enable them to modify the shop-level production plans generated during the previous phase. The production plan (schedule) generated
in phase two is primarily based on due dates and does not take any other objectives or constraints into account. This phase, also implemented in dBASE III PLUS, was integrated with the databases and the initial schedule generation module (phases 1 and 2). The report generation utilities were also linked with this phase so that the user could generate all the standard reports after balancing the work load interactively.

The fourth and the final phase of the project was to develop a set of automatic work load balancing procedures. The objectives of these procedures, along with the constraints and performance measures in the system were identified. The development of these procedures was done iteratively by implementing, testing and validating them. These procedures were coded in Microsoft C and were interfaced with the information management module of the system to form an integrated system.

The organization of this thesis is as follows: Chapter 2 gives an overview of the entire system. Chapter 3 discusses the Information Management Module of the system (phases 1,2 and 3). Chapter 4 describes the development of the automatic load balancing procedures (phase 4). Finally, a summary of results, conclusions, and implications for further research are enumerated in Chapter 5.
2. SYSTEM OVERVIEW

The system consists of two main components. Figure 2 shows a schematic view of the system configuration. The two components are the information management module and the automatic work load balancing module. These components are further divided into sub components which serve specific functions.

The information management module is divided into the database module and the scheduling module. The database module is made up of the product structure management component and the sales orders management component. The objective of the information management module is to help the user to manage and maintain detailed sales order, product structures and production capacities information. The scheduling module generates production schedules for each section or shop of the facility, and also helps the user to interactively reschedule the orders to achieve a balanced work load or to satisfy scheduling preferences. The automatic work load balancing module applies heuristic load balancing procedures to the initial schedule to balance the work load over a specified planning horizon.

The product structure database and the sales orders database interact with each other to maintain current and accurate information about the sales orders and the products ordered by the customers. The product structure database essentially holds information about the entire line of products manufactured by A-dec. The sales orders database contains the customer sales orders information. Each sales order is in turn composed of many products. If an order contains a product which has to be custom-built with design variations, then the design
Information Management Module

Product Structures Database

Sales Orders Database

Initial Schedule Generation

Interactive Work Load Balancing

Initial Schedule

Automatic Work Load Balancing Module

Work Load Balancing Models

Schedules

Summary Output

Products & Sales Orders

Work Center Capacities

Report Schedules

Figure 2. Schematic Diagram of the System
department provides the scheduler information about the product’s structure. If there is a change in the product’s structure, then this change needs to be incorporated via the product structure management module so that future production schedules reflect this change.

At the outset, the scheduling module does not have any knowledge of the product structures or the capacities of the shop. It interacts with the sales orders and product structure databases to extract the requisite information. There is no new information added to the sales orders database or the product structures database by the scheduling module. The initial schedule generated is stored in the initial schedule database.

The automatic work load balancing module has six different heuristic models built into it. Each of these models start from the initial schedule generated by the scheduling module, and apply the heuristic rules to achieve an improved schedule. The summary output from this module consists of the performance measures such as the reduction in overtime hours, number of early jobs, number of late jobs and reduction in variance of the load distribution. The purpose of the summary output is to guide the user in selecting one of the six schedules proposed. The user can also obtain detailed scheduling information reports once a schedule has been selected.

A. Information Management Module

This component of the system contains the database module and the scheduling module (Figure 3). The database module serves as a user interface for data entry and also as an input to the scheduling module.
Figure 3. Hierarchy of Components of the Information Management Module
1. Database Module

This is further divided into two components - product structure management and sales order management. The product structure management subcomponent contains information on the product structure and the standard hours required to process the product in the various production areas. The sales order management subcomponent of the system contains detailed information regarding each sales order such as the sales order number, due date and a product list. The information management system performs the following functions:

- Adding new products, their structure and standard times,
- Adding new sales orders,
- Purging obsolete product structure or sales order information,
- Changing structure of a product,
- Changing the specifications in a sales order, and
- Report generation for product structures and sales orders.

2. Scheduling Module

This module has two primary functions: schedule generation and interactive load balancing procedure. The schedule generation function is designed to:

- determine an initial production schedule for a time horizon specified by the user for each of the production sections in the facility,
• generate reports and graphs on the status of the sales orders on a weekly basis, and

• present graphically, the production hour requirements as determined in the initial schedule, and the planned capacity levels.

The interactive load balancing procedure is designed to provide a user interface for making changes to the initial schedule developed by the schedule generation function. The user can reschedule the manufacturing completion time of any of the orders from their initial scheduled time period. This activity may be carried out either to achieve a balanced work load over a planning horizon or to satisfy specific conditions and user preferences.

B. Automatic Work Load Balancing Module

This component of the system is designed to apply heuristic procedures to the initial schedules developed to obtain work load balanced production schedules. There are six heuristic work load balancing procedures that have been developed in this research. Each heuristic procedure focuses on a different set of objectives. The procedures are designed to produce good solutions rather than optimal solutions. The system executes all the procedures and provides a summary report to the user. This report provides the user with performance measures that characterize the schedules developed by the procedures. The system generates detailed reports on the production schedule once the user selects a particular schedule.

C. System Characteristics

Characteristics of the system developed in this research include:
1. User friendly. A user with little or no computer background can use and interact with the system successfully.

2. Menu driven. The major keyboard interaction with the user is during the data input phase and the load balancing routine. Keyboard input has been minimized by providing default values wherever possible.

3. Extensive report generation capabilities. The system can produce reports for any time horizon requested by the user.

4. Extensive graphics capabilities. Bar graphs are used to report loading patterns and deviations from the planned capacity levels. Graphs can be produced on a weekly, monthly, quarterly or half yearly basis or for any other user specified time horizon.

5. The changes made to the sales order specifications are reflected in the scheduling information reports immediately. This helps to maintain a dynamic environment.

D. System Inputs

The inputs to the system consist of all relevant product structures, sales orders and shop capacity information that is needed to generate production schedules. The product structure information consists of product name, product identification number and standard times necessary to make the product in the various production sections. The sales orders information consists of sales order number, production priority, manufacturing due date and product list that comprises the sales order along with the quantities of each product. Another
input to the system is information on the facility's production capacities in hours for each of it's sections.

E. System Outputs

The two primary system outputs are management information reports and the production scheduling information reports. The user can obtain reports on the product structures contained in information management system databases or on the sales orders for a desired planning period. The scheduling module produces different types of reports on the production schedules that are generated. Graphical outputs on the work load for any period are also available to the user. Both the interactive load balancing procedure and the automatic work load balancing procedure produce similar reports on production schedules and work load distributions.

F. Development Tools

The databases used in the information management module were created using the dBASE III PLUS system. All the programs that are necessary to implement the system's menus and procedures for the information management system as well as the scheduling module were also written in dBASE III PLUS. The graphical information reports are produced using dGRAPH software package. The dGRAPH package is coupled with the dBASE programs internally; The interactions between these systems is transparent to the user. The automatic work load balancing procedures have been written in Microsoft Quick C in the MS-DOS environment. The individual modules are completely integrated so that the user need not be concerned about passing information between modules. Other than the initial data entry pertaining to products and sales order, the user interaction is primarily through menus.
In the following chapters the information management module and the automatic work load balancing module are discussed in detail. The discussion deals with the design, development and implementation of these systems. First, the underlying assumptions are enumerated followed by the design and development aspects of the system and the implementation and analysis of the results. A detailed description of the operating procedures and an example illustrating the use of the system are given in Adec Production Planning System User’s Manual (Pendakur, 1988).
3. INFORMATION MANAGEMENT SYSTEM

The information management system is composed of the database module and the scheduling module. The database module helps the user to maintain all the shop information necessary to generate the production schedules. The database module is driven by a menu-based user interface, and the operating procedure is highly user-oriented. This module is further divided into the product structure management component and the sales orders management component. In addition to aiding the user in maintaining the information, the database module also provides the user with a wide variety of report generation capabilities.

The scheduling module is designed to generate production schedules based on the due dates of the sales orders. All the orders are scheduled such that the due date constraints are satisfied. The scheduling is at the shop level; the shop level scheduling can then be used by the managers to plan the machine level details. This module also provides the user with a wide variety of tabulated as well as graphical reports. An additional component of this module is the interactive work load balancing procedure. This procedure provides a user interface to reschedule the orders from their initial production scheduled dates to some other acceptable date so as to achieve a balanced work load over the planning horizon, or to meet any other preferences specified by the user.

A. Main Menu

When the system is initialized, the main menu (Figure 4) will appear on the computer screen. The main menu has five options. These options give the user
access to various functions of the system. The selection of an option is made by entering the appropriate number shown on the main menu.

Figure 4 Main Menu of the System

The first option is the Product Structure Management function designed to help the user to create new product structures, purge obsolete product structures, change any of the existing product structures and to review and report the product structures. The information maintained by this subsystem is used by the scheduling module of the system to generate the initial production schedule. Therefore, it is important that this information be error free and current.

The second option in the main menu is the Sales Orders Management function of the system. Its function are similar to the product structure management module except that they focus on sales orders rather than products; specifically, it helps the user to create new sales orders, purge obsolete sales
orders, maintain sales orders information, update sales orders information and produce a number of reports on the sales orders information in the database. The information maintained in the databases is used as an input to the schedule generation process.

The third option in the main menu is the Scheduling Module of the system. This function makes use of the product structures and the sales order specifications to generate initial production schedules at the shop level. This initial schedule can be revised, and the work load balanced through the interactive load balancing routine. The production scheduling information is presented to the user in the form of reports and graphs.

The system has been designed to provide on-line help to the user, if needed. This can be invoked by selecting the fourth option in the main menu. The primary objective of this help function is to aid the user in understanding the terminology and functions of the system; operational details are explained in detail in the user's manual developed for the system [Pendakur, 1988].

The structure and workings of the individual modules of the system are explained below. The discussion is aimed to provide an understanding of the capabilities and scope of the individual modules.

B. Database Module

1. Product Structure Management

This component of the system has been designed to aid the user to maintain an exhaustive database of the structure of the products being manufactured in a facility. The product structures are based on the manufacturing process
sequences of the job. For A-dec Inc., the manufacturing process for any job is composed of four distinct steps. These are machining, assembly, fullup (completely assembled functional testing) and packaging. The database is designed so as to contain the manufacturing standard times information for each product in each of these four sections. Since a generic database design is not feasible, the databases have been tailored to the specific A-dec situation. The framework provided here can be used by any potential user by replacing the database module designed for A-dec with the databases specific to their situation.

An example of the product structure information contained in the product structure database is shown in Figure 5.

<table>
<thead>
<tr>
<th>PRODUCT ID NUMBER :</th>
<th>12-1212-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT NAME :</td>
<td>QW</td>
</tr>
<tr>
<td>STANDARD TIME IN MACHINING :</td>
<td>1.00</td>
</tr>
<tr>
<td>STANDARD TIME IN ASSEMBLY :</td>
<td>1.00</td>
</tr>
<tr>
<td>STANDARD TIME IN FULLUP :</td>
<td>1.00</td>
</tr>
<tr>
<td>STANDARD TIME IN PACKAGING :</td>
<td>2.50</td>
</tr>
<tr>
<td>TOTAL PROCESSING TIME :</td>
<td>5.50</td>
</tr>
</tbody>
</table>

**Figure 5** An Example Product Structure

When the product structure option is chosen from the main menu, the product structure management menu is displayed (Figure 6).
The options in Figure 6 enable the user to create, maintain and update the product structure information. The operating procedure for these tasks is user oriented and highly interactive. For illustration purposes, the procedure for adding a new product structure into the database is outlined below. To add a new product structure into the database, the user selects option 1 from the product structure management menu (Figure 6). The system provides the user with a full screen form shown in Figure 7.

The user enters the product identification number, product name and description, and standard times for processing the product in each of the production sections. The four production areas or sections in the case study considered are machining, where the individual components required to manufacture the product are produced; assembly, where the individual components are assembled; fullup, which refers to the full scale functional testing of the product; and packaging, where the product is packaged for shipping. The terminology used in the database is specific to A-dec.
At the start of each phase, an introduction screen explains the parameters to be specified in that phase. The editing capability of the system makes it possible to change a specific parameter without re-entering all the parameters. This is an important consideration in a system such as this as there is substantial data input to be made by the user.

The report generation option has built-in flexibility to generate different kinds of reports to meet the varying needs of the users. When the user starts up this task, the system will provide the user with the product structure reports menu (Figure 8). An example product structure report is shown in Figure 10.
2. Sales Orders Management

This part of the system is designed to aid the user to maintain relevant information about the sales orders received at the facility. The databases that hold the sales orders information are designed so as to contain the sales order number, priority, due date, product list and any other descriptions and comments that go with the sales orders/product list. All the sales orders received at the facility need to be stored in the system. The Sales Orders Management part of the system helps the user to add, delete, update and maintain the sales orders information.

Figure 8 Product Structure Reports Menu

Figure 9 Sales Orders Management Menu
### PRODUCT STRUCTURE REPORT

<table>
<thead>
<tr>
<th>PRODUCT ID#</th>
<th>MACHINING</th>
<th>ASSEMBLY</th>
<th>FULLUP</th>
<th>PACKAGING</th>
<th>TOTAL HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>444-444</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>8.00</td>
</tr>
<tr>
<td>44433</td>
<td>2.00</td>
<td>3.00</td>
<td>2.00</td>
<td>0.00</td>
<td>7.00</td>
</tr>
<tr>
<td>45-4545-45</td>
<td>1.00</td>
<td>2.50</td>
<td>0.00</td>
<td>0.00</td>
<td>3.50</td>
</tr>
<tr>
<td>45-4545-45/05</td>
<td>2.00</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
<td>6.00</td>
</tr>
<tr>
<td>48-4848-46</td>
<td>4.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>13.00</td>
</tr>
<tr>
<td>48-4848-49</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>8.00</td>
</tr>
<tr>
<td>49-4949-49</td>
<td>3.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>9.00</td>
</tr>
<tr>
<td>54-5454-54</td>
<td>2.00</td>
<td>3.00</td>
<td>2.00</td>
<td>2.00</td>
<td>9.00</td>
</tr>
<tr>
<td>55-0100-00</td>
<td>2.00</td>
<td>3.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.00</td>
</tr>
<tr>
<td>55-0108-00</td>
<td>2.50</td>
<td>2.50</td>
<td>0.00</td>
<td>0.00</td>
<td>5.00</td>
</tr>
<tr>
<td>55-0140-00</td>
<td>4.00</td>
<td>6.00</td>
<td>0.00</td>
<td>0.00</td>
<td>10.00</td>
</tr>
<tr>
<td>55-0150-00</td>
<td>7.00</td>
<td>7.40</td>
<td>0.00</td>
<td>0.00</td>
<td>14.40</td>
</tr>
<tr>
<td>55-0195-00</td>
<td>5.00</td>
<td>6.00</td>
<td>0.00</td>
<td>0.00</td>
<td>11.00</td>
</tr>
<tr>
<td>55-1195-00</td>
<td>5.00</td>
<td>6.00</td>
<td>0.00</td>
<td>0.00</td>
<td>11.00</td>
</tr>
<tr>
<td>55-1234-34</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>4.00</td>
</tr>
<tr>
<td>55-1234-67</td>
<td>2.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>5.00</td>
</tr>
<tr>
<td>55-55</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>8.00</td>
</tr>
<tr>
<td>55-5555-55</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>2.00</td>
<td>8.00</td>
</tr>
<tr>
<td>55-5555-55/01</td>
<td>2.00</td>
<td>1.00</td>
<td>1.00</td>
<td>2.00</td>
<td>6.00</td>
</tr>
<tr>
<td>55-5678-34</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>4.00</td>
</tr>
<tr>
<td>55-5678-88</td>
<td>1.00</td>
<td>2.00</td>
<td>1.00</td>
<td>0.00</td>
<td>4.00</td>
</tr>
<tr>
<td>55-7698-99/01</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>56-56</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>3.00</td>
<td>12.00</td>
</tr>
</tbody>
</table>

Figure 10 An Example Product Structure Report
The Menu for the Sales Orders Management function of the system is shown in Figure 9. For example, to add new sales order information into the database the user selects option 1 from the sales orders management menu. The system provides the user with the screen form shown in Figure 11, in which the relevant information for the sales order is entered.

<table>
<thead>
<tr>
<th>NEW SALES ORDERS ENTRY FORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sales Order Number [or 0 to Quit] : 112789-01</td>
</tr>
<tr>
<td>Priority : A2</td>
</tr>
<tr>
<td>Internal (y/n) : F</td>
</tr>
<tr>
<td>Product Id Number : 12-1212-12</td>
</tr>
<tr>
<td>Comments : spec. 34</td>
</tr>
</tbody>
</table>

Is the above information correct? (y/n)

**Figure 11** Adding a New Sales Order into the System

Besides the sales order number, other characteristics of the sales order that are required are the manufacturing due date, priority, any comments that go with the sales order itself and whether the sales order is an "internal" order. Priorities are assigned to the sales orders depending on whether the item is being produced for shelf stock or to a customer order. An "internal" order means that the order came from within the organization. These orders are not shipped out, but are sent to the department within the organization from which the order originated. Additionally, the products that are to be manufactured under a sales order are also required in the database.
3. Report Generation

This option helps the user to print out reports which are useful for auditing purposes. When this option is selected, the system will provide the user with the sales orders reports menu (Figure 12). The options provide the user with two different kinds of reports. The first option helps to print out sales orders information on all the sales orders from the sales orders database. The second option helps the user to print out sales orders information based on the due dates of the orders. For example, the user can obtain a report on all sales orders that are due by a certain date.

<table>
<thead>
<tr>
<th>SALES ORDERS REPORTS MENU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - Exit</td>
</tr>
<tr>
<td>1 - Report on all Sales Orders</td>
</tr>
<tr>
<td>2 - Report Sales Orders based on a Duedate</td>
</tr>
</tbody>
</table>

Figure 12  Sales Orders Reports Menu

4. Design of the Databases

The databases used in the system are implemented using dBASE III PLUS system. The databases are relational in structure and are maintained in the form of two-dimensional tables. Each of these tables has a primary key which helps the user to search, sort or arrange the data in the tables. The primary
key usually is a unique attribute for each record (a row in the table). This ensures that each record, when accessed on the primary key, is unique. The advantages of this relational database design is that the integrity of the database is easily maintained; it provides easy accessibility to the data in the records of the database; and fast and efficient searching and sorting is made possible. The structure of the databases used by the database module of the system are explained below.

Product Structure Database:

The structure of the product database is shown in Figure 13. The fields of the database contain the necessary information about the product name, identification number and the standard times for processing the product in the four sections of the facility. The product identification number (Product_Id in Figure 13) is used as the primary key for this database.

Sales Order Database:

The sales orders database structure is shown in Figure 14. The field Sales_Num contains the sales order number and is used as the primary key in this database. The other fields contain information on the products to be manufactured under this order and their quantities, due dates, and any comments about the products that might be useful for scheduling/production purposes.

Sales Order Priorities Database:

The sales order priority database structure is shown in Figure 15. This database contains the sales order priority information along with due dates and any special comments that go with the sales orders. Here again, the sales order number is used as the primary key to the database.
Information for both the sales orders database and the priorities database is entered by the user through the sales orders management menu (Figure 10). However, internally this information is stored in two different databases to avoid duplication of information in the databases.

Initial Schedule Database:

This database is designed to hold the scheduled production hours information along with the production capacity information on a weekly basis for each of the four sections. The information in this database comes from the scheduling module. The initial schedule generation process inserts the planned production hours information into this database. The capacity information is entered by the
<table>
<thead>
<tr>
<th>Field</th>
<th>Field Name</th>
<th>Type</th>
<th>Width</th>
<th>Decimals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sales_Num</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Proct_Id</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Quantity</td>
<td>Numeric</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Due_Date</td>
<td>Date</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Comments</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 14** Sales Orders Database Design

<table>
<thead>
<tr>
<th>Field</th>
<th>Field Name</th>
<th>Type</th>
<th>Width</th>
<th>Decimals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sales_Num</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Priority</td>
<td>Character</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Internal</td>
<td>Logical</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Due_Date</td>
<td>Date</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Comments</td>
<td>Character</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 15** Sales Order Priority Database Design

user in the scheduling module and is updated before generating the scheduling information reports. This database structure is shown in Figure 16. The field
WeekNumber is used as the primary key to the database. All the scheduled production times are recorded in hours.

<table>
<thead>
<tr>
<th>Field</th>
<th>Field Name</th>
<th>Type</th>
<th>Width</th>
<th>Decimals</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WeekNumber</td>
<td>Date</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Section</td>
<td>Character</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Hours</td>
<td>Numeric</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Month</td>
<td>Character</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Capacity</td>
<td>Numeric</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 16 Initial Schedule Database Design*

The report generation is made possible by combining information from these various databases. The databases are designed so as to avoid redundancy, and to increase efficiency of search operations.

C. Scheduling Module

This part of the system is designed to produce scheduling information on a weekly basis. This procedure makes use of the information in the product structure and the sales orders databases. This option produces reports and graphs about the loading patterns in the machining and assembly sections of the facility. The scheduled production hours obtained can be used for
effective planning of manpower and equipment. This information also assists in maintaining a steady capacity utilization.

The scheduling information is obtained by scheduling the sales orders based on their due dates. Any sales order coming into the facility goes through machining, assembly, fullup and packaging before being shipped out. Based on current A-dec practices, for macro-level planning at the shop level, assembly, fullup and packaging are combined into one section, referred to here as assembly. Thus, scheduling information is reported and depicted on graphs for machining and assembly.

1. Schedule Generation

The job-shop situation is assumed to follow a discrete-time process. This means that the movement of the jobs from one section of the facility to another can only occur at the start or at the end of a time period. The schedules are generated to satisfy the due date constraints on the sales orders. Shop level scheduling rather than machine level scheduling is the focus of this research.

The objectives here are: (1) to generate production schedules so that the due date constraints on the sales orders are satisfied, and (2) to provide scheduled production hours information on a weekly basis for each section of the facility. The capacity information entered by the user is not used as a constraint; instead the capacity information is used to provide the user with a graphical comparison of the capacity and the scheduled production hours.

The inputs to the scheduling module are the sales orders and product structures information. The scheduling module extracts the necessary information
from the database module. This is done internally and is completely transparent to the user. The schedule generation process assumes that the relevant product structure and sales orders information is available in the databases. If such information is not available, the system will request the information from the user, or the scheduling process will terminate.

The outputs from the scheduling module are the tabulated and graphical scheduled production hours information reports. The report generation facility of the system has many options that satisfy the different user needs. The information is formatted such that it can be used directly for shop floor planning and for tracking production activities.

a. Report Generation

The scheduling information needed by the user can be obtained from the scheduling information reports menu (Figure 17). The menu has a wide variety of options to suit the varying needs of the user. All the reports produced from this menu have detailed information about the production schedules for a requested time horizon on a weekly basis for each section of the facility.

<table>
<thead>
<tr>
<th>SCHEDULING INFORMATION REPORTS MENU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - Exit</td>
</tr>
<tr>
<td>1 - Report between two Dates</td>
</tr>
<tr>
<td>2 - Report on a particular Week</td>
</tr>
<tr>
<td>3 - Report on all Planned Weeks ahead</td>
</tr>
<tr>
<td>4 - Report on all Weeks this Year</td>
</tr>
<tr>
<td>5 - GRAPHS</td>
</tr>
</tbody>
</table>

**Figure 17** Scheduling Information Reports Menu
Each of the options in the scheduling information reports menu provides the user with five types of reports (Figure 18).

Select the type of Report you need:

1. Hours for Machining and Assembly
2. Hours for Machining only
3. Hours for Assembly only
4. Detailed Scheduling Information report for Machining
5. Detailed Scheduling Information report for Assembly

Enter Selection ( 1-5 OR 0 to Quit ) :

Figure 18 Types of Scheduling Information Reports

An example scheduling information report for assembly section is shown in Figure 19.

b. Graphical Scheduling Information Reports

The fifth option from the scheduling information reports menu (Figure 17) provides the user with graphical output. These graphs are used to make an initial assessment of the work load distribution, and to assess the initial schedule generated. The graphs also guide the user in determining whether there is a need for work load balancing on the initial schedule; and if there is, whether to use the interactive or automatic load balancing procedures. When this option is selected, the system will interface with the graphics program and display the graphics menu shown in Figure 20.
### ADEC DENTAL FURNITURE SHOP

#### ASSEMBLY SECTION - SCHEDULING INFORMATION REPORT

<table>
<thead>
<tr>
<th>MFG_DATE</th>
<th>DUE_DATE</th>
<th>PRTY</th>
<th>SALES_ORDER_#</th>
<th>PRODUCT_LIST</th>
<th>QTY</th>
<th>HRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>04/22/88</td>
<td>04/22/88</td>
<td>A2</td>
<td>2111</td>
<td>33-3333-33/02, 55-5555-55/01</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2</td>
<td>240</td>
<td>55-5555-55/01</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A4</td>
<td>1206</td>
<td>23-2323-23</td>
<td>20</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B3</td>
<td>1210</td>
<td>55-5555-55</td>
<td>15</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C3</td>
<td>1208</td>
<td>11-1111-11</td>
<td>20</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C3</td>
<td>23000</td>
<td>99-9999-99/09</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>465</td>
<td></td>
</tr>
<tr>
<td>04/29/88</td>
<td>04/29/88</td>
<td>A1</td>
<td>230</td>
<td>21-1212-12/12, 23-3434-34/23</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2</td>
<td>1211</td>
<td>12-1212-12</td>
<td>15</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2</td>
<td>1342</td>
<td>23-2323-23</td>
<td>12</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2</td>
<td>2002</td>
<td>33-3333-33/02, 55-5555-55/01, SSS!</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>G1</td>
<td>1223</td>
<td>45-4545-45</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>287</td>
<td></td>
</tr>
<tr>
<td>05/13/88</td>
<td>05/13/88</td>
<td>A1</td>
<td>134</td>
<td>12-1212-12/12</td>
<td>12</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A1</td>
<td>26</td>
<td>44-4444-44, testing</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48-4848-48, 54-5454-54</td>
<td>4</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>55-5555-55/01, testing</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>67-6767-67, 76-7676-76</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>89-8989-99, spl</td>
<td>4</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2</td>
<td>25</td>
<td>23-2323-23</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23-2323-23/23, 33-3333-33</td>
<td>15</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>total:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>495</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 19** An Example Scheduling Information Report
The graphic options produce bar graphs for the reports requested by the user. The first two options produce bar graphs between two dates specified by the user for the machining and assembly sections, respectively. An example of a bar graph for the machining section is shown in Figure 21. The third option produces bar graphs for both the machining and assembly sections on a monthly basis. The fourth option is designed to provide bar graphs on a quarterly basis; similarly, the fifth option produces graphs on a semi-annual basis. The last option produces two separate graphs by splitting the computer screen; these two graphs show the machining and assembly section scheduled loading.

2. Interactive Load Balancing

This component of the system is designed to provide the user with an interface to interactively balance or make changes to the initial schedule generated by the system. The initial scheduled load generated by the system is obtained by scheduling the orders into machining and assembly shops so as to meet all the due dates. Since no limit is placed on the production time in each period, the resulting work load could be unevenly distributed over the planning horizon. This may result in overtime work to be scheduled in some of the weeks.
Figure 21 An Example Bar Graph for Machining Section

The interactive load balancing procedure provides a mechanism by which the user can interactively request the rescheduling of orders over some specified time horizon to meet the user's objectives of overtime and load balancing. The flow chart shown in Figure 22 summarizes the inputs, procedure and outputs for the interactive load balancing procedure.

The implications of rescheduling have to be considered and realized by the user before requesting the rescheduling of orders for load balancing purposes. The system is designed to allow the user to perform 'what if' analysis. The user can investigate the consequences of rescheduling on due dates and work loads before actually incorporating the changes in the schedule. This flexibility built
START

Generate Initial Schedule

Activate Work Load Balancing Menu

Balance Work Load Interactively?

Yes

Make Rescheduling Requests

Reschedule the requested jobs

More Requests?

Yes

No

Retain Initial Schedule
Discard new Schedule

Evaluate the new Schedule

Finalise Schedule?

Yes

No

Discard Initial Schedule
Retain New Schedule

STOP

Figure 22 Flow Diagram for the Interactive Load Balancing Procedure
into the system is very important because it is not easy to foresee the implications of every rescheduling request.

The following chapter presents an alternative approach to the interactive work load balancing procedure. The chapter will discuss procedures and implementation issues for automatic work load balancing.
4. AUTOMATIC WORK LOAD BALANCING MODULE

As described in the previous chapters, the scheduling module uses a procedure which generates the initial schedules to specifically satisfy the due date constraints. However, this resulting schedule may result in an unbalanced load distribution over the planning horizon. This chapter deals with the development of some mechanisms to deal with this limitation of the initial schedule. The proposed methods may be applied to the initial schedule so as to refine and smooth the initial schedule while reducing the total cost of the schedule.

