AD-A257 837

NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
DEC 0 8 1992
S A D

THESIS

IMPLEMENTATION OF A CONFIGURATION AND MAINTENANCE MANAGEMENT SYSTEM FOR LOCAL AREA NETWORKS

by

David G. Dickison

September 1992

Thesis Advisor: Norman F. Schneidewind

Approved for public release; distribution is unlimited

92-31053

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
IMPLEMENTATION OF A CONFIGURATION AND MAINTENANCE MANAGEMENT SYSTEM FOR LOCAL AREA NETWORKS

12 PERSONAL AUTHOR(S)
Dickison, David G.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	September 1992	152

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	LAN, Database, LAN Management, Configuration Management

19 ABSTRACT (Continue on reverse if necessary and identify by block number)
The management of Local Area Networks today is one of the most challenging tasks which face a system manager. The plethora of system software, application software and accessories available combined with the various computer clones, and open architecture present in microcomputers makes tracking Local Area Network configuration and maintenance a very daunting task. This thesis modifies a previous design for a Local Area Network configuration and maintenance database and implements the new design in Dbase IV version 1.1 using the Naval Postgraduate School Administrative Science Department Local Area Newtworks as a prototype database. The database was designed to be simple to use, to protect data integrity and to be expandable to new equipment, new technology and new data applications. Due to its general nature, the Local Area Network Maintenance and Configuration System can be used on virtually any Local Area Network application.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Norman F. Schneidewind	22b TELEPHONE (Include Area Code) (408)646-2494 / 22c OFFICE SYMBOL As/Ss

DD Form 1473, JUN 86 *Previous editions are obsolete*
S/N 0102-LF-014-6603

Approved for public release; distribution is unlimited.

Implementation of a Configuration and
Maintenance Management System for
Local Area Networks

by

David G. Dickison
Lieutenant, United States Navy
B.S., University of Kentucky, 1983

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1992

Author: _____
David G. Dickison

Approved By: _____
Norman F. Schneidewind, Thesis Advisor

_____
Myung W. Suh, Second Reader

_____
David R. Whipple, Chairman
Department of Administrative Sciences

ABSTRACT

The management of Local Area Networks today is one of the most challenging tasks which face a system manager. The plethora of system software, application software and accessories available combined with the various computer clones, and open architecture present in microcomputers makes tracking Local Area Network configuration and maintenance a very daunting task. This thesis modifies a previous design for a Local Area Network configuration and maintenance database and implements the new design in dBASE IV version 1.1 using the Naval Postgraduate School Administrative Science Department Local Area Networks as a prototype database. The database was designed to be simple to use, to protect data integrity and to be expandable to new equipment, new technology and new data applications. Due to its general nature, the Local Area Network Maintenance and Configuration System can be used on virtually any Local Area Network application.

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DTIC QUALITY INSPECTED 3

TABLE OF CONTENTS

I. INTRODUCTION 1

A. TERMINOLOGY 1

B. THE LAN CONFIGURATION AND MAINTENANCE PROBLEM . 1

C. A LAN MAINTENANCE AND CONFIGURATION DATABASE SYSTEM 3

D. PURPOSE OF THIS THESIS 4

II. SYSTEM DESIGN 5

A. THE OLD DESIGN 5

1. Cable Plant Equipment 9

2. Maintaining Referential Integrity 9

3. The Concept of a System Unit Object 11

B. THE NEW DESIGN 12

1. The New Drive Object 17

2. Changes to Data Fields 17

3. System unit and Node Objects Revisited . . 18

III. IMPLEMENTATION 20

A. METHODOLOGY 20

1. Implementation Goals 20

a. Data Integrity 20

b. User-System Interface 21

c. System Flexibility 22
d. System Expandability 22
e. System Maintenance 24
2. Data and Object Access 24
3. Security Issues 25
4. Main Menu Organization 27
B. THE DATABASE TOOL 29
C. THE SYSTEM PROTOTYPE 29

IV. CONCLUSIONS AND LESSONS LEARNED 30
A. CONCLUSIONS 30
1. Satisfaction of Requirements 30
2. Database Tool Effectiveness 30
B. LESSONS LEARNED 31
1. Coding Requirements 31
2. Requirements Changes 31
3. User-Designer Communication 32

V. RECOMMENDATIONS FOR FURTHER SYSTEM ENHANCEMENTS . . 34
A. ADDITION OF A MAINTENANCE REPORT OBJECT . . . 34
B. SQL COMMAND LIBRARY 34
C. IMPLEMENTATION OF A FRONT END EXPERT SYSTEM . . 35

APPENDIX A 36
A. GENERAL OPERATION 36
1. Moving around in the Application Main Menu 36

2. Using Pop Up Menus 36
3. Data Entry on Formatted Data Record Screens 37
B. THE LAN MENU 38
1. Nodes Submenu 39
2. The System Unit Submenu 39
3. The Drives Submenu 40
4. The Accessories Submenu 40
5. The Application Software Submenu 40
6. The System Software Submenu 41
7. The Cable Plant Submenu 41
C. THE SPARES MENU 42
1. The System Unit Submenu 43
2. The Drives Submenu 44
3. The Accessories Submenu 44
4. The Application Software Function 44
5. The System Software Function 44
6. The Cable Plant Function 45
D. THE REPORTS MENU 45
1. The LAN Submenu 45
2. The LAN Equipment Submenu 46
3. The Spares Submenu 46
E. THE QUIT MENU 46
1. The Maintenance Submenu 47
2. The SQL Function 47
3. The Quit to DOS Function 47
4. The Quit to Control Center Function 47

F. ADDING AND DELETING KEY FIELD VALUES 48
G. RETURNING TO THE APPLICATION FROM THE CONTROL CENTER 49
H. SUMMARY 50

APPENDIX B-SYSTEM MENUS 51

APPENDIX C-DATA ENTRY SCREENS 61

APPENDIX D-RELATION DEFINITIONS 64

APPENDIX E-PROGRAM FILES AND DEFINITIONS 67
A. DATABASE FILES 67
B. MENU FILES 68
C. DATA ENTRY FORMAT FILES 69
D. REPORT FORMAT FILES 70
E. SYSTEM APPLICATION FILES 71

APPENDIX F- APPLICATION PROGRAM FILES 73
1. Acc_Ed 73
2. Acc_ent 74
3. Acc_del 75
4. Acc_S_Ed 77
5. Acc_S_Del 78
6. Ac_S_Ent 79
7. Add_app 80

8. Add_Node 81
9. Add_Sys 82
10. App_Del 83
11. App_Ent 84
12. App_Spa 86
13. Cabl_Add 87
14. Cabl_Del 88
15. Cabl_Sp 90
16. Driv_Del 90
17. Drive_Ent 92
18. Drv_Quer 93
19. Dr_S_Del 94
20. Dr_S_Ent 96
21. Dr_S_Que 97
22. Get_Unit 97
23. Nodent_1 100
24. Nodent_2 103
25. Node_Ch 107
26. Node_Del 110
27. Node_Ed 111
28. Node_Que 112
29. N_Choice 113
30. Spare_Qu 114
31. Sys_Del 115
32. Sys_Ent 117
33. Sys_Spa 119

34. Unit_2 120
35. Unit_Add 123
36. Unit_Ed 126
37. Unit_Ent 128
38. Unit_Qu 131
39. Unit_Rem 133
40. Un_Sp_Dl 134
41. Un_Sp_Ed 136

LIST OF REFERENCES 139

BIBLIOGRAPHY 140

INITIAL DISTRIBUTION LIST 141

I. INTRODUCTION

A. TERMINOLOGY

It is assumed that the reader is conversant with the terms associated with microcomputers and Local Area Networks. In the following discussions, a **node** refers to a station on a network. A **node** is usually a microcomputer but can be a peripheral device such as a printer. A **server** refers to a microcomputer node which is responsible for supplying services, such as application software or printing, to the other computers on the LAN. A **user** refers to a computer node which receives services from a **server**. A **user/server** refers to a computer node which acts as a server to other nodes on the network, but can also function as an end **user**.

B. THE LAN CONFIGURATION AND MAINTENANCE PROBLEM

Local area networks have gained widespread use and acceptance in business and education. They offer economy of scale by allowing users to share accessories such as printers, plotters and network bridges. For the system manager, a Local Area Network (LAN) also offers the opportunity to enforce organization software standards, something not easily enforceable with stand alone microcomputers. Unfortunately along with these advantages, local area networks can create

interoperability problems which seldom occur with stand alone microcomputers.

Microcomputer clones from different vendors will often use different interrupt levels for their drives and accessories. While this is not usually a problem for a stand alone computer, it may cause serious compatibility problems when linking many computers from different vendors on the same network. Local area network boards often can only operate on one or two interrupt settings. If these settings are already taken by another card, drive or accessory, a serious compatibility problem may arise. For this reason, on Local Area Networks, it is vital to keep track of what interrupts are being used by which devices on each computer on the LAN.

The compatibility problem aside, documentation of the configuration of each node on a LAN can be a monumental task, especially on larger LANs. It is conceivable that every node on a LAN could have a different configuration with various accessories, system software, application software and disk drives. In order to effectively and efficiently maintain a LAN and its associated hardware, the configuration records must be readily available for each node and available spare parts documented. In summary, to effect proper maintenance and avoid configuration compatibility problems, it is vital for the LAN manager to have at his disposal detailed records for each node on a LAN regarding the node's drives,

accessories, and application and system software and to have a detailed record of spare parts and hardware.

C. A LAN MAINTENANCE AND CONFIGURATION DATABASE SYSTEM

The maintenance and configuration problem stated above can be effectively tackled by implementing a relational database application through database development tools. The dBASE IV Version 1.1 database development tool was chosen to implement this particular application for the following reasons:

- The dBASE IV 1.1 tool was a mature technology which was familiar to a large body of database developers. Thus it would probably be easier to maintain than some lesser used tools or languages.
- dBASE IV 1.1 can be installed on the LAN or LANs to be documented. It is compatible with most operating systems and network operating systems.
- The dBASE IV 1.1 tool was available on one of the prototype LANs.
- The author was conversant with the dBASE IV 1.1 tool.

The system design was modified from a design from Brewer [Ref. 1]. It was desired to modify the original design as little as possible as this project was primarily an implementation. However further analysis of requirements and design implementation revealed that the Brewer design had shortcomings which required modification in order to improve the flexibility, usability and adaptability of the system. These shortcomings will be discussed later in detail.

D. PURPOSE OF THIS THESIS

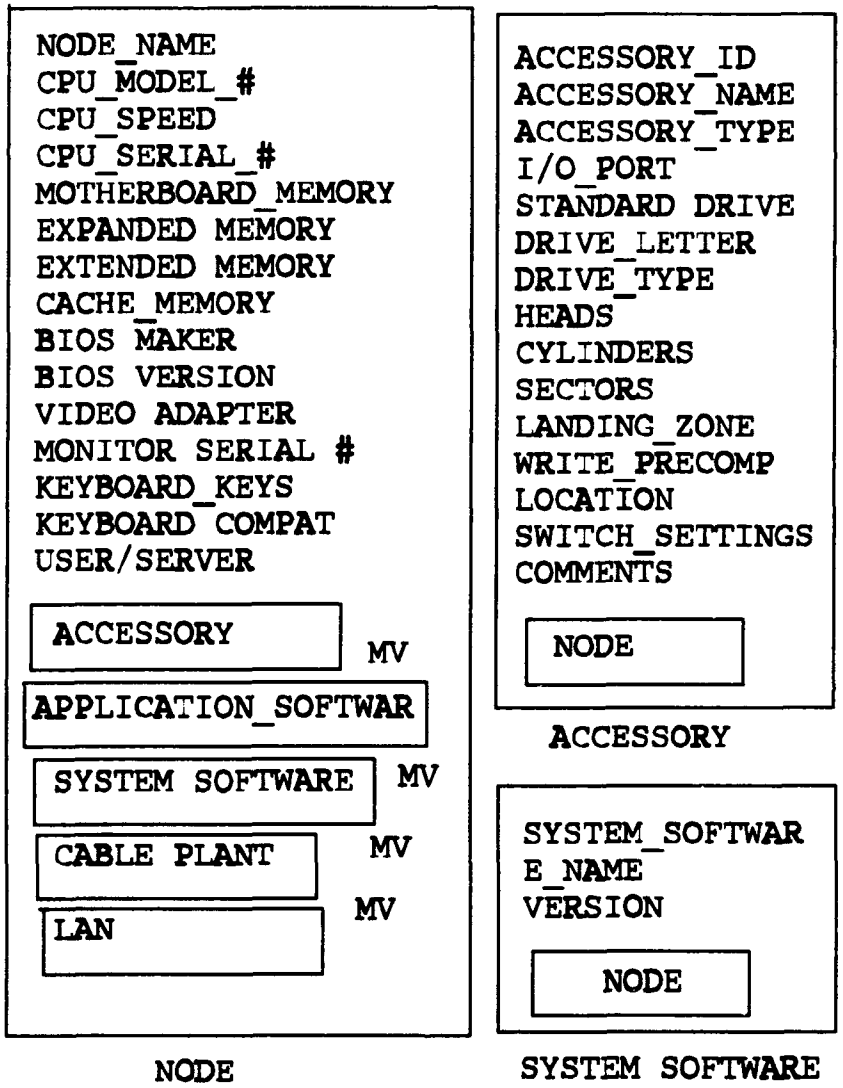
The main purpose of the work was to effectively implement a LAN maintenance and configuration system which could be run on a LAN. This task was successfully completed. Personnel training on the system was also conducted and a users guide created to teach systems managers about system capabilities. The following topics and themes will be treated in the body of the work:

- The rationale behind the changes made to the original system design.
- The features, functions and applications which the implementation is capable of performing.
- The design philosophy which drove the redesign and implementation of the system.
- The effectiveness of the dBASE IV tool in implementing the database design.
- Lessons learned and difficulties encountered not related to the dBASE IV 1.1 tool.
- Possible follow on improvements and applications.

II. SYSTEM DESIGN

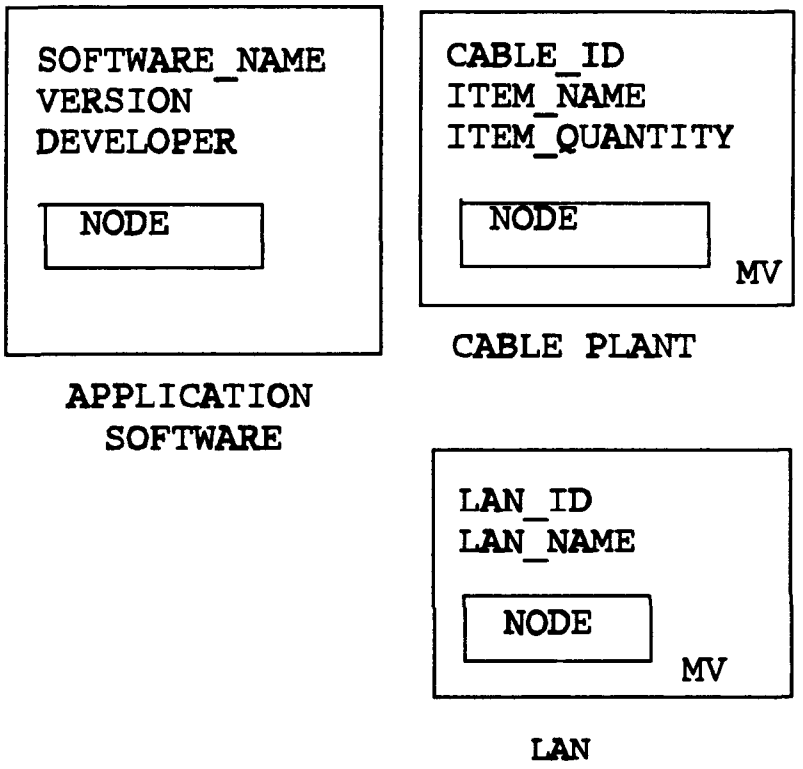
A. THE OLD DESIGN

The object and relationship diagrams for the design created by Brewer [Ref. 1] are shown on the following pages as figures 1.1, 1.2 and 1.3 respectively. This design, which was a modification of a previous design by Suriano [Ref. 2] had many good qualities. The design used an object oriented approach as proposed by Kroenke and Dolan [Ref. 3] to develop objects and relations among the various objects. It also created many useful fields for the objects which enabled the user to enter a great deal of information on hardware types and settings which would aid the system manager in configuration decisions. While most of the fields proposed were implemented into the new design, many of the objects and relationships required changes. Interviews were conducted with Professor N. Schneidewind and Leon Sahlman [Ref. 4], the users and system managers of the Administrative Science Department LANs. The main purpose of the interviews was to verify system requirements and address any design problems. In the course of the discussions, several justifiable new requirements and design problems were identified.



MV: MULTI VALUE RELATIONSHEIP

FIGURE 1.1 -OLD OBJECT DIAGRAMS



MV: MULTI VALUE RELATIONSHIP

FIGURE 1.2

OLD OBJECT DIAGRAMS (CONT)

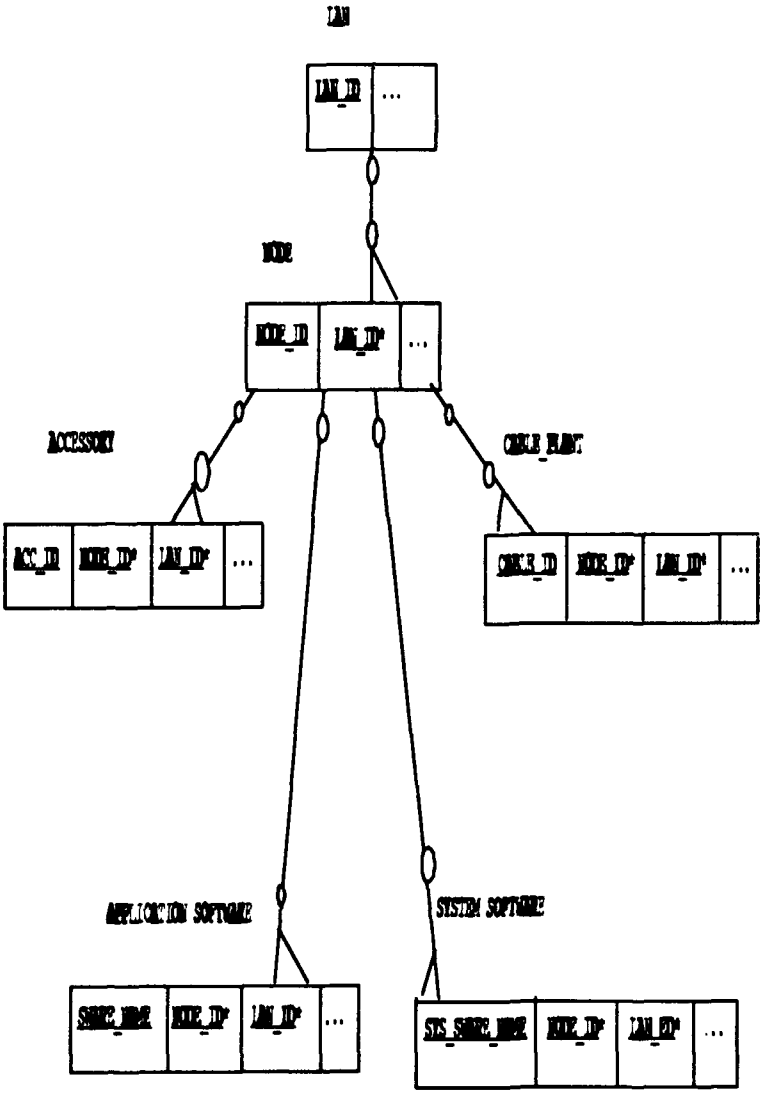


FIGURE 1.3- OLD RELATIONSHIP DIAGRAM

1. Cable Plant Equipment

Cable Plant Equipment within this discussion refers to LAN equipment or hardware which is used to connect nodes and peripheral equipment or to supply power to nodes and peripheral equipment. An example would be the Multi Access Units (MAU) used on the IBM token ring network to connect nodes into a logical ring/physical star configuration. Brewer's design had cable plant equipment as a multi-value relation of the node object. While some of the cable plant equipment could be related to nodes in a LAN, much of the cable plant equipment, multi access units for example, could not be related to a single node. Therefore it is more appropriate to treat cable plant equipment as a relation to the LAN object under which all cable plant equipment could be associated.

2. Maintaining Referential Integrity

Brewer treats the accessory, cable plant, application software and system software relations with the node object as one to many. Although this design can certainly be effectively implemented, the one to many relationship does not fully utilize the data control and display capabilities of the dBASE IV 1.1 database tool. Additionally, the one to many relationships allow for the possibility of modification anomalies in the data.

In order to help enforce data integrity, it is desirable in any database application to limit the data choices which the user may enter into the database. This is especially critical if the field in question is a key field. A slight typographical error on the user's part in the key field can prevent the system from locating the desired record. Using a many to many relationship approach, all the desired values for a particular key field can be stored in a database file. Then any time a value must be entered into the field, the acceptable field values can be displayed using a pop up menu. This allows only the desired field values to be entered into the database and greatly increases the data integrity of the database.

A further advantage from a design interface standpoint is that the user does not have to remember data fields. The user only has to recognize the desired field from a pop up menu. In this way the user friendliness of the system-user interface is increased.

It is also desirable to have the possible choices for key data fields on record in the database for query purposes even if there are no records in the database presently using the given field value. This serves to avoid modification anomalies involving key fields.

3. The Concept of a System Unit Object

During discussion with the users, it became evident that a new object needed to be added to the database. The users viewed the actual microcomputers, or **system units** as a different entity than the nodes on the LAN. A node was viewed as a position on the LAN. This position or node could be a microcomputer (system unit) or as in the case of the Apple Talk LAN, even a printer. A node, by definition had to be associated with a particular LAN. A node could not be moved from one LAN to another without changing the node name. This made the present node name useless as a key field. Additionally, the users found it desirable at times to query the node equipment by the system unit serial number. The system unit hardware was identified by the hardware serial number whereas the node was identified by its parent LAN and its purpose in the network. To further confuse the issue, faulty system units assigned to nodes were occasionally replaced with different system units which changed the system unit assigned while leaving the node name the same.

The approach taken to solving this problem was to break the old **node** object into two new objects, a **new node** object and a new **system unit** object. The new node object kept its many to one relationship with the LAN object and described the node's function or purpose in the network. This new node object also had a one to one relationship with the new system unit object which contained all the hardware related

microcomputer attributes. This arrangement, although adding complexity to the database, had the great advantage of allowing nodes to be used as key fields and to remain with their respective LANs. Meanwhile, system units, the actual hardware, could be moved from LAN to LAN or be completely removed from any LAN and still be effectively tracked in the database. Thus in the new approach, the node is viewed as a virtual part of a network which can take on hardware (system unit) and software attributes.

B. THE NEW DESIGN

Figures 2.1 through 2.4 on the following pages give the object and relationship diagrams for the new modified design. The main design changes involve the following:

- Making the cable plant object a relation of the LAN object vice that of the node object.
- The old node object was broken down into two new objects: a new node object and a system unit object.
- The cable plant and LAN objects were associated in a many to many relationship.
- The two software objects were associated in a many to many relationship with the new node object respectively.
- The accessory object was given a many to many relationship with the system unit object.
- The accessory object was broken down into two new objects: a new accessory object and a drive object.

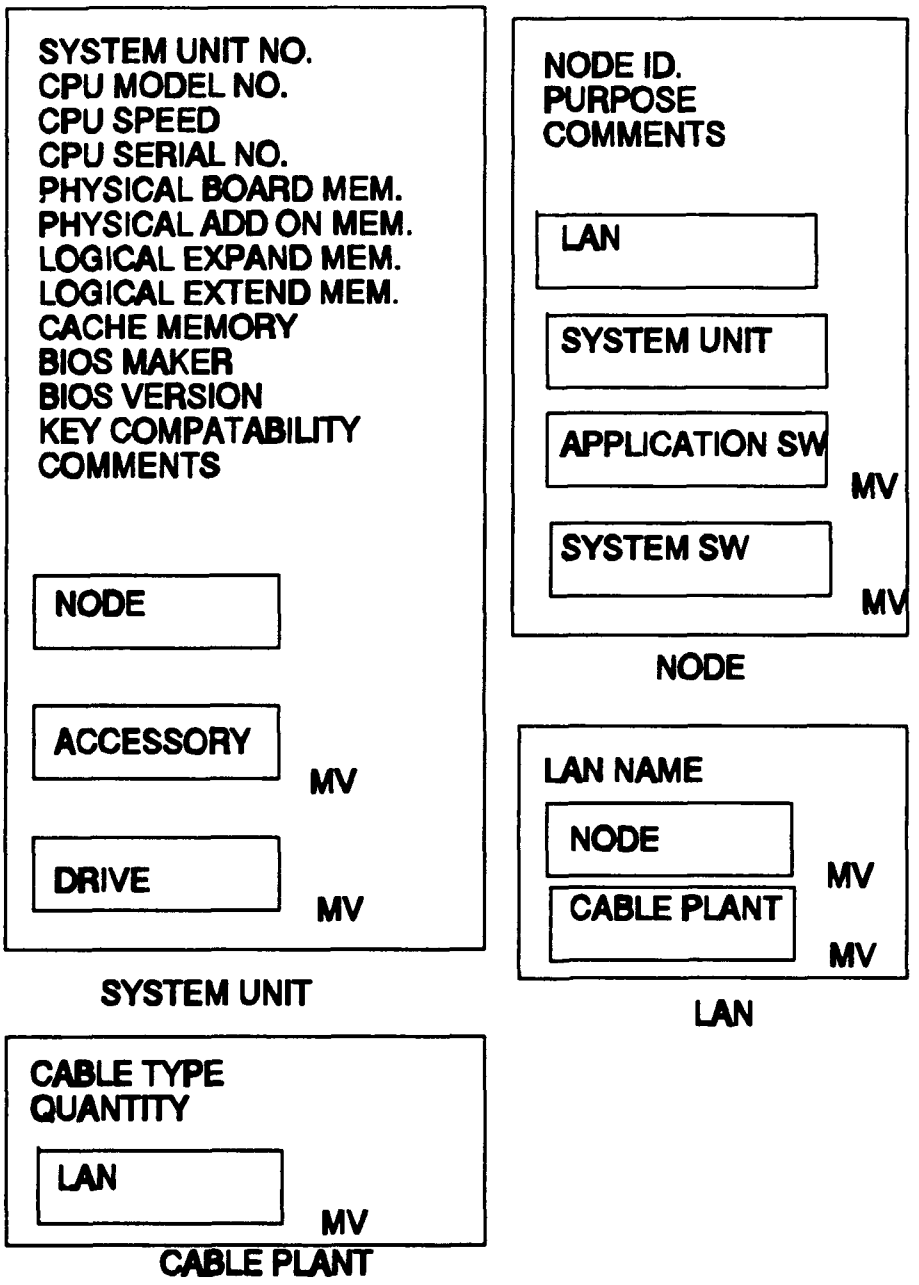


FIGURE 2.1 -NEW OBJECT DIAGRAMS

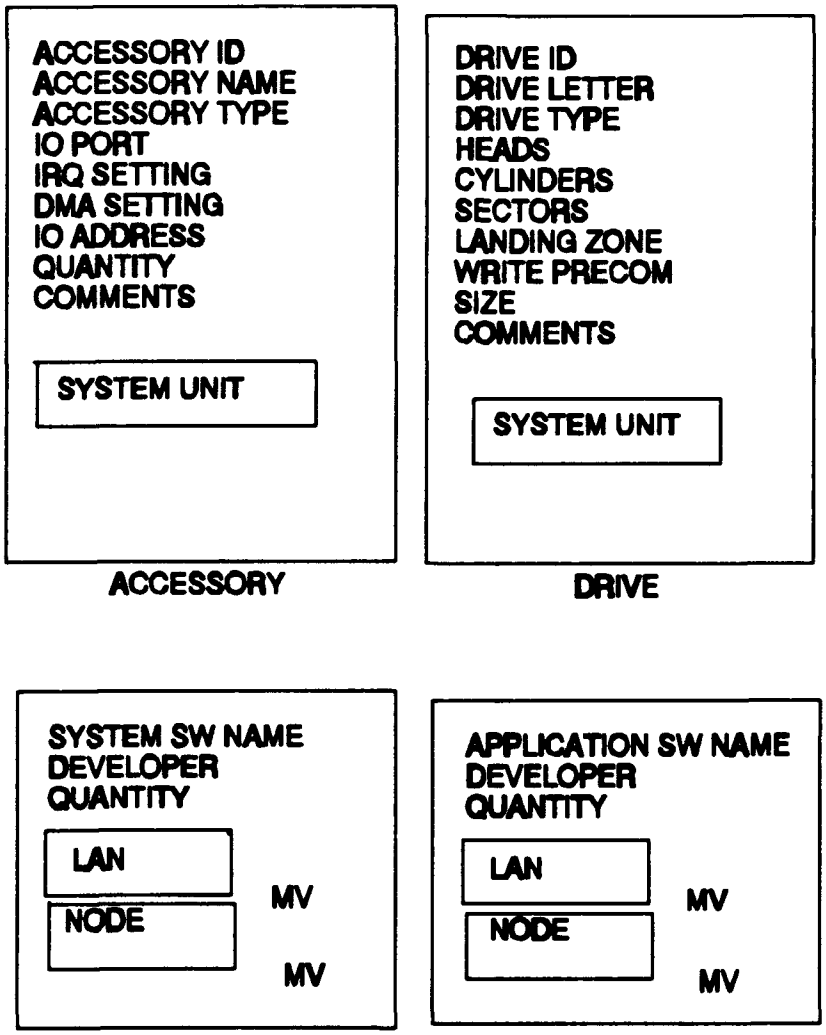


FIGURE 2.2 -NEW OBJECT DIAGRAMS (CONT)

