MICROCOMPUTER BASED MANAGEMENT INFORMATION SYSTEM

By

Francis Lin

B.S., Tatung Institute of Technology, Taiwan, R.O.C.,1971

A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1986

Approved By:

RAm Bridle

Major Professor

TABLE OF CONTENTS

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation For This Project

In recent years, managerial costs have been increasing while those of computer processing have been decreasing; therefore, to employ computers to aid in managing a business is highly desirable. One way of using a computer to aid management is through the use of a Management Information System (MIS). MIS is a computer software package which manipulates data in databases to provide users with needed information for effective business planning and to support decision making.

1.2 Requirements Specification

The requirements of a Management Information System are different from organization to organization based upon their practical needs. However, some features are generally demanded by most business fields. Examples of generally useful features include :

- Providing the sales personnel with the cost analysis of a given product together with the estimated delivery time for given quantities of the product.
- Keeping track of raw material inventory status in order to make that material available without undue inventory buildup or delay in the production schedule.

The system that this project has implemented is a MIS which incorporates features that can be utilized in a wide spectrum of businesses. In order to tailor the system to a specific business organization, users can either reconfigure this system or add

more functions and thereby make it more flexible, and suitable for their specific needs.

The major features that this project is concerned with include: Production Requirements Planning, Material Requirements Planning, Cost Analysis, Financial Management, and Management Operation Research. The following five paragraphs describe each of these concerns in turn.

Production Requirement Planning - Given a certain quantity of a product, this feature provides the cost analysis and an estimated delivery time for the product. The delivery time is based upon the past quarter's inventory status and the production rate of the product. This feature enables the user to produce an estimate of the selling price and delivery time for a customer upon receiving an order or an inquiry.

Material Requirement Planning - For a given raw material name, this feature will respond with a suggested, best reordering level. This is accomplished by calculating the consumption rate of the raw material over the past quarter year and then referring to the delivery time in recorded the purchasing history of this material.

Cost Analysis - Given a product name and a past time period, this feature will respond with a cost analysis report of the product for the given period of time. This feature also possesses a graphics capability, which can be used to display the results in a chart form.

Financial Management - After entering a time period, this feature will respond with a debt/asset balance table of the organization. This is accomplished by referring to the accounting information stored in the organization's database. In addition, this subsystem also possesses the ability to show the details of different financial categories so that a manager can quickly obtain an idea of the organization's financial status.

Management Operation Research - This feature uses the simplex linear programming approach to give an optimal value for a set of given constraints. In this feature, the user will be requested to define the constraints of the problem. While defining constraints, the system provides three options for users:

1) Select a set of predefined constraints that already exist in a criteria file to perform the linear programming,
2) Modify a set of predefined constraints,
3) Create a set of new constraints.

Each coefficient in the constraints can be a either user defined constant, or a datum derived from the database.

1.3 MIS Functional Overview

The environment provided by a MIS normally includes a Database Management System (DBMS), along with a set of related database files. (The DBMS is a data manipulation system which acts as a data librarian; it stores and retrieves data.) In order to insure that the system provides correct information, a user must define: a set of criteria on how the system will select input data, what

operation(s) will be performed on the data, and how the results will be used [KRO84].

Storing the user defined criteria is the initial process of the prototype MIS system. First of all, the system will perform a series of queries or questions to the user based on the information needed by the system. All of the user's responses to the queries will be stored into a criteria file for later use. During its execution, the application programs will extract data from the database based on elements of the selected criteria file. Finally, the extracted data will be converted into a text file whose format allows it to be shared by programs which are not written in dBase III programming language.

1.4 Organization Of The Report

Chapter II of the report provides an overview of Computer Based Information Systems and reviews some of the literature on Management Information Systems. The overall system design with the data structure and the control flow diagram of each module is presented in Chapter III by examining the data and control flow of each module in the author's system. Chapter IV presents the conclusions that have been reached during the construction of this prototype MIS, also described are ideas for future work related to this project. Finally, appendices are provided for those who intend to use the system or modify the system's programs. Appendix-A contains a description of how a user can interact with the system, and a listing of the dBase III record descriptors used in configuring the system for a hypothetical business. The entire source code of the system is listed in Appendix-B.

CHAPTER II

LITERATURE REVIEW

2.1 Types Of Computer Based Information Systems

Computer Based Information Systems (CBIS) [KRO84] can be classified into four major types: transaction processing systems (TPS), office automation systems (OAS), management information systems (MIS), and decision support systems (DSS). The major differences between these four types of CBIS are summarized as follows:

Transaction Processing Systems process data transactions on a database and permit operations such as adding, deleting, indexing, sorting, and updating data. This type of system also allows the user to generate detailed data reports.

An Office Automation Systems performs such functions as the manipulation of office documents, managing the scheduling of appointments, and address lists, etc.. It is a multifunction, integrated type of system that allows many office activities to be performed in an electronic mode.

Management Information Systems process management-oriented data transactions and generate reports. This type of system consists of simple, preprogrammed management information modules which generate routine reports to support structured decision making.

Decision Support Systems may contain an artificial intelligence component [HUS81] which allows users to interactively ask "what if" questions; it also possesses graphic tools which can

display the results along with some decision models which can generate information needed to support semi-structured or unstructured decision making tasks.

In this chapter, the functions of each of these types of computer based information systems are introduced. There is a functional overlap of these systems when they have common goals. Therefore, how a TPS provides support of a MIS, and how a MIS to be used as an aid to DSS, will be briefly discussed in section 2.2 and 2.3. Lastly, it is shown how the subsystems of the prototype MIS developed by the author support managerial activities in real firms.

2.2 Management Information Systems

Management Information Systems are the outgrowth of the data processing systems of the late 1950s [KRO84]. Whereas data processing was concerned primarily with record keeping such as payroll and billing, a MIS is concerned more with managerial functions, such as planning, controlling, and decision making. The record keeping and other clerical processes are still needed in MIS, but they are there mainly to satisfy the information needs of management. Therefore, the functions provided by TPSes give a necessary support for a MIS.

2.3 Relations Between MIS And DSS

In highly-technological firms, rapidly changing environmental factors make planning decisions both difficult and critical. A MIS can be adapted readily to the specialized requirements of

decision support. However, in order to support semi-structured or unstructured decisions, an interactive MIS is needed. Artificial intelligence, a report generator, and some specific simulation models can be combined with a MIS to allow the user interactively access to the database and ask "what if" questions. The resulting system supports semi-structured or unstructured decision making. Such an upgraded, interactive MIS is called Decision Support System.

2.4 MIS In Support Of Management Control

Management control is resource-oriented. In organizations, managers need to have control over the acquisition and use of the basic resources, such as material, labor, time, and money in order to carry out production tasks. The management control is concerned with: resources and production, comparative prices of input resources, and the return on investment. The MIS is used to compute the efficiency with which resources are used, and to provide information such as costs of the products, so that the return on investment can be maximized.

2.5 DSS As An Aid To Strategic Planning

Strategic planning is oriented toward the establishment of long-range organizational goals and objectives. The issue of strategic planning is not the tasks to be performed but the purpose of performing them.

Organizational goals and objectives take many forms. They may define the scope of organizational activities, normally, the

products to be made, the services to be offered, or the markets to be reached. Goals may also establish performance standards, such as, market share, profit, total sales, and the like. Used in another way, they may convey the future direction of the organization. A MIS can do little more than provide and analyze historical data in support of strategic planning. But just as TPSes have complemented MIS in management control functions, A MIS complements decision support systems in order to provide needed assistance in strategic planning.

2.6 MIS In Real Organizations

In most organizations, departments are formed by grouping together individuals responsible for similar functions. An accounting department is made up of those individuals responsible for accounting functions, a production department consists of the people who carry out the production activities, and so forth. Not all organizations perform the same business functions, but the principle of functional organization applies nonetheless. Managers in departments, particularly mid-level types, who tend to be the primary users of MIS, have few responsibilities that require information about the activities of the other functionally different departments.

Although the division of a MIS into functional subsystems may be convenient for middle or first-line managers, top-level executives are more likely to need information based on two or more functions. To serve these needs, a MIS must be able to exchange information among the functional subsystems and to

integrate functional information for strategic planning. For example, in forecasting the production schedule, a system must integrate the information of inventory status from the production department, sales records from the marketing department, and profit analysis (including the cost analysis and the sales price of the products) from the financial department. Thus, it is useful to allow a MIS to have access information from all functional departments in an organization.

Department-oriented functions represent only one level of MIS subsystems. In large organizations, functional departments are further subdivided into more specialized sections or divisions. For example, a production department may have engineering, quality control, purchasing, and other divisions. Each subunit has its own peculiar information needs, also the subunits of a department must share information with each other as well as with other functional departments. A MIS can reflect this structure by incorporating specialized modules representing the functional subsystems.

2.7 Project Comparisons

The implemented project consists of five subsystems: Production Requirements Planning for use by a marketing department, Material Requirements Planning for a production department, Cost analysis and Financial Management for a financial department, and Management Operation Research. The first four subsystems only permit the user to perform some specific functions at the managerial control level. The fifth sybsystem, Management Operation Research,

can be used in strategic planning. For example, to find the most profitable mix of products to manufacture.

The MIS which this project has implemented processes data transactions using a hierarchy of modules and generates routine reports in an effort to support structured decision making. For example, the Production Requirements Planning extracts data from the daily production record files based upon a set of predefined criteria. After the data extraction process has completed, this subsystem then calls the report generation module to manipulate the extracted data and to generate a report which contains the cost of a given product together with an estimated delivery time for given quantities of the product. This planning feature can be used to support sales personnel's offering price quotation to their customers. Because of the features offered, and the way they are utilized, the proposed system can be classified as a Management Information System. The following five paragraphs describe the use of each of the five subsystems.

The Production Requirements Planning subsystem is concerned with the sales of the products. The entire marketing effort ultimately culminates in sales, and the success or failure of marketing is often attributed to sales personnel. This subsystem addresses the function of supplying the sales personnel with the cost analysis of a given product, and providing them with the estimated delivery time for given quantities of that product. The inputs to this subsystem include the daily production records, and the current inventory status (to calculate the estimated

delivery time), and the cost analysis of the product (from financial department). With this subsystem, a sales manager can effectively offer customers a price quotation for a product, and the estimated delivery time of a desired quantity of the product.

The Material Requirements Planning (MRP) combines two of the most important activities in a manufacturing operation, namely, the scheduling and control of materials. Scheduling can be viewed as very short term planning in which specific times are assigned for various production activities. The material control activity is concerned with the quantity of the materials that are needed to produce the products within the specific time. MRP integrates material control with scheduling by making production materials available in a timely manner, without undue inventory buildup or delays in the production schedule. The inputs to this subsystem include the production schedule, the current material inventory status, and the purchasing history file (to calculate the average delivery time of purchased material). The use of MRP is most appropriate when the demand for a manufactured product is subject to a great deal of variation and the product is assembled from subcomponents.

The Cost Analysis subsystem is designed to analyze costs month by month so that the top-level managerial personnel can understand the variation of production costs. The results of this subsystem are normally used by the executive manager to decide the sales price, or to find out the production factors which have caused the cost to increase. The subsystem can also be used in combination with the linear programming package to set up a

production schedule which maximizes profits. This subsystem accesses all of the production cost files including fixed and variable costs, and then divides each of the costs by the production quantity of the given product to get the average values. The output of the results can be displayed in either numeric or graphical form.

The Financial Management subsystem includes the classification, recording, and summarization of monetary transactions. The primary purpose of this subsystem is to paint a financial picture of the organization for investors and creditors, and to satisfy legal requirements. This subsystem is also designed to possess the ability to show the details of the financial accounts such as the cash on hand, the accounts receivable, notes payable, wages, etc., so that the manager can quickly obtain an idea of the organization's financial status. The inputs to this subsystem include the fiscal year whose financial statements are to be analyzed, and all of the accounting ledger files from which the accounting information are extracted.

The Management Operation Research subsystem provides a linear programming technique to find the optimal solution to various problems. For example, linear programming method can be used to find the optimum mix of products to manufacture under certain production constraints, or to determine the most profitable combination of production lines. The subsystem can also be used to find the most profitable combination of investments that meets a client's investment philosophy, the least expensive shipping

plan for a distribution network, or the solutions to many other typical problems.

The system that this project has implemented only shows a prototype of the microcomputer based MIS. Since the requirements of a MIS are different from one organization to another, the problem of how to reconfigure this system and expand it to fit with a particular user's requirements must be solved each time the system is applied to a new business.

CHAPTER III

SYSTEM DESIGN

3.1 System Overview

The objective of this project is to implement a microcomputer based Management Information System which is suitable for small to medium sized corporations. As the previous chapter mentioned, not all firms require the same MIS functions, but the principle of functional organization applies nonetheless. The microcomputer based Management Information System that this project has implemented only allows users to fill some of their specific needs at the managerial control level. However, this project did result in a basic MIS which can be configured to suit a user's needs, and proved that it is feasible to implement a microcomputer based MIS without an extensive commitment of manpower.

There are two ways in which the reconfiguration of the prototype system might be accomplished; one is to change the contents of source program modules to comply with a user's specific needs. For example, a programmer can change the items of a selection list, which have been included in the menu generation module, to fit an organization's needs. Another configuration technique is to write a system installation program which stores all of the relevant configuration information into a data file so that later the system modules can make use of it. For example, to configure a menu generation module, the installation program might ask the user to input all of the items which should be contained in the menu, and then store these items into a configuration data file.

Later, when the menu generation module is executed, the system will read the contents of the menu from the configuration data file, and replace the variables in the module with data read from the configuration data file. A sample algorithm for the menu selection module and its configuration procedure are respectively presented in Figure 3-1 and Figure 3-2.

```
READ ITEM_NUMBER;
WRITE ITEM_NUMBER TO DATA_FILE;
ITEM_COUNT <-- 0;

WHILE ITEM_COUNT < ITEM_NUMBER DO
   ITEM_COUNT <-- ITEM_COUNT + 1;
   READ INPUT;
   WRITE INPUT TO DATA_FILE;
END WHILE;
END CONFIGURATION PROCEDURE;
```

Fig. 3-1 Algorithm For Configuration

```
READ ITEM_NUMBER FROM DATA_FILE;
ITEM_COUNT <-- 1;
CONVERT ITEM_COUNT TO STRING;

WHILE ITEM_COUNT <= ITEM_NUMBER DO
   READ CONTENT FROM DATA_FILE;
   WRITE 'SELECT &STRING &CONTENT';
ENDDO;
END MENU SELECTION PROCEDURE;
```

Fig. 3-2 Algorithm For The Menu Selection Module

The programming languages and environment which were selected to the host prototype Management Information System on micro-computers were dBase III, a relational database management system from Ashton-Tate, and Turbo Pascal from Borland International. The reason for using both dBase III and Turbo Pascal as tools is due to their disparate capabilities. The dBase III programming language provides excellent support for data transaction processing, but it is weak in certain problem solving areas. Examples

of these shortcomings include: an awkwardness in attempting to produce a hard copy in a desired format from dynamically created variables; also, generating a graphical output is difficult. Therefore, Turbo Pascal has been utilized to handle those problems which dBase III cannot solve.

The system that this project has implemented consists of five subsystems:

(1) The Production Requirements Planning which provides the cost analysis and the estimated delivery time of a given product,

(2) The Material Requirements Planning presents a suggested, best reordering level for a given material,

(3) The Cost Analysis subsystem provides the cost analysis of a given product on a monthly basis within a selected time period,

(4) The Financial Management subsystem gives a view of the financial status of an organization, and

(5) The Management Operation Research provides a linear programming software package which allows the optimal value for a set of given constraints to be calculated.

Since the details of the system module descriptions and their documented source code is presented in APPENDIX-B, this chapter only gives the general design of the project.

3.2 MIS Control Module

In order to make the system more "user friendly", a main selection menu is provided which allows the user to access any of the five subsystems through a single key stroke. Figure 3-3 gives the structure of the control module. The MIS control module displays all functions, that are included by the system, through the use of a main selection menu; it then waits for the user's selection. After receiving the user's response, this module will call the selected subsystem routine and respond to the user with a result. The details of each of the subsystems are depicted in separate subsections.

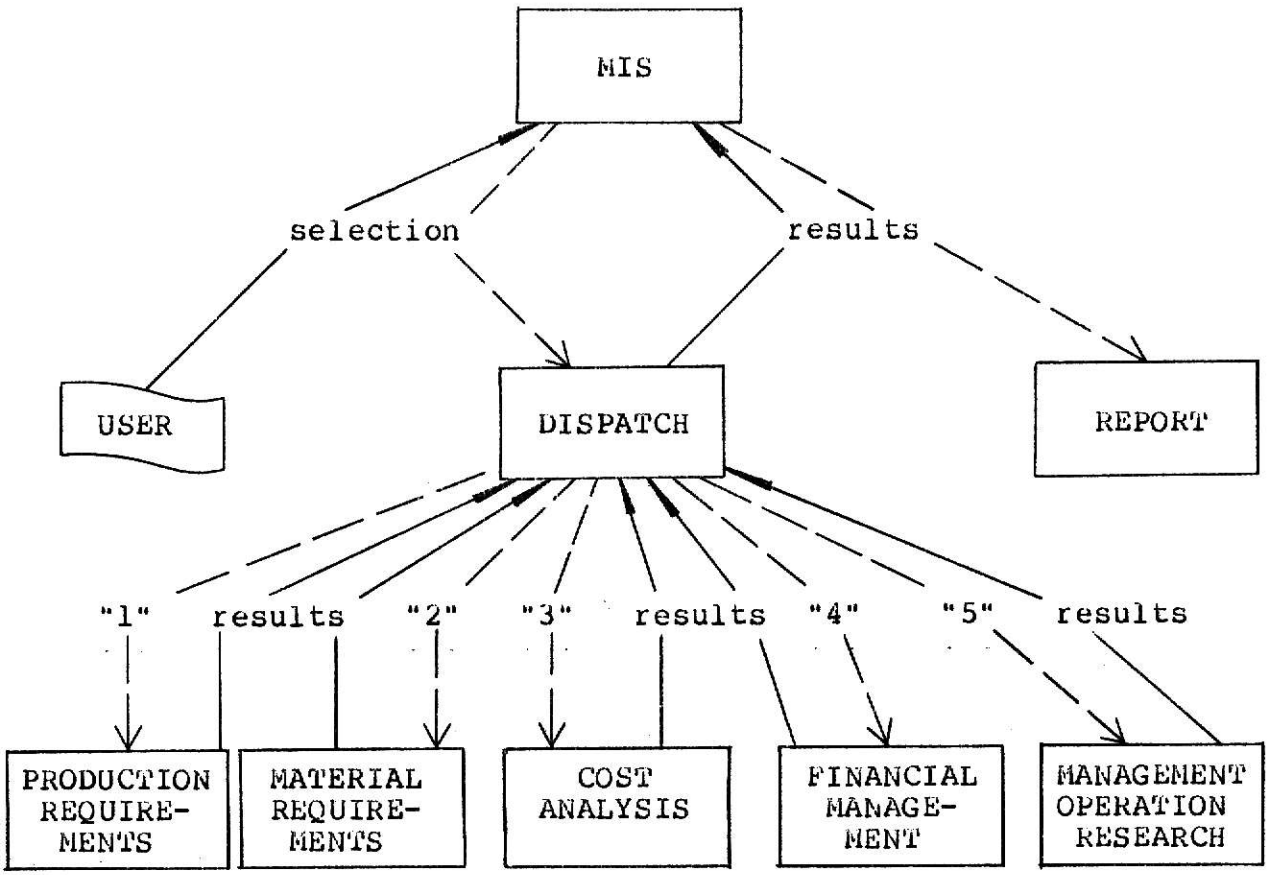

Fig. 3-3 MIS Overview Diagram

3.3 Production Requirements Planning

The Production Requirements Planning subsystem allows the function of cost analysis for a given product to be performed. Also, it provides the estimated delivery time for a desired quantity of some product. With this subsystem, a sales manager can effectively give a customer the price quotation of the requested product and the estimated delivery time of the quantity needed.

Figure 3-4 is a diagram which captures both the control flow and the data flow of this subsystem. It first calls the PRODUCT NAME INQUIRY module which asks the user to input a product name and the quantity of it which is required. The PRODUCT NAME INQUIRY module will also access the daily production records and the inventory status to calculate an estimated delivery time which is based on the quantity required. After the PRODUCT NAME INQUIRY module has finished executing, the subsystem will call the FIXED COSTS module in order to access all of the different fixed cost files in the database (such as equipment depreciation, the production management cost, and the fixed labor costs) in order to calculate the amount of each of the fixed costs which is involved. The next step is to call the CRITERIA FILE SELECTION module. This module has three options for the user to select from: CRITERIA FILE GENERATION, CRITERIA FILE MODIFICATION, and USE EXISTING CRITERIA FILE. Each of these options corresponds to a module that performs specific tasks for the subsystem. The following three paragraphs describe the detailed structure of these modules and their usefulness.

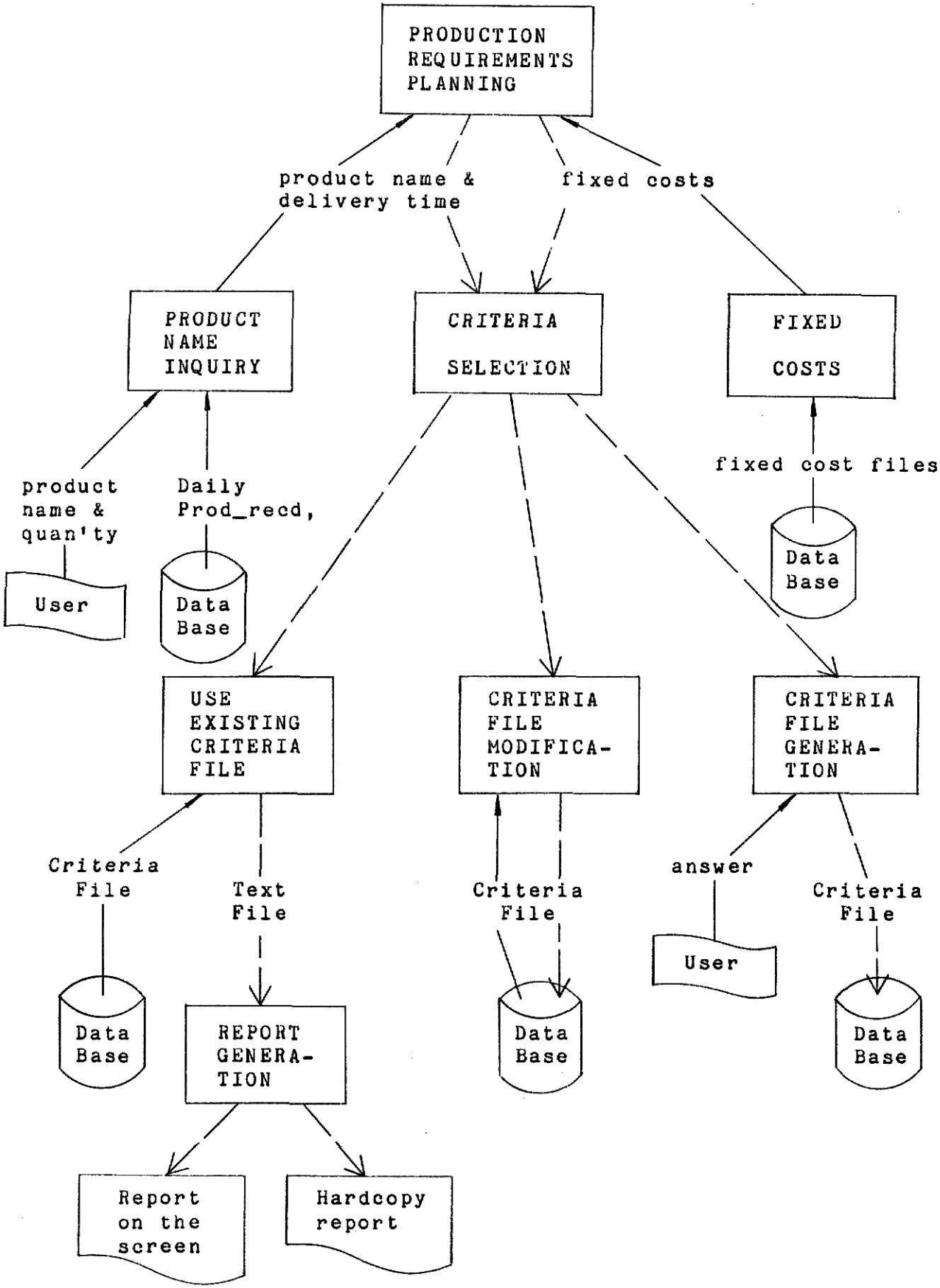

Fig. 3-4 Production Requirements Planning Structure Diagram

CRITERIA FILE GENERATION - generates a series of questions whose answers will specify conditions for extracting data from the database. The questions include asking for: the name of the derived cost which will identify the derived datum, the file name of the database to be worked with, the name of a field to project on, the selection criteria (or the filter scope [ASH84]), and the statistical function such as AVERAGE and SUM which will be used to derive data. All of the user's responses to these questions will be stored into a database file for use in data extraction. At the beginning of this module, the system will ask the user to input the number of records which will be included in the criteria file; each record represents information relating to one extraction of data from a database file. After the number of records has been entered, the system will ask the above mentioned questions for each of the number of defined data extractions. Whenever the actual execution of data extraction module begins, the system will read the extraction information from the criteria file, and then extract data based upon these conditions. Details of how each criterion is used, are shown in the following USE EXISTING CRITERIA FILE module.

USE EXISTING CRITERIA FILE - this module extracts data from a database file based on the conditions specified in a record of the designated criteria file. It then proceeds to convert the retrieved data into a text working file. The text working file is used to provide data interchange between different programming environments. This conversion process is necessary because dBase-III files contain control information useful in the dBase III environment in addition to data values.