The initial schedule has a single objective, that of meeting due date constraints. It does not take capacity limitations into consideration. This may result in two consequences: (1) the schedule may result in overtime costs to an extent that they are undesirable, and (2) the schedule may have a high variance in its load distribution over the planning horizon. If the objective is to meet the due dates at any cost, the initial schedule provides a solution. But this may not be desirable in many situations.

Scheduling is generally a multi-objective problem with objectives that are conflicting in nature. The problem then is to develop a methodology that can provide good workable solutions yet be able to account for the constraints imposed by the conflicting objectives. Traditional optimization techniques may not be able to provide a solution in a feasible amount of time. Also, optimization is not required in most cases; consistent and good results in most cases satisfy the needs of the user. To be pragmatic, a solution procedure needs to go through a large search space in a reasonable amount of computing time to provide an acceptable solution. The initial schedule provides a starting point for this
procedure. A heuristic procedure would be relatively efficient as compared to an optimization approach in guiding through the search process.

The idea behind refining the initial schedule then is to minimize the cumulative effect of variation of the work load overtime, overtime costs, due date violation costs and finished goods storage costs, and to reasonably satisfy the capacity constraints. This schedule will result in low variance work load distribution, and hence a relatively constant utilization of the facility.

The interactive load balancing procedure discussed earlier is one way of refining the initial schedule to the satisfaction of the scheduler/manager. This procedure depends to a large extent on the intuition of the scheduler. There are limitations to this option because the user has to be fully aware of the implications of each of the rescheduling requests that are made. This does not seem to be a very practical way to solve the problem, particularly if the system involves a large number of jobs that need to be rescheduled. Hence the need for an automatic load balancing algorithm. The user may still fine tune the schedule resulting from the automatic load balancing algorithm. However, at this point, the task would most probably be a minor one since the automatic method would have considerably improved the initial schedule.

The automated work load balancing module developed here is integrated with other system components. The user interface to the automated work load balancing system is also completely menu-driven. Once the user has developed the initial schedule, the system will display the load balancing menu as shown in Figure 23. The menu is designed so that the user can select the interactive load balancing procedure or the automatic load balancing procedure depending
on their preferences. This also facilitates comparison of the results obtained by
the automatic load balancing methods with the results of the user developed
schedule through interactive load balancing.

![Work Load Balancing Menu](image)

**Figure 23 Work Load Balancing Menu**

The menu for automatic load balancing procedures is displayed on the
computer screen when this option is chosen (Figure 24). The automatic load
balancing module processes the initial schedule and displays the summary results
obtained from the heuristic models (Figure 32). The 'heuristic models' are
explained below. The flow diagram shown in Figure 25 shows the process of
automatic load balancing. The system has also been designed to provide detailed
scheduling information reports once the user has finalized on a schedule.
A. Design of Automated Work Load Balancing Models

The models developed in this section are heuristic in nature; they yield good workable solutions most of the time but do not attempt to provide the user with any kind of optimal solutions. The following discussion is organized to take the reader through the developmental process of the models. The underlying assumptions and scope of the models are enumerated and the heuristic mechanisms and policies used in the heuristic load balancing models are established. The structure and workings of the resultant models are discussed with flow diagrams.

1. Assumptions

The underlying assumptions in the development of the heuristic models can be grouped into two categories: operational and heuristic mechanism design. The operational assumptions help to characterize the facility and the operating policies of the organization. The heuristic mechanism design
Figure 25 Automatic Work Load Balancing Process Flow
assumptions determine the structure, operation and the stopping criteria used in the mechanisms.

a. Operational Assumptions

The analyses are based on discrete-time models of a job-shop situation. The models are built with the assumption that there is a discrete time period that determines the transitions within the job-shop. The movement of the jobs from one section to another are assumed to take place only at the start (or alternatively at the end) of a time period. Also the new job arrivals to the shop are assumed to occur only at the start (or at the end) of a time period. These assumptions mean that if a job is completed during a time period at its work center, it can only move to the next work center at the start of the next time period. The implication is that the time period chosen has to be meaningful to make the models realistic and useful. Typically this time is arrived at by considering the policies, procedures and work methods of the facility being analyzed. For illustrative purposes, a time period of one week is used.

The models developed here deal with a single stage discrete parts manufacturing process. The focus of the load balancing procedures is to balance the work load at a macro level rather than a micro level. Thus, the heuristics are designed to balance the loading across the manufacturing units rather than across machines in a manufacturing unit.

b. Heuristic Mechanism Design Assumptions

The search procedures are designed to reschedule only one job in one
iteration. The procedure terminates after 20 iterations have been performed. Load balancing is achieved by rescheduling jobs only within the planning horizon described by the user. The processing times are assumed to be the standard times required to produce a product through a section (for example, machine shop).

It is also assumed that finishing jobs earlier than their due dates (early jobs) is preferable to late jobs. Hence, the procedures first try to reschedule a job to an earlier time period from the specified due date; only if this does not work is an attempt made to reschedule a job to a later time period. This ensures that the late jobs are minimized and the slack available in the earlier weeks is utilized effectively. The procedures are also designed to find the 'nearest neighbor' time period from the job's initial scheduled time period. This ensures that the deviation from the due dates for the jobs is minimized.

2. Heuristic Mechanisms

There are three heuristic mechanisms. The objectives to be "optimized" define the iterative process of the heuristic mechanisms.

Heuristic Mechanism 1

The objective of this mechanism is to minimize the total slack (TS) value around a desired capacity utilization level (DCUL) for the planning horizon. The total slack can be represented mathematically as:

\[
TS = \sum_{i=1}^{N} (S_i)
\]

\[
S_i = (DCUL_i - AL_i)
\]
$S_i$ is positive or the negative slack for time period $i$,

$A_i$ is the actual scheduled load for period $i$,

$N$ is the number of periods in the planning horizon, and

DCUL is Desired Capacity Utilization Level for the period $i$. Note that DCUL represents a percentage of the full capacity of the shop.

Figure 26 illustrates the concept of total slack. As shown in Figure 26, there are weeks (periods) in which the loading is either less than or greater than the desired capacity utilization level. The positive slacks (or under utilization) occur when the actual loading is less than DCUL whereas negative slacks (or overtime) occurs when the actual loading is greater than DCUL. The sum of positive slacks and the absolute values of the negative slacks results in the total slack.

The objective is to minimize the value of this total slack from one iteration to the next until the improvement in the objective function is less than a specified threshold limit.

Heuristic Mechanism 2

The objective of this mechanism is to minimize the variance of the work load distribution around the mean work load (MWL) over the planning horizon. This mechanism does not require the use of the desired capacity levels in its objective function; hence, the capacity limitations are ignored when using this mechanism.

Variance is mathematically defined as:
Total Slack $= \sum_{i=1}^{5} (S_i)$

$S_i$ is positive or negative slack for time period $i$.

Legend:
- Positive Slack (Actual Load less than Capacity)
- Negative Slack (Overtime)

Figure 26 Concept of Total Slack
$$Variance = \frac{\sum_{i=1}^{n}(AL_i - MWL_i)}{(N - 1)}$$

where

$AL_i$ is actual loading of the shop in period i,

$MWL_i$ is the Mean Work Load over the planning horizon, and

$N$ is the total number of time periods in the planning horizon.

The variance is a simple way to measure the spread of the work load distribution over the planning horizon. This is used as an objective because it is a good indicator of how well the work load is balanced over the planning horizon.

Heuristic Mechanism 3

The objective of this mechanism is to minimize the total slack around the desired capacity utilization Level while incorporating the concept of a 'planning window' within a planning horizon. Figure 27 illustrates the concept of the planning window by using a planning window of size three weeks. Each planning window is defined to be $\pm k$ time periods from the initial scheduled date of a job.

The jobs are rescheduled only within their planning windows. The total slack value is minimized within each planning window.
Planning Window for jobs in week 2
week 2 +/− 1 week

Figure 27 Concept of a Planning Window
3. Rescheduling Policies

There are two rescheduling policies used in this research work. The rescheduling policies define the order in which jobs are selected for rescheduling. The first policy (MinJob policy) is to select the job with the minimum processing time from the week with the maximum loading, and reschedule it to a feasible slack week (a 'nearest neighbor' slack week that can accept the job). The second policy (BestJob policy) is to identify the week with the maximum loading and the feasible week (feasible week is identified as the week with the most slack available in the planning horizon) and then select the job that results in the best improvement of the objective function of the model. For example, if the objective function is to minimize the total slack, the job selected should be the one that results in the greatest total slack reduction after the rescheduling action.

4. Resultant Load Balancing Models

The heuristic mechanisms and the rescheduling policies are combined to form the load balancing models shown in Figure 28. Models 1 through 3 incorporate the 'MinJob' policy; models 4 through 6 are based on the 'BestJob' policy. To clarify, the terminology used in describing the models is shown in Figure 29.
## Heuristic Mechanisms

<table>
<thead>
<tr>
<th>Rescheduling Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinJob Policy</td>
</tr>
<tr>
<td>BestJob Policy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cf Total Slack over the Planning Horizon</td>
<td>cf Variance around Mean Work Load</td>
<td>cf Total Slack within the Planning Windows</td>
</tr>
<tr>
<td>Model 1</td>
<td>Model 2</td>
<td>Model 3</td>
</tr>
<tr>
<td>Model 4</td>
<td>Model 5</td>
<td>Model 6</td>
</tr>
</tbody>
</table>

**Figure 28 Experimental Design (Resultant Models)**

**Model 1**: Minimize Total Slack using MinJob Policy

The flow diagram in Figure 30 shows the working of Model 1. The objective is to minimize the total slack of the schedule over the planning horizon utilizing the 'MinJob' rescheduling policy.

The process starts by identifying the weeks with the overtime hours; of these weeks, the week with the maximum overtime (or over-utilization) hours is
Legend:

- A job of size 't' hours
- Positive Slack available

MinJob: Job with the smallest processing time from MaxWeek

MaxWeek: Week in the planning horizon with most overtime hours

NearSlackWeek: The slack week 'closest' to MaxWeek and has Positive Slack greater than or equal to MinJob

DCUL: Desired Capacity Utilization Level - a percentage of the full capacity of the shop

Figure 29 Terminology Used in the Models
Identify Week with Maximum Overtime Hrs. (MaxWeek)

Identify Job with Minimum Processing time from MaxWeek (MinJob)

Compute Total Slack value for the Planning Horizon (TS)

Identify Slack Weeks (Weeks with Actual Load less than DCUL)

Number of Slack Weeks > 0?

Select Slack Week closest to due date that can accept MinJob (NearSlackWeek)

NearSlackWeek Exists?

STOP

Figure 30 Flow Diagram of the Load Balancing Model 1
Figure 30 continued.
Figure 30 continued.

Reschedule MinJob from MaxWeek to NearSlackWeek
selected as the 'MaxWeek'. To reschedule a job from this MaxWeek to another week, the procedure first identifies the job that can be rescheduled. Since this model is based on the MinJob policy, the job with the minimum processing time is selected from the MaxWeek as the MinJob. The procedure next identifies the slack week that can accommodate the MinJob. Among the slack weeks, the procedure identifies the week that is nearest, and if possible, earlier than the due date of the MinJob (NearSlackWeek). Once the NearSlackWeek is identified, the MinJob is moved to this week, and the procedure enters the next iteration. If none of the slack weeks can accommodate the MinJob without resulting in overtime, then the week with the maximum slack (MaxSlackWeek) is identified. If there is an improvement (reduction) in the total slack on moving the MinJob to the MaxSlackWeek, then the MinJob is moved to the MaxSlackWeek. Otherwise, the current MaxWeek is taken out of consideration, and the week with the next highest overtime hours is identified as the new MaxWeek. The process continues until there is no significant improvement in the total slack value between two consecutive iterations or twenty iterations have been performed.

The flow diagrams for the other five models (Model 2 through Model 6) are given in Appendix D.

5. Performance Measures

The performance measures used for analyzing the heuristic load balancing models are total production costs, number of jobs rescheduled, the average distance moved per job, the percentage of variance reduction, number of late jobs and the number of early jobs.
Total Production Cost

This performance measure represents the total production costs incurred by the facility on implementing a schedule. The objective is to minimize the total cost, which is the sum of the production costs, storage costs and due date violation costs. This objective can be represented as a composite cost expression:

\[ TC = W_1 \times OTC + W_2 \times SC + W_3 \times DDVC \]

where

\( TC = \) Total Cost of a schedule for the requested planning horizon,

\( SC = \) Storage Costs,

\( OTC = \) Overtime Costs,

\( DDVC = \) Due Date Violation Costs, and

\( W_1, W_2, W_3 \) are weights applied to the individual cost components representing their relative contribution to the cost factor, and

\( W_1, W_2, W_3 \geq 0, \) and

\( W_1 + W_2 + W_3 = 1.0 \)

The total cost expression is an integrated way of measuring the relative merit of each model's output. The lower the total cost of a schedule, the more desirable
is that schedule. The models refine the initial schedule and output total cost of a schedule so as to aid the user in the decision making process. The individual components of the total cost expression are briefly discussed below.

Overtime Costs

These costs are incurred when the production scheduled as per a schedule cannot be completed within the capacity limitations of the shop. For example, if a schedule calls for 500 production hours to be planned for a week, and the available capacity in that week is only 450 hours, then production can only be met by incurring an overtime of 50 hours. Typically, the hourly labor cost for overtime is much higher than the regular production rate. For example, at A-dec Inc., the overtime costs are 1.5 times the regular production costs. Additionally, overtime production may imply higher overheads in support and supervisory staff, and in utilities. Thus, the overtime costs are an important component of the total cost figure. The weight, $W_1$, assigned to this component is dependent on the industrial environment, policies, shop conditions, management preferences, and other such factors.

Storage Costs

These costs result due to storage of finished goods before they are shipped out, or work-in-process between production operations. These costs have to be accounted for if products are produced ahead of time to meet a future demand or to account for uncertain fluctuations in demand. Again, these costs are industry and product dependent. The nature and value of these costs vary significantly. For A-dec application, the costs of finished goods storage was
quite important because the products were bulky and due to their structural limitations they could not be stacked. This meant that either there were little or no inventory, large storage space, or investment in some sort of scaffolding and material handling equipment.

Due Date Violation Costs

Due date violation costs are costs incurred due to violation of shipping dates to customers. These costs include the cost due to loss of future business with the customer and loss of reputation. With an emphasis in manufacturing on high quality and customer satisfaction, meeting due dates becomes critical. According to estimates by managers at A-dec, the due date violation costs were approximately three times as important as storage costs.

Number of Jobs Rescheduled

This represents the total number of jobs rescheduled from their initial scheduled production time during work load balancing. This measure is an indicator of the working efficiency of the load balancing procedure used.

Number of Early Jobs

Early jobs are jobs which are produced earlier than their due date and have to be stored in the facility until they can be shipped to the customers. Thus, they represent finished goods inventory costs. The costs involved with early jobs vary depending on the type of industry and its policies. The cost associated with early jobs is also considered in the total cost figure as storage costs.
Number of Late Jobs

Late jobs are jobs which cannot be completed by the requested due date. These may result in loss of future business to the company, loss of reputation, canceled orders, etc. In the total cost figure, the late jobs are represented by the due date violation costs.

Average Distance Moved per Job

The distance measure refers to the difference in the time period between the original scheduled date and the rescheduled date for a job. The average distance moved per job is calculated by dividing the total distance for all jobs rescheduled by the number of jobs that are rescheduled.

6. Data For Model Comparison

A total of six data sets were generated for comparison and analysis of the models. The shop load levels were generated as random variables from a normal distribution with a mean of 250 hours and a standard deviation of 80 hours. The jobs to be processed over the planning horizon were also generated as random variables from a normal distribution with a mean of 20 minutes and a standard deviation of 5 minutes. The shop load levels are projected scheduled loading for the shop on a weekly basis. The procedure followed in generating the data is shown in the flow diagram in Figure 31.
START

TotalLoad = 0.0

Generate Load for the Week Wi
Wi = N(250.0, 80.0)

Generate Jobs
Job = N(20.5)

Assign Job to Week Wi
TotalLoad = (TotalLoad + Job)
Balance = (Wi - TotalLoad)

Is Balance > 0.0 ?

Yes

More Weeks to Plan ?

STOP

No

Figure 31 Flow Diagram for Data Generation
7. Results and Interpretation

The six models were evaluated using the data sets generated as described above. An example summary output from the automatic load balancing module is shown in Figure 32.

<table>
<thead>
<tr>
<th>Mod Num</th>
<th>Final Overtime</th>
<th>Final Variance</th>
<th>jobs resched.</th>
<th>Early Jobs</th>
<th>Late Jobs</th>
<th>Average move/job</th>
<th>% Variance Reduction</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000</td>
<td>2089.179</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>1.333</td>
<td>51.04%</td>
<td>46.91</td>
</tr>
<tr>
<td>2</td>
<td>0.000</td>
<td>28.139</td>
<td>20</td>
<td>9</td>
<td>5</td>
<td>1.400</td>
<td>99.34%</td>
<td>109.34</td>
</tr>
<tr>
<td>3</td>
<td>0.000</td>
<td>2089.179</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>1.333</td>
<td>51.04%</td>
<td>46.91</td>
</tr>
<tr>
<td>4</td>
<td>0.000</td>
<td>226.872</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3.000</td>
<td>94.70%</td>
<td>117.10</td>
</tr>
<tr>
<td>5</td>
<td>0.000</td>
<td>18.309</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>3.429</td>
<td>99.57%</td>
<td>179.00</td>
</tr>
<tr>
<td>6</td>
<td>0.000</td>
<td>1534.442</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1.200</td>
<td>64.03%</td>
<td>45.95</td>
</tr>
</tbody>
</table>

Figure 32 Automatic Work Load Balancing Procedures - Summary Output
This report summarizes the performance measures for the schedules generated by the models. The report gives information such as the total number of jobs in the schedules, the initial and final overtime hours, initial and final load variance, total number of jobs that were rescheduled, the number of early jobs, number of late jobs, the average distance moved per job, the variance reduction achieved by the models and the total cost estimate of implementing the schedule. The weights used in the total cost equation are based on the A-dec operating environment. These were: 0.30 for overtime work hours, 0.50 for due date violations, 0.20 for storage costs. As can be seen from the summary output, the schedule that achieves the lowest total cost is generated by model 6. Since the weights used in the cost expression are subjective in nature, consideration should be given to other performance measures as well. For example in Figure 4-11, even though the schedule from Model 6 results in the least total cost, it is not the best schedule if variance reduction is used as the performance criterion. A small reduction in variance means that the schedule still has a relatively high variance, and did not result in a well-balanced production load in terms of the work load. This is also illustrated graphically in Figure 33.

Figure 33 shows the graphical comparisons of the initial schedule and the final schedule resulting from the six models. As can be seen from the graphs, there is a definite improvement in all of the six schedules. The initial overtime of 90.7 hours has been eliminated in all six schedules. The total elimination of overtime in this particular case is a result of the random data that was generated and the planned capacity levels used. In general, there is no guarantee that any of the models will reduce the overtime to zero. The load variance was also reduced for all six schedules, though the amount of reduction varies considerably. Models 2 and 5 result in maximum variance reduction The reason being that these models
Figure 33 a & b. Graphical Comparison of Models 1 & 2
Figure 33 c & d. Graphical Comparison of Models 3 & 4
Figure 33 e & f. Graphical Comparison of Models 5 & 6
use reduction of load variance around the mean work load as the criterion for work load balancing.

The number of jobs rescheduled during the load balancing process also depends on the heuristic mechanism incorporated by the models. For example, Models 4, 5 and 6 using the BestJob rescheduling policy, result in fewer rescheduling actions as compared to Models 1, 2 and 3. A higher number of rescheduling actions does not necessarily mean higher due date violations. This is because over the course of many iterations, some jobs may be rescheduled more than once, that is a job moved to a later week may again be rescheduled to a different week during the iterative process. Thus in the summary report (Figure 32) the sum of early and late jobs does not necessarily equal the total jobs rescheduled (for example, in Model 2).

The number of late jobs as a result of the load balancing procedures varies among the proposed schedules. Model 2 resulted in the five late jobs whereas Model 4 resulted in only two late jobs. This again may be a result of the rescheduling policy used. Generally, the BestJob policy seems to result in fewer due date violations because it does a better job at finding a match in rescheduling starting with 'prior' weeks. Another fact that the user needs to be aware is that the number of jobs does not necessarily reflect the job's actual work content. For example, the five late jobs from Model 2 may add up to a total work content of 56 hours whereas the two late jobs from Model 4 may constitute a total work content of 72 hours.

Another performance measure that is shown in Figure 32 is the number of early jobs. This might be important if there are constraints on early shipping or
on storage of finished goods. Generally, the schedules proposed by all the models tend to have more early jobs than late jobs. This is because the mechanisms are designed with the assumption that early jobs are preferable to late jobs. What this means is that the mechanisms try to make use of all the early slack times before creating any late jobs.

8. Analysis of the Heuristic Models

Statistical analysis of the model outputs (on the six data sets) was performed on all performance measures discussed above except the total cost. The reason for not including the total cost in the statistical analysis was the subjective nature of the weighting factors used in the cost equation. Each of the performance measure was analyzed using a two-way analysis of variance (ANOVA) to study the interaction of the two factors (heuristic mechanisms and the rescheduling policies). The data sets and the ANOVA analyses are shown in Appendices E-1 to E-6.

a. Two-Way ANOVA Results

The data was collected by running the six load balancing models with the six data sets. Subsequently, each model produced six sets of results corresponding to the six data sets. The ANOVA was conducted on each of the performance measures.

The ANOVA tests showed that there is insufficient evidence to suggest significant interaction between the two factors at a 99% confidence level (see Appendix F-2 and F-6) for all the performance measures. This means that the
choice of a model (combination of the mechanism and the rescheduling policy) does not have any significant effect on the performance measures. However, the main effects were significant for some of the performance measures. Specifically:

1. The number of jobs rescheduled are affected by the choice of rescheduling policies.

2. The average distance moved per job is significantly influenced by both the mechanism and the rescheduling policies.

3. Variance reduction is affected by the heuristic mechanism chosen.

There are two outcomes of this research: (1) a framework for an information management and schedule generation system, and (2) a set of heuristic methods for balancing work loads automatically. Hence, we have both empirical and implementation specific results. The framework developed in this research can be adapted to a number of situations by modifying the databases to reflect the specific situation.

The results of the analysis of the heuristic procedures did not show any conclusive evidence to believe that the choice of the model (combination of the heuristic mechanisms and the rescheduling policies) would affect the performance measures. However a number of other interesting observations can be made. The choice of the rescheduling policies affected the performance associated with the rescheduling actions. This implies that to improve the efficiency of the heuristic procedures, the BestJob policy is provides better results. The affect that is produced by rescheduling a large number of jobs using the MinJob policy can be
achieved with the BestJob policy in fewer number of iterations because of the selection mechanism used.

The choice of the heuristic mechanism also affected the amount of variance reduction achieved. This is intuitively obvious because two of the six models, Model 2 and 4, work to reduce variance to achieve load balance. The other four models indirectly reduce variance but their primary objective is to reduce the total slack of the load. These models terminate the procedure when the overtime has been minimized, and do not continue even if the variance of the load can be further reduced. Hence, a user who uses a well balanced load distribution as a primary criterion may prefer schedules generated by Models 2 and 4. However, it is best to examine and compare all of the performance measures of the schedules before selecting a specific one.

The choice of the heuristic mechanism also has an effect on the average distance moved per job during the rescheduling process. This is because of Models 3 and 6, which use the planning window concept. These models would probably achieve the lowest average distance moved per job because of the constraint to reschedule the jobs only within their planning windows. If the user's objective is to pick a schedule that has the minimum deviations from the due dates for early and late jobs, the schedules generated by Model 3 and 6 would be good choices. If Models 3 and 6 were left out of the analysis, the effect of heuristic mechanism on the average distance moved per job is insignificant.

b. Summary of Results

There are two outcomes of this research: (1) a framework for an information
management and schedule generation system, and (2) a set of heuristic methods for balancing work loads automatically. Hence, we have both empirical and implementation specific results. In the next chapter, conclusions about this research are drawn and some directions for future research are provided.
5. CONCLUSIONS

A. Summary of Accomplishments and Conclusions

The research and development work conducted during the course of this project has resulted in a practical and useful system. The system is user-friendly and has been designed so that a user with no prior computer background can learn to use it as an effective tool in a short time. The system has been implemented for use at A-dec. The managers and schedulers at A-dec were quite familiar with the system by the time its implementation was completed because of their continuous involvement in the evaluation of various prototypes. This was an advantage because the personnel at A-dec were able to use the system on a "real-time" basis immediately. A-dec has been using the system since November 1988. The feedback from the managers at A-dec indicates that the system is able to meet the needs of the company adequately.

The databases in the system facilitate streamlining and standardization of the information used on the shop floor. The shop level schedules generated by the system helps the user in developing detailed machine level schedules and also to assist in manpower planning. The graphical output of the load distributions for the work centers have enhanced the capacity planning process.

The automatic work load balancing procedures developed provide an effective way to refine the initial schedules and to balance the work load over the planning horizon. Selecting one of the six proposed schedules is a process that is heavily dependent on the type of industry, type of product, and the procedures and policies of the company. The framework provided in this work
can be utilized in a number of discrete part manufacturing situations by easily adapting the system to the specific situation.

A user's manual [Pendakur, 1988] has also been written for the system separate from this thesis. The manual has been designed to guide the user through the workings of the system and an example session is provided for user reference. The on-line help information, even though limited, gives the user a brief description of the capabilities of the system.

B. Directions for Future Research

1. Heuristic Methods for Multi-Stage Production Process

The heuristic methods for work load balancing developed in this work were designed for a single-stage production process. The applicability of these methods can be broadened by modifying the heuristic methods for a multi-stage process. The key issues to consider would be the transition of the jobs from one work center to another and the sequencing of the work stations and the batch sizes involved. If the transitions are at fixed intervals, the process of adapting these methods would be simpler.

2. Rescheduling Multiple Jobs per Rescheduling Action

The load balancing heuristics developed in this research used a strategy of rescheduling one job in an iteration. If a week with overtime hours has a large number of jobs that need to be rescheduled to reduce the overtime, it could take several iterations to achieve this. An alternative approach that can be explored is to identify a number of jobs that can be rescheduled in a single iteration. This
would probably make the heuristics more efficient by reducing the execution time of the procedures.

3. A Graphical User Interface

The user interfaces used for interactive load balancing in the current system relies on keyboard entry. The rescheduling requests are made by keyboard entry, and the effect of the rescheduling actions on the load distribution and the schedules is not displayed immediately. To see a graphical display of the changed schedule, the user has to request the appropriate graphical output.

An alternative would be to dynamically change the load distribution graphs and show them on the computer screen as the rescheduling requests are made. This will ease the load on user memory enhancing the decision making process. The graphical display can contain graphic objects that can be activated by a mouse to make rescheduling requests. This would reduce the keyboard input and would minimize the keyboard entry errors. If the user interfaces for the interactive load balancing module are enhanced by implementing this dynamic graphical display, the expert system discussed below could be a very powerful analysis tool.

4. Expert System Aid for Enhanced Decision Making

Interactive load balancing activity is a difficult task because of the demands on the user’s judgement and memory. The user has to keep track of the implications of the rescheduling requests. This may hinder the user’s decision making process. To aid the user in this evaluation process, an expert system can be developed that keeps the user informed of the system status and also makes
suggestions during the interactive load balancing activity. The system can also be made to be able to train the user to make good decisions. Additional functions that can be delegated to the expert system include analysis of alternative schedules (developed by both the interactive and automatic load balancing methods), rating of schedules, and sensitivity analysis by varying the weights assigned to the cost components. The databases used in this research along with information provided by the user can serve as knowledge bases for the expert system.