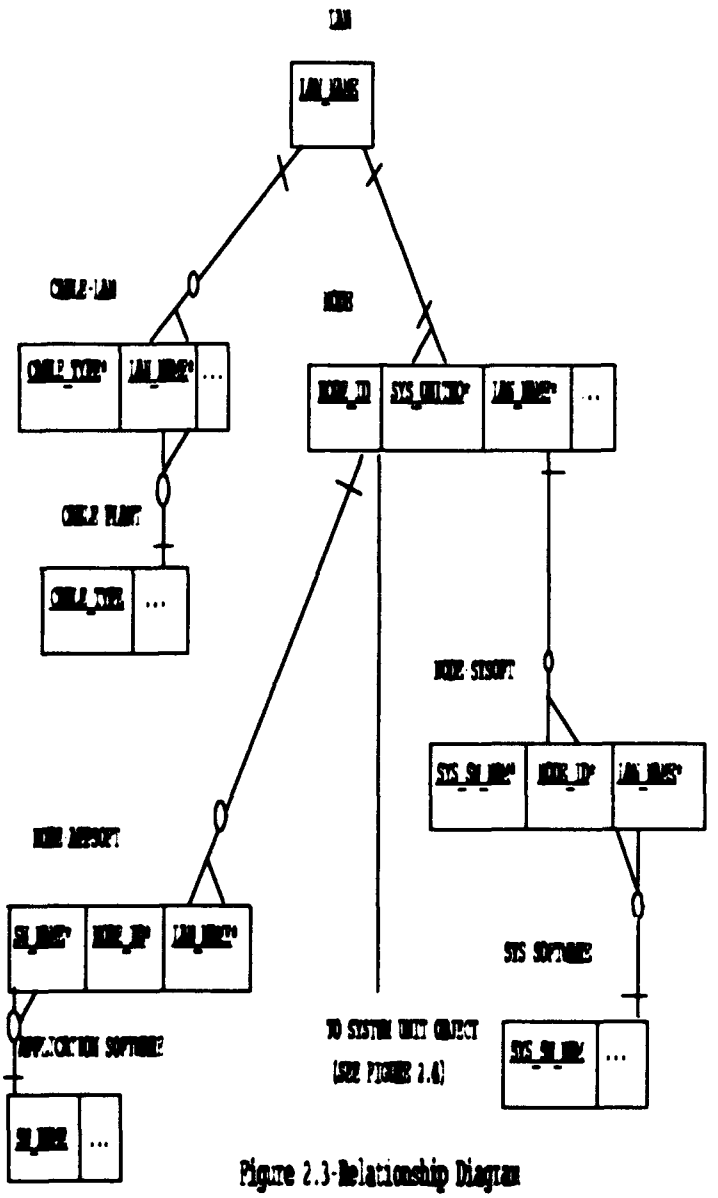


Figure 2.3-Relationship Diagram

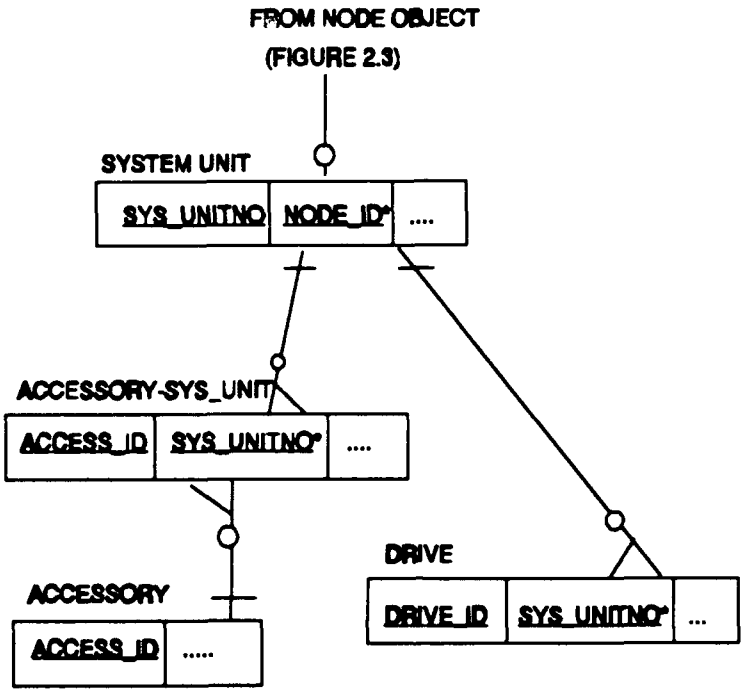


Figure 2.4 -Relationship Drawing (Cont)

1. The New Drive Object

The rationale for the first five changes are documented in the previous section. The last change, breaking the accessory object into two new objects, a new accessoryobject and a drive object, was done for several reasons: Drives and accessories used very few of the same fields in the old object, so it made sense from a housekeeping standpoint to separate them into two different objects. Secondly, most users do not consider drives to be an accessory, but an integral part of the system unit. Finally drives often require more maintenance and repairs than other accessories. Creating an object for drives allows for better records of drive maintenance and easier access to the records.

2. Changes to Data Fields

Some minor changes were also made to some of the fields in the new system unit object and the accessory object. These changes generally consisted of either dropping a field because the users deemed the field unnecessary or changing the field nomenclature to make the meaning of the field more clear and useful to the users. This was particularly true in the fields related to system unit memory where more descriptive fields and field names were desired by the users. Most of the field changes affected non-key fields. Two fields which were created which were key fields were the **sys_unitno** field for the new system unit object and the **Drive_ID** field for the new

drive object. These two fields deserve special mention since they are the primary key field for the system unit and drive objects, respectively.

3. System unit and Node Objects Revisited

Creation of a new system unit object and new node object significantly changed the relationships between the objects from what they had been with the old system. In light of this, further discussion of their relationship with each other and with the other objects is desirable.

In essence, the node object can be thought of as being virtual. It does not exist as a piece of equipment or anything physical but as a station on a network which has certain software attributes and certain hardware attributes. All of the node's hardware attributes are related to it through the system unit object which is assigned to it. The system unit on the other hand, has all of the physical attributes such as accessories, and drives and none of the network attributes.

Having the software related to the node instead of the system unit may at first seem to cause problems. It must be remembered that the system is designed to track local area network configurations. The software, except for the spare software is of no importance when it is associated with equipment which is not part of a local area network. If the system was also required to track stand alone microcomputers,

this would be a serious design deficiency. Instead, by relating the software to the node, software stays with the LAN even if the system unit is changed out. This turns out to be a more logical representation of LAN operations.

III. IMPLEMENTATION

A. METHODOLOGY

1. Implementation Goals

The implementation stressed the following themes in the system structure, design and user-system interface.

a. Data Integrity

It is critical in any database application that the data, especially the key fields, be protected as much as possible from user input errors. A small typographical error while entering a key field could make it impossible to recall the data record from the database file without system operator intervention. It is therefore very important in the implementation for the user to have as little direct input of key fields as possible.

This was accomplished by using database files which contained correct key field choices. The correct key field choices are displayed using a pop up menu and the user simply chooses the correct key field from the list. No direct data entry is required by the user. Key data field choices may be changed, added or deleted by a system operator with a rudimentary familiarity of the database tool, in this case dBASE IV 1.1. This greatly reduces the chance of losing data records. Only two key data fields in the design can presently

be entered directly by keyboard: the system unit number and the node identification code. Only the system unit must be searched for by entering the field directly. A node, after it is entered into the database, may be located by its local area network using pop up menus.

b. User-System Interface

A database is useless unless people actively use the system. While this may seem intuitively obvious, even the most powerful database application may sit in a proverbial 'corner' gathering dust if the user-system interface is so cumbersome and difficult to understand that it intimidates a user.

In this light the database system implementation was designed to be as straightforward and simple as possible. A menu system is used which clearly details the various functions and capabilities of the system. Most of the queries also involve using visually appealing and easy to use pop up menus. All input errors invoke concise and useful error messages to be generated and system use message prompts are used liberally throughout the application.

From an ergonomic point of view, the Dbase default screen colors of bright white and blue were found to be satisfactory. The two colors allowed good contrast while not being overly wearing on the eye.

c. System Flexibility

Given the rapid change in technology which the computer industry is undergoing, it is likely that these changes will also be felt in LAN hardware and software. A fine example of this is the CD ROM which is now available on the market. Any database system which attempts to document this hardware and software must have the inherent flexibility to add new equipment and software as it comes available without requiring wholesale changes to the database application. The system implementation allows for the adding and deleting of new types of drives, accessories, cable plant equipment and software as the technology changes through the Dbase IV control center. The key field identifiers for the accessories and software are stored in database files in the application. An individual moderately familiar with Dbase can easily make these changes. Barring major architectural design changes, this built in flexibility will allow the database application to remain useful with very few changes almost indefinitely.

d. System Expandability

The key to any successful database is the well thought out and systematic design of its data objects and relations. If a database is properly designed, the applications the design is capable of providing are almost infinite.

This implementation focused on data input, edit, query and delete functions. While these are necessary for the proper functioning of the database, the true power of any database lies with add hoc queries, reports and the data management functions the system is capable of performing.

It is rare that all of the requirements for a project are known at the start of a system design. A good design however allows for the addition of functions which conform with the previously defined data fields, data objects and relations.

This application (and indeed the dBASE IV tool) is designed to allow easy addition of new functions as new requirements are identified by the users (as they always are in an evolutionary design process). Adding functionality to this database is made relatively easy by the built in design modularity of the dBASE IV 1.1 tool. Only very minor changes to the main application pop up windows are required to add new functions. Even new objects may be added without changing the basic database structure.

It is envisioned that further requirements and system functionality can be easily added, possibly as a follow on thesis topic by a person who has access to the system design and is a competent dBASE IV 1.1 programmer. In this way the system can grow as user requirements grow and change.

e. System Maintenance

Although the database application went through extensive testing of its functions, it is realized that not all bugs will be found. Fortunately, the dBASE IV 1.1 modular programming approach enforces structured programming techniques. Proper programming nomenclature and format and extensive notes and comments are used throughout the code. Additionally, to assist in identifying programs, a list of programs by file name and their respective functions is included in the Appendix of this document.

2. Data and Object Access

Together the **system unit** and the **node** are the focus of this database application. All of the other objects fan off as relations of these two objects. This is true not only of the system design, but of the way the users approach the LAN. Once the system manager determines which LAN is desired, the next logical thought process is which node needs to be located. Since drives and accessories are related to the system unit on a given node and software is related to the node, all or any part of this information can be located by choosing the correct LAN and then the specific node in the LAN.

This thought process is implemented by using a pop up menu which contains all the possible LAN choices. Once a LAN is chosen, the nodes associated with that particular LAN are

displayed in another pop up menu. Depending on which application is chosen by the user, accessories, resident software, drives or the node or system unit may be edited, deleted or added. This procedure is graphically illustrated in figure 3.1 where a monitor associated with node ENET2 of the 3COM Ethernet LAN is located using pop ups. Accessories, drives and system units may also be located directly by entering their associated system unit number.

Two exceptions to this approach are the spare parts and the database files which contain the key field choices. Access to spare system units, accessories and drives is provided directly by choosing the desired object by its identification number off of a pop up menu as graphically depicted in Figure 3.2. This is necessary since the spares have no association with a LAN or a node. Any changes to the key field choices is accomplished by entering the dBASE control center and entering the data file directly. These procedures are further delineated in the Users Manual located in appendix A of this document.

3. Security Issues

Limiting access to the LAN Configuration and Maintenance System is desirable. The system program and database files could be seriously damaged or altered by a well meaning but inexperienced operator or a computer vandal.

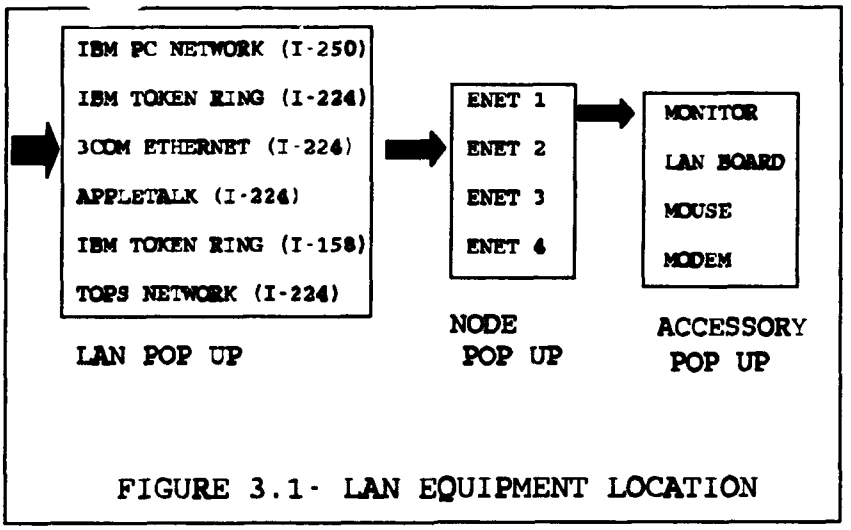


FIGURE 3.1- LAN EQUIPMENT LOCATION

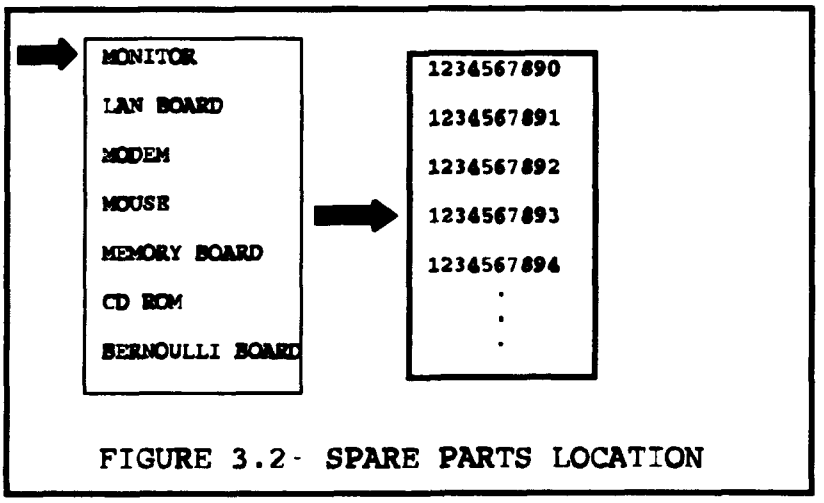


FIGURE 3.2- SPARE PARTS LOCATION

Most LAN operating systems allow the system operator, and sometimes even users to classify files on the server and limit file access. This capability would be sufficient to safeguard the system files. Access to the actual program can be controlled in several ways. dBASE IV 1.1 has an available password function which may be invoked. Another option is to build a password function into the batch file which calls the application.

4. Main Menu Organization

The main menus for the database application are the **LAN** menu, the **spares** menu, the **reports** menu and the **quit** menu. Figure 4 depicts the application main menu screen with all of the pop up menus displayed (normally only the pop up which is chosen on the bar menu is visible). All of the menus are arranged in an object-action style.

The **LAN** menu contains all of the submenus and functions for adding, deleting, editing and querying all objects associated with a given LAN. The **spares** menu executes the same functions for the spare parts. The **reports** menu generates reports on LANs, their equipment and spare parts. The **quit** menu allows the user to exit the application either to the control center or the network operating system. The **quit** menu also allows for database backup, database record packing and SQL queries. The **reports** section was separated from the LAN and spares menu to allow expanding the system's

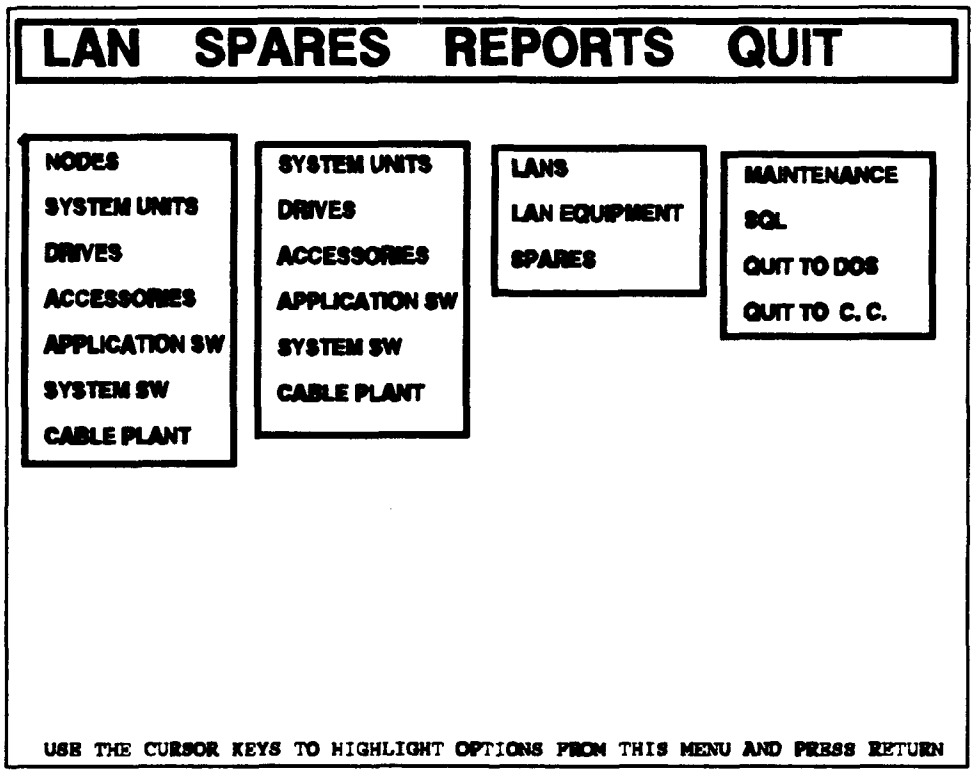


FIGURE 4.- MAIN MENU WITH ALL PULL DOWN MENUS SHOWN

report and query generation capability while requiring changes to as few menus in the main application as possible.

B. THE DATABASE TOOL

As previously mentioned, the database tool chosen to implement the application was dBASE IV 1.1. It is a very mature tool which the designer was familiar with and which was also available on the Administrative Science Department's 3COM LAN.

The dBASE IV 1.1 tool has the power and flexibility necessary to easily implement the database application. All of the above stated goals were achieved through use of the database tool. The pop up menu approach especially fit in well with achieving the desired goals.

One feature of dBASE IV 1.1 which may turn out to be extremely useful in the future is its compatibility with several expert systems such as VP Expert. Using an expert system as a front end to the database, the queries which the system could perform would be greatly expanded.

C. THE SYSTEM PROTOTYPE

The system prototype was designed on a stand alone microcomputer and then tested on the Administrative Science Department 3COM Local Area Network. The prototype was not a 'throw away' and was designed to be ultimately implemented. System tests on the 3COM network proved satisfactory.

IV. CONCLUSIONS AND LESSONS LEARNED

A. CONCLUSIONS

1. Satisfaction of Requirements

The design and implementation of the LAN Configuration Maintenance and Management System satisfied user requirements. Additionally, the system is designed in such a manner that many follow on requirements and upgrades can be implemented. It is expected that the system will provide a useful and versatile tool to LAN system managers.

2. Database Tool Effectiveness

The dBASE IV 1.1 database tool readily met the challenge of implementing the system design. It has many built in functions which assist not only with fulfilling the technical specifications and requirements of the system, but also in fulfilling the more subtle implementation goals of user friendliness, expandability, maintainability and database integrity.

Another advantage of the dBASE IV 1.1 tool as mentioned previously is its compatibility with many expert systems. An expert system can be used as a front end to a database combining great user friendliness with substantial query capability.

While the dBASE IV 1.1 tool is very powerful, it is not particularly easy to learn. It requires a system designer knowledgeable not only in database design techniques, but also in structured programming and the dBASE IV programming language. The coding and testing of applications was often a long and arduous process.

B. LESSONS LEARNED

1. Coding Requirements

The time required to write code for applications was continuously underestimated. Although a large amount of time was allotted to the project for code writing and testing, at least twice as much time was spent in this phase as expected. This was due partially to a fairly steep learning curve by the programmer in learning the functions and versatility of the Dbase language. In addition, there seems to be a natural tendency in any software development program, even on the larger scale developments, to be overly optimistic regarding time constraints when code writing and testing are involved.

2. Requirements Changes

In any system development process, justifiable requirements and design changes will happen throughout the course of development. Even the best thought out requirements will probably require at least minor changes. This is only natural since users often only have a vague idea at first of

how the system is capable of helping them. Since requirements changes are simply the nature of the beast in all but the most well defined and mature systems, the system designer must be able to prepare and react to these changes.

The system must be designed to allow for as much change as possible. While this is very application and database tool dependent, most of the new tools use structured programming techniques which makes changes and program maintenance much less painful. Even more important, the objects and relationships must be designed to allow for as much change as possible.

3. User-Designer Communication

For the system to evolve properly, the system designer must keep in contact with the users throughout all phases of the system development. The earlier any requirements changes are learned, the less work is required to implement the necessary changes.

This approach is especially important when different personnel are executing the various phases of the development process. System documentation does not always adequately express user concerns or fully answer design questions. Even with well documented systems, it is advisable to interview the users as soon as possible. Contact with the other personnel involved in the development process in also highly recommended.

This particular implementation required changes, some major and some minor, to system design and requirements throughout the whole development process. All of the changes were desirable and beneficial. Without constant communication with the users, it is doubtful many of these changes would have come to light until much later in the development process causing serious development and implementation delays.

V. RECOMMENDATIONS FOR FURTHER SYSTEM ENHANCEMENTS

The development lifecycle of a software system does not end until the system is replaced. Enhancements, new applications and system maintenance are an ongoing fact of life. There are several excellent applications any one of which would make a very useful addition to the system and an interesting thesis topic.

A. ADDITION OF A MAINTENANCE REPORT OBJECT

The addition of a **maintenance report** object along with associated applications would allow tracking of all maintenance on the LANs and their associated equipment in the database. This enhancement would provide a very powerful tool for system management, allowing statistical and trend analysis to be performed on the LAN equipment through queries.

The new report object would not require any changes in system objects or relations presently in the database. With the exception of a report identification number, only key fields already in use would be used. Only minor changes to the main application menu system would be required.

B. SQL COMMAND LIBRARY

The dBASE IV 1.1 tool is able to use the standard SQL language to make queries. A library of SQL queries for the database can be created to enhance the capability of the system. This would require no change in the database structure since the system is capable of performing SQL queries.

C. IMPLEMENTATION OF A FRONT END EXPERT SYSTEM

As previously mentioned, the ability of Dbase to interface with many decision support systems and expert systems makes these tools ideal as a front end to the database. A properly designed expert system front end would allow queries to be performed and programmed with much greater ease and flexibility. These powerful and user friendly tools have to potential of adding a whole new dimension to the present system.

APPENDIX A

USERS MANUAL

A. GENERAL OPERATION

1. Moving around in the Application Main Menu

The **application main menu** consists of a bar menu at the top of the screen and a pull down menu which gives the options for the highlighted portion of the bar menu. The bar menu choices are as follows:

- LAN
- SPARES
- REPORTS
- QUIT

The highlighted bar menu choice will always display its associated pull down menu. The bar menu choice is changed by using the horizontal arrow keys on the keyboard. The choice highlighted on the pull down menu can be changed by using the up and down arrow keys on the keyboard. A choice on a pull down menu is accepted by highlighting the desired choice and pressing enter.

2. Using Pop Up Menus

Choosing an option off of one of the main screen pull down menus will either invoke a system application or a

submenu. Pop up submenus invoked from a pull down menu behave the same as for the pull down menus. The desired application is chosen by highlighting it using the horizontal arrow keys and pressing the carriage return key. Pop up submenus may be backed out of by pressing escape. This will return the user to the previous menu.

Pressing escape while in one of the main pull down menus is, however to be avoided. This may lead to the main program being exited without saving database record changes. **The main menu should always be exited by using the exit options available on the quit pull down menu.**

3. Data Entry on Formatted Data Record Screens

Most of the Objects in the LAN Maintenance and Configuration Database System have formatted record entry templates which present a user friendly screen on which to enter the necessary data fields for the record. When such a screen is invoked for editing an existing record or adding a record, the cursor will be in the first field of the record.

The cursor may be moved forward to the next desired field in two ways. The tab key may be used to advance to the next field or the user can space bar forward. When using the latter method, the cursor will skip to the next data field once it reaches the last space in the present data field.

The cursor may be moved backward to a previous data field by using the shift-tab key combination. Backspacing will also accomplish this.

Again, **using the escape key is to be avoided when in a record template**. This could lead to records being lost or only partially saved. The proper way to exit a record template is to carriage return through all of the data fields (hopefully filling them with data along the way).

B. THE LAN MENU

The LAN menu is for adding, deleting, queries or modifications to any of the nodes, software or equipment assigned to the LANs in the database. There are presently two methods of locating LAN equipment in the database and one method of locating nodes and software.

The first and easiest method for locating equipment records in the database is by locating the equipment's LAN and node through the use of pop up menus. The second method is to enter the system unit which the equipment is associated with. The first method is decidedly easier since the user only needs to know the LAN and node which the equipment is on and this information is displayed using pop up screens.

The following is a summary of the system applications which are available on the LAN pull down menu:

1. Nodes Submenu

The nodes submenu is invoked by choosing **nodes** on the **LAN** pull down menu. This system applications of this submenu perform the following functions:

- **Create a new Node:** This choice gives the user a second pop up menu which displays two system applications: 1) The choice of creating a new node and then adding a new system unit which is not in the database. 2) Creating a new node and assigning to it a system unit which is already in the database system.
- **Change a node System Unit:** This allows the system unit assigned to a node in the database to be changed to another system unit. The new system unit must already exist in the database.
- **Edit a Node:** Allows a node to be edited.
- **Delete a Node:** Deletes a node and its associated software from the database.
- **Query a Node:** Displays the Node information and its associated software and equipment.

2. The System Unit Submenu

The **System Unit** submenu is invoked by choosing **system unit** from the **LAN** pull down menu. The system applications of this submenu perform the following functions:

- **Change the Node assignment of a System Unit:** Changes a System Unit's node assignment to a different node. The new node must be in the system database.
- **Edit a System Unit:** Allows the user to edit the system unit and its associated drives and accessories.
- **Query a System Unit:** Displays the system unit pertinent information and its associated LAN, software, drives and accessories.
- **Remove System Unit from Node:** Disassociates a system unit from a node.

3. The Drives Submenu

The **drive** submenu is invoked by choosing **drive** from the **LAN** pull down menu and pressing return. The system applications of this submenu perform the following functions:

- **Add a Drive:** Adds a drive to an existing system unit in the database.
- **Query/Edit a Drive:** Displays a drive's information for editing or for a simple query.
- **Delete a Drive:** Deletes a drive from a system unit. This application deletes the drive record. **If it is desired to keep a record of the drive as a spare, it must be added to the spare database.**

4. The Accessories Submenu

The **accessory** submenu is invoked by choosing **accessory** from the **LAN** pull down menu and pressing return. The system applications of this submenu perform the following functions:

- **Add an Accessory:** Adds an accessory to an existing system unit in the database.
- **Query/Edit an Accessory:** Displays an accessory's information for editing or for a simple query.
- **Delete a Accessory:** Deletes a drive from a system unit. As with the accessory delete application, the record is deleted. If record of the accessory is to be saved as a spare, it must be re-entered into the database as a spare.

5. The Application Software Submenu

The **applications software** submenu is invoked by choosing **applications software** from the **LAN** pull down menu and pressing return. The systems applications of this submenu perform the following functions:

- **Add Applications Software:** Adds application software to a designated node.

- **Delete Application Software:** Deletes application software from a designated node.

6. The System Software Submenu

The **systems software** submenu is invoked by choosing **system software** from the **LAN** pull down menu and pressing return. The systems applications of this submenu perform the following functions:

- **Add System Software:** Adds system software to a designated node.

- **Delete System Software:** Deletes system software from a designated node.

- **Add System Software to a LAN:** Adds a type of system software to every node on a given LAN.

- **Delete System Software from a LAN:** Deletes a type of system software from every node on a given LAN.

7. The Cable Plant Submenu

The **cable plant** submenu is invoked by choosing **cable plant** from the **LAN** pull down menu and pressing return. The system applications of this submenu perform the following functions:

- **Add Cable Plant Equipment:** Allows cable plant equipment to be added to a designated LAN.

- **Remove Cable Plant Equipment:** Allows cable plant equipment to be removed from a designated LAN.

- **Edit/Query Cable Plant Equipment:** Allows cable plant equipment to be edited/viewed for a particular LAN.

C. THE SPARES MENU

The **spares** menu is for adding, editing, deleting and queries of equipment and software which is not associated with a LAN, or in other words, are spare parts. Unlike the LAN menu, the equipment spares are located by using their serial numbers. The method for additions, deletions and queries for the cable plant are very similar to the LAN menu. The spare software is tracked by it's number of copies. Adding, deleting and editing are all one operation.

The records for spare system units, drives and accessories are stored in the same databases which hold the these records for the LAN menu. The spare records are differentiated by having the word '**SPARE**' inserted in the node field (in the case of a system unit) or the system unit field (in the case of drives or accessories).