Whenever this module is first invoked, the system will ask the user to input a criteria file name. Next, the data extraction function is called to perform a projection and selection of data based on the records stored in the criteria file. The criteria file consists of five fields: cost name, file to use, field to use, filter scope, and function. The dBase III-oriented algorithm for data extraction is presented in Fig. 3-5.

```
WHILE .NOT. EOF(Criteria File) DO
   READ Cost_name
   READ File_to_use
   READ Field_to_use
   READ Filter_scope
   READ Function
   OPEN File_to_use
   Function Field_to_use TO Variable FOR Filter_scope
   Convert the Variable to a text file
   SKIP
ENDDO
```

Fig. 3-5 Algorithm For Data Extraction

After the data extraction process has completed, the module will call the external REPORT GENERATION program. The REPORT GENERATION program will read all of the costs in the text working file and generate a report. The object code of programs such as the REPORT GENERATION program, which is written in Turbo Pascal, and the MPS-PC linear programming package, which is written in Basic Language, can be executed from inside the dBase III environment by simply using the command "RUN <file_name>".

CRITERIA FILE MODIFICATION - allows the user to select an existing criteria file and then to modify its contents. This module will first display all of the old criteria on the screen, and then allow the user to modify each criterion in turn.

3.4 Material Requirement Planning

The goal of the material requirement planning feature in this project is to provide the user with a best, suggested reordering level for replenishing a given material. With this subsystem, users can make materials available without undue inventory buildup or delays in the production schedule. Figure 3-6 is a Yourdon-Constantine diagram [YOU79] of the subsystem. This subsystem starts with the MATERIAL NAME INQUIRY module which asks the user to input a material name. Next, the system will access the daily production record file, which also contains the material consumption records, in order to calculate the daily consumption rate of the specified material. After this calculation has completed, the subsystem will access the purchasing history file to calculate the average delivery time for the material. The suggested material reordering level is obtained from multiplying the daily consumption rate by the average delivery time of purchasing the material.

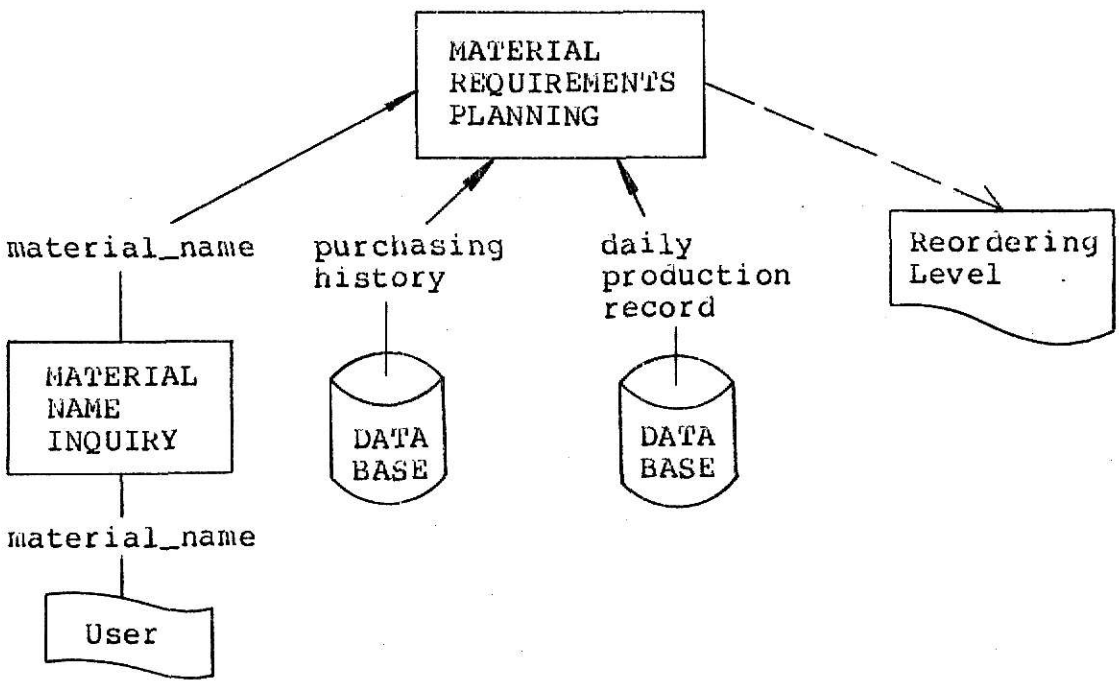

Fig. 3-6 Material Requirement Planning Structure Diagram

3.5 Cost Analysis

The purpose of the Cost Analysis subsystem is to analyze costs on a month-by-month basis so that the executive manager can view the variation of production costs. The results of the subsystem can also be used to decide a sales price, or to find out the production factors which cause a product's cost to increase.

Fig. 3-7 gives the structure diagram of the subsystem. The subsystem accesses all of the production files which contain either fixed cost or variable cost in order to get the total cost of a given product over a time period. (Variable costs are items of cost that vary directly with the production volume; for example, direct labor, direct material, and power costs are treated as variable costs. Fixed costs are items of cost that do not vary at all with the production volume; for example, the costs such as building depreciation, property taxes, supervisory salaries, and the depreciation of production facilities are treated as fixed costs [LAU84].) Each production cost will be divided by the total production quantity of the given product to yield the average cost contribution per product unit.

To start this subsystem, the user will be requested to input the product name along with the starting and ending date over which the costs are to be analyzed. The only structure difference between this subsystem and the cost analysis performed in the Production Requirements Planning is that this subsystem analyzes monthly costs rather than analyzing costs based upon the previous quarter's production records.

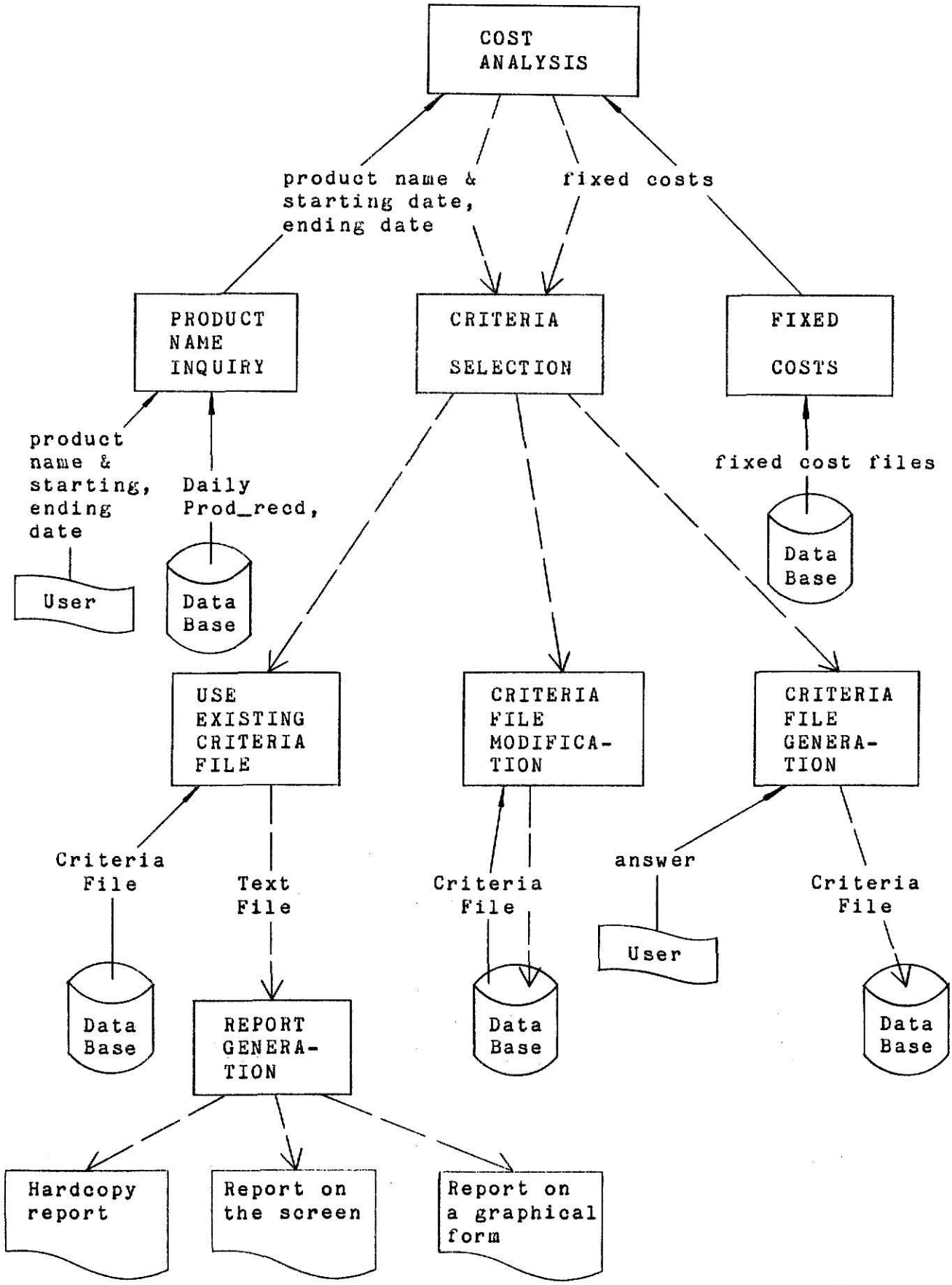

Fig. 3-7 Cost Analysis Structure Diagram

3.6 Financial Management

The purpose of the Financial Management subsystem in the prototype MIS is to provide a financial statement of the organization for investors and creditors. This subsystem also possesses the ability to show the details of different financial categories so that a manager can quickly obtain an idea of the organization's financial status, in terms such as the cash on hand, the accounts receivable, and the notes payable.

Fig. 3-8 is a diagram which captures both the control flow and the data flow of this subsystem. It starts with asking the user to input a fiscal year over which the financial statements will be summarized. Next, the SUM ACCOUNT module is invoked to respectively sum the elements of each of the account categories (which include asset, liability, equity, expense, and revenue), into variables for later use. After the execution of the SUM ACCOUNT module has completed, the system will call the PRINT OUT RESULT module to ask the user to select an output destination. If the user chooses "screen", then the SHOW RESULT module is called to display the results on the screen. On the other hand, if the user selects "printer", then the PRINT OUT RESULT module will convert all of the information to a text working file, and then call the PRINT FINANCIAL external program to generate a hardcopy report. When the execution of the SHOW RESULT module is finished, the subsystem will call the SHOW DETAIL module to display an account category menu for the user to select from. The account category menu includes: SHOW ASSET, SHOW LIABILITY, SHOW EQUITY, SHOW EXPENSE, and SHOW REVENUE. Each of these items corresponds

to a submodule that displays a set of account ledger files for the user to select from, and then shows details of the ledger file based upon the user's selection.

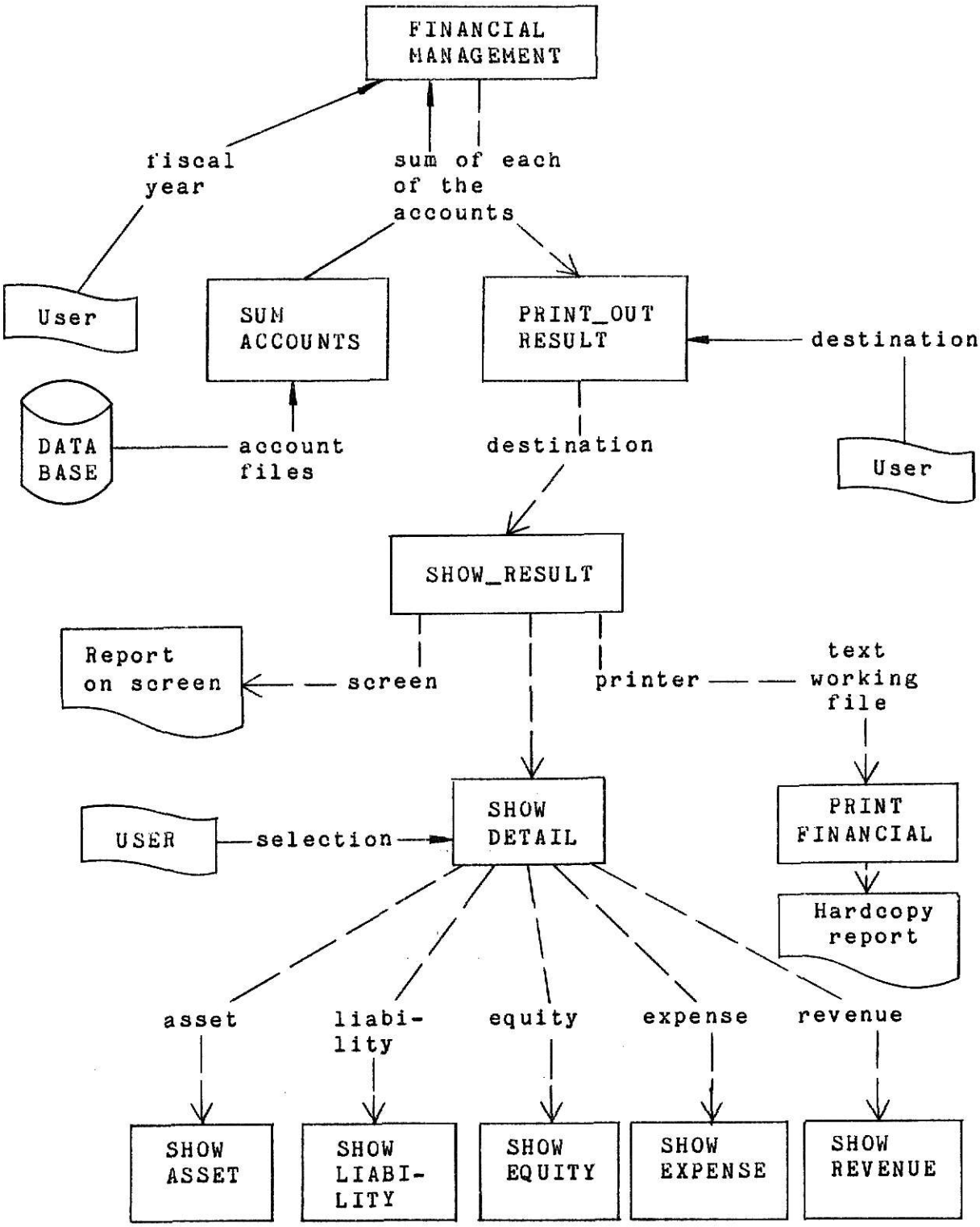

Fig. 3-8 Financial Management Structure Diagram

3.7 Management Operation Research

The Management Operation Research subsystem provides a linear programming software package for the user to use in carrying out strategic planning tasks. Typical examples of these tasks include, finding the optimum mix of products to manufacture under a specified set of production constraints, and searching for the most profitable combination of investments that meets a client's investment philosophy.

To apply operations research techniques to business problems, activities such as problem definition and constraint formulation must first be attacked. Problem definition is a specification of the objective of the operations research effort [ECK76]. An example of a common objective is to find the most profitable combination of products to manufacture. In this example, the objective of the effort is "profit", therefore, the profit of each of the products must first be analyzed. The activity of formulating constraints must enumerate all relevant factors that influence the objective. For example, time available, material limitations, and legal requirements (such as the minimum production quantities in order to complete sales contracts) are the factors that might influence the objective. The problem definition and the constraint formulations are to be respectively interprated in terms of the objective function and the side constraints for use of the linear programming package.

In order to define a statement of a business problem to be solved, this subsystem submits a tool for generating/modifying the specification of the objective and the constraint formulations. The specification is used in finding the optimum solution to a

problem by means of a linear programming method. This subsystem will also reformat the constraint formulations to comply with the MPS-PC Linear Programming Package [RES84] which runs on IBM compatible microcomputers. Additionally, this subsystem provides a way to derive information, for use as a coefficient of a constraint, from databases. For example, the system can be requested to access a material inventory database file, and extract the current material inventory status for use as the coefficient of the material limitation constraint. Details of how the derivation process is performed will be shown shortly. This additional feature enhanced the power of the linear programming system, and helped to automate it.

Fig. 3-9 gives a system structure diagram for the Management Operation Research feature. The subsystem starts with the CONSTRAINT SELECTION module which displays a constraint selection menu for the user to select from. The selection menu contains three items: Constraint Generation, Constraint Modification, and Convert Constraint. Each of these items corresponds to a module that performs requested tasks under the subsystem.

Constraint Generation - initializes the definition of the problem activities by allowing the user to define factors which influence the business activities (i.e., the constraints of the problems); these constraints will be saved in a database file for later use.

This module starts by showing a series of questions which ask for : the name of the file to contain information about the criteria, the type of problem (maximum or minimum), the number of constraints, the number of variables, the number of constraints

involving a "greater than" inequality, the number of constraints involving equality, and the number of constraints with a "less than" inequality. After the user has answered all of the questions, the module will ask the user to define the coefficients of each of the constraints. A coefficient can be defined as either a constant or a datum derived from databases. If a coefficient has been defined as a derived datum, then the module will call the DERIVE module to extract the data from databases. The DERIVE module provides the user with a set of questions that define: which file the data is going to be derived from, how to derive the data, and what function must be performed in order to derive the data.

Constraint Modification - allows the user to modify a set of predefined constraints. It first shows a menu for the user to select the item which he intends to modify. The selection items include: the objective function coefficients, the type of the problem, the coefficients of the constraints, and the type of the constraints (greater than, equal to, or less than). After the user has completed the selection, the help menu of the system will guide the user through the modification.

Convert Constraints - allows the user to select a predefined constraints database file. The selected file is then converted to a text working file. After the conversion process has completed, this module will call the external REFORMAT program to reformat the text working file so that the data which, represent the constraints, are in a form acceptable to the MPS-PC linear programming package.

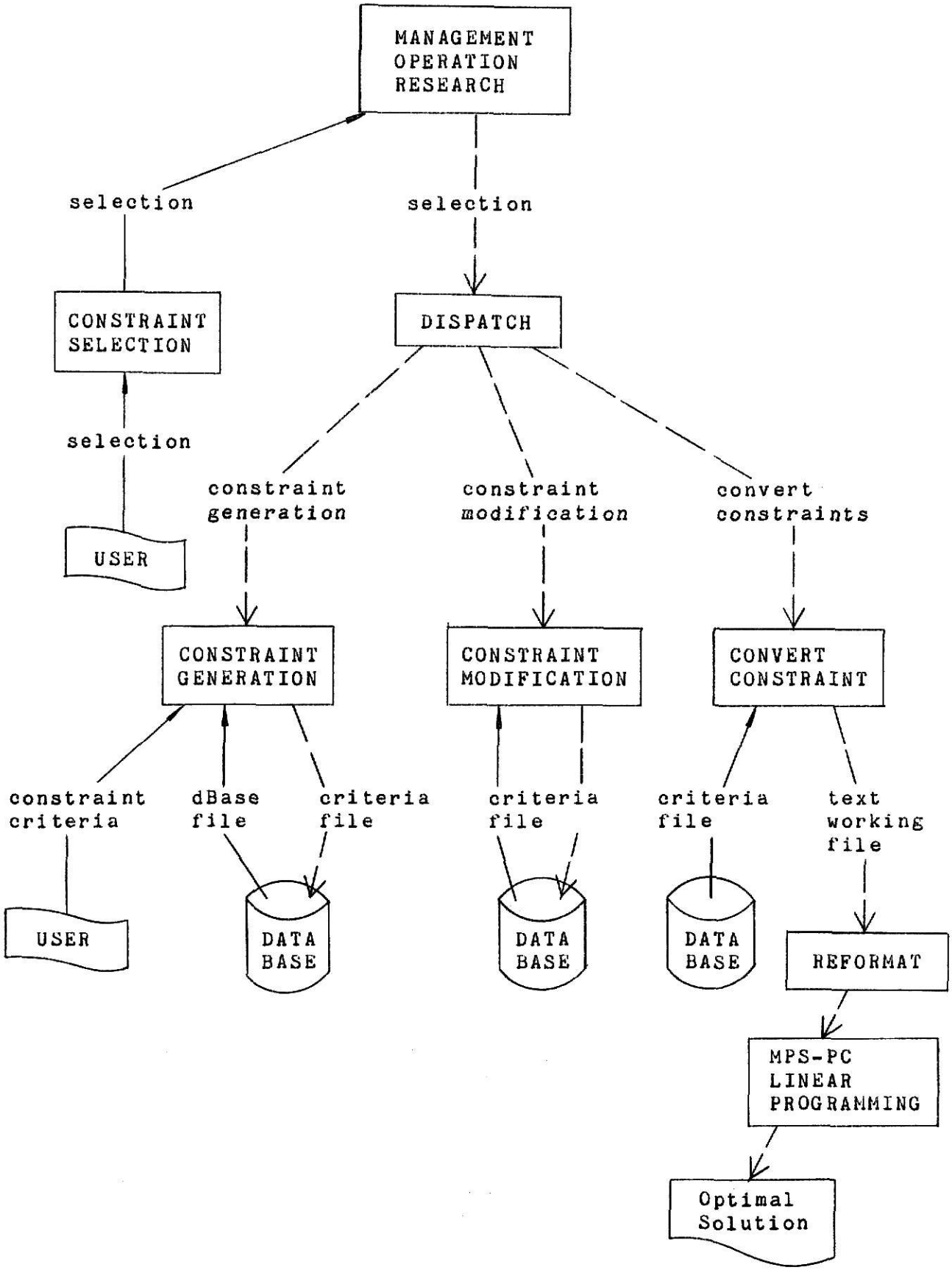

Fig. 3-9 Management Operation Research Structure Diagram

3.8 Conclusions

The prototype MIS includes five subsystems: Production Requirements Planning, Material Requirements Planning, Cost Analysis, Financial Management, and Management Operation Research. The two main reasons for selecting these five subsystems is that they can be employed by a wide variety of business organizations and they support the primary mission of a MIS. The mission of a MIS is to form an efficient way of using historical data to provide users with the needed information to support present decision making, and to support future business planning.

All of the five subsystems perform basic functions which are generally demanded by most businesses; these can be reconfigured to fit with any business organization's needs. In the beginning of this chapter, the author described the system configuration strategy. How to reconfigure the databases to fit with an organization's needs will be described in Appendix-A. Moreover, this system can be used as the basis for future MIS implementations, each of which can be fitted to meet the needs of a specific business. Therefore, the author believes that this prototype MIS can be a valuable aid when studying or implementing microcomputer based Management Information Systems.

CHAPTER IV

CONCLUSIONS AND FUTURE WORKS

For the last decade, Management Information Systems have been widely used in businesses to support executive, managerial, and administrative activities by providing informational and computational resources. However, because the execution of a traditional MIS requires a large amount of computer memory, MIS previously ran only on mainframes. As microcomputers have been aggressively developed in recent years, their memory addressing ability has been greatly improved (such as the memory of IBM PC-AT which is extensible to three mega-byte); thus, it has become feasible to construct a microcomputer based MIS.

The prototype MIS developed by the author was targeted toward the environment of a small concrete reinforced rod mill which produces various sized concrete reinforced rods in four production lines. Detailed input and output specifications of this system are presented in Appendix A (User's Guide). It should be noted that some of the selection menus demonstrated on the system are only suitable for the rod mill business. However, to meet their specific needs, users can use the configuration methods which were mentioned in CHAPTER III, to reconfigure the system.

The hardware environment for the prototype MIS requires a 16 bit word microcomputer with at least 560K of random access memory, a terminal, a printer, and a Winchester disk storage device. dBase III itself needs a 320K working area in order to open up to ten database files at a time. Additionally, the supplemental

software packages invoked by the system needs extra memory to run inside the dBase III environment. The Winchester disk storage device is needed for updating and storing the large number of daily records, because dBase III doesn't allow a file to be split and separately stored on two or more floppy disks.

The requirements of a MIS varies from one organization to another and are based upon an organization's practical needs. The proposed system can be used as the basis of a future MIS implementation and can be fitted to meet the needs of a specific business. Some features which can be added to the proposed system without any reconfiguration of the dBase III database schema include:

- A procurement information system which will trigger a warning whenever the material inventory status reaches the minimum reordering level. The system can also give users a list of supplier for the material. This list could be ordered by either cost per unit or estimated delivery time.

- An automatic tracking system for accounting transactions. This system will automatically compare the credit and debit of each of the relevant accounting transactions, to make sure that all of the input for accounting transactions are correct. This supplemental system can give users a list of abnormal accounting transactions by means of comparing the cross reference number (i.e., an attribute contained in the accounting database system).

This report has presented a skeletal MIS and proved that a microcomputer based MIS is feasible. The only trade off of the microcomputer based MIS is its less than rapid processing speed, which is caused mainly by the slowness in handling the disk I/O. Quite obviously, the microcomputer based MIS is a high productive and useful system which will soon be widely used in the business field.

REFERENCES

[ADR84] ADR/EMPIRE case study, Applied Data Research, 1984.

[ANT75] Anthony, Robert N., Reece, James S. Management Accounting, Richard D. Irwin, Inc., 1975.

[ARI85] Ariav, Gad, and Ginzberg, Michael J. "A Systemic View of Decision Support," Communication of the ACM, 28(10):1045-1052, October 1985.

[ASH84] dBase III User's Manual, Ashton-Tate Inc., 1984.

[BOR85] Turbo Pascal Version 3.0 Reference Manual, Borland International, 1985.

[ECK76] Eck, D. Roger Operation Research for Business, Wadsworth Publishing Company, Inc., 1976.

[GIL76] Gilett, Billy Z., Introduction to Operation Research, Mcgraw-Hill, Inc., 1976.