5. Development of an Improved Total Cost Expression

The total cost expression developed in this research work takes into account the cost of production, cost of inventory and the cost of due date violations. It does not directly consider the cost of load balance/unbalance and costs associated with underutilization of resources. A cost expression that included these components would be a more accurate representation of the system components.
BIBLIOGRAPHY


APPENDICES
APPENDIX A : Hardware and Software Requirements of the System

1. An IBM PC, XT, AT, COMPAQ or other 100% IBM PC Compatible Computer with a monochrome/color monitor with Hercules/CGA/EGA graphics adaptor.
2. A hard disk of at least 5M Bytes and 640K Bytes of RAM.
3. One or two floppy disk drive capability.
4. A printer with at least 80 columns capability.
5. dBASE III PLUS system disks, Adec Production Planning System disks 1, 2 & 3 and dGRAPH system disks 1 & 2.
APPENDIX B: dBASE III PLUS Source Code

*=================================================================
* PROGRAM: ADECMAIN.PRG
* This program brings up the main menu for the system and
* activates routines selected by the user.
*=================================================================

CLEAR ALL
SET TALK OFF
SET ECHO OFF
SET DEBUG OFF
SET BELL OFF
SET STATUS OFF
SET ESCAPE OFF
SET CONFIRM OFF

DO WHILE .T.
* INITIALIZE NEW VARIABLES
today=DATE()
selectnum=0
* We need to clear the screen and display the main menu.
CLEAR
22,2 TO 6,77 DOUBLE
24,3 SAY CHR(176)
24,76 SAY CHR(176)
23,3 SAY REPLICATE(CHR(177),74)
25,3 SAY REPLICATE(CHR(177),74)
27,2 TO 23,77 DOUBLE
24,9 SAY "ADEC PRODUCTION PLANNING SYSTEM"
10,10 SAY "1. Product Structure Management"
12,10 SAY "2. Sales Orders Management"
14,10 SAY "3. Schedule Generation"
16,10 SAY "4. Help"
18,10 SAY "0. Quit"
21,3 TO 21,76
22,18 SAY "Enter Selection ( 1-4 OR 0 to Quit ) : ":
11,53 SAY today
10,50 TO 12,64
10,51 SAY "Date"
15,50 TO 17,64
15,51 SAY "Time"
* This segment is to take care of the time display on the
* main menu screen.
DO WHILE .T.
i=0
DO WHILE i=0
i=INKEY()
@ 16,53 SAY TIME()
@ 22,58 SAY ""
IF CHR(i)$"01234"
EXIT
ELSE
ENDIF
i=0
ENDDO
@ 22,58 SAY UPPER(CHR(i))
EXIT
ENDDO
DO CASE
CASE CHR(i)$"0"
SET BELL ON
SET ESCAPE ON
Welcome to the ADEC Production Planning System. This system is designed to help you in managing the information for all products used or manufactured at ADEC and for helping managers to determine the gross production hours over a planning horizon.

When you start up the dBASE III PLUS SYSTEM and then the Adec Production Planning system, you will see the Adec Production Planning Main menu on the screen. The main menu has 5 options. These options give you access to various functions of the system. To quit the main menu, you need to enter selection 0. This will take you back to the Dot(.) prompt of the dBASE III PLUS SYSTEM. The other options are selected by typing in their respective selection numbers.

Note: Quitting from "any" menu in the Adec Production Planning System is by selecting 0. Remember not to press the "Enter" key after you make the selection because you need to press only the selection number to activate an option.
Management function of the system. This function helps you to create new products, purge obsolete products, change any of the existing products and to review and report on the product structures. The information maintained by this function of the system is used by the Scheduled Production Hours function of the system to produce Scheduling information. And hence, it is absolutely necessary that this information is error free and current.

The menu selections available with the Product Structure Management menu are given below.

1. Enter new Product information
2. Change Product information
3. Delete Product information
4. Review Product information
5. Reports

ENDTEXT

WAIT SPACE(20)="Press any key to continue"

CLEAR

TEXT

Sales Orders Management
----------------------

Selecting option "2" from the main menu starts up the Sales Order Management function of the system. This function helps you to create new sales orders, purge obsolete sales orders, maintain sales orders information and update sales orders information. The information maintained should be correct and current.

The options available under the Sales Orders Management menu are given below.

1. Add new Sales Order information
2. Change Sales Order information
3. Purge Sales Order information
4. Review Sales Order information
5. Reports

ENDTEXT

WAIT SPACE(20)="Press any key to continue"

CLEAR

TEXT

Scheduled Production Hours
-------------------------

Selecting option "3" from the main menu starts up the Scheduled Production Hours function of the system. This function makes use of the product structures and the sales order specifications to come up with the scheduling information based on the customer due dates. This initial schedule can be revised and the work load balanced through the interactive load balancing routine.

Once you have studied the initial schedule, you can balance the work loading obtained in the initial schedule to your satisfaction.
When you exit the Reports Menu, the system will give you an opportunity to balance the work load. If you decide not to balance the work load, the system will take you back to the Main Menu. The scheduling information is presented to you in the form of reports and graphs.

4. Selecting option "4" from the main menu brings up this help information available on-line. For a more detailed help information, you need to refer to the APPS USER'S MANUAL.

The user's manual is also available as Manual.txt file on the APPS diskette #5. To review/edit this file, you can use the PcWrite software provided (PcWrite files should be available on the Hard disk in DBASE directory). To load the file Manual.txt onto the computer and read/edit it, follow the commands given below at the DOS prompt "C>".

   type in CD\DBASE and ENTER
   Insert APPS diskette #3 in drive A.
   type in ED A:MANUAL.TXT and ENTER
   Press "Esc" to read/edit the file.
   Once you are finished reading/editing/ printing the file, exit PcWrite by typing in "F1" and "F2" keys.

** Press any key to return to Main Menu **
@ 15,42 GET choice
READ
ENDDO
@7+VAL(choice),22 SAY "**"
cnt=0
DO WHILE cnt<6
  IF cnt<>VAL(choice)
    @ 7+cnt,24 SAY SPACE(40)
  ENDIF
  cnt=cnt+1
ENDDO
DO CASE
  CASE choice = "0"
    CLEAR ALL
    RETURN
  CASE choice = "1"
    * DO ADD INFORMATION
    DO PSM1
  CASE choice = "2"
    * DO CHANGE INFORMATION
    DO PSM4
  CASE choice = "3"
    * DO REMOVE INFORMATION
    DO PSM2
  CASE choice = "4"
    * DO REVIEW INFORMATION
    DO PSM3
  CASE choice = "5"
    * DO PRINT REPORT

DO WHILE .T.
  answer=""
  CLEAR
  @1,1 TO 3,79 DOUBLE
  @4,1 TO 12,79 DOUBLE
  @2,25 SAY " Product Structure Report Menu "
  @6,25 SAY " 0. Exit"
  @8,25 SAY " 1. Product Structure Report"
  @10,25 SAY " 2. Product Description/Total hrs. Report"
  @12,36 SAY "Select | |"

DO WHILE .NOT. answer$"012"
  answer=""
  @ 12,44 GET answer
  READ
ENDDO
IF answer="0"
  EXIT
ENDIF
IF answer="1"
  @ 19,0 CLEAR
  WAIT SPACE(15)+"Get your Printer Ready and hit Return..."
  USE PRODUCTS INDEX PRODS
  REPORT FORM PMREPORT TO PRINT
  WAIT SPACE(15)+"Press any key to continue..."
  USE
ENDIF
IF answer="2"
  @ 19,0 CLEAR
  WAIT SPACE(15)+"Get your Printer Ready and hit Return..."
  USE PRODUCTS INDEX PRODS
  REPORT FORM PM1RPRT TO PRINT
WAIT SPACE(15)+"Press any key to continue..."
USE
ENDIF
ENDDO
ENDDO T
RETURN

* EOF: PSM.PRG

=======================================================
* PROGRAM : PSM1.PRG
* adding new products into product structure databases
*=======================================================
USE PRODUCTS INDEX PRODS
DO WHILE .T.
  * INITIALIZE ALL THE MEMORY VARIABLES
  mproduct_id=SPACE(15)
  mprodctname=SPACE(15)
  mstime_mc=0.00
  mstime_ass=0.00
  mstime_flp=0.00
  mstime_pckg=0.00
  mtotal_hrs=0.00
  monk=.T.
  DO WHILE monk
    DO WHILE .T.
      CLEAR
      DO PSM11
        @7,44 GET mproduct_id PICTURE "!!!!!!!!!!!!!!" READ
      * Test for exit condition
      IF mproduct_id =SPACE(15) .OR. mproduct_id="0" .OR. mproduct_id="- - "
        WAIT SPACE(20)+"Press any key to return to menu"
        CLOSE DATABASES
        RETURN
      ENDF
      STORE LTRIM(mproduct_id) TO mproduct_id
      IF mproduct_id =SPACE(15) .OR. mproduct_id="0" .OR. LEN(mproduct_id) = 0
        WAIT SPACE(20)+"Press any key to return to menu"
        CLOSE DATABASES
        RETURN
      ENDF
     * Check for duplicate ID number
     SEEK mproduct_id
     IF FOUND()
       @ 22,15 SAY " PRODUCT ID already exists, give new ID. reenter"
       WAIT SPACE(27)+"Press any key to reenter"
     ELSE
       IF EOF()
         EXIT
       ENDIF
     ENDF
     ENDDO
     @ 22,0 CLEAR
     * without a partname, a record is not complete
     DO WHILE .T.
       @9,29 GET mprodctname PICTURE "!!!!!!!!!!!!!!!"
READ
IF mprodctname <> SPACE(15)
  EXIT
ENDIF
@ 21,25 SAY " Please give a product name"
ENDDO
@21,0 CLEAR
@14,32 GET mstime_mc PICTURE "9999999.99" RANGE 0,
@15,32 GET mstime_ass PICTURE "9999999.99" RANGE 0,
@16,32 GET mstime_flp PICTURE "9999999.99" RANGE 0,
@17,32 GET mstime_pckg PICTURE "9999999.99" RANGE 0,
READ
mtotal_hrs=(mstime_mc+mstime_ass+mstime_flp+mstime_pckg)
@19,32 SAY mtotal_hrs PICTURE "9999999.99"

*CHECK TO SEE IF INFORMATION IS CORRECT

answer=""
DO WHILE .NOT. answer$"YyNn"
  answer = " "
  @ 21,10 SAY " Is the above information correct ? (y/n)";
  GET answer
  READ
ENDDO

@ 21,10 CLEAR
* if answer is yes we need to exit loop
* otherwise we need to reenter the data
IF answer$"Yy"
  monk=.F.
ELSE
  @ 21,25 SAY " Please type in correct data 
  WAIT SPACE(27)+"Press any key to reenter"
ENDIF
ENDDO monk

IF UPPER(answer)="Y"
* the data is correct so we need to add to the database
APPEND BLANK

  REPLACE PRODUCT_ID WITH UPPER(mproduct_id)
  REPLACE PRODCTNAME WITH UPPER(mprodctname)
  REPLACE STIME_MC WITH mstime_mc
  REPLACE STIME_ASS WITH mstime_ass
  REPLACE STIME_FLP WITH mstime_flp
  REPLACE STIME_PCKG WITH mstime_pckg
  REPLACE TOTAL_HRS WITH mtotal_hrs
ENDIF
* option to enter or not enter another product info.

answer=""
DO WHILE .NOT. answer$"YyNn"
  answer = " "
  @ 21,10 SAY " Would you like to add another product "+
  "information ? (y/n)";
  GET answer
  READ
ENDDO

@ 21,10 CLEAR
IF UPPER(answer)="N"
  EXIT
ENDIF
* Clear current entries from the entry form to make room
* for new entries.
CLEAR
ENDDO loop1
* close out files and return to the menu.
CLOSE DATABASES
RETURN
* Eof:PSM1.PRG program to add a product structure into the database.

*=================================================================================================
*PROGRAM : PSM1.PRG
*screen form for adding new product information
*
*=================================================================================================

@ 3, 25 SAY "ADD NEW PRODUCT INFORMATION"
@ 4, 25 to 4, 51
@ 7, 3 SAY "-->> PRODUCT ID NUMBER [OR 0 TO QUIT] :"
@ 9, 3 SAY "-->> PRODUCT NAME :"
@ 11, 3 SAY "-->> ENTER THE STANDARD TIMES FOR PROCESSING THE PRODUCT"
@ 12, 8 SAY "IN EACH OF THE FOLLOWING SECTIONS IN HOURS"
@ 14, 17 SAY "MACHINING :"
@ 15, 17 SAY "ASSEMBLY :
@ 16, 17 SAY "FULL UP :
@ 17, 17 SAY "PACKAGING :
@ 19, 6 SAY "TOTAL PROCESSING TIME :"
@ 2, 0 TO 20, 79 DOUBLE
@ 5, 1 TO 5, 78
RETURN
*EOF:PSM1.PRG

*=================================================================================================
*PROGRAM : PSM2.PRG
* Deleting products from product structure databases
*=================================================================================================

USE PRODUCTS INDEX PRODS
DO WHILE .T.

CLEAR
* initialize the memory variables
mproduct_id=SPACE(15)
*input part_id of the part to be deleted
@ 1, 1 TO 19, 79 DOUBLE
@ 2, 15 SAY " ENTRY FORM FOR DELETING PRODUCTS FROM DATABASE "
@ 3, 2 TO 3, 78 DOUBLE
@ 9, 2 TO 12, 77
@ 10, 5 SAY " Enter Product ID Number [or 0 to Quit] : ";
GET mproduct_id PICTURE "8888888888"
READ
*
*testing for exit condition
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. mproduct_id="
EXIT
ENDIF
STORE LTRIM(mproduct_id) TO mproduct_id
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. LEN(mproduct_id) = 0
EXIT
ENDIF

@ 9, 2 CLEAR TO 12, 77
* Search for part to be deleted in the database
SEEK mproduct_id
IF FOUND()
* If the record is located, display the fields
  5,10 SAY " PRODUCT ID NUMBER : "
  7,10 SAY " PRODUCT NAME : "
  5,30 SAY Product_id PICTURE "BBXXXXXXXXXXXXXXX"
  7,30 SAY Prodctname PICTURE "QBXXXXXXXXXXXXXXX"
  9,10 SAY " STANDARD TIME - MACHINING :"
  9,45 SAY Stime_mc PICTURE "BB999999999.99"
 10,10 SAY " STANDARD TIME - ASSEMBLY :"
 10,45 SAY Stime_ass PICTURE "BB999999999.99"
 11,10 SAY " STANDARD TIME - FULLUP :"
 11,45 SAY Stime_flp PICTURE "BB999999999.99"
 12,10 SAY " STANDARD TIME - PACKAGING :"
 12,45 SAY Stime_pckg PICTURE "BB999999999.99"
 14,10 SAY " TOTAL PROCESSING TIME :"
 14,45 SAY Total_hrs PICTURE "BB999999999.99"

* ask the user whether this is the right part to be deleted
  answer= ""
  DO WHILE .NOT. answer$ "YyNn"
    answer= ""
    20,10 SAY " Is this the Product to be deleted ? (y/n)";
    GET answer
    READ
  ENDDO
  20,0 CLEAR
  IF UPPER(answer)="Y"
    20,10 CLEAR
    DELETE FOR Product_id=mproduct_id
    20,25 SAY "Product Deleted from Database"
    WAIT SPACE(25)+"Press any key to continue..."
  ELSE
    20,31 SAY " Not Deleted"
    WAIT SPACE(25)+"Press any key to continue..."
  ENDIF
  ELSE
* if product was not located in the database, clear screen
    CLEAR
    20,30 SAY " Product cannot be found !"
    WAIT SPACE(30)+"Press any key to continue..."
    20,10 CLEAR
  ENDIF
* ask the user whether he wants to delete another part
  20,10 CLEAR
  20,10 SAY " Would you like to delete another product ? (y/n) "
  answer= ""
  DO WHILE .NOT. answer$ "YyNn"
    answer= ""
    20,70 GET answer
    READ
  ENDDO
  20,10 CLEAR
  IF UPPER(answer) = "N"
    EXIT
  ENDIF
* clear the screen
  CLEAR
  ENDDO
  WAIT SPACE(23)+"Press any key to return to menu"
  PACK
  CLOSE DATABASES
  RETURN
*EOF:PSM2.PRG
*====================================================================================================================================*
* Program: PSM3.PRG
* Reviewing product structures
*====================================================================================================================================*

SET HEADING OFF
DO WHILE .T.

* erase screen and draw menu
CLEAR
A2,0 TO 12,79 DOUBLE
A 4,1 TO 4,78
A 5,22 SAY "REVIEW PRODUCT STRUCTURES - MENU"
A 7,22 SAY "0 - Exit"
A 9,22 SAY "1 - Review a particular Product Structure"
A 9,22 SAY "2 - Review a number of Products at a time"
A 12,33 SAY "select | | ":

* input valid menu option
choice ="5"
DO WHILE .NOT. choice$"012"
choice = ""
A 12,42 GET choice
READ
ENDDO

* display asterisk next to menu selection
A 6+VAL(choice),21 SAY "*"

* erase other selections from menu
cnt = 0
DO WHILE cnt < 4
  IF cnt <> VAL(choice)
    A 6+cnt,23 SAY SPACE(45)
  ENDIF
  cnt = cnt + 1
ENDDO

* test for exit condition
IF choice $" 0"
EXIT
ENDIF
IF choice$" 1"
  A15,0 CLEAR
  DO PSM31
ENDIF

IF choice$"2"
  A15,0 CLEAR TO 17,79
  USE PRODUCTS INDEX PRODS
  GO TOP
  DO WHILE .NOT. EOF()
    CLEAR
    A 3,30 SAY "PRODUCT STRUCTURE REVIEW"
    A 4,30 SAY "-----------------------------"
    A6,0 TO 6,79
    A7,0 SAY "PRODUCT ID# MACHINE ASSEMBLY FULLUP PACKAGING TOTAL HRS. "
    A8,0 TO 8,79 DOUBLE
    LIST NEXT 10 ;
    PRODUCT_ID,STIME_MC,STIME_ASS,STIME_FLP,STIME_PCKG,TOTAL_HRS OFF
    IF EOF()
222,20 SAY "No more Products in the Database"
WAIT SPACE(23) "$Press any key to continue..."
EXIT
ELSE
SKIP
ENDIF
choice=" "
DO WHILE .NOT. choice$"yYNn"
choice=" "
221,20 SAY "Do you want to continue? (y/n) :";
GET choice
READ
ENDIF
EXIT
ENDDO
ENDIF
ENDDO
CLOSE DATABASES
SET HEADING ON
RETURN
*eof:PSM3.PRG

*==================================================================
* program :PSM3.PRG
* product structure review-option
*==================================================================

USE PRODUCTS INDEX PRODS
DO WHILE .T.
CLEAR
*initialize all memory variables
mproduct_id=SPACE(15)
*set up exit condition
a 1,1 TO 4,79 DOUBLE
a 5,1 TO 17,79 DOUBLE
a 2,8 SAY "PRODUCT STRUCTURE REVIEW WINDOW"
a 3,8 SAY "-----------------------------"
a 10,10 SAY " Enter Product ID Number [OR 0 to Quit] : ";
GET mproduct_id PICTURE "!!!";
READ
* TEST for exit condition
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. mproduct_id="
WAIT SPACE(20) "$Press any key to return to menu
CLOSE DATABASES
RETURN
ENDIF
STORE LTRIM(mproduct_id) TO mproduct_id
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. LEN(mproduct_id) = 0
WAIT SPACE(20) "$Press any key to return to menu
CLOSE DATABASES
RETURN
ENDIF
a 10,10 SAY SPACE(65)
SEEK mproduct_id
* if found, display the fields
a 7,5 SAY " PRODUCT ID NUMBER :
a 7,30 SAY Product_id PICTURE "BBBBBBBBBBBBBBBBBBB"
a 9,5 SAY " PRODUCT NAME :

USE PRODUCTS INDEX PRODS
CLEAR
DO WHILE .T.
  CLEAR
  *initialize the memory variables
  mproduct_id=SPACE(15)
  mprodctname=SPACE(15)
  mstime_mc=0.00
  mstime_ass=0.00
  mstime_flp=0.00
  mstime_pckg=0.00
  mtotal_hrs=0.00
  monk=.T.
  DO WHILE monk
    CLEAR
    * Set up the working environment
    @ 1,1 TO 3,79 DOUBLE
    @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
    @ 6,1 TO 17,79 DOUBLE
    * Set up exit condition
    *input part_id of the part whose structure is to changed
    @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
    GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
    READ
    * Test for exit condition
    ELSE
      CLEAR
      DO WHILE .T.
        CLEAR
        mproduct_id=SPACE(15)
        mprodctname=SPACE(15)
        mstime_mc=0.00
        mstime_ass=0.00
        mstime_flp=0.00
        mstime_pckg=0.00
        mtotal_hrs=0.00
        monk=.T.
        DO WHILE monk
          CLEAR
          * Set up the working environment
          @ 1,1 TO 3,79 DOUBLE
          @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
          @ 6,1 TO 17,79 DOUBLE
          * Set up exit condition
          *input part_id of the part whose structure is to changed
          @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
          GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
          READ
          * Test for exit condition
          ENDIF
          ELSE
            CLEAR
            DO WHILE .T.
              CLEAR
              mproduct_id=SPACE(15)
              mprodctname=SPACE(15)
              mstime_mc=0.00
              mstime_ass=0.00
              mstime_flp=0.00
              mstime_pckg=0.00
              mtotal_hrs=0.00
              monk=.T.
              DO WHILE monk
                CLEAR
                * Set up the working environment
                @ 1,1 TO 3,79 DOUBLE
                @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
                @ 6,1 TO 17,79 DOUBLE
                * Set up exit condition
                *input part_id of the part whose structure is to changed
                @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
                GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
                READ
                * Test for exit condition
                ELSE
                  CLEAR
                  DO WHILE .T.
                    CLEAR
                    mproduct_id=SPACE(15)
                    mprodctname=SPACE(15)
                    mstime_mc=0.00
                    mstime_ass=0.00
                    mstime_flp=0.00
                    mstime_pckg=0.00
                    mtotal_hrs=0.00
                    monk=.T.
                    DO WHILE monk
                      CLEAR
                      * Set up the working environment
                      @ 1,1 TO 3,79 DOUBLE
                      @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
                      @ 6,1 TO 17,79 DOUBLE
                      * Set up exit condition
                      *input part_id of the part whose structure is to changed
                      @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
                      GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
                      READ
                      * Test for exit condition
                      ENDIF
                      ELSE
                        CLEAR
                        DO WHILE .T.
                          CLEAR
                          mproduct_id=SPACE(15)
                          mprodctname=SPACE(15)
                          mstime_mc=0.00
                          mstime_ass=0.00
                          mstime_flp=0.00
                          mstime_pckg=0.00
                          mtotal_hrs=0.00
                          monk=.T.
                          DO WHILE monk
                            CLEAR
                            * Set up the working environment
                            @ 1,1 TO 3,79 DOUBLE
                            @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
                            @ 6,1 TO 17,79 DOUBLE
                            * Set up exit condition
                            *input part_id of the part whose structure is to changed
                            @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
                            GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
                            READ
                            * Test for exit condition
                            ELSE
                              CLEAR
                              DO WHILE .T.
                                CLEAR
                                mproduct_id=SPACE(15)
                                mprodctname=SPACE(15)
                                mstime_mc=0.00
                                mstime_ass=0.00
                                mstime_flp=0.00
                                mstime_pckg=0.00
                                mtotal_hrs=0.00
                                monk=.T.
                                DO WHILE monk
                                  CLEAR
                                  * Set up the working environment
                                  @ 1,1 TO 3,79 DOUBLE
                                  @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
                                  @ 6,1 TO 17,79 DOUBLE
                                  * Set up exit condition
                                  *input part_id of the part whose structure is to changed
                                  @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
                                  GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
                                  READ
                                  * Test for exit condition
                                  ELSE
                                    CLEAR
                                    DO WHILE .T.
                                      CLEAR
                                      mproduct_id=SPACE(15)
                                      mprodctname=SPACE(15)
                                      mstime_mc=0.00
                                      mstime_ass=0.00
                                      mstime_flp=0.00
                                      mstime_pckg=0.00
                                      mtotal_hrs=0.00
                                      monk=.T.
                                      DO WHILE monk
                                        CLEAR
                                        * Set up the working environment
                                        @ 1,1 TO 3,79 DOUBLE
                                        @ 2,20 SAY " PRODUCT STRUCTURE CHANGING ROUTINE "
                                        @ 6,1 TO 17,79 DOUBLE
                                        * Set up exit condition
                                        *input part_id of the part whose structure is to changed
                                        @10,10 SAY "Enter Product ID Number [OR 0 to Quit] : ";
                                        GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
                                        READ
                                        * Test for exit condition
                                        ELSE
                                          CLEAR
                                          DO WHILE .T.
                                            CLEAR
                                            mproduct_id=SPACE(15)
                                            mprodctname=SPACE(15)
                                            mstime_mc=0.00
                                            mstime_ass=0.00
                                            mstime_flp=0.00
                                            mstime_pckg=0.00
                                            mtotal_hrs=0.00
                                            monk=.T.
                                            DO WHILE monk
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. mproduct_id=""  RETURN
ENDIF
STORE LTRIM(mproduct_id) TO mproduct_id
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. LEN(mproduct_id) = 0
WAIT SPACE(20)+"Press any key to return to menu "
CLOSE DATABASES
RETURN
ENDIF

* Find the part to be changed
SEEK mproduct_id
IF FOUND()
  * If found, display the information and confirm
  CLEAR
  @ 9,0 TO 13,79 double
  @ 10,25 SAY "Product ID Number : "
  @ 10,44 SAY Product_id PICTURE "#BXXXXXXXXXXX "
  @ 12,25 SAY "Product Name : "
  @ 12,44 SAY Prodctname PICTURE "#BXXXXXXXXXXX "
  * ask the user whether this is the right part
  answer=" "
  DO WHILE .NOT. answerS"yNn"
    answer=" "
    @20,18 SAY " Is this the product you want to change? (y/n) ";
    GET answer
    READ
  ENDDO
  @20,5 CLEAR
  IF UPPER(answer)="Y"
    EXIT
  ELSE
    CLEAR
    @ 22,20 SAY "Product not found !"
    WAIT SPACE(28)+"Press any key to reenter"
    CLEAR
  ENDIF
ENDDO
STORE Product_id TO mproduct_id
STORE Prodctname TO mprouctname
STORE Stime_mc TO mstime_mc
STORE Stime_ass TO mstime_ass
STORE Stime_flp TO mstime_flp
STORE Stime_pckg TO mstime_pckg
STORE Total_hrs TO mtotal_hrs
CLEAR
DO WHILE .T.
  CLEAR
  @ 2,1 TO 5,78 DOUBLE
  @ 3,15 SAY " GO AHEAD AND MAKE THE NECESSARY CHANGES"
  @ 4,15 SAY "----------------------------------------
  @ 6,1 TO 21,78
  @ 8,10 SAY "Product ID Number [OR 0 to Quit] ";
  GET mproduct_id PICTURE "!!!-------------------"
  READ
  IF mproduct_id="0" .OR. mproduct_id=SPACE(15)
    WAIT SPACE(13)+"Press any key to continue -- nothing was changed"
    EXIT
  ENDIF
ENDIF
STORE LTRIM(mproduct_id) TO mproduct_id
IF mproduct_id="0" .OR. mproduct_id=SPACE(15) .OR. LEN(mproduct_id) = 0
   WAIT SPACE(13)+"Press any key to continue -- nothing was changed"
   EXIT
ENDIF
@ 8,10 SAY SPACE(60)
@ 8,10 SAY "PRODUCT ID NUMBER :"+mproduct_id
@ 10,10 SAY "PRODUCT NAME :";
GET mprodctname PICTURE "!!!!!!!!!!!!!!!!!!!"
@ 12,10 SAY "STANDARD TIME -- MACHINING :";
GET mstime_mc PICTURE "9999999.99"
@ 14,10 SAY "STANDARD TIME -- ASSEMBLY :";
GET mstime_ass PICTURE "9999999.99"
@ 16,10 SAY "STANDARD TIME -- FULLUP :";
GET mstime_flp PICTURE "9999999.99"
@ 18,10 SAY "STANDARD TIME -- PACKAGING :";
GET mstime_pckg PICTURE "9999999.99"
READ
mtotal_hrs=(mstime_mc+mstime_ass+mstime_flp+mstime_pckg)
@ 20,10 SAY "TOTAL PROCESSING TIME :"
@ 20,35 SAY mtotal_hrs PICTURE "@9999999.99"
* CHECK whether the information is correct
answer=""
DO WHILE .NOT. answer$ "yYnN"
   @ 22,20 SAY "Is this information correct ? (y/n)";
   GET answer
   READ
ENDDO
@ 22,10 CLEAR
IF UPPER(answer)="Y"
   * now we can replace the old values by the new ones
   REPLACE Product_id WITH UPPER(mproduct_id)
   REPLACE Prodctname WITH UPPER(mprodctname)
   REPLACE Stime_mc WITH mstime_mc
   REPLACE Stime_ass WITH mstime_ass
   REPLACE Stime_flp WITH mstime_flp
   REPLACE Stime_pckg WITH mstime_pckg
   REPLACE TOTAL_HRS WITH mtotal_hrs
@ 22,0 CLEAR
@ 22,18 SAY " Requested changes completed "
   EXIT
ELSE
   @22,20 SAY "Please type in correct data"
   WAIT SPACE(20)+"Press any key to continue..."
ENDDO
ENDIF
choice=""
DO WHILE .NOT. choice$"yYnN"
   choice=""
   @ 23,13 SAY " Do you want to change another product? (y/n)" ;
   GET choice
   READ
ENDDO
IF UPPER(choice)="W"
   EXIT
ENDIF
@22,0 clear to 23,79
WAIT SPACE(18)+"Press any key to return to menu.."
CLOSE DATABASES
RETURN
*EOF:PSM4.PRG
Program: SOM.PRG
main program for sales order management

