A PROPERTY MANAGEMENT SYSTEM FOR THE ADMINISTRATIVE
SCIENCES DEPARTMENT (U) NAVAL POSTGRADUATE SCHOOL
MONTEREY CA T M SEXTON SEP 87

UNCLASSIFIED
THESIS

A PROPERTY MANAGEMENT SYSTEM
FOR THE ADMINISTRATIVE SCIENCES
DEPARTMENT

by

Timothy M. Sexton

September 1987

Thesis Advisor Tung Bui

Approved for public release; distribution is unlimited.
A PROPERTY MANAGEMENT SYSTEM FOR THE ADMINISTRATIVE SCIENCES DEPARTMENT

Sexton, Timothy M.

Master's Thesis

1987 September

152

The Administrative Sciences Department (AS DEPT) of NPS maintains a considerable amount of computing and office equipment (property) to support its Students, Staff, Office, and Management Personnel. This thesis provides a relational database application - The Property Management System (PMS) to support the management and accountability of the AS DEPT property. The systems analysis and design methodology of a relational database is outlined. The implementation is undertaken on a microcomputer using dBase III plus. A data dictionary, program listings, and User's Manual are included.
Approved for public release; distribution is unlimited.

A Property Management System
for the Administrative Sciences Department

by

Timothy M. Sexton
Lieutenant, United States Navy
B.A., State University New York, Stonybrook, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

Author: Timothy M. Sexton

Approved by: Tung Bui, Thesis Advisor

Y. B. Mortagy, Second Reader

Willis R. Greer, Jr., Chairman,
Department of Administrative Sciences

Kneale T. Marshall,
Dean of Information and Policy Sciences
ABSTRACT

The Administrative Sciences Department (AS DEPT) of NPS maintains a considerable amount of computing and office equipment (property) to support its Students, Staff, Office, and Management Personnel. This thesis provides a relational database application - The Property Management System (PMS) to support the management and accountability of the AS DEPT property. The systems analysis and design methodology of a relational database is outlined. The implementation is undertaken on a microcomputer using dBase III plus. A data dictionary, program listings, and User's Manual are included.
TABLE OF CONTENTS

I. INTRODUCTION .......................................... 9
   A. BACKGROUND ....................................... 9
   B. PURPOSE ......................................... 9
   C. CHAPTER DESCRIPTION ............................ 10

II. PMS SYSTEMS ANALYSIS AND DESIGN ....................... 12
   A. METHODOLOGY .................................... 12
      1. Analysis ....................................... 12
      2. Design ....................................... 12
   B. PROPERTY MANAGEMENT SYSTEM DEVELOPMENT ......... 14
      1. Analysis ....................................... 14
      2. Logical Design ................................ 15
      3. Physical Design ............................... 19
      4. Implementation ............................... 20

III. CONCLUSIONS .......................................... 22

APPENDIX A: DATA DICTIONARY ............................. 24

APPENDIX B: PROGRAM LISTINGS ............................ 34
   1. ADDCOMP.PRG ................................... 34
   2. ADDMENU.PRG ................................... 43
   3. ADDPART.PRG ................................... 44
   4. ADD_HELP.PRG .................................. 49
   5. ADHOC.PRG ..................................... 50
   6. DELCOMP.PRG ................................... 52
   7. DELMENU.PRG ................................... 59
   8. DELPART.PRG ................................... 60
   9. DEL_HELP.PRG .................................. 69
<table>
<thead>
<tr>
<th>No.</th>
<th>File Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.</td>
<td>MAIN_HELP.PRG</td>
<td>71</td>
</tr>
<tr>
<td>11.</td>
<td>MODCOMP.PRG</td>
<td>73</td>
</tr>
<tr>
<td>12.</td>
<td>MODLOC.PRG</td>
<td>79</td>
</tr>
<tr>
<td>13.</td>
<td>MODMENU.PRG</td>
<td>91</td>
</tr>
<tr>
<td>14.</td>
<td>MODPART.PRG</td>
<td>93</td>
</tr>
<tr>
<td>15.</td>
<td>MOD_HELP.PRG</td>
<td>108</td>
</tr>
<tr>
<td>16.</td>
<td>OWNERS.PRG</td>
<td>109</td>
</tr>
<tr>
<td>17.</td>
<td>PMANF.PRG</td>
<td>110</td>
</tr>
<tr>
<td>18.</td>
<td>PMOD.PRG</td>
<td>112</td>
</tr>
<tr>
<td>19.</td>
<td>PROPERTY.PRG</td>
<td>114</td>
</tr>
<tr>
<td>20.</td>
<td>QRY_HELP.PRG</td>
<td>117</td>
</tr>
<tr>
<td>21.</td>
<td>QTR_RPT.PRG</td>
<td>117</td>
</tr>
<tr>
<td>22.</td>
<td>REPORTS.PRG</td>
<td>120</td>
</tr>
<tr>
<td>23.</td>
<td>RPT_HELP.PRG</td>
<td>121</td>
</tr>
<tr>
<td>24.</td>
<td>SLOCATIO.PRG</td>
<td>121</td>
</tr>
<tr>
<td>25.</td>
<td>SOWNER.PRG</td>
<td>125</td>
</tr>
<tr>
<td>26.</td>
<td>SUM_RPT.PRG</td>
<td>128</td>
</tr>
</tbody>
</table>

APPENDIX C: PMS USER'S MANUAL

1. INTRODUCTION
   a. Getting Started
   b. Passwords

2. PROPERTY MANAGEMENT SYSTEM OPERATIONS
   a. Help
   b. Lists or Searches
   c. Property Reports
   d. Enter New Property
   e. Delete Property
   f. Modify Property

3. SPECIAL OPERATIONS
   a. General Editing
   b. ESCape
   c. Printing
   d. Backups
e. Exiting ................................................. 149

LIST OF REFERENCES ........................................... 150

INITIAL DISTRIBUTION LIST ..................................... 151
LIST OF TABLES

1. DATA VOLUME STORAGE REQUIREMENTS .......................................................... 21
2. DATA FILES ...................................................................................................... 25
3. DATA ELEMENTS .............................................................................................. 26
4. FILES USED BY PROGRAMS ........................................................................... 29
5. CALLS .............................................................................................................. 31
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Data Flow Diagram</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Data Structure Diagram</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Hierarchy Diagram</td>
<td>20</td>
</tr>
<tr>
<td>C.1</td>
<td>Initial PMS Screen</td>
<td>132</td>
</tr>
<tr>
<td>C.2</td>
<td>Passwords</td>
<td>132</td>
</tr>
<tr>
<td>C.3</td>
<td>Main Menu</td>
<td>133</td>
</tr>
<tr>
<td>C.4</td>
<td>List and Search Menu</td>
<td>134</td>
</tr>
<tr>
<td>C.5</td>
<td>Component Search Screen</td>
<td>135</td>
</tr>
<tr>
<td>C.6</td>
<td>Custodian Listing Screen</td>
<td>136</td>
</tr>
<tr>
<td>C.7</td>
<td>Mfg Search Screen</td>
<td>137</td>
</tr>
<tr>
<td>C.8</td>
<td>Quarterly Report Screen</td>
<td>138</td>
</tr>
<tr>
<td>C.9</td>
<td>Component Entry Screen</td>
<td>139</td>
</tr>
<tr>
<td>C.10</td>
<td>Part Entry Screen</td>
<td>141</td>
</tr>
<tr>
<td>C.11</td>
<td>Component Deletion Screen</td>
<td>142</td>
</tr>
<tr>
<td>C.12</td>
<td>Part Deletion Screen</td>
<td>143</td>
</tr>
<tr>
<td>C.13</td>
<td>Modify Property Screen</td>
<td>144</td>
</tr>
<tr>
<td>C.14</td>
<td>Modify Component Assignment Screen</td>
<td>144</td>
</tr>
<tr>
<td>C.15</td>
<td>Sample Component Reassignment</td>
<td>145</td>
</tr>
<tr>
<td>C.16</td>
<td>Modify Component Screen</td>
<td>146</td>
</tr>
<tr>
<td>C.17</td>
<td>Sample Component Modification</td>
<td>146</td>
</tr>
<tr>
<td>C.18</td>
<td>Modify Part Screen</td>
<td>147</td>
</tr>
<tr>
<td>C.19</td>
<td>Sample Part Modification</td>
<td>148</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

A. BACKGROUND

A structured analysis was conducted in 1984 to determine the computing needs of the Administrative Sciences Department (AS DEPT) of the Naval Postgraduate School (NPS). This analysis defined three different sub-systems:

1. Financial
2. Personnel
3. Property

to keep track of all information pertaining to the management and control of departmental activities. A prototype system was developed and implemented in 1986 in part as a feasibility study, implementing some of the features of each subsystem outlined in the previous analysis. The prototype proved the system feasible by showing that many of the manual procedures could be automated. [Ref. 1,2]

However, various factors lead to the decline of use, and ultimate abandonment of the prototype. The limited amount of information provided in any of the three sub-systems was the major complaint. This motivated the AS DEPT to select the Property sub-system for full development.

The term "property" used in this thesis does not connote the meaning of real estate. Throughout this thesis the term "property" will be used to refer to equipment and accessories for which the AS DEPT desires to maintain accountability. The AS DEPT maintains a considerable amount of office and computing equipment to support department office personnel, teaching staff, and students. This property is either assigned to, or made available for use both on and off campus. Faced with a small office staff, and a high turnover rate, property accounting never received the attention desired by management.

B. PURPOSE

This thesis has two main objectives:

1. Design, develop, and implement a database application - the Property Management System (PMS) - to improve the property accountability for the Administrative Science Department of the Naval Postgraduate School.
2. Outline the database development process using this application as an example.
The first objective of this thesis should assist the AS DEPT in managing department resources, provide better services, as well as furnish the administrative accounting requirements established by the Naval Postgraduate School for certain classifications of property. By centrally automating the property accounting function, timely information can be provided quickly and accurately. Therefore this will assist in planning both service support and property acquisition.

Database system development is similar to other type business applications, but can be more complicated due to the amount of data stored, and the degree of sharing involved. This thesis uses the generally accepted methodology known as systems analysis and design (SAD) to accomplish the database development. SAD is a six step methodical and iterative process as the system moves from concept to implementation. These six steps or stages make up the system life cycle. The steps are:

1. problem definition
2. feasibility study
3. analysis
4. design
5. implementation
6. maintenance

To achieve the second objective of this thesis, the focus will be on the system design, and implementation steps of the life cycle. The previous life cycle steps were in essence performed in the earlier structured analysis and system prototype. However, the process of designing the Property Management System required the verification and update of prior works to correct identified inadequacies. The purpose in outlining the design, and implementation is to assist in the system maintenance, by providing the rationale behind these key decisions.

C. CHAPTER DESCRIPTION

Chapter II reviews database development activities which provides the framework for the PMS development. Design and implementation concerns related to the PMS application will be presented using this framework as an outline.

Chapter III discusses usability and expandability issues. Usability pertains to prevalent system operation supported by the PMS application. Capabilities beyond standard data requests are addressed for qualified dBase III plus programmers. Finally the chapter presents the author's opinions on the PMS expandability.
The appendices provide useful documentation for maintenance and system operation. In Appendix A, the data dictionary includes descriptions of files and data elements. Appendix B contains the program listings of the installed system. The user’s manual is reproduced in Appendix C. It serves as a reference for the user providing direction and operation guidance.
II. PMS SYSTEMS ANALYSIS AND DESIGN

A. METHODOLOGY

1. Analysis

As stated in the previous chapter the development of the Property Management System began with the analysis stage of the system analysis and design life cycle. The focus of analysis is logical, concentrating on what needs to be done, not how. During analysis goals and constraints are identified for the user's approval. Yourdon, a major proponent of Structured Analysis techniques calls this package specification [Ref. 3: p. 51].

As the system moves toward development it is imperative for the analyst to functionally understand the system to be developed. A set of tools are available to assist in analysis. Two such tools are the Data Flow Diagram (DFD) and, the Data Dictionary.

The DFD is the primary means for the analyst to communicate this understanding to the user. A DFD is an idealized model of the proposed system ignoring implementation details. It is used to describe graphically the contents and behavior of the system. A DFD reflects the system functions that must be performed, identifying the data, data flows, data stores, and processes involved in transforming the data. Additionally, a DFD outlines the system boundaries by identifying the sources and destinations of data.

The data dictionary is used for supporting documentation. A data dictionary is a collection of data about the data. Data elements are defined and described, sources and use are also identified.

The final output of analysis is a physical constraints document. User requirements not involved with the logical model of the system that limit design are outlined. This is a text of specifications that are physical in nature. Examples are hardware selection, interactive processing or specified response times.

2. Design

The next stage of development is system design. Database design is a two step process. The first step is logical database design and involves building a logical
A data structure called a schema, conceptual schema, or logical schema. The next step entails translating the logical schema into a physical design. Physical design is dependent upon the particular database management system (DBMS) used for implementation. [Ref. 4]

Relations, tuples, and attributes are the elements of a relational database. Relations correspond to files, tuples to records, and attributes to data elements respectively. The contents of a tuple are a fixed number of attributes, the set of possible values of an attribute comprise the domain for that attribute. The DFD and data dictionary are excellent sources for these values.

A database logical structure is an overview of the data. It consists of determining the relations and the relationships between them. The approach to logical design involves aggregating and classifying data according to different user's views (meanings) of the data. Data is consolidated to represent the relations according to these user perceptions. A data structure diagram is one method to represent a logical structure. Like a DFD it is a graphic representation and used to model the database. This diagram illustrates the associations between relations. Four relationships are possible: none, one-to-one, one-to-many, and many-to-many.

A relation has certain identifiable properties. A relation is a flat file, each row (tuple) has a fixed number of fields (attributes). All tuples are unique with no duplicates allowed. A key uniquely identifies a tuple. The key may be a single attribute or a set of attributes. It is possible to have more than one key, and a primary key must be chosen. Alternate keys are referred to as candidate keys. Every relation has a key, since in the worst case a combination of all the attributes could be the key.

Relational database theory has outlined some important considerations in developing alternative logical schema. To eliminate inconsistencies within the database, redundancy needs to be minimized. Anomalies are consistency problems that arise due to data redundancy and are resultant of operations on the relations such as update, insertion, and deletion of attributes or tuples. To reduce these problems, larger relations are decomposed or projected into smaller relations. The projection is done vertically, selecting a common attribute between the two relations. If necessary the information can be recreated by joining the two smaller relations. Schema design is the essence of normalization outlined in relational database theory. [Ref. 5]
The second stage of database design involves transforming the logical schema into physical data constructs and designing the program modules necessary to manipulate the data. Two languages specific to a DBMS are provided for these purposes. Data constructs are declared using the Data Definition Language (DDL). This process requires specifying field, record, and file formats, and their constraints. Programs are created using the Data Manipulation Language (DML). Program modules are designed to manipulate the database to furnish desired outputs as well as providing the means to store the data. A hierarchy diagram is one method used to depict the structure of the program relationships.

B. PROPERTY MANAGEMENT SYSTEM DEVELOPMENT

1. Analysis

To begin designing the Property Management System interviews were conducted with the principal users to identify their needs and desires to provide the accountability of departmental property. The department at present has no established procedures and no one single individual assigned to maintain property. This makes it necessary to interview most of the AS department office personnel.

A number of interviews were conducted with the NPS property manager to identify items missed during the users interviews. It was a stated requirement of the AS department Chairman to maintain accountability within the schools guidelines. Upon completion of the interviews with the NPS property manager it was concluded that the transactions and data elements requested by the AS office personnel would provide the necessary information to maintain the accountability established by the NPS property manager.

The school requires that a DD1342 paper document be kept up to date and on file with the property manager for all plant property. A new classification - minor property, will have similar accountability requirements. Since at this time the requirements had not been established for minor property, the new system will provide the same accountability as plant property for all department property. This should meet the schools policy, once established, requiring minor changes if any. With the data elements identified it will be possible to identify and locate all AS department plant property and provide the information necessary to keep the DD1342 accurate.

The following physical constraints and requirements were compiled from the interviews:
1. The proposed system will be implemented on an IBM XT microcomputer, which the AS DEPT already owns. The existing daisy wheel printer is also to be utilized. The micro has two 10 MB hard disk drives, all of which may be utilized.

2. The proposed system will be on-line thereby allowing entries, deletions, and modifications as transactions occur.

3. The proposed system will be written with dBase III Plus which is much more familiar to department personnel and has a much greater chance of being maintained. The new system will be both interactive and menu-driven. It will also require minimal training for the user, given they are in possession of a general familiarity with the operation of microcomputers. A comprehensive user's manual will be provided to assist in training.

4. The new DBMS will provide the ability to answer ad-hoc queries concerning all necessary inventory information of the accountable property of the AS department. The data is to include a physical item description, the actual physical location of the item, the individual charged with the custody of the item, how the item is configured, and financial accounting information (price and requisition number). Queries will be provided to answer questions (singly or categorically) about items, custodians, property types and numbers, or locations of all the department property inventory. The new DBMS will also provide summary and report data for a quarterly department inventory report.

Basic transactions and data elements were extracted and identified from the interviews. A preliminary data dictionary and a data flow diagram were presented to the department supervisor along with the above requirements. The data elements were intentionally left from the DFD, see Figure 2.1. The DFD was used at this point to verify the system boundaries and basic transactions. This proved a useful method in extracting the user views, beginning the design process.

2. Logical Design

Users views were compiled and used to create the relations (files) necessary to maintain the information requested. The data elements identified were assembled from the user interviews and the DD1342 document. The user views match the data stores depicted in the Data Flow Diagram. The user views are listed as follows:

1. Components (aliases: Property, Inventory)
2. Owners (alias: Custodians)
3. Parts
4. History (alias: Deleted property)
5. Homes
Figure 2.1   Data Flow Diagram.
a. User View No: 1
- User View Name: Components (aliases: Property, Inventory)
- Description: Information describing a piece of equipment belonging to the Admin Science Department inventory.
- Data Elements:
  - Manufacturer
  - Model
  - Description
  - Mfg_serial #
  - Requisition #
  - Property type (plant, minor, other)
  - Custodian
  - Location code (storage, lab, office, home)
  - Date of issue

b. User View No: 2
- User View Name: Owner (alias: Custodian)
- Description: Information gathered about a custodian of a piece of equipment when such equipment is entered into inventory or reassigned.
- Two owner classes:
  1. AS Department - storage items or lab items
  2. Personnel - home or office items
- AS Department Data Elements:
  - AS Department
  - Location
- Personnel Data Elements:
  - Last Name
  - First Name
  - Office
  - Street *
  - City *
  - Home Phone *
* Removed to homes (see User View #5)
- Revised Personnel Data Elements:
  - Last Name
  - First Name
c. User View No: 3
- User View Name: Parts
- Description: Information describing a trackable part for a particular component.
- Data Elements:
  - Model
  - Description
  - Mfg Serial #
  - Property type
  - Property #
  - Price
  - Requisition #
  - Component Serial # (if blank, assumed storage and not a part of a component)

d. User View No: 4
- User View Name: History (alias: Deleted property)
- Description: Information used to track minor and plant property (parts or components) when deleted.
- Data Elements:
  - Mfg
  - Model
  - Serial #
  - Property type
  - Property #
  - Deletion date

e. User View No: 5
- User View Name: Homes
- Description: Information tracked for custodians that use components at home.
- Data Elements:
  - Last name
  - First name
  - Street
  - City
  - Phone
Relationships were identified and depicted in the Data Structure Diagram, see Figure 2.2. Components can have more than one part, and an owner may have more than one component. These are one-to-many relationships. A one-to-one relationship exists between an owner and home, since a custodian will have only one home address. Notice to get the home address location of a component will require linking these relations through an owner. The history relation shows no dependent relationships.

![Data Structure Diagram](image)

Figure 2.2 Data Structure Diagram.

3. Physical Design

The next phase in the design process was to translate the users views into the structures necessary to implement them in dBase III plus.

The data definition language provided with dBase III plus places restrictions on the name sizes of fields and files. Files are limited to 8 character names and use a .DBF extension. Each field must have a name of 10 characters or less. Also each field requires declaring a type and a length. Appendix A contains a listing of the files and data elements for the PMS.

A hierarchy diagram shown in Figure 2.3 outlines the modules necessary to implement the transactions identified during requirements analysis. Since it is intended for the program to be menu driven, the child nodes of the hierarchy diagram will be options presented to the user called from the parent nodes. Program descriptions are contained in Appendix A.
4. Implementation

Implementation required first writing code for each of the modules. Appendix B is a listing of all programs. Test data were generated and each module was tested thoroughly to ensure that no failure would occur. Due to the requirement that a user not need have a knowledge of dBase III plus this was deemed critical. Programs were written so that a user is walked through each step, with the program in control.

Testing also involved checking for data inconsistencies. As modules were written they were tested alone and in combination with calling modules. The testing proved very tedious but very rewarding. When a near final version was prepared for screening by the DEPT supervisor no system errors were discovered. This helped instill a sense of confidence with the system. A few inconspicuous errors were discovered and required correction.

The screening was an opportunity for feedback about the output presentations and formats. No major modifications were noted and once the system was finalized it was tested with the real data. The system proved reliable and accepted as satisfactory by the dept supervisor.

Estimates were made on the maximum file sizes and a storage requirements analysis is outlined in Table 1. This showed that the entire database can reside on a

Figure 2.3 Hierarchy Diagram.
single floppy disk which will allow an easy method of backing up the database files. Copy commands are included in an Autoexec.bat file contained on the system boot up disk. When the system is exited copies are done without a user having to remember to do so.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA VOLUME STORAGE REQUIREMENTS</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>COMPONENTS</td>
</tr>
<tr>
<td>PARTS</td>
</tr>
<tr>
<td>OWNERS</td>
</tr>
<tr>
<td>HOMES</td>
</tr>
<tr>
<td>HISTORY</td>
</tr>
</tbody>
</table>

Finally a user's manual was prepared and delivered along with a backup copy of the system. With empty database files the system will fit on a single floppy disk. If a problem with the hard disk did occur it would be necessary to copy the systems disk and the latest copy of the database files. The user's manual is reproduced in Appendix C.
III. CONCLUSIONS

The Property Management System is presently being utilized and is providing a useful means of maintaining accountability for the AS DEPT property accomplishing the first objective of this thesis. The printed reports, and screen presentations of information output were furnished for this purpose.

The objective of the maintenance stage of the life cycle is to keep the system functioning. As a new application program, errors may surface requiring correction. Additionally it was observed that there was some degree of user uncertainty whether or not all outputs will be useful, or if additional outputs might be helpful. These are only a few of the likely future maintenance problems. For the continued success of this system it will have to be maintained.

None of the users are dBase III plus programmers. It would be beneficial for a member of the AS DEPT office to learn the language. Capable faculty and students are available and should be recruited to handle maintenance functions in the interim. With the documentation contained in Appendixes of this thesis it will be possible for an individual with a reasonable level of dBase III plus to either modify existing programs or create new ones to service new user requirements.

The data manipulation language provided with dBase III plus allows accessing and manipulating the database without having to write programs to do so. There is no method of taking advantage of this feature from the PMS application. However the database files can be accessed and called directly into dBase III plus for use. For a dBase III plus programmer this would be the easiest way to handle single output requests.

Property management was only one of the functions addressed during the previous prototype development, the others were Personnel and Financial. The PMS application could easily be adapted to include a personnel database. This was not deemed necessary during development and not included. The department is very satisfied with the manual procedures established.

The AS DEPT has expressed a desire to automate its financial system. This should be the next development project to be undertaken. The PMS application would
work very well as a stand alone system without the need to integrate it with a financial system for the following reasons:

1. Property is a very small subset of the financial expenditures within the AS DEPT.
2. Very little data would be duplicated.
3. A spreadsheet application such as Lotus may serve better utility for a financial system.

Database development was outlined in the main body of the thesis in accordance with the second objective. The Systems Analysis and Design methodology proved very useful during the Property Management System development. The success of the PMS development using this technique can be attributed to three reasons:

1. A thorough understanding of the user requirements.
2. Attention to database structure prior to system design.
3. The communication and documentation provided with the analysis and design tools.

In conclusion, a goal of any development effort is to deliver a satisfactory system on time and within budget. This goal was accomplished with the Property Management System.
APPENDIX A
DATA DICTIONARY

This Appendix provides necessary information for the maintenance of the Property Management System. The information is organized into four sections and presented in table format. The usefulness of the four sections are discussed as follows:

1. Data Files - This section lists database and index files. The usefulness of database files should be obvious, all data are stored in these files. Index files are data related files used for output format and fast retrieval of data for output.

2. Data Elements - This section is the data dictionary in dBase III plus format. The structure and source of each data element is provided. Due to the 10 character name limitation a description is also given to help decipher the purpose of each element.

3. Files Used By Programs - This section provides the database and index files used by each program module. This section will prove useful in determining the effects of any program modifications.