The system presently allows only the system unit records to be transferred from a LAN to a spare parts category or vice versa in one operation. The reason behind this has to do with a lack of detailed records on the serial numbers of the drives and accessories present in the various LANs. As stated previously, the serial numbers are required to locate these objects while they are spare parts since they are not associated with LANs. The LANs would require dismantling to access the necessary serial numbers, a time consuming and non-viable option.

The unfortunate backlash of this situation is that if drives and accessories are removed from their respective node on a LAN, their record must be deleted and re-inserted as a spare, a two step process. The converse process does not hold however. A spare drive or accessory can be added to a system unit by editing the system unit field to read the system unit number of the desired system unit. This is not recommended since it can create key field data errors. **For transferring drives and accessories from a node to spare parts or vice versa, it is recommended that a two step process be used of deleting the record and then re-entering it.**

The following is a summary of the system applications which are available on the **spares** menu:

1. The System Unit Submenu

The **system unit** submenu is invoked by choosing **system unit** from the **spares** pull down menu and pressing enter. The system applications of this submenu perform the following functions:

- **Add a System Unit Spare:** Adds a spare system unit to the database and allows the user to add drives and accessories to the system unit.
- **Delete a System Unit Spare:** Deletes a system unit spare and its associated drives and accessories from the database.
- **Edit a System Unit Spare:** Allows the user to edit a system unit and its associated drives and accessories.
- **Query a System Unit Spare:** Allows the user to query a system unit and its associated drives and accessories.

2. The Drives Submenu

The **drives** submenu is invoked by choosing **drives** from the **spares** pull down menu and pressing enter. The system applications of this submenu perform the following functions:

- **Add a Spare Drive:** Adds a spare drive to the database.
- **Delete a Spare Drive:** Deletes a spare drive from the database.
- **Query/Edit a Spare Drive:** Allows a user to query/edit a spare drive record.

3. The Accessories Submenu

The **accessories** submenu is invoked by choosing **accessories** from the **spares** pull down menu and pressing enter. The system applications of this submenu perform the following functions:

- **Add a Spare Accessory:** Adds a spare accessory to the database.
- **Delete a Spare Accessory:** Deletes a spare accessory from the database.
- **Query/Edit a Spare Accessory:** Allows the user to query/edit a spare accessory record.

4. The Application Software Function

Choosing **application software** invokes a function which allows a user to add to or delete from the number of spare copies of a certain type of application software.

5. The System Software Function

Choosing **system software** invokes a function which allows a user to add to or delete from the number of spare copies of a certain type of system software.

6. The Cable Plant Function

Choosing **cable plant** invokes a function which allows a user to add to or delete from the number of spare copies of a certain type of system software.

D. THE REPORTS MENU

This menu is responsible for the formatted printed and screen reports on the LANs and spare parts. Future reports may be added with minor changes to the menus. The following is a summary of the system reports which are available on the reports pull down menu:

1. The LAN Submenu

The **LAN** submenu is invoked by choosing **LAN** from the **report** pull down menu and pressing enter. This submenu creates the following reports:

- **LAN Report:** This report lists all of the equipment and software for a given LAN.
- **Node Report:** This report lists pertinent information on all of the nodes in a given LAN.
- **Server Report:** This report list the servers, their software and equipment for a given LAN.
- **Application Software Report:** Lists all of the application software available on a given LAN.
- **System Software Report:** Lists all of the system software on a given LAN.
- **Cable Plant Report:** Lists all of the cable plant equipment associated with a given LAN.

2. The LAN Equipment Submenu

The **LAN Equipment** submenu is invoked by choosing **LAN Equipment** from the **report** pull down menu and pressing enter. This submenu creates the following reports:

- **Node Report:** Lists node information and all of the software and equipment associated for a given node.
- **Drives Report:** Lists all of the drives in the database.
- **Accessories Report:** Lists all of the accessories in the database.
- **System Unit Report:** Lists all of the system units in the database.

3. The Spares Submenu

The **spares** submenu is invoked by choosing **spares** from the **report** pull down menu and pressing enter. This submenu creates the following reports:

- **Spare System Unit Report:** Lists all spare system units.
- **Spare Drives Report:** Lists all spare drives.
- **Spare Accessory Report:** Lists all spare accessories.
- **Spare System Software Report:** Lists all spare copies of all system software by type.
- **Spare Application Software Report:** Lists all spare copies of all application software by type.
- **Spare Cable Plant Report:** Lists all spare cable plant equipment by type.

E. THE QUIT MENU

This menu might be more aptly named the utilities menu. Besides allowing the user to quit either to the operating system or the dBASE control center, it allows the user to pack

database files, back up database files and perform SQL queries. The following is a summary of the system applications available on the quit pull down menu:

1. The Maintenance Submenu

The **maintenance** submenu is invoked by choosing **maintenance** from the **quit** pull down menu and pressing enter. The system applications of this submenu perform the following functions:

- **Pack Databases:** Packs the system database files. Should be performed at least monthly.
- **Backup Databases:** Backs up the system database files. Should be performed at least weekly.

2. The SQL Function

The **SQL** function is invoked by choosing **SQL** from the **quit** pull down menu and pressing enter. This function places the user in the dBASE SQL mode where SQL commands can be entered. This screen exits to the dBASE control center by pressing the F2 key.

3. The Quit to DOS Function

The **Quit to DOS** function is invoked by choosing **quit to DOS** from the **quit** pull down menu and pressing enter. This function returns the user to the operating system.

4. The Quit to Control Center Function

The **quit to control center** function is invoked by choosing **quit to control center** from the **quit** pull down menu

and pressing enter. This function places the user in the dBASE control center.

F. ADDING AND DELETING KEY FIELD VALUES

The lan, system software, application software, accessory and cable plant fields each have a database file which contains all the available choices for these fields. In order to add a new type of accessory to the database, add a new LAN or possibly delete a type of obsolete software from the database, the field must be added to or deleted from the pertinent database file. The following database files hold the pertinent field information:

- ACC_NAME: Accessory key field names.
- APP_SW: Application Software key field names. This database file is also used for storing the number of spare software packages.
- CABL_PL: Cable Plant equipment key field names. This database file is also used for storing the number of spare cable plant equipment.
- LAN: Lan key field names.
- SYS_SW: System Software key field names. This database file is also used for storing the number of spare system software packages.

Unlike most other operations of the database which are executed within the system application, these changes must be accomplished within the control center. Since adding or deleting these values will not occur very often, it was not

considered worthwhile to create a system application for this function. The following steps can be taken to add, edit or delete key field values:

1. Enter the dBASE control center through the **quit** pull down menu.

2. Go to the data file section of the control center by using the cursor keys, choose the data file which is to be edited and press enter.

3. From the pop up menu, choose **display data** and press enter.

4. The data field will now be displayed. They may be edited by scrolling up and down using cursor keys. A new field may be added by pressing the **F10** key and choosing the **add a new record** function. A field may be deleted by pressing **F10** and choosing the **mark records for deletion** function.

5. The data file can be exited by either pressing escape or **F10** and choosing the exit bar menu choice.

6. Further information on use of the dBASE control center is available through the tool's manuals.

G. RETURNING TO THE APPLICATION FROM THE CONTROL CENTER

This function may be accomplished through the following procedure:

1. Use the cursor keys to enter the application portion of the control center.

2. Choose the **Lanmaint** application and press enter.

3. Choose the **Run Program** option from and pop up menu and enter **yes** when prompted.

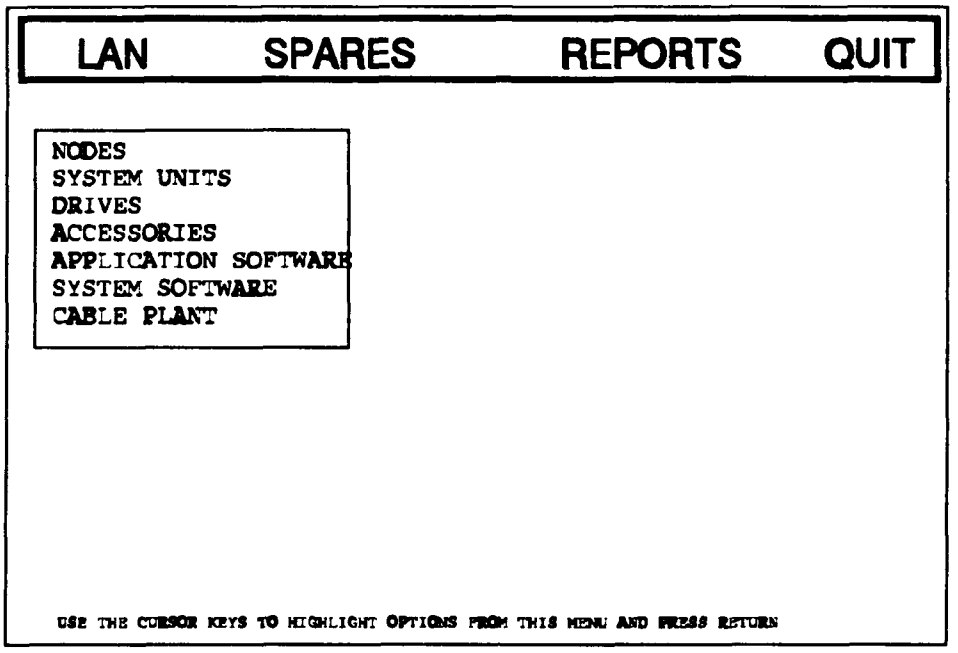
4. This action will return the user to the LAN Configuration and Maintenance database system.

H. SUMMARY

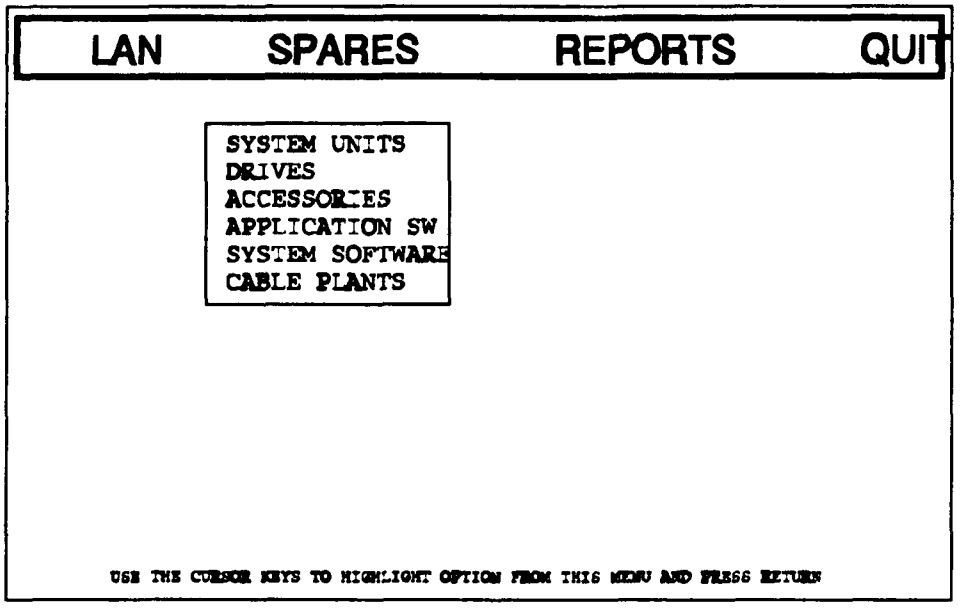
The LAN Configuration and Maintenance Database System is a powerful and flexible tool for managing the maintenance and configuration of almost any LAN configuration. This users manual is designed to introduce the system to a new user and explain its functions.

While a knowledge of the dBASE tool is not necessary to use this application, it will certainly aid in understanding the system and its capabilities. More knowledge of the dBASE IV language can be gotten from the tool's manuals. It is assumed the users of this system will be LAN system managers and will have a nominal knowledge of databases and their terminology.