DO WHILE .T.
  CLEAR
  @ 2, 0 TO 15,79 DOUBLE
  @ 3,18 SAY [SALES ORDERS MANAGEMENT]
  @ 4,1 TO 4,78 DOUBLE
  @ 7,25 SAY '0. Return to Main Menu'
  @ 8,25 SAY [1. Add Sales Order Information]
  @ 9,25 SAY [2. Change Sales Order Information]
  @ 10,25 SAY [3. Purge Sales Order Information]
  @ 11,25 SAY [4. Review Sales Order Information]
  @ 12,25 SAY [5. Reports]
  choice="6"
  DO WHILE .NOT. choice$"012345"
    choice=""
    @ 15,33 SAY " select | | "
    @ 15,42 GET choice
    READ
  ENDDO
  @ 7+VAL(choice),23 SAY "="
  cnt=0
  DO WHILE cnt<6
    IF cnt<>VAL(choice)
      @ 7+cnt,25 SAY SPACE(40)
    ENDF
    cnt=cnt+1
  ENDDO
  DO CASE
    CASE choice = "0"
      CLEAR ALL
      RETURN
    CASE choice = "1"
      * DO ADD INFORMATION
      DO SOM1
    CASE choice = "2"
      * DO CHANGE INFORMATION
      DO SOM4
    CASE choice = "3"
      * DO REMOVE INFORMATION
      DO SOM2
    CASE choice = "4"
      * DO REVIEW INFORMATION
      DO SOM3
    CASE choice = "5"
      * DO PRINT REPORT
      @ 19,0 CLEAR
      DO SOM41
  ENDCASE
ENDDO T
DO WHILE .T.
  * initialize all the memory variables
  msales_num=SPACE(15)
  mproduct_id=SPACE(15)
  mquantity=0
  mdue_date=DATE()
  mpriority=SPACE(3)
  mcomments=SPACE(45)
  mcomment1=SPACE(10)
  minternal=.N.
  monk=.T.
  CLEAR
  DO SOM11
  DO WHILE monk
    * setup exit condition
    DO WHILE .T.
      @7,38 GET msales_num PICTURE "!!!!!!!!!!!!!!!"
      READ
      * Test for exit condition
      IF msales_num = "0" .OR. msales_num=SPACE(15)
        WAIT SPACE(23)+"Press any key to Return to Menu"
        CLOSE DATABASES
        RETURN
      ENDFILE
      STORE LTRIM(msales_num) TO msales_num
      IF msales_num = "0"
        WAIT SPACE(23)+"Press any key to Return to Menu"
        CLOSE DATABASES
        RETURN
      ENDFILE
      USE SALES INDEX SNUM
      SEEK msales_num
      IF FOUND()
        @21,0 CLEAR TO 22,79
        @21,22 SAY "This Sales Order already exists!"
        answer=""
        DO WHILE .NOT. answer$"YnN"
          answer=""
          @22,10 SAY "Would you like to make changes to the existing Order? (y/n)"
        GET answer
        READ
        ENDDO
      IF UPPER(answer)="Y"
        CLEAR
        DO SOM12 WITH msales_num
        CLOSE DATABASES
        CLEAR
        RETURN
      ELSE
        @21,0 CLEAR TO 22,79
        @21,20 SAY "Please type in correct Sales Order Number"
        @22,20 SAY "******************************
        LOOP
ENDIF
ELSE
@21,0 CLEAR TO 22,79
EXIT
ENDIF
@21,0 CLEAR TO 22,79
DO SOM11
@21,10 SAY "Please type in Sales Order Number or Enter 0 to quit"
@22,10 SAY "*****************************************************************************"
ENDDO
USE
@21,10 CLEAR
@7,66 GET mdue_date
@9,13 GET mpriority PICTURE "!XX"
@9,28 GET mcomments
@11,20 GET minternal
@16,22 GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
@16,56 GET mquantity PICTURE "9999999999" RANGE 0,
@18,13 GET mcomment1
READ
*check to see if information is correct
IF mpriority=SPACE(3)
   @21,15 SAY "Please Check the Priority information again !!"
   @22,15 SAY "*****************************************************************************"
   LOOP
ELSE
   IF LEN(mpriority) < 2
      @21,15 SAY "Please Check the Priority information again !!"
      @22,15 SAY "*****************************************************************************"
      LOOP
ENDIF
ENDIF
@21,10 CLEAR
* if some one enters a invalid product id , reenter with warning
IF mproduct_id=SPACE(15)
   @21,15 SAY "Please Check the Product Id information again !!"
   @22,15 SAY "*****************************************************************************"
   LOOP
ELSE
   STORE LTRIM(mproduct_id) TO mproduct_id
   IF mproduct_id="0" .OR. LEN(mproduct_id) < 1
      mproduct_id=SPACE(15)
      @21,15 SAY "Please Check the Product Id information again !!"
      @22,15 SAY "*****************************************************************************"
      LOOP
   ENDIF
ENDIF
answer=" "
DO WHILE .NOT. answer$"YyNn"
   answer = " "
   @21,10 SAY "Is the above information correct ? (y/n)"
   GET answer
   READ
ENDDO
@21,0 CLEAR TO 23,79
* if answer is yes we need to exit loop
IF answer$"YY"
   EXIT
ENDIF
* otherwise we need to reenter the data
IF answer$"NN"
   @21,23 SAY " Please type in correct data "
   @22,23 SAY "*****************************************************************************"
   LOOP
ENDIF
ENDDO monk
IF UPPER(answer)="Y"
    STORE mproduct_id TO m1product
    USE PRODUCTS INDEX PRODS
    SEEK mproduct_id
    IF .NOT. FOUND()
        CLEAR
        05,10 SAY "PRODUCT ID NUMBER :
        05,32 SAY mproduct_id PICTURE "9999999999999999999999999"
        06,10 SAY "This Product DOES NOT EXIST in the Product Structure"
        07,10 SAY "Database. You need to Add this now -OR- Promptly before"
        08,10 SAY "you run the Scheduled Production Hours Program!"
        zeta=" 
        DO WHILE .NOT. zeta$ "yYnN"
            zeta=" 
            10,10 SAY "Would you like to add it now? (y/n) ";
            GET zeta
            READ
        ENDDO
        IF zeta$"yY"
            CLEAR
            DO SOM13 WITH mproduct_id
            CLEAR
        ENDF
        IF zeta$"nN"
            13,10 SAY "RE M I N D E R :"
            14,10 SAY "Remember to add the product structure before"
            15,10 SAY "running the Scheduled Production Hours program!"
            17,10 CLEAR
            WAIT SPACE(20)+"Press any key to continue ... "
        ENDF
    USE
    * the data is correct so we need to add to the database
    USE SALES INDEX SNUM
    APPEND BLANK
    REPLACE SALES_NUM WITH UPPER(msales_num)
    REPLACE Product_id WITH UPPER(m1product)
    REPLACE QUANTITY WITH mquantity
    REPLACE DUE_DATE WITH mdue_date
    REPLACE COMMENTS WITH mcomment1
    USE
ENDIF
    21,0 CLEAR
    * option to enter or not enter another product info.
    answer=" 
    DO WHILE .NOT. answer$"yYnN"
        answer=" 
        21,7 SAY " Are there more Products in This Sales *;"
        "Order ? (y/n)"
        GET answer
        READ
    ENDDO
    IF UPPER(answer)="Y"
        CLEAR
        mproduct_id=SPACE(15)
        mquantity=0
        mcomment1=SPACE(10)
        DO WHILE .T.
            CLEAR
            04,1 TO 17,79 DOUBLE
            06,25 SAY " NEXT PRODUCT "
            07,25 SAY "-------------------------"
            10,10 SAY "ENTER PRODUCT ID [OR 0 TO EXIT] :
            GET mproduct_id PICTURE ""
READ
"test for exit condition
IF mproduct_id=SPACE(15)
   EXIT
ELSE
   STORE LTRIM(mproduct_id) TO mproduct_id
   IF mproduct_id="0"
      EXIT
   ENDIF
ENDIF
ENDIF
@ 12,10 SAY "ENTER QUANTITY : ";
GET mquantity RANGE 0,
@ 14,10 SAY "COMMENTS : " GET mcomment1
READ
"check if the info is correct
confirm=""
DO WHILE .NOT. confirm$"yYnN"
   confirm=""
   @ 20,10 SAY "Is the above information correct ? (y/n) ";
   GET confirm
   READ
ENDO
@ 20,0 CLEAR
IF confirm$"yY"
   USE PRODUCTS INDEX PRODS
   STORE mproduct_id TO m2product
   SEEK mproduct_id
   IF .NOT. FOUND()
      CLEAR
      85,10 SAY "PRODUCT ID NUMBER : ":
      85,32 SAY mproduct_id PICTURE "@XXXXXXXXXXXXXXXX":
      86,10 SAY "This product does not exist in the product structure":
      87,10 SAY "Database. You need to add this now -or- promptly before"
      88,10 SAY "You run the Scheduled Production Hours Program!"
      zeta=""
      DO WHILE .NOT. zeta$"yYnN"
         zeta=""
         810,10 SAY "Would you like to add it now? (y/n) ";
         GET zeta
         READ
      ENDDO
      IF zeta$"yY"
         CLEAR
         DO SOM13 WITH mproduct_id
         CLEAR
      ENDDO
      IF zeta$"nN"
         @ 13,10 SAY "REMINDER :"
         @ 14,10 SAY "Remember to add the product structure before"
         @ 15,10 SAY "running THE Scheduled Production Hours program!"
         WAIT SPACE(13)+"Press any key to continue ...
      ENDIF
   ENDIF
ENDIF
USE
USE SALES INDEX SNUM
APPEND BLANK
REPLACE Sales_num WITH UPPER(msales_num)
REPLACE Product_id WITH UPPER(m2product)
REPLACE Quantity WITH mquantity
REPLACE DUE_DATE WITH mdue_date
REPLACE COMMENTS WITH mcomment1
USE
mproduct_id=SPACE(15)
mquantity=0
mcomment1=SPACE(10)
ENDDO
ENDIF
ENDIF
a 21,0 CLEAR
USE PRIORITY INDEX SNUMBER
APPEND BLANK
REPLACE Sales_num WITH UPPER(msales_num)
REPLACE Priority WITH UPPER(mpriority)
REPLACE Comments WITH mcomments
REPLACE Due_date WITH mdue_date
REPLACE MFG_DATE WITH mdue_date
REPLACE INTERNAL WITH minternal
REPLACE SCHEDULED WITH .F.
USE
answer=""
DO WHILE .NOT. answer$"yYN"
   answer=""
   @ 21,10 SAY " Do you want to add another order ? (y/n) :";
   GET answer
   READ
ENDDO
IF UPPER(answer)="N"
   EXIT
ENDIF
*Clear current entries from the entry form to make room
*for new entries.
CLEAR
ENDDO loop1
CLOSE DATABASES
RETURN
* EOF:SOM1.PRG

*==================================================================
* PROGRAM : SOM11.PRG
* sales order entry form
*==================================================================
@ 3, 24 SAY "NEW SALES ORDERS ENTRY FORM"
@ 4, 24 to 4,50
@ 7, 2 SAY "Sales Order Number [or 0 to Quit] :
@ 7, 55 SAY "Due Date :
@ 9, 17 SAY "Comments :
@ 9, 2 SAY "Priority :
@ 11, 2 SAY "Internal (y/n) :
@ 12, 2 SAY "If there are more than one products in this sales order, start by"
@ 13, 2 SAY "entering the first product's details. The system will prompt you"
@ 14, 2 SAY "for the next product."
@ 16, 2 SAY "Product Id Number : Quantity :
@ 18, 2 SAY "Comments :
@ 2, 0 TO 5, 79 DOUBLE
@ 6, 0 TO 20, 79 DOUBLE
RETURN
* EOF:SOM11.PRG

*==================================================================
*PROGRAM : SOM12.PRG
*called by program, SOM1.PRG which helps to add new sales orders and is
*called if
* user enters an existing sales order number and
* on inquiry, wants to make some changes to it
*==================================================================
PARAMETERS msales_num
SELECT A
USE SALES INDEX SNUM
SELECT B
USE PRIORITY INDEX SNUM
SELECT C
USE NEW1 INDEX NUMB
REINDEX

SELECT A
DO WHILE .T.
  CLEAR
  *initialize the memory variables
  mproduct_id=SPACE(15)
  mquantity=0
  mpriority=SPACE(3)
  mdue_date=DATE()
  mcomments=SPACE(45)
  mcomment1=SPACE(10)
  osales_num=msales_num
  mfinalised=.F.
  mfinalweek=DATE()
  indicator=0
  minternal=.F.
  monk=.T.
  SEEK osales_num
  STORE Sales_num TO osales_num
  STORE Sales_num TO msales_num
  STORE Product_id TO mproduct_id
  STORE Quantity TO mquantity
  STORE Comments TO mcomment1
  SELECT B
  SEEK osales_num
  IF FOUND()
    IF SCHEDULED
      REPLACE CHANGED WITH .T.
    ENDIF
    STORE Priority TO mpriority
    STORE Due_date TO mdue_date
    STORE Comments TO mcomments
    STORE INTERNAL TO minternal
  ENDIF
  CLEAR
  DO WHILE .T.
    a1,1 CLEAR TO 20,79
    @ 1,14 TO 3,55
    a 1,15 SAY " Go ahead and make the necessary Changes"
    @ 10,10 SAY "Sales Order Number [OR 0 to Quit]: ";
    GET msales_num PICTURE "!!!!!!!!!!!!!!!"
    READ
    IF msales_num="0" .OR. msales_num=SPACE(15)
      WAIT SPACE(13)+"Press any key to continue -- nothing was changed"
      EXIT
    ENDIF
    STORE LTRIM(msales_num) TO msales_num
    IF msales_num="0" .OR. msales_num=SPACE(15)
      WAIT SPACE(13)+"Press any key to continue -- nothing was changed"
      EXIT
    ENDIF
  ENDWHILE
  IF osales_num<>msales_num
    SELECT B
    SEEK msales_num
IF FOUND()
    @22,10 SAY "This Sales Order already exists - Give new Number"
    LOOP
ENDIF
ENDIF
a 10,0 CLEAR
a 10,10 SAY "Sales Order #:"
a 10,27 SAY msales_num
a 10,45 SAY "Internal (y/n) :" GET minternal
a 12,10 SAY "Priority :"
a 12,23 GET mpriority PICTURE "!99"
a 12,30 SAY "Due date :"
a 12,42 GET mdue_date
a 14,10 SAY "Comments :"
a 14,22 GET mcomments
a 16,10 SAY "Product Id Number :"
a 16,30 GET mproduct_id PICTURE ""1111111111"
a 18,10 SAY "Quantity :
    a 18,27 GET mquantity PICTURE "9999"
READ
" CHECK whether the information is correct
IF UPPER(answer)="Y"
    READ
ENDDO
IF answer="Y"
    a 20,0 clear to 23,79
    a 22,10 SAY "Please wait..."
    * now we can replace the old values by the new ones
    SELECT A
    IF osales_num <> msales_num
        DO WHILE .T.
            GO TOP
            SEEK osales_num
            IF FOUND()
                REPLACE SALES_NUM WITH UPPER(msales_num)
            ELSE
                EOF()
                EXIT
            ENDIF
        ENDDO
    ENDIF
    SELECT A
        GO TOP
        SEEK msales_num
        IF FOUND()
            REPLACE Product_id WITH UPPER(mproduct_id)
            REPLACE Quantity WITH mquantity
            REPLACE DUE_DATE WITH mdue_date
            REPLACE COMMENTS WITH mcomment1
            SKIP
        ENDIF
    ENDIF
    SELECT A
        GO TOP
        SEEK osales_num
        IF FOUND()
            REPLACE SALES_NUM WITH UPPER(msales_num)
            REPLACE PRIORITY WITH UPPER(mpriority)
            REPLACE DUE_DATE WITH mdue_date
            REPLACE MFG_DATE WITH mdue_date
            REPLACE COMMENTS WITH mcomments
            SKIP
        ENDIF
    ENDIF
    SELECT B
        GO TOP
        SEEK osales_num
        IF FOUND()
            REPLACE SALES_NUM WITH UPPER(msales_num)
            REPLACE PRIORITY WITH UPPER(mpriority)
            REPLACE DUE_DATE WITH mdue_date
            REPLACE MFG_DATE WITH mdue_date
            REPLACE COMMENTS WITH mcomments
            SKIP
        ENDIF
    ENDIF
ENDIF
REPLACE INTERNAL WITH minternal

ENDIF

*Start of process to change all products in this order

SELECT C

GO TOP

IF osales_num <> msales_num

DO WHILE .T.

GO TOP

SEEK osales_num

IF FOUND()

REPLACE SALES_NUM WITH UPPER(msales_num)

ELSE

IF EOF()

EXIT

ENDIF

ENDIF

ENDDO

ENDIF

SELECT A

DO WHILE .T.

IF Sales_num=msales_num

mproduct_id=Product_id

mquantity=Quantity

mcomment1=COMMENTS

CLEAR

3,1 TO 17,79

5,30 SAY "NEXT PRODUCT"

6,30 SAY "***************"

8,10 SAY "SALES ORDER NUMBER :

8,32 SAY msales_num PICTURE "99999999999999999999999999"

8,48 SAY "CONTINUED ..."

9,10 SAY "--------------"

9,48 SAY "--------------"

10,10 SAY "PRODUCT ID NUMBER [Enter 0 to Quit] :

READ

IF mproduct_id="0" .OR. mproduct_id=SPACE(15)

indicator=1

EXIT

ENDIF

12,10 SAY "QUANTITY :

GET mquantity PICTURE "9999"

14,10 SAY "COMMENTS :

GET mcomment1

READ

REPLACE PRODUCT_ID WITH UPPER(mproduct_id)

REPLACE QUANTITY WITH mquantity

REPLACE DUE_DATE WITH mdue_date

REPLACE COMMENTS WITH mcomment1

ELSE

EXIT

ENDIF

SKIP

IF EOF()

EXIT

ENDIF

ENDDO

IF indicator=1

WAIT SPACE(13)"Press any key to continue - requested changes completed"

EXIT

ENDIF

20,10 SAY "No more products in this sales order"

request=" "
DO WHILE .NOT. request$="YyNn"
    request=" "
    @21,10 SAY "Would you like to Add a Product into this"
    " Sales Order? (y/n) ";
    GET request
    READ
ENDO
IF UPPER(request)="Y"
    STORE SPACE(15) TO mproduct_id
    STORE 0 TO mquantity
    STORE SPACE(10) TO mcomment1
    CLEAR
    DO WHILE .T.
        @5.5 TO 17.75 DOUBLE
        @7.10 SAY "Adding a Product into Sales Order :"
        @7.46 SAY msales_num PICTURE "99XXXXXXXXXXXXXXX"
        @8.10 TO 8.61
        @10.10 SAY "Enter Product ID # [OR 0 to Quit] ":
        GET mproduct_id PICTURE "!"!
        READ
        IF mproduct_id=SPACE(15) .OR. mproduct_id="0" .OR.
            mproduct_id="-
        ENDIF
        EXIT
        STORE LTRIM(mproduct_id) TO mproduct_id
        IF mproduct_id="0"
            EXIT
        ENDIF
        @12.10 SAY "Enter Quantity :"
        GET mquantity PICTURE "9999" R ange 0,
        @ 14.10 SAY "COMMENTS :" GET mcomment1
        READ
        correct=" "
        DO WHILE .NOT. correct$="YyNn"
            correct=" "
            @ 20.20 SAY "Is this information correct? (y/n)";
            GET correct
            READ
        ENDDO
    ENDIF
    STORE LTRIM(mproduct_id) TO mproduct_id
    IF mproduct_id="0"
        EXIT
    ENDIF
    @22,20 SAY " Please type in correct information 
    @23.20 SAY "------------------------------------------"
    LOOP
ENDIF
checker=" "
@21,0 CLEAR TO 23,79
DO WHILE .NOT. checker$="YyNn"
    checker=" "
    @20.10 SAY "Do you want to add another product into"
    " this sales order? (y/n) ";
    GET checker
    READ
ENDDO
IF UPPER(checker)="N"
    EXIT
ENDIF
STORE SPACE(15) TO mproduct_id
STORE 0 TO mquantity
STORE SPACE(10) TO mcomment1
Q20,0 CLEAR TO 23,79
ENDDO
ENDIF
WAIT SPACE(13)+"Press any key to continue - requested changes completed"
EXIT
ELSE
Q20,0 clear to 22,79
WAIT SPACE(20)+"Press any key to type in correct data.."
ENDIF
ENDDO
EXIT
STORE SPACE(15) TO msales_num
CLOSE DATABASES
RETURN
*EOF:SOM12.PRG

*=================================================================================================
* program : SOM13.PRG
* interrupts scheduling for unknown product structure
*=================================================================================================

PARAMETERS mproduct_id
USE PRODUCTS INDEX PRODS
*loop1 begins here.
DO WHILE .T.
  * initialize all the memory variables
  mprodcname=SPACE(15)
  mstime_mc=0.00
  mstime_ass=0.00
  mstime_flp=0.00
  mstime_pckg=0.00
  mtotal_hrs=0.00
  monk=.T.
  DO WHILE monk
    DO SOM131
      @7,29 SAY mproduct_id
      * without a partname, a record is not complete
      DO WHILE .T.
        @9,29 GET mprodcname PICTURE "!!!!!!!!!!"
        READ
        IF mprodcname <> SPACE(15)
          EXIT
        ENDIF
      ENDDO
      @21,0 CLEAR
      @21,25 SAY "Please give a product name"
    ENDDO
    Q21,0 CLEAR
    @14,32 GET mstime_mc PICTURE "9999999.99" RANGE 0,
    @15,32 GET mstime_ass PICTURE "9999999.99" RANGE 0,
    @16,32 GET mstime_flp PICTURE "9999999.99" RANGE 0,
    @17,32 GET mstime_pckg PICTURE "9999999.99" RANGE 0,
    READ
    mtotal_hrs=(mstime_mc+mstime_ass+mstime_flp+mstime_pckg)
    @19,32 SAY mtotal_hrs PICTURE "9999999.99"
    *check to see if information is correct
    answer=""
    DO WHILE .NOT. answer$"YyNn"
      answer = " "
      @21,15 SAY "Is the above information correct ? (y/n)"
    GET answer
    READ
    ENDDO
IF answers "Yy"
  * the data is correct so we need to add to the database
  APPEND BLANK
  REPLACE PRODUCT_ID WITH UPPER(mproduct_id)
  REPLACE PRODCTNAME WITH UPPER(mprodctname)
  REPLACE STIME_MC WITH mstime_mc
  REPLACE STIME_ASS WITH mstime_ass
  REPLACE STIME_FLP WITH mstime_flip
  REPLACE STIME_PKG WITH mstime_pckg
  REPLACE TOTAL_HRS WITH mtotal_hrs
ENDIF
EXIT
ENDDO loop1
CLEAR
RETURN
*Eof:SOM13.PRG

PROGRAM : SOM131.PRG
*This program forms the screen form(template) for adding new
*products in to the product structure database.
*==============================================================

a 3, 25 SAY "ADD NEW PRODUCT INFORMATION"
a 4, 25 to 4,51
a 7, 3 SAY ""PRODUCT ID NUMBER :"
a 9, 3 SAY ""PRODUCT NAME :"
a 11,3 SAY ""ENTER THE STANDARD TIMES FOR PROCESSING THE PRODUCT"
a 12, 8 SAY "IN EACH OF THE FOLLOWING SECTIONS IN HOURS"
a 14, 17 SAY "MACHINING :"
a 15, 17 SAY "ASSEMBLY :
"a 16, 17 SAY "FULL UP :
"a 17, 17 SAY "PACKAGING :
"a 19, 6 SAY "TOTAL PROCESSING TIME :"
a 2, 0 TO 20, 79 DOUBLE
a 5, 1 TO 5, 78
RETURN
*EOF:SOM131.PRG

Program: SOM2.PRG
* purging sales orders from the database
*=================================================================

mdue_date=DATE()
begnin=DATE()
end=DATE()

DO WHILE .T.
* erase screen and draw menu
CLEAR
@ 2,0 TO 12,79 DOUBLE
@ 4,1 TO 4,78
@ 3,25 SAY "PURGE SALES ORDERS - MENU"
@ 6,22 SAY " 0 - Exit"
@ 7,22 SAY " 1 - Purge a particular Sales order"
@ 8,22 SAY " 2 - Purge Sales orders based on one Duedate"
@ 9,22 SAY " 3 - Purge Sales orders between 2 Duedates"
@ 12,33 SAY " select | | "

* input valid menu option
choice = "5"
DO WHILE .NOT. choice$ "0123"
choice = " 
@ 12,42 GET choice
READ
ENDDO

* display asterisk next to menu selection
@ 6+VAL(choice),21 SAY "***"

* erase other selections from menu
cnt = 0
DO WHILE cnt < 4
  IF cnt <> VAL(choice)
    @ 6+cnt,23 SAY SPACE(45)
  ENDF
  cnt = cnt + 1
ENDDO

* test for exit condition
IF choice $ " 0"
EXIT
ENDDO
IF choice$ " 1"
  CLEAR
  DO SOM21
ENDIF
IF choice$ " 2"
  @ 15,0 CLEAR TO 17,79
  @ 15,10 SAY "Purge all Sales Orders due on :" GET mdue_date
  READ
  @ 22,20 SAY "Processing - Please Wait...."
  USE SALES INDEX SNUM
  DELETE ALL FOR DUE_DATE=mdue_date
  PACK
  USE PRIORITY INDEX SNUMBER
  DELETE ALL FOR DUE_DATE=mdue_date
  @ 22,0
  WAIT SPACE(15)+"Purging complete - Press any key to continue...
ENDIF
IF choice$ " 3"
  @ 15,0 CLEAR TO 19,79
  DO WHILE .T.
    @ 15,10 SAY "Enter the starting Due_date :"
    GET begin
    @ 17,10 SAY "Enter the ending Due_date :"
    GET end
    READ
    IF begin<=end
EXIT
ENDIF
@ 15,0 CLEAR TO 17,79
@ 18,10 SAY "Please reenter - " +
" starting date has to be earlier than ending date"
ENDDO
@ 22,20 SAY "Processing - Please wait...."
USE SALES INDEX SNUM
DELETE ALL FOR DUE_DATE <= end .AND. DUE_DATE >= begin
PACK
USE PRIORITY INDEX SNUMBER
DELETE ALL FOR DUE_DATE <= end .AND. DUE_DATE >= begin
@ 22,0
WAIT SPACE(15) + " Purging complete - Press any key to continue...
ENDIF
ENDDO
CLOSE DATABASES
@ 22,32 SAY "Please Wait...."
DO SOM 22
RETURN
*eof:SOM2.PRG

*=================================================================
* PROGRAM : SOM21.PRG
* deleting product sales orders from database-option
*=================================================================

DO WHILE .T.
   * initialize the memory variables
   msales_num=SPACE(15)
   * input sales_num of the sales order to be deleted
   @ 1,1 TO 19,79 DOUBLE
   @ 2,15 SAY " ENTRY FORM FOR DELETING SALES ORDERS FROM DATABASE 
   @ 3,2 TO 3,78 DOUBLE
   @ 9,2 TO 12,77 DOUBLE
   @ 10,5 SAY " Enter Sales Order Number [or 0 to Quit] : ";
   GET msales_num PICTURE "111111111111111I"
READ
   *
   * testing for exit condition
   IF msales_num="0" .OR. msales_num=SPACE(15)
      EXIT
   ENDIF
   STORE LTRIM(msales_num) TO msales_num
   IF msales_num="0" .OR. LEN(msales_num) = 0
      EXIT
   ENDIF
   @ 9,2 CLEAR TO 12,77
   * search for sales order to be deleted in the database
   USE SALES INDEX SNUM
   SEEK msales_num
   IF FOUND()
      * if the record is located, display the fields
      @ 5,10 SAY "Sales Order Number :"
      @ 5,32 SAY Sales_num PICTURE "@@XXXXXXXXXXXXXXXX"
      USE
      USE PRIORITY INDEX SNUMBER
      SEEK msales_num
      IF FOUND()
         @ 7,10 SAY "Priority :"
         @ 7,22 SAY Priority PICTURE "BBBB"
         @ 7,30 SAY "Due Date :"
         @ 7,42 SAY Due_date
ask the user whether this is the right order to be deleted
answer=""
DO WHILE .NOT. answer$ "YyNn"
   answer=""
   20,20 SAY " Is this the Order to be Deleted ? (Y/N)";
   GET answer
READ
ENDDO
@ 20,0 CLEAR
IF UPER(answer)="Y"
   @ 20,10 CLEAR
   USE
   USE SALES INDEX SNUM
   DELETE FOR Sales_num=msales_num
   PACK
   USE
   USE PRIORITY INDEX SNUMBER
   DELETE FOR Sales_num=msales_num
   USE
   @ 20,20 SAY "Order Deleted from Database"
   WAIT SPACE(20)+"Press any Key to Continue"
ELSE
   @ 20,37 SAY " Not Deleted"
   WAIT SPACE(28)+"Press any Key to Continue"
ENDIF
ELSE
   * IF ORDER WAS NOT LOCATED IN THE DATABASE,CLEAR SCREEN
   CLEAR
   @ 20,25 SAY " Order Cannot be Found !"
   WAIT SPACE(25)+"Press any Key to Continue"
   @ 20,10 CLEAR
ENDIF
* ask the user whether he wants to delete another part
@ 20,15 SAY " Would you like to Delete another Order ? (Y/N) "
answer=""
DO WHILE .NOT. answer$ "YyNn"
   answer=""
   20,70 GET answer
READ
ENDDO
@ 20,10 CLEAR
IF UPER(answer) = "N"
EXIT
ENDIF
* clear the screen
CLEAR
ENDDO
CLOSE DATABASES
RETURN
*EOF:SOM21.PRG