4. Calls - This section provides a description of each of the program modules and shows the relationship between modules. The programs called by each module and the calling program of a particular module are provided.
## TABLE 2
### DATA FILES

<table>
<thead>
<tr>
<th>FILE NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPS.DBF</td>
<td>DATA</td>
<td>ATTRIBUTES ABOUT A COMPONENT PROPERTY ITEM</td>
</tr>
<tr>
<td>PARTS.DBF</td>
<td>DATA</td>
<td>ATTRIBUTES ABOUT A PART PROPERTY ITEM</td>
</tr>
<tr>
<td>OWNERS.DBF</td>
<td>DATA</td>
<td>NAME AND LOCATION OF A PROPERTY CUSTODIAN</td>
</tr>
<tr>
<td>HOMES.DBF</td>
<td>DATA</td>
<td>HOME ADDRESS OF A CUSTODIAN WITH HOME PROPERTY</td>
</tr>
<tr>
<td>HISTORY.DBF</td>
<td>DATA</td>
<td>AN ARCHIVE FOR DELETED PLANT OR MINOR PROPERTY</td>
</tr>
<tr>
<td>NAMES.NDX</td>
<td>INDEX</td>
<td>ON LAST_NAME + FIRST_NAME + LOCATION</td>
</tr>
<tr>
<td>COMP_SER.NDX</td>
<td>INDEX</td>
<td>ON COMP_SER</td>
</tr>
<tr>
<td>NAME_LOC.NDX</td>
<td>INDEX</td>
<td>ON LAST_NAME + FIRST_NAME + LOCATION</td>
</tr>
<tr>
<td>C_SER.NDX</td>
<td>INDEX</td>
<td>ON COMP_SER</td>
</tr>
<tr>
<td>L_FNAMES.NDX</td>
<td>INDEX</td>
<td>ON LAST_NAME + FIRST_NAME</td>
</tr>
<tr>
<td>TEMP.DBF</td>
<td>DATA</td>
<td>JOINS COMPS WITH OWNERS</td>
</tr>
<tr>
<td>TEMP2.DBF</td>
<td>DATA</td>
<td>JOINS COMPS WITH HOME</td>
</tr>
<tr>
<td>TEMP3.DBF</td>
<td>DATA</td>
<td>JOINS COMPS WITH PARTS</td>
</tr>
<tr>
<td>TEMP3.NDX</td>
<td>INDEX</td>
<td>ON LAST_NAME + FIRST_NAME + LOCATION + C_MFG + C_MODEL</td>
</tr>
<tr>
<td>TYPE_NUM.NDX</td>
<td>INDEX</td>
<td>ON P_TYPE + P_NUM</td>
</tr>
<tr>
<td>TEMP.NDX</td>
<td>INDEX</td>
<td>ON C_PTYPE + C_PNUM (FOR SUM.RPT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ON LAST_NAME + FIRST_NAME</td>
</tr>
</tbody>
</table>

"25"
<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>TYPE</th>
<th>WIDTH</th>
<th>SOURCE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CITY</td>
<td>CHAR</td>
<td>15</td>
<td>HOMES.DBF</td>
<td>A HOME CUSTODIAN'S HOME ADDRESS</td>
</tr>
<tr>
<td>COMP_SER</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>MANUFACTURER'S SERIAL # (KEY)</td>
</tr>
<tr>
<td>COMP_SER</td>
<td>CHAR</td>
<td>15</td>
<td>PARTS.DBF</td>
<td>COMPONENT MFG SERIAL # A PART IS ASSIGNED</td>
</tr>
<tr>
<td>C_DESC</td>
<td>CHAR</td>
<td>50</td>
<td>COMPS.DBF</td>
<td>DESCRIPTION OF A COMPONENT</td>
</tr>
<tr>
<td>C_MFG</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>COMPONENT MFG</td>
</tr>
<tr>
<td>C_MODEL</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>COMPONENT MODEL</td>
</tr>
<tr>
<td>C_PNUM</td>
<td>CHAR</td>
<td>10</td>
<td>COMPS.DBF</td>
<td>COMPONENT PROPERTY NUMBER (MINOR AND PLANT TYPES)</td>
</tr>
<tr>
<td>C_PRICE</td>
<td>NUM</td>
<td>10</td>
<td>COMPS.DBF</td>
<td>COMPONENT PRICE, USES A TEMPLATE (99,999.99)</td>
</tr>
<tr>
<td>C_TYPE</td>
<td>CHAR</td>
<td>1</td>
<td>COMPS.DBF</td>
<td>COMPONENT PROPERTY TYPE (PLANT, MINOR, OTHER)</td>
</tr>
<tr>
<td>C_REQN</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>DEPT REQUISITION #, USES A TEMPLATE 9999-NNNN/NNNNNN</td>
</tr>
<tr>
<td>DEL_DATE</td>
<td>DATE</td>
<td>8</td>
<td>HISTORY.DBF</td>
<td>DATE A COMPONENT OR PART IS PLACED INTO HISTORY</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>CUSTODIAN'S FIRST NAME (AS DEPT GETS ROOM #)</td>
</tr>
<tr>
<td>ELEMENT</td>
<td>TYPE</td>
<td>WIDTH</td>
<td>SOURCE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>------------</td>
<td>----------</td>
<td>-------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td>CHAR</td>
<td>15</td>
<td>OWNERS.DBF</td>
<td>CUSTODIAN'S FIRST NAME (COMPOSITE KEY)</td>
</tr>
<tr>
<td>FIRST_NAME</td>
<td>CHAR</td>
<td>15</td>
<td>HOMES.DBF</td>
<td>HOME CUSTODIAN'S FIRST NAME (COMPOSITE KEY)</td>
</tr>
<tr>
<td>ISSUE_DATE</td>
<td>DATE</td>
<td>8</td>
<td>COMPS.DBF</td>
<td>DATE A COMPONENT ASSIGNED TO A CUSTODIAN</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>CUSTODIAN'S LAST NAME (AS DEPT GETS AS DEPT)</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>CHAR</td>
<td>15</td>
<td>OWNERS.DBF</td>
<td>CUSTODIAN'S LAST NAME (COMPOSITE KEY)</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>CHAR</td>
<td>15</td>
<td>COMPS.DBF</td>
<td>HOME CUSTODIAN'S LAST NAME (COMPOSITE KEY)</td>
</tr>
<tr>
<td>LOCATION</td>
<td>CHAR</td>
<td>8</td>
<td>OWNERS.DBF</td>
<td>BLDG-ROOM # OF A COMPONENT CUSTODIAN</td>
</tr>
<tr>
<td>LOC_CODE</td>
<td>CHAR</td>
<td>1</td>
<td>COMPS.DBF</td>
<td>LOCATION CODE (HOME, OFFICE, LAB, STORAGE)</td>
</tr>
<tr>
<td>MFG</td>
<td>CHAR</td>
<td>15</td>
<td>HISTORY.DBF</td>
<td>MFG OF DELETED COMPONENT</td>
</tr>
<tr>
<td>MODEL</td>
<td>CHAR</td>
<td>15</td>
<td>HISTORY.DBF</td>
<td>COMPONENT OR PART MODEL OF DELETED PROPERTY</td>
</tr>
<tr>
<td>PART_SER</td>
<td>CHAR</td>
<td>15</td>
<td>PARTS.DBF</td>
<td>MANUFACTURER'S SERIAL #</td>
</tr>
<tr>
<td>PHONE</td>
<td>CHAR</td>
<td>13</td>
<td>HOMES.DBF</td>
<td>A HOME CUSTODIAN'S HOME PHONE # USES A TEMPLATE (999)999-9999</td>
</tr>
</tbody>
</table>
### TABLE 3

**DATA ELEMENTS (CONT'D.)**

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>TYPE</th>
<th>WIDTH</th>
<th>SOURCE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNUM</td>
<td>CHAR</td>
<td>10</td>
<td>HISTORY.DBF</td>
<td>PROPERTY # OF A DELETED PART OR COMPONENT</td>
</tr>
<tr>
<td>PTYPE</td>
<td>CHAR</td>
<td>1</td>
<td>HISTORY.DBF</td>
<td>PROPERTY TYPE OF A DELETED PART OR COMPONENT</td>
</tr>
<tr>
<td>P_DESC</td>
<td>CHAR</td>
<td>50</td>
<td>PARTS.DBF</td>
<td>A DESCRIPTION OF A PARTICULAR PART</td>
</tr>
<tr>
<td>P_MODEL</td>
<td>CHAR</td>
<td>15</td>
<td>PARTS.DBF</td>
<td>PART MODEL</td>
</tr>
<tr>
<td>P_NUM</td>
<td>CHAR</td>
<td>10</td>
<td>PARTS.DBF</td>
<td>PART PROPERTY # (MINOR AND PLANT TYPES)</td>
</tr>
<tr>
<td>P_PRICE</td>
<td>NUM</td>
<td>8</td>
<td>PARTS.DBF</td>
<td>PRICE OF A PARTICULAR PART, USES A TEMPLATE 9,999.99</td>
</tr>
<tr>
<td>P_TYPE</td>
<td>CHAR</td>
<td>1</td>
<td>PARTS.DBF</td>
<td>PART PROPERTY TYPE (PLANT,MINOR,OTHER)</td>
</tr>
<tr>
<td>P_REQN</td>
<td>CHAR</td>
<td>15</td>
<td>PARTS.DBF</td>
<td>DEPT REQUISITION # USES A TEMPLATE 9999-NNNN/NNNNN</td>
</tr>
<tr>
<td>SERIAL_NUM</td>
<td>CHAR</td>
<td>15</td>
<td>HISTORY.DBF</td>
<td>MFG SERIAL # OF A DELETED PART OR COMPONENT</td>
</tr>
<tr>
<td>STREET</td>
<td>CHAR</td>
<td>25</td>
<td>HOMES.DBF</td>
<td>A HOME CUSTODIAN'S ADDRESS</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>DATABASE FILES</td>
<td>INDEXES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDCOMP. PRG</td>
<td>HOMES. DBF</td>
<td>L_FNAMES. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OWNERS. DBF</td>
<td>NAMES. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPS. DBF</td>
<td>NAME_LOC. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD_Help. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDMENU. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDPART. PRG</td>
<td>PARTS. DBF</td>
<td>C_SER. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPS. DBF</td>
<td>COMP_SER. NDX,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NAME_LOC. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADHOC. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELCOMP. PRG</td>
<td>HISTORY. DBF</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARTS. DBF</td>
<td>C_SER. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOMES. DBF</td>
<td>L_FNAMES. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OWNERS. DBF</td>
<td>NAMES. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPS. DBF</td>
<td>COMP_SER. NDX,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NAME_LOC. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEL_Help. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELMENU. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELPART. PRG</td>
<td>HISTORY. DBF</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARTS. DBF</td>
<td>C_SER. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPS. DBF</td>
<td>COMP_SER. NDX,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NAME_LOC. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAIN_Help. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODCOMP. PRG</td>
<td>PARTS. DBF</td>
<td>C_SER. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HOMES. DBF</td>
<td>L_FNAMES. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPS. DBF</td>
<td>COMP_SER. NDX,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NAME_LOC. NDX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOD_Help. PRG</td>
<td>NONE</td>
<td>NONE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

29
<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>DATABASE FILES</th>
<th>INDEXES</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODLOC.PRG</td>
<td>OWNERS.DBF</td>
<td>NAMES.NDX</td>
</tr>
<tr>
<td></td>
<td>COMPS.DBF</td>
<td>COMPSER.NDX, NAME_LOC.NDX</td>
</tr>
<tr>
<td>MODMENU.PRG</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>MODPART.PRG</td>
<td>PARTS.DBF</td>
<td>C_SER.NDX</td>
</tr>
<tr>
<td></td>
<td>HOMES.DBF</td>
<td>L_FNAMES.NDX</td>
</tr>
<tr>
<td></td>
<td>OWNERS.DBF</td>
<td>NAMES.NDX</td>
</tr>
<tr>
<td></td>
<td>COMPS.DBF</td>
<td>COMP_SER.NDX, NAME_LOC.NDX</td>
</tr>
<tr>
<td>OWNERS.PRG</td>
<td>HOMES.DBF</td>
<td>L_FNAMES.NDX</td>
</tr>
<tr>
<td></td>
<td>OWNERS.DBF</td>
<td>L_FNAMES.NDX, NAMES.NDX</td>
</tr>
<tr>
<td>PMANF.PRG</td>
<td>COMPS.DBF</td>
<td>NONE</td>
</tr>
<tr>
<td>PMOD.PRG</td>
<td>COMPS.DBF</td>
<td>NONE</td>
</tr>
<tr>
<td>PROPERTY.PRG</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>QRY_HELP.PRG</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>QTR_RPT.PRG</td>
<td>PARTS.DBF</td>
<td>C_SER.NDX</td>
</tr>
<tr>
<td></td>
<td>OWNERS.DBF</td>
<td>NAMES.NDX</td>
</tr>
<tr>
<td></td>
<td>COMPS.DBF</td>
<td>COMP_SER.NDX, NAME_LOC.NDX</td>
</tr>
<tr>
<td>REPORTS.PRG</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>RPT_HELP.PRG</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>SLOCATIO.PRG</td>
<td>COMPS.DBF</td>
<td>NAME_LOC.NDX</td>
</tr>
<tr>
<td>SOWNER.PRG</td>
<td>COMPS.DBF</td>
<td>NAME_LOC.NDX, NAMES.NDX</td>
</tr>
<tr>
<td>SUM_RPT.PRG</td>
<td>PARTS.DBF</td>
<td>C_SER.NDX</td>
</tr>
<tr>
<td></td>
<td>COMPS.DBF</td>
<td>COMP_SER.NDX, NAME_LOC.NDX</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>CALLS</td>
<td>CALLED BY</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>ADDCOMP.PRG</td>
<td>ENTERS COMPONENTS INTO COMPS.DBF AND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASSIGNS CUSTODIANS PLACING THEM INTO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OWNERS OR HOMES IF NOT ON FILE</td>
<td>ADDMENU.PRG</td>
</tr>
<tr>
<td>ADD_Help.PRG</td>
<td>DESCRIBES IN GENERAL THE PROCEDURES TO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENTER A PART OR COMPONENT</td>
<td></td>
</tr>
<tr>
<td>ADDMENU.PRG</td>
<td>MENU DISPLAY OF THE CHOICE TO ENTER A</td>
<td>ADDCOMP.PRG</td>
</tr>
<tr>
<td></td>
<td>PART OR COMPONENT</td>
<td>PROPERTY.PRG</td>
</tr>
<tr>
<td>ADDPART.PRG</td>
<td>ENTER PARTS AND PLACES THEM INTO PARTS,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ASSIGNING THEM TO STORAGE OR A COMPONENT</td>
<td>ADDMENU.PRG</td>
</tr>
<tr>
<td>ADHOC.PRG</td>
<td>DISPLAYS THE MENU FOR LISTS AND SEARCHES</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OWNERS.PRG</td>
<td>PROPERTY.PRG</td>
</tr>
<tr>
<td></td>
<td>OWNER.PRG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SLOCATION.PRG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SORY HELP.PRG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MANUFACTURE.PRG</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PMOD.PRG</td>
<td></td>
</tr>
<tr>
<td>DELCOMP.PRG</td>
<td>DELETES COMPONENTS AND PLACES PLANT AND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MINOR PROPERTY INTO HISTORY</td>
<td>DELMENU.PRG</td>
</tr>
<tr>
<td>DEL_HELP.PRG</td>
<td>DESCRIBES IN GENERAL THE PROCEDURES TO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE A PIECE OF PROPERTY</td>
<td>DELMENU.PRG</td>
</tr>
<tr>
<td>DELMENU.PRG</td>
<td>DISPLAY THE CHOICE TO DELETE A PART OR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>COMPONENT</td>
<td>DELCOMP.PRG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELPART.PRG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELHELP.PRG</td>
</tr>
<tr>
<td>DELPART.PRG</td>
<td>DELETES PARTS, PLACES PLANT AND MINOR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PROPERTY INTO HISTORY</td>
<td>DELMENU.PRG</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>CALLS</th>
<th>CALLED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN_HLP.PRG</td>
<td>DESCRIBES THE PROPERTY SYSTEM AND EXPLAINS IN GENERAL THE TASKS AVAILABLE</td>
<td>PROPERTY.PRG</td>
</tr>
<tr>
<td>MODCOMP.PRG</td>
<td>ALLOWS THE MODIFICATION OF A COMPONENT RECORD (EXCEPT FIELDS TO ASSIGN OWNERSHIP), IF THE SERIAL # IS CHANGED IT IS REFLECTED IN PARTS</td>
<td>MODMENU.PRG</td>
</tr>
<tr>
<td>MOD_HELP.PRG</td>
<td>DESCRIBES IN GENERAL THE PROCEDURES USED TO CHANGE A PROPERTY RECORD</td>
<td>MODMENU.PRG</td>
</tr>
<tr>
<td>MODLOC.PRG</td>
<td>ALLOWS REASSIGNING A COMPONENT TO A NEW CUSTODIAN, IF NOT ON FILE CUSTODIAN ADDED TO OWNERS OR HOME</td>
<td>MODMENU.PRG</td>
</tr>
<tr>
<td>MODMENU.PRG</td>
<td>DISPLAYS THE CHOICES TO MODIFY A COMPONENT OR PART RECORD, ALSO ALLOWS REASSIGNING THEM A NEW CUSTODIAN</td>
<td>MODCOMP.PRG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODLOC.PRG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MODPART.PRG</td>
</tr>
<tr>
<td>MODPART.PRG</td>
<td>ALLOWS MODIFICATION OF A PART RECORD, IF REASSIGNED TO A DIFFERENT COMPONENT CHECKS TO SEE IF COMPONENT IS ON FILE</td>
<td>MODMENU.PRG</td>
</tr>
<tr>
<td>OWNERS.PRG</td>
<td>LISTS CUSTODIANS ON FILE (NAME AND LOCATION)</td>
<td>ADHOC.PRG</td>
</tr>
<tr>
<td>PMANF.PRG</td>
<td>DISPLAYS THE COMPONENTS OF A DESIRED MFG</td>
<td>ADHOC.PRG</td>
</tr>
<tr>
<td>PMOD.PRG</td>
<td>FINDS ALL COMPONENTS OF A DESIRED MFG’S MODEL</td>
<td>ADHOC.PRG</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>CALLS</td>
<td>CALLED BY</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>PROPERTY.PRG</td>
<td>MAIN MENU TO DISPLAY THE VARIOUS TASKS THAT ARE AVAILABLE FOR THE USER, CHECKS PASSWORD/ACCESS</td>
<td>DOS DBASE III</td>
</tr>
<tr>
<td>QRY_HELP.PRG</td>
<td>DESCRIBES IN GENERAL THE LIST AND SEARCH CAPABILITIES</td>
<td>NONE ADHOC.PRG</td>
</tr>
<tr>
<td>QTR_RPT.PRG</td>
<td>PRINTS 3 REPORTS: 1. COMPONENTS GROUPED CUSTODIAN/LOCATION 2. PARTS GROUPED BY CUSTODIAN/COMPONENT 3. STOCK PARTS NOT ASSIGNED</td>
<td>TEMP.FRM TEMP3.FRM STOKPART.FRM</td>
</tr>
<tr>
<td>REPORTS.PRG</td>
<td>MENU TO DISPLAY THE CHOICES TO PRINT A QUARTERLY OR SUMMARY REPORT</td>
<td>QTR_RPT.PRG PROPERTY.PRG</td>
</tr>
<tr>
<td>RPT_HELP.PRG</td>
<td>DESCRIBES IN GENERAL THE PROCEDURES TO PRINT THE PROPERTY REPORTS</td>
<td>NONE REPORTS.PRG</td>
</tr>
<tr>
<td>SLOCATIO.PRG</td>
<td>DISPLAYS COMPONENTS GROUPED BY THE LOCATION CODE</td>
<td>NONE ADHOC.PRG</td>
</tr>
<tr>
<td>SOWNER.PRG</td>
<td>DISPLAYS COMPONENTS OF A PARTICULAR CUSTODIAN</td>
<td>NONE ADHOC.PRG</td>
</tr>
<tr>
<td>SUM_RPT.PRG</td>
<td>PRINTS 3 REPORTS: 1. COMPONENTS GROUPED BY PROPERTY TYPE &amp; NUMBER 2. ASSIGNED PARTS GROUPED 3. STOCK PARTS SAME GROUPING</td>
<td>TEMP1.FRM PARTSTOK.FRM PARTSUM.FRM</td>
</tr>
</tbody>
</table>
APPENDIX B
PROGRAM LISTINGS

1. ADDCOMP.PRG

*************** Program: ADDCOMP.PRG ***************
*Author...........: TIM SEXTON
*Purpose..........: enter components into COMPS and assign custodians placing them into OWNERS or HOMES if not presently on file
*Calls............: None
*Reserved.........: None
*Input/Output Files.: COMPS.DBF, OWNERS.DBF, HOMES.DBF

clear
set confirm on
set exact off
select c
use homes index 1_fnames
select b
use owners index names
select a
use comps index name_loc,comp_ser
do while .t.
    blank = space(15)
    mmfg = blank
    mmofd = blank
    mdesc = space (50)
    mser = blank
    mtype = " "
    mnnum = space(10)
    mprice = 0.00
    mreqn = blank
    mlcode = " "
    mtoday = date( )
    mlname = blank
    mfname = blank
    mstreet = space(25)
    mcity = blank
    mlocation = space(8)
    mphone = space(13)
    entering = .t.
    finished = .f.
    addowner = .f.
    addhome = .f.
    @ 0.16 say "COMPONENT ENTRY SCREEN"
    @ 1.0 to 1,79 double
    @ 2.0 say "Enter Component Information;"
    @ 21.0 to 21.79
do while .not. finished && entering a component
    do while entering && component information
        @ 2.56 say "date: "
        @ 2.61 say mtoday
        @ 3.12 say "mfg: "
    "

* entries begin at column 18
  c = 18

* enter mfg or exit
  @  3,c  get mmfg PICTURE "@N!"
  read
  if mmfg = blank
    set confirm off
    close databases
    release all
    return
  else
    @ 22.0 clear to 23.79
  endif
* enter model
  @  4,c  get mmod PICTURE "@N!"
  read
* enter mfg serial# (mandatory)
  no_ser = .t.
  do while no_ser
    @  5,c  get mser PICTURE "@N!"
    read
    if mser = blank
      @ 22.24 say "serial # may not be blank"
      delay = 0
      do while delay < 25
        delay = delay + 1
      enddo
      @ 22.0 clear to 22.79
    else
      no_ser = .f.
    endif
  enddo
* enter description
  @  6,c  get mdesc PICTURE "@N!"
  read
* enter location code
  set confirm off
  @  8.1  say "designated use:"
  @  8.20  say "(Office / Lab / Storage / Home)"
  @  8,c  get mloc_code PICTURE "@!A"
  read
  do while .not. mloc_code "$OoLlSsHh"
    mloc_code = " "
    @  8,c  get mloc_code PICTURE "@!A"
    read
  enddo
  set confirm on
* enter custodian information, search to see if on file
  * use owners for loc_codes O,S,L
  * use homes for loc_code H
  * last name, first name, and office or home addresses are mandatory
  do case
  * office use
    case mloc_code = "O"
      no_lname = .t.
      do while no_lname
        @  9.0  clear to 15.79
        @  9.6  say "Custodian"
        @  10.6  say "last name:"
        @ 10.9  get mlname PICTURE "@!A"
read
if mname = blank
   @ 22,24 say "Custodian's name may not be blank"
   delay = 0
   do while delay < 25
      delay = delay + 1
   enddo
   @ 22,0 clear to 22,79
else
   no lname = .f.
endif
enddo no lname

* check to see if on file
select owners
goto top
searched = .f.
checked = .f.
located = .f.
addowner = .f.
do while .not. searched
   if eof()
      searched = .t.
   else
      seek trim(mlname)
   endif
   if .not. found()
      searched = .t.
      addowner = .t.
   endif
   if found().and. location = "HOME"
      searched = .t.
      located = .t.
      checked = .t.
   endif
   if found().and. location <> "HOME"
      searched = .t.
      located = .t.
      checked = .f.
   endif
enddo

do while located
   do while .not. checked
      @ 11,5 say "first name: "
      @ 11,c say first_name
      @ 12,9 say "office: "
      @ 12,c say location
      set confirm off
      ans = ""
      do while .not. (ans AndAlso yYnN"
         ans = ""
         @ 22,25 say;
            "Is this the correct custodian? :"
         @ 23.34 say "(Yes / No)"
         @ 22,56 get ans picture "@!A" read
      enddo
      @ 22,0 clear to 23,79
      set confirm on
      if upper(ans) = "Y"
         located = .t.
         checked = .t.
         mfname = first_name
         mlocation = location
      else
         checked = .t.
      @ 11,c clear to 12,79
   enddo
enddo located

36
endif

enddo checked

do while located .and. checked .and. .not. eof()
skip
if eof()
  located = .f.
  addowner = .t.
endif
if last_name = mlname
  checked = .f.
else
  located = .f.
  addowner = .t.
endif
if location = "HOME"
  located = .t.
  checked = .t.
endif
endo located and checked
endo located

if addowner
  no_f_name = .t.
  do while no_f_name
    @ 11,5 say "first name: "
    @ 11,c get mfname picture "@!A"
    read
    if mfname = blank
      @ 22,20 say "A first name or first";+;
      " initial is required"
      delay = 0
      do while delay < 25
        delay = delay + 1
      enddo
    @ 22,0 clear to 22,79
    else
      no_f_name = .f.
    endif
  enddo no_f_name
@ 12.9 say "office: "
@ 12.27 say "(bldg-room)"
no_office = .t.
  do while no_office
    @ 12,c get mlocation PICTURE "@! (A-999)"
    read
    if mlocation = space(8)
      @ 22,28 say "Office may not be blank"
      delay = 0
      do while delay < 25
        delay = delay + 1
      enddo
    @ 22,0 clear to 22,79
    else
      no_office = .f.
    endif
  enddo no_office
endif no add owner

* supply use
  case mloc_code = "S"
  @ 9.0 clear to 15,79
  @ 11.8 say "custodian: AS DEPT"
  @ 12.9 say "location: (1-200)"
  mlname = "AS DEPT"
  mfname = "(1-200)"
  mlocation = "STORAGE "

37
* check to see if on file
  select owners
goto top
addowner = .f.
if eof()
  addowner = .t.
else
  locat for last_name = trim(mlname) .and.;
     first_name = trim(mfname) .and.;
     location = trim(mlocation)
endif
if .not. found()
  addowner = .t.
endif

* lab use
  case mloc_code = "L"
  @ 9.0 clear to 15.79
  @ 10.5 say " A - (I-158) Front  B - (I-158) Back  C - (I-224)"
  @ 11.5 say " D - (I-250)"
  @ 13.3 say "Enter one of the above lab locations : ;"
  set confirm off
  lab = " "
do while .not. lab $ "AaBbCcDd"
    lab = " "
    @ 13.41 get lab picture "@!A"
    read
  enddo
  set confirm on
  mlname = "AS DEPT"
mlocation = "LAB"
do case
  case upper(lab) = "A"
    mfname = "(I-158)F"
  case upper(lab) = "B"
    mfname = "(I-158)B"
  case upper(lab) = "C"
    mfname = "(I-224)"
  case upper(lab) = "D"
    mfname = "(I-250)"
  endcase