[GUP75] Gupta, Shiv K., Cozzolino, John M. Fundamentals of Operations Research for Management, Holden-Day, Inc., 1975.

[HUS81] Hussain, Donna, Hussan, K. M. Information Processing Systems for Management, Richard D. Irwin, Inc., 1981.

[KAT84] Katzan, Herry Jr. Management Support Systems, Van Nostrand Company Inc., 1984.

[KRO84] Kroeber, Donald W. Watson, Hugh J. Computer Based Information System, Macmillan Publishing Company, 1984.

[KRO82] Kroeber, Donald W. Management Information System, Free Press, Inc., N.Y. 1982.

[LAU84] Laughlin, Eugene J., Financial Account, John Wiley & Son, 1984.

[LIE75] Lientz, Bennet P., Computer Application in Operation Analysis Prentice-Hall, Inc., 1975.

[RES84] MPS-PC User's Manual, Research Corporation, 1984.

[REI85] Reimann, Bernard C., and Waren, Allan D. "User-oriented criteria for the selection of DSS software," Communications of the ACM, 28(2):166-179, February 1985.

[SEN82] Senn, James A., Information Systems in Management, Wadsworth Company, 1982.

[YOU79] Yourdon, E. N. and Constantine, Larry L., Structure Design, Prentice Hall, 1979.

APPENDIX A

USER'S GUIDE

A. 1 Introduction

A sample microcomputer based MIS has been configured for a small, hypothetical concrete reinforced rod mill. The mill is assumed to have four production lines, each of which can be used to produce one of fifteen different kinds of reinforced rod. This project also assumes that the database management system for keeping the daily production records of the mill has previously been established using dBase III and that all of the database files are stored on a Winchester disk.

The system consists of five subsystems as mentioned previously. Sections A.2 of this appendix describes the operation of the system modules. Sample reports which are generated by each of the subsystems are also presented in that section. Finally, section A.3 shows the detailed dBase III file structures of this sample system and describes how they correspond to information needed by other businesses.

A.2 System Module and Operation Menu

In this section, the operation of each module and the appearance of each of its related screens is briefly discussed. Operation of this system does not require knowledge of how the modules are coded since the online menus provide sufficient information to allow a novice user to become an effective operator. However, knowledge of the dBase III programming language and the dBase III record definitions is necessary if a user wants to create or modify any of the criteria for extracting information from the database.

A.2.1 Starting up the system

The system is started from inside the environment provided by the dBase III system. After typing the command "DO MODEL", the screen should show the main menu as Fig. A-1.

```
**************************************************
*          MANAGEMENT INFORMATION SYSTEM         *
*                SELECTION MENU                  *
*                ==============                  *
*     0>  QUIT                                   *
*     1>  PRODUCTION REQUIREMENTS PLANNING       *
*     2>  MATERIAL REQUIREMENTS PLANNING         *
*     3>  COST ANALYSIS                          *
*     4>  FINANCIAL MANAGEMENT                   *
*     5>  MANAGEMENT OPERATION RESEARCH          *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : _____
```

Fig. A-1 Main Menu

The user simply enters the selection number, and the system will call the subsystem based on the entered number.

A.2.1 Running the Production Requirements Planning (PRP) Subsystem

The user should press 1 and RETURN in response to the main menu in order to enter the PRP subsystem. The following selection menu is then shown on the screen.

```
**************************************************
*                                                *
*            PRODUCTS LIST SELECTION MENU        *
*            ============================        *
*                                                *
*        AISI NO.  DIA.          AISI NO.  DIA.  *
*        ========  ====          ========  ====  *
*  1 ==> 1010      10        9 ==> 1045      19   *
*  2 ==> 1010      12       10 ==> 1045      22   *
*  3 ==> 1010      16       11 ==> 1065      10   *
*  4 ==> 1010      19       12 ==> 1065      12   *
*  5 ==> 1010      22       13 ==> 1065      16   *
*  6 ==> 1045      10       14 ==> 1065      19   *
*  7 ==> 1045      12       15 ==> 1065      22   *
*  8 ==> 1045      16                             *
*                                                *
**************************************************

 Enter a selection (0 - 15)          : _______

 Enter the quantity required (Kg.)   : _________
```

Fig. A-2 The Products List Selection Menu

Fig. A-2 shows all the product names that the user can select from. For instance, selection number 1 in the menu corresponds to the product which is an AISI 1010 reinforcing rod that is 10 mm in diameter.

The user should enter the selection number (i.e., a number in the range of 1 to 15) of the desired product shown on the menu, and then input the quantity of that product which is required. The system will begin to access database files which correspond to fixed costs in order to calculate costs. Next, the system will access the daily production record file to calculate the daily production rate of the given product, and then divide the required quantity by the daily production rate to get the estimated delivery time. While doing the calculation, the system will display a message that it is busy in performing the calculation.

After the system completes the calculation of fixed costs and the estimated delivery time, the following Criteria Selection Menu will be displayed on the screen.

```
**************************************************
*          CRITERIA FILE SELECTION MENU          *
*          ============================          *
*                                                *
*     0>  RETURN TO MAIN MENU                    *
*     1>  USE AN EXISTING CRITERIA FILE          *
*     2>  MODIFY AN EXIST CRITERIA FILE          *
*     3>  CREATE A NEW CRITERIA FILE             *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-3 Screen Display of Criteria Selection Module

A.2.1.1 Criteria File Creation

To create a new criteria file, the user should enter 3 and RETURN in response to Criteria Selection Menu. The following question is then displayed:

ENTER THE CRITERIA FILE NAME : _____ .

Simultaneously, a HELP SCREEN will be presented on the bottom of the screen which shows the names of all existing criteria files (i.e., files with a 'CRD' file extension) contained in the file directory. The system will reject any duplicate file name which is entered. Next, the system will ask the user to input the number of records which will be included in the criteria file; each record represents information concerning a transaction which extracts data from the database. After the number of records has been entered, the following questions should be shown on the screen (i.e., Fig. A-4), and the system will wait for the user to respond. All of the user's responses to the questions on this screen will be stored as one record into a criteria file for use in data extraction. Thus, this screen will be repeated as many times as is necessary, to define each record in the criteria file.

```
+------------------------------------------------------+
|   ITEM NAME  : _____          FILE NAME  : _____     |
|                                                      |
|   FIELD NAME : _____                                 |
|                                                      |
|   ENTER THE FILTER SCOPE  : _____________________    |
|   _____________________________________________      |
|   _____________________________________________      |
|                                                      |
|   ENTER THE FUNCTION  : _____                        |
|                                                      |
|********************  HELP SCREEN  *******************|
|  CRITERIA FILES CONTAINED IN THE DIRECTORY :         |
|                                                      |
|   DEMO_FL1.CRD       DEMO_FL2.CRD       DEMO_FL3.CRD |
+------------------------------------------------------+
```

Fig. A-4 Screen Display of Criteria File Generation Queries

The first question (ITEM NAME) asks for the user to supply a cost name identifier; any character string of up to eight letters is acceptable. The FILE NAME prompt asks for a dBase III data file name to be input; the cost is calculated from data in this file. The FIELD NAME item asks for the attribute name, in the selected dBase III file, from which the cost is derived. The ENTER THE FILTER SCOPE prompt requests the user to input a filter scope; this filter scope specifies which of the data records of the selected dBase III file are to be selected. The item ENTER THE FUNCTION asks for the name or a dBase III function to be input. A dBase III function such as AVERAGE and SUM can be used to correc-tly summarize the data extracted from the selected data file.

The user must answer all the questions shown on the screen. The HELP SCREEN will give the user necessary information in an effort to guide him through the process of answering the questions. For example, at the step when the FIELD NAME item must be entered, the HELP SCREEN will show all field names which are contained in the chosen dBase III data file. Note that in answering these questions, the user must be fully familiar with the dBase III schema, since erroneous criteria will cause wrong results or system termination when executing the extraction module.

As part of the Filter Scope question and the Function question, the system will supply default values; these values are decided during the configuration of the system. Therefore, unless the user wants to reconfigure the system to suit another organization or the user is extremely familiar with the system, they should not be changed.

A.2.1.2 Criteria File Modification

In order to modify an existing criteria file, the user enters a 2 followed by a RETURN in response to the Criteria Selection Menu, and then the following question is displayed:

ENTER THE CRITERIA FILE NAME : _____ .

Meanwhile, a HELP SCREEN which shows all of the .CRD files in the file directory will also be displayed. After a file name has been entered, the first criteria record contained in the chosen criteria file is displayed on the screen (Fig. A-5 shows an example of this). The prompt cursor will lead the user, field by field, through each of the criteria records in the selected file. For each field, a HELP SCREEN will be displayed which shows information that could be an aid to the user at that point. The user has the option to change or leave each field in a criteria record as it is.

```
+-------------------------------------------------------------+
|          THE CRITERIA FILE MODIFICATION QUERIES             |
|                                                             |
|   ITEM NAME  : MATERIAL          FILE NAME  : MATERIAL      |
|                                                             |
|   FIELD NAME : UNIT_PRICE                                   |
|                                                             |
|   ENTER THE FILTER SCOPE  : DATE_TODAY >= (DATE() - 90) .AND|
|   . AISI_NBR = PROD_IDNBR .AND. UNIT_PRICE <> 0_____________|
|   ____________________________________________________      |
|                                                             |
|   ENTER THE FUNCTION  : AVERAGE_____                        |
|                                                             |
|********************  HELP SCREEN  ************************|
|   FIELDS CONTAINED IN  MATERIAL FILE :                      |
|                                                             |
|   AISI_NBR      IN_STOCK       UNIT_PRICE     CONSUMPTION   |
|   DATE_TODAY    BALANCE                                     |
+-------------------------------------------------------------+
```

Fig. A-5 An Example for
a Screen Display of Criteria File Modification

A.2.1.3 Use An Existing Criteria file

To use a previously constructed criteria file, the user simply enters the file name to be used for extracting data followed by a RETURN.

Fig. A-6 is an example of a report which was generated by the Production Requirements Planning subsystem. For this example, the product AISI 1045 with the diameter 19 millimeter was selected, and the quantity required was 30,000 Kgs.

```
DATE : 09/01/85

          THE COST ANALYSIS OF  1045 X  19

            QUANTITY REQUIRED    30.00 MT
          ==================================

              LABOR          :     18.96
              ENERGY         :      3.95
              MATERIAL       :    170.00
              UTIL COST      :      5.84
              FIXED LABOR    :     20.71

          ==================================
              TOTAL COST     :    219.45

        ESTIMATED DELIVERY TIME :  11/05/85
```

Fig. A-6 An Example of a Production Requirements Planning Report

A.2.2 Material Requirements Planning (MRP)

The user must press 2 and RETURN in response to the main menu in order to start the MRP feature. The product selection menu is then displayed. This menu is similar to the menu of Fig. A-2, but it does not query the user for the quantity required. After entering the selection number of the material name, the system will do the calculation tasks automatically and respond with an optimal reordering level of the given material on the screen.

Fig. A-7 is an example of a report which was generated by the MRP subsystem. The material name selected for this example was AISI 1010.

```
DATE : 09/01/85

        THE SUGGESTED REORDERING LEVEL OF AISI 1010
        ===========================================

            CONSUMPTION RATE :     9.80   MT/DAY
            DELIVERY TIME    :    35      DAYS
            SAFETY FACTOR    :   130 %

        ===========================================

          SUGGESTED REORDERING LEVEL :  445.90 MT
```

Fig. A-7 An Example of a Material Requirements Planning Report

A.2.3 Cost Analysis

To perform a cost analysis, the user should press 3 and RETURN in response to the main menu. Next, the user must select a product name from the product name list. The module to process the product name selection menu is the same as the one used by the Production Requirements Planning (see Fig. A-2), but it does not ask for a "quantity required" to be input by the user. After entering the product name selection number, the user should see the following questions:

```
ENTER THE STARTING DATE  : __/__/__

ENTER THE ENDING DATE    : __/__/__  .
```

The starting and ending date over which the cost of the product will be analyzed must then be entered. Next, the system will provide a Criteria Selection Menu (as in Fig. A-3). The modules which carry out this selection are identical to those used by the Production Requirements Planning, and so they are not restated.

Fig. A-8 is an example of a report which was generated by the Cost Analysis subsystem. The product selected was AISI 1045 with 19 millimeter in diameter, and the input starting date and ending date were specified as 04/01/85, /08/31/85 respectively.

DATE : 09/01/85

THE COST ANALYSIS OF 1045 X 19

START DAY	END_DATE	COST NAME	COST
04/01/85	05/01/85	ENERGY	6.64
04/01/85	05/01/85	MATERIAL	150.00
04/01/85	05/01/85	LABOR	23.17
05/01/85	05/31/85	ENERGY	6.63
05/01/85	05/31/85	MATERIAL	140.00
05/01/85	05/31/85	LABOR	21.75
05/31/85	06/30/85	ENERGY	6.63
05/31/85	06/30/85	MATERIAL	23.30
05/31/85	06/30/85	LABOR	140.00
06/30/85	07/30/85	ENERGY	6.67
06/30/85	07/30/85	MATERIAL	23.88
06/30/85	07/30/85	LABOR	139.00
07/30/85	08/29/85	ENERGY	6.59
07/30/85	08/29/85	MATERIAL	145.00
07/30/85	08/29/85	LABOR	23.12

Fig. A-8 An Example of a Cost Analysis Report

A.2.4 Financial Management

The user should press 4 followed by a RETURN in response to the main menu in order to start this feature. The system asks the following questions:

ENTER THE FISCAL YEAR : ____ .

The user must enter the fiscal year over which the financial statement is to be performed. Next, the system will calculate the sum of all account categories automatically and provide the following screen for the user to select from:

```
**************************************************
*                                                *
*            MIS FINANCIAL MANAGEMENT            *
*                OUTPUT SELECTION                *
*                ================                *
*                                                *
*                0>  QUIT                        *
*                1>  SCREEN                      *
*                2>  PRINTER                     *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ____
```

Fig. A-9 Financial Statements Output Selection Menu

The user enters the desired output destination for displaying the financial balance table and the income statement, and the results will be shown as requested. Fig. A-10 and Fig. A-11 are samples of the types of reports generated by the Financial Management subsystem.

DATE : 09/01/85

MIS FINANCIAL STATEMENTS
Balance Sheet as of 1985

ASSETS		LIABILITIES	
CASH	283742.00	ACCOUNT PAYABLE	115904.00
INVENTORY	11872.00	NOTE PAYABLE	70556.00
PRE-INSURANCE	5500.00	WAGE PAYABLE	12800.00
ACCOUNT RECEIVABLE	189980.00	INTEREST PAYABLE	427.00
EQUIPMENT	941000.00	BANK LOAN	55000.00
FIXED PROPERTIES	548000.00		
		TOTAL LIABILITIES	254687.00
		CAPITAL STOCK	1526000.00
		RETAINED EARNING	199407.00
TOTAL ASSETS	1980094.00	TOTAL EQUITY	1980094.00

Fig. A-10 An Example of Financial Statements Report

MIS INCOME STATEMENTS
Income statement for the year

TOTAL SALES :		315480.00
EXPENSES :		
Interest	427.00	
Salaries	33272.00	
Tax expense	18200.00	
Advertising	630.00	
Energy	12342.00	
Wages	67000.00	
Material	165355.00	
Transportation	4142.00	
Social expense	688.00	
Insurance	500.00	
Miscellaneous	517.00	
SUBTOTAL :	303073.00	315480.00
GROSS INCOME		12407.00

Fig. A-11 An Example of Income Statements Report

After displaying the financial balance table and the income statement, the system will ask:

DO YOU WANT TO SEE DETAILS (Y/N) ??? : _ .

If the answer is "Y" then a selection menu like Fig. A-12 will be presented on the screen, otherwise the subsystem will return to the main menu.

Figures A-13 to A-17 show the different submenus that can arise from choosing an item in the Accounts Categories Selection menu (Fig. A-12). For any of these submenus, the user can pick any item in any category, and the system will respond with the detailed records which are contained in the account ledger files.

```
**************************************************
*                                                *
*          MIS LEDGER ACCOUNTING FILE            *
*              ACCOUNT CATEGORIES                *
*              ==================                *
*                                                *
*     0>  QUIT                 3> EQUITY         *
*     1>  ASSETS               4> EXPENSES       *
*     2>  LIABILITIES          5> REVENUES       *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-12 Account Categories Selection Menu

```
**************************************************
*                                                *
*          MIS LEDGER ACCOUNTING FILE            *
*              ASSET CATEGORY                    *
*              ==============                    *
*                                                *
*               0>  QUIT                         *
*               1>  CASH                         *
*               2>  INVENTORY                    *
*               3>  PREPAID INSURANCE            *
*               4>  ACCOUNTS RECEIVABLE          *
*               5>  EQUIPMENT                    *
*               6>  FIXED PROPERTY               *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-13 Asset Items Selection Menu

```
**************************************************
*                                                *
*            MIS LEDGER ACCOUNTING FILE          *
*               LIABILITY  CATEGORIES            *
*               ======================           *
*                                                *
*     0>  QUIT                3> WAGES PAYABLE   *
*     1>  ACCOUNTS PAYABLE    4> INTEREST PAYABLE*
*     2>  NOTES PAYABLE       5> BANK LOANS      *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-14 Liability Items Selection Menu

```
**************************************************
*                                                *
*            MIS LEDGER ACCOUNTING FILE          *
*                EQUITY CATEGORIES               *
*                ==================              *
*                                                *
*                0>  QUIT                        *
*                1>  CAPITAL STOCK               *
*                2>  RETAINED EARNINGS           *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-15 Equity Items Selection Menu

```
**************************************************
*                                                *
*            MIS LEDGER ACCOUNTING FILE          *
*               EXPENSE CATEGORIES               *
*               ==================               *
*                                                *
*     0>  QUIT                6> WAGES           *
*     1>  INTEREST            7> MATERIAL        *
*     2>  SALARIES            8> TRANSPORTATION  *
*     3>  TAXES / REVENUE     9> SOCIAL EXPENSES *
*     4>  ADVERTISING        10> INSURANCE       *
*     5>  ENERGY             11> MISCELLANEOUS   *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-16 Expense Items Selection Menu

```
**************************************************
*                                                *
*            MIS LEDGER ACCOUNTING FILE          *
*                REVENUE CATEGORIES              *
*                ==================              *
*                                                *
*                  0>  QUIT                      *
*                  1>  SALES                     *
*                  2>  MISCELLANEOUS INCOME      *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-17 Revenue Items Selection Menu

A.2.5 Management Operation Research

The user should press 5 followed by a RETURN in response to the main menu in order to start this feature. The system will display a Criteria File Selection Menu which is the same as the one used with the Production Requirements Planning module (see Fig. A-3).

A.2.5.1 Linear Programming Criteria File Creation

In order to create a file containing a description of a linear programming model, the user should respond to the Criteria File Selection Menu (Fig. A-3) with a 3 and RETURN; the system then displays the following questions on the screen:

```
ENTER CRITERIA FILE NAME : ________

ENTER THE PROBLEM TYPE (MAX OR MIN)   : ___

HOW MANY CONSTRAINTS IN THIS PROBLEM : __

HOW MANY VARIABLES IN THIS PROBLEM   : __

NUMBER OF LESS_THAN CONSTRAINTS      : __

NUMBER OF EQUAL_TO CONSTRAINTS       : __

NUMBER OF GREATER_THAN CONSTRAINTS   : __ .
```

The user should answer these questions based on his model or tableau. Next, the system will iteratively prompt the user with the following message to define the type of each of the constraints or rows.

```
ENTER ROW CONSTRAINT TYPE (L/E/G)  : _ .
```

After the constraint types have been defined, the following screen is displayed (Fig. A-18):

```
**************************************************
*                                                *
*            COL  1  OBJECT COEFFICIENT          *
*           ============================         *
*                                                *
*           1>  USER DEFINED                     *
*           2>  DERIVED FROM THE DATABASE        *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : __
```

Fig. A-18 Selection Menu for Defining Coefficient

If the user's selection is 1, then the system will ask for the coefficient's value to be immediately input. However, if the selection is 2, the system will display a series of questions on the screen (see Fig. A-19), and await the answers. The way to proceed with answering these questions is similar to the manner in which the Criteria File Creation process was carried out (in section A.2.1). After the criteria have been defined, the system will derive data from databases based on the defined criteria, and the derived value will be assigned as the coefficient of the column shown on the screen.

```
+-----------------------------------------------------------+
|   FILE NAME   : ________                                  |
|                                                           |
|   FIELD NAME  : ___________                               |
|                                                           |
|   ENTER THE FILTER SCOPE  : _____________________________ |
|   _________________________________________________________|
|   _________________________________________________________|
|                                                           |
|   ENTER THE FUNCTION  : ___________                       |
|                                                           |
|**********************  HELP SCREEN  **********************|
|   FIELDS CONTAINED IN  PROD_RD FILE :                     |
|                                                           |
|   DATE_TODAY     PROD_ID         PROD_LNE        PROD_QTY |
|   LABOR_COST                                              |
+-----------------------------------------------------------+
```

Fig. A-19 Screen Display of Criteria File Generation Queries

After the Objective Function Coefficients have been defined, the the system will iteratively ask the user to define each of the constraint coefficients. The process of defining constraint coefficients is similar to the manner in which the coefficients of the Objective Function are defined. Finally, the system will save all of the defined coefficients in a criteria file with the extension ".ldb" for later use.

A.2.5.2 Linear Programming Criteria File Modification

If MODIFY AN EXISTING CRITERIA FILE has been selected from the Criteria File Selection menu, then the following screen will be displayed:

```
**************************************************
*                                                *
*        LINEAR PROGRAMMING MODIFICATION         *
*                                                *
*                SELECTION MENU                  *
*                ==============                  *
*                                                *
*    0>  QUIT                                    *
*    1>  OBJECTIVE FUNCTION COEFFICIENTS         *
*    2>  CONSTRAINTS SET COEFFICIENTS            *
*    3>  THE NATURE OF THE OPTIMIZATION          *
*    4>  THE CONSTRAINT TYPES (L,E,OR G)         *
*    5>  SAVE THE CHANGES                        *
*                                                *
**************************************************

SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ___
```

Fig. A-20 Linear Programming Constraints Modification Menu

If the user selects the item 1 in Fig. A-20, the system will display all the objective function coefficients on the HELP SCREEN. The user must then enter the column number corresponding to the objective coefficient that he wants to change; this must be followed by the user inputting the new coefficient value.

If the user selects item 2, then the system will display the largest row number., i.e., the number of side constraints that are available to the user. The user must then enter the number of the row corresponding to the constraint to be changed; this number should be smaller than or equal to the row number shown on the screen. Next, the system will display all of the coefficients contained in the selected constraint and prompt the following questions:

ENTER THE COLUMN NAME : ___

ENTER THE CHANGED VALUE : _____ .

The user simply enters the column number corresponding to the coefficient which he wants to change and also the new value of the selected coefficient. The system will update the constraints only when the user responds with 5 to the Linear Programming Modification Selection Menu after all the changes have been completed.

A.2.5.3 Use Existing Criteria File for Linear Programming

To run the linear programming package for an existing Criteria File, the user should press 1 in selecting from the Criteria File Selection Menu (Fig. A-2), the following message is then displayed.

WHICH FILE YOU WANT TO CONVERT ??? : _____ .

Meanwhile, all the dBase III files with the extension .dbf which are contained in the file directory will also be displayed on the HELP SCREEN for the user's reference. After entering a file name, the system will convert the selected file to a MPS-PC compatible working file, and then run the MPS-PC linear programming package (for detailed information on how to use MPS-PC linear programming problem solving package, please refer to the user's manual of the package [RES84].)

Fig. A-21 is a sample of the reports which are generated by the Management Operation Research subsystem.

****

LP PROBLEM FILE NAME: REPORT
PROBLEM TYPE : MAX
OPTIMAL SOLUTION REACHED IN 2 ITERATIONS

DATE: 01-12-1986
ALGORITHM START TIME = 10:03:00
ALGORITHM END TIME = 10:03:06

OBJECTIVE FUNCTION = 676.50000

****

SECTION 1 - ROWS

NBR	TYPE	ROW	AT	ACTIVITY	SLACK_ACTI	LOWER_LMT	UPPER_LMT	DUAL_ACTI
1.	L	R1	UL	123.00	.	NONE	123.00	5.50
2.	L	R2	BS	307.50	252.50	NONE	560.00	.
3.	L	R3	BS	492.00	178.00	NONE	670.00	.

****

SECTION 2 - COLUMNS

NBR	COLUMN	AT	ACTIVITY	INPUT_COST	LOWER_LMT	UPPER_LMT	REDUCED
1.	C1	BS	61.50	11.00		NONE	.
2.	C2	LL	.	12.00		NONE	-4.50
3.	C3	LL	.	13.00		NONE	-9.00

****

Fig. A-21 Sample Report Generated by MPS-PC.

A. 3 dBase III Record Descriptors

The dBase III files described here are the files that have been used in simulating a concrete reinforced rod mill. Also, a supplementary dBase file, FLE_FLD.DBF, is used for storing all of the name of each file holding a record type and its attribute names. When creating or modifying data extraction criteria, this supplementary file is used by the MIS for showing on the HELP SCREEN which files or fields can be accessed. This FLE_FLD.DBF must be updated whenever the MIS relevant dBase III file(s) have been changed.

Fig. A-22 is an Entity-Relationship diagram (E-R diagram) for the production department of the simulated reinforced rod mill. Files which have been included in the set of dBase III files, but which are not shown on the E-R diagram, belong to other departments of the mill. Such records have no relationship with the production department, but may still provide useful for the prototype MIS. For example, both the FIXED LABOR file and the UTILITY file do not belong to the production department, but they are used by the MIS when calculating the fixed costs of a product. All of the five accounting files are independent, they do not have any relationship with each others. Therefore, E-R diagram for the accounting department is not available. Some of the attributes such as AISI_NBR, PROD_LNE, and DIAMETER, have been created specifically for the simulated business. However, to suit other organization's needs, users may reconfigure the file structure by means of the dBase III utility command 'modify structure'. This command allows users to add, delete, or modify the physical structure of any of the dBase III files.

A new user who does not have any database system on the computer may restructure the dBase III records and files specified in this system to suit their own needs. Organizations which already have their own database system on the microcomputer may extend their database system to fit the information requirements of this prototype MIS. When configuring dBase III file structures in order to run this system, the following factors must be considered:

- The cost analysis feature in the Production Requirements Planning subsystem is concerned with all costs, including fixed costs and the variable costs, that are related to a certain product. There are five types of costs which are included in the simulated system: energy cost, material cost, labor cost, fixed labor cost, and utility cost. Users may add or delete any of the costs based upon their practical needs.

- The Material Requirements Planning subsystem should be used only for production businesses. The material inventory file and the order history file can be made to fit most businesses by simply changing the attribute which reflects the name of the material (Aisi_nbr is the material name used in the database of the hypothetical rod mill).

- The Financial Management subsystem calculates the sum of each of the five main accounting categories, which include: asset, liability, equity, expense, and revenue, in order give the user a financial statements of a company. The five main accounting categories are generally used by most of business firms. Therefore, the accounting file structures specified in this system should be very similar with the accounting file structures which are used by most of business firms. The "inverse category" field in this system will be used only when a user wants to construct an automatic tracking system, which has been mentioned in CHAPTER IV, on accounting transactions.

After the file structures have been altered, the user must reconfigure the prototype MIS to let the system know: where to get data, how the system will input data, and what operations will be performed on the data.

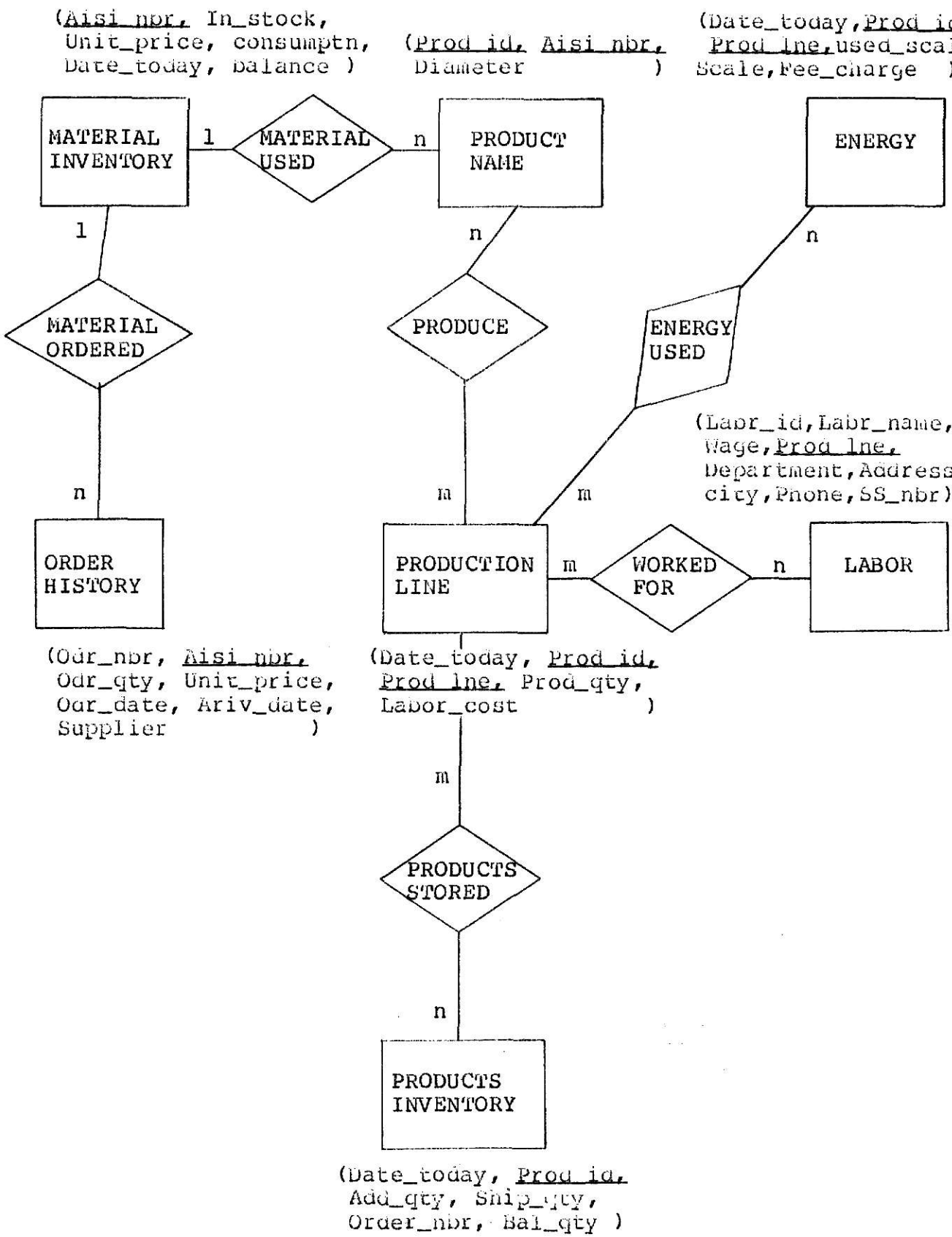

Fig. A-22 E-R Diagram For The Production Department

A.3.1 Data File Structures

The system for the hypothetical rod mill contains fourteen dBase III files. Six of them belong to the production department; they are: production record (prod_rd.dbf), material inventory (material.dbf), energy (energy.dbf), labor (labor.dbf), product name (product.dbf), product inventory (inventory.dbf). Three of the dBase III files belong to the financial department; they are: fixed labor (fixlabor.dbf), production equipment (utility.dbf), and order history (ord_hist.dbf). The files which are contained by the accounting department include: asset (act_asst.dbf), liability (act_liab.dbf), equity (act_eqit.dbf), expense (act_expn.dbf), and revenue (act_revn.dbf).

The type of information for each record type is described as follows:

- The data dictionary name gives a descriptive name of the file which holds instances of the record.
- The description item describes the use of the file which contains instances of the record.
- The dBase III file name gives the actual file name which is used by dBase III to hold record instances.
- The field column is used to order the attributes of a record.
- The field name is the actual attribute name used (all attributes of a record are listed).
- The alias name column is used to provide a more meaningful attribute name.
- The type column gives the data type of the attributes.
- The width item tells the length of an attribute.
- The dec item tells the decimal point of a numeric data.

All of the records used by the rod mill business are now described.

FILES BELONG TO THE PRODUCTION DEPARTMENT:

Data dictionary name : daily production record
Description : holds the daily production record for each production line.
dBase III file name : prod_rd.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	PROD_ID	PRODUCT_ID	Numeric	2	
3	PROD_LNE	PRODUCTION_LINE	Numeric	1	
4	PROD_QTY	QUANTITY_PRODUCE	Numeric	6	
5	LABOR_COST	TOTAL_LABOR_COST	Numeric	6	2

Data dictionary name : material inventory record
Description : holds the daily material consumption rate and the inventory balance.
dBase III file name : material.dbf

Field	Field name	Alias name	Type	Width	Dec
1	AISI_NBR	MATERIAL_NAME	Character	8	
2	IN_STOCK	REPLENISH_QTY	Numeric	7	2
3	UNIT_PRICE		Numeric	7	2
4	CONSUMPTN		Numeric	7	2
5	DATE_TODAY	CURRENT_DATE	Date	8	
6	BALANCE		Numeric	7	2

Data dictionary name : energy consumption
Description : keep track of the energy meter on each production line and convert the energy consumption to monetary value.
dBase III file name : energy.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	PROD_ID	PRODUCT_ID	Numeric	2	
3	PROD_LNE	PRODUCTION_LINE	Numeric	1	
4	SCALE	ELECTRIC_METER_SCALE	Numeric	8	
5	USED_SCALE	ELECTRICITY_USED	Numeric	8	
6	FEE_CHARGE	ELECTRICITY_COST	Numeric	6	2

Data dictionary name : labor cost
Description : holds the on line labor information and the job description.
dBase III file name : labor.dbf

Field	Field name	Alias name	Type	Width	Dec
1	LABR_ID	LABOR_ID	Numeric	2	
2	LABR_NAME	LABOR_NAME	Character	22	
3	WAGE		Numeric	4	
4	PROD_LNE	PRODUCTION_LINE	Numeric	1	
5	DEPARTMENT	DEPARTMENT_IN	Character	12	
6	ADDRESS		Character	20	
7	CITY		Character	15	
8	PHONE		Character	12	
9	SS_NBR	SOCIAL_SECURITY_NBR	Character	11	

Data dictionary name : products inventory
Description : holds the products inventory records and control the shipment
dBase III file name : inventory.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	PROD_ID	PRODUCT_ID	Numeric	2	
3	ADD_QTY	QUANTITY_PRODUCED	Numeric	6	
4	SHIP_QTY	QUANTITY_SHIPPED	Numeric	7	
5	ORDER_NBR	ORDER_NUMBER	Character	9	
6	BAL_QTY	BALANCE_QUANTITY	Numeric	8	

Data dictionary name : product name
Description : holds products name.
dBase III file name : product.dbf

Field	Field name	Alias name	Type	Width	Dec
1	PROD_ID	PRODUCT_ID	Numeric	2	
2	AISI_NBR	MATERIAL_NAME	Character	8	
3	DIAMETER	ROD_DIAMETER	Numeric	2	

FILES BELONG TO THE FINANCIAL DEPARTMENT:

Data dictionary name : fixed labor cost
Description : holds the information of the labor who works in a fixed position.
dBase III file name : fixlabor

Field	Field name	Alias name	Type	Width	Dec
1	LABR_ID	LABOR_ID	Numeric	2	
2	LABR_NAME	LABOR_NAME	Character	22	
3	WAGE		Numeric	4	
4	DEPARTMENT	DEPARTMENT_IN	Character	12	
5	ADDRESS		Character	20	
6	CITY		Character	15	
7	PHONE		Character	12	
8	SS_NBR	SOCIAL_SECURITY_NBR	Character	11	

Data dictionary name : production equipment
Description : holds the production equipment list and their depreciation status.
dBase III file name : utility.dbf

Field	Field name	Alias name	Type	Width	Dec
1	UTIL_NAME	EQUIPMENT_NAME	Character	20	
2	COST	EQUIPMENT_COST	Numeric	9	2
3	P_DATE	PURCHASING_DATE	Date	8	
4	SUPPLIER	SUPPLIER_NAME	Character	30	
5	LIFE_TIME	DEPRECIATION_TIME	Numeric	2	
6	REM_VALUE	REMAINING_VALUE	Numeric	7	

Data dictionary name : order history
Description : holds the material purchasing history records for the use of Material Requirement Planning.
dBase III file name : odr_hist.dbf

Field	Field name	Alias name	Type	Width	Dec
1	ODR_NBR	ORDER_NUMBER	Character	6	
2	AISI_NBR	MATERIAL_NAME	Character	8	
3	ODR_QTY	ORDER_QUANTITY	Numeric	7	2
4	UNIT_PRICE		Numeric	7	2
5	ODR_DATE	DATE_ORDERED	Date	8	
6	ARIV_DATE	DATE_ARRIVED	Date	8	
7	SUPPLIER	SUPPLIER_NAME	Character	25	

FILES BELONG TO THE ACCOUNTING DEPARTMENT:

Data dictionary name : assets account
Description : holds the assets ledger account
dBase III file name : act_asst.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	ITEM_CATE	ITEM_CATEGORY	Character	3	
3	INVERS_CAT	INVERSE_CATEGORY	Character	3	
4	CREDIT		Numeric	9	2
5	DEBIT		Numeric	9	2
6	JOURNAL_NO	JOURNAL_NUMBER	Numeric	7	

Data dictionary name : liability account
Description : holds the liability ledger account
dBase III file name : act_liab.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	ITEM_CATE	ITEM_CATEGORY	Character	3	
3	INVERS_CAT	INVERSE_CATEGORY	Character	3	
4	CREDIT		Numeric	9	2
5	DEBIT		Numeric	9	2
6	JOURNAL_NO	JOURNAL_NUMBER	Numeric	7	

Data dictionary name : equity account
Description : holds the equity ledger account
dBase III file name : act_eqit.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	ITEM_CATE	ITEM_CATEGORY	Character	3	
3	INVERS_CAT	INVERSE_CATEGORY	Character	3	
4	CREDIT		Numeric	9	2
5	DEBIT		Numeric	9	2
6	JOURNAL_NO	JOURNAL_NUMBER	Numeric	7	

Data dictionary name : expense account
Description : holds the expense ledger account
dBase III file name : act_expn.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	ITEM_CATE	ITEM_CATEGORY	Character	3	
3	INVERS_CAT	INVERSE_CATEGORY	Character	3	
4	CREDIT		Numeric	9	2
5	DEBIT		Numeric	9	2
6	JOURNAL_NO	JOURNAL_NUMBER	Numeric	7	

Data dictionary name : revenue account
Description : holds the revenue ledger account
dBase III file name : act_revn.dbf

Field	Field name	Alias name	Type	Width	Dec
1	DATE_TODAY	CURRENT_DATE	Date	8	
2	ITEM_CATE	ITEM_CATEGORY	Character	3	
3	INVERS_CAT	INVERSE_CATEGORY	Character	3	
4	CREDIT		Numeric	9	2
5	DEBIT		Numeric	9	2
6	JOURNAL_NO	JOURNAL_NUMBER	Numeric	7	

Note:

dBase III is a relational database in which all retrievals have to be done by means of a sequential search. Thus, a join process in dBase III takes a large amount of time. For instance, to join two files, each of which contains 1000 records will take about two minutes to complete. In the above dBase III file structures listing, some redundant attributes such as the PROD_ID and the PROD_LNE appear in several occurrences. The existence of redundant attribute(s) causes the second normal form of the relational database rules to be violated. However, in order to save time in retrieving data, a common technique is to have logically redundant data.

APPENDIX B

SYSTEM SOURCE CODE

```
**  MODEL.PRG  ************************************************
*   This module provides a main menu for the user to access any *
*   or the five subsystems through a single key strock.         *
***************************************************************
CLEAR
SET TALK OFF
SET SAFETY OFF
SET HEADING OFF
PUBLIC action
PUBLIC ch
DO WHILE .T.
SET COLOR TO 10/1,6/5,4
ch = chr(220)
@  3,14 SAY '*************************************************'
@  4,14 SAY '*                                               *'
@  5,14 SAY '*          MANAGEMENT INFORMATION SYSTEM        *'
@  6,14 SAY '*                 SELECTION MENU                *'
@  7,14 SAY '*               '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+'                    *'
@  8,14 SAY '*                                               *'
@  9,14 SAY '*     0>  QUIT                                  *'
@ 10,14 SAY '*     1>  PRODUCTION REQUIREMENTS PLANNING      *'
@ 11,14 SAY '*     2>  MATERIAL REQUIREMENTS PLANNING        *'
@ 12,14 SAY '*     3>  COST ANALYSIS                         *'
@ 13,14 SAY '*     4>  FINANCIAL MANAGEMENT                  *'
@ 14,14 SAY '*     5>  MANAGEMENT OPERATION RESEARCH         *'
@ 15,14 SAY '*                                               *'
@ 16,14 SAY '*************************************************'
action = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET action RANGE 0,5
READ
  DO CASE
    CASE action = 1
       DO prod_inq
       DO cri_sele
    CASE action = 2
       DO prod_inq
       DO matr_req
    CASE action = 3
       DO prod_inq
       DO cri_sele
    CASE action = 4
       DO fina_mgt
    CASE action = 5
       DO cri_sele
    CASE action = 0
       CLEAR
       CLEAR ALL
       SET COLOR TO 10/1,6/5,8
       SET HEADING ON
       SET TALK ON
       RETURN
  ENDCASE
ENDDO
```

```
**  PROD_INQ.PRG  ********************************************
*   This module submits a product name list selection menu for  *
*   the user to select a product ID number from, and asks the   *
*   user to input the quantity required.  Based on these two    *
*   values, dBase III will retrieve the daily production record *
*   file to calculate the daily production rate, and then       *
*   respond to the users with a estimated delivery time.        *
**********************************************************
CLEAR
PUBLIC prod_idnbr, qty_requir, prod_rate, date_ok, prod_nbr, dia
USE product
GOTO TOP
@  3,14 SAY '*************************************************'
@  4,14 SAY '*                                                *'
@  5,14 SAY '*            PRODUCTS LIST SELECTION MENU        *'
@  6,14 SAY '*          '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+'        *'
@  7,14 SAY '*                                                *'
@  8,14 SAY '*        AISI NO.   DIA.          AISI NO.    DIA.  *'
@  9,14 SAY '*        ========   ====          ========    ====  *'
@ 10,14 SAY '*   1 ==>  1010      10       9 ==>  1045       19   *'
@ 11,14 SAY '*   2 ==>  1010      12      10 ==>  1045       22   *'
@ 12,14 SAY '*   3 ==>  1010      16      11 ==>  1065       10   *'
@ 13,14 SAY '*   4 ==>  1010      19      12 ==>  1065       12   *'
@ 14,14 SAY '*   5 ==>  1010      22      13 ==>  1065       16   *'
@ 15,14 SAY '*   6 ==>  1045      10      14 ==>  1065       19   *'
@ 16,14 SAY '*   7 ==>  1045      12      15 ==>  1065       22   *'
@ 17,14 SAY '*   8 ==>  1045      16                              *'
@ 18,14 SAY '*                                                *'
@ 19,14 SAY '*************************************************'
prod_idnbr = 00
qty_requir = 0.00
@ 21,18 SAY 'Enter a selection (0 - 15)     : ' GET prod_idnbr ;
RANGE 1,15
READ
LOCATE FOR prod_id = prod_idnbr
prod_nbr = aisi_nbr
dia = STR(diameter,3)
IF action = 1 THEN
   DO time_est
   DO fix1
ENDIF
IF action = 3 THEN
   DO get_time
   CLEAR
@ 10,13 SAY '*************************************************'
@ 11,13 SAY '*                                               *'
@ 12,13 SAY '*                                               *'
@ 12,22 SAY 'PROCESSING DATA, PLEASE WAIT !!!'
@ 13,13 SAY '*                                               *'
@ 14,13 SAY '*************************************************'
   DO fix1
ENDIF
RETURN
```

```
**  TIME_EST.PRG  ***********************************************
*                                                                 *
*   This submodule is called by the PROD_INQ.PRG module; it       *
*   will ask for the quantity required to be input by the user.   *
*   Next, this submodule will calculate the estimated delivery    *
*   time based upon the given required quantity.                  *
*                                                                 *
*******************************************************************
@ 23,18 SAY 'Enter quantity required (Kg) : ' GET qty_requir
READ
CLEAR
SET COLOR TO 10/1,6/5,8
@ 10,13 SAY '*************************************************'
@ 11,13 SAY '*                                               *'
@ 12,13 SAY '*                                               *'
@ 12,22 SAY 'PROCESSING DATA, PLEASE WAIT !!!'
@ 13,13 SAY '*                                               *'
@ 14,13 SAY '*************************************************'
SET COLOR TO 14/8,6/5,8
*******************************************************************
*  The following process calculates the estimated delivery time.  *
*******************************************************************
USE prod_rd
SUM prod_qty to prod_rate FOR date_today > (DATE() - 90) .AND.;
  prod_id = prod_idnbr
prod_rate = prod_rate / 90
USE inventory
GOTO BOTTOM
DO WHILE prod_id <> prod_idnbr .AND. .NOT. BOF()
   SKIP -1
ENDDO
IF .NOT. BOF() THEN
   inv = bal_qty
   ELSE
      inv = 0
ENDIF
date_need = INT((qty_requir - inv) / prod_rate) + 2
IF date_need < 0 THEN
   date_ok = date()
   ELSE
   date_ok = date() + date_need
ENDIF
```

```
**  FIX1.PRG  ************************************************
*                                                              *
*   This module is used for calculating the fixed cost of the  *
*   given product.                                             *
*                                                              *
****************************************************************
PUBLIC utl_cost, fix_labor
USE UTILITY
GOTO TOP
ttl = 0
DO WHILE .NOT. EOF()
   ttl = ttl + (cost - rem_value) / life_time / 12
   SKIP
ENDDO
SELE 2
USE PROD_RD
SUM prod_qty TO summ FOR date_today >= CTOD ('04/01/85') .AND. ;
date_today <= CTOD('08/31/85')
utl_cost = ttl * 5 * 1000 / summ
****************************************************************
*              Calculating the FIX LABOR COST                  *
****************************************************************
SELE 1
USE fixlabor
SUM wage to fix_labor
fix_labor = (fix_labor * 5 * 1000) /summ
RETURN
```

```
**  CRI_SELE.PRG  ************************************************
*  This module provides a selection menu for the user to select *
*  either an existing criteria file, modify an existing criteria*
*  file, or create a new one.                                   *
************************************************************************
DO WHILE .T.
   CLEAR
@  6,14 SAY '*****************************************************'
@  7,14 SAY '*                                                   *'
@  8,14 SAY '*          CRITERIA FILE SELECTION MENU             *'
@  9,14 SAY '*          '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+'               *'
@ 10,14 SAY '*                                                   *'
@ 11,14 SAY '*     0>  RETURN TO MAIN MENU                       *'
@ 12,14 SAY '*     1>  USE AN EXIST CRITERIA FILE                *'
@ 13,14 SAY '*     2>  MODIFY AN EXIST CRITERIA FILE             *'
@ 14,14 SAY '*     3>  CREATE A NEW CRITERIA FILE                *'
@ 15,14 SAY '*****************************************************'
selectn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET selectn RANGE 0,3
READ
  DO CASE
    CASE selectn = 1
       IF action = 1 THEN
          DO extract
          RUN proj1
       ENDIF
       IF action = 3 THEN
          DO extract3
          RUN proj3
       ENDIF
       IF action = 5 THEN
          DO convert
          RUN proj5
          RUN mps-alg
       ENDIF
       CLEAR
    CASE selectn = 2
       IF action = 5 THEN
          DO lp_modi
          ELSE
             DO cri_modi
       ENDIF
    CASE selectn = 3
       IF action = 5 THEN
          DO lp_gen
          ELSE
             DO cri_gen
       ENDIF
    CASE selectn = 0
       CLEAR
       RETURN
  ENDCASE
ENDDO
```

```
**  CRI_GEN.PRG  ***********************************************
*                                                                *
*   This module provides a series of questions for the user to   *
*   respond in order to create a criteria file for the data      *
*   extraction use.                                              *
*                                                                *
******************************************************************
CLEAR
SET COLOR TO 10/1,6/5,8
bell = chr(7)+chr(7)+chr(7)+chr(7)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ ' ****  HELP SCREEN  **** '
@ 15,54 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch
? bell
SET COLOR TO 14/8,6/5,8
USE sample
******************************************************************
*   Generate a criteria data base file to hold all the query     *
*   information.                                                 *
******************************************************************
COPY to TEMP STRUCTURE EXTENDED
USE temp
DELETE ALL
PACK
APPEND BLANK
REPLACE field_name WITH 'item_name'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 10
APPEND BLANK
REPLACE field_name WITH 'file_name'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 8
APPEND BLANK
REPLACE field_name WITH 'field_name'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 10
APPEND BLANK
REPLACE field_name WITH 'filter'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 150
APPEND BLANK
REPLACE field_name WITH 'function'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 10
USE
STORE '         ' TO cri_file
bool = .F.
@ 16,1 SAY 'DIRECTORY OF CRITERIA FILE'
@ 17,1 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch
DO CASE
   CASE action = 1
      ext = 'crd'
   CASE action = 3
```

```
      ext = 'cra'
ENDCASE
DIR *.&ext
bool = .F.
DO WHILE .NOT. bool
   @ 6,10 SAY 'Enter the criteria file name  : ' GET cri_file
   READ
   IF (FILE('&cri_file..&ext')) .OR. (SUBSTR(cri_file,1,1) = ' ') ;
   THEN
      ? bell
      @ 24,10 SAY 'Error input or duplicate file name, ;
please re-enter'
      ELSE
         bool = .T.
   ENDIF
ENDDO
cri_name = TRIM(cri_file)
CREATE &cri_name..&ext FROM temp
STORE '  ' TO no_file
@ 8,10 SAY 'Enter the number of files related : ';
  GET no_file PICTURE '99'
READ
@ 6,10 SAY '                                                          ;
         '
@ 8,10 SAY '                                                          ;
         '
SET COLOR TO 10/1,6/5,8
no_f = VAL(no_file)
i = 1
DO WHILE i <= no_f
   j = STR(i,2)
   STORE '          ' TO filename
   STORE '            ' TO itm_name
   STORE '            ' TO fld_name
   @ 2,21SAY 'Enter the &j cost item information'
   @ 3,19 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
  +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
   @ 5,3 SAY 'Enter the name of the cost : ' GET itm_name
   READ
   bool = .F.
   @ 16,0 CLEAR
   @ 16,1 SAY 'DIRECTORY OF .DBF FILENAME'
   @ 17,1 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
     +ch+ch+ch+ch+ch+ch+ch+ch+ch
   SELECT 2
   USE file_fld
   GOTO TOP
   n = 18
   m = 1
   filedir = 'NULLFILE'
   DO WHILE .NOT. EOF()
      IF filedir <> file_name THEN
         riledir = file_name
         @ n,m get riledir
         CLEAR GET
```