*program:SOM22.PRG
*program used to adjust the hours of the deleted orders from the
*and plan.dbf and new1.dbf and priority.dbf

SET ESCAPE ON
RELEASE ALL LIKE M*
CLOSE DATABASES

SELECT A
USE PRIORITY INDEX SNUMBER

SELECT B
USE NEW1

SELECT C
USE PLAN INDEX WEEKNUM

SELECT A
DO WHILE .NOT. EOF()
   IF DELETED() .AND. scheduled
      IF .NOT. INTERNAL
         msales_num=Sales_num
         mmach_hrs=Mach_hrs
         mass_hrs=Assy_hrs
         SELECT B
         GO TOP
         DO WHILE .NOT. EOF()
            IF Sales_num=msales_num .AND. Section="MACHINING"
               STORE PRODDATE TO mpdate1
               EXIT
            ELSE
               SKIP
            ENDIF
         ENDDO
         GO TOP
         DO WHILE .NOT. EOF()
            IF Sales_num=msales_num .AND. Section="ASSEMBLY"
               STORE PRODDATE TO mpdate2
               EXIT
            ELSE
               SKIP
            ENDIF
         ENDDO
      ENDIF
   ENDDO
   GO TOP
   DO WHILE .NOT. EOF()
      IF WEEKNUMBER=mpdate1 .AND. SECTION="MACHINING"
         hrs1=Hours-mmach_hrs
         REPLACE HOURS WITH hrs1
         EXIT
      ELSE
         SKIP
      ENDIF
   ENDDO
   GO TOP
   DO WHILE .NOT. EOF()
      IF WEEKNUMBER=mpdate2 .AND. SECTION="ASSEMBLY"
         hrs2=Hours-mass_hrs
         REPLACE HOURS WITH hrs2
         EXIT
      ELSE
         SKIP
      ENDIF
   ENDDO
   SELECT A
   SKIP
RELEASE msales_num
RELEASE mmach_hrs
RELEASE mass_hrs
RELEASE mpdate1
RELEASE mpdate2

ELSE
    IF INTERNAL
        change_ntr=1
        msales_num=Sales_num
        mmach_hrs=Mach_hrs
        SELECT B
        GO TOP
        DO WHILE .NOT. EOF()
            IF Sales_num=msales_num .AND. Section="MACHINING"
                STORE PRODDATE TO mpdate1
                EXIT
            ELSE
                SKIP
            ENDIF
        ENDDO
        GO TOP
        DELETE ALL FOR Sales_num=msales_num
        SELECT C
        GO TOP
        DO WHILE .NOT. EOF()
            IF WEEKNUMBER=mpdate1 .AND. SECTION="MACHINING"
                hrs1=Hours-mmach_hrs
                REPLACE HOURS WITH hrs1
                EXIT
            ELSE
                SKIP
            ENDIF
        ENDDO
        SELECT A
        SKIP
        RELEASE msales_num
        RELEASE mmach_hrs
        RELEASE mpdate1
        ENDF

ELSE
    ENDF
    SKIP
ENDDO

SELECT B
PACK

SELECT A
PACK

CLOSE DATABASES

RETURN
*eof : SOM22.PRG

******************************************************************************
* Program: SOM3.PRG  
* reviewing sales orders
******************************************************************************

SET STATUS OFF
SET ESCAPE OFF
SET ECHO OFF
SET TALK OFF
SET BELL OFF
mproduct_id=SPACE(15)
mdue_date=DATE()
DO WHILE .T.
  * erase screen and draw menu
  CLEAR
  @ 2,0 TO 12,79 DOUBLE
  @ 4,1 TO 4,78
  @ 3,25 SAY "REVIEW SALES ORDERS - MENU"
  @ 6,22 SAY " 0 - Exit"
  @ 7,22 SAY " 1 - Review a particular Sales order"
  @ 8,22 SAY " 2 - Browse through Sales Orders Database"
  @ 9,22 SAY " 3 - Review Sales Orders based on product lists"
  @ 10,22 SAY " 4 - Review Sales orders by Duedates"
  @ 12,33 SAY " select | "

  * input valid menu option
  choice ="5"
  DO WHILE .NOT. choice$"01234"
  choice=" "
  @ 12,42 GET choice
  READ
  ENDDO

  * display asterisk next to menu selection
  @ 6+VAL(choice),21 SAY "*"

  * erase other selections from menu
  cnt = 0
  DO WHILE cnt < 5
    IF cnt <> VAL(choice)
      @ 6+cnt,23 SAY SPACE(47)
    ENDIF
    cnt = cnt + 1
  ENDDO

  * test for exit condition
  IF choice$ "0"
    EXIT
  ENDIF

  IF choice$"1"
    CLEAR
    DO SOM31
  ENDIF

  IF choice$"2"
    @15,0 CLEAR TO 17,79
    DO SOM32
  ENDIF

  IF choice$"3"
    SET HEADING OFF
    DO WHILE .T.
      @ 15,10 SAY "Enter the Product id # (Or 0 to Exit) :";
      GET mproduct_id PICTURE "IIIIIIIIIIII/!!!!!
      READ
      IF mproduct_id="0" OR mproduct_id=SPACE(15) OR mproduct_id=" - - - - - - "
        EXIT
      ENDIF
      STORE LTRIM(mproduct_id) TO mproduct_id
IF mproduct_id="0" .OR. LEN(mproduct_id) = 0
    EXIT
ENDIF

USE SALES INDEX SNUM
CLEAR
DO WHILE .T.
    IF PRODUCT_ID=mproduct_id
        CLEAR
        USE SALES INDEX SNUM
        GO TOP
        @ 2, 5 SAY " ALL THE SALES ORDERS BELOW CONTAIN THE PRODUCT:"
        @ 2, 54 SAY mproduct_id PICTURE "B8XXXXXXXXXXX"
        @4, 5 SAY "SALES ORDER # DUE_DATE"
        @5, 5 TO 5, 32 DOUBLE
        LIST ALL "",SALES_NUM,"",DUE_DATE FOR PRODUCT_ID=mproduct_id OFF
        EXIT
    ELSE
        SKIP
        IF EOF()
            CLEAR
            10, 10 SAY "There are no Sales Orders which contain this product"
            WAIT SPACE(17)+"Press any key to continue.."
            EXIT
        ENDIF
        ENDFIELD
    ENDIF
ENDDO

22, 0 CLEAR
choice = ""
DO WHILE .NOT. choice$"yYNn"
    choice = ""
    22, 10 SAY "Do you want to review based on another product? (y/n) :";
    GET choice
    READ
ENDDO
IF UPPER(choice)="W"
    EXIT
ENDIF
mproduct_id=SPACE(15)
CLEAR
ENDDO
SET HEADING ON
ENDIF
IF choice$="y"
    SET HEADING OFF
    DO WHILE .T.
        CLEAR
        12, 10 SAY "Enter the Duedate :";
        GET mdue_date
        READ
        USE PRIORITY INDEX SNUMBER
        CLEAR
        DO WHILE .T.
            IF DUE_DATE=mdue_date
                CLEAR
                USE PRIORITY INDEX SNUMBER
                GO TOP
                @ 2, 5 SAY "All the Sales Orders listed below are Due on :"
                @ 2, 54 SAY mdue_date
                @4, 5 SAY "SALES ORDER #:
                @5, 5 TO 5, 18 DOUBLE
                LIST ALL "",SALES_NUM FOR DUE_DATE=mdue_date OFF
                EXIT
            ELSE
                ...
IF EOF()
  CLEAR
  10,10 SAY "There are no Sales Orders Due on: "
  10,46 SAY mdue_date
  WAIT SPACE(17)"Press any key to continue.."
  EXIT
ENDIF
ENDIF
ENDDO

Q22,0 CLEAR
choice =""
DO WHILE .NOT. choice$ "yYnN"
  choice="H"
  Q22,10 SAY "Do you want to review based on another due date (y/n) : ";
  GET choice
  READ
ENDDO
IF UPPER(choice)="N"
  EXIT
ENDIF
mdue_date=DATE()
CLEAR
ENDDO
SET HEADING ON
ENDIF
ENDDO
CLOSE DATABASES
RETURN
*eof:SOM3.PRG

*********************************************************************
*program :SOM31.PRG
*sales order review-option
*********************************************************************

SET HEADING OFF
DO WHILE .T.
  CLEAR
  *initialize all memory variables
  msales_num=SPACE(15)
  *set up exit condition
  2,15 SAY " DATABASE REVIEW WINDOW "
  3,15 TO 3,60 DOUBLE
  10,10 SAY " Enter Sales Order Number [Or 0 to Quit] : ";
  GET msales_num PICTURE "111111111111111"
  READ
  *test for exit
  IF msales_num = "0" .OR. msales_num = SPACE(15)
    WAIT SPACE(13)"Press any key to return to Menu "
    CLOSE DATABASES
    RETURN
  ENDFI
  STORE LTRIM(msales_num) TO msales_num
  IF msales_num="0" .OR. LNK(msales_num)=0
    WAIT SPACE(13)"Press any key to return to Menu "
    CLOSE DATABASES
    RETURN
  ENDFI
  *search the database for the record to be displayed
  4,1 CLEAR TO 17,79
  USE PRIORITY INDEX SNUMBER
SEEK msales_num
IF FOUND()
  04,2 SAY "Sales Order Number :"
  04,24 SAY Sales_num PICTURE "98XXXXXXXXXXXXXXXX"
  04,64 SAY "Internal :"
  04,75 say Internal
  05,2 SAY "Priority :"
  05,16 SAY Priority PICTURE "9BAXX"
  04,41 SAY "Due Date :"
  04,53 SAY Due_date
  05,21 SAY "Comments :"
  05,33 SAY Comments
ELSE
  CLEAR
  02,10 SAY "Sales Order cannot be found in the Database !"
  WAIT SPACE(20)+"Press any key to continue..."

  answer=""
  DO WHILE .NOT. answer$"yYnN"
    answer=""
    WAIT SPACE(10)+"Would you like to review another order? (y/n) :";
    TO answer
    ENDDO
    IF UPPER(answer)="N"
      EXIT
    ENDDO
  CLEAR
  LOOP
ENDIF
USE
USE SALES INDEX SNUM
SEEK msales_num
IF FOUND()
  * if found, display the fields
  06,0 TO 7,79
  07,10 SAY "PRODUCT ID #"
  07,27 SAY "QUANTITY"
  07,40 SAY "COMMENTS"
  08,0 TO 8,79 DOUBLE
  DISPLAY "",Product_id,Quantity,"",Comments FOR Sales_num=msales_num
ELSE
  CLEAR
  20,10 SAY "Sales Order cannot be found in the Database !"
  WAIT SPACE(20)+"Press any key to continue..."
ENDIF
answer=""
DO WHILE .NOT. answer$"yYnN"
  answer=""
  WAIT SPACE(15)+"Would you like to review another order? (Y/N) :";
  TO answer
  ENDDO
  IF UPPER(answer)="N"
    EXIT
  ENDDO
  CLEAR
ENDDO
*WAIT SPACE(13)+"Press any key to return to Menu"
CLOSE DATABASES
RELEASE ALL
RETURN
*EOF:SON31.PRG

*====================================================================
program: SOM32.PRG
* review sales orders-option

SET HEADING OFF
msales_num=SPACE(15)
SELECT A
USE PRIORITY INDEX SNUMBER
SELECT B
USE SALES INDEX SNUM
SELECT A
DO WHILE .T.
CLEAR
22,30 SAY "REVIEW SALES ORDERS"
23,30 TO 3,48 DOUBLE
STORE SALES_NUM TO msales_num
25,2 SAY "Sales Order #:"
25,17 SAY Sales_num PICTURE "98XXXXXXXXXXXXXXXX"
25,60 say "Internal :"
25,71 say Internal
26,2 SAY "Priority :"
26,13 SAY Priority PICTURE "2BAXX"
25,36 SAY "Due Date :"
25,48 SAY Due_date
26,18 SAY "Comments :"
26,30 SAY Comments
SELECT B
GO TOP
SEEK msales_num
IF FOUND()
@ 7,0 TO 7,79
@ 8,10 SAY "Product Id #"
@ 8,28 SAY "Quantity"
@ 8,40 SAY "Comments"
@ 9,0 TO 9,79 DOUBLE
DISPLAY ",Product_id," ",Quantity," ",Comments FOR Sales_num=msales_num OFF
ELSE
CLEAR
@ 20,10 SAY " Sales Order cannot be found in the Database !"
WAIT SPACE(20)+"Press any key to continue.. ",
ENDIF
choice=" 
DO WHILE .NOT. choice$"Yynm"
choice=" 
WAIT SPACE(10)+"Do you want to continue to the next Sales Order? (y/n) :"
TO choice
ENDDO
IF UPPER(choice)="N"
EXIT
ENDIF
CLEAR
SELECT A
SKIP
IF EOF()
@23,20 SAY "No more Sales Orders in the database !"
WAIT SPACE(24)+"Press any key to return to Menu"
EXIT
ENDIF
ENDDO
CLOSE DATABASES
RELEASE ALL
RETURN
*EOF: SOM32.PRG

*=================================================================================================================================
*PROGRAM: SOM4.PRG
*sales orders changing routine
*=================================================================================================================================

SELECT A
USE SALES INDEX SNUM
SELECT B
USE PRIORITY INDEX SNUMBER
SELECT C
USE NEW1 INDEX NUMB
REINDEX

SELECT A

DO WHILE .T.
  CLEAR
  *initialize the memory variables
  msales_num=SPACE(15)
  mproduct_id=SPACE(15)
  mquantity=0
  mpriority=SPACE(3)
  mdue_date=DATE()
  mcomments=SPACE(45)
  mcomment1=SPACE(10)
  osales_num=SPACE(15)
  mfinalised= .F.
  mfinalweek= DATE()
  indicator=0
  minternal=.F.
  monk=.T.
  DO WHILE monk
    * Set up the working environment
    @ 1,1 TO 4,79 DOUBLE
    @ 5,1 TO 15,79
    @ 2,25 SAY "SALES ORDER CHANGING ROUTINE"
    @ 3,25 TO 3,52
    *input sales order# of the order whose specs. are to changed
    @12,10 SAY "Enter Sales Order Number (OR 0 to Exit) :";
    GET osales_num PICTURE "!!!!/111111111"
    READ
    * Test for exit condition
    IF osales_num="0" OR. osales_num=SPACE(15)
      WAIT SPACE(22)+"Press any key to return to menu"
      CLOSE DATABASES
      RETURN
    ENDIF
    STORE LTRIM(osales_num) TO osales_num
    IF osales_num = "0"
      WAIT SPACE(22)+"Press any key to return to menu"
      CLOSE DATABASES
      RETURN
    ENDIF
    * Find the sales order to be changed
    SELECT A
    SEEK osales_num
    IF FOUND()
      * If found, display the information and confirm
      CLEAR
      @ 9,1 TO 13,79 DOUBLE
      @ 11,25 SAY "Sales Order Number : 
      @ 11,48 SAY Sales_num PICTURE "@BXXXXXXXXXXXXXXX"
* ask the user whether this is the right order
  answer=" "
  DO WHILE .NOT. answer$ "Y/N"
     answer=" "
     20,15 SAY "Is this the Sales Order you want to change? (Y/N)?";
     GET answer
     READ
  ENDDO
  20,5 CLEAR
  IF UPPER(answer)="Y"
    EXIT
  ELSE
    CLEAR
    22,20 SAY "Please type in correct Sales Order Number "
    WAIT SPACE(26)+"Press any key to continue..."
  ENDIF
  ELSE
    CLEAR
    20,19 SAY "Sales Order not found in database"
    WAIT SPACE(23)+"Press any key to reenter"
    CLEAR
    "end of if-endif loop
  ENDIF
ENDDO

STORE Sales_num TO osales_num
STORE Sales_num TO msaless_num
STORE Product_id TO mproduct_id
STORE Quantity TO mquantity
STORE Comments TO mcomment1
SELECT B
SEEK osales_num
IF FOUND()
  IF SCHEDULED
    REPLACE CHANGED WITH .T.
  ENDIF
  STORE Priority TO mpriority
  STORE Due_date TO mdue_date
  STORE Comments TO mcomments
  STORE INTERNAL TO minternal
ENDIF

CLEAR
DO WHILE .T.,
  01,0 CLEAR TO 21,79
  1,14 TO 3,55
  2,15 SAY "Go ahead and make the necessary Changes"
  10,10 SAY "Sales Order Number [OR 0 to Quit]: ";
  GET msaless_num PICTURE "9999999999m
  READ
  22,0
  IF msaless_num=0 .OR. msaless_num=SPACE(15)
    WAIT SPACE(13)+"Press any key to continue -- nothing was changed"
    EXIT
  ENDIF
  STORE LTRIM(msales_num) TO msaless_num
  IF msaless_num=0 .OR. msaless_num=SPACE(15)
    WAIT SPACE(13)+"Press any key to continue -- nothing was changed"
    EXIT
  ENDIF
  IF osales_num<>msales_num
    SELECT B
    SEEK msaless_num
  ENDIF
IF FOUND()
  @22,10 SAY "This Sales Order already exists - Give new Number"
  LOOP
ENDIF
ENDIF

a 10,0 CLEAR
a 10,10 SAY "Sales Order #:"
@ 11,10 SAY "Sales Order #:"
@ 10,27 SAY msales_num
a 10,45 SAY "Internal (y/n):" GET minternal
a 12,10 SAY "Priority :"
a 12,23 GET mpriority PICTURE "!99"
@ 12,30 SAY "Due date :"
@ 12,42 GET mdue_date
a 14,10 SAY "Comments :"
@ 14,22 GET mcomments
a 16,10 SAY "Product Id Number :"
@ 16,30 GET mproduct_id PICTURE "!!!!!!!!!!!!!!"
a 18,10 SAY "Quantity :"
@ 18,27 GET mquantity PICTURE "9999"
@ 20,10 SAY "Comments :" GET mcomment1
READ
* CHECK whether the information is correct
answer=""
DO WHILE .NOT. answer$ "yYnN"
  a 22,10 SAY "Is this information correct ? (y/n)";
  GET answer
  READ
ENDDO
a 20,10 CLEAR TO 23,79
IF UPPER(answer)="Y"
  @22,30 SAY "Please Wait..."
  * now we can replace the old values by the new ones
  SELECT A
    IF osales_num <> msales_num
      DO WHILE .T.
        GO TOP
        SEEK osales_num
        IF FOUND()
          REPLACE SALES_NUM WITH UPPER(msales_num)
          ELSE
            IF EOF() THEN
              EXIT
          ENDIF
        ENDDO
      ENDIF
    ENDIF
  ENDIF
ENDDO
ENDIF

SELECT A
GO TOP
SEEK msales_num
IF FOUND ()
  REPLACE Product_id WITH UPPER(mproduct_id)
  REPLACE Quantity WITH mquantity
  REPLACE DUE_DATE WITH mdue_date
  REPLACE COMMENTS WITH mcomment1
  SKIP
ENDIF
SELECT B
GO TOP
SEEK osales_num
IF FOUND ()
  REPLACE SALES_NUM WITH UPPER(msales_num)
REPLACE PRIORITY WITH UPPER(mpriority)
REPLACE DUE_DATE WITH mdue_date
REPLACE MFG_DATE WITH mdue_date
REPLACE COMMENTS WITH mcomments
REPLACE INTERNAL WITH minternal
ENDIF

SELECT C
IF osales_num <> msales_num
    DO WHILE .T.
        GO TOP
        SEEK osales_num
        IF FOUND()
            REPLACE SALES_NUM WITH UPPER(msales_num)
        ELSE
            IF EOF()
                EXIT
            ENDIF
        ENDIF
    ENDDO
ENDIF

*Start of process to change all products in this order
SELECT A
DO WHILE .T.
    IF Sales_num=msales_num
        mproduct_id=Product_id
        mquantity=Quantity
        mcomment1=COMMENTS
        CLEAR
        @ 3,1 TO 17,79
        @ 5,30 SAY "NEXT PRODUCT"
        @ 6,30 SAY "--------------------------"
        @ 8,10 SAY "SALES ORDER NUMBER : "
        @ 8,32 SAY msales_num PICTURE "990000000000000000"
        @ 8,48 SAY "CONTINUED ..."
        @ 9,10 SAY "--------------------------"
        @ 9,48 SAY "---------"
        @ 10,10 SAY "PRODUCT ID NUMBER [Enter 0 to Quit] :";
        GET mproduct_id PICTURE "990000000000000000"
        READ
        IF mproduct_id="0" .OR. mproduct_id=SPACE(15)
            indicator=1
            EXIT
        ENDIF
        @ 12,10 SAY "QUANTITY :";
        GET mquantity PICTURE "9999"
        @ 14,10 SAY "COMMENTS :" GET mcomment1
        READ
        REPLACE PRODUCT_ID WITH UPPER(mproduct_id)
        REPLACE QUANTITY WITH mquantity
        REPLACE DUE_DATE WITH mdue_date
        REPLACE COMMENTS WITH mcomment1
        ELSE
            EXIT
        ENDIF
    SKIP
    IF EOF()
        EXIT
    ENDIF
ENDDO

If indicator=1
    WAIT SPACE(13)+"Press any key to continue - requested changes completed "
EXIT
ENDIF
@20,10 SAY "No more products in this sales order"
request=" "
DO WHILE .NOT. request$"YyNn"
    request=" "
    @21,10 SAY "Would you like to Add a Product into this Sales Order? (y/n)"
    GET request
    READ
ENDDO
IF UPPER(request)="Y"
    STORE SPACE(15) TO mproduct_id
    STORE 0 TO mquantity
    STORE SPACE(10) TO mcomment1
    CLEAR
    DO WHILE .T.
        @5,5 TO 17,75 DOUBLE
        @7,10 SAY "Adding a Product into Sales Order :"
        @7,46 SAY msales_num PICTURE "@BXXXXXXXXXXXXXXX"
        @8,10 TO 8,61
        @10,10 SAY "Enter Product ID # (OR 0 to Quit) :";
        GET mproduct_id PICTURE "!!!!!!!!!!!!!!!"
        READ
        IF mproduct_id=SPACE(15) .OR. mproduct_id="0" .OR.
            EXIT
        ENDIF
        STORE LTRIM(mproduct_id) TO mproduct_id
        IF mproduct_id="0"
            EXIT
        ENDIF
        @12,10 SAY "Enter Quantity :"
        GET mquantity PICTURE "9999" RANGE 0,
        @ 14,10 SAY "COMMENTS :" GET mcomment1
        READ
        correct=" 
        DO WHILE .NOT. correct$"YyNn"
            correct=" 
            @ 20,20 SAY "Is this information correct? (y/n)"
            GET correct
            READ
        ENDDO
        IF UPPER(correct)="Y"
            SELECT A
            APPEND BLANK
            REPLACE SALES_NUM WITH UPPER(msales_num)
            REPLACE PRODUCT_ID WITH UPPER(mproduct_id)
            REPLACE QUANTITY WITH mquantity
            REPLACE DUE_DATE WITH mdue_date
            REPLACE COMMENTS WITH mcomment1
            ELSE
                @22,20 SAY " Please type in correct information "
                @23,20 SAY "------------------------------------------"
            ENDIF
            LOOP
        ENDIF
        checker=" 
        @21,0 CLEAR TO 23,79
        DO WHILE .NOT. checker$"YyNn"
            checker=" 
            @20,10 SAY "Do you want to add another product into this sales order? (y/n)"
            GET checker
            READ
        ENDDO
IF UPPER(checker)="N"
   EXIT
ENDIF
STORE SPACE(15) TO mproduct_id
STORE 0 TO mquantity
STORE SPACE(10) TO mcomment1
@20,0 CLEAR TO 23,79
ENDDO
ENDIF
EXIT
ELSE
   WAIT SPACE(20)+"Press any key to type in correct data.."
ENDIF
ENDDO
@ 20,10 CLEAR
choice=" "
DO WHILE .NOT. choice$"yYnN"
   choice=" "
   @ 23,13 SAY " Do you want to change another Sales Order ? (y/n)" ;
   GET choice
   READ
ENDDO
IF UPPER(choice)="N"
   EXIT
ENDIF
ENDDO
CLOSE DATABASES
RETURN
*EOF:SOM4.PRG

*================================================================================================== *
* Program: SOM41.PRG
*
* sales orders reports routine
*================================================================================================== *

SET STATUS OFF
SET ESCAPE OFF
SET ECHO OFF
SET TALK OFF
SET BELL OFF
begin=DATE()
end=DATE()
DO WHILE .T.
   *
   * erase screen and draw menu
   CLEAR
   @ 2,0 TO 12,79 DOUBLE
   @ 4,1 to 4,78
   @ 3,24 SAY "SALES ORDERS REPORTS MENU"
   @ 6,27 SAY " 0 - Exit"
   @ 7,27 SAY " 1 - Report on all Sales Orders"
   @ 8,27 SAY " 2 - Report Sales Orders based on a Duedate"
   @ 12,33 SAY " select | | "
   *
   * input valid menu option
   choice ="6"
   DO WHILE .NOT. choice$"012"
      choice=" "
      @ 12,42 GET choice
      READ
   ENDDO
   *
   *
* display asterisk next to menu selection
  @ 6+VAL(choice),26 SAY "***"

* erase other selections from menu
  cnt = 0
  DO WHILE cnt < 6
    IF cnt <> VAL(choice)
      @ 6+cnt,28 SAY SPACE(45)
    ENDIF
    cnt = cnt + 1
  ENDDO

* test for exit condition
  IF choice $ " O"
    EXIT
  ENDF

IF choice$ "1"
  @16,0
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  DO SOM412
    WAIT SPACE(15)+"Press any key to continue..."
  ENDF

IF choice$ "2"
  ERASE TEMPSALES.DBF
  ERASE SNO.NDX
  ERASE PRO.NDX
  ERASE TEMPPRIO.DBF
  @15,10 SAY "DUE DATE :" GET begin
  READ @15,0
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  USE SALES INDEX SNUM
  COPY TO TEMPSALES FOR DUE_DATE=begin
  USE TEMPSALES
    INDEX ON SALES_NUM TO SNO
  USE PRIORITY INDEX SNUMBER
  COPY TO TEMPPRIO FOR DUE_DATE=begin
  USE TEMPPRIO
    INDEX ON SALES_NUM TO PRO
  DO SOM411
    ERASE TEMPSALES.DBF
    ERASE SNO.NDX
    ERASE PRO.NDX
    ERASE TEMPPRIO.DBF
    WAIT SPACE(15)+"Press any key to continue..
  ENDF

ENDDO
RETURN
*eof:SOM411.PRG

==============================================
*Program: SOM411.PRG
*program to print out sales orders and their details
==============================================

EJECT
SELECT A
USE TEMPSALES INDEX SNO
SELECT B
USE TEMPPRIO INDEX PRO
SELECT B
SET DEVICE TO PRINT
line=7
subline=""  
pager=1
timer=TIME()
page="Page No."
DO WHILE .T.
   02,1 SAY DATE()
   02,10 SAY timer
   02,66 say page
   02,77 SAY pager PICTURE "@8999"
   03,15 SAY "ADEC DENTAL FURNITURE -- SALES ORDERS REPORT"
   04,15 SAY "="
   05,13 say "Sales Orders Report based on DueDate :"
   05,53 say Due_date
   06,1 say subline
   DO WHILE .NOT. EOF()
      STORE SALES_NUM TO msales_num
      STORE PRIORITY TO mpriority
      STORE INTERNAL TO minternal
      STORE DUE_DATE TO mdue_date
      STORE COMMENTS TO mcomments
      SELECT A
      SEEK msales_num
      IF FOUND()
         aline,1 SAY "Sales Order #"
         aline,15 SAY msales_num
         aline,32 SAY "Due Date:"
         aline,41 SAY mdue_date
         aline,52 SAY "Priority:"
         aline,61 SAY mpriority
         aline,65 say "Internal:"
         aline,75 say minternal
         line=line+1
         aline,1 SAY "Comments:"
         aline,10 SAY mcomments
         line=line+2
         aline,10 SAY "Product Id"
         aline,30 SAY "Quantity"
         aline,40 SAY "Comments"
         aline+1,10 SAY "="
         line=line+2
      DO WHILE SALES_NUM=msales_num
         aline,10 SAY Product_id
         aline,30 say Quantity
         aline,40 SAY Comments
         line=line+1
      IF line>=56
         EJECT
         pager=pager+1
         02,1 SAY DATE()
         02,10 SAY timer
         02,66 say page
         02,77 SAY pager PICTURE "@8999"
         03,15 SAY "ADEC DENTAL FURNITURE -- SALES ORDERS REPORT"
         04,15 SAY "="
         05,13 say "Sales Orders Report based on DueDate :"
         05,53 say mdue_date
         07,1 say "Continued from previous page..."
         line=8
         aline,1 SAY "Sales Order #"
&line,15 SAY msales_num
&line,32 SAY "Due Date:"
&line,41 SAY mdue_date
&line,52 SAY "Priority:"
&line,61 SAY mpriority
&line,65 say "Internal:"
&line,75 say minternal
line=line+1
&line,1 SAY "Comments:"
&line,10 SAY mcomments
line=line+2
&line,10 SAY "Product Id"
&line,30 SAY "Quantity"
&line,40 SAY "Comments"
&line+1,10 SAY "-----------------------------------"
line=line+2