* check to see if on file
  select owners
goto top
addowner = .f.
if eof()
  addowner = .t.
else
  locat for last_name = trim(mlname) .and.;
     first_name = trim(mfname) .and.;
     location = trim(mlocation)
endif
if .not. found()
  addowner = .t.
endif

* home use
  case mloc_code = "H"
mlocation = "HOME"
no_lname = .t.
do while no_lname
  @ 9.0 clear to 15.79
  @ 9.6 say "Custodian"

@ 10.6 say "last name: "
@ 10,c get m lname PICTURE "$!A"
read
if m lname = space(15)
  @ 22,24 say "Custodian's name may not be blank"
delay = 0
  do while delay < 25
delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
else
  no lname = .f.
endif
endo no last name

* check to see if on file
select homes
goto top
searched = .f.
located = .f.
checked = .f.
addowner = .f.
addhome = .f.
do while .not. searched
if eof()
  searched = .t.
else
  seek trim(mlname)
endif
if .not. found()
  searched = .t.
  addowner = .t.
  addhome = .t.
endif
if found()
  searched = .t.
  located = .t.
  checked = .f.
  mfname = first_name
  mstreet = street
  mcity = city
endif
endo not searched

do while located
  do while .not. checked
    @ 11,5 say "first name: 
    @ 11,c say first_name
    @ 12,9 say "street: 
    @ 12,c say street
    @ 13,11 say "city: 
    @ 13,c say city
    @ 14,10 say "phone: 
    @ 14,c say phone
    set confirm off
    ans = " 
    do while .not. ans "$YnN"
      ans = 
      @ 22,25 say;
      "Is this the correct custodian?: "
      @ 23,34 say "[ Yes / No ]"
      @ 22,56 get ans picture "$!A"
      read
    enddo
    set confirm on
    @ 22,0 clear to 23,79
    if upper(ans) = "Y"
      located = .f.
    endif
checked = .t.
else
  checked = .t.
  @ 11,c clear to 14,79
endif
enddo checked
do while located .and. checked .and. .not. eof()
  skip
  if eof()
    located = .f.
    addowner = .t.
    addhome = .t.
  endif
  if last_name = mlname
    checked = .f.
  else
    located = .f.
    addowner = .t.
    addhome = .t.
  endif
enddo located and checked
enddo located
if addowner
  no fname = .t.
do while no_fname
  @ 11,5 say "first name:
  @ 11,c get mfname picture "@!A"
  read
  if mfname = blank
    @ 22,20 say "A first name or first"
    " initial is required"
    delay = 0
    do while delay < 25
      delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
  else
    no_fname = .f.
  endif
enddo no_fname
no-address = .t.
do while no_address
  @ 12,9 say "street: 
  @ 12,c get mstreet PICTURE "@!"
  @ 13,11 say "city: 
  @ 13,c get mcity PICTURE "@!A"
  @ 14,10 say "phone: 
  @ 14,c get mphone PICTURE "(999)999-9999"
  read
  if mstreet = space(25) .or. mcity = blank 
    @ 22,25 say "Street or City may not be blank"
    delay = 0
    do while delay < 25
      delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
  else
    no_address = .f.
  endif
enddo no_address
endif addowner
endcase location code
* enter property type (mandatory)
set confirm off
@ 16,2 say "property type: "
40
@ 16,20 say "(Plant / Minor / Other)"
@ 16,c get mptype PICTURE "@A"
  read
  do while .not. mptype $ "mNoOpP"
    mptype = " "
  @ 16,c get mptype PICTURE "@A"
  read
  enddo
  set confirm on
  * no property# for other type property
  if upper(mptype) = "O" && ensure it is blank
    mpnum = space(10)
  endif
  * enter property# (mandatory for plant and minor property types)
  if upper(mptype) = "M" .or. upper(mptype) = "P"
    @ 17,5 say "property #: "
    no_num = .t.
    do while no_num
      @ 17,c get mpnum PICTURE "@!
      read
      if mpnum = space(10)
        @ 22,15 say "Minor and Plant property"+;
        " require a property number"
        delay = 0
        do while delay < 25
          delay = delay + 1
        enddo
      else
        no_num = .f.
      endif
    enddo
  endif
  * enter price and requisition#
  @ 19,10 say "price: $"
  @ 20,9 say "reqn #: "
  @ 20,c get mprice PICTURE "@ 99999,99"
  @ 20,c get mreqn PICTURE "@R 9999-NNNNN/NNNNN"
  read
  * allow editing or abandon current entry
  @ 22,0 clear to 23,79
  @ 22,23 say "Is the above information correct? : ;"
  @ 23,28 say "[ Yes / No / Abandon ]"
  set confirm off
  ans = " "
  do while .not. ans $ "yYnNaA"
    ans = " "
    @ 22,57 get ans picture "@A"
    read
  enddo
  @ 22,15 clear to 23,79
  set confirm on
  * clear mvar that controls creating a new owner or home record
  if upper(ans) = "N"
    addowner = .f.
    addhome = .f.
    @ 9,0 clear to 20,79
  endif
  * abandon entry -> clear mvar and close any open dbf
  if upper(ans) = "A"
    set confirm off
    close databases
    release all
    return
  endif
* place entry into dbf
  if upper(ans) = "Y"  && add 'mvar to dbf
    entering = .f.
  endif
enddo entering
@ 22,20 say "Standby while your entry is placed on file"

* place in owners
  * location = HOME for home use
  * location = LAB for lab use
  if addowner
    select owners
    append blank
    replace last_name with trim(mlname)
    replace first_name with trim(mfname)
    replace location with trim(mlocation)
  endif

* place in homes
  if addhome
    select homes
    append blank
    replace last_name with trim(mlname)
    replace first_name with trim(mfname)
    replace street with trim(mstreet)
    replace city with trim(mcity)
    replace phone with mphone
  endif

* place in comps
  select comps
  append blank
  replace c_mfg with trim(mmfg)
  replace c_model with trim(mmod)
  replace c_desc with trim(mdesc)
  replace c_comp_ser with trim(mser)
  replace c_pctype with mptype
  replace c_pnum with trim(mpnum)
  replace c_price with mprice
  replace c_reqn with trim(mreqn)
  replace last_name with trim(mlname)
  replace first_name with trim(mfname)
  replace issue_date with mtoday
@ 22,0 clear to 23,79
@ 22,18 say "Do you have additional components to enter? : "
@ 23,28 say "(Yes / No )"
set confirm off
ans = " ">
do while .not. ans $ "yYN"
  ans = " "
  @ 22,62 get ans
  read
endo
d
if upper(ans) = "N"
  finished = .t.
clear
else
  && set up for the next one
  entering = .t.
  addowner = .f.
  addhome = .f.
  mmfg = blank
  mmod = blank
  mdesc = space (50)
  mser = blank
  mptype = " "

42
mpnum = space(10)
mprice = 0.00
mreqn = blank
mloc_code = " "
@ 22.0 clear to 23.79
@ 22.17 say; "Is the next component for the same custodian? : "
@ 23.31 say "[ Yes / No ]"
ans = " "
do while .not. ans "$yNn"
  ans = " "
  @ 22.63 get ans
  read
endo
set confirm on
@ 22.0 clear to 23.79
if upper(ans) = "N"
  miname = blank
  mfname = blank
  mstreet = space(25)
  mcity = blank
  mlocation = space(8)
  mphone = space(13)
endif
endif set up
@ 3.0 clear to 20.79
endo finished
set confirm off
release all
close databases
return
endo
* EOF addcomp.prg

2. ADDMENU.PRG

*************** Program: ADDMENU.PRG ***************
*Author............. TIM SEXTON
*Purpose............. Menu displays the choices: enter components or parts
*Calls............. ADDCOMP.PRG, ADDPART.PRG
*Input/Output Files.: NONE

* set up the screen environment
clear
set confirm off
* display the dialogue menu
do while .t.
clear
  @ 2.10 to 13.69 double
  @ 3.30 say "Property Entry Menu"
  @ 4.11 to 4.68 double
  @ 6.27 say " 1 - COMPONENT entry"
  @ 7.27 say " 2 - PART entry"
  @ 9.27 say " H - HELP"
  @ 11.27 say " 0 - RETURN to main menu"
@ 13,30 say " selection : ": "
choice = " "
@ 13,42 get choice
read
* place an asteriks next to a valid choice and erase all other rows
if choice $ "h012"
do case
case upper(choice) = "H"
   @ 9,26 say "**"
   choicerow = 9
   case choice = "0"
      @ 11,26 say "**"
      choicerow = 11
   otherwise
      @ 5+val(choice),26 say "**"
      choicerow = 5+val(choice)
endcase
firstrow = 6
rows = 7
rowcnt = 0
do while rowcnt < rows
   if rowcnt+firstrow <> choicerow
      @ firstrow+rowcnt,27 say space(25)
   endif
   rowcnt = rowcnt + 1
endoendf
* do a valid choice or loop back thru this program
   do case
case choice = "0"
      return
case choice = "1"
      do addcomp
case choice = "2"
      do addpart
case upper(choice) = "H"
      do add_help
   otherwise
      @ 17,22 say "****** not a valid selection ******"
   endif
endcase
endo
*EOF ADDMENU.PRG

3. ADDPART.PRG

*************** Program: ADDPART.PRG ***************
*Author ............: TIM SEXTON
*Purpose ............: enter parts and place into PARTS, assigning the part to storage or to a component on file
*Calls .............: None
*Reserved ...........: None
*Input/Output Files.: PARTS.DBF, COMPS.DBF

clear
set confirm on
set exact on
select b
use parts index c_ser
select a
use comps index comp_ser,name_loc

do while .t.
  blank = space(15)
  mmod = blank
  mdesc = space(50)
  mp_ser = blank
  mc_ser = blank
  mptype = " "
  mpnum = space(10)
  mprice = 0.00
  mreqn = blank
  entering = .t.
  still more = .t.
  finished = .f.
  
@ 0,21 say "PART ENTRY SCREEN"
@ 1,0 to 1,79 double
@ 3,0 say "Enter Part Information:"
@ 21,0 to 21,79 do while .not. finished && entering parts
do while entering && part information
  @ 5,10 say "model:
  @ 6,7 say "serial #:
  @ 7,4 say "description:
  @ 22,25 say "To EXIT leave model blank"

* entries begin at column 18
  c = 18
* enter model or exit
  @ 5,c get mmod PICTURE "@N!"
  read
  if mmod = blank
    set confirm off
    close databases
    release all
    return
  else
    @ 22,0 clear to 23,79
  endif
* enter mfg serial#
  @ 6,c get mp_ser PICTURE "@N!"
  read
* enter description
  @ 7,c get mdesc PICTURE "@N!"
  read
* determine if stock or component use
  set confirm off
  p_use = " "
  @ 9,1 say "designated use:"
  @ 9,20 say "(Storage / Component)"
  @ 9,c get p_use PICTURE "@!A"
  read
  do while .not. p_use "$CcSs"
    p_use = " "
    @ 9,c get p_use PICTURE "@!A"
    read
  enddo
  set confirm on

* enter component serial# for use = C, search to see if on file
* comp_ser = blank for storage use
* use Comps for use = C component ser # mandatory
* storage use
  if p_use = "S"
    mc_ser = blank
  endif
* component use
  if p_use = "C"
    @ 10,0 clear to 13,79
    @ 11,6 say "Component"
    @ 12,6 say "serial #:"
    searched = .f.
    do while .not. searched
      no_ser = .t.
      do while no_ser
        @ 12,c get mc_ser PICTURE "@!N"
        read
        if mc_ser = blank
          @ 22,20 say;
          "component's serial # may not be blank"
          delay = 0
          do while delay < 25
            delay = delay + 1
          enddo
          @ 22,0 clear to 22,79
        else
          no_ser = .f.
        endif
      enddo
    no_ser = .f.
  endif
  endif
  no component serial#

* check to see if on file
  select comps
  set order to 1
  goto top
  on_file = .f.
  done = .f.
  if eof() on_file = .f.
  else
    seek trim(mc_ser)
  endif
  if found()
    searched = .t.
    on_file = .t.
    done = .t.
  endif
  if .not. found() .or. .not. on_file
    @ 22,22 say "Component not on file !!!"
    delay = 0
    do while delay < 25
      delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
    set confirm off
    ans = ""
    do while .not. ans $"YyNn"
      ans = ""
      @ 22,20 say "Is this the correct serial #: :"
      @ 23,26 say "[ Yes / No ]"
      @ 22,50 get ans picture "@!A"
      read
    enddo
    @ 22,0 clear to 23,79
    set confirm on
    if upper(ans) = "Y"
      entering = .f.
      searched = .t.
  endif
still_more = .f.
done = .t.
@ 22,15 say;
"This component must be entered first"
delay = 0
do while delay < 25
delay = delay + 1
enddo
@ 22,0 clear to 22,79
else
@ 22,15 say;
"Please re-enter the component serial#"
delay = 0
do while delay < 25
delay = delay + 1
enddo
@ 22,0 clear to 22,79
mc_ser = blank
endif
endif not found
enddo searched
endif
if still_more
* enter property type (mandatory)
set confirm off
@ 14,2 say "property type: "
@ 14,20 say "(Plant / Minor / Other)"
@ 14,c get mptype PICTURE "@@A"
read
do while .not. mptype "$mMoOpP"
  mptype = ""
  @ 14,c get mptype PICTURE "@@A"
  read
endo
set confirm on
* no property# for other type property
if upper(mptype) = "O" && ensure it is blank
  mnunum = space(10)
endif
* enter property# (mandatory for plant and minor property types)
if upper(mptype) = "M" .or. upper(mptype) = "P"
  @ 15,5 say "property #: "
  no_num = .t.
  do while no_num
    @ 15,c get mnum PICTURE "@@!
    read
    if mnum = space(10)
      @ 22,12 say "Minor and Plant property ";
      "require a property number"
      delay = 0
      do while delay < 25
        delay = delay + 1
      enddo
      @ 22,0 clear to 22,79
    else
      no_num = .f.
    endif
  enddo
endif
* enter price and requisition#
@ 17,10 say "price: 
@ 18,9 say "reqn #: 
@ 17,c get mprice PICTURE "@@ 9.999,99"
@ 18,c get mreqn PICTURE "@R 9999-NNNN/NNNNN"
read
* allow editing or abandon current entry
  @ 22,0 clear to 23,79
  @ 22,20 say "Is the above information correct?: "
  @ 23,26 say "[ Yes / No / Abandon ]"
  set confirm off
  ans = "$" "
  do while .not. ans $ "YnNnA"
    ans = "$" "
    @ 22,54 get ans picture "@A" read
  enddo
  @ 22,15 clear to 23,79
  set confirm on

  * clear mvar that controls creating a new owner or home record
  if upper(ans) = "N"
    @ 9,0 clear to 20,79
  endif

  * abandon entry -> clear mvar and close any open dbf
  if upper(ans) = "A"
    set confirm off
    close databases
    release all
    return
  endif

  * place entry into dbf
  if upper(ans) = "Y" && add mvar to dbf
    entering = .f.
    @ 22,10 say;
    " standby while your entry is placed on file "
  endif

  * place in parts
    select parts
    append blank
    replace p_model with trim(mmod)
    replace p_desc with trim(mdesc)
    replace part_ser with trim(mp_ser)
    replace comp_ser with trim(mcs_ser)
    replace p_type with mtype
    replace p_pnum with trim(mpnum)
    replace p_price with mprice
    replace p_reqn with trim(mregn)
  endif

  endif still more
endo entering
  @ 22,0 clear to 23,79
  @ 22,15 say "Do you have additional parts to enter?: "
  @ 23,26 say "[ Yes / No ]"
  set confirm off
  ans = "$" "
  do while .not. ans $ "YnNnA"
    ans = "$" "
    @ 22,54 get ans read
  enddo
  if upper(ans) = "N"
    finished = .t.
    clear
  else
    && set up for the next one
    @ 22,0 clear to 23,79
    entering = .t.
    still_more = .t.
    mmod = blank
    mdesc = space(50)
mp_ser = blank
mp_type = ""
mpnum = space(10)
mp_price = 0.00
mpreqn = blank

@ 22.0 clear to 23.79
if upper(p_use) = "C"
@ 22.15 say "Is the next part for the same component?: "
@ 23.26 say "[ Yes / No ]"
ans = " "
do while .not. ans "$YnN"
ans = " "
@ 22.56 get ans
read
endo
did confirm on
@ 22.0 clear to 23.79
if upper(ans) = "N"
mc_ser = blank
endif
endif p_use = C
endif set up
@ 4.0 clear to 20.79
endo finished
set confirm off
release all
close databases
return
endo
* EOF addpart.prg

4. ADD_HELP.PRG

******************** Program: ADD_HELP.PRG ********************
*  
*Author.............: TIM SEXTON
*Purpose............: describes the options available to the user
*Calls..............: None
*Input/Output Files.: None

* begin the text dialogue
clear
@ 0.17 say "ENTER PROPERTY HELP MENU"
@ 1.0 to 1.79 double
text

* Designed to enable the user to enter all new Property either a Component or Part.
* Entering selection is extremely important!!! Ensure all the information is correct.
* If errors are made, the user should ensure that "NO" or "ABANDON" options are selected before continuing on.
* If a false record is filed by mistake then GOTO the "MODIFY or DELETE" section and follow the instructions to correct the mistake.

endtext
@ 21.0 TO 21.79
WAIT " -> press any key for more help or ESC to exit"
clear
@ 1.0 to 1.79 double
text

"CAUTIONS"
* Extremely important that all information is entered correctly
* Mistakes will be made so utilize the "NO" or "ABANDON" commands before resuming
* Recommend using DELETION and ARROW keys for modifying data entered so not to deviate outside designated fields

dendtext
@ 21.0 TO 21.79
WAIT " -> press any key for more help or ESC to exit"
clear
@ 1.0 to 1.79 double
text

"WARNINGS"
* Use of BACKSPACE key can cause PREMATURE exiting of an entry field, loop back for re-entrance of data.
* Serial Numbers must be accurate and precise

dendtext
@ 21.0 to 21.79
Wait " -> press any key to exit"
*EOF ADD_HELP.PRG

5. ADHOC.PRG

************************** Program: ADHOC.PRG **************************
*
*Author...............: TIM SEXTON
*Purpose...............: display the menu for queries and lists
*Calls...............: SLOCATION.PRG, SOWNER.PRG, OWNERS.PRG
*Input/Output Files.: None

do while .t.
* re-set the system incase a user ESCaped out of one of the program selections
   release all
   close databases
   set confirm off
   clear
* display the menu
   @ 2.10 to 17.69 double
   @ 3.19 say "$ List and Search Menu"
   @ 4.11 to 4.68 double
   @ 5.21 say "$ - Components assigned to a Custodian"
   @ 7.21 say "$ - Components assigned by Locations"
   @ 8.21 say "$ - Custodian Listing"
   @ 10.21 say "$ - Components of a single Manufacturer"
   @ 11.21 say "$ - Components of a single Model"

@ 13.21 say "H - HELP"
@ 15.21 say "0 - Return to MAIN MENU"
@ 17.30 say "selection : : ":
choice = " "
@ 17.42 get choice
read
* place an asterisks next to a valid choice and erase the other
* selections
if choice $H0123455 do case
  case upper(choice) = "H"
    @ 13.20 say "*"
    choicerow = 13
  case choice = "0"
    @ 15.20 say "*"
    choicerow = 15
  case choice $'1235
    @ 5+val(choice),20 say "*"
    choicerow = 5+val(choice)
  otherwise
    @ 5+val(choice),20 say "*"
    choicerow = 6+val(choice)
endcase
firstrow = 6
rows = 10
rowcnt = 0
do while rowcnt < rows
  if rowcnt+firstrow <> choicerow
    @ firstrow+rowcnt,17 say space(50)
  endif
  rowcnt = rowcnt + 1
endo
* do the choice selection if valid or loop back thru this
* program
do case
  case choice = "0"
    clear
    release all
    return
  case choice = "1"
    do sowner
  case choice = "2"
    do slocation
  case choice = "3"
    do owners
  case choice = "4"
    do pmanf
  case choice = "5"
    do pmod
  case upper(choice) = "H"
    do qry_help
  otherwise
    @ 19,22 say "****** not a valid selection ******"
? wait
@ 19,0 clear to 21,79
loop
endo
*EOF ADHOC.PRG
6. **DELCOMP.PRG**

************** Program: DELCOMP.PRG **************

*Author:.........* TIM SEXTON  
*Purpose:.........* delete components and place plant or minor property in history  
*Calls...........* None  
*Reserved..........* None  
*Input/Output Files:.* PARTS.DBF, COMPS.DBF, HISTORY.DBF, OWNERS.DBF, HOMES.DBF

```plaintext
clear
set confirm on
select e
use history
select d
use parts index c_ser && indexed on comp_ser
select c
use homes index l_fnames && indexed on last,first names
select b
use owners index names && indexed on last,first names
select a
use comps index comp_ser,name_loc && indexed on comp_ser && indexed on last,first names

do while .t.
  blank = space(15)
  mmfg = blank
  mmod = blank
  mser = blank
  mloc_code = " "
  mptype = " "
  mpnum = space(10)
  mtoday = date( )
  lname = blank
  fname = blank
  mpart_ser = blank
  entering = .t.
  finished = .f.
  delcomp = .f.
  delpart = .f.
  @ 0,16 say "COMPONENT DELETION SCREEN"
  @ 1,0 to 1,79 double
  @ 2,0 say "Enter Component"
  @ 21,0 to 21,79 double
  do while .not. finished
    && deleting a component
    && component entered
  do while entering
    && entries begin at column 18
    c = .8
    searched = .f.
    do while .not. searched
      * enter serial # or exit
      @ 2,56 say "date: "
      @ 2,61 say mtoday
      @ 4,7 say "serial #: "
      @ 4,c get mser PICTURE "@N!"
      @ 22,25 say "To EXIT leave serial # blank"
      read
      if mser = blank
        close databases
        release all
```
return
else
  @ 22,0 clear to 23,79
endif

* check to see if on file
select comps
set order to 1
goto top
on_file = .f.
correct = .f.
if .not. eof()
  set exact on
  seek trim(mser)
else
  on_file = .f.
correct = .f.
endif
if found()
correct = .t.
mmod = c_model
mmfg = c_mfg
mptype = c_pctype
mpnum = c_pnum
lname = last_name
fname = first_name
mloc_code = loc_code
@ 6,0 clear to 10,79
@ 6,12 say "mfg: "
@ 6,c say c_mfg
@ 7,10 say "model: "
@ 7,c say c_model
@ 8,4 say "description: "
@ 8,c say c_desc
@ 9,2 say "property type: "
do case
  case c_pctype = "P"
    @ 9,c say "Plant"
  case c_pctype = "M"
    @ 9,c say "Minor"
  otherwise
    @ 9,c say "Other"
endcase
if c_pctype = "M" .or. c_pctype = "P"
  @ 10,5 say "property #: "
  @ 10,c say c_pnum
endif
set confirm off
ans = " "
do while .not. ans "$YnN"
  ans = " "
  @ 22,22 say "Is this the correct component?: "
  @ 23,29 say "[ Yes / No ]"
  @ 22,53 get ans picture '@!A"
read
dendo
set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "Y"
  on_file = .t.
  searched = .t.
else
  on_file = .f.
correct = .t.
  @ 22,11 say "The component shown is the only "t;
  "one on file with this serial #"
delay = 0
do while delay < 40
delay = delay + 1
endif @ 22,0 clear to 22,79
@ 22,13 say "Your component is not on ":
"file no need to delete it"
delay = 0
do while delay < 40
delay = delay + 1
endo
delay = 0
do while delay < 40
delay = delay + 1
endo
if .not. found() .or. .not. correct
endif found
if .not. found() .or. .not. correct.
delay = 0
do while delay < 40
delay = delay + 1
endo
set confirm off
ans = " "
do while .not. ans "$YnN"
ans = " "
@ 22,22 say "Is this the correct serial #?: :"
@ 23,28 say "[ Yes / No ]"
@end get ans picture "@!A"
read
endo @ 22,0 clear to 23,79
set confirm on
if upper(ans) = "Y"
correct = .t.
on_file = .f.
@end say "Your component is not on file":
"no need to delete it"
delay = 0
do while delay < 40
delay = delay + 1
endo
@end say;
"Please re-enter the component serial # or"
delay = 0
do while delay < 40
delay = delay + 1
endo
@end set confirm on
if .not. on_file .and. correct
@ 22,15 say;
"Do you have additional components to delete?: :"
@ 23,29 say "[ Yes / No ]"
set confirm off
ans = " "
do while .not. ans "$YnN"
ans = " "
@end get ans
read
endo
set confirm on
if upper(ans) = "N"
set confirm off
release all
    close databases
return
else
    mser = blank
    correct = .f.
    @ 6,0 clear to 10,79
    @ 22,0 clear to 23,79
endif
endif not on file and done
enddo searched

* check to see parts are on file for this component
select parts
goto top
if eof()
    delpart = .f.
    delcomp = .t.
else
    seek trim(mser) && component serial #
endif
if .not. found() && no parts for this component
delpart = .f.
delcomp = .t.
endif not found
if found()  
    @ 12,5 say;
        "The following PART(S) are on file for this component:
    @ 14,5 say "Model"
    @ 14,18 say "Property Type"
    @ 14,36 say "Property #"
    line = 15
    do while comp_ser = mser .and. .not. eof()
        @ line,5 say p_model
        @ line,23 say p_pnum
        @ line,36 say p_pnum
        skip
        line = line + 1
    if line = 20
        @ 22,23 say "Additional parts are on file"
        delay = 0
        do while delay < 50
            delay = delay + 1
        enddo
        @ 22,0 clear to 22,79
        @ 15,0 clear to 19,79
        line = 15
    endif
enddo
@ 22,0 clear to 23,79
@ 22,19 say "Do you wish to delete the PART(S)? : ":
@ 23,26 say "[ Yes / No ]"
set confirm off
ans = " 
do while .not. ans "$y\text{YnN}$"
    ans = " "
    @ 22,54 get ans
    read
    set confirm on
if upper(ans) = "$N"
    @ 22,0 clear to 23,79
    @ 22,5 say "The PART(S) must be reassigned"+;
        " before this component can be deleted"
    delay = 0
    do while delay < 40
55
delay = delay + 1

enddo

@ 22,0 clear to 22,79
@ 22,16 say:
    "Do you have additional components to delete?: ":
@ 23,29 say "[ Yes / No ]"
set confirm off
ans = " "
do while .not. ans $ "YnN"
    ans = " "
    @ 22,61 get ans
    read
endo
set confirm on
if upper(ans) = "N"
    set confirm off
    release all
close databases
return
else
    mser = blank
    @ 22,0 clear to 23,79
    @ 4,0 clear to 20,79
endo
else
    && ans = yes
delpart = .t.
delcomp = .t.
    @ 22,0 clear to 23,79
endo
endif found
if delcomp
    * allow editing or abandon current entry
    @ 22,0 clear to 23,79
    @ 22,16 say "Do you wish to delete this component?: ":
    @ 23,26 say "[ Yes / No ]"
set confirm off
ans = " "
do while .not. ans $ "YnNaA"
    ans = " "
    @ 22,54 get ans picture "!A"
    read
endo
    @ 22,15 clear to 23,79
set confirm on
* clear mvar that controls deleting a component or part record
    if upper(ans) = "N"
        @ 22,20 say "Please re-enter a component serial # or"
        delay = 0
        do while delay < 40
            delay = delay + 1
        enddo
        @ 22,0 clear to 22,79
        mser = blank
delcomp = .f.
delpart = .f.
        @ 9,0 clear to 20,79
        @ 4,0 clear to 20,79
endo
    if upper(ans) = "Y"    && delete records
        entering = .f.
        @ 22,0 clear to 23,79
endo delcomp
endo entering
@ 22,20 say " Standby while the component is deleted 
if delcomp
    select comps
go to top
set exact on
delete for comp_ser = trim(mser)
select history
append blank
replace mfg with trim(mmfg)
replace model with trim(mmod)
replace serial_num with trim(mser)
replace ptype with mptype
replace pnum with trim(mpnum)
replace del_date with mtoday
endif delcomp
if delpart
    select parts
    goto top
    set exact on
    seek trim(mser)
    if found()
do while mser = trim(comp_ser) .and. .not. eof()
        mpart_ser = part_ser
        mmod = p_model
        mptype = p_ptype
        mpnum = p_pnum
        if upper(mptype) <> "O" && only plant or minor
            select history
            append blank
            replace model with trim(mmod)
            replace serial_num with trim(mpart_ser)
            replace ptype with mptype
            replace pnum with trim(mpnum)
            replace del_date with mtoday
endif
    skip
dodo
done
endif
* check to see if owner has more property on file
select comps
set order to 2
go to top
if eof()
    delowner = .t.
delhome = .t.
else
    set exact off
    seek trim(lname),trim(fname)
endif
if .not. found()
    delowner = .t.
delhome = .t.
endif not found
if found()
    delowner = .t.
delhome = .t.