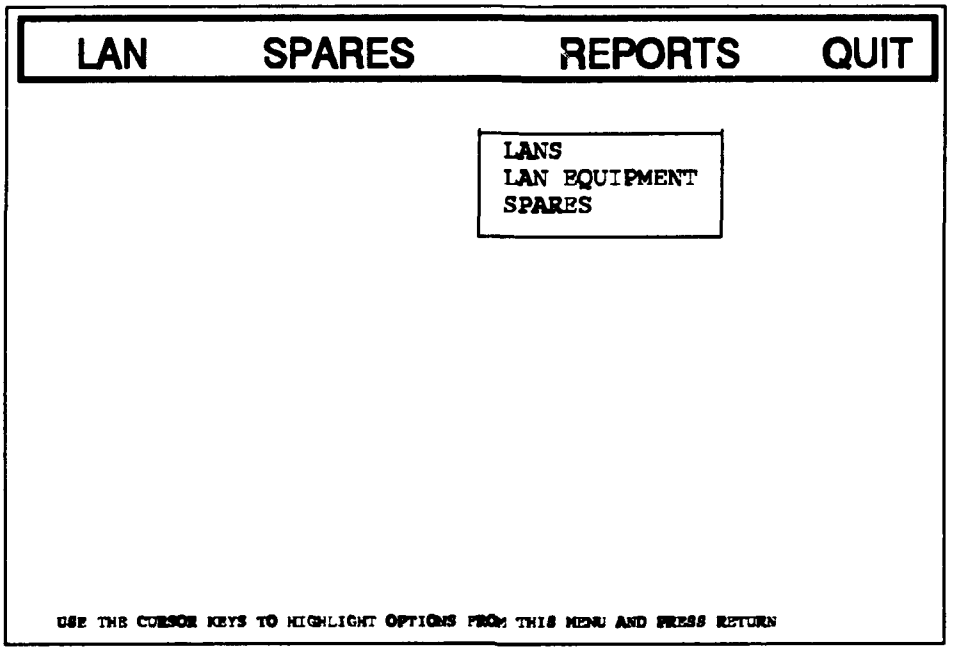
APPENDIX B-SYSTEM MENUS



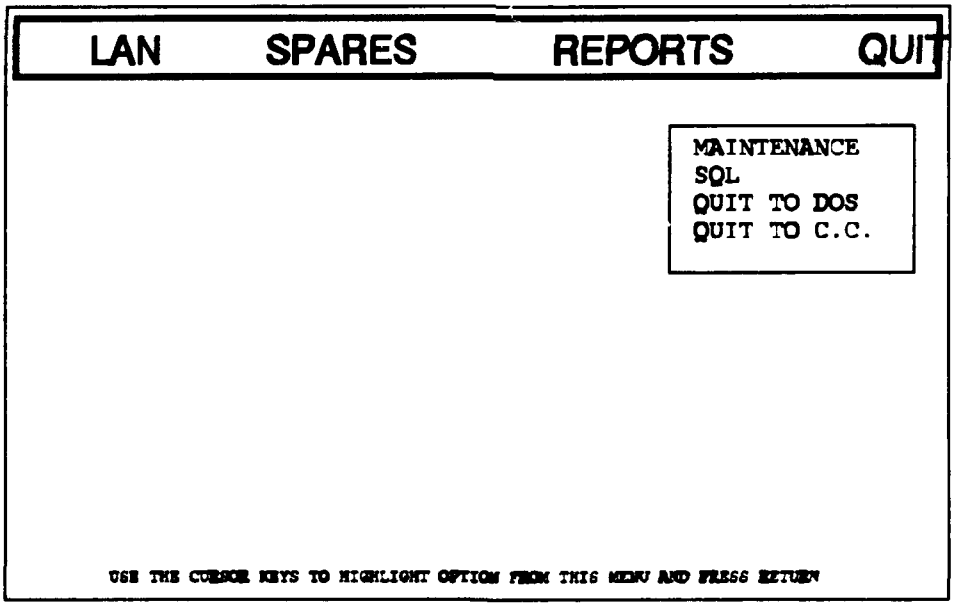
LAN PULL DOWN MENU



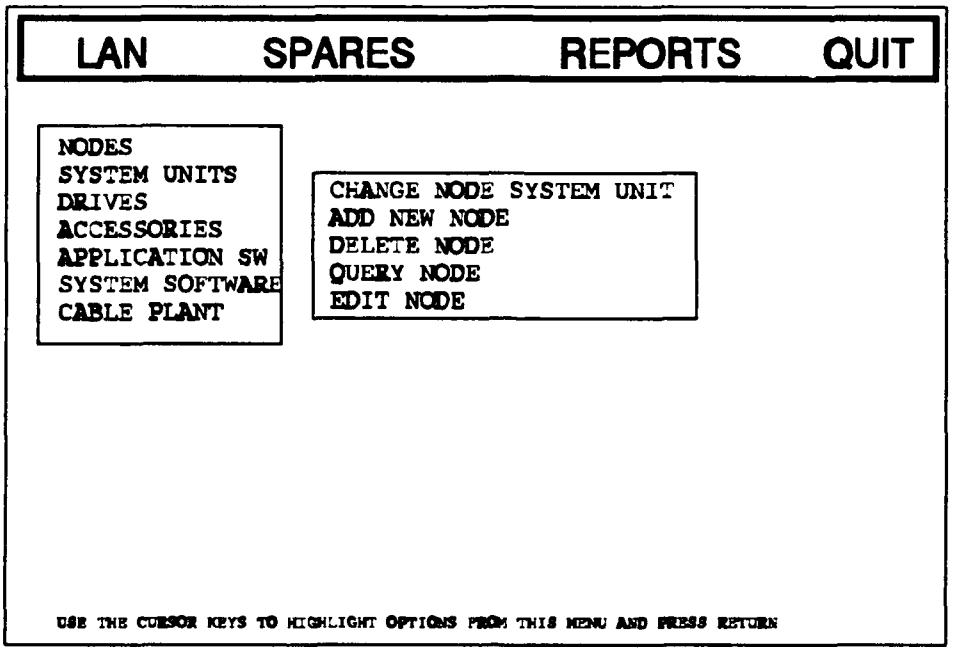
SPARES PULL DOWN MENU



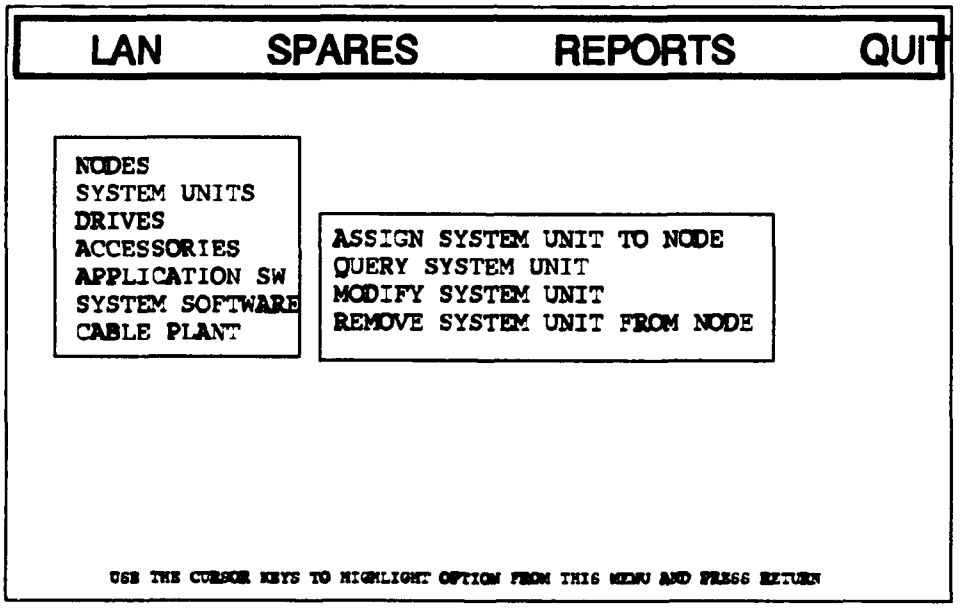
REPORTS PULL DOWN MENU



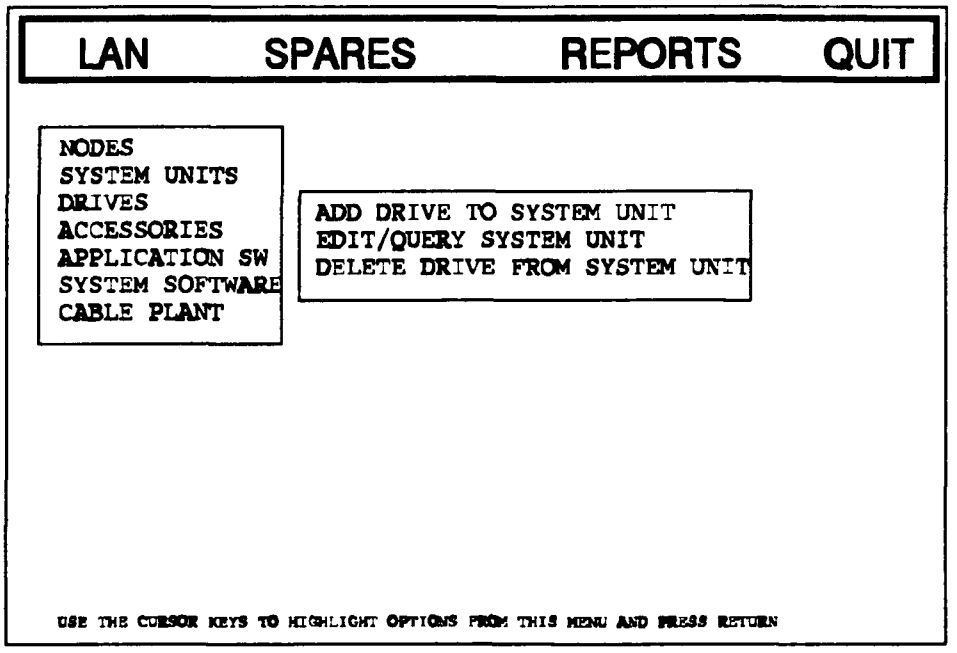
QUIT PULL DOWN MENU



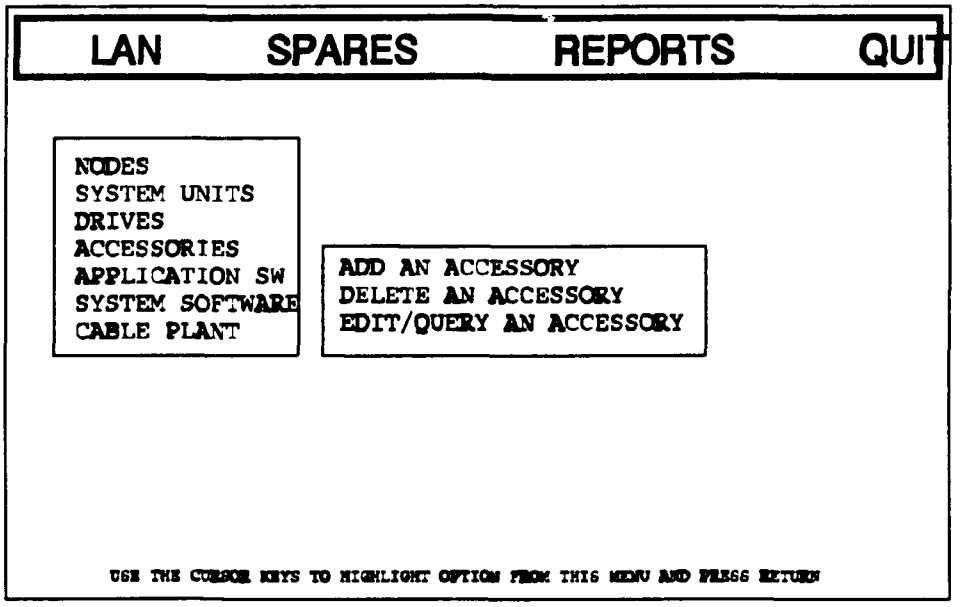
NODE POP UP SUBMENU



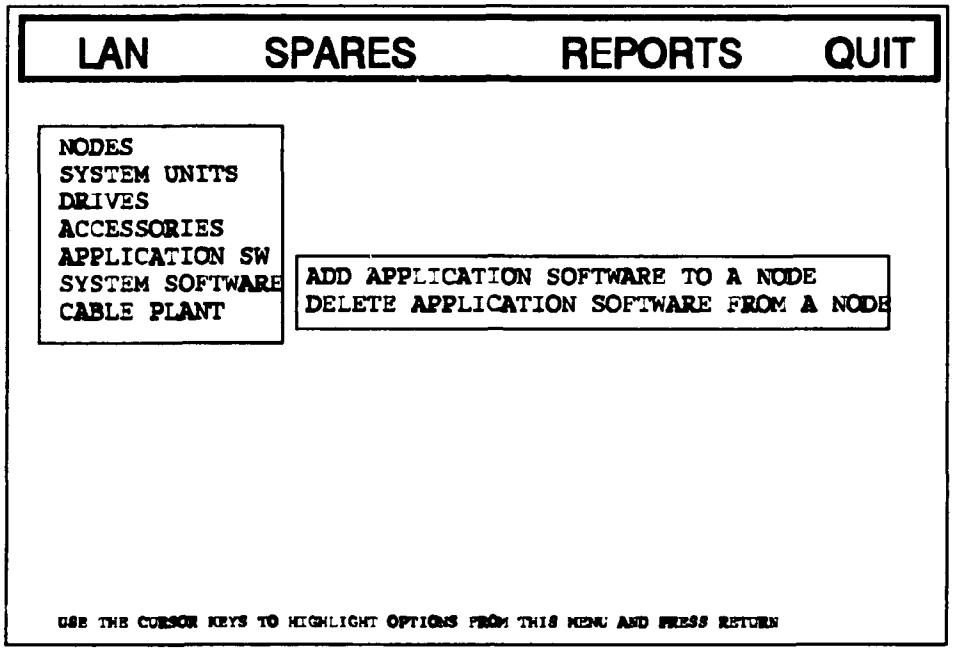
SYSTEM UNIT POP UP SUBMENU



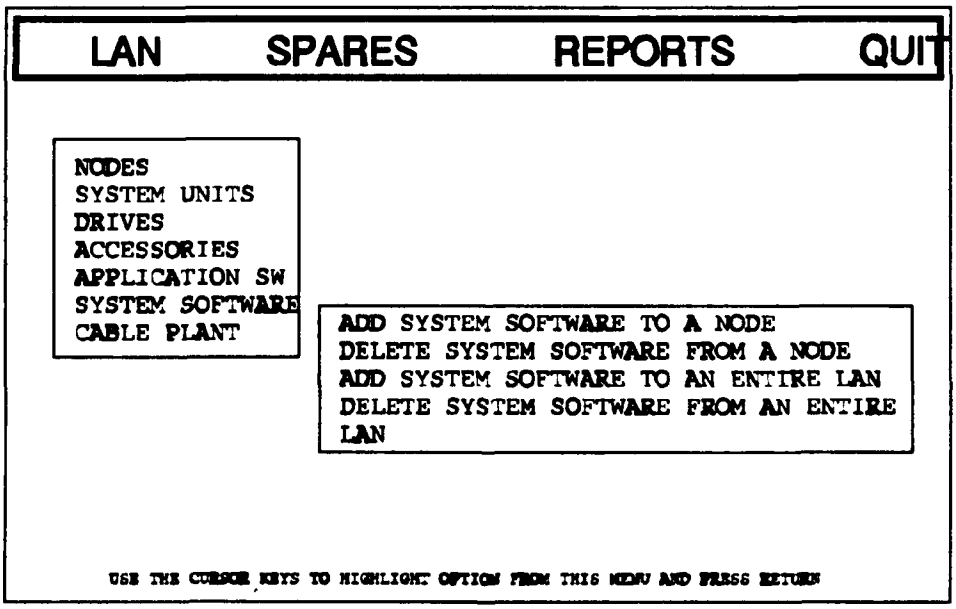
DRIVE POP UP SUBMENU



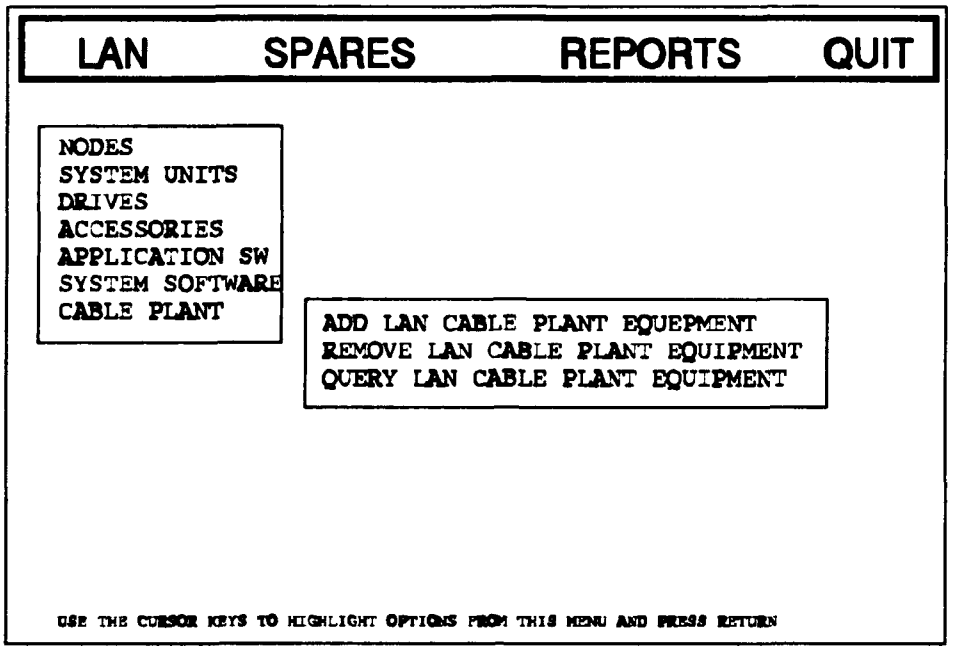
ACCESSORIES POP UP SUBMENU



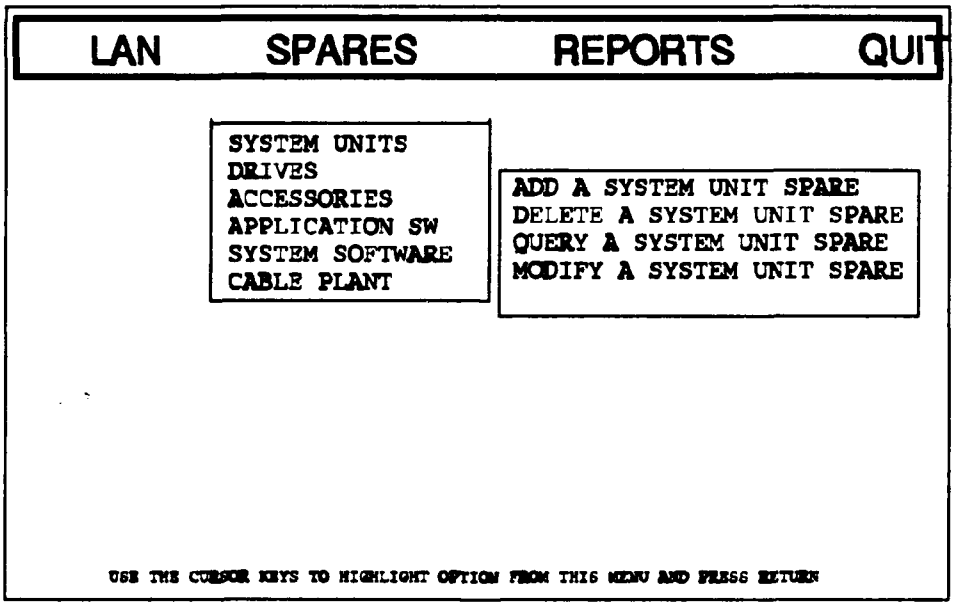
APPLICATION SOFTWARE POP UP SUBMENU



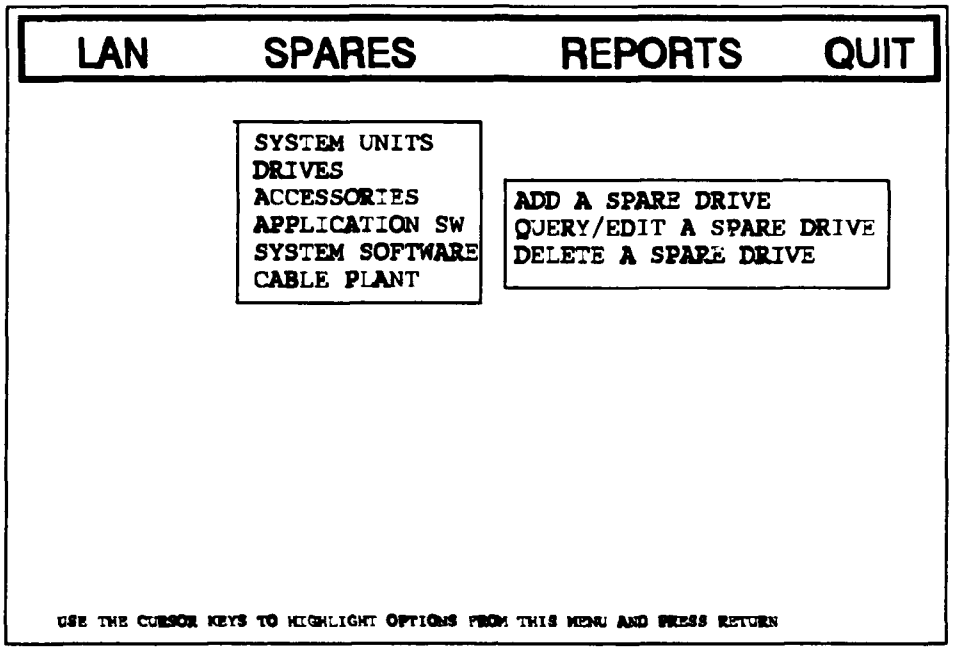
SYSTEM SOFTWARE POP UP SUBMENU



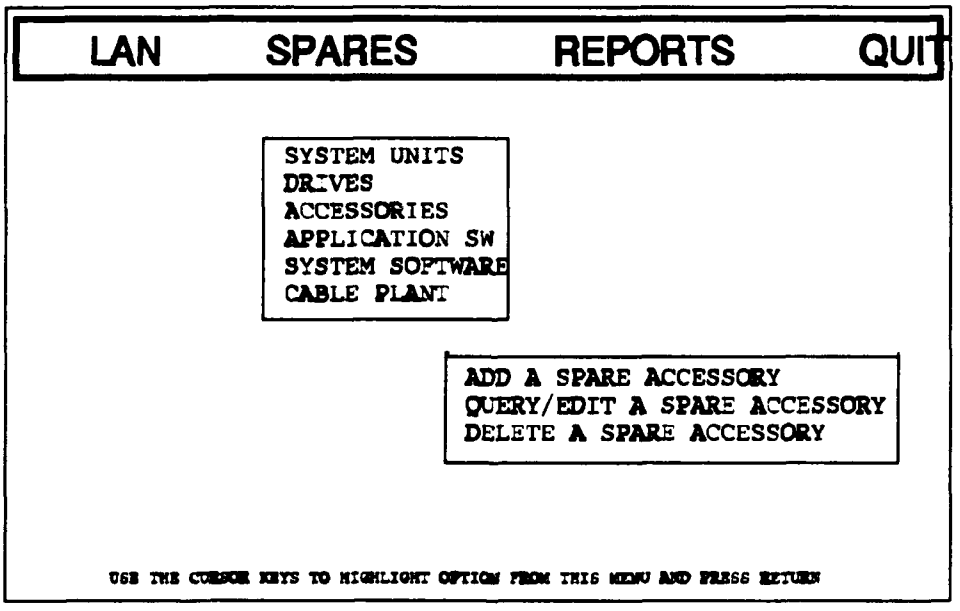
CABLE PLANT POP UP SUBMENU



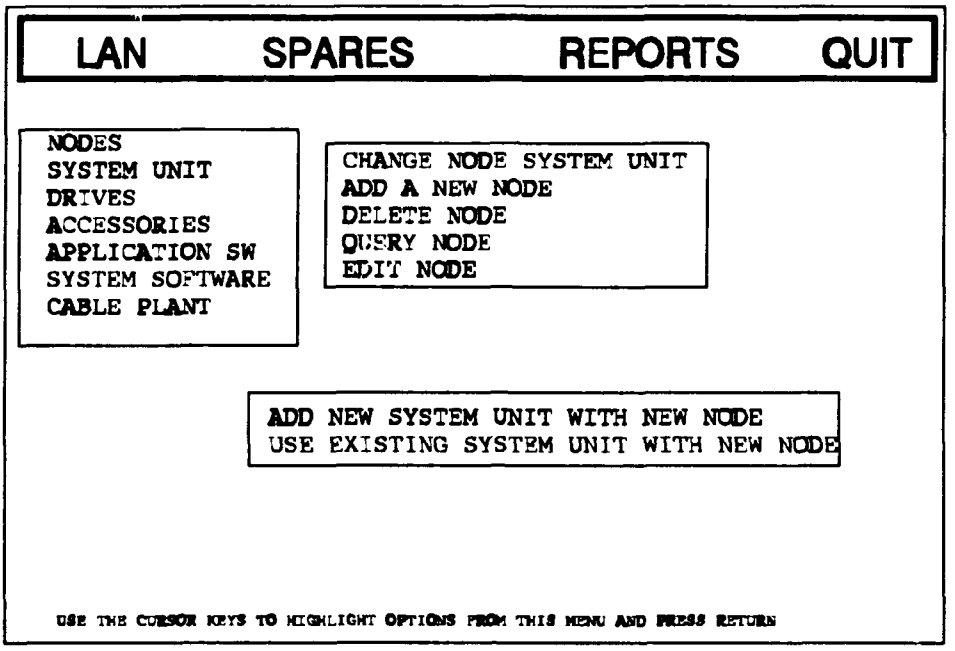
SYSTEM UNIT SPARE POP UP SUBMENU



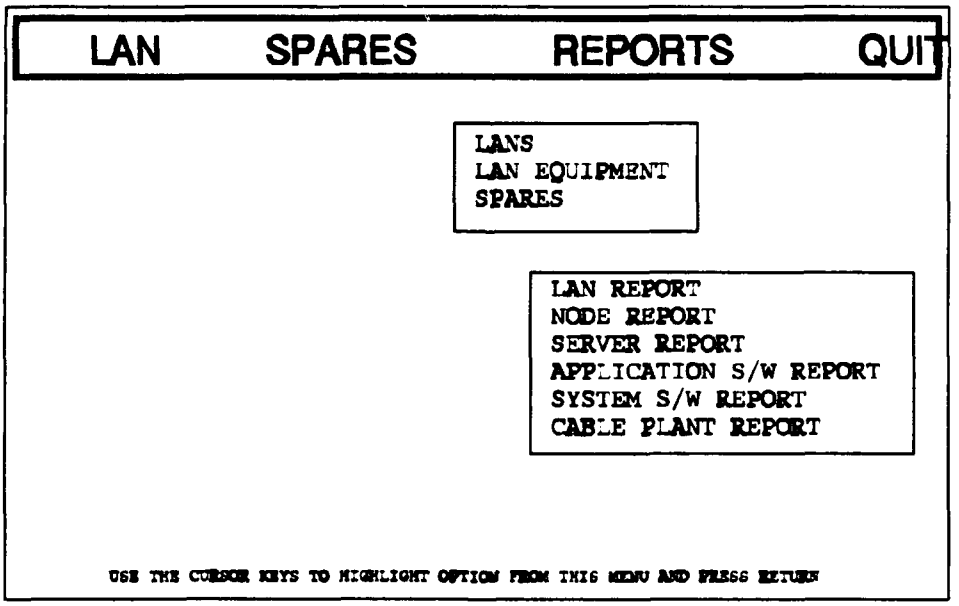
SPARE DRIVE POP UP SUBMENU



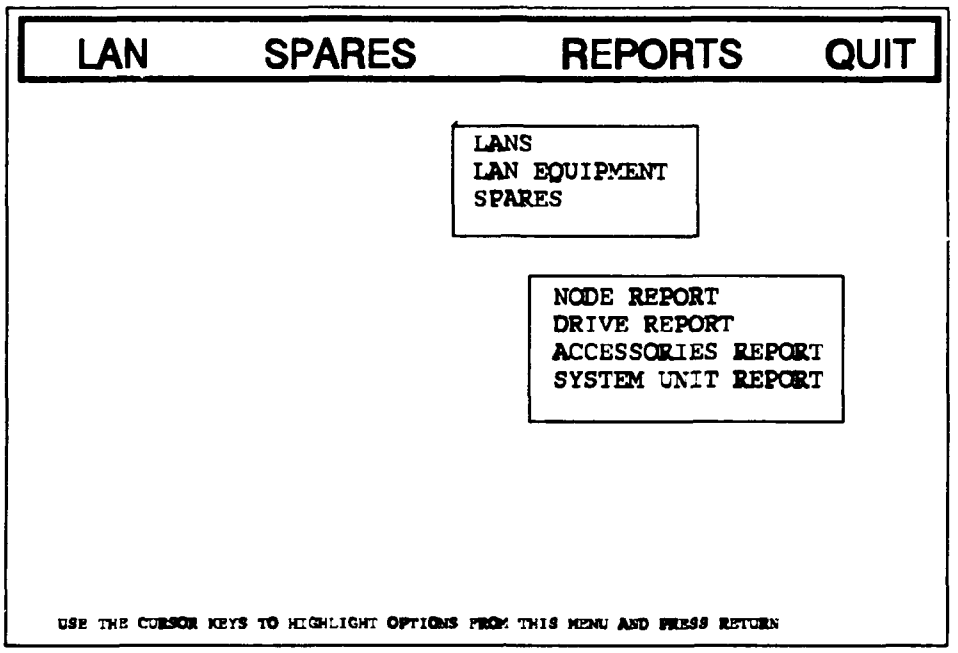
SPARE ACCESSORY POP UP SUBMENU



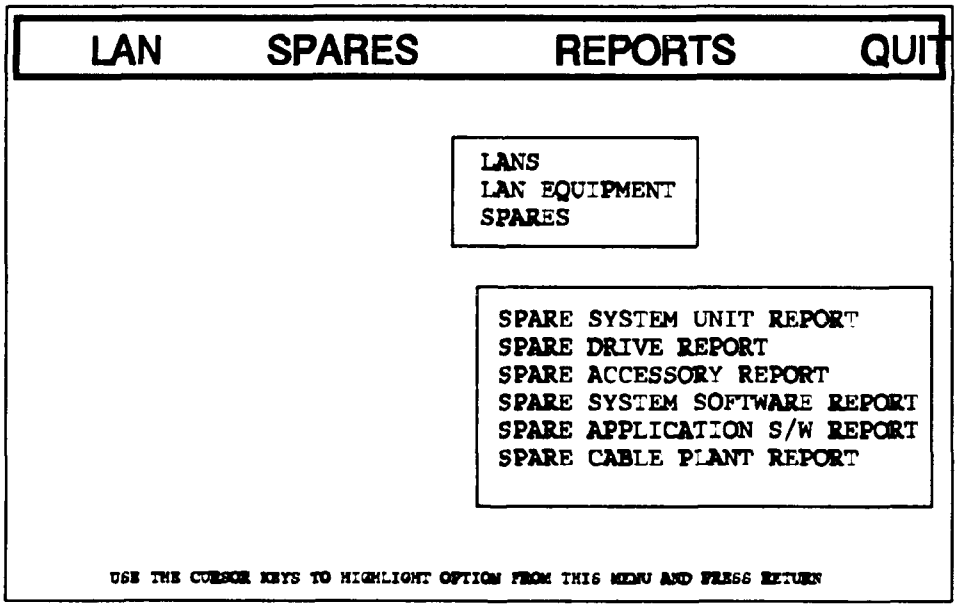
ADD A NODE POP UP SUB-SUBMENU



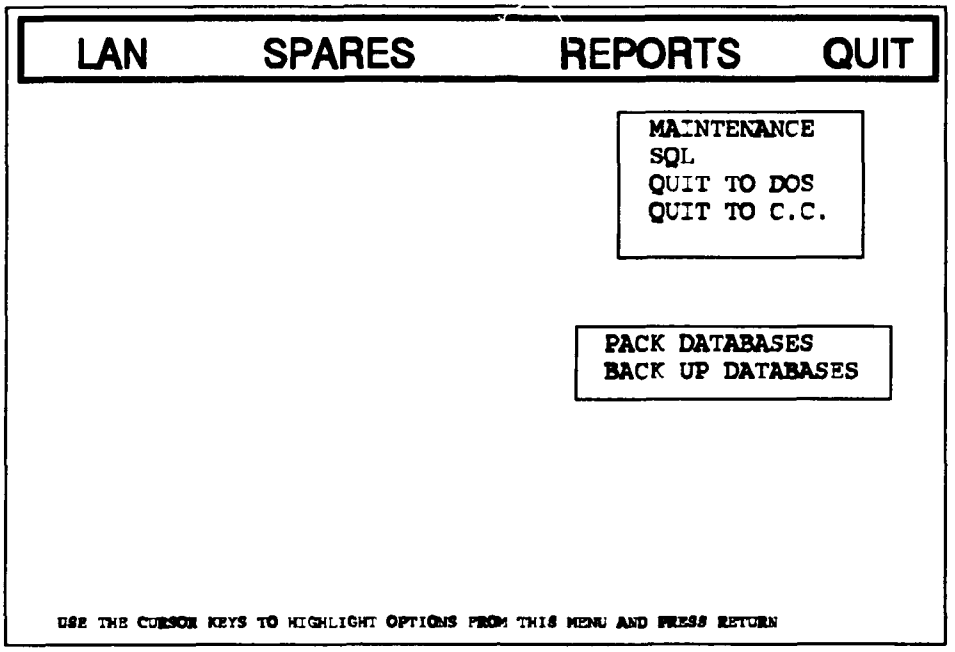
LANS REPORTS POP UP SUBMENU



LAN EQUIPMENT REPORTS POP UP SUBMENU

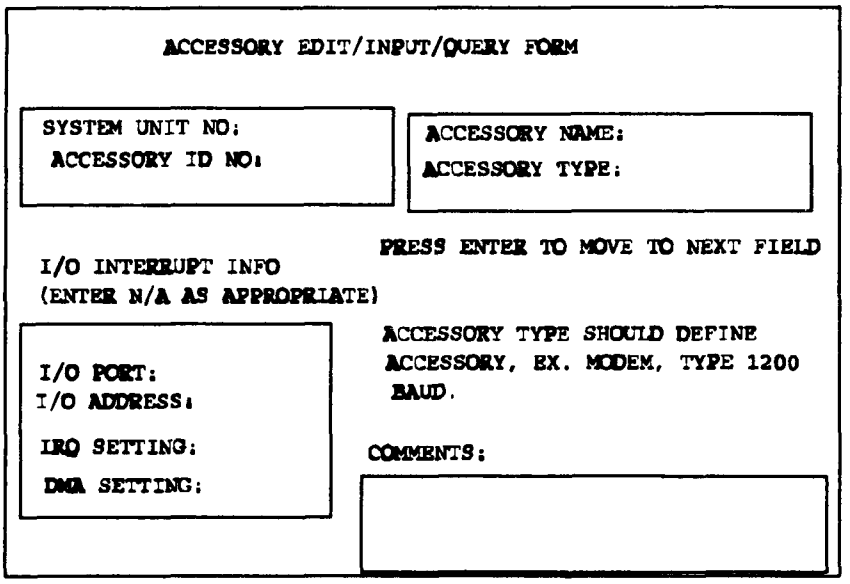


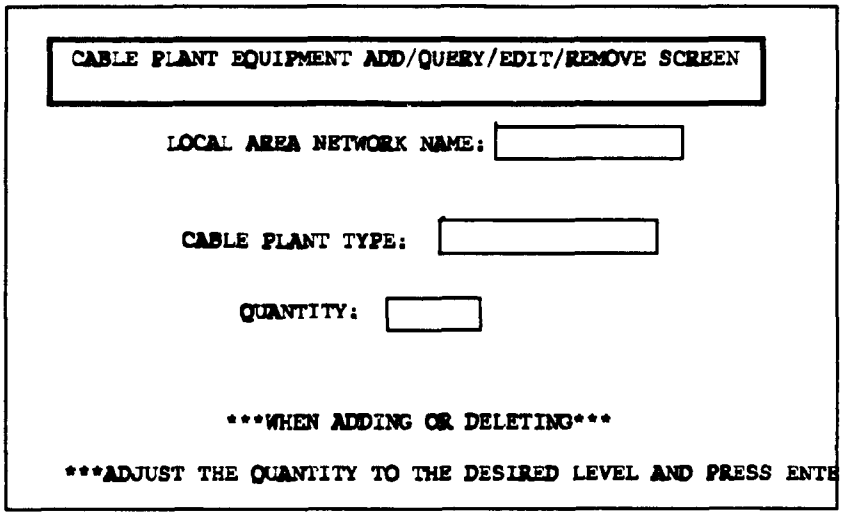
SPARES REPORTS POP UP SUBMENU

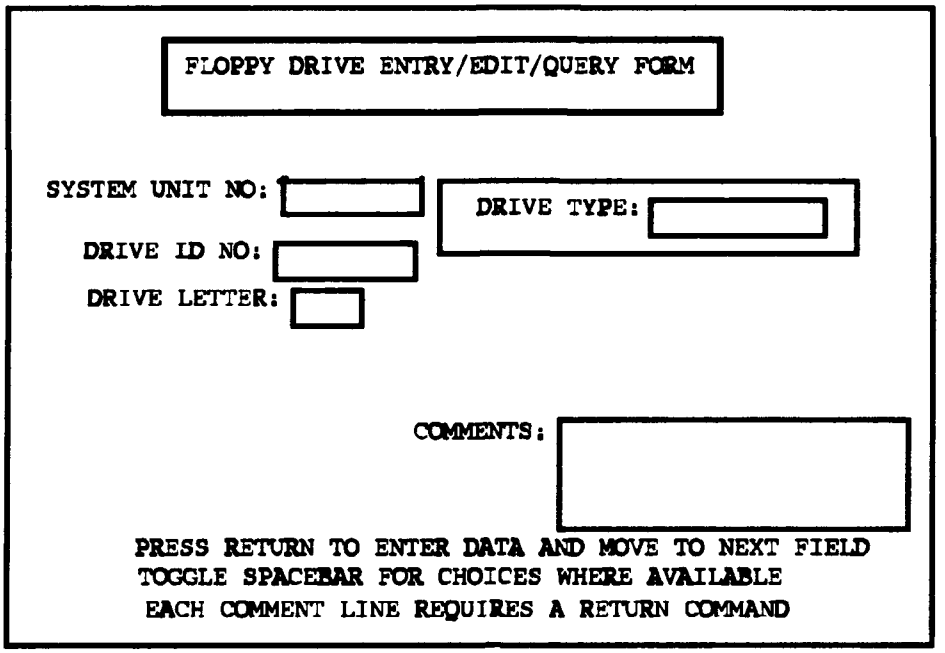


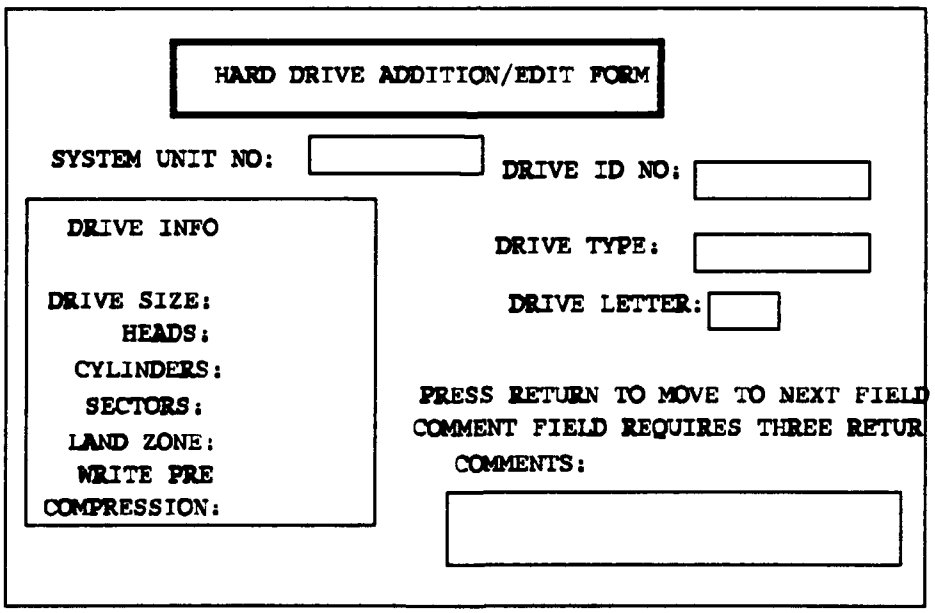
MAINTENANCE OPTIONS POP UP SUBMENU

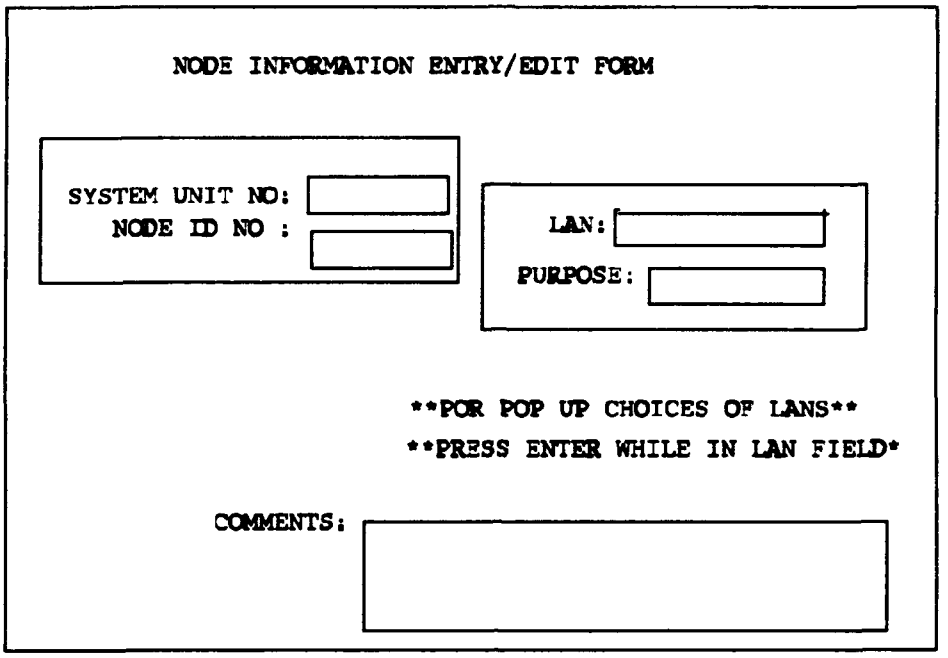
APPENDIX C-DATA ENTRY SCREENS

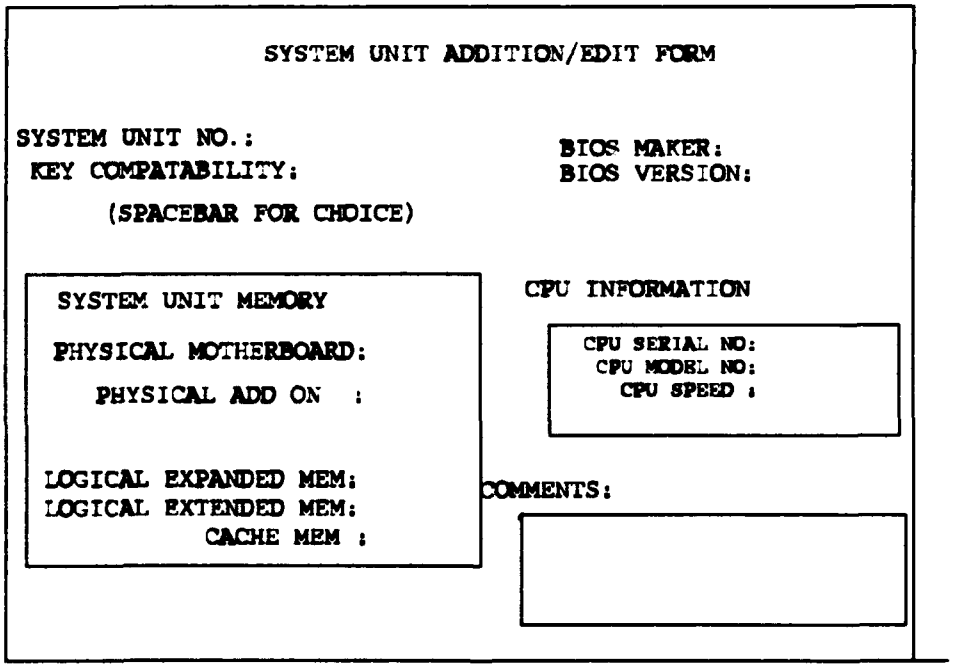





FLOPPY DRIVE ENTRY/EDIT/QUERY FORM
SYSTEM UNIT NO:
DRIVE TYPE:
DRIVE ID NO:
DRIVE LETTER:
COMMENTS:
PRESS RETURN TO ENTER DATA AND MOVE TO NEXT FIELD
TOGGLE SPACEBAR FOR CHOICES WHERE AVAILABLE
EACH COMMENT LINE REQUIRES A RETURN COMMAND


HARD DRIVE ADDITION/EDIT FORM
SYSTEM UNIT NO:
DRIVE ID NO:
DRIVE INFO
DRIVE TYPE:
DRIVE SIZE:
HEADS:
DRIVE LETTER:
CYLINDERS:
SECTORS:
LAND ZONE:
WRITE PRE
COMPRESSION:
PRESS RETURN TO MOVE TO NEXT FIELD
COMMENT FIELD REQUIRES THREE RETUR
COMMENTS:


NODE INFORMATION ENTRY/EDIT FORM
SYSTEM UNIT NO:
NODE ID NO :
LAN:
PURPOSE:
POR POP UP CHOICES OF LANS
**PRESS ENTER WHILE IN LAN FIELD*
COMMENTS:


SYSTEM UNIT ADDITION/EDIT FORM
SYSTEM UNIT NO.:
KEY COMPATABILITY:
(SPACEBAR FOR CHOICE)
BIOS MAKER:
BIOS VERSION:
SYSTEM UNIT MEMORY
PHYSICAL MOTHERBOARD:
PHYSICAL ADD ON :
LOGICAL EXPANDED MEM:
LOGICAL EXTENDED MEM:
CACHE MEM :
CPU INFORMATION
CPU SERIAL NO:
CPU MODEL NO:
CPU SPEED :
COMMENTS:

APPENDIX D-RELATION DEFINITIONS

RELATION	ATTRIBUTE	LENGTH	TYPE
LAN	LAN_NAME	25	A/N
NODE	NODE_ID	10	A/N
	LAN_NAME	25	A/N
	SYS_UNITNO	15	A/N
	PURPOSE	10	A/N
	COMMENTS	50	A/N
SYSTEM UNIT	SYS_UNITNO	15	A/N
	CPU_MOD_NO	5	A/N
	CPU_SPEED	2	A/N
	CPU_SER_NO	10	A/N
	PHYS_MOTH	5	A/N
	LOG_EXP_MEM	5	A/N
	LOG_EXT_MEM	5	A/N
	PHY_ADDON	5	A/N
	CACHE_MEM	5	A/N
	BIOS_MAKER	10	A/N
	BIOS_VERS	5	A/N
	KEY_COMPAT	6	A/N
	COMMENTS	36	A/N
ACCESSORY	SYS_UNITNO	15	A/N
	ACCESS_ID	15	A/N
	ACCESS_NAM	15	A/N
	ACCESS_TYP	12	A/N
	IO_PORT	4	A/N
	IRQ_SET	2	A/N
	DMA_SET	8	A/N
	IO_ADDRESS	8	A/N
	QUANTITY	3	A/N
	COMMENTS	40	A/N

RELATION	ATTRIBUTE	LENGH	TYPE
DRIVE	SYS_UNITNO	15	A/N
	DRIVE_ID	15	A/N
	DRIVE_LTR	1	A/N
	DRIVE_TYPE	15	A/N
	HEADS	2	A/N
	CYLINDERS	4	A/N
	SECTORS	2	A/N
	LAND_ZONE	4	A/N
	WR_PRECOM	4	A/N
	SIZE	6	A/N
	COMMENTS	35	A/N
ACCESSORY NAME	ACCESS_NAM	15	A/N
APP SOFTWARE	SW_NAME	20	A/N
	DEVELOPER	15	A/N
	QUANTITY	3	A/N
CABLE PLANT	CABLE_TYPE	20	A/N
	QUANTITY	3	A/N
CABLE LAN	LAN_NAME	25	A/N
	CABLE_TYPE	20	A/N
	QUANTITY	3	A/N

RELATION	ATTRIBUTE	LENGHT	TYPE
NODE-APP	SW_NAME	20	A/N
	NODE_ID	10	A/N
	LAN_NAME	25	A/N
NODE_SYS	SYS_SW_NAM	20	A/N
	NODE_ID	10	A/N
	LAN_NAME	25	A/N
SYS SOFTWARE	SYS_SW_NAM	20	A/N
	DEVELOPER	15	A/N
	QUANTITY	3	A/N

APPENDIX E-PROGRAM FILES AND DEFINITIONS

A. DATABASE FILES

- **Accessor.dbf**: Database file for the accessory object.
- **Acc_Name.dbf**: Database file for the accessory name key field choices.
- **App_SW.dbf**: Database file for the application software key fields. This database also holds the spare application software information.
- **Cable_Pl.dbf**: Database file for the cable plant key fields. This database also holds the spare cable plant information.
- **Cabl_Lan**: Intersection relation for the Cable_Pl and LAN database files. Holds the data for what LANs have what amount of cable plant equipment.
- **Drive**: Database file for the drive object.
- **LAN**: Database file for the LAN names (and also the key fields).
- **Node**: Database file for the node object.
- **Node_App**: Intersection relation for the Node.dbf and App_SW database files. Describes what application software is assigned to each node.
- **Node_Sys**: Intersection relation for the Node and Sys_SW database files. Describes what system software is assigned to each node.
- **Sys_SW**: Database file for the system software key field names. Also holds the spare system software information.
- **Sys_Unit**: Database file for the system unit object.