ENDIF
ENDIF
ENDDO

SELECT A
USE SALES INDEX SNUM
SELECT B
USE PRIORITY INDEX SNUMBER
SELECT B
SET DEVICE TO PRINT
line=7
subline="-----------------------------------"
pager=1
timer=TIME()
pagenumber="Page No."
DO WHILE .T.
a2,1 SAY DATE()
a2,10 SAY timer
a2,66 say page
a2,77 SAY pager PICTURE "@9999"

*------------------------------------------------------------------ *
*PROGRAM: SOM411.PRG *
*program to print out sales orders and their details *
*------------------------------------------------------------------ *

CLOSE ALL
SELECT A
USE SALES INDEX SNUM
SELECT B
USE PRIORITY INDEX SNUMBER
SELECT B
SET DEVICE TO SCREEN
RETURN
*EOF:SOM412.PRG
DO WHILE .NOT. EOF()
  STORE SALES_NUM TO msales_num
  STORE PRIORITY TO mpriority
  STORE DUE_DATE TO mdue_date
  STORE COMMENTS TO mcomments
  SELECT A
  SEEK msales_num
  IF FOUND()
    aline,1 SAY "Sales Order #"
    aline,15 SAY msales_num
    aline,32 SAY "Due Date:"
    aline,41 SAY mdue_date
    aline,52 SAY "Priority:"
    aline,61 SAY mpriority
    aline,65 say "Internal:"
    aline,75 say minternal
    line=line+1
    aline,1 SAY "Comments:"
    aline,10 SAY mcomments
    line=line+2
    aline,10 SAY "Product Id"
    aline,30 SAY "Quantity"
    aline,40 SAY "Comments"
    aline=1,line+1
    IF line>=56
      EJECT
      pager=pager+1
      a2,1 SAY DATE()
      a2,10 SAY timer
      a2,66 say page
      a2,77 SAY pager PICTURE "9999"
    a3,15 SAY "ADEC DENTAL FURNITURE -- SALES ORDERS REPORT"
    a4,15 SAY "_________________________________________"
    a5,22 say "Report on all the Sales Orders"
    a6,1 say subline
    DO WHILE SALES_NUM=msales_num
      aline,10 SAY Product_id
      aline,30 say Quantity
      aline,40 SAY Comments
      line=line+1
      IF line>=56
        EJECT
        pager=pager+1
        a2,1 SAY DATE()
        a2,10 SAY timer
        a2,66 say page
        a2,77 SAY pager PICTURE "9999"
        a3,15 SAY "ADEC DENTAL FURNITURE -- SALES ORDERS REPORT"
        a4,15 SAY "_________________________________________"
        a5,22 say "Report on all the Sales Orders"
        a7,1 say "Continued from previous page..."
        line=8
        aline,1 SAY "Sales Order #"
        aline,15 SAY msales_num
        aline,32 SAY "Due Date:"
        aline,41 SAY mdue_date
        aline,52 SAY "Priority:"
        aline,61 SAY mpriority
        aline,65 say "Internal:"
        aline,75 say minternal
        line=line+1
        aline,1 SAY "Comments:"
        aline,10 SAY mcomments
        line=line+2
        aline,10 SAY "Product Id"
        aline,30 SAY "Quantity"
        aline,40 SAY "Comments"
        aline=1,line+1
      ENDIF
CLEAR ALL

* This segment of the program gives the user some introduction to the program and it's working and underlying assumptions.

CLEAR
S1,1 TO 4,78 DOUBLE
S2,22 SAY "Schedule Generation Routine"
S3,22 to 3,48
S5,1 TO 17,78
S7,10 SAY "This routine processes all the sales orders and provides the"
S8,10 SAY "user with the Scheduled Production Hour requirements for"
S9,10 SAY "all the sections of the facility on a weekly basis."
S11,10 SAY "The System assumes that the total elapsed production time for"
S12,10 SAY "any product to be 2 weeks. And also that any product will"
S13,10 SAY "spend a maximum of one week in any section."

answer=""
DO WHILE .NOT. answer$"YyNn"
  answer="
  S18,20 SAY "DO YOU WANT TO CONTINUE ? (Y/N) ";
  GET answer
  READ
ENDDO
IF UPPER(answer)="N"
CLOSE ALL
RETURN
ENDIF
* THE ACTUAL PROCESSING OF ALL THE SALESORDERS CONTINUES HERE. *

*all the old information is deleted so that it can be replaced with updated information.

CLEAR
@ 13,15 TO 17,52
@ 15,20 SAY "PROCESSING - PLEASE WAIT..."

SELECT A
USE PRIORITY INDEX SNUMBER

SELECT B
USE NEW1

SELECT C
USE PLAN INDEX WEEKNUM

SELECT A
DO WHILE .NOT. EOF()
  IF Changed
    IF .NOT. INTERNAL
      REPLACE SCHEDULED WITH .F.
      REPLACE CHANGED WITH .F.
      msales_num=Sales_num
      mmach_hrs=Mach_hrs
      mass3rs=Assy_hrs
    SELECT B
  GO TOP
  DO WHILE .NOT. EOF()
    IF Sales_num=msales_num .AND. Section="MACHINING"
      STORE PRODDATE TO mpdate1
      EXIT
    ELSE
      SKIP
  ENDDO
ENDIF
ENDDO

GO TOP
DO WHILE .NOT. EOF()
  IF Sales_num=msales_num .AND. Section="ASSEMBLY"
    STORE PRODDATE TO mpdate2
    EXIT
  ELSE
    SKIP
  ENDDO
ENDIF

GO TOP
DELETE ALL FOR Sales_num=msales_num
SELECT C
GO TOP
DO WHILE .NOT. EOF()
  IF WEEKNUMBER=mpdate1 .AND. SECTION="MACHINING"
    hrs1=Hours-mmach_hrs
    REPLACE HOURS WITH hrs1
    EXIT
  ELSE
    SKIP
  ENDDO
ENDIF
ENDDO
GO TOP
DO WHILE .NOT. EOF()
   IF WEEKNUMBER=mpdate2 .AND. SECTION="ASSEMBLY"
      hrs2=Hours-mass_hrs
      REPLACE HOURS WITH hrs2
      EXIT
   ELSE
      SKIP
   ENDIF
ENDDO
SELECT A
SKIP
RELEASE msales_num
RELEASE mmach_hrs
RELEASE mass_hrs
RELEASE mpdate1
RELEASE mpdate2
ELSE
   IF INTERNAL
      REPLACE SCHEDULED WITH .F.
      REPLACE CHANGED WITH .F.
      change_ntr=1
      msales_num=Sales_num
      mmach_hrs=Mach_hrs
   SELECT B
   GO TOP
   DO WHILE .NOT. EOF()
      IF Sales_num=msales_num .AND. Section="MACHINING"
         STORE PROODDATE TO mpdate1
         EXIT
      ELSE
         SKIP
      ENDIF
   ENDIF
ENDDO
GO TOP
DELETE ALL FOR Sales_num=msales_num
SELECT C
GO TOP
DO WHILE .NOT. EOF()
   IF WEEKNUMBER=mpdate1 .AND. SECTION="MACHINING"
      hrs1=Hours-mmach_hrs
      REPLACE HOURS WITH hrs1
      EXIT
   ELSE
      SKIP
   ENDIF
ENDDO
SELECT A
SKIP
RELEASE msales_num
RELEASE mmach_hrs
RELEASE mmach_hrs
RELEASE mpdate1
ENDIF
ELSE
   SKIP
ENDIF
ENDDO
SELECT B
PACK
CLOSE DATABASES
DO SI1
CLEAR ALL
RETURN
*eof : SI.PRG

*=================================================================
*program :SI1.PRG
*one of the schedule generation routines
*=================================================================

CLOSE DATABASES

SELECT A
USE PRIORITY INDEX SNUMBER

SELECT B
USE SALES INDEX SNUM

SELECT C
USE PRODUCTS INDEX PRODS

SELECT D
USE PLAN INDEX WEEKNUM

SELECT E
USE NEW1

SELECT A

DO WHILE .T.
*initialize all the memory variables
mproduct_id = SPACE(15)
minternal=.F.
delta=DATE()
mweek1=DATE()
mweek2=DATE()
mmfg_date=DATE()
mcomments=SPACE(10)
mrecnum=1
mquantity=0
msystem_id=0
msystem_ass=0
msystem_flp=0
msystem_pckg=0
counter1=0
counter2=0
mscheduled=.T.
answer=""
choice=""
mrecord=0
mpriority=SPACE(3)
DO WHILE .T.
   IF .NOT. Scheduled
      msales_num=Sales_num
      mpriority=Priority
      STORE DUE_DATE TO mdue_date
      STORE MFG_DATE TO mmfg_date
      STORE INTERNAL TO minternal
      EXIT
   ELSE
      IF EOF()
         EXIT
         EXIT
*These three rules determine the planning horizon based on the due dates
* It assumes that the due date will either be the friday of the week or
*the thursday or the wednesday.
IF minternal
  mweek1=mmfg_date
ELSE
  IF CDOW(mmfg_date)="Friday"
    mweek1=mmfg_date-7
    mweek2=mmfg_date
  ELSE
    IF CDOW(mmfg_date)="Thursday"
      mweek1=mmfg_date-6
      mweek2=mmfg_date
    ELSE
      IF CDOW(mmfg_date)="Wednesday"
        mweek1=mmfg_date-5
        mweek2=mmfg_date
      ELSE
        IF CDOW(mmfg_date)="Tuesday"
          mweek1=mmfg_date-4
          mweek2=mmfg_date
        ELSE
          IF CDOW(mmfg_date)="Monday"
            mweek1=mmfg_date-3
            mweek2=mmfg_date
        ENDIF
    ENDIF
  ENDIF
ENDIF
ENDIF
ENDIF

SELECT B
SEEK msales_num
IF FOUND()
  DO WHILE Sales_num=msales_num
    STORE Product_id TO mproduct_id
    STORE Quantity TO mquantity
    STORE Quantity TO mqty
    STORE COMMENTS TO mcomments
    SELECT C
    SEEK mproduct_id
    IF FOUND()
      STORE Stime_mc TO mstime_mc
      STORE Stime_ass TO mstime_ass
      STORE Stime_flp TO mstime_flp
      STORE Stime_pckg TO mstime_pckg
      counter1 = (counter1 + (mstime_mc*mquantity))
    ENDIF
    counter2 =
counter4 = counter2 + (mstime_ass * mquantity) + (mstime_flp * mquantity) + (mstime_pckg * mquantity) # used to append info into new1.dbf for reports
SELECT E
APPEND BLANK
REPLACE SALES_NUM WITH msales_num
REPLACE PRIORITY WITH mpriority
REPLACE DUE_DATE WITH mdue_date
REPLACE PRODUCT_ID WITH mproduct_id
REPLACE QUANTITY WITH mqty
REPLACE PRODDATE WITH mweek1
REPLACE SECTION WITH "MACHINING"
REPLACE COMMENTS WITH mcomments
REPLACE HOURS WITH counter3

IF NOT. minternal
APPEND BLANK
REPLACE SALES_NUM WITH msales_num
REPLACE PRIORITY WITH mpriority
REPLACE DUE_DATE WITH mdue_date
REPLACE PRODUCT_ID WITH mproduct_id
REPLACE QUANTITY WITH mqty
REPLACE PRODDATE WITH mweek1
REPLACE SECTION WITH "MACHINING"
REPLACE COMMENTS WITH mcomments
REPLACE HOURS WITH counter3
ENDIF

RELEASE mstime_mc
RELEASE mstime_ass
RELEASE mstime_flp
RELEASE mstime_pckg
RELEASE mquantity

SELECT B
SKIP
ELSE

IF EOF()
CLEAR
@2,1 TO 10,79
@5,10 SAY "PRODUCT ID NUMBER : "
@5,52 SAY mproduct_id PICTURE "GXXXXXXXXXXXXXXX"
@6,10 SAY "THIS PRODUCT DOES NOT EXIST IN THE DATABASE"
@7,10 SAY "YOU NEED TO ADD THIS RIGHT NOW TO CONTINUE"
@12,0 CLEAR
answer=""
DO WHILE .NOT. answer$ "yYnN"
answer=""
@14,10 SAY "DO YOU WANT TO CONTINUE ? (Y/N) ?";
GET answer
@16,10 SAY "NOTE : If you say N, Program will terminate right now"
READ
ENDDO
IF answer$ "yY"
CLEAR
DO SOM13 WITH mproduct_id
@13,15 TO 17,52
@15,20 SAY "PROCESSING - PLEASE WAIT..."
SELECT B
ELSE
IF answer$ "nN"
clear all


WAIT SPACE(13)+"PROGRAM ABORTED -- PRESS A KEY TO RETURN TO MENU"
RETURN
ENDIF
ENDIF
ENDIF
ENDDO
ELSE
IF EOF()
EXIT
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
END
IF .NOT. minternal
SELECT D
SEEK mweek2
IF FOUND()
DO WHILE .T.
IF Weeknumber=mweek2 .AND. Section="ASSEMBLY"
REPLACE SECTION WITH "ASSEMBLY"
REPLACE HOURS WITH Hours+counter2
IF CAPACITY=0
REPLACE CAPACITY WITH 700
ENDIF
EXIT
ELSE
IF WEEKNUMBER=mweek2 .AND. SECTION=SPACE(15)
REPLACE SECTION WITH "ASSEMBLY"
REPLACE HOURS WITH Hours+counter2
IF CAPACITY=0
REPLACE CAPACITY WITH 700
ENDIF
EXIT
ELSE
IF EOF()
APPEND BLANK
REPLACE Weeknumber WITH mweek2
REPLACE Section WITH "ASSEMBLY"
REPLACE Hours WITH counter2
REPLACE MONTH WITH CMONTH(mweek2)
REPLACE CAPACITY WITH 700
EXIT
ENDIF
ENDIF
ENDIF
ENDDO
ELSE
IF EOF()
APPEND BLANK
REPLACE Weeknumber WITH mweek2
REPLACE Section WITH "ASSEMBLY"
REPLACE Hours WITH counter2
REPLACE MONTH WITH CMONTH(mweek2)
REPLACE CAPACITY WITH 700
ENDIF
ENDIF
ENDIF
SELECT A
REPLACE SCHEDULED WITH mscheduled
REPLACE MACH_HRS WITH counter1
REPLACE ASSY_HRS WITH counter2
SKIP
IF EOF()
EXIT
ENDIF
ENDDO
CLOSE ALL
@15,20 SAY SPACE(30)
@15,23 SAY "PROCESS COMPLETED!"
@18,0 CLEAR
WAIT SPACE(20)+"Press any key to continue..."
USE PLAN INDEX WEEKNUM
DELETE ALL FOR HOURS=0
PACK
USE
DO SI11
CLEAR
answer = " "
DO WHILE .NOT. answer$"YnN"
        answer = " ",
        run be window 1,1,23,78 white on blue explode
        &10,10 SAY "WOULD YOU LIKE TO BALANCE THE SCHEDULED "+
        "WORK LOAD ? (Y/N) ";
        GET answer
        READ
ENDDO

IF answer$"Yy"
        clear

DO WHILE .T.
        choice = " ",
        DO WHILE .NOT. choice$"ilaqO"
        choice = " ",
        *run be window 2,0,24,79 bright yellow on blue explode
        *run be window 3,24,5,56 bright black on white explode
        run be window 4,11,20,68 white explode
        &24,28 SAY "Work Load Balancing Menu"
        &10,25 SAY "[A]utomatic Work Load Balancing"
        &11,25 SAY "[I]nteractive Work Load Balancing"
        &12,25 SAY "[Q]uit"
        &18,20 SAY "Press a letter to select an option ";
        GET choice
        READ
ENDDO

        IF choice$"qO"
            exit
        ENDIF

        IF choice$"aA"
            clear
            run thesis1
        ENDIF

        IF choice$"il"
            clear
            DO SI12
        ENDIF

clear
ENDDO

ENDIF

CLEAR ALL
RETURN
*EOF:SI11.PRG
SET STATUS OFF
SET ESCAPE OFF
SET ECHO OFF
SET TALK OFF
SET BELL OFF
CLEAR ALL
begin=DATE()
end=DATE()
ERASE NEW10.DBF
ERASE NEW2.DBF
DO WHILE .T.
  * erase screen and draw menu
  CLEAR
  @ 2,0 TO 12,79 DOUBLE
  @ 4,1 TO 4,78 DOUBLE
  @ 3,23 SAY "SCHEDULING INFORMATION REPORTS MENU"
  @ 5,27 SAY " 0 - Exit"
  @ 6,27 SAY " 1 - Report between two Dates"
  @ 7,27 SAY " 2 - Report on a particular Week"
  @ 8,27 SAY " 3 - Report on all Planned Weeks ahead"
  @ 9,27 SAY " 4 - Report on all Weeks this Year"
  @10,27 SAY " 5 - GRAPHS"
  @12,33 SAY " select | | |
  * input valid menu option
choice ="6"
DO WHILE .NOT. choice$"012345"
  choice=" 
  @ 12,42 GET choice
  READ
ENDDO

  * display asterisk next to menu selection
  @ 5+VAL(choice),26 SAY "*
  * erase other selections from menu
  cnt = 0
  DO WHILE cnt < 6
    IF cnt <> VAL(choice)
      @ 5+cnt,28 SAY SPACE(40)
    ENDIF
    cnt = cnt + 1
  ENDDO

  * test for exit condition
  IF choice $ " 0"
    EXIT
  ENDIF

IF choice$"1"
  DO WHILE .T.
    @15,0 CLEAR TO 17,79
    @15,10 SAY "Enter Starting Date :" GET begin
  READ
    @17,10 SAY "Enter Ending Date :" GET end
  READ
  IF begin<=end
    EXIT
  ENDIF
ENDIF
@8,10 SAY "Ending date should occur after the starting date - please reenter"
ENDIF
DO SI111
ENDIF

IF choice$="2"
  @15,0 CLEAR TO 19,79
  @16,10 SAY "Enter the Date(Friday of the week):" GET end
  READ
  DO SI112
ENDIF

IF choice$="3"
  DO SI113
ENDIF

IF choice$="4"
  @15,0 CLEAR
  DO SI114
ENDIF
IF choice$="5"
  DO SI115
ENDIF

ENDDO
ERASE NEW10.DBF
ERASE NEW2.DBF
RETURN
*eof:SI111.PRG

*==============================================
*PROGRAM: SI111.PRG
*menu for type of scheduling information reports
*==============================================

SET STATUS OFF
SET BELL OFF

DO WHILE .T.
  selectnum="8"
  * We need to clear the screen and display the main menu.
  CLEAR
  @4,10 SAY "Select the type of Report you need:"
  @5,10 TO 5,44
  @7,10 SAY "1. Hours for Machining and Assembly"
  @8,10 SAY "2. Hours for Machining only"
  @9,10 SAY "3. Hours for Assembly only"
  @10,10 SAY "4. Detailed Scheduling Information report for Machining"
  @11,10 SAY "5. Detailed Scheduling Information report for Assembly"
  @15,0 to 15,79
  selectnum=""
  DO WHILE .NOT. selectnum$"012345"
    selectnum=""
    @18,18 SAY "Enter Selection (1-5 OR 0 to Quit):" GET selectnum
    READ
  ENDDO

  DO CASE
  CASE selectnum$="0"
    RETURN
  CASE 1
    "case 1 code here"

  CASE 2
    "case 2 code here"

  CASE 3
    "case 3 code here"

  CASE 4
    "case 4 code here"

  CASE 5
    "case 5 code here"

  ENDCASE
CASE selectnum$"1"
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  USE PLAN INDEX WEEKNUM
  REPORT FORM SCHRPRT FOR WEEKNUMBER>=begin .AND. WEEKNUMBER<=end TO PRINT
CASE selectnum$"2"
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  USE PLAN INDEX WEEKNUM
  REPORT FORM SCHRPRT1 FOR SECTION='MACHINING' .AND. WEEKNUMBER>=begin; .AND. WEEKNUMBER<=end TO PRINT
CASE selectnum$"3"
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  USE PLAN INDEX WEEKNUM
  REPORT FORM SCHRPRT2 FOR SECTION='ASSEMBLY' .AND. WEEKNUMBER>=begin; .AND. WEEKNUMBER<=end TO PRINT
CASE selectnum$"4"
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  ERASE NEW2.DBF
  USE NEW1
  COPY TO NEW2 FOR PRODDATE>=begin .AND. PRODDATE<=end .AND. SECTION="MACHINING"
  USE NEW2
  IF RECCOUNT()=0
    CLEAR
    WAIT SPACE(13)+"NOT ENOUGH INFORMATION FOR REPORT - PRESS A KEY.."
    EXIT
  ENDIF
  SORT ON PRODDATE,PRIORITY,SALES_NUM,PRODUCT_ID TO NEW10
  USE
  DO SI1111
  ERASE NEW10.DBF
CASE selectnum$"5"
  WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
  ERASE NEW2.DBF
  USE NEW1
  COPY TO NEW2 FOR PRODDATE>=begin .AND. PRODDATE<=end .AND. SECTION="ASSEMBLY"
  USE NEW2
  IF RECCOUNT()=0
    CLEAR
    WAIT SPACE(13)+"NOT ENOUGH INFORMATION FOR REPORT - PRESS A KEY.."
    EXIT
  ENDIF
  SORT ON PRODDATE,PRIORITY,SALES_NUM,PRODUCT_ID TO NEW10
  USE
  DO SI1112
  ERASE NEW10.DBF
  ERASE NEW2.DBF
ENDCASE
ENDDO T
RETURN

*EOF:SI1111.PRG

*--------------------------------------------------------------------------
* PROGRAM :SI1111.PRG
* Type of report
*--------------------------------------------------------------------------

SELECT A
USE NEW1
SELECT B
USE NEW10
SELECT B
mdate=date()
mdue_date=DATE()
mquantity=0
mtotal= 0
msubtotal=0
msales_num=SPACE(15)
mproduct_id=SPACE(15)
msection=SPACE(15)
mpriority=SPACE(3)
x=11
y=1
z=1
p=0
c=0
timer=TIME()
pager="Page No."
SET DEVICE TO PRINT
title="ADEC DENTAL FURNITURE SHOP"
titlel="MACHINING SECTION - SCHEDULING INFORMATION REPORT"
underline="-------------"
subtitle="MFG_DATE DUE_DATE PRTY SALES_ORDER_# PRODUCT_LIST COMMENTS QTY HRS"
subline="-------------------------------"
EJECT
@2,1 SAY DATE()
@2,11 SAY timer
@2,68 SAY pager
@2,77 SAY y PICTURE "99"
@3,15 SAY title
@4,15 SAY underline
@6,15 SAY titlel
@8,1 SAY subline
@9,1 SAY subtitle
@10,1 SAY subline
@12,1 SAY PRODDATE
@12,10 SAY DUE_DATE
DO WHILE .T.
  IF z>1 .AND. PRODDATE<>mdate
    @x+1,67 say "total:";
    @x+1,74 say msubtotal PICTURE "99999"
    @x+2,67 say "============="
    x=x+4
    @x-1,1 SAY subline
    @x,1 SAY PRODDATE
    @x,10 SAY DUE_DATE
    msubtotal=0
  ELSE
    IF z>1 .AND. PRODDATE=mdate .AND. DUE_DATE<>mdue_date
      x=x+2
      @x,10 SAY DUE_DATE
    ELSE
      x=x+1
    ENDIF
  ENDIF
ENDIF
STORE PRODDATE TO mdate
STORE PRIORITY TO mpriority
STORE SALES_NUM TO msales_num
STORE DUE_DATE TO mdue_date
@x,19 SAY PRIORITY
@x,24 SAY SALES_NUM
DO WHILE .T.
IF PRODDATE=mdate .AND. PRIORITY=mpriority .AND. SALES_NUM=msales_num

tax,41 SAY PRODUCT_ID

tax,58 SAY COMMENTS

tax,70 SAY QUANTITY PICTURE "9999"
tax,75 SAY HOURS PICTURE "9999"
p=p+1
x=x+1
mtotal=HOURS+mtotal
msubtotal=Hours+msubtotal

ENDIF

IF x>58
  EJECT
  a2,1 SAY DATE()
a2,11 SAY timer
  a2,68 SAY pager
  y=y+1
  a2,77 SAY y PICTURE "99"
a3,15 SAY title
  a4,15 SAY underline
  a6,15 SAY title1
  a7,1 SAY subtitle
  a8,1 SAY subtitle
  a9,1 SAY subtitle
  a12,1 SAY mdate
  a12,10 SAY mdue_date
  a12,19 say Priority
  a12,24 say Sales_num
  STORE 12 TO x

ENDIF

SKIP

IF EOF()
  p=0
  EXIT
ENDIF

IF PRODDATE<>mdate .OR. DUE_DATE <> mdue_date .OR. PRIORITY<>mpriority .OR.
SALES_NUM<>msales_num
  p=0
  EXIT
ENDIF

ENDDO

IF EOF()
  ax+1,67 say "total:"
  ax+1,74 say msubtotal PICTURE "99999" 
  ax+2,67 say "=============="
x=x+2
  EXIT
ENDIF

ENDDO

CLOSE ALL
ERASE NEW10.DBF
ax+2,1 SAY subtitle
ax+3,60 SAY "TOTAL HOURS :
ax+3,74 SAY mtotal PICTURE "99999"
ax+5,1 SAY "*************** End of Report - Machining Section *******************"
ax+6,0 SAY CHR(7)
EJECT
SET DEVICE TO SCREEN
* EOF: SI1111.PRG

* PROGRAM: SI1112.PRG
* REPORTS

SELECT A
USE NEW1
SELECT B
USE NEW10
SELECT B
mdate=date()
mdue_date=date()
quantity=0
total= 0
msubtotal=0
msales_num=SPACE(15)
mproduct_id=SPACE(15)
mssection=SPACE(15)
mpriority=SPACE(3)
x=11
y=1
z=1
p=0
c=0
timer=TIME()
pager="Page No."
SET DEVICE TO PRINT
title="ADEC DENTAL FURNITURE SHOP"
title1="ASSEMBLY SECTION - SCHEDULING INFORMATION REPORT"
underline="*
subtitle="MFG_DATE DUE_DATE PRTY SALES_ORDER_# PRODUCT_LIST COMMENTS QTY HRS"
subline="----------------------------------
EJECT
@2, 1 SAY DATE()
@2, 11 SAY timer
@2, 68 SAY pager
@2, 77 SAY y PICTURE "99"
@3, 15 SAY title
@4, 15 SAY underline
@6, 15 SAY title
@8, 1 SAY subline
@9, 1 SAY subtitle
@10, 1 SAY subline
@12, 1 SAY PRODDATE
@12, 10 SAY DUE_DATE

DO WHILE .T.