57
do while last_name = trim(lname) .and.;
  first_name = trim(fname) .and. .not. eof()
  do case
    case upper(loc_code) = "H"
      delhome = .f.
      delowner = .f.
    case upper(loc_code) = "0"
      delowner = .f.
  endcase
  skip
  enddo
endif found

* no need to delete AS DEPT (labs or storage)
if mloc_code = "L" .or. mloc_code = "S"
  delhome = .f.
  delowner = .f.
endif
if delowner
  select owners
  set exact off
  goto top
  delete for last_name = trim(lname) .and.;
  first_name = trim(fname)
endif
if delhome .and. mloc_code = "H"
  select homes
  goto top
  set exact off
  delete for last_name = trim(lname) .and.;
  first_name = trim(fname)
  select owners
  set exact off
  goto top
  delete for last_name = trim(lname) .and.;
  first_name = trim(fname) .and. location = "HOME"
endif

@ 22,0 clear to 23,79
@ 22,16 say "Do you have additional components to delete?:
@ 23,26 say "[ Yes / No ]"
set confirm off
ans = " "
do while .not. ans $ "YyNn"
  ans = " "
  @ 22,61 get ans
  read
enddo
if upper(ans) = "N"
  finished = .t.
  clear
else
  & set up for the next one
  &
  @ 22,0 clear to 23,79
  mmfg = blank
  mmod = blank
  mser = blank
  mloc_code = " "
  mtype = " "
  mtype = space(10)
  mtoday = date( )
  lname = blank
  fname = blank
  mpart_ser = blank
  entering = .t.

58
finished = .f.
endif set up
@ 4,0 clear to 20,79
endo finished
@ 8,15 say "***** Standby while the files are updated *****"
select comps
pack
select parts
pack
select owners
pack
select homes
pack
set confirm off
release all
close databases
return
endo
* EOF delcomp.prg

7. DELMENU.PRG

**************** Program: DELMENU.PRG ****************
* *Author.............. TIM SEXTON
* *Purpose.............. displays the choice to delete parts
or components, and place them in the history
* *Calls.............. DELCOMP.PRG,DELPART.PRG
* *Input/Output Files.: None
* set up the screen environment
clear
set confirm off
* begin the menu dialogue
do while .t.
clear
@ 2,10 to 13.69 double
@ 3,30 say "Delete Property Menu"
@ 4,11 to 4.68 double
@ 6,27 say "1 - COMPONENT deletion"
@ 7,27 say "2 - PART deletion"
@ 9,27 say "H - HELP"
@ 11,27 say "0 - RETURN to main menu"
@ 13.30 say "selection : : "
choice = ""
@ 13.42 get choice
read
* place an asterisks next to a valid choice, and erase the other
* rows
if choice $ "hH012"
do case
  case upper(choice) = "H"
    @ 9.26 say "*"
    choicerow = 9
  case choice = "0"
    @ 11.26 say "*"
    choicerow = 11
otherwise
  @ 5+val(choice),26 say "**"
  choicerow = 5+val(choice)
endcase
firstrow = 6
rows = 7
rowcnt = 0
do while rowcnt < rows
  if rowcnt+firstrow <> choicerow
    @ firstrow+rowcnt,27 say space(25)
  endif
  rowcnt = rowcnt + 1
enddo
* do a valid choice or loop back thru this program
* do case
  case choice = "0"
    return
  case choice = "1"
    do delcomp
  case choice = "2"
    do delpart
  case upper(choice) = "H"
    do del_help.prg
  otherwise
    @ 17,22 say "****** not a valid selection *******"
    wait
  endcase
enddo
* EOF DELMENU.PRG

8. DELPART.PRG

*********************** Program: DELPART.PRG ***********************
*Author..............: TIM SEXTON
*Purpose.............: delete parts and place plant or minor property
  in history
*Calls...............: None
*Reserved........... : None
*Input/Output Files.: PARTS.DBF, HISTORY.DBF

clear
set confirm on
set exact on
select c
use history
select b
use parts index c_ser     && indexed on comp_ser
select a
use comps index comp_ser,name_loc      && indexed on comp_ser
                             && indexed on last,first names
do while .t.
  usage = " "
  blank = space(15)
  mmod = blank
  description = space(50)
  mpart_ser = blank
  mptype = " "
  mpart_num = space(10)
  component = blank
mtoday = date()
mc_mod = blank
mc_mfg = blank
mc_desc = space(50)
mcptype = ""
mc_pnum = space(10)
lname = blank
fname = blank
entering = .t.
finished = .f.
delete_it = .f.
@ 0,20 say "PART DELETION SCREEN"
@ 1,0 to 1.79 double
@ 21,0 to 21,79 do while .not. finished & & deleting parts
entering = .t.
do while entering & & part information
find_use = .t.
do while find_use
located = .f.
* entries begin at column 18
   c = 18
   @ 3,0 clear to 20,79
   @ 2,56 say "date:" 
   @ 2,61 say mtoday
   @ 3,0 say "Enter Part"
* enter use or exit
   @ 4,2 say "current usage:" 
   @ 4,20 say "(Storage / Component)"
   set confirm off
   @ 4, c get usage picture "@N!"
   @ 22,25 say "To EXIT leave current usage blank"
   read
   do while .not. usage "$ sScC"
       usage = " "
       @ 4, c get usage picture "@N!"
       read
   enddo
set confirm on
if usage = " "
   set confirm off
   close databases
   release all
   return
else
   @ 22,0 clear to 23,79
endif
if upper(usage) = "S"
   @ 3,0 clear to 20,79
   locating = .f.
   located = .t.
   component = blank
   find_use = .f.
endif
if upper(usage) = "C"
   @ 4,0 clear to 20,79
   @ 3,0 say "Enter Component"
   @ 4,8 say "serial#:
   locating = .t.
endif
on_file = .t.
sought = .f.
do while locating
  @ 4,c get component picture "@N!"
  @ 22,0 clear to 23,79
  @ 22,16 say:
    "To ABANDON component entry leave serial # blank"
  read
  if component = blank
    locating = .f.
    usage = " "
    @ 22,0 clear to 23,79
  else
    @ 22,0 clear to 23,79
  * check to see if component on file
    select comps
    set order to 1
    goto top
  endif
  if .not. eof()
    seek trim(component)
    sought = .t.
  else
    on_file = .f.
  endif
  if found() .and. sought
    mc_mod = c_model
    mc_mfg = c_mfg
    mc_ptype = c_ptype
    mc_pnum = c_pnum
    lname = last_name
    fname = first_name
    @ 6,0 clear to 10,79
    @ 6,12 say "mfg: "
    @ 6,c say c_mfg
    @ 7,10 say "model: "
    @ 7,c say c_model
    @ 8,4 say "description: "
    @ 8,c say c_desc
    @ 9,2 say "property type: "
    do case
      case c_ptype = "P"
        @ 9,c say "Plant"
      case c_ptype = "M"
        @ 9,c say "Minor"
      otherwise
        @ 9,c say "Other"
    endcase
    if c_ptype = "M" .or. c_ptype = "P"
      @ 10,5 say "property #: "
      @ 10,c say c_pnum
    endif
  set confirm off
  ans = " "
do while .not. ans "$yYnN"
  ans = " "
  @ 22,0 clear to 23,79
  @ 22,22 say "Is this the correct component? : "
  @ 23,29 say "[ Yes / No ]"
  @ 22,53 get ans picture "@!A"
  read
endo
set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "Y"
  locating = .f.
located = .t.
find_use = .f.
else
  @ 22,11 say "The component shown is the only "+
  "one on file with this serial #"
  delay = 0
  do while delay < 40
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
  @ 22,18 say "Check the component"+
  " serial #, part usage, or"
  locating = .f.
  usage = "" "
  component= blank
  delay = 0
  do while delay < 40
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
endif

endif found

if (.not. found() .or. .not. on_file) .and. locating
  @ 22,0 clear to 23,79
  @ 22,26 say "Component not on file !!!"
  delay = 0
  do while delay < 40
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
  set confirm off
  ans = ""
  do while .not. ans "$YnN"
    ans = ""
    @ 22,22 say "Is this the correct serial #? : "
    @ 23,28 say "[ Yes / No ]"
    @ 22,52 get ans picture '@A'
    read
  enddo
  @ 22,0 clear to 23,79
  set confirm on
  if upper(ans) = "Y"
    @ 22,18 say "Check the component"+
    " serial #, part usage, or"
    delay = 0
    do while delay < 40
      delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
    usage = ""
    component= blank
    locating = .f.
  else
    @ 22,20 say "Please re-enter "+
    "the component serial # or"
    delay = 0
    do while delay < 40
      delay = delay + 1
    enddo
    component = blank
    @ 22,0 clear to 22,79
  endif
endif not found
endo locating
enddo find_use
checking = .t.
do while checking .and. located
   delete_it = .f.
   @ 3.0 clear to 20.79
   @ 3.0 say "Fill in if known:"
   @ 4.5 say "Part model:"
   @ 4.c get mmod picture "@N!" read
   @ 5.5 say "serial #:"
   @ 5.c get mpart_ser picture "@N!" read
   set confirm off
   @ 6.10 say "ptype: (Plant/Minor/Other)"
   @ 6.c get mptype picture "@N!" read
   do while .not. mptype $ " oOmMpP"
      mptype = ""
      @ 6.c get mptype picture "@N!" read
   enddo
   set confirm on
   @ 7.5 say "property #:"
   @ 7.c get mpnum picture "@N!" read
   rec = -1
   more = .f.
   select parts
   goto top
   done = .f.
   aa = .f.
   bb = .f.
   cc = .f.
   dd = .f.
   ee = .f.
   if mpart_ser <> blank .and. .not. done
      aa = .t.
      done = .t.
      locate for part_ser = trim(mpart_ser) .and.;
      comp_ser = trim(component)
      if found()
         rec = recno()
      endif
   endif
   if mmod <> blank .and. .not. done
      bb = .t.
      done = .t.
      locate for p_model = trim(mmod) .and.;
      comp_ser = trim(component)
      if found()
         rec = recno()
         continue
         if found()
            more = .t.
         endif
      endif
   endif
   if mptype <> "" .and. .not. done
      cc = .t.
      done = .t.
      locate for pptype = mptype .and.;
      comp_ser = trim(component)
      if found()
rec = recno()
continue
if found(
    more = .t.
endif
endif

if mpnum <> space(10) .and. .not. done
    cd = .t.
    done = .t.
locate for p_pnum = trim(mpnum) .and.:
    comp_ser = trim(component)
    if found(
        rec = recno()
        continue
        if found()
            more = .t.
        endif
    endif
endif
endif
if .not. done
    ee = .t.
    done = .t.
locate for comp_ser = trim(component)
    if found(
        rec = recno()
        continue
        if found()
            more = .t.
        endif
    endif
endif
endif
do case
case rec <> -1 .and. .not. more
    goto record rec
@ 3,0 clear to 20,79
@ 3,0 say "Only one part on file:"
@ 5,10 say "model: "
@ 5,c say p_model
@ 6,4 say "description: "
@ 6,c say p_desc
@ 7,2 say "property type: "
do case
    case p_ptype = "P"
        @ 7,c say "Plant"
    case p_ptype = "M"
        @ 7,c say "Minor"
    otherwise
        @ 7,c say "Other"
endcase
if p_ptype = "M" .or. p_ptype = "P"
    @ 8,5 say "property #: "
@ 8,c say p_pnum
endif
set confirm off
ans = " "
do while .not. ans $"YnN"
    ans = " "
    @ 22,26 say "Is this the correct part?: :
    @ 23,32 say "[ Yes / No ]"
    @ 22,52 get ans picture '@!A'
read
enddo
set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "N":  
  @ 22,16 say "Your PART cannot be found, and"+:  
" may not be on file"
  delay = 0
  do while delay < 40
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
  component = blank
  usage = ""
  checking = .f.
else
  & ans = yes
  delete_it = .t.
  checking = .f.
  @ 22,0 clear to 23,79
endif ans

case rec = -1
  @ 22,16 say "Your PART cannot be found, and"+;
  " may not be on file"
  delay = 0
  do while delay < 40
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
  component = blank
  usage = ""
  checking = .f.
  *exit checking

case rec <> -1 .and. more
  goto record rec
  still_more = .t.
  correct = .f.
  @ 3,0 clear to 20,79
  @ 4,0 say "More than one part on file:" 
  @ 4,0 say " (select the correct part)"
  do while still_more
    @ 5,c clear to 9,79
    @ 6,10 say "model: "
    @ 6,c say p_model
    @ 7,4 say "description: "
    @ 7,c say p_desc
    @ 8,2 say "property type: "
    @ 8,c clear to 11,79
    case pptype = ":P"
      @ 8,c say "Plant"
    case pptype = ":H"
      @ 8,c say "Minor"
    otherwise
      @ 8,c say "Other"
    endcase
    if pptype = ":M" .or. pptype = ":P"
      @ 9,5 say "property #:"
      @ 9,c say p_num
    endif
    set confirm off
    ans = ""
    do while .not. ans "$aAyYnN"
      ans = ""
      @ 22,26 say "Is this the correct part?: "
      @ 23,26 say "[ Yes / No / Abandon ]"
      @ 22,52 get ans picture "$!A"
      read
enddo
set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "A"
correct = .t.
still_more = .f.
component = blank
usage = ""
checking = .f.
endif
if eof()
still_more = .f.
checking = .f.
endif
if upper(ans) = "N" .and. .not. eof()
skip
do case
  case aa
    if part_ser = trim(mpart_ser) .and.:  
      comp_ser = trim(component)
      rec = recno()
      rec = recno()
      else
      still_more = .f.
      endif
  case bb
    if p_model = trim(mmod) .and.:  
      comp_ser = trim(component)
      rec = recno()
      else
      still_more = .f.
      endif
  case cc
    if p_type = mptype .and.:  
      comp_ser = trim(component)
      rec = recno()
      else
      still_more = .f.
      endif
  case dd
    if p_num = trim(mpnum) .and.:  
      comp_ser = trim(component)
      rec = recno()
      else
      still_more = .f.
      endif
  case ee
    if comp_ser = trim(component)
      rec = recno()
      else
      still_more = .f.
      endif
endcase
endif
if upper(ans) = "Y" .and. .not. eof()
delete_it = .t.
correct = .t.
still_more = .f.
checking = .f.
endif
enddo still_more
if .not. correct .and. .not. still_more
@ 3,0 clear to 11,79
@ 22,16 say "Your PART cannot be found, and"++;
delay = 0
do while delay < 50
  delay = delay + 1
endo
delay = 0
@ 22,0 clear to 22,79
component = blank
usage = ""
checking = .f.
endo

endcase
dendo checking
endif
endif
if delete_it
  @ 22,19 say "Do you wish to delete this PART?::
  @ 23,26 say "[ Yes / No ]"
  set confirm off
  ans = ""
  do while .not. ans "$yNn"
    ans = ""
    @ 22,52 get ans
    read
  enddo
  set confirm on
  @ 22,0 clear to 23,79
  if upper(ans) = "N"
    delete_it = .f.
  endif
endif

if located .and. .not. delete_it
  @ 22,16 say "Do you have additional parts to delete?:
  @ 23,26 say "[ Yes / No ]"
  set confirm off
  ans = ""
  do while .not. ans "$yNn"
    ans = ""
    @ 22,56 get ans
    read
  enddo
  set confirm on
  if upper(ans) = "N"
    set confirm off
    release all
    close databases
    return
  else
    component = blank
    usage = ""
    @ 22,0 clear to 23,79
  endif
endif
if delete_it
  @ 22,23 say "Standby while the part is deleted"
  select parts
goto rec
mptype = p_pctype
if upper(mptype) = "P" .or. upper(mptype) = "M"
  select history
  append blank
  replace model with trim(mmod)
  replace serial_num with trim(mpart_ser)
  replace ptype with mpt
  replace pnum with trimfmpnum
  replace del_date with mtoday
else
  component = blank
  usage = ""
  @ 22,0 clear to 23,79
endif }
endif
select parts
delete
@ 22.0 clear to 23.79
@ 22.16 say "Do you have additional parts to delete?: "
@ 23.26 say "[ Yes / No ]"
set confirm off
ans = " ".
@ 22.51 get ans
read
enddo
if upper(ans) = "N"
   entering = .f.
   finished = .t.
else
   @ 22.0 clear to 23.79
   usage = " ".
   mmod = blank
   description = space(50)
   mpart_ser = blank
   mptype = " ".
   mpnum = space(10)
   component = blank
   mtoday = date( )
   mc_mod = blank
   mc_mfg = blank
   mc_desc = space(50)
   mc_ptype = " ".
   mc_pnum = space(10)
   lname = blank
   fname = blank
   entering = .t.
   delete_it = .f.
   @ 4.0 clear to 20.79
endif set up
   && set up for the next one
endif delete it
endo entering
@ 8.15 say:
   "**** Standby while the files are updated ****"
endo finished
set confirm off
return
endo
* EOF delpart.prg

9. DEL_HELP.PRG

*********************** Program: DEL_HLP.PRG ***********************
*  Author ..................: TIM SEXTON
*  Purpose ..................: describes the options available to the
*                           user
*  Calls ..................: None

69
begin the text dialogue

* Deletions are made by either Component Serial # or by Part
* Correct serial number must be known, when requested, components are deleted by serial #
* When deleting a component, any parts assigned to that component must be deleted or re-assigned
* To delete a part, first identify if the desired part is presently in storage or assigned to a component

dextendtext

"CAUTIONS"
* Extremely important that all information is entered correctly
* Mistakes will be made so utilize the "NO" or "ABANDON" commands before resuming
* Recommend using DELETION and ARROW keys for modifying data entered and ensure not to deviate outside designated fields

dextendtext

"WARNINGS"
* Use of BACKSPACE key can cause PREMATURE exiting of an entry field, loop back for re-entrance of data
* Serial Numbers must be accurate and precise
* There is no recovery for deleted property, ensure you a certain you are deleting the proper item
**10. MAIN HELP.PRG**

********** Program: MAIN_HLP.PRG **********

*Author.............: TIM SEXTON
*Purpose............: describes the options available to the user
*Calls...............: None
*Input/Output Files.: None

* begin the text dialogue

程序: MAIN_HLP.PRG

作者: TIM SEXTON

目的: 描述可用给用户的选项

调用: 无

输入/输出文件: 无

* 开始文本会话

欢迎来到房地产管理帮助菜单

这个菜单是为不熟悉用户设计的，以帮助他们识别和理解使用此应用软件时遇到的困难区域。帮助菜单被分为五个方面，如下所示：

1. 名单和搜索
2. 房产报告打印
3. 输入房产
4. 删除房产
5. 修改房产

endtext

@ 21,0 to 21,79 double
Wait " -> press any key for more help or ESC to exit"

clear
@ 0.0 to 1,79 double
@ 0.23 say "LISTS AND SEARCHES"

text

* 设计用于为用户提供房产名单，并回答查询
* 查询用于识别所有者和所关心的地点

endtext

@ 21,0 to 21,79
Wait " -> press any key for more help or ESC to exit"

clear
@ 1.0 to 1,79 double
@ 0.17 say "PROPERTY REPORT PRINTING"

text

* 确保打印机已打开并准备好
* 报告内容可预格式化
* 提供两种类型的报告，如下所示：
  ** 季报——按看管人/所有者分组
  ** 概要——按房产类型和房产编号分组
Property is entered as a Component or a Part, parts are items used in a component (eg. card)
Mfg Serial Numbers is required for component entry
Part Model is required for part entry
Custodian and Location must be known for component entries, parts require a component serial # if not being placed into storage

Deletes current data by two methods:
1. Component
2. Part
Serial Numbers are inputted to identify component or part to be deleted from database
Ensure correct entry to avoid costly re-entrance

Modifications are made to components and parts changing accountability or to the record itself
Serial Numbers are required to access the data to be updated in the database
Modifications cannot be made until data has been entered
Once modifications are entered be sure to answer prompts correctly in order that new modified data be recorded into the database
11. MODCOMP.PRG

*************** Program: MODCOMP.PRG ***************
*Author...............: TIM SEXTON
*Purpose...............: allows the modification of the component
*record, with the exception of the fields to
*assign ownership. If the serial # is changed,
*then parts with that serial # are also changed
*accordingly
*Calls................: None
*Input/Output Files.: COMPS.DBF, PARTS.DBF

clear
set confirm on
select b
use parts index c_ser
select a
use comps index comp_ser,name_loc
do while .t.
  blank = space(15)
  mfg = blank
  mmod = blank
  mdesc = space(50)
  mser = blank
  mptype = " "
  mpnum = space(10)
  mprice = 0.00
  mreqn = blank
  mloc_code = " "
  mtoday = date( )
  mlname = blank
  mfname = blank
  finished = .f.