```
            m = m + 10
            IF m > 70 THEN
               m = 1
               n = n + 2
            ENDIF
            ELSE
               SKIP
         ENDIF
      ENDDO
      DO WHILE .NOT. bool
         @ 5,45 SAY 'Enter the file name : ' GET filename
         READ
         IF FILE('&filename..dbf') THEN
            bool = .T.
            ELSE
               ? bell
               @ 24,20 SAY 'File does not exist !!!! '
         ENDIF
      ENDDO
      @ 16,0 CLEAR
      @ 16,1 SAY 'FIELDS CONTAINED IN THE &filename : '
      @ 17,1 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
        +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
      LOCATE FOR file_name = UPPER(filename)
      n = 19
      m = 0
      DO WHILE .NOT. EOF()
         @ n,m SAY '' GET field_name
         CLEAR GET
         m = m + 15
         IF m > 65 THEN
            n = n + 2
            m = 0
         ENDIF
         CONTINUE
      ENDDO
      store '           ' to fld_name
      DO WHILE bool
         @ 7,3 SAY 'Enter the field name          : ' GET fld_name
         READ
         LOCATE FOR FILE_NAME = UPPER(filename) .AND.;
         FIELD_NAME = UPPER(fld_name)
         IF .NOT. EOF() THEN
            bool = .F.
            ELSE
               ? bell
               @ 23,10 SAY 'Field does not exist, please refer to ;
the help screen'
         ENDIF
      ENDDO
      IF action = 1 THEN
      DO CASE
      CASE UPPER(filename) = 'MATERIAL'
      STORE '2' TO pick
      STORE 'DATE_TODAY >= (DATE() - 90) .AND. AISI_NBR = PROD_NBR ;
```

```
   .AND. UNIT_PRICE <> 0                                                ;
                                        ' to conditn
   OTHERWISE
   STORE '1' TO pick
   STORE 'DATE_TODAY >= (DATE() - 90) .AND. PROD_ID = PROD_IDNBR;
                                                                        ;
                                        ' to conditn
   ENDCASE
   ENDIF
   IF action = 3 THEN
   DO CASE
   CASE UPPER(filename) = 'MATERIAL'
   STORE 'DATE_TODAY >= START_TIME .AND. DATE_TODAY <= END_TIME ;
   .AND. AISI_NBR = PROD_NBR .AND. UNIT_PRICE <> 0                      ;
                                        ' to conditn
   STORE '2' TO pick
   OTHERWISE
   STORE 'DATE_TODAY >= START_TIME .AND. DATE_TODAY <= END_TIME ;
   .AND. PROD_ID = PROD_IDNBR                                           ;
                                        ' to conditn
   STORE '1' TO pick
   ENDCASE
   ENDIF
   @ 16,0 SAY '  Filter Scope Example : <SCOPE1> [.AND. <SCOPE2>;
[.OR. <SCOPE3>] [.....]]      '
   @ 17,0 SAY '  "SAME" stand for the scope same as the previous;
cost'
   bool = .T.
   DO WHILE bool
      @  9,3 SAY 'Enter the filter scope       : '
      @ 10,3 GET conditn
      READ
      answer = ' '
      @ 24,10 SAY 'Is the scope correct (Y/N) ?? : ' get answer
      READ
      IF UPPER(answer) = 'Y' THEN
         bool = .F.
      ENDIF
   ENDDO
   tempcond = UPPER(conditn)
   IF (i <> 1) .AND. (TRIM(tempcond) = 'SAME') THEN
      conditn = scope
   ENDIF
   scope = conditn
   @ 16,0 CLEAR
   @ 16,3 SAY 'Function available  : '
   @ 18,10 SAY '0> NO FUNCTION USED'
   @ 19,10 SAY '1> SUM '
   @ 20,10 SAY '2> AVERAGE '
   @ 22,3 SAY 'Please pick up one of the three numbers'
   bool = .T.
   DO WHILE bool
      @ 13,3 SAY 'Enter the selection number   : ' GET pick ;
             picture '9'
      READ
```

```
      STORE '        ' TO functn
      DO CASE
         CASE pick = '0'
            functn = '        '
            bool = .F.
         CASE pick = '1'
            functn = 'SUM     '
            bool = .F.
         CASE pick = '2'
            functn = 'AVERAGE'
            bool = .F.
         OTHERWISE
            ? bell
            @ 23,3 SAY 'Sorry !!! Only three numbers available'
      ENDCASE
   ENDDO
   SELECT 1
   APPEND BLANK
   REPLACE item_name WITH UPPER(itm_name)
   REPLACE file_name WITH UPPER(filename)
   REPLACE field_name WITH UPPER(fld_name)
   REPLACE filter WITH UPPER(conditn)
   REPLACE function WITH UPPER(functn)
   i = i + 1
   CLEAR GET
ENDDO
CLEAR
RETURN
```

```
**  CRI_MODI.PRG  ***********************************************
*                                                                *
* This module allows a user to modify an existing criteria file.*
*                                                                *
******************************************************************
CLEAR
SET TALK OFF
SET COLOR TO 10/1,6/5,8
bell = chr(7)+chr(7)+chr(7)+chr(7)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+'  ****  HELP SCREEN  ****  '
@ 15,54 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch
? bell
SET COLOR TO 14/8,6/5,8
filename = '         '
DO CASE
   CASE action = 1
      ext = 'cru'
   CASE action = 3
      ext = 'cra'
ENDCASE
@ 17,0
DIR *.&ext
bool = .F.
DO WHILE .NOT. bool
   @ 7,10 SAY 'WHICH FILE YOU WANT TO MODIFY ?? : ' GET filename
   READ
   IF SUBSTR(filename,1,1) <> ' ' THEN
      file_name = TRIM(filename)
      IF FILE('&file_name..&ext') THEN
         USE &file_name..&ext
         bool = .T.
         ELSE
            ? bell
            @ 24,10 SAY 'File does not exist.  Please refer to;
 the help screen'
      ENDIF
      ELSE
         CLEAR
         RETURN
   ENDIF
ENDDO
GOTO TOP
@ 7,10 SAY '                                                    '
SELECT 2
USE file_rla
DO WHILE .NOT. EOF()
   SELECT 1
   temp_item = item_name
   temp_file = file_name
   temp_field = field_name
   temp_filtr = filter
   temp_funct = function
   SELECT 2
```

```
   @ 16,0 CLEAR
   @ 16,1 SAY 'FIELDS CONTAINED IN THE &temp_file : '
   @ 17,1 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
      +ch+ch
   @ 17,20 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
   LOCATE FOR file_name = UPPER(temp_file)
   n = 19
   m = 0
   DO WHILE .NOT. EOF()
      @ n,m SAY '' GET field_name
      CLEAR GET
      m = m + 15
      IF m > 65 THEN
         n = n + 2
         m = 0
      ENDIF
      CONTINUE
   ENDDO
   SELECT 1
   @ 2,28 SAY 'Criteria Modification'
   @ 3,28 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
       +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
   @ 5,5 SAY 'Item name     :  ' GET temp_item
   @ 5,45 SAY 'File name     :  ' GET temp_file
   @ 7,5 SAY 'Field name    :  ' GET temp_field
   @ 9,5 SAY 'Filter scope :  '
   @ 10,5 GET temp_filtr
   @ 13,5 SAY 'Function     :  ' GET temp_funct
   READ
   SELECT 1
   IF UPPER(SUBSTR(temp_item,1,4)) = 'DELE' THEN
      DELETE
      ELSE
         REPLACE item_name WITH UPPER(temp_item)
         REPLACE file_name WITH UPPER(temp_file)
         REPLACE field_name WITH UPPER(temp_field)
         REPLACE filter WITH UPPER(temp_filtr)
         REPLACE function WITH UPPER(temp_funct)
   ENDIF
   SKIP
ENDDO
answer = ' '
boolean = .T.
DO WHILE boolean
   ? bell
   @ 23,9 SAY 'DO YOU WANT TO ADD MORE ITEM ??? (Y/N)' GET answer
   READ
   IF UPPER(answer) = 'Y' THEN
      @ 5,3 SAY '                                                    ;
                                '
      @ 7,3 SAY '                                                    '
      @ 9,3 SAY '                                                    '
      @ 10,3 SAY '                                                   ;
                                                                     ;
                                                          '
```

```
      @ 13,3 SAY '                                                    '
      STORE '          ' TO filename
      STORE '            ' TO itm_name
      STORE '            ' TO fld_name
      @ 5,3 SAY 'Enter the name of the cost : ' GET itm_name
      READ
      @ 16,0 CLEAR
      @ 16,1 SAY 'DIRECTORY OF .DBF FILENAME'
      @ 17,1 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
         +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
      SELECT 2
      GOTO TOP
      n = 18
      m = 1
      filedir = 'NULLFILE'
      DO WHILE .NOT. EOF()
         IF filedir <> file_name THEN
            filedir = file_name
            @ n,m get filedir
            CLEAR GET
            m = m + 10
            IF m > 70 THEN
               m = 1
               n = n + 2
            ENDIF
            ELSE
               SKIP
         ENDIF
      ENDDO
      bool = .F.
      DO WHILE .NOT. bool
         @ 5,45 SAY 'Enter the file name : ' GET filename
         READ
         IF FILE('&filename..dbf') THEN
            bool = .T.
            ELSE
               ? bell
               @ 24,10 SAY 'File does not exist. Please refer to;
the help screen'
         ENDIF
      ENDDO
      SELECT 2
      @ 16,0 CLEAR
      @ 16,1 SAY 'FIELDS CONTAINED IN THE &filename : '
      @ 17,1 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
        +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
      LOCATE FOR file_name = UPPER(filename)
      n = 19
      m = 0
      DO WHILE .NOT. EOF()
         @ n,m SAY '' GET field_name
         CLEAR GET
         m = m + 15
         IF m > 65 THEN
            n = n + 2
```

```
            m = 0
         ENDIF
         CONTINUE
      ENDDO
      store '           ' to fld_name
      DO WHILE bool
         @ 7,3 SAY 'Enter the field name          : ' GET fld_name
         READ
         LOCATE FOR FILE_NAME = UPPER(filename) .AND.;
         FIELD_NAME = UPPER(fld_name)
         IF .NOT. EOF() THEN
            bool = .F.
            ELSE
               ? bell
               @ 23,10 SAY 'Field does not exist, please refer ;
to the help screen'
         ENDIF
      ENDDO
      STORE '                                                    ;
                                                                 ;
                                        ' to conditn
      STORE ' ' TO pick
      @ 16,0 SAY '  Filter Scope Example : <SCOPE1> [.AND. ;
<SCOPE2> [.OR. <SCOPE3>] [.....]]       '
      @ 17,0 SAY '                                              '
      bool = .T.
      DO WHILE bool
         @  9,3 SAY 'Enter the filter scope       : '
         @ 10,3 GET conditn
         READ
         answer = ' '
         @ 24,10 SAY 'Is the scope correct (Y/N) ?? : ' get answer
         READ
         IF UPPER(answer) = 'Y' THEN
            bool = .F.
         ENDIF
      ENDDO
      @ 16,0 CLEAR
      @ 16,3 SAY 'Function available  : '
      @ 18,10 SAY '1> SUM'
      @ 19,10 SAY '2> AVERAGE'
      @ 20,10 SAY '0> NOT NECESSARY'
      @ 22,3 SAY 'Please pick up one of the three numbers'
      bool = .T.
      DO WHILE bool
         @ 13,3 SAY 'Enter the selection number   : ' GET pick ;
             picture '9'
         READ
         STORE '       ' TO functn
         DO CASE
            CASE pick = '0'
               functn = 'NA      '
               bool = .F.
            CASE pick = '1'
               functn = 'SUM     '
```

```
                bool = .F.
            CASE pick = '2'
                functn = 'AVERAGE'
                bool = .F.
            OTHERWISE
                ? bell
                @ 23,3 SAY 'Sorry !!! Only three numbers available'
         ENDCASE
      ENDDO
      SELECT 1
      APPEND BLANK
      REPLACE item_name WITH UPPER(itm_name)
      REPLACE file_name WITH UPPER(filename)
      REPLACE field_name WITH UPPER(fld_name)
      REPLACE filter WITH UPPER(conditn)
      REPLACE function WITH UPPER(functn)
      CLEAR GET
      ELSE
         boolean = .F.
   ENDIF
ENDDO
PACK
CLEAR
RETURN
```

```
**  EXTRACT.PRG  ************************************************
*                                                                 *
*   This module is used by the PRODUCTION REQUIREMENT PLANNING    *
*   subsystem.  It allows users to select an existing criteria    *
*   file, and then do extraction tasks based on the information   *
*   stored in the selected criteria file.                         *
*                                                                 *
*******************************************************************
CLEAR
SET SAFETY OFF
SET HEADING OFF
SET COLOR TO 10/1,6/5,8
bell = chr(7)+chr(7)+chr(7)+chr(7)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ '  ****  HELP SCREEN  ****  '
@ 15,54 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
SET COLOR TO 14/8,6/5,8
? bell
sele_file = '        '
@ 17,0
DIR *.crd
bool = .F.
DO WHILE .NOT. bool
   @ 7,10 SAY 'WHICH FILE YOU WANT TO USE ?? : ' GET sele_file
   READ
   file_used = TRIM(sele_file)
   IF FILE('&file_used..crd') THEN
      SELE 1
      USE &file_used..crd
      bool = .T.
      ELSE
         ? bell
         @ 24,10 SAY 'File does not exist.  Please refer to;
 the help screen'
   ENDIF
ENDDO
SET ALTERNATE TO ref.rle
SET ALTERNATE ON
SET CONSOLE OFF
? DATE()
LIST OFF TRIM(file_name) + '.EXT'
SET ALTERNATE OFF
CLOSE ALTERNATE
SET CONSOLE ON
CLEAR
SET COLOR TO 10/1,6/5,8
@ 10,13 SAY '*************************************************'
@ 11,13 SAY '*                                                *'
@ 12,13 SAY '*                                                *'
@ 12,22 SAY 'EXTRACTING DATA, PLEASE WAIT !!!'
@ 13,13 SAY '*                                                *'
@ 14,13 SAY '*************************************************'
SET COLOR TO 14/8,6/5,8
GOTO TOP
```

```
nbr = STR(RECNO(),3)
itemname = item_name
file2open = TRIM(file_name)
field2use = TRIM(field_name)
condition = filter
func = function
string = TRIM(prod_nbr) + ' X ' + dia
DO WHILE .NOT. EOF()
   SELECT 2
   USE &file2open
   time = time()
   @ 16,18 SAY 'Time of extracting no. &nbr : ' GET time
   CLEAR GET
   SET CONSOLE OFF
   IF SUBSTR(condition,1,1) <> ' ' THEN
      COPY TO temp FOR &condition FIELD &field2use
      ELSE
         COPY TO temp FIELD &field2use
   ENDIF
   SELE 3
   USE TEMP
*******************************
*   Begin to convert data     *
*******************************
   SET ALTERNATE TO &file2open..ext
   SET ALTERNATE ON
   DO CASE
      CASE action = 1
          ? 'PRODUCTION_REQUIREMENT'
          ? string
          ? qty_requir
          ? prod_rate
          ? date_ok
          ? utl_cost
          ? fix_labor
      CASE action = 2
          ? 'MATERIAL REQUIREMENT'
      CASE action = 3
          ? 'COST_ANALYSIS'
      Case action = 4
          ? 'FINANCIAL_MANAGEMENT'
      Case action = 5
          ? 'LINEAR_PROGRAMMING'
   ENDCASE
   ? itemname
   IF SUBSTR(func,1,1) <> ' ' THEN
      funct = TRIM(func)
      &funct &field2use TO result
      ELSE
         result = 0
   ENDIF
   LIST OFF &field2use
   ? 9999
   ? funct, result
   USE
```

```
    SET ALTERNATE OFF
    CLOSE ALTERNATE
    SELECT 1
    SKIP
    SET CONSOLE ON
    time = TIME()
    @ 18,18 SAY 'Time of finishing  no. &nbr : ' GET time
    CLEAR GET
    nbr = STR(RECNO(),3)
    itemname = item_name
    file2open = TRIM(file_name)
    field2use = TRIM(field_name)
    condition = filter
    func = function
ENDDO
CLEAR
RETURN
```

```
***  MATR_REQ.PRG  ***********************************************
*                                                                  *
*    This module is used for calculating the material consum-      *
*    ption rate of a specific material based on the previous       *
*    quarter's production records. Next, it will access the        *
*    purchasing history file to calculate the average delivery     *
*    time in order to give a best reordering level of the          *
*    material.                                                     *
*                                                                  *
********************************************************************
PUBLIC prod_rate
CLEAR
SET COLOR TO 10/1,6/5,8
@ 10,13 SAY '*************************************************'
@ 11,13 SAY '*                                                *'
@ 12,13 SAY '*                                                *'
@ 12,22 SAY 'PROCESSING DATA, PLEASE WAIT !!!'
@ 13,13 SAY '*                                                *'
@ 14,13 SAY '*************************************************'
SET COLOR TO 14/8,6/5,8
USE prod_rd
DO CASE
   CASE prod_idnbr <= 5
      SUM prod_qty to prod_rate FOR date_today > (DATE() - 180);
      .AND. prod_id <= 5
   CASE prod_idnbr <= 10 .AND. prod_idnbr >= 6
      SUM prod_qty to prod_rate FOR date_today > (DATE() - 180);
      .AND. prod_id <= 10 .AND. prod_id >= 6
   CASE prod_idnbr <= 15 .AND. prod_idnbr >= 11
      SUM prod_qty to prod_rate FOR date_today > (DATE() - 180);
      .AND. prod_id <= 15 .AND. prod_id >= 11
ENDCASE
prod_rate = prod_rate / 180 / 1000
USE odr_hist
AVERAGE (ariv_date - odr_date) TO time_need FOR ;
aisi_nbr = prod_nbr
odr_lev1 = prod_rate * time_need * 1.3
CLEAR
today = DTOC(DATE())
time_ned = STR(time_need,6) + ' DAYS'
p_rate = STR(prod_rate, 6,2) + ' MT  '
odr_level = STR(odr_lev1,6,2) + ' MT'
@ 5,1 SAY 'DATE : ' GET today
@ 7,19 SAY 'REORDERING LEVEL OF MATERIAL AISI &prod_nbr'
@ 8,17 SAY '========================================='
@ 10,20 SAY 'Consumption per day    : ' Get p_rate
@ 11,20 SAY 'Average delivery time : ' Get time_ned
@ 12,20 SAY 'Safety factor used     :     130 % '
@ 14,17 SAY '========================================='
@ 15,19 SAY 'Suggested reordering level  : ' GET odr_level
@ 23,1
WAIT
CLEAR
RETURN
```

```
**  EXTRACT3.PRG  ***********************************************
*                                                                 *
* This module is used by the COST ANALYSIS subsystem. It allows *
* users to select an existing criteria file and do extraction   *
* tasks based upon the information stored in the selected cri-  *
* teria file.                                                     *
*                                                                 *
*******************************************************************
CLEAR
SET TALK OFF
SET SAFETY OFF
SET HEADING OFF
SET COLOR TO 10/1,6/5,8
bell = chr(7)+chr(7)+chr(7)+chr(7)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ '  ****  HELP SCREEN  ****  '
@ 15,54 SAY cn+cn+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+cn;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
SET COLOR TO 14/8,6/5,8
? bell
sele_file = '        '
@ 17,0
DIR *.cra
bool = .F.
DO WHILE .NOT. bool
   @ 7,10 SAY 'WHICH FILE YOU WANT TO USE ?? : ' GET sele_file
   READ
   file_used = TRIM(sele_file)
   IF FILE('&file_used..cra') THEN
      SELE 1
      USE &file_used..cra
      bool = .T.
      ELSE
         ? bell
         @ 24,10 SAY 'File does not exist.  Please refer to;
 the help screen'
   ENDIF
ENDDO
*******************************************************************
*   The following process creates a .dbf file called RESULT.ANA *
*   which respectively holds the month-by-month cost analysis.  *
*******************************************************************
SELE 2
USE sample
COPY to TEMP STRUCTURE EXTENDED
USE temp
DELETE ALL
PACK
APPEND BLANK
REPLACE field_name WITH 'start_day'
REPLACE field_type WITH 'date'
REPLACE field_len WITH 8
APPEND BLANK
REPLACE field_name WITH 'end_day'
REPLACE field_type WITH 'date'
```

```
REPLACE field_len WITH 8
APPEND BLANK
REPLACE field_name WITH 'cost_name'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 9
APPEND BLANK
REPLACE field_name WITH 'cost'
REPLACE field_type WITH 'n'
REPLACE field_len WITH 10
REPLACE field_dec WITH 2
APPEND BLANK
REPLACE field_name WITH 'prod_spd'
REPLACE field_type WITH 'n'
REPLACE field_len WITH 7
REPLACE field_dec WITH 2
CREATE result.ana FROM temp
USE result.ana
CLEAR
SET COLOR TO 10/1,6/5,8
@  8,13 SAY '*************************************************'
@  9,13 SAY '*                                               *'
@ 10,13 SAY '*                                               *'
@ 10,22 SAY 'EXTRACTING DATA, PLEASE WAIT !!!'
@ 11,13 SAY '*                                               *'
@ 12,13 SAY '*************************************************'
SET COLOR TO 14/8,6/5,8
SELE 1
GOTO TOP
x = 1
set_count = STR(x,2)
string = TRIM(prod_nbr) + ' X ' + dia
DO WHILE (time_end - time_start) >= 29
   start_time = time_start
   end_time   = start_time + 30
   nbr = STR(RECNO(),3)
   @ 14,27 SAY 'Extracting &set_count set data'
   SELE 4
   USE prod_rd
   SUM prod_qty to prod_rate FOR date_today >= start_time .AND.;
   date_today <= end_time .AND. prod_id = prod_idnbr
   prod_rate = prod_rate / 30
   SELE 4
   USE
   SELE 1
   DO WHILE .NOT. EOF()
      itemname = item_name
      file2open = TRIM(file_name)
      field2use = TRIM(field_name)
      condition = filter
      func = function
      SELECT 3
      USE &file2open
      time = time()
      @ 16,18 SAY 'Time of extracting no. &nbr : ' GET time
      CLEAR GET
```

```
      IF SUBSTR(condition,1,1) <> ' ' .AND. SUBSTR(func,1,1) ;
      <> ' ' THEN
         func = TRIM(func)
         &func &field2use TO result for &condition
         ELSE
            ? bell
            @ 23,1 SAY 'CRITERION ERROR !!! PLEASE MODIFY ;
CRITERIA FILE.'
            WAIT
            RETURN
      ENDIF
      SELE 3
      USE
      SELE 2
      APPEND BLANK
      REPLACE start_day WITH start_time
      REPLACE end_day WITH end_time
      REPLACE cost_name WITH itemname
      REPLACE cost WITH result
      REPLACE prod_spd WITH prod_rate
      SELECT 1
      SKIP
      time = TIME()
      @ 18,18 SAY 'Time of finishing  no. &nbr : ' GET time
      CLEAR GET
      nbr = STR(RECNO(),3)
   ENDDO
   time_start = end_time
   SELE 1
   GOTO TOP
   x = x + 1
   set_count = STR(x,2)
ENDDO
*****************************************************************
*     Converting the RESULT file to a ASCII working file.       *
*****************************************************************
SET HEADING OFF
SELE 2
SET CONSOLE OFF
SET ALTERNATE TO COST_ANA.RES
SET ALTERNATE ON
? date()
? string
? util_cost
? fix_labor
LIST OFF
SET CONSOLE ON
SET ALTERNATE OFF
CLOSE ALTERNATE
CLEAR
RETURN
```

```
****  FINA_MGT.PRG  ***********************************************
*  This module shows the debt/asset balance table of a company   *
*  refering to the accounting information stored in databases.    *
*  It also allows users to view details of any of the account     *
*  ledger files.                                                  *
***************************************************************
CLEAR
PUBLIC fiscal
fisca = '1985'
bool = .T.
DO WHILE bool
   @ 10,10 SAY 'ENTER THE FISCAL YEAR : ' GET fisca
   READ
   fiscal = VAL(fisca)
   IF (fiscal > 2000) .OR. (fiscal < 1984) THEN
      ? CHR(7)
      @ 23,15 SAY 'FISCAL YEAR MUST > 1984 OR < 2000 !!!'
      ELSE
         bool = .F.
   ENDIF
ENDDO
SELE 1
USE act_asst
SELE 2
USE act_liab
SELE 3
USE act_eqit
SELE 4
USE act_expn
SELE 5
USE act_revn
CLEAR
@ 10,13 SAY '*************************************************'
@ 11,13 SAY '*                                               *'
@ 12,13 SAY '*                                               *'
@ 12,22 SAY 'PROCESSING DATA, PLEASE WAIT !!!'
@ 13,13 SAY '*                                               *'
@ 14,13 SAY '*************************************************'
SET CONSOLE OFF
cash        = 0
inventory   = 0
pre_insure  = 0
act_receiv  = 0
equipment   = 0
fix_proprt  = 0
act_payabl  = 0
note_payab  = 0
wage_payab  = 0
int_payabl  = 0
bank_loan   = 0
capital     = 0
r_earning   = 0
interest    = 0
salaries    = 0
taxes_expn  = 0
```