IF x>1 .AND. PRODDATE<>mdate
@x+1, 67 say "total:"
@x+1, 74 say msubtotal PICTURE "99999"
@x+2, 67 say "="
x=x+4
@x-1, 1 SAY subline
@x, 1 SAY PRODDATE
@x, 10 SAY DUE_DATE
msubtotal=0
ELSE
IF z>1 .AND. PRODDATE=mdate .AND. DUE_DATE<>mdue_date
x=x+2
ELSE
x=x+1
ENDIF
ENDIF
STORE PRODDATE TO mdate
STORE PRIORITY TO mpriority
STORE SALES_NUM TO msales_num
STORE DUE_DATE TO mdue_date
STORE PRODDATE TO mdate
STORE PRIORITY TO mpriority
STORE SALES_NUM TO msales_num
DO WHILE J.
IF PRODDATE=mdate .AND. PRIORITY=mpriority .AND. SALES_NUM=msales_num
x=41 SAY PRODUCT_ID
x=58 SAY COMMENTS
x=70 SAY QUANTITY PICTURE "9999"
x=75 SAY HOURS PICTURE "9999"
p=p+1
x=x+1
mtotal=HOURS+mtotal
msubtotal=Hours+msubtotal
ENDIF
IF x>58
EJECT
x=2,1 SAY DATE()
x=2,11 SAY timer
x=2,68 SAY pager
y=y+1
x=2,77 SAY y PICTURE "99"
x=3,15 SAY title
x=4,15 SAY underline
x=6,15 SAY title1
x=7,1 SAY subtitle
x=8,1 SAY subtitle
x=9,1 SAY subtitle
x=12,1 SAY mdate
x=12,10 SAY mdue_date
x=12,19 SAY PRIORITY
x=12,24 SAY Sales_num
STORE 12 TO x
ENDIF
SKIP
IF EOF()
p=0
EXIT
ENDIF
IF PRODDATE<>mdate .OR. DUE_DATE <> mdue_date .OR. PRIORITY<>mpriority .OR.
SALES_NUM<>msales_num
p=0
EXIT
ENDIF
ENDDO
IF EOF()
x=1,67 say "total:"
x=1,74 say msubtotal PICTURE "999999"
x=2,67 say "===========

DO WHILE .T.
  selectnum="8"
  * We need to clear the screen and display the main menu.
  CLEAR
  @4,10 SAY "Select the type of Report you need:"
  @5,10 TO 5,44
  @7,10 SAY "1. Hours for Machining and Assembly"
  @8,10 SAY "2. Hours for Machining only"
  @9,10 SAY "3. Hours for Assembly only"
  @10,10 SAY "4. Detailed Scheduling Information report for Machining"
  @11,10 SAY "5. Detailed Scheduled Information report for Assembly"
  @15,0 to 15,79
  selectnum=" 
  DO WHILE .NOT. selectnum$"012345"
    selectnum=" "
    @18,18 SAY "Enter Selection ( 1-5 OR 0 to Quit ):" GET selectnum
    READ
  ENDDO
  DO CASE
    CASE selectnum$"0"
      RETURN
    CASE selectnum$"1"
      WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
      USE PLAN INDEX WEEKNUM
      REPORT FORM SCHWK FOR WEEKNUMBER=end TO PRINT
    CASE selectnum$"2"
      WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
      USE PLAN INDEX WEEKNUM
      REPORT FORM SCHWK1 FOR SECTION='MACHINING'.AND. WEEKNUMBER=end TO PRINT
    CASE selectnum$"3"
      WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
      USE PLAN INDEX WEEKNUM
      REPORT FORM SCHWK2 FOR SECTION='ASSEMBLY'.AND. WEEKNUMBER=end TO PRINT
  END
CASE selectnum$"4"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    ERASE NEW10.DBF
    USE NEW1
    COPY TO NEW2 FOR PRODDATE=end .AND. SECTION="MACHINING"
    USE NEW2
    IF RECCOUNT()=0
        CLEAR
        WAIT SPACE(13)+"NOT ENOUGH INFORMATION FOR REPORT - PRESS A KEY.."
        EXIT
    ENDIF
    SORT ON PROODATE,PRIORITY,SALES_NUM,PRODUCT_ID TO NEW10
    USE
    DO SI1111
    ERASE NEW10.DBF
    ERASE NEW2.DBF
CASE selectnum$"5"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    ERASE NEW10.DBF
    ERASE NEW2.DBF
    USE NEW1
    COPY TO NEW2 FOR PRODDATE=end .AND. SECTION="ASSEMBLY"
    USE NEW2
    IF RECCOUNT()=0
        CLEAR
        WAIT SPACE(13)+"NOT ENOUGH INFORMATION FOR REPORT - PRESS A KEY.."
        EXIT
    ENDIF
    SORT ON PROODATE,PRIORITY,SALES_NUM,PRODUCT_ID TO NEW10
    USE
    DO SI1112
    ERASE NEW10.DBF
    ERASE NEW2.DBF
ENDCASE
ENDDO T
RETURN
*EOF:SI112.PRG

===============================================
*PROGRAM: SI113.PRG
*menu for type of scheduling information report-option
===============================================

SET STATUS OFF
SET BELL OFF

DO WHILE .T.
    selectnum="8"
    * We need to clear the screen and display the main menu.
    CLEAR
    84,10 SAY "Select the type of Report you need:"
    85,10 TO 5,44
    87,10 SAY "1. Hours for Machining and Assembly"
    88,10 SAY "2. Hours for Machining only"
    89,10 SAY "3. Hours for Assembly only"
    91,10 SAY "4. Detailed Scheduling Information report for Machining"
    91,10 SAY "5. Detailed Scheduled Information report for Assembly"
    95,0 to 15,79
    selectnum=" "
    DO WHILE .NOT. selectnum$"012345"
        selectnum=" "


148,18 SAY "Enter Selection ( 1-5 OR 0 to Quit ):" GET selectnum
READ
ENDDO

DO CASE
  CASE selectnum$="0"
    RETURN
  CASE selectnum$="1"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    USE PLAN INDEX WEEKNUM
    REPORT FORM SCHF1 FOR WEEKNUMBER => DATE() TO PRINT
  CASE selectnum$="2"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    USE PLAN INDEX WEEKNUM
    REPORT FORM SCHF2 FOR SECTION = 'MACHINING' AND WEEKNUMBER => DATE() TO PRINT
  CASE selectnum$="3"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    USE PLAN INDEX WEEKNUM
    REPORT FORM SCHF3 FOR SECTION = 'ASSEMBLY' AND WEEKNUMBER => DATE() TO PRINT
  CASE selectnum$="4"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    ERASE NEW10.DBF
    ERASE NEW2.DBF
    USE NEW1
    COPY TO NEW2 FOR PRODDATE => DATE() AND SECTION = "MACHINING"
    USE NEW2
    IF RECCOUNT() = 0
      CLEAR
      WAIT SPACE(13)+"NOT ENOUGH INFORMATION FOR REPORT - PRESS A KEY.."
      EXIT
    ENDIF
    SORT ON PRODDATE, PRIORITY, SALES_NUM, PRODUCT_ID TO NEW10
    USE
    DO SI1111
    ERASE NEW10.DBF
    END CASE
  CASE selectnum$="5"
    WAIT SPACE(15)+"Get your Printer Ready and hit Return.."
    ERASE NEW10.DBF
    ERASE NEW2.DBF
    USE NEW1
    COPY TO NEW2 FOR PRODDATE => DATE() AND SECTION = "ASSEMBLY"
    USE NEW2
    IF RECCOUNT() = 0
      CLEAR
      WAIT SPACE(13)+"NOT ENOUGH INFORMATION FOR REPORT - PRESS A KEY.."
      EXIT
    ENDIF
    SORT ON PRODDATE, PRIORITY, SALES_NUM, PRODUCT_ID TO NEW10
    USE
    DO SI1112
    ERASE NEW10.DBF
    ERASE NEW2.DBF
  END CASE
ENDDO
RETURN
*EOF:SI113.PRG
APPENDIX C: Microsoft Quick C Source Code

Main1.c: mdl#1
******************************************************************************
#include "mdl#1.h"
#define MAXITERATIONS 20

struct Assy_weeks plannedWeeks[50];
void DontMoveMinJob();
void NextIteration();
void RecordIterResults();
void PrintResults();
void CreatFiles(void);

float ReturnVariance();
void WriteToDisplayFile(void);

FILE *fp_iter,*fp_results,*fpdat1,*fpdat2,*display;
char inputFile[20],rFile[20],itFile[10],rtFile[10];
char disp[30],inFile[30],jobFile[30];

int dataNum=1,modNum=1,earlyJobs=0,lateJobs=0,numMoves=0,counter=0,totalJobs=0;
float avgDevPerJob=0,sumMoveDist=0,initialOt=0,finalOt=0,capacity=300;
float InitialLoad[20],FinalLoad[20],initVar=0,finalVar=0;

main(int argc,char *argv[])
{
    float maxth,maxSlack;
    int maxiter=20,imaxSlackWeek,closedFiles;
    int imaxOtWeek,iminJob,iminJobWeek,icloseWeek,iterCount;
    float totalSlack=0.0;
    ED();
    printf(" \n\nWork Load Balancing Model: 1 being executed..\n\n");

    if((dataNum=atoi(argv[1]))==0)
    {
        dataNum=1;
    }

    CreatFiles();
    GetData(inFile,&counter);
    LoadJobs(jobFile,counter);
    if(maxiter ==0)
    {
        maxiter =MAXITERATIONS;
    }

    iterCount=1;
    while(iterCount <= maxiter)
    {
        FindSlack(iterCount,counter,&totalSlack);
        if(CheckOtHours(counter) > 0)
        {
            FindMaxWeek(counter,&imaxOtWeek);
            FindMinJob(imaxOtWeek,&iminJob);
            iminJobWeek=imaxOtWeek;

            if(CheckMinJobSlacks(counter,iminJob,iminJobWeek) >0)
            {
                icloseWeek=FindCloseWeek(counter,iminJob,iminJobWeek);
                fprintf(fp_iter,"\n\nmaxweek: %d; iminJob: %d; icloseWeek: %d\n",
                    imaxOtWeek,iminJob,icloseWeek);
                MoveMinJob(iminJob,iminJobWeek,icloseWeek);
                numMoves=numMoves+1;
                sumMoveDist=sumMoveDist+ abs(icloseWeek-iminJobWeek);
                NextIteration(&totalSlack,&iterCount);
            }
            else
            {
                ...
            }
        }
    }
}
GetMaxSlackWeek(counter,&maxSlack,&imaxSlackWeek);
if(maxSlack <= 0.0)
{break;}
if( (TestTotalSlack(counter,totalSlack,imaxSlackWeek,
iminJob,iminJobWeek)) >0)
{
    fprintf(fp_iter, \"maxweek: %d, iminJob: %d, imaxSlackWeek: %d\n", 
iminJobWeek,iminJob,imaxSlackWeek);
    MoveMinJob(iminJob,iminJobWeek,imaxSlackWeek);
    numMoves=numMoves+1;
    sumMoveDist=sumMoveDist+ abs(icloseWeek-iminJobWeek);
    NextIteration(&totalSlack,&iterCount);
}
else
{
    DontMoveMinJob(iminJobWeek);
    NextIteration(&totalSlack,&iterCount);
}
else
{
    RecordIterResults(iterCount,totalSlack);
    PrintResults(counter);
    WriteToDisplayFile();
    exit(03);
}

void NextIteration(tSlack,iterCurrent)
int *iterCurrent;
float *tSlack;
{
    fprintf(fp_iter, \"iteration: %d, Total Slack:%f\n",*iterCurrent,*tSlack);
    fprintf(fpdatl,"%10.2d %10.2f\n",*iterCurrent,*tSlack);
    *iterCurrent=*iterCurrent+1;
    *tSlack=0.0;
    return;
}

void RecordIterResults(iterCurrent,tSlack)
int iterCurrent;
float tSlack;
{
    fprintf(fpdatl,"%10.2d %10.2f",iterCurrent,tSlack);
    fprintf(fp_iter, \"iteration: %d, Total Slack:%f\n",iterCurrent,tSlack);
    return;
}
void Don'tMoveMinJob(i)
int i;
{
    plannedWeeks[i].tag=1;
}

void CreatFiles(void)
{
    strcpy(inFile,"S");
    sprintf(&inFile[1],"%1d.dat",dataNum);
    strcpy(jobFile,"J");
    sprintf(&jobFile[1],"%1d.dat",dataNum);
    strcpy(iFile,"IT-M");
    sprintf(&iFile[4],"%1d.M",modNum,dataNum);
    strcpy(rFile,"RT-M");
    sprintf(&rFile[4],"%1d.D%1d",modNum,dataNum);
    strcpy(itFile,"ITM");
    sprintf(&itFile[3],"%1d.M",modNum,dataNum);
    strcpy(rtFile,"RTM");
    sprintf(&rtFile[3],"%1d.D%1d",modNum,dataNum);
    strcpy(disp,"show.dis");
    if((display = fopen(disp,"w")) == NULL)
    {
        perror("can't open the display file.");
        exit(210);
    }

    if((fp_iter = fopen(iFile,"w")) == NULL)
    {
        perror("can't open the iterResult file.");
        exit(21);
    }

    if((fp_results = fopen(rFile,"w")) == NULL)
    {
        perror("can't open the Results file.");
        exit(31);
    }

    if((fpdat1 = fopen(itFile,"w")) == NULL)
    {
        perror("can't open the itdat file: %s",iFile);
        exit(41);
    }

    if((fpdat2 = fopen(rtFile,"w")) == NULL)
    {
        perror("can't open the rtdat file.");
        exit(51);
    }
    fprintf(fpdat1,"Iteration TotalSlack\n");
    fprintf(fpdat2,"WeekNumber InitialLoad FinalLoad\n");
}
float ReturnVariance(data, counter)
float data[];
int counter;
{
    int i;
    float total=0.0, mean=0.0;
    float diffSquared=0.0, dSqTotal=0.0, variance=0.0;

    for(i=0; i<counter; i++)
    {
        total=total+data[i];
    }
    mean=(total/counter);

    for(i=0; i<counter; i++)
    {
        diffSquared = ((data[i] - mean) * (data[i] - mean));
        dSqTotal = (dSqTotal + diffSquared);
    }

    variance= (dSqTotal / (counter-1));

    return (variance);
}

void PrintResults(counter)
int counter;
{
    int i, j;
    float totalLoad;
    for(i=0; i<counter; i++)
    {
        totalLoad=0.0;
        fprintf(fp_Results,"weeknumber: %d, plannedWeeks[i].weekNumber);
        fprintf(fp_Results,"\t\tweekDate: %d\n",plannedWeeks[i].weekDate);
        fprintf(fp_Results,"Order Number Hours Original WeekNumber\n");
        for(j=0; j<plannedWeeks[i].numOrders; j++)
        {
            fprintf(fp_Results,"%15s",plannedWeeks[i].salesOrders[j].orderNumber);
            fprintf(fp_Results,"%10.2f",plannedWeeks[i].salesOrders[j].totalHours);
            //fprintf(fp_Results,"%15ld",
            plannedWeeks[i].salesOrders[j].origDate);*/
            fprintf(fp_Results,"%10d\n", plannedWeeks[i].salesOrders[j].origNumber);

            if(plannedWeeks[i].salesOrders[j].origNumber > plannedWeeks[i].weekNumber)
            {
                earlyJobs=earlyJobs+1;
                if(plannedWeeks[i].salesOrders[j].origNumber < plannedWeeks[i].weekNumber)
                {
                    lateJobs=lateJobs+1;
                    totalLoad=totalLoad+plannedWeeks[i].salesOrders[j].totalHours;
                }
            }
        }
        fprintf(fp_Results,"\n Total Hours are : %.2f\n\n\n", totalLoad);
        FinalLoad[i]=totalLoad;
        if(totalLoad > capacity)
```c
void WriteToDisplayFile(void)
{
    if(numMoves > 0)
    {
        avgDevPerJob = (sumMoveDist/numMoves);
        initVar=ReturnVariance(InitialLoad,counter);
        finalVar=ReturnVariance(FinalLoad,counter);
        fprintf(display,"Work Load Balancing Procedures
        Summary Report
        Total Number of Jobs : %d",totalJobs);
        fprintf(display,"Num Initial Final Initial Final jobs re-
        move/job"
        "Early Late Average ");
        fprintf(display,"Num Overtime Variance Variance sched.
        Jobs Jobs move/job"
        "A2d",modNum);
        fprintf(display,"%10.3f",initial0t);
        fprintf(display,"%11.3f",final0t);
        fprintf(display,"%10.3f",initVar);
        fprintf(display,"%10.3f",finalVar);
        fprintf(display,"%7d",numMoves);
        fprintf(display,"%8d",earlyJobs);
        fprintf(display,"%6d",lateJobs);
        fprintf(display,"%10.3f",avgDevPerJob);
        fcloseall();
    }
}
```

```c
main2.c;
#include "mdl#2.h"
#define MAXITERATIONS 20
struct Assy_weeks plannedWeeks[30];
float ReturnVariance();
void WriteToDisplayFile(void);
FILE *fp_iter,*fp_results,*fopen(),*fpdat1,*fpdat2,*display;
char iFileI203,rFile[20],itFile[10],rtFileI101;
char disp[30],inFile(30),jobFile(30];int dataNuw1,modNuw2,earlyJobs=0,lateJobs=0,numMoves=0,counter=0,totalJobs=0;
float avgDevPerJob=0,sumMoveDist=0,initial0t=0,final0t=0,capacity=300;
float InitialLoad(20),FinalLoad(20),initVar=0,finalVar=0;
main(int argc,char *argv[])
{
    float meanWorkLoad=0.0,varLoad=0.0,OldVarLoad=0.0;
    int imaxWeek,iminJob,iminJobWeek,icloseWeek,iterCount;
    int iminWeek,ibestJob,closedFiles;
    ED();
    printf("Work Load Balancing Model : 2 being executed...
    "
```
if ( (dataNum=atoi(argv[1]))==0) 
(dataNum=1;)

CreatFiles();
GetData(inFile,&counter);
LoadJobs(jobFile,counter);

iterCount=1;
MeanVarLoad(plannedWeeks,counter,&meanWorkLoad,&varLoad);
RecordInitialValues(plannedWeeks,counter,meanWorkLoad,varLoad);

while(iterCount <= MAXITERATIONS) 
{ 
    if(iterCount > 1) 
    { 
        MeanVarLoad(plannedWeeks,counter,&meanWorkLoad,&varLoad);
    }
    if(OldVarLoad==varLoad) 
    { 
        break;
    }
    OldVarLoad=varLoad;
    if(varLoad >= 5.0) 
    { 
        FindMaxMinWeek(counter,&imaxWeek,&iminWeek);
        FindMinJob(imaxWeek,&iminJob);
        if(FindCloseWeek(imaxWeek,&icloseWeek,counter,varLoad,iminJob)>0) 
        { 
            MoveMinJob(iminJob,imaxWeek,icloseWeek);
            numMoves=numMoves+1;
            sumMoveDist=sumMoveDist+ abs(icloseWeek-imaxWeek);
            fprintf(fp_Iter,"\n\nmaxweek: %d, moveToWeek: %d, moveToWeek: %d\n",imaxWeek,iminJob,icloseWeek);
            NextIteration(&varLoad,iterCount);
        }
        else 
        { 
            DontMoveMinJob(imaxWeek);
            NextIteration(&varLoad,iterCount);
        }
    }
    else 
    { 
        RecordFinalIteration(iterCount,varLoad);
        PrintResults(counter);
        WriteToDisplayFile();
        exit(03);
    }
}

PrintResults(counter);
WriteToDisplayFile();
return;

/*******************************************************************************/
main3.c;mdl#3
*******************************************************************************/

#include "mdl#3.h"

struct Assy_weeks plannedWeeks[50];
void ResetTagsForOldPw();
void WriteToDisplayFile(void);  

FILE *fp_Iter,*fp_Results,*fopen(),*fpdat1,*fpdat2,*display;  
char iFile[20],rFile[20],itFile[10],rtFile[10];  
char disp[30],inFile[30],jobFile[30];  

int dataNum=1,modNum=3,earlyJobs=0,l lateJobs=0,totalJobs=0,numMoves=0,counter=0;  
float avgDevPerJob=0,sumMoveDist=0,initialOt=0,FinalOverTime=0,capacity=300;  
float InitialLoad[50],FinalLoad[50],initVar=0,finalVar=0;  

main(int argc,char *argv[])  
{
    float maxTh,maxSlack;  
    int imaxSlackWeek,pwStart,pwStop;  
    int nWeeks=1; /* planning window size: i+-nWeeks*/  
    
    float totalSlack=0.0;  
    int closedFiles,maxiter,Checker;  

    ED();  
    printf(" 

Work Load Balancing Model : 3 being executed..\n");  

    if(argc==1) {  
       {  
            CreatFiles();  
            GetData(inFile,&counter);  
            LoadJobs(jobFile,counter);  
            
            for(i=0;i<=counter;i++)  
            {  
                    ResetTagsForOldPw(pwStart,pwStop);  
                    iterCount=1;  
                    totalSlack=0.0;  
                    if(i==0)  
                    {  
                            pwStart=0;  
                            pwStop=pwStart+nWeeks;  
                        }  
                    else if(i+nWeeks == counter-1)  
                    {  
                            /*this else block is for the last planning window*/  
                            pwStart=i-nWeeks;  
                            pwStop=i+nWeeks;  
                            i=counter+1;  
                        }  
                else  
                {  
                        pwStart=i-nWeeks;  
                        pwStop=i+nWeeks;  
                    }  
            }  
            
            while(iterCount <= 6)  
            {  
                    FindSlack(iterCount,pwStart,pwStop,&totalSlack);  
                    if(CheckOtHours(pwStart,pwStop) > 0)  
                    {  
                    }  
            }


}
FindMaxWeek(pwStart, pwStop, &imaxOtWeek);
iminJobWeek = imaxOtWeek;
incloseWeek = FindCloseWeek(pwStart, pwStop, &iminJob, 
iminJobWeek, &Checker);

if (Checker > 0)
{
   fprintf(fp_Iter, "\n\nmaxweek: %d; iminJob: %d; incloseWeek: %d\n",
   imaxOtWeek, iminJob, incloseWeek);
   MoveMinJob(iminJob, iminJobWeek, incloseWeek);
   numMoves = numMoves + 1;
   sumMoveDist = sumMoveDist + abs(incloseWeek - iminJobWeek);
   NextIteration(pwStart, pwStop, &totalSlack, &IterCount);
}
else
{
   GetMaxSlackWeek(pwStart, pwStop, &maxSlack, &imaxSlackWeek);
   if (maxSlack <= 0.0)
   { break; }
   if (TestTotalSlack(pwStart, pwStop, totalSlack, 
imaxSlackWeek, iminJob, iminJobWeek)) > 0)
   {
      fprintf(fp_Iter, "\n\nmaxweek: %d, iminJob: %d, imaxSlackWeek: %d\n",
      imaxSlackWeek, iminJob, iminJobWeek);
      MoveMinJob(iminJob, iminJobWeek, imaxSlackWeek);
      numMoves = numMoves + 1;
      sumMoveDist = sumMoveDist + abs(imaxSlackWeek - iminJobWeek);
      NextIteration(pwStart, pwStop, &totalSlack, &IterCount);
   }
   else
   { 
      DontMoveMinJob();
      NextIteration(pwStart, pwStop, &totalSlack, &IterCount);
   }
}
else
{
   lastIterCount = iterCount;
   iterCount = 1;
   RecordIterResults(pwStart, pwStop, lastIterCount, totalSlack);
   break;
}
}

if (lastIterCount <= 6)
{
   RecordIterResults(pwStart, pwStop, lastIterCount, totalSlack);
}
PrintResults(counter);
WriteToDisplayFile();
return;
#include "mdl#4.h"
#define MAXITERATIONS 20

struct Assy_weeks plannedWeeks[50];

float ReturnVariance();
void WriteToDisplayFile(void);

FILE *fp_Iter,*fp_Results,*fopen(),*fpdat1,*fpdat2,*display;
char ifile[20],rfile[20],itfile[10],rtfile[10];
char dispC30,inFile(30),jobFileC30;
int dataNum,modNumF4,earlyJobs=0,lateJobs,0,totalJobs=0,numMoves=0,counter=0;
float avgDevPerJob=0,sumMoveDist=0,initialOt=0,finalOt=0,capacity=300;
float InitialLoad,FinalLoadC20,initVar=0,finalVar=0;

main(int argc,char *argv[])
{
    float maxth,maxSlack;
    int closedFiles;
    int imaxWeek,iminWeek,ibestJob,iterCount,maxiter;
    float totalSlack=0.0,prevTotSlack=0.0;
    ED();
    printf(" Work Load Balancing Model : 4 being executed..\n");
    if( (dataNum=atoi(argv[1]))==0)
    (dataNum=1);
    CreatFiles();
    GetData(inFileAcounter);
    LoadJobs(jobFile,counter);
    if(maxiter ==0)
    {maxiter=MAXITERATIONS;}
    iterCount=1;
    while(iterCount <= maxiter)
    {
        FindSlack(iterCount,counter,&totalSlack);
        if(prevTotSlack >0)
        {
            if(totalSlack >= prevTotSlack)
                break;
        }
        prevTotSlack=totalSlack;
        if(CheckOtHours(counter) > 0)
        {FindMaxMinWeek(counter,&imaxWeek,&iminWeek);
        if(FindBestJob(counter,imaxWeek,iminWeek,ibestJob,totalSlack)>0)
        {fprintf(fp_Iter,\n\nmaxweek: %d; ibestJob: %d;iminWeek:%d\n",imaxWeek,ibestJob,iminWeek);
            MoveMinJob(ibestJob,imaxWeek,iminWeek);
            numMoves=numMoves+1;
            sumMoveDist=sumMoveDist+ abs(iminWeek-imaxWeek);
            NextIteration(&totalSlack,&iterCount);
        }
        else
        {DontMoveMinJob(imaxWeek);
            
        }
    }
}
NextIteration(&totalSlack, &iterCount);

else
{
    RecordIterResults(iterCount, totalSlack);
    PrintResults(counter);
    WriteToDisplayFile();
    exit(0);
}

PrintResults(counter);
WriteToDisplayFile();
return;

void RecordIterResults(int iterCurrent, float tSlack)
{
    fprintf(fp_iter, "iteration: %d, Total Slack: %f\n", iterCurrent, tSlack);
    fprintf(fpdat1, "X%d X%10.2f\n", iterCurrent, tSlack);
    return;
}

#include "mdl5.h"
#define MAXITERATIONS 20

struct Assy_weeks plannedWeeks[30];

float ReturnVariance();
void WriteToDisplayFile(void);

FILE *fp_iter,*fp_Results,*fopen(),*fpdat1,*fpdat2,*display;
char iFile[20],rFile[20],itFile[10],rtFile[10];
char disp[30],inFile[30],jobFile[30];
int dataNum=1,modNum=5,earlyJobs=0,lateJobs=0,totalJobs=0,numMoves=0,counter=0;
float avgDevPerJob=0,sumMoveDist=0,initialOt=0,finalOt=0,capacity=300;
float InitialLoad[20],FinalLoad[20],initVar=0,finalVar=0;

main(int argc,char *argv[])
{
    float meanWorkLoad=0.0,varLoad=0.0;
    int imaxWeek,iminJob,iminJobWeek,icloseWeek,iterCount;
    int iminWeek,ibestJob,closedFiles,maxiter=0;

    printf("Work Load Balancing Model : 5 being executed..\n");
    if( (dataNum=atoi(argv(1))==0)
        (dataNum=1);
    CreatFiles();
    GetData(inFile,&counter);
    LoadJobs(jobFile,counter);

    if(maxiter ==0)
        (maxiter=MAXITERATIONS);
iterCount=1;
MeanVarLoad(plannedWeeks,counter,&meanWorkLoad,&varLoad);
RecordInitialValues(plannedWeeks,counter,meanWorkLoad,varLoad);

while(iterCount <= maxiter)
{
    if(iterCount > 1)
    {
        MeanVarLoad(plannedWeeks,counter,&meanWorkLoad,&varLoad);
    }
    if(varLoad >= 5.0)
    {
        FindMaxMinWeek(counter,&imaxWeek,&iminWeek);
        if(CheckAllJobs(imaxWeek,iminWeek,counter,varLoad,&ibestJob) >0)
        {
            MoveMinJob(ibestJob,imaxWeek,iminWeek);
            numMoves=numMoves+1;
            sumMoveDist=sumMoveDist+ abs(iminWeek-imaxWeek);
            fprintf(fp_iter,\"%d,moveToWeek:%d\n\",imaxWeek,ibestJob,iminWeek);
            NextIteration(&varLoad,&iterCount);
        }
        else
        {
            DontMoveMinJob(imaxWeek);
            NextIteration(&varLoad,&iterCount);
        }
    }
    else
    {
        RecordFinalIteration(iterCount,varLoad);
        PrintResults(counter);
        WriteToDisplayFile();
        exit(0);
    }
}

return;

main6.c;;mdl#6
#include "mdl#6.h"

struct Assy weeks plannedWeeks[50];

float ReturnVariance();
void WriteToDisplayFile(void);

FILE *fp_iter,*fp_results,* fopen(),*fpdat1,*fpdat2,*display;
char ifile[20],rfile[20],itfile[10],rtfile[10];
char disp[30],infie[30],jofie[30];
int dataNum=1,modNum=6,earlyJobs=0,lateJobs=0,totalJobs=0,numMoves=0,counter=0;
float avgDevPerJob=0,sumMoveDist=0,initialQ=0,finalQ=0,capacity=300;
float InitialLoad[20],FinalLoad[20],initVar=0,finalVar=0;

main(int argc,char *argv[])
int pwStart=0,pwStop=0;
int nWeeks=1; /*planning window size: +/-nWeeks*/
int i,lastIterCount,maxiter,closedFiles;
int imaxWeek,iminWeek,ibestJob,iterCount;
float totalSlack=0.0,prevTotalSlack=0.0;