@ 0,16 say " MODIFY COMPONENT SCREEN"
@ 1,0 to 1,79 double
  do while .not. finished                 && modifying a component
    @ 21,0 to 21,79
    @ 2,56 say "date: "
    @ 2,61 say mtoday
    searched = .f.
    do while .not. searched
      * enter serial # or exit
      entering = .t.
      do while entering                 && component information
        @ 3,0 say "Enter Component:"   && entries begin at column 18
        c = 18
        @ 4,7 say "serial #: "
        @ 4,c get' mser PICTURE "@!"
        @ 22,28 say "To EXIT leave serial # blank"
        read
  @ 3,79 say 
    @ 70 say "MODIFY COMPONENT SCREEN"
  @ 79 double
if mser = blank
    set confirm off
    close databases
    release all
    return
else
    @ 22,0 clear to 23,79
endif

* check to see if on file
select comps
set order to 1
goto top
on_file = .f.
correct = .f.
rec = -1
if .not. eof()
    set exact on
    seek trim(mser)
else
    on_file = .f.
correct = .f.
endif
if found()
    rec = recno()
correct = .t.
    @ 3,0 clear to 20,79
    mmmod = c_model
    mmfg = c_mfg
    mdesc = c_desc
    mpctype = c_ptype
    mpnum = c_pnum
    mprice = c_price
    mregn = c_regn
    mlname = last_name
    mfname = first_name
    mlloc_code = loc_code
    @ 3,6 say "Custodian: "
    @ 3,c say trim(last_name) +", " + first_name
    @ 4,1 say "designated use: "
do case
    case loc_code = "0"
        @ 4,c say "Office"
    case loc_code = "H"
        @ 4,c say "Home"
    case loc_code = "I"
        @ 4,c say "Lab"
    otherwise
        @ 4,c say "Storage"
    endcase
    @ 6,12 say "mfg: "
    @ 6,c say mmfg
    @ 7,10 say "model: "
    @ 7,c say mmmod
    @ 8,7 say "serial #: "
    @ 8,c say mser
    @ 9,4 say "description: "
    @ 9,c say mdesc
    @ 11,2 say "property type: "
do case
    case c_ptype = "P"
        @ 11,c say "Plant"
    case c_ptype = "M"
        @ 11,c say "Minor"
    otherwise
        @ 11,c say "Other"
    endcase
if c_type = "W" .or. c_type = "P"
  @12.5 say "property #: "
  @12.5 say c_pnum
endif
@14.10 say "price: $"
@14.5 say c_price
@15.9 say "Rqsn #: "
@15.5 say c_rqsn PICTURE "@!R 9999-NNNN/NNNNNN"
set confirm off
ans = ""
12,10, say c.pnum
14,.c say c.price
15,9, say c_rqsn
set confirm off
ans = ""
do while .not. ans $"YnN"
  ans = "" 
  @22.25 say "Is this the correct component?: :"
  @23.33 say ":[ Yes / No ]"
  @22.55 get ans picture "$!A"
  read
endo
delay = 0 
do while delay < 40 
  delay = delay + 1
endo
@22.0 clear to 23.79
set confirm on
@endif
defound
if .not. found() .or. .not. correct
  @22.28 say "Component not on file !!!"
delay = 0 
do while delay < 40 
  delay = delay + 1
endo
@22.0 clear to 22.79
if upper(ans) = "Y"
  on_file = .t.
  searched = .t.
  entering = .f.
else
  on_file = .f.
  correct = .t.
@22.10 say:
  "The component shown is the only one "+; "on file with this serial #"
  delay = 0 
do while delay < 40 
  delay = delay + 1
endo
@22.0 clear to 23.79
@22.14 say "Your component maybe not on "+; "file, check the serial #"
  delay = 0 
do while delay < 40 
  delay = delay + 1
endo
@22.0 clear to 22.79
@3.0 clear to 20.79
endif
endif
set confirm off
ans = ""
do while .not. ans $"YnN"
  ans = "" 
  @22.25 say "Is this the correct serial #: :"
  @23.33 say "[ Yes / No ]"
  @22.55 get ans picture "$!A"
  read
endo
@22.0 clear to 23.79
set confirm on
if upper(ans) = "Y"
  correct = .t.
  on_file = .f.
@22.14 say "Your component maybe not on "+;
file, check the serial #

delay = 0
do while delay < 40
delay = delay + 1
enddo
@ 22,0 clear to 22,79
else
  correct = .f.
  @ 22,20 say;
      Please re-enter the component serial #
  delay = 0
  do while delay < 40
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
  mser = blank
endif
endif not found
if .not. on file .and. correct
  mser = blank
endif not on file and done
endo entering
endo searched
* allow editing the component record
new_mfg = blank
new_mod = blank
new_desc = space(50)
new_ser = blank
new_ptype = " 
new_pnum = space(10)
new_price = 0.00
new_reqn = blank
change_it = .t.
changed = .f.
do while change_it .and.;
  .not. changed   && component information
    @ 6,c clear to 20,79
    @ 22,16 say;
        "Make required changes, press RETURN when correct"
    new_mfg = mmfg
    @ 6,c get new_mfg PICTURE "@N!"
    read
    new_mod = mmod
    @ 7,c get new_mod PICTURE "@N!"
    read
* enter mfg serial# (mandatory)
  no_ser = .t.
do while no_ser
  new_ser = mser
  @ 8,c get new_ser PICTURE "@N!"
  read
  if new_ser = blank
    @ 22,0 clear to 22,79
    @ 22,24 say "serial # may not be blank"
    delay = 0
    do while delay < 25
      delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
    @ 22,16 say "Make required changes", press RETURN when correct"
  else
    no_ser = .f.
  endif
endo

enddo
new_desc = mdesc
@ 9,c get new_desc PICTURE "&N!"
read

* enter property type (mandatory)
set confirm off
@ 11,20 say "(Plant / Minor / Other)"
new_type = mptype
@ 11,c get new_type PICTURE "&A"
read
do while .not. new_type $ "mMoOpP"
new_type = mptype
@ 11,c get new_type PICTURE "&A"
read
endo
set confirm on

* no property# for other type property
if upper(new_type) = "O"
    @ 12,0 clear to 12,79
    new_pnum = space(10)
endif

* enter property# (mandatory for plant and minor property types)
if upper(new_type) = "M" .or. upper(new_type) = "P"
    @ 12,5 say "property #: ":
    no_num = .t.
do while no_num
    new_pnum = mpnum
    @ 12,c get new_pnum PICTURE "&N!"
    read
    if new_pnum = space(10)
        @ 22,0 clear to 22,79
        @ 22,12 say "Minor and Plant property ";
        "require a property number"
        delay = 0
        do while delay < 25
            delay = delay + 1
        enddo
        @ 22,0 clear to 22,79
        @ 22,16 say "Make required changes ";
        ", press RETURN when correct"
    else
        no_num = .f.
    endif
endo
endif

* enter price and requisition#
new_price = mprice
@ 14,c get new_price PICTURE "&R 99,999.99"
read
new_reqn = mreqn
@ 15,c get new_reqn PICTURE "&R 9999-NNNN/NNNNN"
read
set confirm off
@ 22,0 clear to 22,79
ans = ""
do while .not. ans $ "aAyYnN"
    ans = ""
    @ 22,25 say "Are the modifications correct?: ":
    @ 23,29 say "[ Yes / No / Abandon]"
    @ 22,56 get ans picture "&A"
    read
endo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "A"
change_it = .f.
endif
if upper(ans) = "Y"        && add mvar to dbf
changed = .t.
endif
enddo change
* if original record was changed replace the old fields
* with the new ones
if changed
set confirm off
@ 22,0 clear to 22,79
ans = " "
do while .not. ans $"YnN"
ans = " "  
@ 22,18 say:
   "Do you wish to file the modified component? : "
@ 23,29 say "[ Yes / No ]"
@ 22,62 get ans picture "@!A"
read
endo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "N"
changed = .f.
endif
if upper(ans) = "Y"        && add mvar to dbf
@ 22,20 say "Standby while your entry is placed on file"
* place in comps
   select comps
   goto rec
   replace c_mfg with trim(new_mfg)
   replace c_model with trim(new_mod)
   replace c_desc with trim(new_desc)
   replace comp_ser with trim(new_ser)
   replace c_type with trim(new_type)
   replace c_pnum with trim(new_pnum)
   replace c_price with new_price
   replace c_reqn with trim(new_reqn)
   changed = .f.
* replace comp_ser in parts file
   select parts
   goto top
   set exact on
   replace comp_ser with trim(new_ser) for;
   comp_ser = trim(mser)
endif
endif changed
if .not. changed
@ 22,0 clear to 23,79
@ 22,14 say "Do you have additional component"+
   " records to modify?: "
@ 23,34 say "[ Yes / No ]"
set confirm off
ans = " "
do while .not. ans $"YnN"
ans = " "
@ 22,66 get ans
read
endo
if upper(ans) = "N"
   finished = .t.
clear
else        && set up for the next one
    78
MODLOC.PRG

***************
Program: MODLOC.PRG ***************
* Author.............: TIM SEXTON
* Purpose............: allows reassigning a component to a new
* custodian if the custodian is not on file a
* new record is created in owners and or homes,
* if the owner is not on file for any
* components any longer then they are deleted
* Calls...............: None
* Input/Output Files.: COMPS.DBF, OWNERS.DBF, HOMES.DBF

clear
set confirm on
set exact off
select c
use homes index l_fnames && indexed on last,first names
select b
use owners index names && indexed on last,first names
select a
use comps index comp_ser,name_loc && indexed on comp_ser
do while .t.
    blank = space(15)
    mmfg = blank
    mmod = blank
    mdesc = space(50)
    mser = blank
    mptype = " "
    mpnum = space(10)
endif
endif not changed
endo finished
set confirm off
release all
close databases
return
endo
* EOF modcomp.prg
mloc_code = "  
mtoday = date( )
mlname = blank
mfname = blank
mlocation = space(8)
mstreet = space(25)
cmp = blank
mphone = space(13)
addhome = .f.
addowner = .f.
finished = .f.

@ 0,6 say ;
"MODIFY COMPONENT ASSIGNMENT SCREEN"
@ 1,0 to 1,79 double
do while .not. finished  && modifying a component
  @ 21.0 to 21,79
  @ 2.56 say "date: 
  @ 2.61 say mtoday
  searched = .f.
  do while .not. searched
* enter serial # or exit
  entering = .t.
  do while entering  && component information
    @ 3,0 say "Enter Component:
    c = 18
    @ 4.7 say "serial #: 
    @ 4,c get mser PICTURE "@N!"
    @ 22,28 say "To EXIT leave serial # blank"
    read
    if mser = blank
      set confirm off
      close databases
      release all
      return
    else
      @ 22,0 clear to 23,79
      endif
* check to see if on file
  select comps
  set order to 1
  goto top
  on_file = .f.
  correct = .f.
  c_rec = -1
  if .not. eof()
    set exact on
    seek trim(mser)
    set exact off
  else
    on_file = .f.
    correct = .f.
  endif
  if found()
    c_rec = recno()
    correct = .t.
  @ 3,0 clear to 20,79
  mmod = c_model
  mmfg = c_mfg
mdesc = c_desc
mptype = cptype
mpnum = cphnum
mlname = last_name
mfname = first_name
mloc_code = loc_code
if mloc_code = "O"
select owners
locate for last_name = trim(mlname) .and.;
   first_name = trim(mfname) .and.;
   location <> "HOME"
   mlocation = owners->location
endif
if mloc_code = "H"
select homes
set relation to last_name+first_name into owners
locate for last_name = trim(mlname) .and.;
   first_name = trim(mfname) .and.;
   owners->location = "HOME"
   mlocation = owners->location
   mstreet = homes->street
   mcity = homes->city
   mphone = homes->phone
   set relation to
endif
@ 3,12 say "mfg: \\
@ 3,c say mmfg
@ 4,10 say "model: \\
@ 4,c say mmmd
@ 5,7 say "serial #: \\
@ 5,c say mser
@ 6,4 say "description: \\
@ 6,c say mdesc
@ 8,2 say "property type: \\
do case
   case mptype = "P"
      @ 8,c say "Plant"
   case mptype = "H"
      @ 8,c say "Minor"
   otherwise
      @ 8,c say "Other"
endcase
if mptype = "H" .or. mptype = "P"
   @ 9,5 say "property #: \\
   @ 9,c say mpnum
endif
@ 11,1 say "designated use: \\
do case
   case mloc_code = "0"
      @ 11,c say "Office"
   case mloc_code = "H"
      @ 11,c say "Home"
   case mloc_code = "L"
      @ 11,c say "Lab"
   otherwise
      @ 11,c say "Storage"
endcase
@ 13,6 say "Custodian: \\
do case
   case mloc_code = "L"
      @ 13,c say mlname
      @ 14,12 say "lab: \\
      @ 14,c say mfname
   case mloc_code = "0"
      @ 13,c say trim(last_name) +"", " + first_name
@endif
@ 14.9 say "office: "
@ 14,c say mllocation

    case mloc_code = "H"
        @ 13,c say trim(last_name) "", " + first_name
        @ 14,9 say "street: "
        @ 14,c say mstreet
        @ 15,11 say "city: "
        @ 15,c say mcity
        @ 16,10 say "phone: "
        @ 16,c say mphone
    endcase

    case mloc_code = "S"
        @ 13,c say "AS DEPT"
    endcase

set confirm off
ans = " "
do while .not. ans "$yNnY"
    ans = " "
    @ 22.25 say "Is this the correct component?: :"
    @ 23.33 say "/ [Yes / No ]"
    @ 22.56 get ans picture "$A" read
endo

set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "Y"
    on_file = .t.
    searched = .t.
    entering = .f.
else
    on_file = .f.
    correct = .t.
    @ 22.10 say;
        "The component shown is the only one "+
        "on file with this serial "."

delay = 0
    do while delay < 40
        delay = delay + 1
    enddo
    @ 22.0 clear to 23,79
    @ 22.14 say "Your component maybe not on "+
        "file, check the serial "."

delay = 0
    do while delay < 40
        delay = delay + 1
    enddo
    @ 22.0 clear to 22,79
    @ 3.0 clear to 20,79
endo

endif found
if .not. found() .or. .not. correct
    @ 22.28 say "Component not on file !!!"
    delay = 0
    do while delay < 40
        delay = delay + 1
    enddo
    @ 22.0 clear to 22,79
    set confirm off
    ans = " "
do while .not. ans "$yNnY"
    ans = " "
    @ 22.25 say "Is this the correct serial #: :"
    @ 23.33 say "/ [Yes / No ]"
    @ 22.55 get ans picture "$A" read
endo
@ 22.0 clear to 23,79

set confirm on
if upper(ans) = "Y"
correct = .t.
on_file = .f.
@ 22,14 say "Your component maybe not on "+
"file, check the serial #"
delay = 0
do while delay < 40
delay = delay + 1
endo
@ 22,0 clear to 22,79
else
correct = .f.
@ 22,20 say;
"Please re-enter the component serial #"
delay = 0
do while delay < 40
delay = delay + 1
do
@ 22,0 clear to 22.79
mser = blank
endif
endif not found
if .not. on_file .and. correct
mser = blank
endif not on file and done
endo entering
endo searched
* allow reassigning the component
addhome = .f.
addowner = .f.
change_it = .t.
changed = .f.
do while change_it .and.;
   .not. changed && component information
   @ 11,c clear to 11,79
   @ 12,0 clear to 20,79
   @ 22,26 say "Enter component new assignment"
* enter location code
new_code = ""
set confirm off
@ 11,20 say "(Office / Lab / Storage / Home)"
@ 11,c get new_code PICTURE "@!A"
read
do while .not. new_code "$OoLlSsHh"
   new_code = ""
   @ 11,c get new_code PICTURE "@!A"
   read
endo
set confirm on
addhome = .f.
addowner = .f.
newiname = blank
newfname = blank
newloc = space(8)
newstreet = space(25)
ewcity = blank
newphone = space(13)

* enter custodian information, search to see if on file
* use owners for loc_codes 0,5,L
* use homes for loc_code H
* last name, first name, and office or home addresses are mandatory

* allow reassigning the component
addhome = .f.
addowner = .f.
newiname = blank
newfname = blank
newloc = space(8)
newstreet = space(25)
ewcity = blank
newphone = space(13)
* office use

case new_code = "0"
    @ 22.0 clear to 22.79
    @ 22.26 say "Enter component new Custodian"
    no_lname = .t.
    do while no_lname
        @ 13.0 clear to 20.79
        @ 13.6 say "Custodian"
        @ 14.6 say "last name: "
        @ 14.c get new_lname PICTURE "?A"
        read
        if new_lname = blank
            @ 22.0 clear to 22.79
            @ 22.24 say "Custodian's name may not be blank"
            delay = 0
            do while delay < 25
                delay = delay + 1
            enddo
        enddo
    else
        no_lname = .f.
    endif
dendo no lname

* check to see if on file

select owners
    goto top
    searched = .f.
    checked = .f.
    located = .f.
    do while .not. searched
        if eof()
            searched = .t.
        else
            seek trim(new_lname)
        endif
        if .not. found()
            searched = .t.
            addowner = .t.
        endif
        if found() and location = "HOME"
            searched = .t.
            located = .t.
            checked = .t.
        endif
        if found() and location <> "HOME"
            searched = .t.
            located = .t.
            checked = .f.
        endif
    enddo
endo

do while located
    do while .not. checked
        @ 15.5 say "first name: "
        @ 15.c say first_name
        @ 16.9 say "office: "
        @ 16.c say location
        set confirm off
        ans = " 
        do while .not. ans "$yYnN"
            ans = " "
        @ 22.0 clear to 22.79
        @ 22.20 say "Is this the correct?"+
            " custodian?:
        @ 23.26 say "[ Yes / No ]"
        @ 22.51 get ans picture '@A"

dendo no lname

read
endo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "y"
    located = .f.
    checked = .t.
    new_fname = first_name
    new_loc = location
else
    checked = .t.
    @ 15,c clear to 16,79
endif
endo checked
endo

do while located .and. checked .and. .not. eof() skip
if eof()
    located = .f.
    addowner = .t.
endif
if last_name = trim(new_lname)
    checked = .f.
else
    located = .f.
    addowner = .t.
endif
if location = "HOME"
    located = .t.
    checked = .t.
endif
endo
located and checked
endo
located
if addowner

    no_fname = .t.
    do while no_fname
        @ 15,5 say "first name: "
        @ 15,c get new_fname picture "@!A"
        read
        if new_fname = space(15)
            @ 22,0 clear to 22,79
            @ 22,20 say "A first name or first";
            " initial is required"
            delay = 0
            do while delay < 25
                delay = delay + 1
            enddo
            @ 22,0 clear to 22,79
        else
            no_fname = .f.
        endif
    enddo
    no_fname
    @ 16,9 say "office: "
    @ 16,27 say "((bldg-room)"
    no_office = .t.
    do while no_office
        @ 16,c get new_loc picture "@! (A-999)"
        read
        if new_loc = space(8)
            @ 22,0 clear to 22,79
            @ 22,26 say "Office may not be blank"
            delay = 0
            do while delay < 25
                delay = delay + 1
            enddo
            @ 22,0 clear to 22,79
        else
            no_office = .f.
        endif
endo
no_office

else
    no_office = .f.
endif
endo no office
endif add owner

* supply use
    case new_code = "S"
        @ 22.0 clear to 22.79
        @ 13.0 clear to 20.79
        @ 13.6 say "custodian: AS DEPT"
        @ 14.9 say "location: (I-200)"
        new_lname = "AS DEPT"
        new_fname = "(I-200)"
        new_loc = "STORAGE"

* check to see if on file
    select owners
    goto top
    if eof()
        addowner = .t.
    else
        locate for last_name = trim(new_lname) .and.;
        first_name = trim(new_fname) .and.;
        location = new_loc
    endif
    if .not. found()
        addowner = .t.
    endif

* lab use
    case new_code = "L"
        @ 22.0 clear to 22.79
        @ 22.26 say "Enter component new assignment"
        @ 13.0 clear to 20.79
        @ 13.5 say "A - (I-158) Front"
        @ 14.5 say "B - (I-158) Back"
        @ 16.3 say "Enter one of the above lab locations : :"
        set confirm off
        lab = ""
        do while .not. lab $ "AaBbCcDd"
            lab = ""
            @ 16.41 get lab picture "@!A"
            read
        enddo
        set confirm on
        new_lname = "AS DEPT"
        new_loc = "LAB"
        do case
            case upper(lab) = "A"
                new_fname = "(I-158)F"
            case upper(lab) = "B"
                new_fname = "(I-158)B"
            case upper(lab) = "C"
                new_fname = "(I-224)"
            case upper(lab) = "D"
                new_fname = "(I-250)"
        endcase
    * check to see if on file
        select owners
        goto top

86
if eof()
    addowner = .t.
else
    locate for last_name = trim(new lname) .and.;
    first_name = trim(new fname) .and.;
    location = new loc
endif
if .not. found()
    addowner = .t.
endif

* home use

case new_code = "H"
@ 22.0 clear to 22.79
@ 22.26 say "Enter component new assignment"
new_loc = "HOME"
no lname = .t.
do while no lname
    @ 13.0 clear to 20.79
    @ 13.6 say "Custodian"
    @ 14.6 say "last name: "
    @ 14.c get new lname PICTURE "@!A"
    read
    if new lname = blank
        @ 22.0 clear to 22.79
        @ 22.24 say "Custodian's name may not be blank"
        delay = 0
        do while delay < 25
            delay = delay + 1
        enddo
        @ 22.0 clear to 22.79
    else
        no lname = .f.
    endif
endo no lname
* check to see if on file
select homes
goto top
searched = .f.
located = .f.
checked = .f.
do while .not. searched
    do while .not. checked
        @ 15.5 say "first name: "
        @ 15.c say first fname
        @ 16.9 say "street: "
        @ 16.c say street
        @ 17.11 say "city: "
        @ 17.c say city
    enddo
endo located
do while located
    do while .not. checked
    @ 15.5 say "first name: "
    @ 15.c say first fname
    @ 16.9 say "street: "
    @ 16.c say street
    @ 17.11 say "city: "
    @ 17.c say city
endo located
endo not searched
@ 18,10 say "phone:  
@ 18,c say phone
set confirm off
@ 22,0 clear to 22,79
ans = " 
do while .not. ans "$YnN"
    ans = " 
    @ 22,20 say "Is this the correct"+;
    if custodian?; ":"
    @ 23,26 say "[ Yes / No ]"
    @ 22,51 get ans picture "@!A" read
enddo
set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "Y"
    located = .t.
    checked = .t.
    new_fname = first-name
    new_street = street
    new_city = city
    new_phone = phone
else
    checked = .t.
@ 15,c clear to 20,79
endif
enddo checked
endo while located .and. checked .and. .not. eof()
skip
if eof()
    located = .f.
    addowner = .t.
    addhome = .t.
endif
if last_name = new_lname
    checked = .f.
else
    located = .f.
    addowner = .t.
    addhome = .t.
endif
enddo located and checked
endo while located
if addowner
    no_fname = .t.
do while no_fname
    @ 15,5 say "first name: 
    @ 15,c get new_fname picture "@!A" read
    if new_fname = blank
        @ 22,0 clear to 22,79
        @ 22,20 say "A first name or first"+; 
        " initial is required"
        delay = 0
        do while delay < 25
            delay = delay + 1
        enddo
        @ 22,0 clear to 22,79
    else
        no_fname = .f.
    endif
endo no_fname
endo no address
no_address = .t.
do while no_address
    @ 16,9 say "street: 
    @ 16,c get new_street PICTURE "@!"

88
@ 17,11  say "city: 
@ 17,c  get new_city PICTURE "@!A"  
@ 18,10  say "phone: "  
@ 18,c  get new_phone PICTURE "(999)999-9999"
read
if new_street = space(25) .or.:    
  new_city = blank
  @ 22,0 clear to 22,79
  @ 22,24 say "Street or City may not be blank"
  delay = 0
  do while delay < 25
    delay = delay + 1
  enddo
  @ 22,0 clear to 22,79
else
  no_address = .f.
endif
enddo
no_address
endif
addowner
endcase
location code
set confirm off
@ 22,0 clear to 22,79
ans = " "
do while .not. ans "$aAyYnN"
  ans = " "
  @ 22,25 say "Are the modifications correct?: ":
  @ 23,29 say "[ Yes / No / Abandon]"
  @ 22,56 get ans picture "@!A"
read
enddo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "A"
  changed = .f.
endif
if upper(ans) = "Y"  
  & add mvar to dbf
  changed = .t.
endif
enddo
change
* if changed replace old fields with new fields
if changed
  set confirm off
  @ 22,0 clear to 22,79
  ans = " "
do while .not. ans "$YnnN"
  ans = " "
  @ 22,18 say "Do you wish to file the " +
  "modified component?: ":
  @ 23,29 say "[ Yes / No ]"
  @ 22,62 get ans picture "@!A"
read
enddo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "N"
  changed = .f.
endif
if upper(ans) = "Y"  
  & add mvar to dbf
  @ 22,20 say "Standby while your entry is placed on file"
* place in comps
  select comps
  goto c_rec
  replace loc_code with new_code
  89
replace last-name with trim(new lname)
replace first-name with trim(new_fname)
replace issue-date with mtoday
* check and see if old owner needs to deleted from owners or homes
  goto top
  del_owner = .f.
del_home = .f.
  if mloc_code = "0" .or. mloc_code = "H"
    locate for last_name = trim(mlname) .and.;
      first_name = trim(mfname) .and.;
        loc_code = mloc_code
      if .not. found()
        del_owner = .t.
        if mloc_code = "H"
          del_home = .t.
        endif
      endif
    endif
  * if the old custodian is not in the comp file
  * delete them from owners
  if del_owner
  select owners
goto top
  locate for last_name = trim(mlname) .and.;
    first_name = trim(mfname) .and.;
      location = mlocation
  delete
  endif
  * if the old custodian is not in the comp file, delete them from
  * homes if the old loc code was HOME
  if del_home
  select homes
  locate for last_name = trim(mlname) .and.;
    first_name = trim(mfname) .and.;
      street = trim(mstreet)
  delete
  endif
  * place in owners if new owner not on file
  if addowner
  select owners
  append blank
  replace last_name with trim(new lname)
  replace first_name with trim(new_fname)
  replace location with new_loc
  endif
  * place in homes if new owner not on file and loc code = HOME
  if addhome
  select homes
  append blank
  replace last_name with trim(new lname)
  replace first_name with trim(new_fname)
  replace street with trim(new_street)
  replace city with trim(new_city)
  replace phone with new_phone
  endif
  changed = .f.
endif
endif changed
if .not. changed
@ 22,0 clear to 23,79
@ 22,14 say "Do you have additional component"+;
set confirm off
ans = ""
do while .not. ans $ "YnN"
   ans = ""
   @ 22,66 get ans
   read
endo
if upper(ans) = "N"
   finished = .t.
   clear
else
   @ 22,0 clear to 23,79
   entering = .t.
   mmfg = blank
   mmid = blank
   mdesc = space(50)
   mser = blank
   mtype = ""
   mnum = space(10)
   mloc_code = ""
   mdate = date( )
   miname = blank
   mfname = blank
   mlocation = space(8)
   mstreet = space(25)
   mcity = blank
   mphone = space(13)
   addhome = .f.
   addowner = .f.
   @ 3,0 clear to 20,79
endif
endif not changed
endo finished
select owners   && pack files
pack
select homes
pack
set confirm off
release all
close databases
return
endo
* EOF modloc.prg

13. MODMENU.PRG

*************** Program: MODMENU.PRG ***************
*Author................: TIM SEXTON
*Purpose................: displays the choices to modify a component
or part record, also allow re-assigning the
*custodians
*Calls................: MODCOMP.PRG, MODLOC.PRG, MODPART.PRG
*Input/Output Files.: None

* set up the screen environment
clear
set confirm off
* begin the menu dialogue  
do while .t.
  clear
  @ 2,10 to 17,69 double
  @ 3,29 say "Modify Property Menu"
  @ 4,11 to 4,68 double
  @ 6,27 say " Modify COMPONENT"
  @ 7,27 say " 1 - Custodian or Location"
  @ 8,27 say " 2 - Record"
  @ 10,27 say " Modify PART"
  @ 11,27 say " 3 - Accountability or Record"
  @ 13,27 say " H - HELP"
  @ 15,27 say " 0 - RETURN to MAIN MENU"
  @ 17,30 say " selection : : ":
  choice = " "
  @ 17,42 get choice  
  read
* place an asterisk next to a valid choice and erase the other  
* rows leaving the heading Part or Component
  line = 5
  if choice "$hH0123"
  do case
    case upper(choice) = "H"
      @ 13,26 say "**"
      choicerow = 13
    case choice = "0"
      @ 15,26 say "**"
      choicerow = 15
    case choice "$12"
      @ 8+val(choice),26 say "*":
      line = 8
      choicerow = 8+val(choice)
    otherwise
      @ 8+val(choice),26 say "*":
      line = 10
      choicerow = 8+val(choice)
  endcase
  firstrow = 6
  rows = 11
  rowcnt = 0
  do while rowcnt < rows
    if (rowcnt+firstrow <> choicerow)
      if (rowcnt+firstrow <> line)
        @ firstrow+rowcnt,27 say space(30)
        endif
      endif
      rowcnt = rowcnt + 1
    enddo
  endif
* do a valid choice or loop back thru this program
  do case
    case upper(choice) = "H"
      do mod_help
    case choice = "0"
      do modloc
    case choice = "1"
      do modcomp
    case choice = "2"
      do modpart
    case choice = "3"
      do modpart
    case else
      "invalid choice"
  endcase
otherwise
@ 20,22 say "****** not a valid selection ******"
?
wait
loop
endcase
enddo
* EOF MODMENU.PRG

14. MODPART.PRG

*************** Program: MODPART.PRG ***************
*Author............: TIM SEATON
*Purpose............: allows the modification of the part record,
*                    if reassigned a different component the
*                    component is checked to see if on file
*                    None
*Input/Output Files.: COMPS.DBF, PARTS.DBF

clear
set confirm on
set exact on
select a
use comps index comp_ser,name_loc    && indexed on comp_ser
select b
use parts index c_ser    && indexed on comp_ser
    && indexed on last,first names
set relation to comp_ser into comps
do while .t.
    usage = " "
    blank = space(15)
    mmod = blank
    description = space(50)
    mpart_ser = blank
    mptype = " "
    mptnum = space(10)
    component = blank
    mtoday = date( )
    mc_mod = blank
    mc_mfg = blank
    mc_desc = space(50)
    mc_ptype = " "
    mc_ptnum = space(10)
    lname = blank
    fname = blank
    known = .f.
    finished = .f.
    change_it = .f.
    do while .not. finished    && modifying parts
        @ 21,0 to 21,79
        @ 0,20 say " MODIFY PART SCREEN"
        @ 1,0 to 1,79 double
        entering = .t.
        do while entering
            && part information
            find_use = .t.
            do while find_use
                located = .f.