B. MENU FILES

- **Main**: The main bar menu for the application.
- **Accessor**: Pop up menu which is attached to the LAN pull down menu. Allows the user to edit, delete and add accessories to system units in the database.
- **Appsoft**: Pop up menu which is attached to the LAN pull down menu. Allows the user to add or delete application software from a node.
- **Cable_Pl**: Pop up menu which is attached to the LAN pull down menu. Allows the user to add, delete or query cable plant equipment associated with a LAN.
- **Drives**: Pop up menu which is attached to the LAN pull down menu. Allows the user to add, edit/query or delete a drive from a system unit.
- **Drive_Sp**: Pop up menu which is attached to the Spares pull down menu. Allows the user to add, edit/query or delete a spare drive.
- **Equ_Rep**: Pop up menu which is attached to the Report pull down menu. Allows the user to choose from a variety of reports on LAN equipment.
- **LAN**: One of the pull down menus attached to the main bar menu. Concerned with LAN equipment configuration functions.
- **Lan_Rep**: Pop up menu which is attached to the Report pull down menu. Allows the user to choose from a variety of reports on Lan configuration.
- **Maint**: Pop up menu which is attached to the Quit pull down menu. Allows the user to perform backups and packs on the database files.
- **Nodes**: Pop up menu which is attached to the LAN pull down menu. Allows the user to add, delete, query nodes and change the system unit assigned to a node.
- **NuNode**: Pop up menu which is attached to the Node pop up menu. Allows the user to add a new node and either assign to it a new system unit or add a system unit already present in the database.

- **Quit**: One of the pull down menus attached to the main bar menu. Concerned with database maintenance and exiting procedures from the application.

- **Reports**: One of the pull down menus attached to the main bar menu. Concerned with formatted reports of the database.

- **Spares**: One of the pull down menus attached to the main bar menu. Concerned with addition, deletion and query/edit of the LAN spares.

- **Spare_Rep**: Pop up menu which is attached to the Spare pull down menu. Allows the user to generate various reports on the system's spare parts.

- **Sys_Soft**: Pop up menu which is attached to the LAN pull down menu. Allows the user to add or delete system software for either a node or a whole LAN.

- **Sys_Uni**: Pop up menu which is attached to the LAN pull down menu. Allows the user to query, edit, remove a system unit from a node and add a system unit to a node.

- **Unit_Spa**: Pop up menu which is attached to the Spares pull down menu. Allows the user to add, delete, edit and query spare system units.

C. DATA ENTRY FORMAT FILES

- **Accssy**: Data entry/edit/query form for the accessory object.

- **App_Spar**: Data entry/edit/query form for the application software spare copies.

- **Cable**: Data entry/edit/query form for the cable plant equipment associated with a LAN.

- **Cab_Spar**: Data entry/edit/query form for the cable plant spare parts.

- **Floppy**: Data entry/edit/query form for floppy disk drives.

- **HrdDrive**: Data entry/edit/query form for hard drives.

- **Node_Pic**: Data entry/edit/query form for the node object.

- **Sys_spar**: Data entry/edit/query form for spare system software.

- **Sys_Unit**: Data entry/edit/query form for system unit object.

D. REPORT FORMAT FILES

- **Ac_e_rep**: Creates report on all accessories in the database.

- **Ac_S_Rep**: Creates report on spare system units in the database.

- **Ap_L_Rep**: Creates a report on the application software available in a given LAN.

- **Ap_S_Rep**: Creates a report on the spare application software in the database.

- **Cab_S_Rep**: Creates a report on the spare cable plant equipment in the database.

- **CB_L_Rep**: Creates a report on the cable plant equipment assigned to a given LAN.

- **Dr_E_Rep**: Creates a report on all the drives in the entire LAN.

- **Dr_S_Rep**: Creates a report on spare drives in the database.

- **LanRep**: Creates a report on the equipment and software available in a given Lan.

- **Nod_E_Re**: Creates a report on all the nodes in the database.

- **Nod_L_Re**: Creates a report on all the nodes in a given LAN.

- **Serv_Rep**: Creates a report on the server(s) in a given LAN.

- **Sy_S_Rep**: Creates a report on the spare system software in the database.

- **Sy_L_Rep**: Creates a report on the System software used in a given LAN.

- **Un_e_Rep**: Creates a report on all of the system units in the database.
- **Un_S_Re**: Creates a report on spare system units.

E. SYSTEM APPLICATION FILES

- **Acc_Ed**: Edit/Query an accessory.
- **Acc_Ent**: Add an accessory to a system unit.
- **Acc_Del**: Delete an accessory from a system unit.
- **Acc_s_Ed**: Edit/query an accessory spare.
- **Ac_S_Del**: Delete a spare accessory.
- **Ac_S_Ent**: Add a spare accessory.
- **AddApp**: Add application software to a node.
- **Add_Node**: Adds a node to a LAN.
- **Add_Sys**: Adds system software to a node.
- **App_Del**: Deletes application software from a node.
- **App_Ent**: Adds application software to a node.
- **App_Spa**: Add/edit/delete spare application software.
- **Cabl_Add**: Adds cable plant equipment to a LAN.
- **Cabl_Del**: Deletes cable plant equipment from a LAN.
- **Cabl_Sp**: Add/edit/deletes spare cable plant equipment.
- **Driv_Del**: Deletes a drive from a system unit.
- **Driv_Ent**: Adds a drive to a system unit.
- **Drv_Quer**: Queries/edits a drive on a system unit.
- **Dr_S_Del**: Deletes a spare drive from the system.
- **Dr_S_Ent**: Adds a spare drive to the system.
- **Dr_S_Que**: Queries/edits a spare drive.

- **Get_Unit**: A program used by other application programs to locate a system unit by either its node or its system unit number.
- **Lanmaint**: The main application program for the LAN maintenance and configuration database.
- **Nodent_1**: Adds a new node and an associated system unit to a LAN.
- **Nodent_2**: Adds a new node to a LAN, but uses a system unit already present in the database.
- **Node_Ch**: Changes the system unit assigned to a node.
- **Node_Del**: Deletes a node from the database.
- **Node_Ed**: Edits a node.
- **Node_Que**: Queries a node.
- **N_Choice**: Program used by other programs to choose a node from a series of pop up menus.
- **Spare_Qu**: Queries a spare system unit.
- **Sys_Del**: Deletes system software from a node.
- **Sys_Ent**: Enters system software onto a node.
- **Sys_Spa**: Allows edit/delete/adding of spare system software to the system.
- **Unit_2**: Used to add spare system units inside other programs.
- **Unit_Add**: Assigns a system unit to a node.
- **Unit_Ed**: Allows the editing of system units.
- **Unit_Ent**: Adds spare system units to the system.
- **Unit_Qu**: Queries a system unit.
- **Unit_Rem**: Removes a system unit from a node.
- **Un_Sp_Dl**: Deletes a spare system unit.
- **Un_Sp_Ed**: allows editing of a spare system unit.

APPENDIX F- APPLICATION PROGRAM FILES

1. Acc_Ed

```
   ************************************************************
   ***************
*   Program Name :  ACC_ED.prg
*   Author       :  D. G. Dickison
*   Date         :  29 July 1992
*   Version      :  Dbase IV Ver 1.1
*   Last Updated :  07 August 1992
*
*   Purpose      :  Edit/Query an Accessory record
*   Programs
*   Called       :  Get_Unit
************************************************************
*************
CLEAR
DO Get_unit
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
CLOSE DATABASES
USE Accessor ORDER Sys_unitno
SET FILTER TO Sys_unitno = Sys_un_num
SEEK Sys_un_num
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY "THERE ARE NO ACCESSORIES ASSOCIATED WITH
SYSTEM UNIT "+TRIM(Sys_un_num)
   @ 12, 17 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
   WAIT " "
   CLOSE DATABASES
   RETURN
ENDIF
CLEAR
@ 8, 5 SAY "CHOSE THE ACCESSORY TO QUERY/EDIT FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
@ 9, 19 SAY "WITH CURSOR KEYS AND PRESS ENTER"
DEFINE POPUP Get_ACC FROM 10, 20 PROMPT FIELD Access_Nam
ON SELECTION POPUP Get_ACC DEACTIVATE POPUP
ACTIVATE POPUP Get_ACC
USE Accessor ORDER Access_Nam
SET FILTER TO Sys_unitno = Sys_un_num
```

```
SEEK PROMPT()
SET FORMAT TO Accssy
READ
CLOSE FORMAT
CLOSE DATABASES
CLEAR
RETURN
```

2. Acc_ent

```
  ************************************************************
  ***************
***          ACCESSORY ENTRY PROGRAM
         ***
************************************************************
*************
*Program Name:  ACC_ENT.prg
*Author       :  D. G. Dickison
*Date         :  16 July 1992
*Version      :  Dbase IV Ver 1.1
*Last Updated:  09 August 1992
*
*Purpose      :  This program adds accessories such as memory
cards,
*             monitors, modems etc to a system unit by either
node
*              ID  or system unit no.
* Programs called:  Get_Unit
************************************************************
************
CLEAR
DO Get_Unit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
YN = "Y"
   DO WHILE UPPER(YN) = "Y"
   CLEAR
   USE Acc_Name ORDER Access_Nam
   @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE DESIRED ACCESSORY"
   @ 9, 12 SAY " AND PRESS RETURN"
   DEFINE POPUP Acc_Nam FROM 10, 20 PROMPT FIELD Access_Nam
   ON SELECTION POPUP Acc_Nam DEACTIVATE POPUP
   ACTIVATE POPUP Acc_Nam
   Nu_Acc = PROMPT()
   CLOSE DATABASE
```

```
   USE Accessor
   CLEAR
   SET FORMAT TO Accssy
   APPEND BLANK
   REPLACE Access_nam WITH Nu_Acc
   REPLACE Sys_unitno WITH UPPER(Sys_un_num)
   READ
   CLOSE FORMAT
   CLOSE DATABASE
   CLEAR
   ACCEPT "ENTER 'Y' TO ADD ANOTHER ACCESSORY, ANY OTHER KEY
TO EXIT" TO YN
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

3. Acc_del

```
  ************************************************************
  ***************
*  Program Name :  ACC_DEL.prg
*  Author       :  D. G. Dickison
*  Date         :  29 July 1992
*  Version      :  Dbase IV Ver 1.1
*  Last Updated :  07 August 1992
*
*  Purpose      :  To query accessories of a system unit
*  Programs
*  Called       :  Get_Unit
************************************************************
************
CLEAR
DO Get_Unit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
YesNo = "Y"
DO WHILE UPPER(YesNO) = "Y"
   Use Accessor ORDER Sys_unitno
   SET FILTER TO Sys_unitno = Sys_un_num
   SEEK Sys_un_num
   IF .NOT. FOUND()
      CLEAR
```

```
      @ 10, 5 SAY "THERE ARE NO ACCESSORIES ASSOCIATED WITH
SYSTEM UNIT "+TRIM(Sys_un_num)
      @ 12,  17 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"

      WAIT " "
      CLOSE DATABASES
      CLEAR
      RETURN
   ENDIF
   CLEAR
   CLOSE DATABASES
   USE Accessor ORDER Access_Nam
   SET FILTER TO Sys_unitno = Sys_un_num
   @ 8, 5 SAY "CHOOSE THE ACCESSORY TO DELETE FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
   @ 9, 17 SAY "WITH CURSOR KEYS AND PRESS ENTER"
   DEFINE POPUP Get_Acc FROM 10, 20 PROMPT FIELD Access_Nam
   ON SELECTION POPUP Get_Acc DEACTIVATE POPUP
   ACTIVATE POPUP Get_Acc
   CLEAR
   ACCEPT "ARE YOU SURE??  ENTER 'Y' TO CONTINUE OR ANY OTHER
KEY TO ABORT" TO YN
   IF UPPER(YN) = "Y"
      SEEK PROMPT()
      DELETE
      CLEAR
      @ 10, 5 SAY TRIM(PROMPT())+" HAS BEEN REMOVED FROM
SYSTEM UNIT "+TRIM(Sys_un_num)
      @ 12, 16 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "

   ELSE
      CLEAR
      @ 10, 5 SAY "PROCEDURE ABORTED,,,RETURNING TO MAIN MENU"
      @ 12, 17 SAY "PRESS ANY KEY"
      WAIT " "
      CLOSE DATABASES
      CLEAR
      RETURN
   ENDIF
   YesNo = SPACE(1)
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO DELETE MORE ACCESSORIES FROM
SYSTEM UNIT"+TRIM(Sys_un_num)
   @ 12, 12 SAY "OR ANY OTHER KEY TO EXIT TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "Y"
         CLEAR
      OTHERWISE
```

```
         CLEAR
         @ 10, 5 SAY "NO FURTHER ACCESSORIES TO BE DELETED
FROM "+TRIM(Sys_Un_num)
         @ 12, 8 SAY "PRESS ANY KEY TO EXIT TO PREVIOUS MENU"
         WAIT " "
   ENDCASE
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

4. Acc_S_Ed

```
   ************************************************************
   ***************
*   Program Name :  ACC_S_ED.prg
*   Author       :  D. G. Dickison
*   Date         :  4 SEP 1992
*   Version      :  Dbase IV Ver 1.1
*   Last Updated :
*
*   Purpose      :  Edit/Query an Accessory spare record
*   Programs
*   Called       :
************************************************************
*************
USE Acc_Name ORDER Access_Nam
CLEAR
@ 8, 5 SAY "CHOSE THE SPARE ACCESSORY CATEGORY TO QUERY/EDIT"
@ 9, 11 SAY "WITH CURSOR KEYS AND PRESS ENTER"
DEFINE POPUP Get_ACC FROM 10, 20 PROMPT FIELD Access_Nam
ON SELECTION POPUP Get_ACC DEACTIVATE POPUP
ACTIVATE POPUP Get_ACC
Acc_Cat = PROMPT()
USE Accessor ORDER Access_Nam
SET FILTER TO Sys_unitno = "SPARE" .AND. Access_Nam = Acc_Cat
SEEK Acc_Cat
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY "THERE ARE NO SPARE "+TRIM(Acc_Cat)+" IN THE
SPARE DATABASE"
   @ 12, 9 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
   WAIT " "
   CLEAR
   CLOSE DATABASES
   RETURN
ENDIF
USE Accessor ORDER Access_ID
```

```
SET FILTER TO Sys_unitno = "SPARE" .AND. Access_Nam = Acc_Cat
CLEAR
@  8,  5  SAY  "USE  CURSOR  KEYS  TO  ENTER  THE  DESIRED
"+TRIM(Acc_Cat)
@ 9, 15 SAY "AND PRESS RETURN"
DEFINE POPUP Get_Spare FROM 10, 20 PROMPT FIELD Access_ID
ON SELECTION POPUP Get_Spare DEACTIVATE POPUP
ACTIVATE POPUP Get_Spare
SEEK PROMPT()
SET FORMAT TO Accssy
READ
CLOSE FORMAT
CLOSE DATABASES
CLEAR
RETURN
```

5. Acc_S_Del

```
  ************************************************************
  ***************
*  Program Name :  AC_S_DEL.prg
*  Author       :  D. G. Dickison
*  Date         :  4 SEP 1992
*  Version      :  Dbase IV Ver 1.1
*  Last Updated :
*
*  Purpose      :  Edit/Query an Accessory record
*  Programs
*  Called       :
************************************************************
*************
USE Acc_Name ORDER Access_Nam
CLEAR
@ 8, 5 SAY "CHOSE THE SPARE ACCESSORY CATEGORY TO DELETE"
@ 9, 11 SAY "WITH CURSOR KEYS AND PRESS ENTER"
DEFINE POPUP Get_ACC FROM 10, 20 PROMPT FIELD Access_Nam
ON SELECTION POPUP Get_ACC DEACTIVATE POPUP
ACTIVATE POPUP Get_ACC
Acc_Cat = PROMPT()
USE Accessor ORDER Access_Nam
SET FILTER TO Sys_unitno = "SPARE" .AND. Access_Nam = Acc_Cat
SEEK Acc_Cat
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY "THERE ARE NO SPARE "+TRIM(Acc_Cat)+" IN THE
SPARE DATABASE"
   @ 12, 9 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
   WAIT " "
   CLEAR
```

```
   CLOSE DATABASES
   RETURN
ENDIF
USE Accessor ORDER Access_ID
SET FILTER TO Sys_unitno = "SPARE" .AND. Access_Nam = Acc_Cat
CLEAR
@  8,  5  SAY  "USE  CURSOR  KEYS  TO  ENTER  THE  DESIRED
"+TRIM(Acc_Cat)
@ 9, 15 SAY "AND PRESS RETURN"
DEFINE POPUP Get_Spare FROM 10, 20 PROMPT FIELD Access_ID
ON SELECTION POPUP Get_Spare DEACTIVATE POPUP
ACTIVATE POPUP Get_Spare
SEEK PROMPT()
CLEAR
ACCEPT "ARE YOU SURE?? ENTER 'Y' TO CONTINUE OR ANY OTHER KEY
TO ABORT" TO YN
IF UPPER(YN) = "Y"
   DELETE
   CLEAR
   @ 10, 5 SAY "ACCESSORY "+TRIM(PROMPT())+" HAS BEEN DELETED
FROM THE SPARES"
   @ 12, 15 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
   WAIT " "
ELSE
   CLEAR
   @ 10,  5  SAY  "PROCEDURE  ABORTED,,,RETURNING  TO  PREVIOUS
MENU"
   @ 12, 10 SAY " PRESS ANY KEY TO CONTINUE"
   WAIT " "
ENDIF
CLEAR
CLOSE DATABASES
RETURN
```

6. Ac_S_Ent

```
  ************************************************************
  ***************
***          ACCESSORY ENTRY PROGRAM
***
************************************************************
*************
*Program Name:  AC_S_ENT.prg
*Author       :  D. G. Dickison
*Date         :  7 SEPT 1992
*Version      :  Dbase IV Ver 1.1
*Last Updated:
*
```

```
*Purpose      :  This program adds spare accessories such as
memory cards,
*             monitors, modems etc to the database.
*
* Programs called:
*****************************************************************
************
CLEAR
YN = "Y"
   DO WHILE UPPER(YN) = "Y"
   CLEAR
   USE Acc_Name ORDER Access_Nam
   @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE DESIRED ACCESSORY"
   @ 9, 12 SAY " AND PRESS RETURN"
   DEFINE POPUP Acc_Nam FROM 10, 20 PROMPT FIELD Access_Nam
   ON SELECTION POPUP Acc_Nam DEACTIVATE POPUP
   ACTIVATE POPUP Acc_Nam
   Nu_Acc = PROMPT()
   CLOSE DATABASE
   USE Accessor
   CLEAR
   SET FORMAT TO Accssy
   APPEND BLANK
   REPLACE Access_nam WITH Nu_Acc
   REPLACE Sys_unitno WITH "SPARE"
   READ
   CLOSE FORMAT
   CLOSE DATABASE
   CLEAR
   ACCEPT "ENTER 'Y' TO ADD ANOTHER ACCESSORY, ANY OTHER KEY
TO EXIT" TO YN
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

7. Add_app

```
*****************************************************************
**************
* Program Name:  AddApp.prg
        *
* Author       :  D. G. Dickison
        *
* Date         :  17 June 1992
        *
```

```
* Version      :  Dbase IV Ver 1.1
          *
* Last Update :  21 July 1992
          *
*
          *
* Purpose       :  Adds a type of Application software to a
Node.          *
*****************************************************************
************
CLEAR
CLOSE DATABASES
USE App_Sw ORDER Sw_Name
@ 8, 5 SAY "USE CURSOR KEYS TO MAKE CHOICE OF OF SOFTWARE"
@ 9, 5 SAY "AND PRESS ENTER TO ADD IT TO NODE "+UPPER(NuNode)
DEFINE POPUP GetApp FROM 10,20 PROMPT FIELD Sw_Name ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
ON SELECTION POPUP GetApp DEACTIVATE POPUP
ACTIVATE POPUP GetApp
NuName = PROMPT()
CLEAR
CLOSE DATABASES
USE Node_App
APPEND BLANK
REPLACE Node_ID WITH UPPER(NuNode)
REPLACE Sw_Name WITH NuName
REPLACE Lan_Name WITH NuLan
? CHR(13)
CLOSE DATABASES
CLEAR
@ 10, 5 SAY  TRIM(NuName)+" ADDED TO NODE "+NuNode
@ 12, 10 SAY "PRESS ANY KEY TO  CONTINUE"
WAIT " "
CLEAR
RETURN
```

8. Add_Node

```
*****************************************************************
***************
* Program Name:  Add_Node.prg
* Author       :  D. G. Dickison
* Date         :  21 July 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  21 July 1992
*
* Purpose      :  Add a Node to the LAN Maintenance Database
```

```
*
************************************************************************
************
CLEAR
PUBLIC NuLan
USE Node
SET FORMAT TO Node_Pic
APPEND BLANK
REPLACE Node_ID WITH UPPER(NuNode)
REPLACE Sys_Unitno WITH UPPER(Sys_un_num)
READ
NuLan = Lan_Name
CLOSE FORMAT
CLOSE DATABASE
CLEAR
RETURN
```

9. Add_Sys

```
  ************************************************************************
  ***************
* Program Name:  ADD_SYS.prg
* Author       :  D. G. Dickison
* Date         :  21 July 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  21 July 1992
*
* Purpose      :  Add a type of System  Software to a Node
************************************************************************
************
CLEAR
CLOSE DATABASES
USE Sys_Sw ORDER Sys_Sw_Nam
DEFINE POPUP GetSys FROM 10,20 PROMPT FIELD Sys_Sw_Nam ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
ON SELECTION POPUP GetSys DEACTIVATE POPUP
@ 8, 8 SAY "USE CURSOR KEY TO HIGHLIGHT SOFTWARE CHOICE"
@ 9, 5 SAY "AND PRESS ENTER TO ADD SOFTWARE TO THE GIVEN NODE"
ACTIVATE POPUP GetSys
NuName = PROMPT()
CLEAR
CLOSE DATABASES
USE Node_Sys
APPEND BLANK
REPLACE Node_ID WITH UPPER(NuNode)
REPLACE Sys_Sw_Nam WITH NuName
REPLACE Lan_Name WITH NuLan
```

```
? CHR(13)
CLOSE DATABASES
CLEAR
@ 10, 5 SAY TRIM(NuName)+" ADDED TO NODE "+NuNode
WAIT
CLEAR
RETURN
```

10. App_Del

```
   ************************************************************
   ***************
* Program Name :  APP_DEL.prg
* Author       :  D. G. Dickison
* Date         :  27 July 1992
* Last Updated :  27 July 1992
*
* Purpose      :  Delete an application program from a node
************************************************************
*************
CLEAR
***************CALL THE N_CHOICE PROGRAM WHICH CHOOSES A NODE
DO N_Choice
CLEAR
CLOSE DATABASES
YesNo = "Y"
NuNode = PROMPT()
DO WHILE UPPER(YesNo) = "Y"
   USE Node_APP ORDER Node_ID
   SET FILTER TO Node_ID = Nunode
   ************** CHECK FOR AN EMPTY SET CONDITION (NO
SOFTWARE)
   SEEK NuNode
   IF .NOT. FOUND()
      CLEAR
      @ 10, 5 SAY "NO APPLICATION SOFTWARE IS PRESENTLY
ASSIGNED TO NODE "+TRIM(NuNode)
      @ 12, 13 SAY "PRESS ANY KEY TO RETURN TO THE PREVIOUS
MENU"
      WAIT " "
      RETURN
   ENDIF
   CLEAR
   USE Node_App ORDER SW_Name
   SET FILTER TO Node_ID = NuNode
   @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE AND SOFTWARE TO BE
DELETED FROM "+TRIM(NuNode)
   @ 9, 11 SAY "AND PRESS ENTER TO DELETE SOFTWARE"
   DEFINE POPUP Get_App FROM 10, 20 PROMPT FIELD SW_Name
```

```
   ON SELECTION POPUP Get_App DEACTIVATE POPUP
   ACTIVATE POPUP Get_App
   ********** SET UP LAST CHANCE
   CLEAR
   ACCEPT "ARE YOU SURE?? ENTER 'Y' TO CONTINUE OR ANY OTHER
KEY TO ABORT" TO YN
   IF UPPER(YN) = "Y"
      SEEK PROMPT()
      DELETE
      CLEAR
      @ 10, 5 SAY TRIM(PROMPT())+" HAS BEEN REMOVED FROM
"+TRIM(NuNode)+" NODE"
      @ 12, 15 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
   ELSE
      CLEAR
      @ 10, 5 SAY "PROCEDURE ABORTED,,, RETURNING TO MAIN
MENU"
      @ 12, 5 SAY "PRESS ANY KEY TO RETURN TO MAIN MENU"
      WAIT " "
      RETURN
   ENDIF
   YesNo = SPACE(1)
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO DELETE MORE APPLICATION SOFTWARE
FROM "+TRIM(NuNode)
   @ 12, 12 SAY "OR ANY OTHER KEY TO EXIT TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNO) = "Y"
         CLEAR
      OTHERWISE
         CLEAR
         @ 10 ,5 SAY "NO FURTHER SOFTWARE CHOSEN TO BE
DELETED"
         @ 12, 5 SAY "PRESS ANY KEY TO EXIT TO PREVIOUS MENU"
         WAIT " "
      ENDCASE
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