```
advertise  = 0
energy     = 0
wages      = 0
material   = 0
transport  = 0
social_exp = 0
insurance  = 0
miscel_exp = 0
sales      = 0
asset_ttl  = 0
liab_ttl   = 0
equity_ttl = 0
revenue_tl = 0
expense_tl = 0
gross_marg = 0
SELE 1
SUM CREDIT TO x FOR item_cate = 'A1 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'A1 ' .AND.;
 YEAR(date_today) = fiscal
cash = cash + x - y
SUM CREDIT TO x FOR item_cate = 'A2 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'A2 ' .AND.;
 YEAR(date_today) = fiscal
inventory = inventory + x - y
SUM CREDIT TO x FOR item_cate = 'A3 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'A3 ' .AND.;
 YEAR(date_today) = fiscal
pre_insure = pre_insure + x - y
SUM CREDIT TO x FOR item_cate = 'A4 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'A4 ' .AND.;
 YEAR(date_today) = fiscal
act_receiv = act_receiv + x - y
SUM CREDIT TO x FOR item_cate = 'A5 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'A5 ' .AND.;
 YEAR(date_today) = fiscal
equipment = equipment + x - y
SUM CREDIT TO x FOR item_cate = 'A6 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'A6 ' .AND.;
 YEAR(date_today) = fiscal
fix_proprt = fix_proprt + x - y
SELE 2
SUM CREDIT TO x FOR item_cate = 'L1 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'L1 ' .AND.;
 YEAR(date_today) = fiscal
act_payabl = act_payabl - x + y
SUM CREDIT TO x FOR item_cate = 'L2 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'L2 ' .AND.;
```

```
 YEAR(date_today) = fiscal
note_payab = note_payab - x + y
SUM CREDIT TO x FOR item_cate = 'L3 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'L3 ' .AND.;
 YEAR(date_today) = fiscal
wage_payab = wage_payab - x + y
SUM CREDIT TO x FOR item_cate = 'L4 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'L4 ' .AND.;
 YEAR(date_today) = fiscal
int_payabl = int_payabl - x + y
SUM CREDIT TO x FOR item_cate = 'L5 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'L5 ' .AND.;
 YEAR(date_today) = fiscal
bank_loan = bank_loan - x + y
SELE 3
SUM CREDIT TO x FOR item_cate = 'Q1 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'Q1 ' .AND.;
 YEAR(date_today) = fiscal
capital = capital - x + y
SUM CREDIT TO x FOR item_cate = 'Q2 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'Q2 ' .AND.;
 YEAR(date_today) = fiscal
r_earning = r_earning - x + y
SELE 4
SUM CREDIT TO x FOR item_cate = 'E1 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E1 ' .AND.;
 YEAR(date_today) = fiscal
interest = interest + x - y
SUM CREDIT TO x FOR item_cate = 'E2 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E2 ' .AND.;
 YEAR(date_today) = fiscal
salaries = salaries + x - y
SUM CREDIT TO x FOR item_cate = 'E3 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E3 ' .AND.;
 YEAR(date_today) = fiscal
taxes_expn = taxes_expn + x - y
SUM CREDIT TO x FOR item_cate = 'E4 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E4 ' .AND.;
 YEAR(date_today) = fiscal
advertise = advertise + x - y
SUM CREDIT TO x FOR item_cate = 'E5 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E5 ' .AND.;
 YEAR(date_today) = fiscal
energy = energy + x - y
SUM CREDIT TO x FOR item_cate = 'E6 ' .AND.;
```

```
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E6 ' .AND.;
 YEAR(date_today) = fiscal
wages = wages + x - y
SUM CREDIT TO x FOR item_cate = 'E7 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E7 ' .AND.;
 YEAR(date_today) = fiscal
material = material + x - y
SUM CREDIT TO x FOR item_cate = 'E8 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E8 ' .AND.;
 YEAR(date_today) = fiscal
transport = transport + x - y
SUM CREDIT TO x FOR item_cate = 'E9 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E9 ' .AND.;
 YEAR(date_today) = fiscal
social_expn = social_expn + x - y
SUM CREDIT TO x FOR item_cate = 'E10' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E10' .AND.;
 YEAR(date_today) = fiscal
insurance = insurance + x - y
SUM CREDIT TO x FOR item_cate = 'E11' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'E11' .AND.;
 YEAR(date_today) = fiscal
miscel_exp = miscel_exp + x - y
SELE 5
SUM CREDIT TO x FOR item_cate = 'R1 ' .AND.;
 YEAR(date_today) = fiscal
SUM DEBIT TO y FOR item_cate = 'R1 ' .AND.;
 YEAR(date_today) = fiscal
sales = sales - x + y
asset_ttl = asset_ttl + cash + inventory + pre_insure;
 + act_receiv + equipment + fix_proprt
liab_ttl = liab_ttl + act_payabl + note_payab + wage_payab;
 + int_payabl + bank_loan
revenue_tl = revenue_tl + sales
expense_tl = expense_tl + interest + salaries + taxes_expn;
 + advertise + energy + wages + material + transport;
 + social_exp + insurance + miscel_exp
gross_marg = sales - expense_tl
r_earning = r_earning + gross_marg
equity_ttl = equity_ttl + capital + r_earning + liab_ttl
fl_name = '          '
CLEAR
@ 11,10 SAY 'ENTER THE FILE NAME TO SAVE  : ' GET fl_name
READ
fl_name = TRIM(fl_name)
SAVE TO &fl_name..mgt
SET CONSOLE ON
DO PRINT_OUT
RETURN
```

```
****  PRINT_OUT.PRG  ***************************************
*                                                           *
*  This module prints out the financial statement of a company *
*  either on the screen or on a printer.                    *
*                                                           *
*************************************************************
CLEAR
SET COLOR TO 10/1,6/5,8
bb = chr(7)
bell = bb+bb
ch = chr(220)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ ' ****  HELP SCREEN  **** '
@ 15,54 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
? bell
filename = '        '
ext = 'mgt'
@ 17,0
DIR *.&ext
bool = .F.
DO WHILE .NOT. bool
   @ 7,10 SAY 'WHICH FILE YOU WANT TO USE ?? : ' GET filename
   READ
   IF SUBSTR(filename,1,1) <> ' ' THEN
      file_name = TRIM(filename)
      IF FILE('&file_name..&ext') THEN
         RESTORE FROM &file_name..&ext ADDITIVE
         bool = .T.
         ELSE
            ? bell
            @ 24,10 SAY 'File does not exist.  Please refer to;
 the help screen'
      ENDIF
      ELSE
         CLEAR
         RETURN
   ENDIF
ENDDO
CLEAR
SET COLOR TO 10/1,6/5,4
@  3,14 SAY '*************************************************'
@  4,14 SAY '*                                               *'
@  5,14 SAY '*            MIS FINANCIAL MANAGEMENT           *'
@  6,14 SAY '*                                               *'
@  7,14 SAY '*                 OUTPUT SELECTION              *'
@  8,14 SAY '*                '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch + '                  *'
@  9,14 SAY '*                                               *'
@ 10,14 SAY '*                0>  QUIT                       *'
@ 11,14 SAY '*                1>  SCREEN                     *'
@ 12,14 SAY '*                2>  PRINTER                    *'
@ 13,14 SAY '*                                               *'
@ 14,14 SAY '*                                               *'
@ 15,14 SAY '*************************************************'
```

```
selectn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET selectn RANGE 0, 2
READ
DO CASE
   CASE selectn = 0
      CLEAR
      RETURN
   CASE selectn = 1
      DO show_res
   CASE selectn = 2
      SET ALTERNATE TO FINANCIL.STM
      SET CONSOLE OFF
      SET ALTERNATE ON
      ? fisca
      ? date()
      ? cash
      ? inventory
      ? pre_insure
      ? act_receiv
      ? equipment
      ? fix_proprt
      ? act_payabl
      ? note_payab
      ? wage_payab
      ? int_payabl
      ? bank_loan
      ? capital
      ? r_earning
      ? interest
      ? salaries
      ? taxes_expn
      ? advertise
      ? energy
      ? wages
      ? material
      ? transport
      ? social_exp
      ? insurance
      ? miscel_exp
      ? sales
      ? asset_ttl
      ? liab_ttl
      ? equity_ttl
      ? revenue_tl
      ? expense_tl
      ? gross_marg
      SET ALTERNATE OFF
      CLOSE ALTERNATE
      SET CONSOLE ON
      RUN FINA_PRT
ENDCASE
CLEAR
RETURN
```

```
**** SHOW_RES.PRG ***********************************************
*                                                                *
*  This module is called by the print_out.prg submodule to show *
*  the financial statement in a table form.                      *
*                                                                *
******************************************************************
CLEAR
date = dtoc(date())
@ 1, 5 SAY 'DATE  : ' GET date
@ 2, 25 SAY 'MIS  FINANCIAL  STATEMENTS'
@ 4, 25 SAY 'Balance Sheet as of ' GET fisca
@ 6, 15 SAY 'ASSETS'
@ 6, 47 SAY 'LIABILITIES'
@ 8, 5 SAY 'CASH                  ' GET cash
@ 8,40 SAY 'ACCOUNT PAYABLE       ' GET act_payabl
@ 9, 5 SAY 'INVENTORY             ' GET inventory
@ 9,40 SAY 'NOTE PAYABLE          ' GET note_payab
@ 10,5 SAY 'PRE-INSURANCE         ' GET pre_insure
@ 10,40 SAY 'WAGE PAYABLE          ' GET wage_payab
@ 11,5 SAY 'ACCOUNT RECEIVABLE   ' GET act_receiv
@ 11,40 SAY 'INTEREST PAYABLE     ' GET int_payabl
@ 12,5 SAY 'EQUIPMENT             ' GET equipment
@ 12,40 SAY 'BANK LOAN             ' GET bank_loan
@ 13,5 SAY 'FIXED PROPERTIES     ' GET fix_proprt
@ 13,61 SAY '-------------'
@ 14,44 SAY 'TOTAL LIABILITY ' GET liab_ttl
@ 16,40 SAY 'CAPITAL STOCK         ' GET capital
@ 17,40 SAY 'RETAINED EARNING     ' GET r_earning
@ 18,26 SAY '-------------'
@ 18,61 SAY '-------------'
@ 19,5 SAY 'TOTAL ASSETS          ' GET asset_ttl
@ 19,44 SAY 'TOTAL EQUITIES  ' GET equity_ttl
@ 23,0
WAIT '              ===>  PRESS ANY KEY TO INCOME STATEMENT  <===='
CLEAR
@ 1, 5 SAY 'DATE  : ' GET date
@ 2, 25 SAY 'MIS   INCOME   STATEMENTS'
@ 4, 23 SAY 'Income statement for the year'
@ 6, 5 SAY 'TOTAL SALES :    '
@ 6, 52 GET sales
@ 8, 5 SAY 'EXPENSES    :    '
@ 9,10 SAY 'Interest           '
@ 9,37 GET interest
@ 10,10 SAY 'Salaries           '
@ 10,37 GET salaries
@ 11,10 SAY 'Tax expense        '
@ 11,37 GET taxes_expn
@ 12,10 SAY 'Advertising        '
@ 12,37 GET advertise
@ 13,10 SAY 'Energy             '
@ 13,37 GET energy
@ 14,10 SAY 'Wages              '
@ 14,37 GET wages
@ 15,10 SAY 'Material           '
@ 15,37 GET material
```

```
@ 16,10 SAY 'Transportation     '
@ 16,37 GET transport
@ 17,10 SAY 'Social expense     '
@ 17,37 GET social_expn
@ 18,10 SAY 'Insurance          '
@ 18,37 GET insurance
@ 19,10 SAY 'Miscellaneous      '
@ 19,37 GET miscel_expn
@ 20,37 SAY '--------------------------'
@ 21,10 SAY '    SUBTOTAL  :    '
@ 21,37 GET expense_tl
@ 21,52 GET revenue_tl
@ 22,5 SAY 'GROSS INCOME            '
@ 22,52 GET gross_marg
CLEAR GET
@ 23,0
WAIT '                        ===>  PRESS ANY KEY TO RETURN  <===='
CLEAR
ans = 'Y'
bool = .T.
DO WHILE bool
@ 11,20 SAY 'DO YOU WANT TO SEE DETAILS ???' GET ans
READ
ans = UPPER(ans)
DO CASE
   CASE ans = 'Y' THEN
      DO ACT_SHOW
      bool = .F.
      CLEAR
      RETURN
   CASE ans = 'N' THEN
      bool = .F.
      CLEAR
      RETURN
   OTHERWISE
      @ 14,25 SAY 'PLEASE ENTER Y OR N'
ENDDO
```

```
**  ACT_SHOW.PRG  ***********************************************
*                                                                 *
*   This module provides a account category menu for the user     *
*   to select from and then calls the corresponding submodule     *
*   based on the user's selection.                                *
*                                                                 *
*******************************************************************

DO WHILE .T.
   SET HEADING ON
   CLEAR
   SET COLOR TO 10/1,6/5,4
@  3,14 SAY '*************************************************'
@  4,14 SAY '*                                               *'
@  5,14 SAY '*            MIS LEDGER ACCOUNTING FILE         *'
@  6,14 SAY '*                                               *'
@  7,14 SAY '*                ACCOUNT CATEGORIES             *'
@  8,14 SAY '*                '+ch+ch+ch+ch+cn+ch+cn+ch+cn+ch+ch;
   +ch+cn+ch+cn+cn+ch+ch+'                *'
@  9,14 SAY '*                                               *'
@ 10,14 SAY '*     0>  QUIT                    3> EQUITY      *'
@ 11,14 SAY '*     1>  ASSETS                  4> EXPENSES    *'
@ 12,14 SAY '*     2>  LIABILITIES             5> REVENUES    *'
@ 13,14 SAY '*                                               *'
@ 14,14 SAY '*************************************************'
actn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET actn RANGE 0, 5
READ
  DO CASE
    CASE actn = 1
       DO show_asst
    CASE actn = 2
       DO show_liab
    CASE actn = 3
       DO show_eqit
    CASE actn = 4
       DO show_expn
    CASE actn = 5
       DO show_revn
    CASE actn = 0
       CLEAR
       SET HEADING OFF
       SET COLOR TO 10/1,6/5,8
       RETURN
  ENDCASE
ENDDO
```

```
**  SHOW_ASST.PRG  ***********************************************
*                                                                  *
*  This module shows details of the selected asset ledger file.  *
*                                                                  *
********************************************************************
SELECT 1
DO WHILE .T.
   CLEAR
   SET COLOR TO 10/1,6/5,4
@  3,14 SAY '*****************************************************'
@  4,14 SAY '*                                                   *'
@  5,14 SAY '*             MIS LEDGER ACCOUNTING FILE            *'
@  6,14 SAY '*                                                   *'
@  7,14 SAY '*                  ASSET CATEGORY                   *'
@  8,14 SAY '*                '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+'                 *'
@  9,14 SAY '*                                                   *'
@ 10,14 SAY '*                   0>  QUIT                        *'
@ 11,14 SAY '*                   1>  CASH                        *'
@ 12,14 SAY '*                   2>  INVENTORY                   *'
@ 13,14 SAY '*                   3>  PREPAID INSURANCE           *'
@ 14,14 SAY '*                   4>  ACCOUNTS RECEIVABLE         *'
@ 15,14 SAY '*                   5>  EQUIPMENT                   *'
@ 16,14 SAY '*                   6>  FIX PROPERTY                *'
@ 17,14 SAY '*                                                   *'
@ 18,14 SAY '*****************************************************'
   selectn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
   GET selectn RANGE 0, 6
   READ
   DO CASE
      CASE selectn = 0
         CLEAR
         RETURN
      CASE selectn = 1
         CLEAR
         LIST FOR item_cate = 'A1 ' .AND. YEAR(date_today) = ;
riscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 2
         CLEAR
         LIST FOR item_cate = 'A2 ' .AND. YEAR(date_today) = ;
riscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 3
         CLEAR
         LIST FOR item_cate = 'A3 ' .AND. YEAR(date_today) = ;
riscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 4
         CLEAR
         LIST FOR item_cate = 'A4 ' .AND. YEAR(date_today) = ;
riscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 5
```

```
         CLEAR
         LIST FOR item_cate = 'A5 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>    ENTER ANY KEY TO RETURN    <==='
      CASE selectn = 6
         CLEAR
         LIST FOR item_cate = 'A6 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>    ENTER ANY KEY TO RETURN    <==='
   ENDCASE
ENDDO
CLEAR
RETURN
```

```
**  SHOW_LIAB.PRG  ***********************************************
*  This module shows details of the selected liability ledger   *
*  file.                                                        *
***************************************************************
SELECT 2
DO WHILE .T.
   CLEAR
@  3,14 SAY '*****************************************************'
@  4,14 SAY '*            MIS LEDGER ACCOUNTING FILE             *'
@  5,14 SAY '*                                                   *'
@  6,14 SAY '*               LIABILITY  CATEGORIES               *'
@  7,14 SAY '*                '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+'                *'
@  8,14 SAY '*                                                   *'
@  9,14 SAY '*     0>  QUIT                 3> WAGES PAYABLE     *'
@ 10,14 SAY '*     1>  ACCOUNTS PAYABLE     4> INTEREST PAYABLE  *'
@ 11,14 SAY '*     2>  NOTES PAYABLE        5> BAML LOANS        *'
@ 12,14 SAY '*****************************************************'
selectn = 0
@ 16,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
   GET selectn RANGE 0, 5
   READ
   DO CASE
      CASE selectn = 0
         CLEAR
         RETURN
      CASE selectn = 1
         CLEAR
         LIST FOR item_cate = 'L1 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 2
         CLEAR
         LIST FOR item_cate = 'L2 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 3
         CLEAR
         LIST FOR item_cate = 'L3 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 4
         CLEAR
         LIST FOR item_cate = 'L4 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 5
         CLEAR
         LIST FOR item_cate = 'L5 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
   ENDCASE
ENDDO
CLEAR
RETURN
```

```
**   SHOW_EQIT.PRG  ***********************************************
*                                                                  *
* This module shows details of the selected equity ledger file.    *
*                                                                  *
********************************************************************
SELECT 3
DO WHILE .T.
   CLEAR
   SET COLOR TO 10/1,6/5,4
@  3,14 SAY '*****************************************************'
@  4,14 SAY '*                                                   *'
@  5,14 SAY '*             MIS LEDGER ACCOUNTING FILE            *'
@  6,14 SAY '*                                                   *'
@  7,14 SAY '*                  EQUITY CATEGORIES                *'
@  8,14 SAY '*                 '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+'                 *'
@  9,14 SAY '*                                                   *'
@ 10,14 SAY '*                  0>  QUIT                         *'
@ 11,14 SAY '*                  1>  CAPITAL STOCK                *'
@ 12,14 SAY '*                  2>  RETAINED EARNINGS            *'
@ 13,14 SAY '*                                                   *'
@ 14,14 SAY '*                                                   *'
@ 15,14 SAY '*****************************************************'
selectn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET selectn RANGE 0, 2
READ
   DO CASE
      CASE selectn = 0
         CLEAR
         RETURN
      CASE selectn = 1
         CLEAR
         LIST FOR item_cate = 'Q1 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 2
         CLEAR
         LIST FOR item_cate = 'Q2 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
   ENDCASE
ENDDO
CLEAR
RETURN
```

```
**  SHOW_EXPN.PRG  ***********************************************
*                                                                 *
* This module shows details of the selected expense ledger file.*
*                                                                 *
*******************************************************************
SELECT 4
DO WHILE .T.
   CLEAR
   SET COLOR TO 10/1,6/5,4
@  3,14 SAY '*****************************************************'
@  4,14 SAY '*                                                   *'
@  5,14 SAY '*             MIS LEDGER ACCOUNTING FILE            *'
@  6,14 SAY '*                                                   *'
@  7,14 SAY '*                 EXPENSE CATEGORIES                *'
@  8,14 SAY '*                '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+'                    *'
@  9,14 SAY '*                                                   *'
@ 10,14 SAY '*      0>  QUIT                6> WAGES             *'
@ 11,14 SAY '*      1>  INTEREST            7> MATERIAL          *'
@ 12,14 SAY '*      2>  SALARIES            8> TRANSPORTATION    *'
@ 13,14 SAY '*      3>  TAXES / REVENUE     9> SOCIAL EXPENSES   *'
@ 14,14 SAY '*      4>  ADVERTISING        10> INSURANCE         *'
@ 15,14 SAY '*      5>  ENERGY             11> MISCELLANEOUS     *'
@ 16,14 SAY '*                                                   *'
@ 16,14 SAY '*****************************************************'
selectn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
   GET selectn RANGE 0,11
   READ
   DO CASE
      CASE selectn = 0
         CLEAR
         RETURN
      CASE selectn = 1
         CLEAR
         LIST FOR item_cate = 'E1 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 2
         CLEAR
         LIST FOR item_cate = 'E2 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 3
         CLEAR
         LIST FOR item_cate = 'E3 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 4
         CLEAR
         LIST FOR item_cate = 'E4 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '              ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 5
         CLEAR
```

```
         LIST FOR item_cate = 'E5 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 6
         CLEAR
         LIST FOR item_cate = 'E6 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 7
         CLEAR
         LIST FOR item_cate = 'E7 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 8
         CLEAR
         LIST FOR item_cate = 'E8 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 9
         CLEAR
         LIST FOR item_cate = 'E9 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 10
         CLEAR
         LIST FOR item_cate = 'E10' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 11
         CLEAR
         LIST FOR item_cate = 'E11' .AND. YEAR(date_today) = ;
fiscal
         WAIT '                ===>   ENTER ANY KEY TO RETURN   <==='
   ENDCASE
ENDDO
CLEAR
RETURN
```

```
**   SHOW_REVN.PRG  ***********************************************
*                                                                  *
* This module shows details of the selected revenue ledger file.*
*                                                                  *
********************************************************************
SELECT 5
DO WHILE .T.
   CLEAR
   SET COLOR TO 10/1,6/5,4
@  3,14 SAY '*****************************************************'
@  4,14 SAY '*                                                   *'
@  5,14 SAY '*             HIS LEDGER ACCOUNTING FILE            *'
@  6,14 SAY '*                                                   *'
@  7,14 SAY '*                 REVENUE CATEGORIES                *'
@  8,14 SAY '*                '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+'                  *'
@  9,14 SAY '*                                                   *'
@ 10,14 SAY '*                  0>  QUIT                         *'
@ 11,14 SAY '*                                                   *'
@ 12,14 SAY '*                  1>  SALES                        *'
@ 13,14 SAY '*                                                   *'
@ 14,14 SAY '*                  2>  MISCELLANEOUS INCOME         *'
@ 15,14 SAY '*                                                   *'
@ 16,14 SAY '*****************************************************'
selectn = 0
@ 20,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET selectn RANGE 0, 2
READ
   DO CASE
      CASE selectn = 0
         CLEAR
         RETURN
      CASE selectn = 1
         CLEAR
         LIST FOR item_cate = 'R1 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '             ===>   ENTER ANY KEY TO RETURN   <==='
      CASE selectn = 2
         CLEAR
         LIST FOR item_cate = 'R2 ' .AND. YEAR(date_today) = ;
fiscal
         WAIT '             ===>   ENTER ANY KEY TO RETURN   <==='
   ENDCASE
ENDDO
CLEAR
RETURN
```

```
*** LP_GEN.PRG ***********************************************
*                                                             *
*  This module defines the objective function and constraint  *
*  formulations, which are interprated by a business problem, *
*  for use of the linear programming package.                 *
*                                                             *
***************************************************************
CLEAR
PUBLIC coefficnt,mm,nn
SET PROCEDURE TO derive
SET COLOR TO 10/1,6/5,8
bell = chr(7) + chr(7)
@ 2,15 SAY '***********  LINEAR EQUATION GENERATOR  ***********'
@ 3,15 SAY '=================================================='
file_name = '         '
prob_type = '    '
row_nbr = 0
col_nbr = 0
ls_than_no = 0
equal2_no = 0
gt_than_no = 0
bool = .T.
DO WHILE bool
   @ 5,15 SAY 'ENTER CRITERIA FILE NAME : ' GET file_name
   @ 7,15 SAY 'ENTER THE PROBLEM TYPE (MAX OR MIN)  : ' ;
      GET prob_type
   @ 9,15 SAY 'HOW MANY CONSTRAINTS IN THIS PROBLEM : ' ;
      GET row_nbr RANGE 1,75
   @ 11,15 SAY 'HOW MANY VARIABLES IN THIS PROBLEM  : ' ;
      GET col_nbr RANGE 1,50
   @ 13,15 SAY 'NUMBER OF LESS_THAN CONSTRAINTS     : ' ;
      GET ls_than_no RANGE 0,75
   @ 15,15 SAY 'NUMBER OF EQUAL_TO CONSTRAINTS      : ' ;
      GET equal2_no RANGE 0,75
   @ 17,15 SAY 'NUMBER OF GREATER_THAN CONSTRAINTS  : ' ;
      GET gt_than_no RANGE 0,75
   READ
   IF (UPPER(prob_type) = 'MAX') .OR. (UPPER(prob_type);
   = 'MIN') THEN
      bool = .F.
      ELSE
         ? bell
         @ 23,10 SAY 'THE PROBLEM TYPE MUST BE EITHER MAX OR MIN'
   ENDIF
   IF (ls_than_no + equal2_no + gt_than_no) <> row_nbr THEN
     bool = .T.
     @ 23,10 SAY 'THE TOTAL NUMBER OF CONSTRAINT TYPES NOT EQUALS;
 TO ROW NUMBER'
   ENDIF
ENDDO
prob_type = UPPER(prob_type)
CLEAR
bool1 = .T.
DO WHILE bool1
  row = 1
```