93
* entries begin at column 18
  c = 18
  @ 3.0 clear to 20.79
  @ 2.56 say "date:"
  @ 2.61 say mtoday
  @ 3.0 say "Enter Part"
  
  * enter use or exit
  @ 4.2 say "current usage:"
  @ 4.20 say "(Storage / Component / ? not known)"
  set confirm off
  @ 4.c get usage picture "@N!"
  @ 22.25 say "To EXIT leave current usage blank"
  read
  do while .not. usage "$ sScC?"
    usage = ""
    @ 4.c get usage picture "@N!"
    read
  enddo
  set confirm on
  if usage = ""
    set confirm off
    close databases
    release all
    return
  else
    @ 22.0 clear to 23.79
  endif
  if upper(usage) = "?"
    @ 3.0 clear to 20.79
    known = .f.
    locating = .f.
    located = .t.
    component = blank
    find_use = .f.
  endif
  if upper(usage) = "S"
    @ 3.0 clear to 20.79
    known = .t.
    locating = .f.
    located = .t.
    component = blank
    find_use = .f.
  endif
  if upper(usage) = "C"
    known = .t.
    @ 4.0 clear to 20.79
    @ 3.0 say "Enter Component"
    @ 4.8 say "serial#:"
    locating = .t.
  endif
  on_file = .t.
  sought = .f.
  do while locating
    @ 4.c get component picture "@N!"
    @ 22.0 clear to 23.79
    @ 22.16 say:
    "To ABANDON component entry leave serial # blank"
    read
    if component = blank
      locating = .f.
      usage = ""
    else
      @ 22.0 clear to 23.79
    endif
**check to see if component on file**

```
select comps
set order to 1
goto top
endif
if .not. eof()
    seek trim(component)
sought = .t.
else
    on_file = .f.
endif
if found() .and. sought
    mc_mod = c_model
    mc_mfg = c_mfg
    mc_ptype = c_ptype
    mc_pnum = c_pnum
    lname = last_name
    fname = first_name
    @ 6,0 clear to 10,79
    @ 6,12 say "mfg: "
    @ 6,c say c_mfg
    @ 7,10 say "model: "
    @ 7,c say c_model
    @ 8,4 say "description: "
    @ 8,c say c_desc
    @ 9,2 say "property type: "
do case
    case c_ptype = "P"
        @ 9,c say "Plant"
    case c_ptype = "M"
        @ 9,c say "Minor"
    otherwise
        @ 9,c say "Other"
endcase
if c_ptype = "M" .or. c_ptype = "P"
    @ 10,5 say "property #: "
    @ 10,c say c_pnum
endif
set confirm off
ans = " ">
    do while .not. ans "$yYnN"
        ans = " ">
        @ 22,0 clear to 23,79
        @ 22,22 say "IS THIS the correct component?"
        @ 23,29 say ", Yes / No ]"
        @ 22,53 get ans picture '@!A'
        read
    enddo
set confirm on
@ 22,0 clear to 23,79
if upper(ans) = "Y"
    locating = .f.
    located = .t.
    find_use = .f.
else
    @ 22,11 say "The..."
    delay = 1
do while delay = 1
        delay = 2
        enddo
```

```
component = blank

delay = 0
do while delay < 40
delay = delay + 1
endo
@ 22,0 clear to 22,79
endif

endif found
if (.not. found) .or. .not. on-file) .and. locating
@ 22,0 clear to 23,79
@ 22,26 say "Component not on file !!!"
delay = 0
do while delay < 40
delay = delay + 1
endo
@ 22,0 clear to 22,79
set confirm off
ans = " "
do while .not. ans "$YnN"

ans = " "
@ 22,22 say "Is this the correct serial #: :"
@ 23,28 say "[ Yes / No ]"
@ 22,52 get ans picture '@A'
read
endo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "Y"
@ 22,18 say "Check the component";
" serial #, part usage, or"
delay = 0
do while delay < 40
delay = delay + 1
endo
@ 22,0 clear to 22,79
usage = " "
component = blank
locating = .f.
else
@ 22,20 say "Please re-enter ";
"the component serial # or"
delay = 0
do while delay < 40
delay = delay + 1
endo
component = blank
@ 22,0 clear to 22,79
endif
endif not found
endo locating
endo find_use

checking = .t.
do while checking .and. located
mmod = blank
mpart_ser = blank
mtype = " "
mpnum = space(10)
change_it = .f.
@ 3,0 clear to 20,79
@ 3,0 say "Fill in if known:"
@ 4,5 say "Part model: "

96
@ 4,c get mmmod picture "@N!":  read
@ 5,7 say "serial #: "
@ 5,c get mpart_ser picture "@N!":  read

set confirm off
@ 6,10 say "ptype: (Plant/Minor/Other)"
@ 6,c get mptype picture "@N!":  read

do while .not. mptype "$ oOmMpP"
    mptype = " "
    @ 6,c get mptype picture "@N!":  read
endo
d
set confirm on
@ 7,5 say "property #: "
@ 7,c get mpnum picture "@N!":  read

rec = -1
more = .f.
select parts
goto top
done = .f.
aa = .f.
bb = .f.
cc = .f.
 dd = .f.
 ee = .f.

if mpart_ser <> blank .and. .not. done
    aa = .t.
done = .t.
    if known
        locate for part_ser = trim(mpart_ser) .and.;
        comp_ser = trim(component)
    else
        locate for part_ser = trim(mpart_ser)
    endif
    if found()
        rec = recno()
    endif
endif

if mmmod <> blank .and. .not. done
    bb = .t.
done = .t.
    if known
        locate for p_model = trim(mmmod) .and.;
        comp_ser = trim(component)
    else
        locate for p_model = trim(mmmod)
    endif
    if found()
        rec = recno()
        continue
    endif
endif

if mftype <> " " .and. .not. done
    cc = .t.
done = .t.
    if known
        locate for p_pftype = mftype .and.;
else
  locate for p_pertype = mptype
endif
if found()
  rec = recno()
  continue
  if found()
    more = .t.
  endif
endif
endif
if mpnum <> blank .and. .not. done
  dd = .t.
  done = .t.
  if known
    locate for p_pnum = mpnum .and.;
        comp_ser = trim(component)
  else
    locate for p_pnum = trim(mpnum)
  endif
if found()
  rec = recno()
  continue
  if found()
    more = .t.
  endif
endif
endif
if .not. done
  ee = .t.
  done = .t.
  if known
    locate for comp_ser = trim(component)
      if found()
        rec = recno()
        continue
        if found()
          more = .t.
        endif
      endif
    else
      if .not. eof()
        rec = recno()
        skip
        if .not. eof()
          more = .t.
        endif
      endif
    endif
  endif
endif
do case
do case rec <> -1 .and. .not. more
goto record rec
  @ 3,0 clear to 20,79
  @ 3,0 say "Only one part on file:"
  @ 4,10 say "model: "
  @ 4,c say p_model
  @ 5,7 say "serial #: "
  @ 5,c say part_ser
  @ 6,4 say "description: "
  @ 6,c say p_desc
  @ 7,2 say "property type: "
do case
do case p_pertype = "P"
  @ 7,c say "Plant"
case p_p_type = "M".
    @ 7. c say "Minor"
    otherwise
    @ 7. c say "Other"
endcase
if p_p_type = "M" .or. p_p_type = "P"
    @ 8. 3 say "property #: ";
    @ 8. c say p_p_num
endif
if .not. known
do case
    case cop_ser = blank
        @ 10. 4 say "current use: STORAGE"
    case comp_ser <> blank
        @ 10. 4 say "current use: COMPONENT"
        @ 11. 6 say "custodian: "
        @ 11. c say comps->last_name +;
            comps-> first_name
        @ 13. 12 say "mfg: "
        @ 13. c say comps-> c_mfg
        @ 14. 10 say "model: "
        @ 14. c say comps-> c_model
        @ 14. 7 say "serial #: "
        @ 14. c say comps->comp_ser
        @ 15. 4 say "description: "
        @ 15. c say comps-> c_desc
    endcase
endif
set confirm off
ans = it1
do while .not. ans "$yYN"
    ans = " "
    @ 22. 26 say "Is this the correct part?: :"
    @ 23. 32 say "[ Yes / No ]"
    @ 22. 52 get ans picture "@!A"
read
set confirm on
@ 22. 0 clear to 23. 79
if upper(ans) = "N"
    @ 22. 16 say "Your PART cannot be found, and"+
    " may not be on file"
    delay = 0
    do while delay < 40
        delay = delay + 1
    enddo
    @ 22. 0 clear to 22. 79
    component = blank
    usage = " "
    checking = .f.
else
    & ans = yes
    change_it = .t.
    checking = .f.
    @ 22. 0 clear to 23. 79
endif
endf ans

case rec = -1
    @ 22. 16 say "Your PART cannot be found, and"+
    " may not be on file"
    delay = 0
    do while delay < 40
        delay = delay + 1
    enddo
    @ 22. 0 clear to 22. 79
    component = blank
usage = " "
checking = .f.
case rec <> -1 and. more
goto rec
still_more = .t.
correct = .f.
@ 3,0 clear to 20,79
@ 3,0 say "More than one part on file;"
@ 4,0 say " (select the correct part)"
do while still_more
@ 5,c clear to 20,79
@ 5,10 say "model:"
@ 5,c say p_model
@ 6,7 say "serial #:"
@ 6,c say part_ser
@ 7,4 say "description:"
@ 7,c say p_desc
@ 8,2 say "property type:"
do case
  case p_type = "P"
    @ 8,c say "Plant"
  case p_type = "M"
    @ 8,c say "Minor"
  otherwise
    @ 8,c say "Other"
@ 9,0 clear to 9,79
endcase
if p_type = "M" .or. p_type = "P"
  @ 9,5 say "property #:"
  @ 9,c say p_num
endif
if .not. known
  do case
    case comp_ser = blank
      @ 11,4 say "current use: STORAGE"
    case comp_ser <> blank
      @ 11,4 say "current use: COMPONENT ->"
    @ 13,12 say "mfg:"
    @ 13,c say comps->c_mfg
    @ 14,10 say "model:"
    @ 14,c say comps->c_model
    @ 15,7 say "serial #:"
    @ 15,c say comps->comp_ser
    @ 16,4 say "description:"
    @ 16,c say comps->c_desc
    @ 18,5 say "custodian:"
    @ 18,c say comps->last_name +
      comps->first_name
  endcase
end
set confirm off
ans = " "
do while .not. ans "$aAyYnN"
  ans = " "
  @ 22,26 say "Is this the correct part?:;"
  @ 23,26 say "[ Yes / No / Abandon ]"
  @ 22,52 get ans picture "@!A"
  read
endo
dest confirm on
@ 22,0 clear to 23,79
if upper(ans) = "A"
correct = .t.
still_more = .f.
component = blank
usage = " "
checking = .f.
endif
if eof()
    still_more = .f.
    checking = .f.
endif
if upper(ans) = "N" .and. .not. eof()
    skip
do case
    case aa
        if .not. known .and.
            part_ser = trim(mpart_ser)
            rec = recno()
            still_more = .t.
        else
            still_more = .f.
        endif
        if known .and.
            part_ser = trim(mpart_ser) .and.
            comp_ser = trim(component)
            rec = recno()
        else
            still_more = .f.
        endif
    case bb
        if known .and.
            p_model = trim(mmod) .and.
            comp_ser = trim(component)
            rec = recno()
        else
            still_more = .f.
        endif
        if .not. known .and.
            p_model = trim(mmod)
            rec = recno()
            still_more = .t.
        else
            still_more = .f.
        endif
    case cc
        if known .and. p_ptype = mptype .and.
            comp_ser = trim(component)
            rec = recno()
        else
            still_more = .f.
        endif
        if .not. known .and. p_ptype = mptype
            rec = recno()
            still_more = .t.
        else
            still_more = .f.
        endif
    case dd
        if known .and.
            p_pnum = trim(mpnum) .and.
            comp_ser = trim(component)
            rec = recno()
        else
            still_more = .f.
        endif
        if .not. known .and.
            p_pnum = trim(mpnum)
            rec = recno()
still_more = .t.
else
  still_more = .f.
endif

case EE
  if known .and.;
    comp_ser = trim(component)
    rec = recno()
  else
    still_more = .f.
  endif
  if .not. known .and. .not. eof()
    rec = recno()
    still_more = .t.
  endif
endcase
endif
if upper(ans) = "Y" .and. .not. eof()
  change_it = .t.
  correct = .t.
  still_more = .f.
  checking = .f.
endif
enddo still_more
if .not. correct .and. .not. still_more
  @ 3.0 clear to 20.79
  @ 22.16 say "Your PART cannot be found, and"+
  " may not be on file"
delay = 0
  do while delay < 50
    delay = delay + 1
  enddo
  @ 22.0 clear to 22.79
  component = blank
  usage = " "
  checking = .f.
endif
endcase
enddo checking
if change_it
  @ 22.19 say "Do you wish to modify this PART?: :
  @ 23.26 say "[ Yes / No ]"
  set confirm off
  ans = " "
  do while .not. ans $ "YnN"
    ans = " "
    @ 22.52 get ans
    read
  enddo
  set confirm on
  @ 22.0 clear to 23.79
  if upper(ans) = "N"
    change_it = .f.
  else
    entering = .f.
    change_it = .t.
  endif
endif
if located .and. .not. change_it
  @ 22.16 say "Do you have additional parts to modify?: :
  @ 23.26 say "[ Yes / No ]"
  set confirm off
ans = "" "
do while .not. ans $ "yYnN"
  ans = "" "
  @ 22,56 get ans
  read
  enddo
set confirm on
if upper(ans) = "N"
  set confirm off
  release all
  close databases
  return
else
  component = blank
  usage = "" "
  @ 22,0 clear to 23,79
endif
endif
  enddo entering
if change_it
  * allow editing the part record
  new_mfg = blank
  new_mod = blank
  new_desc = space(50)
  new_ser = blank
  new_ptype = " "
  new_pnum = space(10)
  new_price = 0.00
  new_reqn = blank
  new_use = " "
  new_comp = blank
  changing_it =.t.
  changed = .f.
  ok = .t. & used if component is found
  do while changing_it .and. .not. changed
    @ 3,0 clear to 20,79
    @ 3,5 say "Enter corrected Part information"
    @ 22,16 say:
      "Make required changes, press RETURN when correct"
    @ 5,10 say "model: "
    @ 6,7 say "serial #: "
    @ 7,4 say "description: "
    new_mod = p_mod
    @ 5,c get new_mod PICTURE "@N!"
    read
    * enter mfg serial#
      new_ser = part_ser
      @ 6,c get part_ser PICTURE "@N!"
      read
    * enter description
      new_desc = p_desc
      @ 7,c get new_desc PICTURE "@N!"
      read
    * determine if stock or component use
      set confirm off
      if comp_ser = blank
        new_use = "S"
      else
        new_use = "C"
      endif
@ 9.1 say "designated use: "
@ 9.20 say "((Storage / Component)"
@ 9.c get new_use PICTURE "@!A"
read

do while .not. new_use "$Ccs$s"
    new_use = "$"
    @ 9.c get new_use PICTURE "@!A"
read
endo
da 7 var confirm on

* enter component serial# for use = C, search to see if on file
* comp_ser = blank for storage use
* use comps for use = C component ser # mandatory
* storage use
    if upper(new_use) = "S"
        new_comp = blank
    endif
* component use
    if new_use = "C"
        @ 10.0 clear to 13,79
        @ 11.6 say "Component"
        @ 12.6 say "serial #: "
        searched = .f.
        do while .not. searched
            no_ser = .t.
            do while no_ser
                new_comp = comp_ser
                @ 12.c get new_comp PICTURE "@!N"
                read
                    if new_comp = blank
                        @ 22.0 clear to 22.79
                        @ 22.20 say "Component's serial # may not be blank"
                        delay = 0
                        do while delay < 25
                            delay = delay + 1
                        enddo
                        @ 22.0 clear to 22.79
                    else
                        no_ser = .f.
                    endif
                endif
            enddo
        endif
        no component serial#
* check to see if on file
    select comps
goto top
set order to 1
goto top
on_file = .f.
done = .f.
if eof() on_file = .f.
else
    seek trim(new_comp)
endif
if found() searched = .t.
on_file = .t.
done = .t.
endif
if .not. found() .or. .not. on_file
    @ 22.0 clear to 22.79
    @ 22.22 say "Component not on file !!!"
    delay = 0
    do while delay < 25

104
delay = delay + 1
enddo
@ 22,0 clear to 22,79
set confirm off
ans = ""
do while .not. ans "$yYN"
   ans = ""
   @ 22,20 say "Is this the correct":
   " serial #: "
   @ 23,26 say "[ Yes / No ]"
   @ 22,50 get ans picture "$!A"
   read
enddo
@ 22,0 clear to 22,79
set confirm on
if upper(ans) = "Y"
   searched = .t.
   ok = .f.
   changing_it = .f.
   done = .t.
   @ 22,15 say "This component must "+;
      "be entered first"
   delay = 0
   do while delay < 25
      delay = delay + 1
   enddo
   @ 22,0 clear to 22,79
else
   @ 22,15 say "Please re-enter the component":
      " serial#"
   delay = 0
   do while delay < 25
      delay = delay + 1
   enddo
   @ 22,0 clear to 22,79
endif
endif not found
enddo searched
endif use = Component
if ok
* enter property type (mandatory)
select parts
   goto rec
set confirm off
   @ 14,2 say "property type:"
   @ 14,20 say "(Plant / Minor / Other)"
new_ptype = p_ptype
   @ 14,c get new_ptype PICTURE "$!A"
   read
   do while .not. new_ptype "$mMoOpP"
      new_ptype = p_ptype
      @ 14,c get new_ptype PICTURE "$!A"
      read
   enddo
set confirm on
* no property# for other type property
   if upper(new_ptype) = "O" && ensure it is blank
      new_pnum = space(10)
   endif
* enter property# (mandatory for plant and minor property types)
   if upper(new_ptype) = "M" .or. upper(new_ptype) = "P"
      @ 15,5 say "property #:"
      no_num = .t.
      do while no_num
new_pnum = p_pnum.
@ 15,c get new_pnum PICTURE "@N1"
read
if new_pnum = space(10)
    @ 22,12 say "Minor and Plant property "+
    "requires a property number"
    delay = 0
    do while delay < 25
        delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
    else
        no_num = .f.
endif
enddo

* enter price and requisition#
@ 17,10 say "price: "$
@ 18,9 say "reqn #: "
new_price = p_price
new_reqn = p_reqn
@ 17,c get new_price PICTURE "@R 9,999.99"
@ 18,c get new_reqn PICTURE "@R 9999-NNNN/NNNNN"
read
set confirm off
@ 22,0 clear to 23,79
ans = ""
do while .not. ans "$AAyNn"
    ans = ""
    @ 22,25 say "Are the modifications correct?: "
    @ 23,29 say "[Yes / No / Abandon]"
    @ 22,56 get ans picture "@!A"
read
endo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "A"
    changing_it = .f.
endif
if upper(ans) = "Y" && add mvar to dbf
    changed = .t.
endif
endif ok
enddo changed or changeing it
endif change_it

* if original record was changed replace the old fields
* with the new ones
if changed
    set confirm off
    @ 22,0 clear to 22,79
    ans = ""
do while .not. ans "$yYnN"
    ans = ""
    @ 22,22 say "Do you wish to file the modified part?: "
    @ 23,29 say "[Yes / No ]"
    @ 22,61 get ans picture "@!A"
read
endo
@ 22,0 clear to 23,79
set confirm on
if upper(ans) = "N"
    changed = .f.
endif
if upper(ans) = "Y" && add mvar to dbf
  @ 22,20 say "Standby while your entry is placed on file"

* place in parts
  select parts
  goto rec
  replace p_model with trim(new_mod)
  replace p_desc with trim(new_desc)
  replace part_ser with trim(new_ser)
  replace comp_ser with trim(new_comp)
  replace p_type with new_p_type
  replace p_pnum with trim(new_pnum)
  replace p_price with new_price
  replace p_preqn with trim(new_preqn)
  changed = .f.
endif changed
endif not changed
if .not. changed
  @ 22,0 clear to 23,79
  @ 22,18 say "Do you have additional part"+
  @ 23,34 say "[ Yes / No ]"
  set confirm off
  ans = "" 
  do while .not. ans $ "YnN"
    ans = " "
    @ 22,65 get ans
    read
  enddo
  if upper(ans) = "N"
    finished = .t.
    clear
  else
    @ 22,0 clear to 23,79
    entering = .t.
    usage = " "
    mmmod = blank
    description = space(50)
    mpart_ser = blank
    mptype = " "
    mpnum = blank
    component = blank
    mc_mod = blank
    mc_mfg = blank
    mc_desc = space(50)
    mc_pctype = " "
    mc_pnum = blank
    lname = blank
    fname = blank
    known = .f.
    finished = .f.
    change_it = .f.
    ok = .t.
    @ 3,0 clear to 20,79
  endif
  endif not changed
enddo finished
set confirm off
release all
close databases
return
endo
* EOF modpart.prg
* begin the text dialogue

清 @ 0,14 说 " MOD I F Y P R O P E R T Y H E L P S C R E E N"
@ 1,0 to 1,79 double

text

* Modifications are made to components and parts changing accountability or to the record itself
* Imperative the correct serial number be inputted in all cases
* Modifications are only accomplished to existing data

detext
@ 21,0 to 21,79

等待 "→ press any key for more help or ESC to exit"

清 @ 1,0 to 1,79 double

"CAUTIONS"

* Extremely important that all information is entered correctly
* Mistakes will be made so utilize the "NO" or "ABANDON" commands before resuming
* Recommend using DELETION and ARROW keys for modifying data entered and ensure not to deviate outside designated fields

detext
@ 21,0 to 21,79

等待 "→ press any key for more help or ESC to exit"

清 @ 1,0 to 1,79 double

"WARNINGS"

* Use of BACKSPACE key can cause PREMATURE exiting of an entry field, loop back for re-entrance of data
* Serial Numbers must be accurate and precise

detext
@ 21,0 to 21,79

等待 "→ press any key to exit"

*EOF MOD_HELP.PRG
16. OWNERS.PRG

******************* Program: OWNERS.PRG *******************
*
*Author...............: TIM SEXTON
*Purpose...............: displays the entire file of custodians, and
*Calls...............: None
*Input/Output Files.: None

clear
set confirm on
set headings off
select 1
use homes index 1_fnames
select 2
use owners index names
set relation to last_name + first_name into homes

end_list = .f.
do while .not. end_list
  lname = space(15)
  fname = space(15)
  line = 5
  c1 = 0
  c2 = 31
  c3 = 43
  @ 0,20 say "C U S T O D I A N   L I S T I N G S "
  @ 1,0 to 1,79 double
  @ 21,0 to 21,79
  @ 3,c1+3 say "Custodian"
  @ 3,c2 say "Office"
  @ 3,c3+3 say "Home Address"
do while .not. eof()
do case
  case last_name = "AS DEPT"
    skip
  otherwise
    @ line,c1 say trim(last_name) + ", " + first_name
    if location <> "HOME"
      @ line,c2 say location
    endif
    if location = "HOME"
      @ line,c3 say trim(homes->street) + "; " + homes->city
    endif
  line = line + 2
  skip
do case
  case lname = last_name .and. fname = first_name;
    .and. location = "HOME"
    if line = 5
      @ line,c3 say trim(homes->street) +;
      "", " + homes->city
    else
      @ line - 2,c3 say trim(homes->street) +;
      "", " + homes->city
    endif
  skip

109
endcase
lname = last_name
fname = first_name
if line > 20
    wait;
space(16)+"Press any key to continue, or ESC to exit"
    @ 4,0 clear to 20,79
    @ 22,0 clear to 24,79
    line = 5
endcase
enddo
if eof()
    @ 22,25 say " That is everyone on file "
delay = 0
    do while delay < 25
        delay = delay + 1
    enddo
    @ 22,25 clear to 22,79
    @ 22,15 say;
        "Press any key to return to Search and Listing Menu"
    end_list = .t.
    wait " "
endif
close databases
release all
return
endo
* EOF owners.prg

17. PMANF.PRG

****************** Program: PMANF.PRG ******************
*Author.............. TIM SEXTON
*Purpose............... displays the components by mfg searches
*Calls................ None
*Input/Output Files.: None

clear
set confirm on
select 1
use owners index names
select 2
use comps.index name.loc
set relation to last_name + first_name into owners
end_list = .f.
do while .not. end_list
    blank = space(15)
    mmfg = blank
    @ 0.14 say "COMPONENT MFG SEARCH SCREEN"
    @ 1.0 to 1.79 double
    @ 21.0 to 21.79
    @ 22,25 say "To EXIT leave mfg name blank"
    @ 4.6 say "Enter Mfg"
    @ 6.6 say " name: " get mmfg PICTURE "@IN"
read
if mmfg = blank
  clear
  release all
  close databases
  return
endif
@ 22,0 clear to 22,79
goto top
set exact off
locate for c_mfg = ltrim(mmfg)
line = 6
c1 = 10
c2 = 28
c3 = 55
if found()
  @ 2,0 clear to 20,79
  @ 2,1 say "MFG:"
  @ 2,6 say mmfg
  @ 3,c1 say "Model"
  @ 4,c1 to 4,c1+5
  @ 3,c2 say "Custodian"
  @ 4,c2 to 4,c2+9
  @ 3,c3 say "Location"
  @ 4,c3 to 4,c3+8
  do while .not. eof()
    @ line,c1 say c_model
    do case
      case last_name = "AS DEPT"
        @ line,c3 say trim(owners->location) + ", " + 
               first_name
      otherwise
        @ line,c2 say trim(last_name) + ", " + first_name
        if loc_code = "H"
          @ line,c3 say "HOME"
        else
          @ line,c3 say owners->location
        endif
    endcase
    @ line+1,15 say "-> " + c_desc
    line = line + 3
  if line > 20
    @ 22,20 say "Press any key to continue, or ESC to exit"
    @ 6,0 clear to 20,79
    @ 22,0 clear to 24,79
    line = 6
  endif
  continue
  if eof()
    @ 22,27 say "Nothing further is on file"
    delay = 0
    do while delay < 50
      delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
  endif
  enddo
else
  @ 22,0 clear to 22,79
  @ 22,27 say "Nothing is on file for this mfg"
  delay = 0
do while delay < 50
    delay = delay + 1
endo
@ 22,0 clear to 22,79
endif
@ 22,15 say "Press any key to return to Search and Listing Menu"
end_list = .t.
wait ""
close databases
release all
return
endo
* EOF pmanf.prg

18.  PMOD.PRG

***************  Program:  PMOD.PRG  ***************
*
*Author............:  TIM SEXTON
*Purpose............:  displays the components by model searches
              on c_model
*Calls...............:  None
*
*Input/Output Files.:  None

clear
set confirm on
select 1
use owners index names
select 2
use comps index name_loc
set relation to last_name + first_name into owners
end_list = .f.
do while .not. end_list
    blank = space(15)
    mmod = blank
    @ 0,12 say: "COMPONENT MODEL SEARCH SCREEN"
    @ 1,0 to 1,79 double
    @ 21,0 to 21,79
    @ 22,22 say "To EXIT leave model name blank"
    @ 4,6 say "Enter Model"
    @ 6,6 say " name: " get mmod PICTURE "@!N"
    read
    if mmod = blank
        clear
        release all
        close databases
        return
    endif
@ 22,0 clear to 22,79
goto top
set exact off
locate for c_model = ltrim(mmod)
line = 6
cl = 10
c2 = 28
c3 = 55
if found()
    @ 2,0 clear to 20,79

do while .not. eof()
    @ line,cl say c_mfg
    do case
        case last_name = "AS DEPT"
            @ line,c2 say last_name
            @ line,c3 say trim(owners->location) + "," + first_name
        otherwise
            @ line,c2 say trim(last_name) + "," + first_name
            if loc_code = "H"
                @ line,c3 say "HOME"
            else
                @ line,c3 say owners->location
            endif
        endcase
    @ line+1,15 say "+" + c_desc
    line = line + 3
    if line > 20
        @ 22,20 say "Press any key to continue, or ESC to exit"
        wait 
        @ 6,0 clear to 20,79
        @ 22,0 clear to 24,79
        line = 6
    endif
    continue
    if eof()
        @ 22,27 say "Nothing further is on file"
        delay = 0
        do while delay < 50
            delay = delay + 1
        enddo
        @ 22,0 clear to 22,79
    endif
    enddo
else
    @ 22,0 clear to 22,79
    @ 22,25 say "Nothing is on file for this model"
    delay = 0
    do while delay < 50
        delay = delay + 1
    enddo
    @ 22,0 clear to 22,79
end if
@ 22,15 say "Press any key to return to Search and Listing Menu"
end list = .t.
wait " "
close databases
release all
return
enddo
* EOF pmod.prg
19. PROPERTY.PRG

************************************************************************ Program: PROPERTY.PRG **************
*Author................: TIM SEXTON
*Purpose.............: Main Menu displays the various tasks that are
  available for the user.
*Calls..............: ADDMENU.PRG, MODMENU.PRG, DELMENU.PRG,
  REPORTS.PRG, ADHOC.PRG
*Input/Output Files.: None

********** set up the working environment **********
* restricts the control and interfaces strictly to this
  application, and not with dBase III plus
set color to gr+/b, r/w, b, b
close databases
clear
clear all
set talk off
set echo off
set scoreboard off
set bell off
set status off
set safety off
set deleted on
on escape return
* display the opening screen
text

Property Management System

The Property Management System is an application program to
assist the Admin Science Dept in maintaining accountability
for departmental property.

endtext
?
?
?
wait

* Check the users password for their access rights, exit the
  application program if no proper password

clear
set intensity off
set confirm on
password = space(10)
access = space(10)
@ 8,20 to 11,60 double
no_code = .t.
do while no_code
  @ 9,21 clear to 10,59
  @ 9.23 say "ENTER YOUR PASSWORD: "
  @ 10,23 say " (or press return to quit)"
* hide the password typed by changing inputs to the same color as
  the background, set it back after input entered
  set color to b/b, b/b, b, b
  @ 9,44 get password
  read
  set color to gr+/b, r/w, b, b
  do case
    case password = space(10)
      set color to w/n, n/w, n, n
quit
release all

clear

case lower(password) = "limited"
   access = "read_only"
   no_code = .f.

case lower(password) = "unlimited"
   access = "read_write"
   no_code = .f.

otherwise
   @ 9,21 clear to 10,59