11. App_Ent

```
*****************************************************************
***************
```

```
* Program Name :  APP_ENT.prg
* Author       :  D. G. Dickison
* Date         :  Version
* Last Updated :  21 July 1992
*
* Purpose      :  Adds Application software to a Node.
************************************************************
*************
CLEAR
CLOSE DATABASES
******* CREATE A POPUP TO CHOOSE THE LAN WHICH THE NODE IS IN
USE Lan ORDER Lan_Name
@ 8, 9 SAY "USE CURSOR KEYS TO CHOOSE LAN WHICH THE NODE IS
IN"
@ 9, 18 SAY "AND PRESS ENTER FOR NODE CHOICES"
DEFINE POPUP GetLan FROM 10,20 PROMPT FIELD Lan_Name ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
ON SELECTION POPUP GetLan DEACTIVATE POPUP
ACTIVATE POPUP GetLan
NuLan = PROMPT()
CLEAR
******* CREATE A POPUP TO CHOOSE THE NODE THE SOFTWARE IS TO
BE ADDED TO
CLOSE DATABASES
USE Node ORDER Lan_Name
SET FILTER TO Lan_Name = PROMPT()
SEEK PROMPT()
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY " NO NODES ARE PRESENTLY ASSIGNED TO THE
"+TRIM(PROMPT())+" LAN"
   @ 12, 13 SAY " PRESS ANY KEY TO RETURN TO THE PREVIOUS
MENU"
   WAIT " "
   RETURN
ENDIF
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE NODE WHICH THE
SOFTWARE "
@ 9, 5 SAY "IS TO BE ADDED AND PRESS ENTER FOR SOFTWARE
CHOICES"
DEFINE POPUP Get_Nod FROM 10, 20 PROMPT FIELD Node_ID ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
ON SELECTION POPUP Get_Nod DEACTIVATE POPUP
ACTIVATE POPUP Get_Nod
NuNode = PROMPT()
CLEAR
CLOSE DATABASES
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   DO Add_App
   YesNo = SPACE(1)
```

```
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO ADD APPLICATION SOFTWARE TO
CHOSEN NODE"
   @ 12, 19 SAY "OR 'N' TO EXIT TO MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "N"
         CLEAR
         @ 10, 5 SAY " NO FURTHER SOFTWARE CHOSEN TO BE
ENTERED"
         @ 12, 5 SAY " PRESS ANY KEY TO RETURN TO PREVIOUS
MENU"
         WAIT " "
      CASE UPPER(YesNo) = "Y"
         CLEAR
      OTHERWISE
         CLEAR
         @ 10, 5 SAY "INVALID ENTRY,,, RETURNING TO MAIN MENU"
         WAIT
   ENDCASE
ENDDO
CLEAR
RETURN
```

12. App_Spa

```
   ************************************************************
   ***************
* Program Name:  APP_SPA.prg
* Author       :  D. G. Dickison
* Date         :  2 Sept 1992
* Version      :  Dbase IV Ver 1.1
* Last Update :
* Purpose      :  Allows user to add/subtract/edit the number
of application
*                 software spares.
************************************************************
*************
CLEAR
USE App_SW ORDER SW_Name
@ 8, 5 SAY "USE CURSORS TO CHOOSE DESIRED APPLICATION
SOFTWARE"
@ 9, 17 SAY "AND PRESS ENTER FOR CHOICE"
DEFINE POPUP Get_App FROM 10, 20 PROMPT FIELD Sw_Name
ON SELECTION POPUP Get_App DEACTIVATE POPUP
ACTIVATE POPUP Get_App
```

```
SEEK PROMPT()
SET FORMAT TO App_Spar
READ
CLOSE FORMAT
CLOSE DATABASE
CLEAR
RETURN
```

13. Cabl_Add

```
   ************************************************************
   ***************
*  Program Name:  CABL_ADD.prg
*  Author       :  D. G. Dickison
*  Date         :  10 August 1992
*  Version      :  Dbase IV Ver 1.1
*  Last Updated:
*
*  Purpose      :  Add Cable Plant Equipment to a Local Area
Network
*
************************************************************
*************
CLEAR
******** CHOOSE THE LAN TO ADD CABLE PLANT DATA TO
USE Lan ORDER Lan_Name
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE DESIRED LAN AND
PRESS ENTER"
DEFINE POPUP Get_Lan FROM 10, 20 PROMPT FIELD Lan_Name
ON SELECTION POPUP Get_Lan DEACTIVATE POPUP
ACTIVATE POPUP Get_Lan
CLEAR
Nu_Lan = PROMPT()
CLOSE DATABASE
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   CLEAR
   **********  CHOOSE THE CABLE PLANT EQUIPMENT TO ADD
   USE Cable_Pl ORDER Cable_Type
   @ 8, 5 SAY "USE CURSORS TO CHOOSE DESIRED CABLE PLANT
EQUIPMENT"
   @ 9, 18 SAY " AND PRESS ENTER"
   DEFINE POPUP Get_Cab FROM 10, 20 PROMPT FIELD Cable_Type
   ON SELECTION POPUP Get_Cab DEACTIVATE POPUP
   ACTIVATE POPUP Get_Cab
   Nu_Cable = PROMPT()
   CLOSE DATABASE
```

```
   USE Cabl_Lan ORDER Cable_Type
   SET FILTER TO Lan_Name = Nu_Lan
   SEEK Nu_Cable
   SET FORMAT TO Cable
   ********** IF NO PREVIOUS CABLE PLANT OF THAT TYPE, APPEND
   IF .NOT. FOUND()
      APPEND BLANK
      REPLACE Lan_Name WITH Nu_Lan
      REPLACE Cable_Type WITH Nu_Cable
      READ
   ELSE
      ********* CABLE PLANT ALREADY EXISTS ON GIVEN LAN, EDIT
NO.
      READ
   ENDIF
   CLOSE FORMAT
   CLOSE DATABASE
   YesNo = SPACE(1)
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO ENTER FURTHER CABLE PLANT
EQUIPMENT ON THE"
   @ 12, 5 SAY  TRIM(Nu_Lan)+" LAN OR ANY OTHER KEY TO RETURN
TO MENU";
   GET YesNo PICTURE "X"
   READ
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

14. Cabl_Del

```
  ************************************************************
  ***************
* Program Name :  CABL_DEL.prg
* Author       :  D. G. Dickison
* Date         :  10 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
* Purpose      :  Remove assigned cable plant equipment from
a LAN
************************************************************
*************
CLEAR
********* CHOOSE THE DESIRED LOCAL AREA NETWORK
USE Lan ORDER Lan_Name
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE DESIRED LAN AND
PRESS ENTER"
DEFINE POPUP Get_Lan FROM 10, 20 PROMPT FIELD Lan_Name
```

```
ON SELECTION POPUP Get_Lan DEACTIVATE POPUP
ACTIVATE POPUP Get_Lan
Nu_Lan = PROMPT()
CLEAR
CLOSE DATABASE
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   CLEAR
   *********CHOOSE THE DESIRED CABLE PLANT EQUIPMENT
   USE Cabl_Lan ORDER  Lan_Name
   SET FILTER TO Lan_Name = Nu_Lan
   SEEK Nu_Lan
   IF .NOT. FOUND()
      CLEAR
      @ 10, 5 SAY " THERE IS NO CABLE PLANT EQUIPMENT
PRESENTLY ASSIGNED"
      @ 12, 12 SAY " TO THE "+TRIM(Nu_Lan)+" NETWORK"
      @ 14, 12 SAY " PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
      WAIT " "
      CLOSE DATABASES
      CLEAR
      RETURN
   ENDIF
   @ 8, 5 SAY "USE CURSORS TO CHOOSE DESIRED CABLE PLANT
EQUIPMENT"
   @ 9, 5 SAY "FOR LAN "+TRIM(Nu_Lan)+" AND PRESS ENTER"
   DEFINE POPUP Get_Cab FROM 10, 20 PROMPT FIELD Cable_Type
   ON SELECTION POPUP Get_Cab DEACTIVATE POPUP
   ACTIVATE POPUP Get_Cab
   Nu_Cable = PROMPT()
   CLOSE DATABASE
   USE Cabl_Lan ORDER Cable_Type
   SEEK Nu_Cable
   SET FORMAT TO Cable
   READ
   ********* CHECK TO SEE IF QUANTITY SET TO ZERO
   IF Quantity = 0
      DELETE
      CLEAR
   ENDIF
   CLOSE FORMAT
   CLOSE DATABASE
   YesNo = SPACE(1)
   CLEAR
   @ 10, 10 SAY "ENTER 'Y' TO REMOVE FURTHER CABLE PLANT
EQUIPMENT ON THE"
   @ 12, 5 SAY TRIM(Nu_Lan)+" LAN OR ANY OTHER KEY TO RETURN
TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
ENDDO
```

```
CLOSE DATABASES
CLEAR
RETURN
```

15. Cabl_Sp

```
*************************************************************
***************
* Program Name :  CABL_SP.prg
* Author       :  D. G. Dickison
* Date         :  2 Sept 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
*  Purpose          :   To add or delete spare cable plant
information from the
*                 database.
*************************************************************
*************
CLEAR
USE Cable_Pl ORDER Cable_Type
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE DESIRED CABLE PLANT"
@ 9,18 SAY "AND PRESS ENTER"
DEFINE POPUP Get_Cab FROM 10, 20 PROMPT FIELD Cable_Type
ON SELECTION POPUP Get_Cab DEACTIVATE POPUP
ACTIVATE POPUP Get_Cab
SEEK PROMPT()
SET FORMAT TO Cab_Spar
READ
CLOSE FORMAT
CLOSE DATABASE
RETURN
```

16. Driv_Del

```
*************************************************************
***************
* Program Name:  DRIV_DEL.prg
* Author       :  D. G. Dickison
* Date         :  04 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  04 August 1992
*
* Purpose      :  To delete a drive from a system unit
*
* Programs called: Get_Unit
```

```
*****************************************************************
*************
CLEAR
DO Get_Unit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   USE Drive ORDER Sys_unitno
   SEEK Sys_un_num
   IF .NOT. FOUND()
      CLEAR
      @ 10, 5 SAY "THERE ARE NO DRIVES ASSOCIATED WITH SYSTEM
UNIT "+TRIM(Sys_un_num)
      @ 12, 17 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
      WAIT " "
      CLOSE DATABASES
      CLEAR
      RETURN
   ENDIF
   CLEAR
   SET FILTER TO Sys_unitno = Sys_un_num
   @ 8, 5 SAY "CHOOSE THE DRIVE TO DELETE FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
   @ 9, 17 SAY "WITH CURSOR KEYS AND PRESS ENTER"
   DEFINE POPUP Get_Driv FROM 10, 20 PROMPT FIELD Drive_Ltr
   ON SELECTION POPUP Get_Driv DEACTIVATE POPUP
   ACTIVATE POPUP Get_Driv
   CLEAR
   ACCEPT "ARE YOU SURE??  ENTER 'Y' TO CONTINUE OR ANY OTHER
KEY TO ABORT" TO YN
   IF UPPER(YN) = "Y"
      LOCATE FOR Drive_Ltr = PROMPT()
      DELETE
      CLEAR
      @ 10, 5 SAY "DRIVE "+TRIM(PROMPT())+" HAS BEEN REMOVED
FROM SYSTEM UNIT "+TRIM(Sys_un_num)
      @ 12, 17 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
   ELSE
      CLEAR
      @ 10, 5 SAY "PROCEDURE ABORTED,,, RETURNING TO MAIN
MENU"
      @ 12, 16 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
      CLOSE DATABASES
      CLEAR
```

```
      RETURN
   ENDIF
   YesNo = SPACE(1)
   CLEAR
   @ 10 ,5 SAY "ENTER 'Y' TO DELETE MORE DRIVES FROM SYSTEM
UNIT "+TRIM(Sys_un_num)
   @ 12, 12 SAY "OR ANY OTHER KEY TO RETURN TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "Y"
         CLEAR
      OTHERWISE
         CLEAR
         @ 10, 5 SAY "NO FURTHER DRIVES TO BE DELETED FROM
"+TRIM(Sys_un_num)
         @ 12, 8 SAY "PRESS ANY KEY TO EXIT TO PREVIOUS MENU"
         WAIT " "
   ENDCASE
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

17. Drive_Ent

```
*************************************************************
*************
*Program Name:  DRIV_ENT.prg
*Author       :  D. G. Dickison
*Date         :  17 June 1992
*Version      :  Dbase IV Format 1.1
*Last Updated:  13 July 1992
*
*Purpose      :  This program adds disk drives to a system unit
*
*************************************************************
*************
CLEAR
DO Get_Unit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
```

```
ENDIF
YN = "Y"
DO WHILE UPPER(YN) = "Y"
   USE Drive
   CLEAR
   ** ASSIGN CORRECT FORMAT FOR APPENDING
   ACCEPT "ENTER 'H' FOR HARDDRIVE, ANY OTHER KEY FOR FLOPPY"
TO FH
   IF UPPER(FH) = "H"
   SET FORMAT TO Hrddrive
   ELSE
   SET FORMAT TO Floppy
   ENDIF
   CLEAR
   APPEND BLANK
   REPLACE Sys_unitno WITH UPPER(Sys_un_num)
   READ
   CLOSE FORMAT
   CLOSE DATABASE
   CLEAR
   ACCEPT "ENTER 'Y' TO ADD ANOTHER DRIVE TO SELECTED UNIT OR
ANY OTHER KEY TO EXIT" TO YN
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

18. Drv_Quer

```
  ************************************************************
  ***************
* Program Name:  DRV_QUER.prg
* Author       :  D. G. Dickison
* Date         :  07 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:
* Purpose      :  Edit/Query a drive
*
* Programs
* Called       :  Get_Unit
************************************************************
*************
CLEAR
DO Get_UNit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
```

```
   CLEAR
   RETURN
ENDIF
USE Drive ORDER Sys_unitno
SET FILTER TO Sys_unitno = Sys_un_num
SEEK Sys_un_num
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY "THERE ARE NO DRIVES ASSOCIATED WITH SYSTEM
UNIT "+TRIM(Sys_un_num)
   @ 12, 17 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
   WAIT " "
   CLOSE DATABASES
   RETURN
ENDIF
CLEAR
@ 8, 5 SAY "CHOOSE THE DRIVE TO QUERY/EDIT FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
@ 9, 19 SAY "WITH CURSOR KEYS AND PRESS ENTER"
DEFINE POPUP Get_Driv FROM 10, 20 PROMPT FIELD Drive_ltr
ON SELECTION POPUP Get_Driv DEACTIVATE POPUP
ACTIVATE POPUP Get_Driv
USE Drive ORDER Sys_unitno
SET FILTER TO Sys_unitno = Sys_un_num
LOCATE FOR Drive_ltr = PROMPT()
IF TRIM(Drive_Type) = "EXTERN HRDDRIVE" .OR. TRIM(Drive_Type)
= "INTERN HRDDRIVE"
   SET FORMAT TO  HrdDrive
ELSE
   SET FORMAT TO FLOPPY
ENDIF
LOCATE FOR Drive_Ltr = PROMPT()
READ
CLOSE FORMAT
CLOSE DATABASE
CLEAR
RETURN
```

19. Dr_S_Del

```
*************************************************************
***************
* Program Name:  DR_S_DEL.prg
* Author       :  D. G. Dickison
* Date         :  04 SEPT 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  04 August 1992
```

```
*
* Purpose      :  To delete a spare drive from a system unit
*
* Programs called: none
*****************************************************************
*************
CLEAR
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   USE Drive ORDER Drive_ID
   SET FILTER TO Sys_unitno = "SPARE"
   @ 8, 5 SAY "CHOOSE THE SPARE DRIVE TO DELETE "
   @ 9, 6 SAY "WITH CURSOR KEYS AND PRESS ENTER"
   DEFINE POPUP Get_Driv FROM 10, 20 PROMPT FIELD Drive_ID
   ON SELECTION POPUP Get_Driv DEACTIVATE POPUP
   ACTIVATE POPUP Get_Driv
   CLEAR
   ACCEPT "ARE YOU SURE??  ENTER 'Y' TO CONTINUE OR ANY OTHER
KEY TO ABORT" TO YN
   IF UPPER(YN) = "Y"
      SEEK  PROMPT()
      DELETE
      CLEAR
      @ 10, 5 SAY "DRIVE "+TRIM(PROMPT())+" HAS BEEN REMOVED
FROM THE DATABASE"
      @ 12, 17 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
   ELSE
      CLEAR
      @ 10, 5 SAY "PROCEDURE ABORTED,,, RETURNING TO MAIN
MENU"
      @ 12, 16 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
      CLOSE DATABASES
      CLEAR
      RETURN
   ENDIF
   YesNo = SPACE(1)
   CLEAR
   @ 10 ,5 SAY "ENTER 'Y' TO DELETE MORE SPARE DRIVES  "
   @ 12, 12 SAY "OR ANY OTHER KEY TO RETURN TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "Y"
         CLEAR
      OTHERWISE
         CLEAR
         @ 10, 5 SAY "NO FURTHER DRIVES TO BE DELETED FROM
"+TRIM(Sys_un_num)
         @ 12, 8 SAY "PRESS ANY KEY TO EXIT TO PREVIOUS MENU"
```

```
         WAIT " "
   ENDCASE
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

20. Dr_S_Ent

```
   ************************************************************
   ***************
*Program Name:  DR_S_ENT.prg
*Author       :  D. G. Dickison
*Date         :  4 SEPTEMBER 1992
*Version      :  Dbase IV Format 1.1
*Last Updated:  9 SEPT 1992
*
*Purpose      :  This program adds spare disk drives to the
database.
*
************************************************************
*************
CLEAR
YN = "Y"
DO WHILE UPPER(YN) = "Y"
   CLOSE DATABASES
   USE Drive
   CLEAR
  ** ASSIGN CORRECT FORMAT FOR APPENDING
   ACCEPT "ENTER 'H' FOR HARDDRIVE, ANY OTHER KEY FOR FLOPPY"
TO FH
   IF UPPER(FH) = "H"
   SET FORMAT TO Hrddrive
   ELSE
   SET FORMAT TO Floppy
   ENDIF
   CLEAR
   APPEND BLANK
   REPLACE Sys_unitno WITH "SPARE"
   READ
   CLOSE FORMAT
   CLOSE DATABASE
   CLEAR
   ACCEPT "ENTER 'Y' TO ADD ANOTHER DRIVE TO SELECTED UNIT OR
ANY OTHER KEY TO EXIT" TO YN
ENDDO
CLOSE DATABASES
```

```
CLEAR
RETURN
```

21. Dr_S_Que

```
*****************************************************************
***************
* Program Name:  DR_S_QUE.prg
* Author       :  D. G. Dickison
* Date         :  07 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:
* Purpose      :  Edit/Query a spare drive
*
* Programs
* Called       :  Get_Unit
*****************************************************************
*************
CLEAR
USE Drive ORDER Drive_ID
SET FILTER TO Sys_unitno = "SPARE"
@ 8, 5 SAY "CHOOSE THE SPARE DRIVE TO QUERY/EDIT "
@ 9, 7 SAY "WITH CURSOR KEYS AND PRESS ENTER"
DEFINE POPUP Get_Driv FROM 10, 20 PROMPT FIELD Drive_ID
ON SELECTION POPUP Get_Driv DEACTIVATE POPUP
ACTIVATE POPUP Get_Driv
SEEK PROMPT()
IF TRIM(Drive_Type) = "EXTERN HRDDRIVE" .OR. TRIM(Drive_Type)
= "INTERN HRDDRIVE"
   SET FORMAT TO  HrdDrive
ELSE
   SET FORMAT TO FLOPPY
ENDIF
READ
CLOSE FORMAT
CLOSE DATABASE
CLEAR
RETURN
```

22. Get_Unit

```
*****************************************************************
***************
* Program Name:  GET_UNIT.prg
```

```
* Author       :  D. G. Dickison
* Date         :  29 July 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  06 August 1992
*
* Purpose      :  Locates a SYSTEM UNIT by either node ID or
System Unit No.
*               for editing querey or deletion of accessories
or drives.
* Programs called: N_Choice
*
*
*****************************************************************
*************
CLEAR
CLOSE DATABASES
PUBLIC Sys_un_num, Node_YN
Sys_un_num = " "
*************** ALLOW QUEREY OF NODE BY EITHER NODE ID OR
SYSTEM UNIT NC
N_or_S = SPACE(1)
YN = "TRUE"
DO WHILE YN = "TRUE"
   YN = "FALSE"
   @ 10, 5 SAY "ENTER 'N' TO SEARCH BY NODE ID,,,,"
   @ 12, 5 SAY "ENTER 'S' TO SEARCH BY SYSTEM UNIT NO,";
   GET N_or_S PICTURE "X"
   READ
   DO CASE
      ************ NODE USED TO LOCATE ACCESSORY
      CASE UPPER(N_or_S) = "N"
         DO N_Choice
         IF Node_YN = "NONE"
            CLOSE DATABASES
            Sys_un_num = "NONE ASSIGNED"
            CLEAR
            RETURN
         ENDIF
         CLEAR
         CLOSE DATABASES
         NuNode = PROMPT()
         USE Node ORDER Node_ID
         SEEK NuNode
         *******CHECK  TO  ENSURE  THERE  IS  A  SYSTEM  UNIT
ASSIGNED TO THE NODE
         IF TRIM(Sys_unitno) = "NONE ASSIGNED"
            Sys_un_num = Sys_unitno
            CLEAR
            @ 10, 5 SAY "THERE IS NO SYSTEM UNIT PRESENTLY
ASSIGNED TO NODE"+Node_ID
```

```
            @ 12, 12 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS
MENU"
            WAIT " "
            CLEAR
            CLOSE DATABASES
            RETURN
         ELSE
            Sys_un_num = Sys_unitno
         ENDIF
         *******CHECK TO ENSURE THERE IS A SYSTEM UNIT IN THE
DATABASE
         CLOSE DATABASES
         CLEAR
         USE Sys_Unit ORDER Sys_unitno
         SEEK Sys_un_Num
         IF .NOT. FOUND()
            CLEAR
            Sys_un_Num = "NONE ASSIGNED"
            @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_Num)+"
CANNOT BE FOUND IN DATABASE"
            @ 12, 8 SAY "PLEASE CHECK THE SYSTEM UNIT ASSIGNED
TO NODE "+Nunode
            @ 14, 13 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS
MENU"
            WAIT " "
            CLEAR
            CLOSE DATABASES
            RETURN
         ENDIF
         CLOSE DATABASES
         RETURN
      ***************FIND SYSTEM UNIT BY SYSTEM UNIT NO
      CASE UPPER(N_or_S) = "S"
         CLEAR
         USE Sys_unit ORDER Sys_unitnc
         Sys_un_num = SPACE(15)
         @ 10, 12 SAY "ENTER THE SYSTEM UNIT NO "
         @ 12, 5 SAY "OR PRESS ENTER TO EXIT TO PREVIOUS
MENU";
         GET Sys_un_num PICTURE "XXXXXXXXXXXXXXX"
         READ
         SEEK UPPER(Sys_un_num)
         DO CASE
            ******** NOTHING ENTERED,,, EXIT TO PREVIOUS MENU
            CASE Sys_un_num = " "
               CLEAR
               CLOSE DATABASES
               Sys_un_num = "NONE ASSIGNED"
               RETURN
            ******** SYSTEM UNIT NOT FOUND IN DATABASE,,,EXIT
            CASE .NOT. FOUND()
```

```
                CLEAR
                @ 10, 5 SAY "NO SYSTEM UNIT LOCATED IN DATABASE
WITH NO. "+TRIM(Sys_un_num)
                @ 12, 15 SAY "PLEASE CHECK SYSTEM UNIT NO
AGAIN"
                @ 14, 12 SAY " PRESS ANY KEY TO RETURN TO
PREVIOUS MENU"
                WAIT " "
                Sys_un_num = "NONE ASSIGNED"
                CLOSE DATABASES
                CLEAR
                RETURN
           ENDCASE
        OTHERWISE
           CLEAR
           @ 10, 5 SAY "ILLEGAL ENTRY!,,,EITHER 'N' OR 'S'"
           @ 12, 9 SAY "PRESS ANY KEY TO TRY AGAIN"
           YN ="TRUE"
    ENDCASE
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

23. Nodent_1

```
  ***************************************************************
  ***************
*Program name :  NODENT_1.prg
*Author       :  D. G. Dickison
*Date         :  18 July 1992
*Version      :  Dbase IV Ver 1.1
*Last Updated :  21 July 1992
*
*Purpose      :  Add a new node to the LAN database while also
adding a
*                new system unit and allowing the addition of
system and
*                application software.
*
***************************************************************
*************
* RUN THE UNIT_ENT PROGRAM TO ADD THE SYSTEM UNIT TO THE
DATABASE
***************************************************************
*************
PUBLIC NuNode
```

```
CLEAR
Adding = .T.
DO UNIT_2
*****************************************************************
************
* GET THE NODE ID AND ENSURE THERE IS NOT ALREADY A NODE BY
GIVEN ID*
*****************************************************************
************
DO WHILE Adding
   USE Sys_Unit ORDER Sys_Unitno
   NuNode = SPACE(10)
   CLEAR
   @ 10, 5 SAY "ENTER THE NODE ID    "
   @ 12, 5 SAY "OR PRESS RETURN FOR PREVIOUS MENU";
   GET NuNode PICTURE "XXXXXXXXXX"
   READ
   SEEK UPPER(Sys_un_num)
   REPLACE Node_ID WITH UPPER(NuNode)
   CLOSE DATABASE
   USE Node ORDER Node_ID
   ** CHECK TO ENSURE SYSTEM UNIT NO IS NOT IN USE
   SEEK UPPER(NuNode)
   DO CASE
      **NO NODE ID ENTERED,,,,,EXIT TO PREVIOUS MENU
      CASE NuNode = " "
         Adding = .F.
         LOOP
      ** NODE ID IN USE,,,,,, EXIT TO PREVIOUS MENU
      CASE FOUND()
         CLOSE DATABASE
         USE Sys_Unit ORDER Sys_Unitno
         SEEK UPPER(Sys_un_num)
         REPLACE Sys_Unitno WITH "SPARE"
         CLOSE DATABASE
         ?  CHR(7)
         CLEAR
         @ 10, 5 SAY " NODE ID ALREADY IN USE"
         @ 12, 5 SAY " PRESS ANY KEY TO TRY AGAIN"
         WAIT " "
         ** NO NODE ID FOUND,,,,,,ENTER NODE DATA
      CASE .NOT. FOUND()
         DO Add_Node

*****************************************************************
***********
*******   ASK USER WHETHER SYSTEM SOFTWARE IS TO BE ADDED TO
NODE  *****
*****************************************************************
************
        CLEAR
```

```
        YesNo = "Y"
        DO WHILE UPPER(YesNo) = "Y"
           YesNo = SPACE(1)
           @10 ,5 SAY "ENTER 'Y' TO ENTER SYSTEM SOFTWARE"
           @12  ,5  SAY  "OR  'N'  TO  PROCEED  TO  APPLICATION
SOFTWARE ENTRY";
           GET YesNo PICTURE "X"
           READ
           DO CASE
              CASE UPPER(YesNo) = "Y"
                 DO Add_Sys
              CASE UPPER(YesNo) = "N"
                 CLEAR
                 @ 10,5 SAY " NO SYSTEM SOFTWARE CHOSEN TO BE
ENTERED"
                 WAIT
                 LOOP
              OTHERWISE
                 ? CHR(7)
                 CLEAR
                 @ 10,5 SAY "INVALID ENTRY,,,, EITHER 'Y' OR
'N'"
                 WAIT
           ENDCASE
        ENDDO
*****************************************************************
************
**************  ADD  APPLICATION  SOFTWARE  TO  THE  NODE
*********************
*****************************************************************
************
        YesNo = "Y"
        DO WHILE UPPER(YesNO) = "Y"
           YesNo = SPACE(1)
           CLEAR
           @ 10, 5 SAY "ENTER 'Y' TO ADD APPLICATION SOFTWARE
TO NODE"
           @ 12, 5 SAY "OR 'N' TO EXIT TO MENU";
           GET YesNo PICTURE "X"
           READ
           DO CASE
              CASE UPPER(YesNo) = "Y"
                 DO Add_App
              CASE UPPER(YesNo) = "N"
                 CLEAR
                 @ 10, 5 SAY " NO APPLICATION SOFTWARE CHOSEN
TO BE ENTERED"
                 @ 12, 5 SAY " PRESS ANY KEY TO RETURN TO
PREVIOUS MENU"
                 Adding = .F.
                 WAIT " "
```

```
                OTHERWISE
                   ? CHR(7)
                   CLEAR
                   @ 10, 5 SAY " INVALID ENTRY,,, ENTER 'Y' OR
'N' "
                   WAIT
             ENDCASE
       ENDDO
   ENDCASE
ENDDO
CLEAR
RETURN
```

24. Nodent_2

```
   ************************************************************
   ***************
* Program Name:  NODENT.prg
* Author       :  D. G. Dickison
* Date         :  23 July 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  4 Sept 1992
*
* Purpose      :  Add a new node to a network where the system
unit to be
*                 used already exists in the Database.
************************************************************
*************
CLEAR
   USE Sys_Unit ORDER Sys_Unitno
   * ASK FOR THE SYSTEM UNIT NO TO BE USED WITH THE NODE
   Sys_un_num = SPACE(15)
   @ 10, 5 SAY "ENTER THE SYSTEM UNIT NO. OF THE SYSTEM UNIT"
   @ 12, 5 SAY "TO BE USED WITH THE NEW NODE,,,,"
   @ 14, 5 SAY "OR PRESS ENTER TO RETURN TO PREVIOUS MENU";
   GET Sys_un_num PICTURE "XXXXXXXXXXXXXXX"
   READ
   * CHECK TO ENSURE THERE IS A SYSTEM UNIT
   SEEK UPPER(Sys_un_num)
   DO CASE
      * NO SYSTEM UNIT NO ADDED,,, EXIT TO PREVIOUS MENU
      CASE Sys_un_num = " "
         RETURN
      * SYSTEM UNIT NO NOT FOUND,,, ERROR MESSAGE AND EXIT
      CASE .NOT. FOUND()
         ? CHR(7)
         CLEAR
         @ 10, 5 SAY " NO SYSTEM UNIT FOUND IN DATABASE WITH
THE"
```