```
  l_nbr = 0
  e_nbr = 0
  g_nbr = 0
  DO WHILE row <= row_nbr
    IF row < 10 THEN
      nn = STR(row,1)
      ELSE
        nn = STR(row,2)
    ENDIF
    bool = .T.
    DO WHILE bool
      ACCEPT '    ENTER ROW&nn CONSTRAINT TYPE (L/E/G) : ' ;
      TO r_type
      r&nn._type = UPPER(r_type)
      DO CASE
        CASE r&nn._type = "L"
           l_nbr = l_nbr + 1
           bool = .F.
        CASE r&nn._type = "E"
           e_nbr = e_nbr + 1
           bool = .F.
        CASE r&nn._type = "G"
           g_nbr = g_nbr + 1
           bool = .F.
        OTHERWISE
           ?
           ? '  THE CONSTRAINT TYPE MUST BE "L" OR "E" OR "G" '
           ?
      ENDCASE
    ENDDO
    row = row + 1
  ENDDO
  IF (l_nbr = ls_than_no) .AND. (e_nbr = equal2_no) .AND. ;
(g_nbr = gt_tnan_no) THEN
    bool1 = .F.
  ELSE
    ?
    ? '        THE NUMBER OF CONSTRAINT TYPES NOT EQUALS TO       '
    ? '        DEFINED NUMBER, PLEASE RE-ENTER                    '
    ?
    ? '   LESS THAN = ' + STR(ls_than_no,2) + ', EQUAL TO = ' + ;
STR(equal2_no,2) + ', GREATER THAN = ' + STR(gt_than_no,2)
    ?
  ENDIF
ENDDO
fl_name = TRIM(file_name)
SAVE TO &fl_name
*****************************************************************
*   Generate a criteria data base file to hold all coefficients *
*   information.                                                *
*****************************************************************
USE SAMPLE
COPY to temp STRUCTURE EXTENDED
USE temp
DELETE ALL
```

```
PACK
APPEND BLANK
REPLACE field_name WITH 'col_name'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 3
APPEND BLANK
REPLACE field_name WITH 'row_name'
REPLACE field_type WITH 'c'
REPLACE field_len WITH 3
APPEND BLANK
REPLACE field_name WITH 'coeffi'
REPLACE field_type WITH 'n'
REPLACE field_len WITH 10
REPLACE field_dec WITH 2
USE
CREATE &fl_name..ldb FROM TEMP
USE &fl_name..ldb
col = 1
DO WHILE col <= col_nbr
   CLEAR
   IF col < 10 THEN
      mm = STR(col,1)
      ELSE
         mm = STR(col,2)
   ENDIF
   CLEAR
   SET COLOR TO 10/1,6/5,4
@  6,14 SAY '*************************************************'
@  7,14 SAY '*                                               *'
@  8,14 SAY '*              COL &mm OBJECT COEFFICIENT       ;
  *'
@  9,14 SAY '*             '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+'          *'
@ 10,14 SAY '*                                               *'
@ 11,14 SAY '*             1>  USER DEFINED                  *'
@ 12,14 SAY '*             2>  DERIVED FROM DATABASE         *'
@ 13,14 SAY '*                                               *'
@ 14,14 SAY '*************************************************'
pickea = 0
@ 18,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN : ' ;
GET picked RANGE 1,2
  READ
  DO CASE
    CASE picked = 1
      INPUT 'ENTER COLUMN &mm OBJECT FUNCTION ;
COEFFICIENT : ' TO obj
    CASE picked = 2
      DO proceed
      obj = coefficnt
  ENDCASE
  APPEND BLANK
  REPLACE col_name WITH "C&mm"
  REPLACE row_name WITH "R0"
  REPLACE coeffi WITH obj
  ?
```

```
  ?
  row = 1
  DO WHILE row <= row_nbr
    IF row < 10 THEN
       nn = STR(row,1)
       ELSE
          nn = STR(row,2)
    ENDIF
    CLEAR
    SET COLOR TO 10/1,6/5,4
@  6,14 SAY '*************************************************'
@  7,14 SAY '*                                               *'
@  8,14 SAY '*              COL &mm ROW &nn COEFFICIENT      ;
    *'
@  9,14 SAY '*             '+ch+ch+ch+ch+ch+cn+cn+cn+cn+cn+ch+ch;
   +ch+cn+cn+ch+ch+ch+cn+ch+ch+cn+ch+ch+cn+ch+cn+cn+'          *'
@ 10,14 SAY '*                                               *'
@ 11,14 SAY '*             1>  USER DEFINED                  *'
@ 12,14 SAY '*             2>  DERIVED FROM DATABASE         *'
@ 13,14 SAY '*                                               *'
@ 14,14 SAY '*************************************************'
picked = 0
@ 18,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ' ;
GET picked RANGE 1,2
    READ
    DO CASE
      CASE picked = 1
        INPUT '   ENTER COLUMN &mm ROW &nn ;
COEFFICIENT : ' TO coefficnt
      CASE picked = 2
        DO proceed
    ENDCASE
    APPEND BLANK
    REPLACE col_name WITH "C&mm"
    REPLACE row_name WITH "R&nn"
    REPLACE coeffi WITH coefficnt
    ?
    row = row + 1
  ENDDO
  col = col + 1
ENDDO
row = 1
DO WHILE row <= row_nbr
  IF row < 10 THEN
    nn = STR(row,1)
    ELSE
      nn = STR(row,2)
  ENDIF
  CLEAR
  SET COLOR TO 10/1,6/5,4
@  6,14 SAY '*************************************************'
@  7,14 SAY '*                                               *'
@  8,14 SAY '*               RHS  ROW &nn COEFFICIENT        ;
    *'
@  9,14 SAY '*             '+ch+ch+ch+ch+ch+cn+ch+ch+ch+ch+ch+ch;
```

```
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+'              *'
@ 10,14 SAY '*                                              *'
@ 11,14 SAY '*            1>  USER DEFINED                  *'
@ 12,14 SAY '*            2>  DERIVED FROM DATABASE         *'
@ 13,14 SAY '*                                              *'
@ 14,14 SAY '**********************************************'
picked = 0
@ 16,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ' ;
GET picked RANGE 1,2
  READ
  DO CASE
    CASE picked = 1
      INPUT '   ENTER COLUMN &mm ROW &nn ;
COEFFICIENT : ' TO coefficnt
    CASE picked = 2
      DO proceed
  ENDCASE
  APPEND BLANK
  REPLACE col_name WITH "RHS"
  REPLACE row_name WITH "R&nn"
  REPLACE coeffi WITH coefficnt
  row = row + 1
ENDDO
CLEAR
RETURN
```

```
*** LP_MODI.PRG ***************************************************
*  This module allows users to modify the selected business      *
*  problem definitions.                                          *
*****************************************************************
CLEAR
SET COLOR TO 10/1,6/5,8
bell = chr(7)+chr(7)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ ' **** HELP SCREEN **** '
@ 15,54 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch
? bell
SET COLOR TO 14/8,6/5,8
filename = '        '
ext = 'ldb'
@ 17,0
DIR *.&ext
bool = .F.
DO WHILE .NOT. bool
   @ 7,10 SAY 'WHICH FILE YOU WANT TO MODIFY ?? : ' GET filename
   READ
   IF SUBSTR(filename,1,1) <> ' ' THEN
      file_name = TRIM(filename)
      IF FILE('&file_name..&ext') THEN
         USE &file_name..&ext
         bool = .T.
         ELSE
            ? bell
            @ 24,10 SAY 'File does not exist.  Please refer to;
 the help screen'
      ENDIF
      ELSE
         CLEAR
         RETURN
   ENDIF
ENDDO
RESTORE FROM &file_name ADDITIVE
DO WHILE .T.
  CLEAR
@  3,14 SAY '*************************************************'
@  4,14 SAY '*                                               *'
@  5,14 SAY '*        LINEAR PROGRAMMING MODIFICATION        *'
@  6,14 SAY '*                                               *'
@  7,14 SAY '*                SELECTION MENU                 *'
@  8,14 SAY '*               '+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+'                    *'
@  9,14 SAY '*                                               *'
@ 10,14 SAY '*     0>  QUIT                                  *'
@ 11,14 SAY '*     1>  OBJECTIVE FUNCTION COEFFICIENTS       *'
@ 12,14 SAY '*     2>  CONSTRAINTS SET COEFFICIENTS          *'
@ 13,14 SAY '*     3>  THE NATURE OF THE OPTIMIZATION        *'
@ 14,14 SAY '*     4>  THE CONSTRAINT TYPES (L,E,OR G)       *'
@ 15,14 SAY '*     5>  SAVE THE CHANGES                      *'
@ 16,14 SAY '*                                               *'
@ 17,14 SAY '*************************************************'
```

```
acton = 0
@ 19,14 SAY 'SELECT ANY NUMBER ON THE MENU AND PRESS RETURN :  ';
GET acton RANGE 0,5
READ
  DO CASE
    CASE acton = 1
      CLEAR
      GOTO 1
      @ 13,1 SAY 'OBJECTIVE COEFFICIENTS CONTAINED IN &fl_name :'
      m = 15
      n = 1
      DO WHIL .NOT. EOF()
        IF row_name = 'R0 ' THEN
          @ m,n GET col_name
          CLEAR GET
          n = n + 5
          @ m,n GET coeffi
          CLEAR GET
          n = n + 12
          IF n > 55 THEN
            n = 1
            m = m + 1
          ENDIF
        ENDIF
        SKIP
      ENDDO
      c_name = '    '
      @ 5,10 SAY 'ENTER THE COLUMN NAME    : ' GET c_name
      READ
      c_name = UPPER(c_name)
      LOCATE FOR row_name = 'R0 ' .and. col_name = c_name
      @ 7,10 SAY 'ENTER THE CHANGED VALUE : ' GET coeffi
      READ
    CASE acton = 2
      CLEAR
      GOTO 1
      r_name = 'R&nn'
      @ 5,10 SAY 'ENTER THE ROW NAME       : ' GET r_name
      READ
      r_name = UPPER(r_name)
      @ 13,1 SAY 'COEFFICIENTS CONTAINED IN &r_name :'
      m = 15
      n = 1
      DO WHIL .NOT. EOF()
        IF row_name = r_name THEN
          @ m,n GET col_name
          CLEAR GET
          n = n + 5
          @ m,n GET coeffi
          CLEAR GET
          n = n + 12
          IF n > 55 THEN
            n = 1
            m = m + 1
          ENDIF
```

```
    ENDIF
    SKIP
  ENDDO
  c_name = '    '
  @ 5,10 SAY 'ENTER THE COLUMN NAME    : ' GET c_name
  READ
  c_name = UPPER(c_name)
  LOCATE FOR row_name = r_name .and. col_name = c_name
  @ 7,10 SAY 'ENTER THE CHANGED VALUE : ' GET coeffi
  READ
CASE acton = 3
  IF prob_type = 'MIN' THEN
    prob_type = 'MAX'
    ELSE
      prob_type = 'MIN'
  ENDIF
  @ 22,20 SAY 'problem type has been changed to &prob_type'
  WAIT
CASE acton = 4
  CLEAR
  m = 15
  n = 1
  cnt = 1
  DO WHILE cnt <= row_nbr
    IF cnt < 10 THEN
      nn = STR(cnt,1)
      ELSE
        nn = STR(cnt,2)
    ENDIF
    @ m,n SAY 'R&nn._type'
    n = n + 10
    @ m,n GET R&nn._type
    CLEAR GET
    n = n + 5
    IF n > 65 THEN
      n = 1
      m = m + 1
    ENDIF
    cnt = cnt + 1
  ENDDO
  r_name = 'R1 '
  @ 5,10 SAY 'ENTER THE ROW NAME       : ' GET r_name
  READ
  tt = VAL(SUBSTR(r_name,2,2))
  IF tt < 10 THEN
    rr = STR(tt,1)
    ELSE
      rr = STR(tt,2)
  ENDIF
  @ 7,10 SAY 'ENTER NEW CONSTRAINT TYPE : ' get R&rr._type
  READ
  R&rr._type = UPPER(R&rr._type)
  cnt = 1
  ls_than_no = 0
  equal2_no = 0
```

```
      gt_than_no = 0
      DO WHILE cnt <= row_nbr
        IF cnt < 10 THEN
          nn = STR(cnt,1)
          ELSE
            nn = STR(cnt,2)
        ENDIF
        IF R&nn._type = 'L' THEN
          ls_than_no = ls_than_no + 1
        ENDIF
        IF R&nn._type = 'E' THEN
          equal2_no = equal2_no + 1
        ENDIF
        IF R&nn._type = 'G' THEN
          gt_than_no = gt_than_no + 1
        ENDIF
        cnt = cnt + 1
      ENDDO
    CASE acton = 5
       SAVE TO &file_name
    CASE acton = 0
       CLEAR
       SET COLOR TO 10/1,6/5,8
       RETURN
  ENDCASE
ENDDO
```

```
*** CONVERT.PRG ************************************************
*                                                               *
*  This module converts all needed dbase information and memory *
*  variables to a text working file for use of the linear pro-  *
*  gramming package.                                            *
*                                                               *
*****************************************************************
CLEAR
SET COLOR TO 10/1,6/5,8
bell = chr(7)+chr(7)+chr(7)+chr(7)
@ 15,0 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ '  ****  HELP SCREEN  ****  '
@ 15,54 SAY ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch+ch;
   +ch+ch+ch+ch+ch+ch+ch+ch+ch+ch
? bell
SET COLOR TO 14/8,6/5,8
filename = '          '
ext = 'ldb'
@ 17,0
DIR *.&ext
bool = .F.
DO WHILE .NOT. bool
   @ 7,10 SAY 'WHICH FILE YOU WANT TO CONVERT ?? : ' GET filename
   READ
   IF SUBSTR(filename,1,1) <> ' ' THEN
      file_name = TRIM(filename)
      IF FILE('&file_name..&ext') THEN
         USE &file_name..&ext
         bool = .T.
         ELSE
            ? bell
            @ 24,10 SAY 'File does not exist.  Please refer to;
 the help screen'
      ENDIF
      ELSE
         CLEAR
         RETURN
   ENDIF
ENDDO
RESTORE FROM &file_name ADDITIVE
IF row_nbr < 10 THEN
   r_nbr = STR(row_nbr,1)
   ELSE
     r_nbr = STR(row_nbr,2)
ENDIF
IF col_nbr < 10 THEN
   c_nbr = STR(col_nbr,1)
   ELSE
     c_nbr = STR(col_nbr,2)
ENDIF
IF ls_than_no < 10 THEN
   ls_nbr = STR(ls_than_no,1)
   ELSE
     ls_nbr = STR(ls_than_no,2)
ENDIF
```

```
IF equal2_no < 10 THEN
   eq_nbr = STR(equal2_no,1)
   ELSE
     eq_nbr = STR(equal2_no,2)
ENDIF
IF gt_than_no < 10 THEN
   gt_nbr = STR(gt_than_no,1)
   ELSE
     gt_nbr = STR(gt_than_no,2)
ENDIF
SET TALK OFF
SET CONSOLE OFF
SET ALTERNATE TO &file_name..lpt
SET ALTERNATE ON
? r_nbr + ',' + c_nbr + ',' + ls_nbr + ',' + eq_nbr + ',';
 + gt_nbr +',' + '"' + prob_type + '"'
cnt = 1
DO WHILE cnt <= row_nbr
  IF cnt < 10 THEN
    cc = STR(cnt,1)
    ELSE
      cc = STR(cnt,2)
  ENDIF
  ? '"' + r&cc._type +'"'
  cnt = cnt + 1
ENDDO
cnt = 1
DO WHILE cnt <= row_nbr
  IF cnt < 10 THEN
    cc = STR(cnt,1)
    ELSE
      cc = STR(cnt,2)
  ENDIF
  ? '"R&cc"'
  cnt = cnt + 1
ENDDO
cnt = 1
DO WHILE cnt <= col_nbr
  IF cnt < 10 THEN
    cc = STR(cnt,1)
    ELSE
      cc = STR(cnt,2)
  ENDIF
  ? '"C&cc"'
  cnt = cnt + 1
ENDDO
GOTO 1
LOCATE FOR row_name = 'R0'
DO WHILE .NOT. EOF()
  DO CASE
    CASE coefri < 10
      coef_char = STR(coefri,1)
    CASE coefri < 100
      coef_char = STR(coefri,2)
    CASE coefri < 1000
```

```
      coef_char = STR(coefri,3)
    CASE coefri < 10000
      coef_char = STR(coefri,4)
    CASE coefri < 100000
      coef_char = STR(coefri,5)
    CASE coefri < 1000000
      coef_char = STR(coefri,6)
    CASE coefri < 10000000
      coef_char = STR(coefri,7)
  ENDCASE
  ? coef_char
  CONTINUE
ENDDO
m = 1
DO WHILE m <= col_nbr
  IF m < 10 THEN
    mm = STR(m,1)
    ELSE
      mm = STR(m,2)
  ENDIF
  n = 1
  DO WHILE n <= row_nbr
    IF n < 10 THEN
      nn = STR(n,1)
      ELSE
        nn = STR(n,2)
    ENDIF
    LOCATE FOR (col_name = 'C&mm') .AND. (row_name = 'R&nn')
    DO CASE
      CASE coefri < 10
        coef_char = STR(coefri,1)
      CASE coefri < 100
        coef_char = STR(coefri,2)
      CASE coefri < 1000
        coef_char = STR(coefri,3)
      CASE coefri < 10000
        coef_char = STR(coefri,4)
      CASE coefri < 100000
        coef_char = STR(coefri,5)
      CASE coefri < 1000000
        coef_char = STR(coefri,6)
      CASE coefri < 10000000
        coef_char = STR(coefri,7)
    ENDCASE
    ? coef_char
    n = n + 1
  ENDDO
  m = m + 1
ENDDO
LOCATE FOR col_name = 'RHS'
DO WHILE .NOT. EOF()
  DO CASE
    CASE coefri < 10
      coef_char = STR(coefri,1)
    CASE coefri < 100
```

```
      coef_char = STR(coeffi,2)
    CASE coeffi < 1000
      coef_char = STR(coeffi,3)
    CASE coeffi < 10000
      coef_char = STR(coeffi,4)
    CASE coeffi < 100000
      coef_char = STR(coeffi,5)
    CASE coeffi < 1000000
      coef_char = STR(coeffi,6)
    CASE coeffi < 10000000
      coef_char = STR(coeffi,7)
  ENDCASE
  ? coef_char
  CONTINUE
ENDDO
? '"END"'
SET ALTERNATE OFF
CLOSE ALTERNATE
SET CONSOLE ON
RETURN
```

```
PROGRAM PROJ1 (INPUT, OUTPUT);
{*************************************************************}
{* This program is called by the Production Requirements Plan- *}
{* ning subsystem. It calculates costs based on the data which *}
{* have been extracted by dBase III,  and then respond a cost  *}
{* analysis list and an estimated delivery time to the user.   *}
{*************************************************************}
TYPE
   PROD_TYPE    = STRING [10];
   COST_TYPE    = STRING [12];
   FUNC_TYPE    = STRING [8];
   FEATURETYPE = STRING [20];
   COST_REC = RECORD
                 COST_ID : COST_TYPE;
                 FUNCTN   : FUNC_TYPE;
                 VALUES   : REAL;
              END;
   COST_ARRAY = ARRAY [1..20] OF COST_REC;
VAR
   CRT_FILE, IN_FILE, OUT_FILE : TEXT;
   DATE_TODAY, DATE_OK         : STRING [9];
   ANSWER, ACTION, NULLCHAR    : CHAR;
   FEATURE     : FEATURETYPE;
   COST_LIST   : COST_ARRAY;
   PROD_NAME   : PROD_TYPE;
   COST_NAME   : COST_TYPE;
   FUNC        : FUNC_TYPE;
   FILE_NAME   : STRING [13];
   SCRETCH     : STRING [255];
   BOOL, OK    : BOOLEAN;
   I, J        : INTEGER;
   QUANTITY, PROD_RATE, VALUE, TOTAL_COST  : REAL;
   UTIL_COST, FIX_LABOR, FUNC_VALUE        : REAL;

PROCEDURE SCRN (VAR ANS : CHAR);
{*************************************************************}
{*                                                           *}
{*   This procedure provides a selection screen for selecting *}
{*   report destination (screen, or printer.)                *}
{*                                                           *}
{*************************************************************}

BEGIN
   GOTOXY (1,7);
   WRITE (' ':10,'*******************************************');
   WRITELN ('***************');
   WRITE (' ':10,'*                                         ');
   WRITELN ('               *');
   WRITE (' ':10,'*                    OUTPUT DESTINATION   ');
   WRITELN ('               *');
   WRITE (' ':10,'*                   ');
   WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
   WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
   WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
   WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
```

```
    WRITELN ('                    *');
    WRITE (' ':10,'*                                        ');
    WRITELN ('                *');
    WRITE (' ':10,'*              S>  OUTPUT TO THE SCREEN  ');
    WRITELN ('                *');
    WRITE (' ':10,'*              P>  OUTPUT TO THE PRINTER ');
    WRITELN ('                *');
    WRITE (' ':10,'*              Q>  QUIT                  ');
    WRITELN ('                *');
    WRITE (' ':10,'*                                        ');
    WRITELN ('                *');
    WRITE (' ':10,'****************************************');
    WRITELN ('****************');
    WRITELN;
    WRITE (' ':20, 'PLEASE SELECT AN ACTION (S/P/Q) : ');
    READLN (ANS);
END;

BEGIN
   CLRSCR; ASSIGN (CRI_FILE, 'REF.FLE'); {$I-}
   RESET (CRI_FILE); {$I+}
   OK := (IORESULT = 0);
   IF NOT OK THEN
      WRITELN ('CANNOT FIND REFERENCE FILE')
   ELSE
   BEGIN
      READLN (CRI_FILE);
      READLN (CRI_FILE, DATE_TODAY);
      I := 1;
      WHILE NOT EOF(CRI_FILE) DO
      BEGIN
         READ (CRI_FILE, NULLCHAR);
         READLN (CRI_FILE, FILE_NAME);
         ASSIGN (IN_FILE, FILE_NAME);
         {$I-}
         RESET (IN_FILE);
         {$I+}
         OK := (IORESULT = 0);
         IF NOT OK THEN
            WRITELN ('CANNOT FIND REFERENCE FILE', FILE_NAME)
         ELSE
         BEGIN
            READLN (IN_FILE);
            READLN (IN_FILE, FEATURE);
            READLN (IN_FILE, PROD_NAME);
            READLN (IN_FILE, QUANTITY);
            QUANTITY := QUANTITY / 1000;
            READLN (IN_FILE, PROD_RATE);
            READLN (IN_FILE, DATE_OK);
            READLN (IN_FILE, UTIL_COST);
            READLN (IN_FILE, FIX_LABOR);
            READLN (IN_FILE, COST_NAME);
            READLN (IN_FILE);
            READLN (IN_FILE, VALUE);
            WHILE VALUE <> 9999 DO
```

```
            READLN (IN_FILE, VALUE);
         READ (IN_FILE, FUNC);
         READLN (IN_FILE, FUNC_VALUE);
         IF COST_NAME = 'ENERGY     ' THEN
            FUNC_VALUE := FUNC_VALUE * 1000 / (PROD_RATE * 90);
         IF COST_NAME = 'LABOR      ' THEN
            FUNC_VALUE := FUNC_VALUE * 1000 / (PROD_RATE * 90);
         WITH COST_LIST [I] DO
         BEGIN
            COST_ID := COST_NAME;
            FUNCTN := FUNC;
            VALUES    := FUNC_VALUE;
         END;
         I := I + 1;
      END;
   END;
   ASSIGN (OUTFILE, 'PROD_REQ.OUT');  REWRITE (OUTFILE);
   WRITELN (OUTFILE, ' DATE : ', DATE_TODAY);
   WRITE (OUTFILE, ' ':18,'THE COST ANALYSIS OF', ' ':2);
   WRITELN (OUTFILE,PROD_NAME);
   WRITELN (OUTFILE);
   WRITE (OUTFILE,' ':20,'QUANTITY REQUIRED');
   WRITELN (OUTFILE, QUANTITY :9:2, ' MT');
   WRITELN(OUTFILE,' ':18,'==================================');
   WRITELN (OUTFILE);
   TOTAL_COST := UTIL_COST + FIX_LABOR;
   FOR J := 1 TO I - 1 DO
   BEGIN
      WITH COST_LIST[J] DO
      BEGIN
         WRITE (OUTFILE, ' ':20,COST_ID :12, ' : ');
         WRITELN (OUTFILE, VALUES :10:2);
         TOTAL_COST := TOTAL_COST + VALUES;
      END;
   END;
   WRITE (OUTFILE, ' ':20, '  UTIL COST ':12, ' : ');
   WRITELN (OUTFILE, UTIL_COST :10:2);
   WRITE (OUTFILE, ' ':20, '  FIX LABOR ':12, ' : ');
   WRITELN (OUTFILE, FIX_LABOR :10:2);
   WRITELN (OUTFILE);
   WRITELN(OUTFILE,' ':18,'==================================');
   WRITELN (OUTFILE,' ':22,'TOTAL COST : ', TOTAL_COST : 10:2);
   WRITELN (OUTFILE);
   WRITE (OUTFILE, ' ':17, 'ESTIMATED DELIVERY TIME : ');
   WRITELN (OUTFILE, DATE_OK:9);
   CLRSCR;
   BOOL   := FALSE;
   ANSWER := ' ';
   WHILE NOT BOOL DO
   BEGIN
      CLRSCR;
      SCRN (ANSWER);
      CASE UPCASE(ANSWER) OF
        'S' : BEGIN
                 CLRSCR;
```