   @ 9,23 say "IMPROPER PASSWORD, PLEASE RE-ENTER"
   delay = 0
   do while delay < 25
      delay = delay + 1
   enddo
   password = space(10)
endo
clear
set intensity on

* the program begins, the start of the dialogue unit, similar
* menu screens are used for options 1 - 5
do while .t.
   set confirm off
   clear
   @ 2,10 to 17,69 double
   @ 3,25 say "Property Management Main Menu"
   @ 4,11 to 4,68 double
   @ 6,27 say "1 - LISTS or SEARCHES"
   @ 7,27 say "2 - PRINT REPORTS"
   @ 9,27 say "3 - ENTER new property"
   @ 10,27 say "4 - DELETE property"
   @ 11,27 say "5 - MODIFY property"
   @ 13,27 say "H - HELP"
   @ 15,27 say "0 - Exit to MS.DOS"
   @ 17,30 say "selection : : "
   choice = ""
   @ 17,42 get choice
   read
* display an asterisk next to the choice, choice row used for
* erasing other rows
   if choice $('#$hH012345')
      do case
         case upper(choice) = "H"
            @ 13,26 say "*"
            choicerow = 13
         case choice = "0"
            @ 15,26 say "*"
            choicerow = 15
         case choice $"12"
            @ 5+val(choice),26 say "*"
            choicerow = 5+val(choice)
         otherwise
            @ 6+val(choice),26 say "*"
            choicerow = 6+val(choice)
         endcase
      firstrow = 6
      rows = 10
      rowcnt = 0
      do while rowcnt < rows
if rowcnt+firstrow <> choicerow.
   @ firstrow+rowcnt,27 say space(25)
endif
rowcnt = rowcnt + 1
enddo

* do choice, access must match the password a user entered
* this is the control section of this program
  do case
   case choice = "0"
      set color to w/n, n/w, n, n
      quit
   case choice = "1"
      if lower(access) = "read_only" .or.;
         lower(access) = "read_write"
      do adhoc
         @ 19,10 say;
         "***** you do not have proper access";
         " for this selection *****"
      ?
      wait
      loop
endif
   case choice = "2"
      if lower(access) = "read_only" .or.;
         lower(access) = "read_write"
      do reports
      else
         @ 19,10 say;
         "***** you do not have proper access";
          " for this selection *****"
      ?
      wait
      loop
endif
   case choice = "3"
      if lower(access) = "read_write"
      do addmenu
      else
         @ 19,10 say;
         "***** you do not have proper access";
          " for this selection *****"
      ?
      wait
      loop
endif
   case choice = "4"
      if lower(access) = "read_write"
      do delmenu
      else
          @ 19,10 say;
          "***** you do not have proper access";
           " for this selection *****"
      ?
      wait
      loop
endif
   case choice = "5"
      if lower(access) = "read_write"
      do modmenu
      else
          @ 19,10 say;
          "***** you do not have proper access";
           " for this selection *****"
      ?
      wait
      loop
loop
endif
case upper(choice) = "H"
do main_hlp
otherwise
    @ 19,22 say "****** not a valid selection ******"
? wait
loop
endcase
enddo
* EOF property.prg

20. QRY_HELP.PRG

***************************** Program: QRY_HELP.PRG *****************************
*
*Author.................: TIM SEXTON
*Purpose................: describes the options available to the
*Calls..................: None
*Input/Output Files.: None
*
* begin the text dialogue
clear
@ 1,0 to 1,79 double
@ 0,15 say "LIST AND SEARCH HELP MENU"
text

* Searches are designed to provide the user with
the following lists:

1. Custodians on file
2. Locations of Components of interest
3. Custodians of various Components
4. A List of a Manufacturer's Components
5. A List by Model of Components

* Component Lists will provide the Component
location

* To search on a custodian, ensure the name is
entered as it is kept on file. Option # 3 will
provide a list of all custodians.
endtext
@ 21,0 to 21,79
Wait " -> press any key to exit"
*EOF QRY_HELP.PRG

21. QTR_RPT.PRG

***************************** Program: QTR_RPT.PRG *****************************
*
*Author.................: TIM SEXTON
*Purpose................: prints three reports
*Calls..................: TEMP.FR M, TEMP3.FR M, STOCKPART.FR M, COMPS.DBF,

117
*Input/Output Files.: none

* This program will join the owners and components to allow
  * printing the component and property reports

set confirm off
set exact on
clear

select 1
use parts
select 2
use owners
select 3
use comps
do while .t.

@ 0.18 say "QUARTERLY REPORT SCREEN"
@ 1.0 to 1.79 double
@ 21.0 to 21.79
@ 8.16 say "********** SET UP THE PRINTER **********";
@ 22.6 say "Standby while the files are joined for preparing";
  "the property reports"

****** temp is used to join campus owner with component ******
select comps
join with owners to temp for loc_code <> "H" .and. owners->;
  location <> "HOME" .and. last_name = owners->last_name .and.;
  first_name = owners->first_name

****** temp2 is used to join home owner with component ******
join with owners to temp2 for loc_code = "H" .and. owners->;
  location = "HOME" .and. last_name = owners->last_name .and.;
  first_name = owners->first_name

************ temp2 is then appended to temp ***************
************ temp is used for component report ***************
select 4
use temp
append from temp2
index on last_name + first_name + location + c_mfg +;
  c_model to temp

************ temp3 is used for part report ***************
join with parts to temp3 for comp_ser = parts->comp_ser
select 5
use temp3
index on last_name + first_name + location + c_mfg + c_model +;
  comp_ser to temp3

close databases

* check if printer is set up or allow abandon current operation
done = .f.
ready = .f.
do while .not. ready
  @ 8.0 clear to 8.79

118
**EOF qtr_rpt.prg**
22. REPORTS.PRG

***************
Program: REPORTS.PRG
***************

*Author............: TIM Sexton
*Purpose............: Menu displays the choice; print a quarterly
property report or a summary report by
property type and number
*Calls..............: QTR_RPT.PRG, SUM_RPT.PRG
*Input/Output Files.: NONE

* set screen environment
clear
set confirm off
* display the dialogue menu
do while .t.
clear
@ 2,10 to 13,69 double
@ 3,30 say "Property Reports Menu"
@ 4,11 to 4,68 double
@ 6,27 say "1 - QUARTERLY report"
@ 7,27 say "2 - PROPERTY SUMMARY report"
@ 9,27 say "H - HELP"
@ 11,27 say "0 - RETURN to main menu"
@ 13,30 say "selection:"
choice = ""
@ 13,42 get choice
read
* place an asterisks next to a valid choice, and erase the other
* rows
if choice $ "H012"
do case
    case upper(choice) = "H"
        @ 9,26 say "*
        choicerow = 9
    case choice = "0"
        @ 11,26 say "*
        choicerow = 11
    otherwise
        @ 5+val(choice),26 say "*
        choicerow = 5+val(choice)
endcase
firstrow = 6
rows = 7
rowcnt = 0
do while rowcnt < rows
    if rowcnt+firstrow <> choicerow
        @ firstrow+rowcnt,27 say space(30)
    endif
    rowcnt = rowcnt + 1
enddo
* do choice if valid, or loop back thru this program
do case
    case choice = "0"
        return
    case choice = "1"
        do qtr_rpt
    case choice = "2"
        do sum_rpt
    case upper(choice) = "H"
        do rpt_help
    otherwise
23. RPT_HELP.PRG

*********************** Program: RPT_HELP.PRG ***********************
* *Author...............: TIM SEXTON
*Purpose...............: describes the options available to the
* user
*Calls..................: None
*Input/Output Files.: None
* begin the text dialogue
clear
@ 1.0 to 1.79 double
@ 0.13 say "PROPERTY REPORTS HELP SCREEN"
text

* Ensure Printer is ON and READY
* Reports are pre-formatted
* Provides two types of Reports, namely:
** Quarterly---Grouped by Custodian/Owner
** Summary----Grouped by Property Type
and Property Number
endtext
@ 21.0 to 21.79 Wait " -> press any key to exit"
*EOF RPT_HELP.PRG

24. SLOCATIO.PRG

*********************** Program: SLOCATIO.PRG ***********************
* *Author..............: TIM SEXTON
*Purpose...............: displays components and the location
* of the component, searches on loc_code
*Calls..................: None
*Input/Output Files.: None

set headings off
set confirm off
select a
use comps index name_loc
finished = .f.
do while .not. finished
  clear
ans = " "
@ 0,11 say: "COMPONENT LOCATION LIST SCREEN"
@ 1,0 to 1,79 double
@ 21,0 to 21,79
@ 22,26 say "Leave choice blank to EXIT"
@ 4,27 say "LOCATIONS"
@ 7,27 say "1. HOME or OFFICE"
@ 9,27 say "2. STORAGE"
@ 11,27 say "3. LAB "
no_ans = .t.
do while no_ans
   @ 16,20 say "PLEASE ENTER YOUR CHOICE:" get ans
   read
   if ans $ "123"
      no_ans = .f.
   else
      ans = " "
   endif
endo
dif ans <> " "
clear
do case
   case ans = "1"
      code = "P"
      @ 0,28 say "PERSONNEL COMPONENTS"
   case ans = "2"
      code = "S"
      @ 0,28 say "STORAGE COMPONENTS"
   case ans = "3"
      code = "L"
      @ 0,28 say "LAB COMPONENTS"
endcase
goto top
@ 1,0 to 1,79
@ 21,0 to 21,79
if code = "P"
   do while .not. eof()
      if last_name = "AS DEPT"
         skip
      else
         line = 7
         lname = last_name
         fname = first_name
         @ 3,4 say "Custodian: "
         @ 3,15 say trim(last_name) + " ," + first_name
         @ 5,2 say "Mfg/Model"
         @ 6,2 to 6,10
         @ 5,32 say "Serial #"
         @ 6,32 to 6,39
         @ 5,50 say "Ptype & number"
         @ 6,50 to 6,63
         @ 5,72 say "Location"
         @ 6,72 to 6,79
         do while last_name = lname .and. first_name = fname
            if loc_code = "O" .or. loc_code = "H"
               @ line,0 say trim(c_mfg)+"/ " + c_model
               @ line,32 say comp_ser
               @ line,50 say cptype + " - " + c_pnum
               if loc_code = "O"
                  @ line,72 say "Office"
               endif
            if loc_code = "H"
               @ line,72 say "Home"
            endif
            next
         enddo
      endif
   if code = "S"
      do while .not. eof()
         if last_name = "AS DEPT"
            skip
         else
            line = 7
            lname = last_name
            fname = first_name
            @ 3,4 say "Custodian: "
            @ 3,15 say trim(last_name) + " ," + first_name
            @ 5,2 say "Mfg/Model"
            @ 6,2 to 6,10
            @ 5,32 say "Serial #"
            @ 6,32 to 6,39
            @ 5,50 say "Ptype & number"
            @ 6,50 to 6,63
            @ 5,72 say "Location"
            @ 6,72 to 6,79
            do while last_name = lname .and. first_name = fname
               if loc_code = "O" .or. loc_code = "H"
                  @ line,0 say trim(c_mfg)+"/ " + c_model
                  @ line,32 say comp_ser
                  @ line,50 say cptype + " - " + c_pnum
                  if loc_code = "O"
                     @ line,72 say "Office"
                  endif
               if loc_code = "H"
                  @ line,72 say "Home"
               endif
            enddo
         endif
      endif
      do while .not. eof()
         if last_name = "AS DEPT"
            skip
         else
            line = 7
            lname = last_name
            fname = first_name
            @ 3,4 say "Custodian: "
            @ 3,15 say trim(last_name) + " ," + first_name
            @ 5,2 say "Mfg/Model"
            @ 6,2 to 6,10
            @ 5,32 say "Serial #"
            @ 6,32 to 6,39
            @ 5,50 say "Ptype & number"
            @ 6,50 to 6,63
            @ 5,72 say "Location"
            @ 6,72 to 6,79
            do while last_name = lname .and. first_name = fname
               if loc_code = "O" .or. loc_code = "H"
                  @ line,0 say trim(c_mfg)+"/ " + c_model
                  @ line,32 say comp_ser
                  @ line,50 say cptype + " - " + c_pnum
                  if loc_code = "O"
                     @ line,72 say "Office"
                  endif
               if loc_code = "H"
                  @ line,72 say "Home"
               endif
            enddo
            do while .not. eof()
               if loc_code = "O" .or. loc_code = "H"
                  @ line,0 say trim(c_mfg)+"/ " + c_model
                  @ line,32 say comp_ser
                  @ line,50 say cptype + " - " + c_pnum
                  if loc_code = "O"
                     @ line,72 say "Office"
                  endif
               if loc_code = "H"
                  @ line,72 say "Home"
               endif
            enddo
diff
endif
line = line + 2
skip
if line > 20 .and. (last_name = lname .and. first_name = fname)
    @ 22.0 clear to 22.79
    @ 22.4 say "Additional property"; "on file for this Custodian, press";
    "any key to continue"
    wait" "
    @ 7.0 clear to 20.79
    @ 22.0 clear to 22.79
    line = 7
endif
else
    skip
endif
if last_name <> lname .and. first_name <> fname
    @ 22.0 clear to 22.79
    @ 22.4 say "That is all";
    "that is on file, press any";
    "key to continue, or ESC to exit"
    wait" "
    line = 7
    @ 3.15 clear to 3.79
    @ 7.0 clear to 20.79
    @ 22.0 clear to 22.79
endif
enddo
endif
endif
if code = "$"
    lname = "AS DEPT"
    seek lname
    if found()
        line = 7
        @ 3.4 say "Custodian: "
        @ 3.15 say "AS DEPT"
        @ 5.2 say "Mfg/Model"
        @ 6.2 to 6.10
        @ 5.32 say "Serial #"
        @ 6.32 to 6.39
        @ 5.50 say "Ptype & number"
        @ 6.50 to 6.63
        @ 5.72 say "Location"
        @ 6.72 to 6.79
        do while last_name = "AS DEPT"
            if loc_code = "$"
                @ line.0 say trim(c_mfg) + " / " + c_model
                @ line.32 say comp_ser
                @ line.50 say c_ptype + " - " + c_pnum
                @ line.72 say "(I-274)"
        line = line + 2
        skip
for line > 20 .and. (last_name = lname .and. loc_code = "$")
    @ 22.0 clear to 22.79
    @ 22.9 say "Additional storage property";
    "on file, press any key to continue"
    wait" "
    @ 7.0 clear to 20.79
    @ 22.0 clear to 22.79
    line = 7
endif
else
  skip
endif
if last_name <> lname
  @ 22,0 clear to 22,79
  @ 22,4 say "That is all";
  " that is on file, press any";
  " key to continue, or ESC to exit"
  wait"
endif
enddo
else
  @ 22,0 clear to 22,79
  @ 22,12 say "No storage components on file,";
  wait"
  clear
endif
enddo
endif
if code = "L"
lname = "AS DEPT"
seek lname.
if found()
  line = 7
  @ 3,4 say "Custodian: "
  @ 3,15 say "AS DEPT"
  @ 5,2 say "Mfg/Model"
  @ 6,2 to 6,10
  @ 5,32 say "Serial #"
  @ 6,32 to 6,39
  @ 5,50 say "Ptype & number"
  @ 6,50 to 6,63
  @ 5,72 say "Location"
  @ 6,72 to 6,79
  do while last-name = "AS DEPT"
    if loc_code = "L"
      @ line,0 say trim(c_mfg)+" / " + c_model
      @ line,32 say comp_ser
      @ line,50 say c_p type + " - " + c_pnum
      @ line,72 say first_name
      line = line + 2
      skip
      if line > 20 .and. (last_name = lname;
        .and., loc_code = "L")
        @ 22,0 clear to 22,79
        @ 22,12 say "Additional lab property";
        " on file, press any key to continue"
        wait"
        @ 7,0 clear to 20,79
        @ 22,0 clear to 22,79
        line = 7
      endif
      else
        skip
      endif
    if last_name <> lname
      @ 22,0 clear to 22,79
      @ 22,4 say "That is all";
      " that is on file, press any";
      " key to continue, or ESC to exit"
      wait"
      clear
    endif
  enddo
enddo
endif

else
  @ 22.0 clear to 22,79
  @ 22,15 say "No lab components on file," +;
  wait "
  clear
  endif
enddo
else
  clear
  release all
close databases
return
endif ans not " "
ans = " "
enddo
*EOF slocatio.prg

25. SOWNER.PRG

************************ Program: SOWNER.PRG ************************
*Author..............: TIM SEXTON
*Purpose..............: displays custodian's components and the
                      location of the component, searches on last
                      and first name
*Calls..............: None
*Input/Output Files.: None

clear
set confirm off
set headings off
select 3
use homes index l_fnames
select 2
use owners index names
select 1
use comps index name_loc
set relation to last_name + first_name into owners
finished = .f.
do while .not. finished
  choice = " "
  miname = space(15)
  mfname = space(15)
  @ 0.18 say "COMPONENT SEARCH SCREEN"
  @ 1.0 to 1.79 double
  @ 22.0 to 21,79
  @ 22,27 say "To EXIT leave choice blank"
  @ 4.4 say "Choose a search for one of the following: ;"
  @ 4.4 say " 1. Components assigned to a Custodian"
  @ 4.4 say " 2. Components of the AS DEPT"
  @ 4.45 get choice
  read
do case
  case choice = " "
    clear
    release all
    close databases
    return
  case choice = " 1."
    clear
    release all
    close databases
    return
case choice = "I"
@ 4,0 clear to 7,79
set confirm on
@ 4,0 say "Enter custodian's"
@ 6,6 say "last name: " get mlname PICTURE "@!A"
@ 22,0 clear to 22,79
@ 22,13 say "To exit leave custodians name blank"
read
if mlname = space(15)
clear
release all
close databases
return
endif
@ 22,0 clear to 22,79
no_fname = .t.
do while no_fname
@ 7,5 say "first name: " get mfname PICTURE "@!A"
read
if mfname = space(15)
    clear
    release all
    close databases
    return
enddo
@ 22,0 clear to 22,79
else
    no_fname = .f.
endif
enddo
set confirm off
select comps
goto top
set exact off
seek mlname,mfname
if found()
    @ 0,0 clear to 0,79
    @ 4,0 clear to 7,79
    @ 22,0 clear to 22,79
    line = 6
    @ 0,25 say "Custodian: " + trim(mlname) + ", " + mfname
    @ 3,5 say "Mfg / Model"
    @ 4,5 to 4,15
    @ 3,35 say "Serial #"
    @ 4,35 to 4,42
    @ 3,52 say "Location"
    @ 4,52 to 4,59
do while last_name = mlname .and. first_name = mfname
    @ line,2 say trim(c_mfg) + " / " + c_model
    @ line,35 say comp_ser
do case
    case loc_code = "H"
        select homes
        seek mlname,mfname
        @ line,52 say trim(street) + ", " + city
        otherwise
        @ line,52 say owners-> location
    endcase
    select 1
    skip
    line = line + 2
    if line > 20 .and. (last_name = mlname .and.;
first_name = mfname)

@ 6,0 clear to 24,79
line = 6
def
if last_name <> mlname .and. first_name <> mfname
@ 22,0 clear to 22,79
@ 22,12 say "Nothing further on file,"+;
    wait" " press any key to continue"
    clear
endif
enddo
else
@ 22,0 clear to 22,79
@ 11,8 say;
    " Either no property on file for this individual"
@ 12,8 say " OR"
@ 13,8 say;
    " The name entered does not match what is on file"
?
    wait
    clear
endif
case choice = "2"
@ 0,0 clear to 0,79
@ 22,0 clear to 22,79
@ 4,0 clear to 7,79
@ 0,32 say "AS DEPT PROPERTY"
mlname = "AS DEPT"
select comps
goto top
set exact off
seek mlname
if found()
    line = 6
    @ 3,5 say "Mfg / Model"
    @ 4,5 to 4,15
    @ 3,35 say "Serial #"
    @ 4,35 to 4,42
    @ 3,52 say "Location"
    @ 4,52 to 4,59
do while last_name = "AS DEPT"
    @ line,2 say trim(c_mfg) + " / " + c_model
    @ line,35 say comp_ser
    @ line,52 say owners-> location + first_name
    skip
    line = line + 2
    if line > 20 .and. last_name = "AS DEPT"
        @ 22,0 clear to 22,79
        @ 22,15 say "Additional property on file,"+;
            " press any key to continue"
            wait" "
            line = 6
        @ 22,0 clear to 22,79
        @ 6,0 clear to 20,79
    endif
    if last_name <> "AS DEPT"
        @ 27,0 clear to 22,79
        @ 22,12 say "Nothing further on file,"+;
            wait" " press any key to continue"
            clear
        endif
    endif
else
endif
endo
delse
@ 22,0 clear to 22,79
@ 22,12 say "No property on file for the AS DEPT"; +
" press any key to continue"
wait" "press any key to continue"
clear
endif
otherwise
@ 22,0 clear to 22,79
@ 22,17 say "Not a valid selection, please re-enter or"
delay = 0
do while delay < 25
delay = delay + 1
endo
@ 22.0 clear to 22,79
endcase
endo
* EOF sowner.prg

26. SUM_RPT.PRG

*************** Program: SUM_RPT.PRG ***************
*Author...............: TIM SEXTON
*Purpose.............: prints three reports
* 1. components grouped by property type and number
* 2. parts assigned to components grouped by property type and number
* 3. parts that are not assigned to components (stock) grouped by property type and number
* involves creating temporary files that are erased after the reports are printed
*Calls.............: TEMP1.FRM, PARTSTOK.FRM, PARTSUM.FRM,
*                   COMPS.DBF, PARTS.DBF,OWNERS.DBF
*Input/Output Files.: none

* This program will join the owners and components to allow
* printing the summary property reports
set confirm off
set exact on
clear
select 1
use parts
select 2
use owners
select 3
use comps
do while .t.
@ 0,20 say "S U M M A R Y R E P O R T S C R E E N"
@ 1,0 to 1.79 double
@ 21.0 to 21.79
@ 8,16 say "******** SET UP THE PRINTER **********"
@ 22,6 say "Standby while the files are joined for preparing"; +
" the property reports"
****** temp is used to join campus owner with component ******
select comps
join with owners to temp for loc_code <> "H" .and. owners->;
location <> "HOME" .and. last_name = owners->last_name .and.:
first_name = owners->first_name

****** temp2 is used to join home owner with component ******
join with owners to temp2 for loc_code = "H" .and. owners->;
location = "HOME" .and. last_name = owners->last_name .and.:
first_name = owners->first_name

********** temp2 is then appended to temp **********
****** temp is used for component report ******
select 4
use temp
append from temp2
index on c_pctype + c_pnum to temp

********** temp3 is used for part report **********
select with parts to temp3 for comp_ser = parts->comp_ser
select 5
use temp3
index on p_pctype + p_pnum to temp3

********** parts are indexed on type & num **********
select parts
index on p_pctype + p_pnum to type_num
close databases

* check if printer is set up or allow abandon current operation
  done = .f.
  ready = .f.
  do while .not. ready
    @ 8,0 clear to 8,79
    @ 22,0 clear to 23,79
    @ 22,22 say "Is the printer ready for printing?: ":
    @ 23,22 say "[ Yes / No / Abandon ]"
    ans = " "
    do while .not. ans $ "yYnWwAa"
      ans = " "
      @ 22,57 get ans picture "@!A"
      read
    enddo
    @ 22,0 clear to 23,79
  * if not ready wait and loop
    if upper(ans) = "N"
      @ 8,18 say "Press any key when ready to ready to continue"
      wait " "
      @ 8,0 clear to 8,79
    endif
    if upper(ans) = "A"
      ready = .t.
      done = .t.
    endif
    if upper(ans) = "Y"
      ready = .t.
    endif
  enddo not ready
  if .not. done

129
clear
select 1
use temp index temp
************ do the component property type & number report
report form templ to print
select 2
use temp3 index temp3
************ do the part property report
report form partsum to print
select 3
use parts index type_num
************ do the stock part list
report form partstok to print for comp_ser = space(10)
done = .t.
endif not done
if done
erase temp.dbf
erase temp2.dbf
erase temp3.dbf
erase tempndx
erase temp3ndx
erase type_numndx
clear
release all
close databases
return
endif done
enddo
*eof sum_rpt.prg
APPENDIX C
PMS USER'S MANUAL

1. INTRODUCTION

This manual is designed to familiarize and serve as a reference for the Property Management System designed specifically for the Administrative Science Department. This system is written as an application program of dBase III plus, installed on an IBM PC XT located in Ingersoll 230. The Property Management System is menu driven, therefore there is no requirement to have a knowledge of dBase III plus. Additionally there is no real requirement for users to have a familiarity with the operation of an IBM microcomputer.

a. Getting Started

The first step in getting started is to take the system boot-up disk provided, inserting it into the floppy drive, then turning on the computer. Turning on the computer involves flipping the toggle switches on the right back side of the two system units, then turning the top switch on the front of the monitor. The printer need not be activated until prompted. If the system is started properly, after a minute or so, an initial license agreement screen for dBase III should appear. At this point depress the return key and commence the operation of the Property Management System. Your screen should appear identical to that of Figure C.1. If you're ready to continue, then as the screen says, press any key to continue.

If the system did not respond exactly as stated, or you receive any sort of error message at all, then it is likely that you used the wrong boot-up disk, or there is a system error. Check to see if the proper boot-up disk was used, if so alert the problem to the attention of the department supervisor for follow up investigation.

b. Passwords

After accessing the Property Management System your screen should appear exactly as Figure C.2, prompting you for your password. To utilize the Property Management System you must have an authorized password. The system makes use of two types of passwords to restrict the access to the system. A restricted password allows read-only access, which permits searching the existing records and printing reports. An unrestricted password allows you to enter, delete, or modify new or existing property records.
Property Management System

The Property Management System is an application program to assist the Admin Science Dept in maintaining accountability for departmental property.

Press any key to continue...

Figure C.1 Initial PMS Screen.

ENTER YOUR PASSWORD:
(or press return to quit)

Figure C.2 Passwords.

You will note that the characters are not echoed back to you on the terminal as you type them. This is an added security feature and not a defect in the system. Should you incorrectly enter your password, or the password is not valid, you will momentarily see an error message on the screen. After the error message simply re-enter your correct password. There is no limit to the number of times you may enter an incorrect password, but without a correct entry you will not progress past this point.

If you are certain you have correctly entered your password and are still being denied access, then you must verify with proper authority that your password is indeed
valid. To exit the system from this point, depress the return key and you will return to MS DOS. Turning the computer off is also an alternative and will not impair the program.

Trying to access restricted operations without a proper password will cause an error message. You must re-start the system and log on with a proper password to access these operations.

2. PROPERTY MANAGEMENT SYSTEM OPERATIONS

After entering a correct password the next screen you will see should be that of Figure C.3. With restricted access you may perform either of the first two listed operations. You may also make use of the on-line help facility by simply typing "H". An unrestricted password will allow you to perform any of the listed operations. If you choose to exit the system, typing a "0" will terminate the Property Management System and exit you to MS DOS, and the computer may be turned off, or used for another application.

![Property Management Main Menu](image)

Figure C.3 Main Menu.

a. Help

All menus throughout the entire program have a Help option as one of the choices. To select this option type an "H" as your selection. Selecting this will provide valuable information pertaining to the choices on the current menu screen. Help is
automatically exited when you have paged through all the information. You may exit prior to that by using the ESCape key.

b. Lists or Searches

If you desire a list of property, a sub-set of property, or to search for a specific item, then select "1" from the main menu. The List and Searches Menu will next be displayed on your screen (Figure C.4).