```
         @ 12, 5 SAY " GIVEN SYSTEM UNIT NUMBER,,,,, PRESS ANY
"
         @ 14, 5 SAY " KEY TO RETURN TO PREVIOUS MENU.
  "
         WAIT " "
         RETURN
      * SYSTEM UNIT NUMBER FOUND,,, TWO CASES
      CASE FOUND()
         *CHECK TO SEE IF THE SYSTEM UNIT IS ASSIGNED TO A
NODE
         CLOSE DATABASES
         USE Node ORDER Sys_unitno
         SEEK UPPER(Sys_un_num)
         DO CASE
            * SYSTEM UNIT IS PRESENTLY ASSIGNED TO A NODE
            CASE FOUND()
               CLEAR
               YesNo = "Y"
               DO WHILE UPPER(YesNo) = "Y"
                  YesNo = SPACE(1)
                  @ 10, 5 SAY "SYSTEM UNIT WITH UNIT NO.
"+TRIM(Sys_un_num)
                  @ 12, 5 SAY "IS PRESENTLY ASSIGNED TO NODE
"+Node_ID
                  @ 15, 5 SAY "ENTER 'Y' TO REMOVE SYSTEM UNIT
FROM NODE "+Node_ID
                  @ 16, 5 SAY "AND ASSIGN IT TO NEW NODE"
                  @ 18, 5 SAY "ENTER 'N' TO QUIT TO PREVIOUS
MENU";
                  GET YesNO PICTURE "X"
                  READ
                  DO CASE
                     * MISTAKE MADE,,,,,,,EXIT TO PREVIOUS
MENU
                     CASE UPPER(YesNo) = "N"
                        CLEAR
                        @ 10, 5 SAY "NO CHANGE
REQUESTED......."
                        @ 12, 5 SAY "PRESS ANY KEY TO RETURN
TO PREVIOUS MENU."
                        WAIT " "
                        RETURN
                     * USE SYSTEM UNIT FOR NEW NODE......
                     CASE UPPER(YesNo) = "Y"
                        CLEAR
                        REPLACE Sys_unitno WITH "NONE
ASSIGNED"
                        YesNo = "N"
                        CLOSE DATABASES
                     OTHERWISE
                        CLEAR
```

```
                        ? CHR(7)
                        @ 10, 5 SAY "INCORRECT KEY,,, ENTER
EITHER 'Y' OR 'N'"
                    ENDCASE
                ENDDO
            * SYSTEM UNIT NOT PRESENTLY IN USE BY A NODE
        ENDCASE
             *MAIN  PROGRAM  WHICH  ENTERS  NODE  AND  ANY
SOFTWARE
             CLEAR
             USE Sys_Unit ORDER Sys_Unitno
             NuNode = SPACE(10)
             CLEAR
             @ 10, 5 SAY "ENTER THE NEW NODE ID.";
             GET NuNode PICTURE "XXXXXXXXXX"
             READ
             SEEK Sys_un_num
             REPLACE Node_ID WITH NuNode
             CLOSE DATABASE
             USE Node ORDER Node_ID
             ** CHECK TO ENSURE THAT SYSTEM UNIT NO IS NOT
BEING USED
            SEEK UPPER(NuNode)
            DO CASE
                 **NOTHING ENTERED,, EXIT TO PREVIOUS MENU
                CASE NuNode = " "
                     CLEAR
                     CLOSE DATABASE
                     USE Sys_unit ORDER Sys_unitno
                     SEEK Sys_un_num
                     REPLACE Node_ID WITH "SPARE"
                     CLOSE DATABASE
                     @   10,   5   SAY   "   NO   NODE   ID
ENTERED,,,RETURNING TO PREVIOUS MENU"
                     WAIT
                     RETURN
                ** NODE ID ALREADY IN USE,,,,,,EXIT TO
PREVIOUS MENU
                CASE FOUND()
                   CLOSE DATABASE
                   USE Sys_unit ORDER Sys_unitno
                   SEEK Sys_unitno
                   REPLACE Node_ID WITH "SPARE"
                   CLOSE DATABASE
                   ? CHR(7)
                   CLEAR
                   @  10,  5  SAY  "NODE  ID  ALREADY  IN
USE,,,RETURNING TO PREVIOUS MENU"
                   WAIT
                   RETURN
```

```
                  *** NODE ID NOT FOUND,,, APPEND NODE AND ADD
SOFTWARE IF DESIRED
                  CASE .NOT. FOUND()
                     DO Add_Node
                     CLEAR
                     **** ADD SYSTEM SOFTWARE
                     YesNo = "Y"
                     DO WHILE UPPER(YesNo) = "Y"
                        YesNo = SPACE(1)
                        @ 10,5 SAY "ENTER 'Y' TO ENTER SYSTEM
SOFTWARE"
                        @ 12,5 SAY "OR 'N' TO PROCEED TO
APPLICATION SOFTWARE ENTRY";
                        GET YesNo PICTURE "X"
                        READ
                        DO CASE
                           CASE UPPER(YesNo) = "Y"
                              DO Add_Sys
                           CASE UPPER(YesNo) = "N"
                              CLEAR
                              @ 10, 5 SAY "NO SYSTEM SOFTWARE
CHOSEN TO BE ENTERED"
                              WAIT
                              LOOP
                           OTHERWISE
                              ? CHR(7)
                              CLEAR
                              YesNo = "Y"
                              @ 10, 5 SAY "INVALID
ENTRY,,,,EITHER 'Y' OR 'N'"
                              WAIT
                        ENDCASE
                     ENDDO
                     *** ADD APPLICATION SOFTWARE
                     YesNo = "Y"
                     DO WHILE UPPER(YesNo) = "Y"
                        YesNo = SPACE(1)
                        CLEAR
                        @ 10, 5 SAY "ENTER 'Y' TO ADD
APPLICATION SOFTWARE TO NODE"
                        @ 12, 5 SAY "OR 'N' TO EXIT TO
PREVIOUS MENU";
                        GET YesNo PICTURE "X"
                        READ
                        DO CASE
                           CASE UPPER(YesNo) = "Y"
                              DO Add_App
                           CASE UPPER(YesNo) = "N"
                              CLEAR
                              @ 10, 5 SAY "NO APPLICATION
SOFTWARE TO BE ENTERED"
```

```
                                    @ 12, 5 SAY "PRESS ANY KEY TO
RETURN TO PREVIOUS MENU"
                                    WAIT " "
                                 OTHERWISE
                                    ? CHR(7)
                                    CLEAR
                                    @ 10, 5 SAY "INVALID ENTRY,,,
ENTER 'Y' OR 'N'"
                                    YesNo = "Y"
                                    WAIT
                              ENDCASE
                           ENDDO
                     ENDCASE
ENDCASE
RETURN
```

25. Node_Ch

```
   ************************************************************
   ***************
* Program Name :  NODE_CH.prg
* Author       :  D. G. Dickison
* Date         :  12 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
*
* Purpose      :  Changes the system unit assigned to a node
* Programs called:  N_CHOICE
************************************************************
*************
CLEAR
DO N_Choice
IF Node_YN = "NONE"
  CLEAR
  CLOSE DATABASE
  RETURN
ENDIF
NuNode = PROMPT()
USE Node ORDER Node_ID
SEEK NuNode
IF Sys_unitno # "NONE ASSIGNED"
*******CHECK TO SEE IF NODE PRESENTLY HAS A SYSTEM UNIT
   CLEAR
   YesNo = SPACE(1)
   @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_unitno)+" IS PRESENTLY
ASSIGNED TO "
```

```
   @ 12, 5 SAY "NODE "+TRIM(NuNode)+" ,,, ENTER 'Y' TO REMOVE
SYSTEM UNIT"
   @ 14, 5 SAY "AND CONTINUE WITH THE PROCESS OR ANY OTHER KEY
TO ABORT";
   GET YesNo PICTURE "X"
   READ
   ****** CONTINUE WITH OPERATION
   IF UPPER(YesNo) # "Y"
      CLEAR
      @ 10, 5 SAY "OPERATION ABORTED, PRESS ANY KEY TO RETURN
TO PREVIOUS MENU"
      WAIT " "
      CLOSE DATABASE
      CLEAR
      RETURN
   ENDIF
ENDIF
Sys_unit2 = Sys_Unitno
CLEAR
******** OBTAIN THE SYSTEM UNIT NO. TO BE ADDED
Sys_un_num = SPACE(15)
@ 10, 5 SAY "ENTER THE SYSTEM UNIT NO OF THE UNIT TO BE ADDED
TO "+TRIM(NuNode);
GET Sys_un_num PICTURE "XXXXXXXXXXXXXXX"
READ
******** ENSURE SYSTEM UNIT IS IN THE DATABASE
USE Sys_Unit ORDER Sys_unitno
SEEK Sys_un_num
******** UNIT NOT FOUND,,, ABORT
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" CANNOT BE
FOUND IN DATABASE"
   @ 12, 10 SAY "PLEASE RE CHECK THE SYSTEM UNIT NUMBER
ENTERED"
   @ 14, 11 SAY "PRESS ANY KEY TO RETURN TO THE PREVIOUS MENU"
   WAIT " "
   CLOSE DATABASE
   RETURN
ENDIF
CLOSE DATABASE
******* CHECK TO SEE IF UNIT IS ASSOCIATED WITH ANOTHER NODE
CLEAR
USE Node ORDER Sys_unitno
SEEK Sys_un_num
IF FOUND()
YesNo = SPACE(1)
   CLEAR
   @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" IS PRESENTLY
ASSOCIATED WITH"
```

```
   @ 12, 5 SAY "NODE "+TRIM(Node_ID)+" ,,, ENTER 'Y' TO
CONTINUE AND REMOVE THIS"
   @ 14, 5 SAY "ASSOCIATION OR ANY OTHER KEY TO ABORT AND
RETURN TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   ****** CONTINUE WITH OPERATION
   IF UPPER(YesNo) = "Y"
      REPLACE Sys_unitno WITH "NONE ASSIGNED"
      CLOSE DATABASE
      USE Sys_unit ORDER Sys_unitno
      SEEK Sys_unit2
      REPLACE Node_ID WITH "SPARE"
      CLOSE DATABASE
   ****** ABORT OPERATION
   ELSE
      CLEAR
      @ 10, 5 SAY "OPERATION ABORTED,, PRESS ANY KEY TO RETURN
TO PREVIOUS MENU"
      WAIT " "
      CLEAR
      CLOSE DATABASE
      RETURN
   ENDIF
ENDIF
CLOSE DATABASE
******  ENTER NEW SYSTEM UNIT NO INTO NODE
USE Node ORDER Node_ID
SEEK NuNode
REPLACE Sys_unitno WITH Sys_un_num
CLOSE DATABASE
USE Sys_Unit ORDER Sys_unitno
SEEK Sys_un_num
REPLACE Node_ID WITH NuNode
CLOSE DATABASE
CLEAR
@ 10, 5 SAY "NODE "+TRIM(NuNode)+" IS NOW USING SYSTEM UNIT
"+TRIM(Sys_un_num)
@ 12, 10 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
WAIT " "
CLOSE DATABASES
CLEAR
RETURN
```

26. Node_Del

```
*************************************************************
***************
* Program Name:  Node_Del.prg
* Author       :  D. G. Dickison
* Date         :  11 August 1992
* Last Updated:
*
* Purpose      :  Delete a node from a network (including
associated
*                 Software records
*************************************************************
************
CLEAR
DO N_Choice
IF Node_YN = "NONE"
   CLEAR
   CLOSE DATABASES
   RETURN
ENDIF
NuNode = PROMPT()
CLEAR
YesNo = SPACE(1)
@ 10, 5 SAY "ARE YOU SURE YOU WANT TO DELETE NODE
"+TRIM(NuNode)+" ?????"
@ 12, 5 SAY "ENTER 'Y' TO CONTINUE OR ANY OTHER KEY TO ABORT
OPERATION";
GET YesNo PICTURE "X"
READ
DO CASE
   CASE UPPER(YesNo) = "Y"
      ******* DELETE NODE
      USE Node ORDER Node_ID
      SEEK NuNode
      DELETE
      CLOSE DATABASE
      USE Sys_unit ORDER Node_ID
      SEEK NuNode
      IF FOUND()
         REPLACE Node_ID WITH "SPARE"
      ENDIF
      CLOSE DATABASE
      ******  DELETE ASSOCIATED APPLICATION SOFTWARE
      USE Node_App ORDER Node_ID
      SEEK NuNode
      DELETE WHILE Node_ID = NuNode
      CLOSE DATABASE
      *******  DELETE ASSOCIATED SYSTEM SOFTWARE
      USE Node_Sys ORDER Node_ID
```

```
      SEEK NuNode
      DELETE WHILE Node_ID = NuNode
      CLOSE DATABASE
      CLEAR
      @ 10, 5 SAY "NODE "+TRIM(NuNode)+" HAS BEEN REMOVED FROM
THE SYSTEM"
      @ 12,10 SAY "PRESS ANY KEY TO RETURN TO THE PREVIOUS
MENU"
   OTHERWISE
      CLEAR
      @ 10, 5 SAY "OPERATION TO DELETE NODE "+TRIM(NuNode)+"
HAS BEEN ABORTED"
      @ 12, 15 SAY "PRESS ANY KEY TO RETURN TO THE PREVIOUS
MENU"
ENDCASE
WAIT " "
CLOSE DATABASES
CLEAR
RETURN
```

27. Node_Ed

```
  ************************************************************
  ***************
* Program Name : NODE_ED.prg
* Author       : D. G. Dickison
* Date         : 14 August 1992
* Version      : Dbase IV Ver 1.1
* Last Update  :
*
* Purpose      : Allows a user to edit fields of a node and
associated
*              : software
* Programs Clld: N_Choice
************************************************************
*************
CLEAR
DO N_Choice
NuNode = PROMPT()
****** CODE TO KEEP ERROR MSG FROM THIS PROGRAM OVERLAYING
N_CHOICE
IF Node_YN = "NONE"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
******** CALL UP NODE FOR EDITING
```

```
USE Node ORDER Node_ID
SEEK NuNode
SET FORMAT TO Node_Pic
READ
CLOSE FORMAT
CLOSE DATABASE
CLEAR
RETURN
```

28. Node_Que

```
*************************************************************
***************
* Program Name:  Node_Que.prg
* Author       :  D. G. Dickison
* Date         :  08 September 1992
* Version      :  Dbase IV Ver 1.1
* Last Update :
*
* Purpose      :  Query a node and display Node information
*
* Programs
* Called       :  N_Choice
*************************************************************
*************
CLEAR
CLOSE DATABASES
DO N_Choice
IF Node_YN = "NONE"
  CLOSE DATABASES
  CLEAR
  RETURN
ENDIF
*********OPEN FILES
SELECT A
USE Node ORDER Node_ID
SELECT B
USE Sys_unit ORDER Node_ID
SELECT C
USE Node_App ORDER Node_ID
SELECT D
USE Node_Sys ORDER Node_ID
SELECT A
SET DEVICE TO SCREEN
SEEK NuNode
********DISPLAY DATA
? SPACE(5)+" NODE: "+Node_ID+"  LAN:   "+Lan_Name
```

```
? SPACE(5)+"SYSTEM UNIT: "+Sys_unitno+"  PURPOSE: "+Purpose
IF TRIM(Sys_unitno) # "NONE ASSIGNED"
   SELECT B
   SEEK NuNode
   ? SPACE(5),Cpu_mod_no+"  "+Cpu_speed+" mHZ"
   ? SPACE(5),Key_Compat+" KEYBOARD          PHYSICAL MB MEM:
"+Phys_moth+" MB"
ENDIF
?     " SYSTEM SOFTWARE"
? REPLICATE("-",30)
SELECT D
SET FILTER TO Node_ID = NuNode
LIST sYS_Sw_Nam OFF
?     " APPLICATION SOFTWARE"
? REPLICATE("-",30)
SELECT C
SET FILTER TO Node_ID = NuNode
LIST Sw_Name OFF
WAIT
CLOSE DATABASES
CLEAR
RETURN
```

29. N_Choice

```
*************************************************************
***************
* Program Name :  N_CHOICE.prg
* Author       :  D. G. Dickison
* Date         :  27 July 1992
* Last Updated :  06 August 1992
*
* Purpose      :  CHOOSES A NODE BY CREATING POPUP CHOICES.
*************************************************************
*************
CLEAR
PUBLIC Node_YN, NuNode
Node_YN = " "
CLOSE DATABASES
********** CREATE A POPUP TO CHOOSE THE LAN WHICH SUBJECT
NODE IS IN
USE Lan ORDER Lan_Name
@ 8, 9 SAY "USE CURSOR KEYS TO CHOOSE THE LAN WHICH THE NODE
IS IN"
@ 9, 20 SAY "AND PRESS ENTER FOR NODE CHOICES"
DEFINE POPUP GetLan FROM 10, 20 PROMPT FIELD Lan_Name ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
```

```
ON SELECTION POPUP Getlan DEACTIVATE POPUP
ACTIVATE POPUP Getlan
CLEAR
********** CREATE A POPUP TO CHOOSE A NODE
CLOSE DATABASES
USE Node ORDER Lan_Name
SET FILTER TO Lan_Name = PROMPT()
SEEK PROMPT()
IF .NOT. FOUND()
   CLEAR
   Node_YN = "NONE"
   @ 10, 15 SAY "NO NODES ARE PRESENTLY ASSIGNED TO THE
"+TRIM(PROMPT())+" LAN"
   @ 12, 20 SAY "PRESS ANY KEY TO RETURN TO THE PREVIOUS MENU"
   WAIT " "
   RETURN
ENDIF
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE DESIRED NODE AND PRESS
ENTER"
DEFINE POPUP Get_Nod FROM 10, 20 PROMPT FIELD Node_ID
ON SELECTION POPUP Get_Nod DEACTIVATE POPUP
ACTIVATE POPUP Get_Nod
NuNode = PROMPT()
CLOSE DATABASES
CLEAR
RETURN
```

30. Spare_Qu

```
  ************************************************************
  ***************
* Program Name : Spare_Qu
* Author       : D. G. Dickison
* Date         : 4 Sept 1992
* Version      : dBase IV Ver 1.1
* Last Update  :
* Purpose      : Query a spare system unit
************************************************************
************
CLOSE DATABASES
CLEAR
USE Sys_unit ORDER Sys_unitno
SET FILTER TO TRIM(Node_ID) = "SPARE"
@ 8,5 SAY "USE CURSOR KEYS TO CHOOSE SYSTEM UNIT SPARE"
@ 9,5 SAY "     AND PRESS ENTER"
DEFINE POPUP Get_Spar FROM 10, 20 PROMPT FIELD Sys_unitno
ON SELECTION POPUP Get_Spar DEACTIVATE POPUP
```

```
ACTIVATE POPUP Get_Spar
CLEAR
Sys_un_num = PROMPT()
CLOSE DATABASE
**** OPEN FILES
SELECT A
USE Sys_unit ORDER Sys_unitno
SELECT B
USE Accessor ORDER Sys_unitno
SELECT C
USE Drive ORDER Sys_unitno
SELECT A
SET DEVICE TO SCREEN
SEEK Sys_un_num
******* Display data
? SPACE(5)+"SYSTEM UNIT NO: "+Sys_unitno+"   PHYSICAL MB MEM:
"+PHYS_MOTH+" MB"
? SPACE(5)+"                                  PHYSICAL ADDON MEM:
"+Phy_addon+" MB"
? SPACE(5)+Cpu_mod_no+"   "+Cpu_speed+"mHZ
LOGICAL EXPANDED MEM: "+Log_expmem+" MB"
? SPACE(5),Key_Compat+" KEYBOARD            LOGICAL EXTENDED
MEM: "+Log_ext_m+" MB"
? SPACE(5)+"                                          CACHE MEM:
"+Cache_mem+" KB"
***************LIST ACCESSORIES AND DRIVES***************
? "  ACCESSORY NAME      TYPE "
? REPLICATE("-",80)
SELECT B
SET FILTER TO Sys_unitno = Sys_un_num
LIST Access_nam,Access_typ OFF
? "  DRIVE LTR      DRIVE TYPE "
? REPLICATE("-",80)
SELECT C
SET FILTER TO Sys_unitno = Sys_un_num
LIST Drive_Ltr,Drive_type OFF
WAIT
CLEAR
CLOSE DATABASES
RETURN
```

31. Sys_Del

```
*****************************************************************
***************
* Program Name : SYS_DEL.prg
* Author       : D. G. Dickison
```

```
* Date          : 27 July 1992
* Last Updated : 27 July 1992
*
* Purpose       : Remove system software from a node.
* Functions/
* Programs called: N_Choice
*****************************************************************
*************
CLEAR
**************** CALL N_CHOICE TO LOCATE DESIRED NODE
DO N_Choice
CLEAR
CLOSE DATABASES
YesNo = "Y"
NuNode = PROMPT()
DO WHILE UPPER(YesNo) = "Y"
   USE Node_Sys ORDER Node_ID
   SET FILTER TO Node_ID = NuNode
   ***************CHECK FOR EMPTY SET CONDITION
   SEEK NuNode
   IF .NOT. FOUND()
      CLEAR
      @ 10, 5 SAY "NO APPLICATION SOFTWARE IS PRESENTLY
ASSGNED TO NODE "+TRIM(NuNode)
      @ 12, 13 SAY "PRESS ANY KEY TWICE TO RETURN TO THE
PREVIOUS MENU"
      WAIT " "
      RETURN
   ENDIF
   CLEAR
   USE Node_Sys ORDER Sys_SW_Nam
   SET FILTER TO Node_ID = NuNode
   @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE SOFTWARE TO BE
DELETED FROM "+TRIM(NuNode)
   @ 9, 11 SAY "AND PRESS ENTER TO DELETE SOFTWARE"
   DEFINE POPUP Get_Sys FROM 10, 20 PROMPT FIELD Sys_SW_Nam
   ON SELECTION POPUP Get_Sys DEACTIVATE POPUP
   ACTIVATE POPUP Get_Sys
   CLEAR
   ACCEPT "ARE YOU SURE?? ENTER 'Y' TO CONTINUE OR ANY OTHER
KEY TO ABORT" TO YN
   IF UPPER(YN) = "Y"
      SEEK PROMPT()
      DELETE
      CLEAR
      @ 10, 5 SAY TRIM(PROMPT())+" HAS BEEN REMOVED FROM
"+TRIM(NuNode)+" NODE"
      @ 12, 15 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
   ELSE
      CLEAR
```

```
      @ 10, 5 SAY "PROCEDURE ABORTED....RETURNING TO MAIN
MENU"
      @ 12, 10 SAY "PRESS ANY KEY TO CONTINUE"
      WAIT " "
      RETURN
   ENDIF
   YesNo = SPACE(1)
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO DELETE MORE SYSTEM SOFTWARE FROM
"+TRIM(NuNode)
   @ 12, 12 SAY "OR ANY OTHER KEY TO EXIT TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNO) = "Y"
         CLEAR
      OTHERWISE
         CLEAR
         @ 10, 5 SAY "NO FURTHER SOFTWARE CHOSEN TO BE
DELETED"
         @ 12, 5 SAY "PRESS ANY KEY TO EXIT TO PREVIOUS MENU"
         WAIT " "
   ENDCASE
ENDDO
CLOSE DATABASES
CLEAR
RETURN
```

32. Sys_Ent

```
  ************************************************************
  ***************
* Program Name:  SYS_ENT.prg
* Author       :  D. G. Dickison
* Date         :  26 July 1992
* Version      :  Dbase IV Ver 1.1
* Last Updated:  26 July 1992
*
* Purpose      :  Add System software to an existing Node
************************************************************
************
CLEAR
CLOSE DATABASES
*****   CREATE A POPUP TO CHOOSE THE LAN WHICH THE NODE IS IN
USE Lan ORDER Lan_Name
@ 8, 9 SAY "USE CURSOR KEYS TO CHOOSE LAN WHICH THE NODE IS
IN"
```

```
@ 9, 18 SAY "AND PRESS ENTER FOR NODE CHOICES"
DEFINE POPUP GetLan FROM 10, 20 PROMPT FIELD Lan_Name ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
ON SELECTION POPUP GetLan DEACTIVATE POPUP
ACTIVATE POPUP GetLan
NuLan = PROMPT()
CLEAR
************** CREATE A POPUP TO CHOOSE THE NODE THE SOFTWARE
IS TO BE ADDED TO
CLOSE DATABASES
USE Node ORDER Lan_Name
SET FILTER TO Lan_Name = PROMPT()
************** CHECK TO ENSURE THE LAN IS NOT EMPTY
SEEK PROMPT()
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY " NO NODES ARE PRESENTLY ASSIGNED TO THE
"+TRIM(PROMPT())+" LAN"
   @ 12, 13 SAY " PRESS ANY KEY TO RETURN TO THE PREVIOUS
MENU"
   WAIT " "
   RETURN
ENDIF
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE NODE WHICH THE
SOFTWARE"
@ 9, 5 SAY "IS TO BE ADDED AND PRESS ENTER FOR SOFTWARE
CHOICES"
DEFINE POPUP Get_Nod FROM 10, 20 PROMPT FIELD Node_ID ;
MESSAGE "PRESS Esc OR "+CHR(27)+" TO RETURN TO MENU"
ON SELECTION POPUP Get_Nod DEACTIVATE POPUP
ACTIVATE POPUP Get_Nod
NuNode = PROMPT()
CLEAR
CLOSE DATABASES
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   DO Add_Sys
   YesNo = SPACE(1)
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO ADD FURTHER SYSTEM SOFTWARE TO
CHOSEN NODE"
   @ 12, 23 SAY "OR 'N' TO EXIT TO MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "N"
         CLEAR
         @ 10, 5 SAY "NO FURTHER SOFTWARE CHOSEN TO BE
ENTERED"
         @ 12, 5 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS
MENU"
```

```
         WAIT " "
      CASE UPPER(YesNo) = "Y"
         CLEAR
      OTHERWISE
         CLEAR
         @ 10, 5 SAY "INVALID ENTRY,,,,,,RETURNING TO PREVIOUS
MENU"
         WAIT
   ENDCASE
ENDDO
CLEAR
RETURN
```

33. Sys_Spa

```
   ************************************************************
   ***************
* Program Name:  SYS_SPA.prg
* Author       :  D. G. Dickison
* Date         :  1 Sept 1992
* Version      :  Dbase IV Ver 1.1
* Last Update :
* Purpose      :  Allows user to change or view number of
system software
*                 spares.
************************************************************
*************
CLEAR
USE Sys_Sw ORDER Sys_SW_Nam
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE DESIRED SYSTEM
SOFTWARE"
@ 9, 20 SAY " AND PRESS ENTER"
DEFINE POPUP Get_Sys FROM 10, 20 PROMPT FIELD Sys_SW_Nam
ON SELECTION POPUP Get_Sys DEACTIVATE POPUP
ACTIVATE POPUP Get_Sys
SEEK PROMPT()
SET FORMAT TO Sys_Spar
READ
CLOSE FORMAT
CLOSE DATABASE
CLEAR
RETURN
```

34. Unit_2

```
  ************************************************************
  *****************
*Program name:  UNIT_2.prg
*Author       :  D.G. Dickison
*Date         :  17 June 1992
*Version      :  Dbase IV Format 1.1
*Last Updated:  10 August 1992
*
*Purpose      :  This program enters new system units into the
database.
*                    It also allows entry of drives, and
accessories
*                associated with the system unit.
************************************************************
**************
* Open database and index files
   PUBLIC Sys_un_num
   CLEAR
************************************************************
**********
* Check to ensure there is not already a system unit with the
same no.
************************************************************
**********
    USE Sys_unit ORDER Sys_unitno
    * Ask for the new system unit number
    Sys_un_num = SPACE(15)
    @ 10,5 SAY "Enter the new system unit number"
    @ 12,5 SAY "or press return to exit to previous menu.";
    GET Sys_un_num PICTURE "XXXXXXXXXXXXXXX"
    READ
    * Check to see if number is already in use.
    SEEK Sys_un_num
    DO CASE
        *----No System Unit No. entered....exit.
        CASE Sys_un_num = " "
        Adding = .F.
        *----System unit no already in use...beep and give
error msg.
        CASE FOUND()
            Adding = .F.
            ? CHR(7)          &&Beep
            @  20,  5  SAY  "Duplicate  System  Unit  No:
"+Sys_un_num
            @ 21, 5 SAY "Press any key return to menu"
            WAIT " "
                  *----System Unit No. not in use...Proceed
with data entry.
```

```
        CASE .NOT. FOUND()
             CLEAR
             USE Sys_unit
             SET FORMAT TO Sy_unit
             APPEND BLANK
             REPLACE Sys_unitno WITH UPPER(Sys_un_num)
             REPLACE Node_ID WITH "SPARE"
             READ
             CLEAR FORMAT
             CLOSE DATABASES
*************************************************************
************
             *----Enter drives for the system unit.
             *----Query about adding drives.
*************************************************************
************
             CLEAR
             YN1 = "Y"
             DO WHILE UPPER(YN1) = "Y"
                ACCEPT "Enter disk drive(s) at this time?
(Y/N)" TO YN1
                DO CASE
                   CASE UPPER(YN1) = "N"
                      CLEAR
                      @ 20,5 SAY "No drives chosen to be
entered"
                      WAIT
                      LOOP
                  CASE UPPER(YN1) = "Y"  && Add a drive
                      CLEAR
                      USE Drive
                      CLEAR
                      ACCEPT"Enter 'H' for Harddrive, any other
key for floppy" TO FH
                      *---Assign correct format for appending
                      IF UPPER(FH) = "H"
                         SET FORMAT TO  Hrddrive
                      ELSE
                         SET FORMAT TO Floppy
                      ENDIF
                      CLEAR
                      APPEND BLANK
                      REPLACE Sys_unitno WITH UPPER(Sys_un_num)
                      READ
                      CLEAR
                      CLOSE FORMAT
                      CLOSE DATABASES
                   OTHERWISE
                      ?  CHR(7)     && BEEP
                      CLEAR
                      YN1 = "Y"
```