```
                    RESET (OUTFILE);
                    WHILE NOT EOF(OUTFILE) DO
                    BEGIN
                       READLN (OUTFILE, SCRETCH);
                       WRITELN (SCRETCH);
                    END;
                    WRITELN;  WRITELN;
                    WRITE (' ':18, 'DO YOU WANT ANOTHER ACTION ');
                    WRITE ('(Y/N) : ');
                    READLN (ACTION);
                    IF UPCASE(ACTION) = 'N' THEN
                    BEGIN
                       BOOL := TRUE;
                       CLRSCR;
                    END;
                 END;
           'P' : BEGIN
                    CLRSCR;
                    RESET (OUTFILE);
                    WHILE NOT EOF(OUTFILE) DO
                    BEGIN
                       READLN (OUTFILE, SCRETCH);
                       WRITELN (LST, SCRETCH);
                    END;
                    WRITELN; WRITELN;
                    WRITE (' ':15, 'DO YOU WANT ANOTHER ACTION ');
                    WRITE ('(Y/N) : ');
                    READLN (ACTION);
                    IF UPCASE(ACTION) = 'N' THEN
                    BEGIN
                       BOOL := TRUE;
                       CLRSCR;
                    END;
                 END;
           'Q' : BEGIN
                    CLRSCR;
                    CLOSE (OUTFILE);
                    BOOL := TRUE;
                 END;
            ELSE
            BEGIN
               WRITELN; WRITE(CHR(7));
               WRITELN(' ':20, 'YOU HAVE TO CHOOSE ONE OF S/P/Q');
               WRITELN(' ':20, 'PRESS ANY KEY TO RETURN TO MENU');
               REPEAT UNTIL KEYPRESSED;
            END;
         END;   {*  END OF CASE  *}
      END;      {*  END OF WHILE  *}
   END;
END.
```

```
PROGRAM PROJ3 (INPUT, OUTPUT);
{*************************************************************}
{*                                                           *}
{*  This program reads the data which have been extracted by *}
{*  dBase III, and then calculates costs month by month. The *}
{*  result can be shown either in a text report form, or in  *}
{*  a bar chart.                                             *}
{*                                                           *}
{*************************************************************}
TYPE
   DATE_TYPE    = STRING [9];
   COST_TYPE    = STRING [9];
   PROD_TYPE    = STRING [11];
   COST_REC =  RECORD
                  START_DATE : DATE_TYPE;
                  END_DATE   : DATE_TYPE;
                  COST_NAME  : COST_TYPE;
                  COST       : REAL;
                  PROD_RATE  : REAL;
               END;
   COST_ARRAY = ARRAY [1..50] OF COST_REC;

VAR
   NDX         : INTEGER;
   COST_LIST   : COST_ARRAY;
   DATE_TODAY  : DATE_TYPE;
   PROD_NAME   : PROD_TYPE;
   UTIL_COST   : REAL;
   FIX_LABOR   : REAL;
   OUT_FILE    : TEXT;
   IN_FILE     : TEXT;
   NULL_CHAR   : CHAR;
   SCRETCH     : STRING [255];
   ANSWER      : CHAR;
   ACTION      : CHAR;
   OK          : BOOLEAN;
   BOOL        : BOOLEAN;
   STR1, STR2  : COST_TYPE;

PROCEDURE SCRN (VAR ANS : CHAR);
{*************************************************************}
{*                                                           *}
{*  This procedure provides a selection menu to output the   *}
{*  results.                                                 *}
{*                                                           *}
{*************************************************************}

BEGIN
   CLRSCR;
   GOTOXY (1,7);
   WRITE (' ':10,'*****************************************');
   WRITELN ('****************');
   WRITE (' ':10,'*                                       ');
   WRITELN ('                *');
   WRITE (' ':10,'*                    OUTPUT DESTINATION    ');
```

```
    WRITELN ('                *');
    WRITE (' ':10,'*                  ');
    WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
    WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
    WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
    WRITE (CHR(220), CHR(220), CHR(220), CHR(220), CHR(220));
    WRITELN ('                  *');
    WRITE (' ':10,'*                                        ');
    WRITELN ('                *');
    WRITE (' ':10,'*             S>  OUTPUT THE REPORT TO THE');
    WRITELN (' SCREEN         *');
    WRITE (' ':10,'*             P>  OUTPUT THE REPORT TO THE');
    WRITELN (' PRINTER        *');
    WRITE (' ':10,'*             G>  SHOW THE RESULT IN BAR C');
    WRITELN ('HART            *');
    WRITE (' ':10,'*             Q>  QUIT                   ');
    WRITELN ('                *');
    WRITE (' ':10,'                                         ');
    WRITELN ('                *');
    WRITE (' ':10,'*****************************************');
    WRITELN ('****************');
    WRITELN;
    WRITE (' ':20, 'PLEASE SELECT AN ACTION (S/P/G/Q) : ');
    READLN (ANS);
END;

PROCEDURE READ_DATA (VAR COST_LST : COST_ARRAY);
{************************************************************}
{*                                                          *}
{*  This procedure reads the extracted data and put them into  *}
{*  an array of record.                                     *}
{*                                                          *}
{************************************************************}

BEGIN
   ASSIGN (IN_FILE, 'COST_ANA.RES');
   {$I-}
   RESET (IN_FILE);
   {$I+}
   OK := (IORESULT = 0);
   IF NOT OK THEN
      WRITELN ('CANNOT FIND REFERENCE FILE')
   ELSE
   BEGIN
      READLN (IN_FILE);
      READLN (IN_FILE, DATE_TODAY);
      READLN (IN_FILE, PROD_NAME);
      READLN (IN_FILE, UTIL_COST);
      READLN (IN_FILE, FIX_LABOR);
      NDX := 1;
      STR1 := 'LABOR      ';
      STR2 := 'ENERGY     ';
      WHILE NOT EOF(IN_FILE) DO
      BEGIN
         READ (IN_FILE, NULL_CHAR);
```

```
          WITH COST_LIST [NDX] DO
          BEGIN
             READ (IN_FILE, START_DATE, END_DATE, COST_NAME);
             READLN (IN_FILE, COST, PROD_RATE);
             IF (COST_NAME = STR1) OR (COST_NAME = STR2) THEN
                COST := COST * 1000 / PROD_RATE / 30;
          END;
          NDX := NDX + 1;
       END;
       NDX := NDX - 1;
    END;
END;

PROCEDURE OUT_SCRN (COST_ARY : COST_ARRAY);
{*****************************************************************}
{*   This procedure prints out the result on the SCREEN.         *}
{*****************************************************************}

VAR
   I : INTEGER;

BEGIN
  CLRSCR;
  WRITELN;
  WRITELN ('DATE : ',DATE_TODAY);
  WRITELN (' ':20, 'COST ANALYSIS LIST OF : ', PROD_NAME);
  WRITELN;
  WRITELN (' ':15,'START DAY   END DATE    COST NAME      COST  ');
  WRITELN (' ':15,'==========  ========  ============  ========');
  WRITELN;
  FOR I := 1 TO NDX DO
  BEGIN
    WITH COST_ARY[I] DO
    BEGIN
      WRITE (' ':15, START_DATE,'   ', END_DATE, '   ', COST_NAME);
      WRITELN ('   ', COST :8:2);
    End;
  END;
END;

PROCEDURE GRAPH (COST_ARY : COST_ARRAY);
{*****************************************************************}
{*                                                               *}
{*   This procedure allows user to show the result in a bar      *}
{*   chart.                                                      *}
{*                                                               *}
{*****************************************************************}

VAR
  I, J, K, L, X : INTEGER;
  M, N, Q, P : INTEGER;    {* Index                          *}
  HIGH, LOW  : INTEGER;       {* Highest and Lowest Value    *}
  HEIGHT     : REAL;          {* Bar height                  *}
  COUNT      : INTEGER;       {* Index                       *}
  HIGHNBR    : INTEGER;       {* Bar height integer value    *}
```

```
  GAP          : INTEGER;        {* Gap value on Y axis           *}
  YBAR         : INTEGER;        {* Actual height of the bar      *}
  YPOS         : INTEGER;        {* Starting position on Y axis   *}
  COST_INT     : INTEGER;        {* Convert cost to integer value *}
  TEMP_DATE    : DATE_TYPE;
  TEMP_NAME    : COST_TYPE;
  TEMP_ARY     : ARRAY [1..12] OF REAL;
  START_NO     : INTEGER;
  RESULT       : INTEGER;
  MON_CHAR     : STRING [2];
  J_TEMP       : INTEGER;

BEGIN
  OUT_SCRN (COST_ARY);
  WRITELN;
  WRITE (' ':18);
  WRITELN (CHR(7),CHR(7),'===>   PRESS ANY KEY TO GRAPH   <===');
  REPEAT UNTIL KEYPRESSED;
  TEMP_DATE := COST_ARY[1].START_DATE;
  I := 1;
  WHILE (COST_ARY[I].START_DATE = TEMP_DATE) DO
  BEGIN
    HIGH := 0;
    LOW  := MAXINT;
    MON_CHAR := COPY(TEMP_DATE,1,2);
    VAL(MON_CHAR,START_NO,RESULT);
    TEMP_NAME := COST_ARY[I].COST_NAME;
    J := 1;
    FOR K := I TO NDX DO
    BEGIN
      IF COST_ARY[K].COST_NAME = TEMP_NAME THEN
      BEGIN
        WITH COST_ARY[K] DO
        BEGIN
          TEMP_ARY[J] := COST;
          COST_INT := TRUNC(TEMP_ARY[J]);
          IF COST_INT > HIGH THEN
            HIGH := COST_INT;
          IF COST_INT < LOW THEN
            LOW  := COST_INT;
        END;     {* END OF WHITH  *}
        J := J + 1;
      END;  {*  END OF IF  *}
    END;  {* END OF FOR LOOP  *}
    J := J - 1;
  {*****************************************************************}
  {*  The following statements calculate the cooradinate value     *}
  {*  on the Y axis'                                               *}
  {*****************************************************************}
    COUNT := 1;
    HIGHNBR := HIGH ;
    WHILE HIGHNBR > 10 DO
    BEGIN
      HIGHNBR := TRUNC ( HIGHNBR / 10 );
      COUNT := COUNT * 10;
```

```
END;
HIGHNBR := HIGHNBR + 1;
HIGH := HIGHNBR * COUNT ;
GAP  := TRUNC (21 / HIGHNBR);
{*  Set color mode and draw X and Y axis                          *}
X := 1;
REPEAT
  J_TEMP := 0;
  IF J > 6 THEN           {*  Only allows 6 bars in a screen  *}
  BEGIN
    J_TEMP := J - 6;
    J := 6;
  END;
  CLRSCR;
  GRAPHCOLORMODE;
  PALETTE (0);
  DRAW (5*8, 1*8, 5*8, 22*8, 3);
  DRAW (5*8, 22*8, 38*8, 22*8, 3);
  {*************************************************************}
  {*  Write coordinate value on Y axis.                        *}
  {*************************************************************}
  Q := 0;
  YPOS := 22;
  REPEAT
    GOTOXY (1, YPOS+1);
    WRITE (Q * COUNT : 4);
    Q := Q + 1;
    IF Q <= HIGHNBR THEN
      YPOS := YPOS - GAP;
  UNTIL Q > HIGHNBR;
  YBAR := 22 - YPOS;
  GOTOXY (12,1);
  WRITE ('COST NAME :: ',TEMP_NAME);
  GOTOXY (12,2);
  WRITE ('X = MONTH, Y = $$$');
  {*************************************************************}
  {*  Draw bars on the chart.                                  *}
  {*************************************************************}
  L := 1;
  M := 0;
  N := 9;
  P := 1;
  WHILE P <= J DO
  BEGIN
    HEIGHT := YBAR * (TEMP_ARY[X] / HIGH);
    IF TEMP_ARY[X] <> 0 THEN
      IF HEIGHT < 1/4 THEN
        HEIGHT := 1/4;
    FOR K := 1 TO 16 DO
      DRAW (8*8+K+M, 22*8, 8*8+K+M, (22*8 - TRUNC (HEIGHT*8)), L)
    GOTOXY (N, 24);
    WRITE (START_NO:2);;
    M := M + 40;
    N := N + 5;
    L := L + 1;
```

```
        IF L > 3 THEN
          L := 1;
        P := P + 1;
        X := X + 1;
        START_NO := START_NO + 1;
        IF START_NO > 12 THEN
           START_NO := 1;
      End;
      REPEAT UNTIL KEYPRESSED;
      J := J_TEMP;
    UNTIL J = 0;
    I := I + 1;
  END;   {*  OF WHILE LOOP  *}
  TEXTMODE;
  GOTOXY (15,10);
END;

PROCEDURE OUT_PRT (COST_ARY : COST_ARRAY);
{*****************************************************************}
{*  This procedure prints out the result on the PRINTER.         *}
{*****************************************************************}

VAR
   I : INTEGER;

BEGIN
  WRITELN(LST);
  WRITELN(LST,'DATE : ', DATE_TODAY);
  WRITELN(LST,' ':20, 'COST ANALYSIS LIST OF : ', PROD_NAME);
  WRITELN(LST);
  WRITELN(LST,' ':15,'START DAY   END DATE    COST NAME      COST');
  WRITELN(LST,' ':15,'=========  ========  ===========  ========');
  WRITELN(LST);
  FOR I := 1 TO NDX DO
  BEGIN
    WITH COST_ARY[I] DO
    BEGIN
      WRITE (LST,' ':15, START_DATE,'   ', END_DATE, '   ');
      WRITELN (LST, COST_NAME, '   ', COST :8:2);
    END;
  END;
END;

{*******************************}
{*       MAIN PROGRAM          *}
{*******************************}
BEGIN
   READ_DATA (COST_LIST);
   BOOL   := FALSE;
   ANSWER := ' ';
   WHILE NOT BOOL DO
   BEGIN
      CLRSCR;
      SCRN (ANSWER);
      CASE UPCASE(ANSWER) OF
```

```
            'S' : BEGIN
                     CLRSCR;
                     OUT_SCRN (COST_LIST);
                     WRITELN; WRITELN;
                     WRITE (' ':18, 'DO YOU WANT ANOTHER ACTION ');
                     WRITE ('(Y/N) : ');
                     READLN (ACTION);
                     IF UPCASE(ACTION) = 'N' THEN
                     BEGIN
                        BOOL := TRUE;
                        CLRSCR;
                     END;
                  END;
            'P' : BEGIN
                     CLRSCR;
                     OUT_PRT (COST_LIST);
                     WRITELN; WRITELN;
                     WRITE (' ':18, 'DO YOU WANT ANOTHER ACTION ');
                     WRITE ('(Y/N) : ');
                     READLN (ACTION);
                     IF UPCASE(ACTION) = 'N' THEN
                     BEGIN
                        BOOL := TRUE;
                        CLRSCR;
                     END;
                  END;
            'G' : BEGIN
                     CLRSCR;
                     GRAPH (COST_LIST);
                     CLRSCR;
                     WRITELN; WRITELN;
                     WRITE (' ':18, 'DO YOU WANT ANOTHER ACTION ');
                     WRITE ('(Y/N) : ');
                     READLN (ACTION);
                     IF UPCASE(ACTION) = 'N' THEN
                     BEGIN
                        BOOL := TRUE;
                        CLRSCR;
                     END;
                  END;
              'Q' : BEGIN
                       CLRSCR;
                       BOOL := TRUE;
                    END;
               ELSE
               BEGIN
                 WRITELN; WRITE(CHR(7));
                 WRITELN(' ':20,'YOU HAVE TO CHOOSE ONE OF S/P/G/Q');
                 WRITELN(' ':20, 'PRESS ANY KEY TO RETURN TO MENU');
                 REPEAT UNTIL KEYPRESSED;
               END;
            END;   {*  END OF CASE  *}
         END;      {*  END OF WHILE  *}
END.
```

```
PROGRAM FINANCIAL_STMT (INPUT, OUTPUT);
{*************************************************************}
{* This program will read the financial data, which have been  *}
{* extracted and converted to a text working file by dBase III,*}
{* and then make a asset/debt balance sheet to print out.      *}
{*************************************************************}

VAR
   FILE_IN     : TEXT;
   NULL_LINE   : STRING[132];
   FISCA       : STRING[6];
   DATE_TODAY  : STRING [11];
   CASH, INVENTORY, PRE_INSURE, ACT_RECEIV        : REAL;
   EQUIPMENT, FIX_PROPRT, ACT_PAYABL, NOTE_PAYAB  : REAL;
   WAGE_PAYAB, INT_PAYABL, BANK_LOAN, CAPITAL     : REAL;
   R_EARNING, INTEREST, SALARIES, TAXES_EXPN      : REAL;
   ADVERTISE, ENERGY, WAGES, MATERIAL, TRANSPORT  : REAL;
   SOCIAL_EXP, INSURANCE, MISCEL_EXP, SALES       : REAL;
   ASSET_TTL, LIAB_TTL, EQUITY_TTL, REVENUE_TL    : REAL;
   EXPENSE_TL, GROSS_MARG                         : REAL;

BEGIN
   ASSIGN (FILE_IN, 'FINANCIL.STM');  RESET (FILE_IN);
   READLN (FILE_IN, NULL_LINE  );   READLN (FILE_IN, FISCA);
   READLN (FILE_IN, DATE_TODAY);    READLN (FILE_IN, CASH);
   READLN (FILE_IN, INVENTORY);     READLN (FILE_IN, PRE_INSURE);
   READLN (FILE_IN, ACT_RECEIV);    READLN (FILE_IN, EQUIPMENT);
   READLN (FILE_IN, FIX_PROPRT);    READLN (FILE_IN, ACT_PAYABL);
   READLN (FILE_IN, NOTE_PAYAB);    READLN (FILE_IN, WAGE_PAYAB);
   READLN (FILE_IN, INT_PAYABL);    READLN (FILE_IN, BANK_LOAN);
   READLN (FILE_IN, CAPITAL);       READLN (FILE_IN, R_EARNING);
   READLN (FILE_IN, INTEREST);      READLN (FILE_IN, SALARIES);
   READLN (FILE_IN, TAXES_EXPN);    READLN (FILE_IN, ADVERTISE);
   READLN (FILE_IN, ENERGY);        READLN (FILE_IN, WAGES );
   READLN (FILE_IN, MATERIAL);      READLN (FILE_IN, TRANSPORT);
   READLN (FILE_IN, SOCIAL_EXP);    READLN (FILE_IN, INSURANCE);
   READLN (FILE_IN, MISCEL_EXP);    READLN (FILE_IN, SALES );
   READLN (FILE_IN, ASSET_TTL);     READLN (FILE_IN, LIAB_TTL);
   READLN (FILE_IN, EQUITY_TTL );   READLN (FILE_IN, REVENUE_TL );
   READLN (FILE_IN, EXPENSE_TL );   READLN (FILE_IN, GROSS_MARG );
   WRITELN (LST); WRITELN (LST); WRITELN (LST); WRITELN (LST);
   WRITELN (LST,' ':5, 'DATE  : ', DATE_TODAY:10);
   WRITELN (LST,' ':25,'MIS  FINANCIAL  STATEMENTS');
   WRITELN (LST);
   WRITELN (LST,' ':25, 'Balance Sheet as of ', FISCA:6);
   WRITELN (LST);
   WRITE (LST,' ':15, 'ASSETS');
   WRITELN (LST,' ':33, 'LIABILITIES');
   WRITELN (LST);
   WRITE (LST,' ':5, 'CASH                  ', cash:10:2);
   WRITELN (LST,' ':8, 'ACCOUNT PAYABLE     ', act_payabl:10:2);
   WRITE (LST,' ':5, 'INVENTORY             ', inventory:10:2);
   WRITELN (LST,' ':8, 'NOTE PAYABLE        ', note_payab:10:2);
   WRITE (LST,' ':5, 'PRE-INSURANCE         ', pre_insure:10:2);
   WRITELN (LST,' ':8, 'WAGE PAYABLE        ', wage_payab:10:2);
```

```
    WRITE (LST,' ':5, 'ACCOUNT RECEIVABLE  ', act_receiv:10:2);
    WRITELN (LST,' ':8, 'INTEREST PAYABLE    ', int_payabl:10:2);
    WRITE (LST,' ':5, 'EQUIPMENTS          ', equipment:10:2);
    WRITELN (LST,' ':8, 'BANK LOAN           ', bank_loan:10:2);
    WRITE (LST,' ':5, 'FIXED PROPERTIES    ', fix_proprt:10:2);
    WRITELN (LST,' ':25, '-------------');
    WRITELN (LST,' ':43, 'TOTAL LIABILITY     ', liab_ttl:10:2);
    WRITELN (LST);
    WRITELN (LST,' ':43, 'CAPITAL STOCK       ', capital:10:2);
    WRITELN (LST,' ':43, 'RETAINED EARNING    ', r_earning:10:2);
    WRITE (LST,' ':22, '-------------');
    WRITELN (LST,' ':25, '-------------');
    WRITE (LST,' ':5, 'TOTAL ASSETS        ', asset_ttl:10:2);
    WRITELN (LST,' ':8, 'TOTAL EQUITIES      ', equity_ttl:10:2);
    WRITELN (LST); WRITELN (LST); WRITELN (LST); WRITELN (LST);
    WRITELN (LST,' ':25, 'MIS   INCOME   STATEMENTS');
    WRITELN (LST);
    WRITELN (LST,' ':23, 'Income statement for the year');
    WRITELN (LST);
    WRITELN (LST,' ':9,'TOTAL SALES :    ',' ':30, sales:10:2);
    WRITELN (LST);
    WRITELN (LST,' ':9, 'EXPENSES    :    ');
    WRITELN (LST,' ':14, 'Interest      ',' ':15, interest:10:2);
    WRITELN (LST,' ':14, 'Salaries      ',' ':15, salaries:10:2);
    WRITELN (LST,' ':14, 'Tax expense   ',' ':15, taxes_expn:10:2);
    WRITELN (LST,' ':14, 'Advertising   ',' ':15, advertise:10:2);
    WRITELN (LST,' ':14, 'Energy        ',' ':15, energy:10:2);
    WRITELN (LST,' ':14, 'Wages         ',' ':15, wages:10:2);
    WRITELN (LST,' ':14, 'Material      ',' ':15, material:10:2);
    WRITELN (LST,' ':14, 'Transportation',' ':15, transport:10:2);
    WRITELN (LST,' ':14, 'Social expense',' ':15, social_exp:10:2);
    WRITELN (LST,' ':14, 'Insurance     ',' ':15, insurance:10:2);
    WRITELN (LST,' ':14, 'Miscellenous  ',' ':15, miscel_exp:10:2);
    WRITELN (LST,' ':44, '----------------------');
    WRITE (LST,' ':22, 'SUBTOTAL  : ', ' ':9,expense_tl:10:2);
    WRITELN (LST,' ':3, revenue_tl:10:2);
    WRITELN (LST);
    WRITELN (LST,' ':9, 'GROSS INCOME ', ' ':34, gross_marg:10:2);
END.
```

```
PROGRAM PROJ5 (INPUT, OUTPUT);
{*************************************************************}
{*  This program is used for eliminating the redundant lines  *}
{*   which have been generated by dBase III while converting   *}
{*   the extracting data to the text working file.             *}
{*************************************************************}

VAR
  IN_FILEVAR, OUT_FILEVAR   : TEXT;
  IN_FILE_NAME  : STRING[8];
  OUT_FILE_NAME : STRING[8];
  IN_FILE       : STRING[12];       {* input file name + ext.  *}
  OUT_FILE      : STRING[12];       {* output file name + ext. *}
  ONE_LINE      : STRING[132];      {* one line of text        *}
  OK            : BOOLEAN;          {* I/O testing boolean     *}

BEGIN
  CLRSCR;
  OK := FALSE;
  WHILE NOT OK DO
  BEGIN
    GOTOXY (20,2); LOWVIDEO;
    WRITELN ('LINEAR PROGRAMMING DATA FILE CONVERSION');
    GOTOXY (20,3);
    WRITELN ('=======================================');
    GOTOXY (15,10);
    HIGHVIDEO;
    WRITE ('ENTER INPUT FILE-NAME       : ');
    READLN (IN_FILE_NAME);
    IN_FILE := CONCAT (IN_FILE_NAME, '.DAT');
    GOTOXY (15,12);
    WRITE ('ENTER OUTPUT DATA-FILE NAME : ');
    READLN (OUT_FILE_NAME);
    OUT_FILE := CONCAT (OUT_FILE_NAME, '.DAT');
    WRITELN; WRITELN;
    ASSIGN (IN_FILEVAR, IN_FILE);
    {$I-}
    RESET (IN_FILEVAR);
    {$I+}
    OK := (IORESULT = 0);
    IF NOT OK THEN
      WRITELN ('CANNOT FIND REFERENCE FILE, PLEASE RE-ENTER')
      ELSE
        OK := TRUE;
  END;
  ASSIGN (OUT_FILEVAR, OUT_FILE);
  REWRITE (OUT_FILEVAR);
  READLN (IN_FILEVAR, ONE_LINE);
  WHILE NOT EOF(IN_FILEVAR) DO
  BEGIN
    READLN (IN_FILEVAR, ONE_LINE);
    WRITELN (OUT_FILEVAR, ONE_LINE);
  END;
  CLOSE (OUT_FILEVAR);
END.
```

MICROCOMPUTER BASED MANAGEMENT INFORMATION SYSTEM

By

Francis Lin

B.S., Tatung Institute of Technology, Taiwan, R.O.C., 1971

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

Kansas State University
Manhattan, Kansas

1986

A microcomputer based Management Information System is a useful tool for small to medium sized business organizations to provide effective business planning and to support decision making. Computer based information systems can be classified into four major types: transaction processing systems (TPS), office automation systems (OAS), management information systems (MIS), and decision support systems (DSS). TPS is related to the operational support of an organization, while the MIS related to the management support, and DSS is related to strategic planning. A MIS provides information to support predefined decision making, while the DSS supports semi-structured or unstructured decision making. Some functional overlap exists between TPS and MIS as well as MIS and DSS.

Five functional applications have been introduced in the MIS proposed in this report. The production requirements planning addresses the function of supplying the sales personnel with the cost analysis of the given product, and providing them with the estimated delivery time for given quantities of a product. The material requirements planning responds to the user with a suggested, best material reordering level. The cost analysis subsystem analyzes monthly costs for the top level managerial personnel to understand the variation of the production costs. The financial management subsystem paints a financial picture of the organization for investors and creditors. The management operation research provides a linear programming package to solve business problems in support of strategic planning.

1430-78
00-53