```
List and Search Menu

1. Components assigned to a Custodian
2. Components assigned by Locations
3. Custodian Listing
4. Components of a single Manufacturer
5. Components of a single Model
H - HELP
0 - Return to MAIN MENU

selection:
```

Figure C.4 List and Search Menu.

This menu displays all the available choices under this category. You may make any selection you choose. Typing a "0" will return you to the main menu.

1. Components Assigned to a Custodian

This option will allow you to search for components by assigned custodian, or components with the AS DEPT as custodian.

Entering a "1" for custodian search, from the Component Search Screen (Figure C.5), you will be prompted for last name of the custodian of interest.

By depressing the return key, you return to the List and Search Menu. To perform a search by custodian the name must be entered as kept on file. (If you have some doubt about this, use the Custodian Listing option from the List and Search Menu).
Choose a search for one of the following:
1. Components assigned to a Custodian
2. Components of the AS DEPT

To EXIT leave choice blank

Figure C.5  Component Search Screen.

Entering a "2" for AS DEPT component search, from the Component Search Screen (Figure C.5), will immediately provide you with a list of components sorted by location that the AS DEPT maintains direct responsibility (lab and storage property).

2. Components assigned by Locations

Selecting "2" from the List and Search Menu (Figure C.4), allows you the option of selecting a list of all components by their physical locations by making a selection from the menu choices of Home or Office, Storage, or Lab.

3. Custodian Listing

This selection provides a listing of all custodians and the locations that they have property on file: either their office or home address. Figure C.6 shows a sample output screen of this selection.

This selection is useful to verify that a custodian owns property and to see the way their name is kept on file. The names are shown a screen at a time alphabetically. Follow the prompts at the bottom of the screen to continue through the listings, or to exit and return to the Listing and Searches Menu.
4. Components of a single Manufacturer

This selection allows you to get a listing of all components on file for a single manufacturer. When prompted, type in the mfg and a list of components will be provided as shown in Figure C.7. The list gives you the model, description, custodian, and the location of each component on file for this particular mfg.

The components are listed one screen at a time, you can continue through the entire list or return to the List and Search menu by following the instructions at the bottom of the screen.

5. Components of a single Model

This selection allows you to get a listing of all components on file for a single model. This selection is similar to that of the list of components by manufacturer. The same information is provided, but the list may include more than one mfg.

c. Property Reports

There are two formatted reports provided by the Property Management System, a Quarterly Report and a Summary Report. To reach the Print Reports Menu, enter a "2" at the Main Menu.

Figure C.6 Custodian Listing Screen.
I. Quarterly Report

This report is a three part report used for verification of the status of the property on file.

1. The component report - this will give a listing of all components sorted by custodian. It will provide the mfg, model, description, nfg serial #, and the location of each component.

2. The part to component report - this will give a listing of all parts assigned to a component sorted by custodian and location. It will provide the model, serial #, and description of each part assigned to a particular component.

3. The stock part report - this will provide a list of all parts in stock that have not been assigned to a component.

Figure C.8 shows the screen after selecting the Quarterly Reports option. You will be prompted to turn on the printer as the files are being combined to print the report. Once the files are ready you will be asked if the printer is ready, answer the appropriate response to the question at the bottom of the screen. If you are not ready the system will wait until you are, or you decide to abandon the report.

* NOTE: You can exit the Quarterly Report selection at any time as long as the system is not told the printer is ready and it really is not. This may cause a lock up, and you will have to re-start the program, no damage should occur by
shutting off the computer but it is recommended that you avoid this if at all possible. To exit without completion of the report, type "ESCape", this will abandon the reports and return you to the Reports Menu.

2. **Property Summary Report**

The Property Summary report is also a three part report. These reports are to be used for inventory purposes to maintain accountability of department property, providing a listing of property on file sorted by Property Type and Property Numbers.

1. The component report - this will give a listing of all components sorted by property type and number. It will provide the mfg, model, mfg serial #, custodian and the location of each component.

2. The part to component report - this will give a listing of all parts assigned to a component sorted by property type and number. It will furnish the model, description, custodian, and location of each part assigned to a particular component.

3. The stock part report - this will provide a list of all parts in stock that have not been assigned to a component, sorted by property type and number.

The procedure for printing the Property Summary report is the same as that of the Quarterly report. See the directions in that section if you have any doubt in generating these reports.
d. Enter New Property

Property is entered as a part or a component. These are the two selections available from the Enter Property Menu. Parts are items such as cards, boards or hard disks that are used in components.

1. Component Entry

Figure C.9 shows a typical component entry. Enter the appropriate information as the fields appear. Once a field entry is made you may need to depress the return key to progress to the next entry. The designated use and property type fields are the exceptions. These require only entering the first letter of the choices displayed. After all entries are made you will be given the opportunity to correct mistakes. It is important that you review each field to ensure they are correct. Once you tell the system that the entries are correct, this component is placed on file. If for some reason an improper component record is placed on file you may use the modify or delete selections from the Main Menu to correct or delete the entry.

<table>
<thead>
<tr>
<th>COMPONENT ENTRY SCREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter Component Information:</td>
</tr>
<tr>
<td>date: 06/26/87</td>
</tr>
<tr>
<td>mfg: IBM</td>
</tr>
<tr>
<td>model: PC XT</td>
</tr>
<tr>
<td>serial #: 123456789012345</td>
</tr>
<tr>
<td>description: 256K MAIN MEMORY WITH TWO FLOPPY DRIVES</td>
</tr>
<tr>
<td>designated use: 0 (Office / Lab / Storage / Home)</td>
</tr>
<tr>
<td>Custodian</td>
</tr>
<tr>
<td>last name: BUI</td>
</tr>
<tr>
<td>first name: TUNG</td>
</tr>
<tr>
<td>office: (1-316)</td>
</tr>
<tr>
<td>property type: P (Plant / Minor / Other)</td>
</tr>
<tr>
<td>property #: 1234567899</td>
</tr>
<tr>
<td>price: $1,234.00</td>
</tr>
<tr>
<td>reqn #: 7123-7199/R75C1</td>
</tr>
<tr>
<td>Is the above information correct?:</td>
</tr>
<tr>
<td>[ Yes / No / Abandon ]</td>
</tr>
</tbody>
</table>

Figure C.9 Component Entry Screen.

Several of the fields are required entries. Mfg is required or you will return to the Enter Property Menu. Other required entries are serial #, designated use, and
the property type. Dependent upon choices associated with these entries, several other entries may be required.

The serial # field is for the mfg serial #, this is a very important and required entry. Care must be taken when entering the serial # since this is the field that maintains a components identity, making it unique. Ensure this entry is correct.

There are four categories of component use. Storage components are assigned to the AS DEPT and will require no further related entries. Lab components are also assigned to the AS DEPT, and you are required to choose one of the four department lab locations. Office and Home use are used for custodian assignments. You will be required to enter the custodians last name, and if the custodian is on file you will need not enter any further custodian information. If they are not on file then you will be required to enter a first name and the appropriate location information. To take care of the situation of two custodians with the same last name, you will be requested to verify the custodian that is on file. This should also prevent entering a custodian twice with two different variations of their name. Remember however that it is possible to have two custodians with the same last name. No exception is made for two custodians with the same last and first names, if this situation ever occurs the only way to distinguish them will be on location.

There are three property type classifications. "P" is for plant property, "M" is for minor property, and "O" is for other than plant or minor property. Plant and Minor property will require an associated property number. This is also an important field, which will provide the accountability of the department property within NPS guidelines. Care should be taken when entering this field.

* NOTE: If the situation ever arises that you do not know the proper information for a required field entry, do not worry. Simply enter any accepted entry, and abandon the entry when you reach the point in Figure C.9. On the other hand, if an incorrect choice was made at some point, you may make the appropriate corrections instead of abandoning the entry. Care should be taken never to file a false or incorrect record.

2. Part Entry

Figure C.10 shows a typical part entry. As with the component entry the entering procedures are the same. To enter a part, you first enter the model which is required, then the part serial # and description, both are optional. At this point you are asked if this part is for stock or to be used in a component. Stock entries are assigned
to the AS DEPT and require no related entries. A part assigned to a component will require that the component serial # be entered. These two entries are required. Property type is also a required entry. Plant and Minor property will also require a property number. Price and reqn # are optional entries.

![PART ENTRY SCREEN](image)

### Enter Part Information:
- **model:** CARD
- **serial #:** 12345
- **description:** COLOR GRAPHICS CARD FOR AN IBM PC
- **designated use:** C (Storage / Component)
- **Component serial #:** 123456789012345
- **property type:** 0 (Plant / Minor / Other)
- **price:** $123.00
- **reqn #:** 

**Is the above information correct?:** [Yes / No / Abandon]

Figure C.10 Part Entry Screen.

* NOTE: It is important that the component serial # is accurate. A search of all components on file will let you know if this component is not on file. If not on file and the serial # is correct you must enter the component first. If you entered it incorrectly you will be allowed to re-enter it.

e. **Delete Property**

To delete property select this option from the Main Menu. This will display the Delete Property Menu, you may delete either a part or a component.

1. **Component Deletion**

To delete a component you must know the component serial #. Enter the proper serial # and you will be shown the component on file with this serial # (Figure C.11). You must verify that this is the correct component. If it is not then check the serial #, no two components should have the same serial #. If the component is correct then the files are checked to see if parts are on file for this component.
Figure C.11 Component Deletion Screen.

If parts are on file for this component, you cannot delete this component if you do not wish to delete the parts. You will have to reassign the parts using the modify selection from the Main Menu. If there are no parts, or you want to delete the parts as well, then answer yes when you are questioned if you wish to delete this component. Be careful when deleting components and parts, once deleted there is no means of recovering this information.

2. Part Deletion

Since parts can be assigned to stock or to a component, to delete a part you must know how a part is being used to delete it. A component part requires that you know the serial # of that component. Enter the component serial #, like a component deletion the files are searched to see if this component is on file. You will have to verify that the component is correct to continue.

If the component is found or if this is a stock part, you are then asked to fill in if known: the part model, part serial #, part property type, or part property number. If you do not know the proper information depress the return key. Depending on the information provided a list of parts will be displayed for you one at a time to determine if it is the correct part or not (Figure C.12).
PART DELETION SCREEN

Only one part on file:

model: CARD
description: COLOR GRAPHICS CARD FOR AN IBM PC
property type: Other

Is this the correct part?: [ Yes / No ]

date: 06/26/87

Figure C.12 Part Deletion Screen.

If no parts are found meeting these constraints then a message will tell you that your part cannot be found and may not be on file. Check the property reports to see if your part information is indeed correct, if so the part probably was previously deleted.

1. **Modify Property**

To modify a property record (either part or component), select this option from the Main Menu. A menu screen will appear for you to choose from the available options (Figure C.13). The selections are explained under the next 3 sub-sections.

Component records have two different modify operations. If you wish to reassign a component to a new custodian or location select option “1”. To modify the other than assignment fields, such as price or property number, choose selection “2”. The modify record option will be used to correct mistakes after entry. There is only one modify option for parts, which allows reassigning the part or changing all fields.

1. **Modify Component Custodian or Location**

Selecting this option from the Modify Menu you must enter the component serial # or return to the Modify Menu. If the component is on file you will be asked if this is the correct component (Figure C.14).
Modify Property Menu

Modify COMPONENT
1 - Custodian or Location
2 - Record

Modify PART
3 - Accountability or Record
H - HELP
0 - RETURN to MAIN MENU

selection :

Figure C.13  Modify Property Screen.

MODIFY COMPONENT ASSIGNMENT
date: 06/27/87

mfg: IBM
model: PC XT
serial #: 123456789012345
description: 256K MAIN MEMORY WITH TWO FLOPPY DRIVES

property type: Plant
property #: 1234567890

designated use: Office

Custodian: BUI, TUNG
office: (1-316)

Is this the correct component?: :
[ Yes / No ]

Figure C.14  Modify Component Assignment Screen.

If this is the correct component you may then modify the designated use and reassign the component to a new custodian, location, or both. Figure C.15 is a sample reassignment.
**Figure C.15** Sample Component Reassignment.

If the modifications are correct then you may file this component record with the new changes. If you made a mistake you may try again or abandon the modifications with no changes having been made.

2. **Modify Component Record**

Selecting this option from the Modify Menu you must enter the component serial # or return to the Modify Menu. If the component is on file you will be asked if this is the correct component. Figure C.16 is a representative display screen and will be used as an example for this section.

If this is the correct component you may then modify the fields not associated to the designated use, custodian, or location. Figure C.17 is a sample modification, note the difference in the component description. Each field that modifications are allowed, will be displayed one at a time with the original information. Make the required changes or hit the return key to leave the original entry as is.

If the modifications are correct then you may file this component record with the new changes. If you made a mistake you may try again or abandon the modifications with no changes having been made.
<table>
<thead>
<tr>
<th>Custodian</th>
<th>BUI, TUNG</th>
<th>date: 06/27/87</th>
</tr>
</thead>
<tbody>
<tr>
<td>designated use</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>mfg</td>
<td>IBM</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>PC XT</td>
<td></td>
</tr>
<tr>
<td>serial #</td>
<td>123456789012345</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>256K MAIN MEMORY WITH TWO FLOPPY DRIVES</td>
<td></td>
</tr>
<tr>
<td>property type</td>
<td>Plant</td>
<td></td>
</tr>
<tr>
<td>property #</td>
<td>1234567890</td>
<td></td>
</tr>
<tr>
<td>price</td>
<td>$ 1234.00</td>
<td></td>
</tr>
<tr>
<td>reqn #</td>
<td>7123-7199/R7SC1</td>
<td></td>
</tr>
</tbody>
</table>

Is this the correct component?: [Yes / No]

Figure C.16  Modify Component Screen.

<table>
<thead>
<tr>
<th>Custodian</th>
<th>BUI, TUNG</th>
<th>date: 06/27/87</th>
</tr>
</thead>
<tbody>
<tr>
<td>designated use</td>
<td>Office</td>
<td></td>
</tr>
<tr>
<td>mfg</td>
<td>IBM</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>PC XT</td>
<td></td>
</tr>
<tr>
<td>serial #</td>
<td>123456789012345</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>640K MAIN MEMORY, 10MB HARD DISK</td>
<td></td>
</tr>
<tr>
<td>property type</td>
<td>P (Plant / Minor / Other)</td>
<td></td>
</tr>
<tr>
<td>property #</td>
<td>1234567890</td>
<td></td>
</tr>
<tr>
<td>price</td>
<td>$ 1,234.00</td>
<td></td>
</tr>
<tr>
<td>reqn #</td>
<td>7123-7199/R7SC1</td>
<td></td>
</tr>
</tbody>
</table>

Are the modifications correct?: [Yes / No / Abandon]

Figure C.17  Sample Component Modification.

146
3. Modify Part Accountability or Record

Parts are assigned usage to stock or to components. If this is known then enter the appropriate entry. To allow for the occasion when this is not known, such as a mistaken entry, an additional option is given for part modifications. Figure C.18 displays the initial Modify Part Screen.

![MODIFY PART SCREEN](image)

Figure C.18 Modify Part Screen.

If the part is assigned a component then you are asked to enter the component serial #. If the component is found, or if this is a stock part, or if the usage is not known, you are then asked to fill in if known: the part model, part serial #, part property type or part property number. If you do not know the proper information enter a return. Depending on the information provided, a list of parts will be displayed for you one at a time to determine if this is the correct part or not (Figure C.19).

If this is the correct part, you may then make the required changes. Each field is presented to you, one at a time, with the original entry. Make the appropriate modifications, or hit the enter key to leave the entry unchanged. Once all the fields have been presented you must reply that they are correct. If the changes are not
Figure C.19  Sample Part Modification.

correct then you can re-edit them or abandon the modifications leaving the original part record as first entered. No changes will be filed until you wish to file them.

3. SPECIAL OPERATIONS

This section will cover operations not covered in the previous sections. The operations covered in this section will help maintain the system, and provide for ease of use.

   a. General Editing

   Normally the keyboard is in the overwrite mode. You may depress the INSert key to allow inserting characters if you wish. You must be careful when using the backspace or arrow keys, it is possible to find yourself outside of the entry field. This really does not cause a problem, but may be a little inconvenient, requiring you to circle around to that field again. There is no reason to use the Pg Up or Pg Dn keys, these will definitely take you outside the field of interest. The only exception to these problems are when you are forced to make a selected entry (eg. Yes / No).
b. **ESCAPE**

This key allows you to return to a previous menu in many instances. Do not use this key if you are not given the option to do so. Using this key indiscriminantly may leave files open, possibly causing errors. Using ESCape when not an option may also cause you to use the enter key when not normally required.

c. **Printing**

Printing is normally accomplished with the Print Reports option from the Main Menu. Ensure there is paper in the printer and that the printer is turned on. It is possible to get a printing of a particular screen by depressing the shift key and PrtSc key simultaneously. Avoid doing this if the printer is not set up for printing.

d. **Backups**

No system is infallible, therefore a system backup is to be maintained by the department supervisor. Additionally the files must be backed up so that the information need not be re-entered if there is some sort of failure. The data files will be copied automatically if the system is exited normally from the Main Menu. The entire application cannot be copied onto a single disk, so the database (dbf) files will only be copied. In case of a loss of database files, the supervisor will see to it that a knowledgeable dBase III plus programmer re-create the system to normal operation.

e. **Exiting**

To exit any menu enter a "0", this will return you to the calling menu. At the Main Menu this will return you to the microcomputer operating system (MS DOS) after making copies of the database files (the system boot-up disk must be in the floppy drive). This is the normal procedure for system exiting, you should not turn off the computer until returned to the MS DOS prompt: C >.
LIST OF REFERENCES


## INITIAL DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Address</th>
</tr>
</thead>
</table>
| 1.  | 2      | Defense Technical Information Center  
        Cameron Station  
        Alexandria, VA 22304-6145                                           |
| 2.  | 2      | Library, Code 0142  
        Naval Postgraduate School  
        Monterey, CA 93943-5002                                             |
| 3.  | 2      | Computer Technology Programs, Code 37  
        Naval Postgraduate School  
        Monterey, CA 93943-5000                                             |
| 4.  | 2      | Department Chairman, Code 54  
        Naval Postgraduate School  
        Monterey, CA 93943-5004                                             |
| 5.  | 1      | Associate Professor Tung Bui, Code 54BD  
        Naval Postgraduate School  
        Monterey, CA 93943-5004                                             |
| 6.  | 1      | LT. Timothy M. Sexton  
        210 Johnson Ave  
        Sayville, NY 11782                                                  |
END
DATE
FILMED
FEB.
1988