```
                    @ 20,5 SAY "INVALID ENTRY.. ENTER Y OR N"
                    WAIT
                ENDCASE
            ENDDO
*****************************************************************
*********
*           ASK USER WHETHER ACCESSORIES SHOULD BE ADDED
      *
*                AND ADD THE ACCESSORIES
      *
*****************************************************************
*********
            CLEAR
            YN2 = "Y"
            DO WHILE UPPER(YN2) = "Y"
               ACCEPT "Enter accessory(ies) at this time?
(Y/N)" TO YN2
               DO CASE
                  CASE UPPER(YN2) = "N"
                     CLEAR
                     @ 20, 5 SAY " No accessories chosen to be
entered."
                     WAIT
                     LOOP
                  CASE UPPER(YN2) = "Y"  && ADD AN ACCESSORY
                     CLEAR
                     *Create popup to choose accessory
category
                     CLEAR
                     USE Acc_Name ORDER Access_Nam
                     @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE
DESIRED ACCESSORY"
                     @ 9, 13 SAY "AND PRESS RETURN"
                     DEFINE POPUP Acc_Nam FROM 10, 20 PROMPT
FIELD Access_Nam
                     ON SELECTION POPUP Acc_Nam DEACTIVATE
POPUP
                     ACTIVATE POPUP Acc_Nam
                     Nu_Acc = PROMPT()
                     CLOSE DATABASE
                     *Open file and append data/assign system
unit no.
                     USE Accessor
                     SET FORMAT TO Accssy
                     APPEND BLANK
                     REPLACE Access_Nam WITH Nu_Acc
                     REPLACE Sys_unitno WITH UPPER(Sys_un_num)
                     READ
                     CLEAR
                     CLOSE FORMAT
                     CLOSE DATABASES
```

```
                    OTHERWISE
                       ?  CHR(7)  &&BEEP
                       YN2 = "Y"
                       CLEAR
                       @ 20,5 SAY "INVALID ENTRY...PLEASE ENTER
'Y' OR 'N'"
                       WAIT
                 ENDCASE
              ENDDO
     ENDCASE
     * ASK USER WHETHER HE DESIRES TO ADD ANOTHER SYSTEM UNIT
     CLEAR
CLEAR
RETURN
```

35. Unit_Add

```
   ************************************************************
   ***************
* Program Name :  UNIT_ADD.prg
* Author       :  D. G. Dickison
* Date         :  12 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :  4 Sept 1992
*
* Purpose      :  Assigns a System unit to an existing node.
************************************************************
*************
CLEAR
Sys_un_num = SPACE(15)
@ 10, 5 SAY "ENTER THE SYSTEM UNIT NUMBER OR PRESS RETURN"
@ 12, 12 SAY "TO RETURN TO PREVIOUS MENU";
GET Sys_un_num PICTURE "XXXXXXXXXXXXXXX"
READ
DO CASE
   ****nothing entered,,, returning to previous menu
   CASE Sys_un_num = " "
      CLOSE DATABASES
      CLEAR
      RETURN
   OTHERWISE
      ******* ENSURE THAT THE SYSTEM UNIT IS IN THE DATABASE
      USE Sys_Unit ORDER Sys_Unitno
      SEEK UPPER(Sys_un_num)
      IF .NOT. FOUND()
         CLEAR
```

```
         @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" NOT
FOUND IN DATABASE"
         @ 12, 12 SAY "PLEASE DOUBLE CHECK YOUR SYSTEM UNIT
NUMBER"
         @ 14, 13 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS
MENU"
         WAIT " "
         CLOSE DATABASES
         CLEAR
         RETURN
      ENDIF
      CLOSE DATABASE
      ********* CHECK TO SEE IF THE SYSTEM UNIT IS ASSIGNED TO
A NODE
      USE Node ORDER Sys_unitno
      SEEK UPPER(Sys_un_num)
      IF FOUND()
         CLEAR
         YesNo = SPACE(1)
         @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" IS
PRESENTLY ASSIGNED TO"
         @ 12, 5 SAY "NODE "+TRIM(Node_ID)+",,, ENTER 'Y' TO
CONTINUE WITH THE "
         @ 14, 15 SAY "OPERATION OR ANY OTHER KEY TO ABORT";
         GET YesNo PICTURE "X"
         READ
         DO CASE
            CASE UPPER(YesNo) = "Y"
               REPLACE Sys_unitno WITH "NONE ASSIGNED"
               CLEAR
            OTHERWISE
               CLEAR
               @ 10, 5 SAY "PROCEDURE ABORTED,, PRESS ANY KEY
TO RETURN TO MENU"
               WAIT " "
               CLOSE DATABASES
               CLEAR
               RETURN
         ENDCASE
      ENDIF
      CLOSE DATABASE
      CLEAR
      USE Lan ORDER Lan_Name
      ******GET THE NODE ID OF THE NODE THE SYSTEM UNIT IS TO
BE ADDED TO
      @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE LAN WHICH THE
DESIRED NODE"
      @ 9, 20 SAY " IS IN AND PRESS ENTER"
      DEFINE POPUP Get_Lan FROM 10, 20 PROMPT FIELD Lan_Name
      ON SELECTION POPUP Get_Lan DEACTIVATE POPUP
      ACTIVATE POPUP Get_Lan
```

```
      CLEAR
      NuLan = PROMPT()
      CLOSE DATABASE
      USE Node ORDER Node_ID
      SET FILTER TO Lan_Name = NuLan
      @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE THE DESIRED NODE"
      @ 9, 10 SAY "AND PRESS ENTER"
      DEFINE POPUP Get_Nod FROM 10, 20 PROMPT FIELD Node_ID
      ON SELECTION POPUP Get_Nod DEACTIVATE POPUP
      ACTIVATE POPUP Get_Nod
      CLEAR
      NuNode = PROMPT()
      SEEK PROMPT()
      ******* CASE OF NODE HAVING NO SYSTEM UNIT
      IF TRIM(Sys_unitno) = "NONE ASSIGNED"
         REPLACE Sys_unitno WITH UPPER(Sys_un_num)
         CLEAR
         CLOSE DATABASE
         USE Sys_unit ORDER Sys_Unitno
         SEEK UPPER(Sys_un_num)
         REPLACE Node_ID WITH UPPER(NuNode)
         @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" HAS
BEEN ASSIGNED"
         @ 12, 5 SAY "TO NODE "+TRIM(PROMPT())+" ..PRESS ANY
KEY TO RETURN"
         WAIT " "
         CLOSE DATABASES
         CLEAR
         RETURN
      ENDIF
      ******** CASE OF NODE HAVING A SYSTEM UNIT
      CLEAR
      @ 10, 5 SAY "NODE "+TRIM(PROMPT())+" PRESENTLY HAS
SYSTEM UNIT "+TRIM(Sys_unitno)
      @ 12, 5 SAY "ASSIGNED TO IT...PRESS 'Y' TO CONTINUE OR
ANY OTHER KEY TO ABORT";
      GET YesNo PICTURE "X"
      READ
      DO CASE
         CASE UPPER(YesNo) = "Y"
            Sys_unit2 = Sys_unitno
            REPLACE Sys_unitno WITH UPPER(Sys_un_num)
            CLEAR
            CLOSE DATABASE
            USE Sys_Unit ORDER Sys_unitno
            NuNode = Prompt()
            SEEK PROMPT()
            REPLACE Node_ID WITH NuNode
            SEEK Sys_unit2
            REPLACE Node_ID WITH "SPARE"
            CLEAR
```

```
            CLOSE DATABASE
            @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" HAS
BEEN ASSIGNED"
            @ 12, 5 SAY "TO NODE "+TRIM(PROMPT())+" ...PRESS
ANY KEY TO RETURN"
         OTHERWISE
            CLEAR
            @ 10, 5 SAY " OPERATION ABORTED,,PRESS ANY KEY TO
RETURN"
      ENDCASE
ENDCASE
WAIT " "
CLOSE DATABASES
CLEAR
RETURN
```

36. Unit_Ed

```
  ************************************************************
  *************
* Program Name :  UNIT_ED.prg
* Author       :  D. G. Dickison
* Date         :  12 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
*
* Purpose      :  Allows the user to edit/change system unit
information
*
* Programs called:  Get_Unit
************************************************************
*************
CLEAR
DO Get_Unit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
*************** CALL UP SYSTEM UNIT FOR EDITING
USE Sys_Unit ORDER Sys_unitno
SET FORMAT TO Sy_Unit
SEEK Sys_un_num
READ
CLOSE FORMAT
CLOSE DATABASES
```

```
CLEAR
*************** ASK USER IF HE/SHE DESIRES TO EDIT DRIVES
YesNo = SPACE(1)
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO EDIT SYSTEM UNIT
"+TRIM(Sys_un_num)+" DRIVES"
   @ 12, 18 SAY "OR ANY OTHER KEY TO CONTINUE";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "Y"
        CLEAR
        USE Drive ORDER Sys_unitno
        SET FILTER TO Sys_unitno = Sys_un_num
        @ 8, 5 SAY "CHOOSE THE DRIVE FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
        @ 9, 12 SAY "TO BE EDITED AND PRESS ENTER "
        DEFINE POPUP Get_Driv FROM 10, 20 PROMPT FIELD
Drive_ltr
        ON SELECTION POPUP Get_Driv DEACTIVATE POPUP
        ACTIVATE POPUP Get_Driv
        CLEAR
        LOCATE FOR Drive_Ltr = PROMPT()
        IF TRIM(Drive_Type) = "EXTERN HRDDRIVE" .OR.
TRIM(Drive_Type) = "INTERN HRDDRIVE"
           SET FORMAT TO Hrddrive
        ELSE
            SET FORMAT TO Floppy
        ENDIF
        READ
        CLOSE FORMAT
        CLOSE DATABASE
      OTHERWISE
         CLEAR
   ENDCASE
ENDDO
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO EDIT SYSTEM UNIT
"+TRIM(Sys_un_num)+" ACCESSORIES"
   @ 12, 10 SAY "OR ANY OTHER KEY TO RETURN TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "Y"
         CLEAR
         USE Accessor ORDER Sys_unitno
         SET FILTER TO Sys_unitno = Sys_un_num
```

```
         @ 8, 5 SAY "CHOOSE THE ACCESSORY FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
         @ 9, 14 SAY "TO BE EDITED AND PRESS ENTER "
         DEFINE POPUP Get_Acc FROM 10, 20 PROMPT FIELD
Access_Nam
         ON SELECTION POPUP Get_Acc DEACTIVATE POPUP
         ACTIVATE POPUP Get_Acc
         CLEAR
         LOCATE FOR Access_Nam = PROMPT()
         SET FORMAT TO Accssy
         READ
         CLOSE FORMAT
         CLOSE DATABASE
         CLEAR
      OTHERWISE
         CLEAR
   ENDCASE
ENDDO
RETURN
```

37. Unit_Ent

```
  ************************************************************
  *****************
*Program name:  UNIT_ENT.prg
*Author       :  D.G. Dickison
*Date         :  17 June 1992
*Version      :  Dbase IV Format 1.1
*Last Updated:  04 SEPT 1992
*
*Purpose      :  This program enters new system units into the
database.
*                       It also allows entry of drives, and
accessories
*                 associated with the system unit.
************************************************************
**************
* Open database and index files
CLEAR
************************************************************
**********
* Check to ensure there is not already a system unit with the
same no.
************************************************************
**********
Adding = .T.
DO WHILE Adding
```

```
    USE Sys_unit ORDER Sys_unitno
    * Ask for the new system unit number
    Sys_un_num = SPACE(15)
    @ 10,5 SAY "Enter the new system unit number"
    @ 12,5 SAY "or press return to exit to previous menu.";
    GET Sys_un_num PICTURE "XXXXXXXXXXXXXXX"
    READ
    * Check to see if number is already in use.
    SEEK Sys_un_num
    DO CASE
        *----No System Unit No. entered....exit.
        CASE Sys_un_num = " "
            Adding = .F.
            LOOP
        *----System unit no already in use...beep and give
error msg.
        CASE FOUND()
            ? CHR(7)          &&Beep
            @  20,  5  SAY  "Duplicate  System  Unit  No:
"+Sys_un_num
            @ 21, 5 SAY "Press any key to try again"
            WAIT " "
        *----System Unit No. not in use...Proceed with data
entry.
        CASE .NOT. FOUND()
            CLEAR
            USE Sys_unit
            SET FORMAT TO Sy_unit
            APPEND BLANK
            REPLACE Sys_unitno WITH UPPER(Sys_un_num)
            REPLACE Node_ID WITH "SPARE"
            READ
            CLOSE FORMAT
            CLOSE DATABASES
************************************************************
************
            *----Enter drives for the system unit.
            *----Query about adding drives.
************************************************************
************
            CLEAR
            YN1 = "Y"
            DO WHILE UPPER(YN1) = "Y"
               ACCEPT "Enter disk drive(s) at this time?
(Y/N)" TO YN1
               DO CASE
                  CASE UPPER(YN1) = "N"
                     CLEAR
                     @ 20,5 SAY "No drives chosen to be
entered"
                     WAIT
```

```
                    LOOP
                 CASE UPPER(YN1) = "Y"  && Add a drive
                    CLEAR
                   USE Drive
                    CLEAR
                    ACCEPT"Enter 'H' for Harddrive, any other
key for floppy" TO FH
                    *---Assign correct format for appending
                    IF UPPER(FH) = "H"
                       SET FORMAT TO  Hrddrive
                    ELSE
                       SET FORMAT TO Floppy
                    ENDIF
                    CLEAR
                    APPEND BLANK
                   REPLACE Sys_unitno WITH UPPER(Sys_un_num)
                    READ
                    CLEAR
                    CLOSE FORMAT
                    CLOSE DATABASES
                 OTHERWISE
                    ?  CHR(7)     && BEEP
                    CLEAR
                    YN1 = "Y"
                    @ 20,5 SAY "INVALID ENTRY.. ENTER Y OR N"
                    WAIT
              ENDCASE
           ENDDO
**********************************************************
*********
*          ASK USER WHETHER ACCESSORIES SHOULD BE ADDED
      *
*               AND ADD THE ACCESSORIES
      *
**********************************************************
*********
           CLEAR
           YN2 = "Y"
           DO WHILE UPPER(YN2) = "Y"
              ACCEPT  "Enter  accessory(ies)  at  this  time?
(Y/N)" TO YN2
              DO CASE
                 CASE UPPER(YN2) = "N"
                    CLEAR
                    @ 20, 5 SAY " No accessories chosen to be
entered."
                    WAIT
                    LOOP
                 CASE UPPER(YN2) = "Y"  && ADD AN ACCESSORY
                    CLEAR
                   CLEAR
```

```
                        USE Acc_Name ORDER Access_Nam
                        @ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE
DESIRED ACCESSORY"
                        @ 9, 14 SAY "AND PRESS RETURN"
                        DEFINE POPUP Acc_Nam FROM 10, 20 PROMPT
FIELD Access_Nam
                        ON SELECTION POPUP Acc_Nam DEACTIVATE
POPUP
                        ACTIVATE POPUP Acc_Nam
                        Nu_Acc = PROMPT()
                        CLOSE DATABASE
                        *Open file and append data/assign system
unit no.
                        USE Accessor
                        SET FORMAT TO Accssy
                        APPEND BLANK
                        REPLACE Access_Nam WITH Nu_Acc
                        REPLACE Sys_unitno WITH UPPER(Sys_un_num)
                        READ
                        CLEAR
                        CLOSE FORMAT
                        CLOSE DATABASES
                    OTHERWISE
                        ?  CHR(7)  &&BEEP
                        YN2 = "Y"
                        CLEAR
                        @ 20,5 SAY "INVALID ENTRY...PLEASE ENTER
'Y' OR 'N'"
                        WAIT
                ENDCASE
            ENDDO
    ENDCASE
    * ASK USER WHETHER HE DESIRES TO ADD ANOTHER SYSTEM UNIT
    CLEAR
ENDDO
CLOSE DATABASE
CLEAR
RETURN
```

38. Unit_Qu

```
*************************************************************
***************
* Program Name : UNIT_QU
* Author       : D. G. Dickison
* Date         : 14 August 1992
* Version      : Dbase IV Ver 1.1
```

```
* Last Update  : 4 Sept 1992
*
* Pupose        : Uses Report Format to create a query of a
system unit.
*
*Programs Clld : Get_Unit, Unit_Q
************************************************************
*************
CLEAR
CLOSE DATABASES
DO Get_Unit
CLEAR
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLEAR
   CLOSE DATABASES
   RETURN
ENDIF
CLOSE DATABASES
*****OPEN FILES AND SET RELATIONS
SELECT A
USE Sys_unit ORDER Sys_unitno
SELECT B
USE Accessor ORDER Sys_unitno
SELECT C
USE Drive ORDER Sys_unitno
SELECT A
SET DEVICE TO SCREEN
SEEK Sys_un_num
********** DISPLAY DATA
? SPACE(5)+"SYSTEM UNIT NO: "+Sys_unitno+"     PHYSICAL MB
MEM:"+PHYS_MOTH+" MB"
? SPACE(5)+"                                  PHYSICAL ADDON
MEM:"+Phy_Addon+" MB"
? SPACE(5),Cpu_mod_no+"   "+Cpu_speed+"MHZ
LOGICAL EXPANDED MEM:"+Log_expmem+" MB"
? SPACE(5),Key_Compat+" KEYBOARD            LOGICAL EXTENDED
MEM:"+Log_ext_m+" MB"
? SPACE(5)+"                                          CACHE
MEM:"+Cache_mem+" KB"
********** LIST ACCESSORIES AND DRIVES
? " ACCESSORY NAME     TYPE                "

? REPLICATE("-",80)
SELECT B
SET FILTER TO Sys_unitno = Sys_un_num
LIST Access_nam,Access_typ OFF
? " DRIVE LTR          DRIVE TYPE  "
? REPLICATE("-",80)
SELECT C
SET FILTER TO Sys_unitno = Sys_un_num
LIST Drive_Ltr,Drive_type OFF
```

```
WAIT
CLOSE DATABASES
CLEAR
RETURN
```

39. Unit_Rem

```
   ************************************************************
   ***************
* Program Name :  UNIT_REM.prg
* Author       :  D. G. Dickison
* Date         :  12 August 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
*
* Purpose        :   Removes the a system unit from a node
association.
*
* Programs Called:  Get_Unit
************************************************************
************
CLEAR
DO Get_Unit
CLOSE DATABASES
IF TRIM(Sys_un_num) = "NONE ASSIGNED"
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
USE Node ORDER Sys_unitno
SEEK Sys_un_num
****************** MAKE SURE SYSTEM UNIT IS ASSIGNED TO A NODE
IF .NOT. FOUND()
   CLEAR
   @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" IS NOT
ASSIGNED TO A NODE"
   @ 12, 9 SAY " PLEASE DOUBLE CHECK THE SYSTEM UNIT NO. OR
NODE ID. "
   @ 14, 12 SAY "PRESS ANY KEY TO RETURN TO PREVIOUS MENU"
   WAIT " "
   CLOSE DATABASES
   CLEAR
   RETURN
ENDIF
CLEAR
YesNo = SPACE(1)
```

```
@ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" IS ASSIGNED TO
NODE "+TRIM(Node_ID)
@ 12, 5 SAY "ENTER 'Y' TO REMOVE NODE ASSIGNMENT OR ANY OTHER
KEY TO ABORT";
GET YesNo PICTURE "X"
READ
DO CASE
   ***** REMOVAL OF ASSIGNMENT CHOSEN
   CASE UPPER(YesNo) = "Y"
      CLEAR
      REPLACE Sys_unitno WITH "NONE ASSIGNED"
      CLOSE DATABASE
      USE Sys_unit ORDER Sys_unitno
      SEEK Sys_un_num
      REPLACE Node_ID WITH "SPARE"
      @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" HAS BEEN
REMOVED FROM"
      @ 12, 5 SAY "NODE "+TRIM(Node_ID)+" PRESS ANY KEY TO
RETURN TO MENU"
   ***** OPERATION ABORTED
   OTHERWISE
      CLEAR
      @ 10, 5 SAY "OPERATION ABORTED,,,PRESS ANY KEY TO RETURN
TO PREVIOUS MENU"
ENDCASE
WAIT " "
CLOSE DATABASES
CLEAR
RETURN
```

40. Un_Sp_Dl

```
*************************************************************
***************
* Program Name :  UN_SP_DL.prg
* Author       :  D. G. Dickison
* Date         :  5 Sept 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
* Purpose      :  Delete Spare System Unit and its associated
accessories
*                 and drives from the database
*************************************************************
*************
CLEAR
USE Sys_unit ORDER Sys_unitno
SET FILTER TO TRIM(Node_ID) = "SPARE"
```

```
@ 8, 5 SAY "USE CURSOR KEYS TO CHOOSE SYSTEM UNIT TO DELETE"
@ 9, 12 SAY "AND PRESS RETURN FOR CHOICE"
DEFINE POPUP Get_Spar FROM 10, 20 PROMPT FIELD Sys_unitno
ON SELECTION POPUP Get_Spar DEACTIVATE POPUP
ACTIVATE POPUP Get_Spar
Sys_un_num = PROMPT()
CLEAR
YesNo = SPACE(1)
@ 10, 5 SAY "ARE YOU SURE THAT YOU WANT TO DELETE SYSTEM UNIT
SPARE"
@ 12, 5 SAY TRIM(Sys_un_num)+" ????   ENTER 'Y' TO CONTINUE OR
ANY "
@ 14, 10 SAY "OTHER KEY TO ABORT THE OPERATION";
GET  YesNo PICTURE "X"
READ
DO CASE
   CASE UPPER(YesNo) = "Y"
      ****DELETE SYSTEM UNIT
      SEEK Sys_un_num
      DELETE
      CLOSE DATABASE
      **** DELETE ACCESSORIES
      USE Accessor ORDER Sys_unitno
      SEEK Sys_un_num
      DELETE WHILE Sys_unitno = Sys_un_num
      CLOSE DATABASE
      **** DELETE DRIVES
      USE Drive ORDER Sys_unitno
      SEEK Sys_un_num
      DELETE WHILE Sys_unitno = Sys_un_num
      CLOSE DATABASE
      **** DISASSOCIATE NODE IF NECESSARY
      USE Node ORDER Sys_unitno
      SEEK Sys_un_num
      IF FOUND()
         REPLACE Sys_unitno WITH "SPARE"
      ENDIF
      CLOSE DATABASE
      CLEAR
      @ 10, 5 SAY "SYSTEM UNIT "+TRIM(Sys_un_num)+" HAS BEEN
DELETED"
      @ 12, 3 SAY " PRESS ANY KEY TO RETURN TO THE PREVIOUS
MENU"
      WAIT " "
   ********** USER CHOOSES NOT TO DELETE THE SYSTEM UNIT
   OTHERWISE
      CLEAR
      @ 10, 5 SAY "PROCEEDURE ABORTED, PRESS ANY KEY TO RETURN
TO MENU"
      WAIT " "
ENDCASE
```

```
CLOSE DATABASES
CLEAR
RETURN
```

41. Un_Sp_Ed

```
  ************************************************************
  *************
* Program Name :  UN_SP_ED.prg
* Author       :  D. G. Dickison
* Date         :  4 Sept 1992
* Version      :  Dbase IV Ver 1.1
* Last Update  :
*
* Purpose      :  Allows the user to edit/change spare system
unit information
*
* Programs called:  none
************************************************************
*************
CLEAR
USE Sys_unit ORDER Sys_unitno
SET FILTER TO TRIM(Node_ID) = "SPARE"
@ 8, 5 SAY "USE CURSOR KEYS TO SELECT DESIRED SPARE SYSTEM
UNIT"
@ 9, 13 SAY "AND PRESS RETURN FOR CHOICE"
DEFINE POPUP Get_Spar FROM 10, 20 PROMPT FIELD Sys_unitno
ON SELECTION POPUP Get_Spar DEACTIVATE POPUP
ACTIVATE POPUP Get_Spar
Sys_un_num = PROMPT()
************** CALL UP SYSTEM UNIT FOR EDITING
SET FORMAT TO Sy_Unit
SEEK Sys_un_num
READ
CLOSE FORMAT
CLOSE DATABASES
CLEAR
***************  ASK USER IF HE/SHE DESIRES TO EDIT DRIVES
YesNo = SPACE(1)
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   CLEAR
   @  10,  5  SAY  "ENTER  'Y'  TO  EDIT  SYSTEM  UNIT
"+TRIM(Sys_un_num)+" DRIVES"
   @ 12, 18 SAY "OR ANY OTHER KEY TO CONTINUE";
   GET YesNo PICTURE "X"
   READ
```

```
   DO CASE
      CASE UPPER(YesNo) = "Y"
        CLEAR
        USE Drive ORDER Sys_unitno
        SET FILTER TO Sys_unitno = Sys_un_num
        @ 8, 5 SAY "CHOOSE THE DRIVE FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
        @ 9, 12 SAY "TO BE EDITED AND PRESS ENTER "
        DEFINE POPUP Get_Driv FROM 10, 20 PROMPT FIELD
Drive_ltr
        ON SELECTION POPUP Get_Driv DEACTIVATE POPUP
        ACTIVATE POPUP Get_Driv
        CLEAR
        LOCATE FOR Drive_Ltr = PROMPT()
        IF TRIM(Drive_Type) = "EXTERN HRDDRIVE" .OR.
TRIM(Drive_Type) = "INTERN HRDDRIVE"
           SET FORMAT TO Hrddrive
        ELSE
            SET FORMAT TO Floppy
        ENDIF
        READ
        CLOSE FORMAT
        CLOSE DATABASE
      OTHERWISE
         CLEAR
   ENDCASE
ENDDO
YesNo = "Y"
DO WHILE UPPER(YesNo) = "Y"
   CLEAR
   @ 10, 5 SAY "ENTER 'Y' TO EDIT SYSTEM UNIT
"+TRIM(Sys_un_num)+" ACCESSORIES"
   @ 12, 10 SAY "OR ANY OTHER KEY TO RETURN TO PREVIOUS MENU";
   GET YesNo PICTURE "X"
   READ
   DO CASE
      CASE UPPER(YesNo) = "Y"
         CLEAR
         USE Accessor ORDER Sys_unitno
         SET FILTER TO Sys_unitno = Sys_un_num
         @ 8, 5 SAY "CHOOSE THE ACCESSORY FROM SYSTEM UNIT
"+TRIM(Sys_un_num)
         @ 9, 14 SAY "TO BE EDITED AND PRESS ENTER "
         DEFINE POPUP Get_Acc FROM 10, 20 PROMPT FIELD
Access_Nam
         ON SELECTION POPUP Get_Acc DEACTIVATE POPUP
         ACTIVATE POPUP Get_Acc
         CLEAR
         LOCATE FOR Access_Nam = PROMPT()
         SET FORMAT TO Accssy
         READ
```

```
            CLOSE FORMAT
            CLOSE DATABASE
            CLEAR
         OTHERWISE
            CLEAR
      ENDCASE
ENDDO
RETURN
```

LIST OF REFERENCES

1. Brewer, Mack L., "Implementation of a Configuration Management System for a Local Area Network," Masters Thesis, Naval Postgraduate School, 1991.

2. Suriano, Douglas A., "The Design of a Local Area Network Configuration Management System for the Naval Postgraduate School Administrative Sciences Department," Masters Thesis, Naval Postgraduate School, 1989.

3. Kroenke, David M. and Dolan, Kathleen A., Database Processing: Fundamentals, Design, Implementation, Science Research Associates Inc., 1988.

4. Schneidewind, Norman F. and Sahlman, Leon R., Interview, March, 1992.

BIBLIOGRAPHY

Brewer, M. L., "Implementation of a Configuration Management System for a Local Area Network," Masters Thesis, Naval Postgraduate School, Monterey, Ca., September, 1991.

Kroenke, D. M., and Dolan, K. A., Database Processing: Fundamentals, Design, Implementation, Science Research Associates, Inc., 1988.

Schatt, S., Understanding Local Area Networks, Second Edition, Howard W. Sams and Company, 1990.

Simpson, A., dBASE IV 1.1: Programmer's Desktop Reference, Sybex, 1991.

Suriano, D. A., "The Design of a Local Area Network Configuration Management System for the Naval Postgraduate School Administrative Sciences Department," Masters Thesis, Naval Postgraduate School, Monterey, Ca., 1989.

Whitten, J. L., Bentley, L. D., and Barlow, V. M., System Analysis and Design Methods, Second Edition, Irwin, 1989.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 22314-6145

2. Library, Code 0142 — 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Professor Norman F. Schneidewind — 2
   Administrative Sciences Department
   Code As/Ss
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Myung W. Suh — 1
   Administrative Sciences Department
   Code As/Su
   Naval Postgraduate School
   Monterey, CA 93940-5000

5. Lieutenant David G. Dickison — 1
   Department Head Class 126
   Surface Warfare Officers School Command
   Newport RI 02841-5012