ED();
printf(" 

Work Load Balancing Model : 6 being executed..\n");
if( (dataNum=atoi(argv[17])==0)
(dataNum=1;)
CreatFiles();
GetData(inFile,&counter);
LoadJobs(jobFile,counter);
for(i=0;i<=counter;i++)
{
    ResetTagsForOldPw(pwStart,pwStop);
    iterCount=1;
    totalSlack=0.0;
    prevTotalSlack=0.0;
    if(i==0)
    (
        pwStart=0;
        pwStop=pwStart+nWeeks;
    )
    else if(i+nWeeks == counter-1)
    {
        /*this else block is for the last planning window*/
        pwStart=i-1-nWeeks;
        pwStop=i+nWeeks;
        i=counter+1;
    }
    else
    {
        pwStart=i-1-nWeeks;
        pwStop=i+nWeeks;
    }

    while(iterCount <= 6)
    {
        FindSlack(iterCount,pwStart,pwStop,&totalSlack);
        if(prevTotalSlack >0)
        {
            if(totalSlack >=prevTotalSlack)
            {
                break;
            }
        }
        prevTotalSlack=totalSlack;
        if(CheckOtHours(pwStart,pwStop) > 0)
        {
            FindMaxMinWeek(pwStart,pwStop,&imaxWeek,&iminWeek);
            if(FindBestJob(pwStart,pwStop,imaxWeek,iminWeek,&ibestJob,totalSlack) >0)
            {
                fprintf(fp_iter,"\n\nmaxweek: %d; ibestJob:
imaxWeek,ibestJob,iminWeek;"

        MoveMinJob(ibestJob,imaxWeek,iminWeek);
        numMoves=numMoves+1;
        }
else

C

PrintResults(counter);
WriteToFile();
return;

else

C

sLuMoveDist=sumMoveDist+ abs(iminWeek-imaxWeek);
NextIteration(pwStart,pwStop,&totalSlack,&iterCount);

DontMoveMinJob();
NextIteration(pwStart,pwStop,&totalSlack,&iterCount);

else

C

lastIterCount=iterCount;
iterCount=1;
RecordIterResults(pwStart,pwStop,lastIterCount,totalSlack);
break;

)

)

PrintResults(counter);
WriteToFile();
return;

/***************************************************************/
Functions used to implement the six work load balancing models.
All the functions are called by more than one models.
/***************************************************************/

int CheckOtHours();
void FindMaxWeek();
void FindSlack();
void FindMinJob();
int CheckMinJobSlacks();
int FindCloseWeek();
int TestTotalSlack();
void MeanVarLoad();
void FindMaxMinWeek();
void CalcVariance();
void FindBestJob();
void CalcTotalSlack();
int CheckAllJobs();
void GetMaxSlackWeek();
void UpdateMinJobWeek();
void UpdateCloseWeek();
void MoveMinJob();

/*Global declarations*/

extern struct Assy weeks plannedWeeks[];

/*FindSlack()***************
This function calculates the total slack for a planning horizon
and return the value through the use of the pointer totSlack.
*/

void FindSlack(iter,counter,totSlack)
int counter,iter;
float *totSlack;
int i,j;
float diff;
static int count=0;
for(i=0;i< counter;i++)
{
    diff = (plannedWeeks[i].capacity - plannedWeeks[i].totalHours);
    if(diff <0)
    {
        diff = diff * (-1);
    }
    *totSlack= (*totSlack + diff);
    /*printf("capacity is \%f ; totalHours are: \%f\n",
            plannedWeeks[i].capacity,plannedWeeks[i].totalHours);*/
}
/*printf("slack for iteration \%d is \%f\n",iter,*totSlack);*/
return;

/*CheckOtHours()*******************************************************************************************/
function that checks all the weeks in the planning window chosen to
see whether there is any week with overtime Hours or -ve slack
and returns 1 if any of the weeks have overtime hours else returns
-1 if none of them have overtime
******************************************************************************/
int CheckOtHours(counter)
int counter;
{
    int i;
    for(i=0;i< counter;i++)
    {
        if(plannedWeeks[i].totalHours > plannedWeeks[i].capacity)
            return 1;
    }
    return -1;
}

/*FindMaxWeek()*******************************************************************************************/
This function finds the week in the planning window which has the
maximum totalHours planned and returns the value of the index to that
week in the planned weeks array
******************************************************************************/
void FindMaxWeek(counter,imaxotweek)
int *imaxotweek,counter;
{
    int i,i1;
    float max,min;
    max=0;
    for(i=0;i< counter;i++)
    {
        if(plannedWeeks[i].totalHours > 0)
        {
            if(plannedWeeks[i].totalHours > max)
            {
                max=plannedWeeks[i].totalHours;
                *imaxotweek=i;
            }
/*GetMaxSlackWeek()*******************************************************************/
function that finds the week in the planning window with the maximum +ve slack Hours available
*************************************************************************************/
void GetMaxSlackWeek(counter,maxsh,imaxsh)
float *maxsh;
int *imaxsh,counter;
{
    int i;
    *maxsh = plannedWeeks[i].slackHours;
    for(i=0;i< counter;i++)
    {
        if(plannedWeeks[i].slackHours > *maxsh)
        {
            *maxsh=plannedWeeks[i].slackHours;
            *imaxsh=i;
        }
    }
    /*printf("max slack: %f,maxIndex is:%d",*maxsh,*imaxsh);*/
}

/*FindMinJob()***************************************************************************/
function finds the job with minimum hours required to manufacture from the selected week
**************************************************************************************/
void FindMinJob(imaxotweek,iminJob)
int imaxotweek,*iminJob;
{
    float minJob=5000.0;
    int i,j=imaxotweek;
    for(i=0;i< plannedWeeks[j].numOrders;i++)
    {
        if(plannedWeeks[j].salesOrders[i].totalHours < minJob)
        {
            minJob=plannedWeeks[j].salesOrders[i].totalHours ;
            *iminJob=i;
        }
    }
    return;
}

/*CheckMinJobSlacks()*******************************************************************/
this function checks the minJob selected against the +ve slacks available in the other weeks of the planning window
*************************************************************************************/
int CheckMinJobSlacks(counter,iminJob,iminJobWeek)
int counter,iminJob,iminJobWeek;
{
    int i;
    for(i=0;i< counter ;i++)
    {
        if(plannedWeeks[iminJobWeek].salesOrders[iminJob].totalHours <= plannedWeeks[i].slackHours)
return 1;
return -1;

/*FindCloseWeek()******************************************************
function to find the closest week to the selected week to move the minJob
from the selected week to the closest week
**************************************************************************/
int FindCloseWeek(counter,iminJob,iminJobWeek)
int counter,iminJob,iminJobWeek;
{
    int i;
    float minJob=plannedWeeks[iminJobWeek].salesOrders[iminJob].totalHours;
    for(i=iminJobWeek-1;i>=0;i--)
    {
        if(plannedWeeks[i].slackHours >= minJob)
        {
            return i;
        }
    }
    for(i=iminJobWeek+1;i<=counter;i++)
    {
        if(plannedWeeks[i].slackHours >= minJob)
        {
            return i;
        }
    }
    printf("error in program function:FindCloseWeek()");
    exit(99);
}

/*TestTotalSlack()**************************************************************
function to check the improvement in the total slack value after moving the
minJob to the close week to decide on the feasibility of the move
******************************************************************************
int TestTotalSlack(counter,PrevtotSlack,icloseWeek,iminJob,iminJweek)
int iminJob,icloseWeek,iminJweek,counter;
float PrevtotSlack;
{
    int i;
    float totalSlack=0.0;
    float diff;
    for(i=0;i< counter;i++)
    {
        if(i==icloseWeek)
        {
            diff = (plannedWeeks[i].capacity - (plannedWeeks[i].totalHours +
            plannedWeeks[iminJweek].salesOrders[iminJob].totalHours));
            if( diff < 0)
            {
                diff= diff * (-1);
            }
            totalSlack=totalSlack + diff;
        }
        else if(i==iminJweek)
        {
```c

diff = (plannedWeeks[i].capacity - (plannedWeeks[i].totalHours -
    plannedWeeks[i].salesOrders[iminJob].totalHours));

if (diff < 0)
{
    diff = diff * (-1);
}

totalSlack = totalSlack + diff;

else
{
    diff = (plannedWeeks[i].capacity - plannedWeeks[i].totalHours);
    if(diff<0)
    {
        diff = diff*(-1);
    }

    totalSlack = totalSlack + diff;
}

if (totalSlack <= PrevtotSlack)
{
    return 1;
}
else
    return -1;

/*****************************************************************************

    function that performs the housekeeping work required to
    move the minJob from it's present week to the close week
*****************************************************************************/

void MoveMinJob(int iminJob, int iminJweek, int icloseWeek)
int iminJob, iminJweek, icloseWeek;
{
    UpdateCloseWeek(iminJweek, iminJob, icloseWeek);
    UpdateMinJobWeek(iminJweek, iminJob);
    return;
}

void UpdateCloseWeek(int iweek, int ijob, int icweek)
int icweek, iweek, ijob;
{
    int orderSlot;
    plannedWeeks[icweek].totalHours = (plannedWeeks[icweek].totalHours +
    plannedWeeks[iweek].salesOrders[ijob].totalHours);

    plannedWeeks[icweek].slackHours = (plannedWeeks[icweek].capacity -
    plannedWeeks[icweek].totalHours);

    orderSlot = plannedWeeks[icweek].numOrders;

    strcpy(plannedWeeks[icweek].salesOrders[orderSlot].orderNumber,
    plannedWeeks[iweek].salesOrders[ijob].orderNumber);

    plannedWeeks[icweek].salesOrders[orderSlot].totalHours =
    plannedWeeks[iweek].salesOrders[ijob].totalHours;
```
plannedWeeks[iweek].salesOrders[orderSlot].origDate =
    plannedWeeks[iweek].salesOrders[ijob].origDate;
plannedWeeks[iweek].salesOrders[orderSlot].origNumber =
    plannedWeeks[iweek].salesOrders[ijob].origNumber;

plannedWeeks[iweek].numOrders=plannedWeeks[iweek].numOrders+1;
return;
)

void UpdateMinJobWeek(iweek,ijob)
int iweek,ijob;
{
    int i,j,orders;
    plannedWeeks[iweek].totalHours= (plannedWeeks[iweek].totalHours-
    plannedWeeks[iweek].salesOrders[ijob].totalHours);
    plannedWeeks[iweek].slackHours = (plannedWeeks[iweek].capacity -
    plannedWeeks[iweek].totalHours);
    orders=plannedWeeks[iweek].numOrders;
    for(i=ijob;i<orders-1;++i)
    {
        strcpy(plannedWeeks[iweek].salesOrders[i].orderNumber,
            plannedWeeks[iweek].salesOrders[i+1].orderNumber);
        plannedWeeks[iweek].salesOrders[i].totalHours=
            plannedWeeks[iweek].salesOrders[i+1].totalHours;
        plannedWeeks[iweek].salesOrders[i].origDate=
            plannedWeeks[iweek].salesOrders[i+1].origDate;
        plannedWeeks[iweek].salesOrders[i].origNumber=
            plannedWeeks[iweek].salesOrders[i+1].origNumber;
    }
    plannedWeeks[iweek].numOrders=plannedWeeks[iweek].numOrders-1;
    return;
}

void MeanVarLoad(pweeks,counter,mean,var)
struct Assy_weeks pweeks[];
int counter;
float *mean,*var;
{
    int i;
    float total=0.0;
    float diffSquared=0.0,dsqTotal=0.0,Average;
    for(i=0;i<counter;i++)
    {
        total=total+pweeks[i].totalHours;
    }
    *mean=(total/counter);
    Average=*mean;
for(i=0;i<counter;i++)
{
    diffSquared = ( ( pweeks[i].totalHours - Average )*
                    ( pweeks[i].totalHours - Average ));
    dSgTotal = ( dSgTotal+diffSquared );
}
*var= ( dSgTotal / (counter-1) );
return;

/*FindMaxMinWeek()**********************************************************************
This function finds the week in the planning window which has the
maximum totalHours and min totalHours planned and returns the value of the
index to that week in the planned weeks array
*******************************************************************************/
void FindMaxMinWeek(counter,imaxWeek,iminWeek)
int *iminWeek,*imaxWeek,counter;
{
    int i;
    float max,min;
    max=0.0;
    min=5000.0;
    for(i=0;i<counter;i++)
    {
        if(plannedWeeks[i].tag ==0)
        {
            if(plannedWeeks[i].totalHours > max)
            {
                max=plannedWeeks[i].totalHours;
                *imaxWeek=i;
            }
        }
    }
    for(i=0;i<counter;i++)
    {
        if(plannedWeeks[i].tag ==0)
        {
            if(plannedWeeks[i].totalHours < min)
            {
                min=plannedWeeks[i].totalHours;
                *iminWeek=i;
            }
        }
    }
    return;
}

void CalcVariance(counter,imaxWeek,iminWeek,JthJob,NewVariance)
int counter,JthJob;
int imaxWeek,iminWeek;
float *NewVariance;
{
    int i;
float total[50];
float mean=0.0,diffSquared=0.0,diffSqTotal=0.0;
float GrandTotal=0.0;

for(i=0;i<counter;i++)
{
    total[i]=plannedWeeks[i].totalHours;
}
total[iminWeek]=plannedWeeks[iminWeek].totalHours + plannedWeeks[imaxWeek].salesOrders[JthJob].totalHours;
total[imaxWeek]=plannedWeeks[imaxWeek].totalHours - plannedWeeks[imaxWeek].salesOrders[JthJob].totalHours;

for(i=0;i<counter;i++)
{
    GrandTotal=GrandTotal+total[i];
}
mean = (GrandTotal/counter);
for(i=0;i<counter;i++)
{
    diffSquared = ((total[i] - mean)*(total[i] - mean));
    diffSqTotal=diffSqTotal+diffSquared;
}

*NewVariance = (diffSqTotal/(counter-1));
return;

void FindBestJob(counter,imaxWeek,iminWeek,ibJob,prevTotalSlack)
int imaxWeek,iminWeek,counter,*ibJob;
float prevTotalSlack;
{
    int im=0,j;
    float newTotalSlack,totSlack[50],min;
    int NumOrders=plannedWeeks[imaxWeek].numOrders;
    for(j=0;j<NumOrders;j++)
    {
        CalcTotalSlack(counter,imaxWeek,iminWeek,j,&newTotalSlack);
        totSlack[j]=newTotalSlack;
    }

    min=totSlack[0];
    for(j=0;j<NumOrders;j++)
    {
        if(totSlack[j] <= min)
        {
            min=totSlack[j];
            im=j;
        }
    }

    if(totSlack[im] < prevTotalSlack)
    {
        *ibJob=im;
        return 1;
    }
    else
    {
        return -1;
    }
}

void CalcTotalSlack(counter,imaxWeek,iminWeek,JthJob,NewTotalSlack)
int counter,JthJob;
int imaxWeek,iminWeek;
float *NewTotalSlack;
{
    int i;
    float total[60],diff;
    float GrandTotal=0.0;
    for(i=0;i<counter;i++)
    {
        total[i]=plannedWeeks[i].totalHours;
    }
    total[iminWeek]=plannedWeeks[iminWeek].totalHours+
    plannedWeeks[imaxWeek].salesOrders[JthJob].totalHours;
    total[imaxWeek]=plannedWeeks[imaxWeek].totalHours-
    plannedWeeks[imaxWeek].salesOrders[JthJob].totalHours;
    for(i=0;i<counter;i++)
    {
        diff=plannedWeeks[i].capacity-total[i];
        if(diff < 0)
        {
            diff=diff*(-1);
        }
        GrandTotal=GrandTotal+diff;
    }
    *NewTotalSlack = GrandTotal;
    return;
}

CheckAllJobs(imaxWeek,iminWeek,counter,prevVariance,ibJob)
int imaxWeek,iminWeek,counter,*ibJob;
float prevVariance;
{
    int imm0,j;
    float newVariance, varArray[50],min;
    int NumOrders=plannedWeeks[imaxWeek].numOrders;
    for(j=0;j<NumOrders;j++)
    {
        CalcVariance(counter,imaxWeek,iminWeek,j,&newVariance);
        varArray[j]=newVariance;
    }
    min=varArray[0];
    for(j=0;j< NumOrders;j++)
    {
        if(varArray[j] <= min)
        {
            min=varArray[j];
            imm=j;
        }
    }
    if(varArray[imm] <= prevVariance)
    {
        *ibJob=imm;
        return 1;
    }
    else
    {
        return -1;
    }
}
APPENDIX D: Flow Diagrams for Model 2 through Model 6

Model 2

START

Compute the Mean and Variance of load over the planning horizon.

Is Variance > 5.0 units?

Yes

Find the week with max. hrs. of loading (MaxWeek) & the week with min. hrs. loading (MinWeek)

Find the MinJob from the MaxWeek

Does Variance reduce if MinJob moved to MinWeek?

Yes

Move MinJob to MinWeek

No

Remove MaxWeek from Consideration

More Iterations?

Yes

No

STOP
Model 3

START

Identify the first (next) planning window in the planning horizon

Any Week with Overtime Hrs.? Yes

Identify Week with Maximum Overtime Hrs. (MaxWeek)

Identify Job with Minimum Processing time from MaxWeek (MinJob)

Compute Total Slack value for the Planning Window (TS)

Identify Slack Weeks (Weeks with Actual Load less than DCUL)

Number of Slack Weeks > 0 ? No

Yes

Select Slack Week closest to due date that can accept MinJob (NearSlackWeek)

NearSlackWeek Exists? Yes

A

No

B

E
Identify Slack Week with Maximum Slack Available (MaxSlackWeek)

Does TS improve if MinJob moved to MaxSlackWeek?

Yes

Move MinJob to MaxSlackWeek

C

More iterations to do?

No

E

Remove MaxWeek from consideration

Yes

D

Model 3 cont.
Model 3 cont.

Reschedule MinJob from MaxWeek to NearSlackWeek
Identify the week with Max. overtime hours (MaxWeek) and the week with Min. hrs. loading (MinWeek)

Compute Total Slack value for the Planning Horizon (TS)

Identify the job that reduces TS most if job is moved to MinWeek (BestJob)

BestJob Exists?

No

Remove MaxWeek from Consideration

Yes

Reschedule BestJob from MaxWeek to MinWeek

More Iterations to do?

No

STOP

Yes

D
Compute the Mean and Variance of load over the planning horizon.

Is Variance > 5.0 units?

Yes

Find the week with max. hrs. of loading (MaxWeek) & the week with min. hrs. loading (MinWeek)

Find the Job from MaxWeek that reduces the variance by most (BestJob) if it is moved to MinWeek

BestJob Exists?

Yes

Move BestJob to MinWeek

No

Remove MaxWeek from Consideration

No

More Iterations?

Yes

STOP
Identify the first (next) planning window

Any Week with Overtime Hrs.?

Yes

Identify the week with Max. overtime hours (MaxWeek) and the week with Min. hrs. loading (MinWeek)

Compute Total Slack value for the Planning Window (TS)

Identify the job that reduces TS most if job is moved to MinWeek (BestJob)

Remove MaxWeek from Consideration

No

BestJob Exists?

Yes

Reschedule BestJob from MaxWeek to MinWeek

More Iterations for this planning window?

No

Yes

D
<table>
<thead>
<tr>
<th>row</th>
<th>Mechanisms</th>
<th>Policies</th>
<th>EarlyJobs</th>
<th>LateJobs</th>
<th>AveMove</th>
<th>TotResched</th>
<th>VarReduced</th>
<th>TotalCost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1.33</td>
<td>9</td>
<td>51.04</td>
<td>46.91</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1.44</td>
<td>9</td>
<td>93.50</td>
<td>118.10</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>1.88</td>
<td>8</td>
<td>94.70</td>
<td>66.41</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>1.56</td>
<td>18</td>
<td>98.40</td>
<td>84.86</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1.67</td>
<td>9</td>
<td>86.40</td>
<td>131.47</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>11</td>
<td>2.25</td>
<td>20</td>
<td>99.68</td>
<td>293.32</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3.00</td>
<td>5</td>
<td>94.70</td>
<td>117.10</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1.40</td>
<td>5</td>
<td>92.53</td>
<td>91.18</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2.50</td>
<td>4</td>
<td>97.10</td>
<td>113.36</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>1.80</td>
<td>10</td>
<td>98.78</td>
<td>101.07</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>1.67</td>
<td>6</td>
<td>89.89</td>
<td>157.04</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>2.67</td>
<td>12</td>
<td>99.66</td>
<td>350.15</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td>1.40</td>
<td>20</td>
<td>99.34</td>
<td>109.34</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>1.47</td>
<td>15</td>
<td>99.39</td>
<td>136.72</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>1.90</td>
<td>20</td>
<td>99.51</td>
<td>128.23</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>1</td>
<td>18</td>
<td>0</td>
<td>1.50</td>
<td>20</td>
<td>98.80</td>
<td>86.46</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>8</td>
<td>1.50</td>
<td>20</td>
<td>95.60</td>
<td>180.98</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td>1.50</td>
<td>20</td>
<td>80.50</td>
<td>148.99</td>
</tr>
<tr>
<td>19</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3.43</td>
<td>7</td>
<td>99.57</td>
<td>179.00</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1.71</td>
<td>7</td>
<td>99.57</td>
<td>127.16</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2.57</td>
<td>7</td>
<td>99.56</td>
<td>134.99</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>2.00</td>
<td>13</td>
<td>99.95</td>
<td>132.15</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>2.18</td>
<td>11</td>
<td>99.78</td>
<td>260.91</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>2.86</td>
<td>14</td>
<td>99.74</td>
<td>359.76</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1.33</td>
<td>9</td>
<td>51.04</td>
<td>46.91</td>
</tr>
<tr>
<td>26</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>1.56</td>
<td>9</td>
<td>75.97</td>
<td>137.09</td>
</tr>
<tr>
<td>27</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1.44</td>
<td>9</td>
<td>87.40</td>
<td>54.50</td>
</tr>
<tr>
<td>28</td>
<td>3</td>
<td>1</td>
<td>17</td>
<td>2</td>
<td>1.26</td>
<td>19</td>
<td>89.70</td>
<td>87.99</td>
</tr>
<tr>
<td>29</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>1.22</td>
<td>9</td>
<td>73.29</td>
<td>127.22</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>7</td>
<td>1.35</td>
<td>17</td>
<td>91.75</td>
<td>149.98</td>
</tr>
<tr>
<td>31</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1.20</td>
<td>5</td>
<td>64.03</td>
<td>45.95</td>
</tr>
<tr>
<td>32</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>1.36</td>
<td>11</td>
<td>91.29</td>
<td>165.49</td>
</tr>
<tr>
<td>33</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1.60</td>
<td>5</td>
<td>78.78</td>
<td>47.02</td>
</tr>
<tr>
<td>34</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>1.31</td>
<td>13</td>
<td>89.46</td>
<td>82.30</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>9</td>
<td>1.33</td>
<td>12</td>
<td>98.63</td>
<td>190.42</td>
</tr>
<tr>
<td>36</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>1.40</td>
<td>15</td>
<td>92.87</td>
<td>173.13</td>
</tr>
</tbody>
</table>
# APPENDIX E-2 : ANOVA - Early Jobs

Analysis of Variance for THESIS.EarlyJobs

<table>
<thead>
<tr>
<th>Source of variation</th>
<th>Sum of Squares</th>
<th>d.f.</th>
<th>Mean square</th>
<th>F-ratio</th>
<th>Sig. level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN EFFECTS</td>
<td>123.41667</td>
<td>3</td>
<td>41.13889</td>
<td>2.227</td>
<td>.1055</td>
</tr>
<tr>
<td>THESIS.Mechanisms</td>
<td>20.05556</td>
<td>2</td>
<td>10.02778</td>
<td>.543</td>
<td>.5867</td>
</tr>
<tr>
<td>THESIS.Policies</td>
<td>103.36111</td>
<td>1</td>
<td>103.36111</td>
<td>5.595</td>
<td>.0247</td>
</tr>
<tr>
<td>2-FACTOR INTERACTIONS</td>
<td>8.7222222</td>
<td>2</td>
<td>4.3611111</td>
<td>.236</td>
<td>.7912</td>
</tr>
<tr>
<td>THESIS.MecTHESIS.Pol</td>
<td>8.7222222</td>
<td>2</td>
<td>4.3611111</td>
<td>.236</td>
<td>.7912</td>
</tr>
<tr>
<td>RESIDUAL</td>
<td>554.16667</td>
<td>30</td>
<td>18.472222</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL (CORR.)</td>
<td>686.30556</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 missing values have been excluded.
## Analysis of Variance for THESIS.LateJobs

<table>
<thead>
<tr>
<th>Source of variation</th>
<th>Sum of Squares</th>
<th>d.f.</th>
<th>Mean square</th>
<th>F-ratio</th>
<th>Sig. level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MAIN EFFECTS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.Mechanisms</td>
<td>10.500000</td>
<td>3</td>
<td>3.5000000</td>
<td>.402</td>
<td>.7527</td>
</tr>
<tr>
<td>THESIS.Policies</td>
<td>6.500000</td>
<td>2</td>
<td>3.2500000</td>
<td>.373</td>
<td>.6918</td>
</tr>
<tr>
<td><strong>2-FACTOR INTERACTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.MechTHESIS.Pol</td>
<td>2.1666667</td>
<td>2</td>
<td>1.0833333</td>
<td>.124</td>
<td>.8835</td>
</tr>
<tr>
<td><strong>RESIDUAL</strong></td>
<td>261.33333</td>
<td>30</td>
<td>8.7111111</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL (CORR.)</strong></td>
<td>274.00000</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 missing values have been excluded.
**APPENDIX E-4 : ANOVA - Total Number of Jobs Rescheduled**

<table>
<thead>
<tr>
<th>Source of variation</th>
<th>Sum of Squares</th>
<th>d.f.</th>
<th>Mean square</th>
<th>F-ratio</th>
<th>Sig. level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MAIN EFFECTS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.Mechanisms</td>
<td>419.16667</td>
<td>3</td>
<td>139.72222</td>
<td>8.969</td>
<td>.0002</td>
</tr>
<tr>
<td>THESIS.Policies</td>
<td>266.77778</td>
<td>1</td>
<td>266.77778</td>
<td>17.126</td>
<td>.0003</td>
</tr>
<tr>
<td><strong>2-FACTOR INTERACTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.Mech THEESIS.Pol</td>
<td>84.722222</td>
<td>2</td>
<td>42.361111</td>
<td>2.719</td>
<td>.0822</td>
</tr>
<tr>
<td><strong>RESIDUAL</strong></td>
<td>467.33333</td>
<td>30</td>
<td>15.577778</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL (CORR.)</strong></td>
<td>971.22222</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 missing values have been excluded.
### APPENDIX E-5: ANOVA - Average distance moved per job

Analysis of Variance for THESIS.AveMove

<table>
<thead>
<tr>
<th>Source of variation</th>
<th>Sum of Squares</th>
<th>d.f.</th>
<th>Mean square</th>
<th>F-ratio</th>
<th>Sig. level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN EFFECTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.Mechanisms</td>
<td>2.9203834</td>
<td>2</td>
<td>1.4601917</td>
<td>9.011</td>
<td>.0009</td>
</tr>
<tr>
<td>THESIS.Policies</td>
<td>1.9754302</td>
<td>1</td>
<td>1.9754302</td>
<td>12.190</td>
<td>.0015</td>
</tr>
<tr>
<td>2-FACTOR INTERACTIONS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.MecTHESIS.Pol</td>
<td>1.2398132</td>
<td>2</td>
<td>.6199066</td>
<td>3.825</td>
<td>.0331</td>
</tr>
<tr>
<td>RESIDUAL</td>
<td>4.8615868</td>
<td>30</td>
<td>.1620529</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL (CORR.)</td>
<td>10.997214</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 missing values have been excluded.
### Analysis of Variance for THESIS.VarReduced

<table>
<thead>
<tr>
<th>Source of variation</th>
<th>Sum of Squares</th>
<th>d.f.</th>
<th>Mean square</th>
<th>F-ratio</th>
<th>Sig. level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MAIN EFFECTS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.Mechanisms</td>
<td>1477.8618</td>
<td>2</td>
<td>738.93089</td>
<td>5.560</td>
<td>.0088</td>
</tr>
<tr>
<td>THESIS.Policies</td>
<td>399.2004</td>
<td>1</td>
<td>399.20040</td>
<td>3.003</td>
<td>.0934</td>
</tr>
<tr>
<td><strong>2-FACTOR INTERACTIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THESIS.MecTHESIS.Pol</td>
<td>28.245650</td>
<td>2</td>
<td>14.122825</td>
<td>.106</td>
<td>.8995</td>
</tr>
<tr>
<td><strong>RESIDUAL</strong></td>
<td>3987.3572</td>
<td>30</td>
<td>132.91191</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTAL (CORR.)</strong></td>
<td>5892.6650</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 missing values have been excluded.