COMPUTERISED REPAIR AND MAINTENANCE SYSTEM (A CASE STUDY OF SENSOR REPAIR STATION OF CENTRAL BANK OF NIGERIA, ABUJA).

BY

ABEGUNDE JULIUS OLUKAYODE
PGD/MCS/098/93/94

DEPARTMENT OF MATHEMATICS/COMPUTER SCIENCE
FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA.

SEPTEMBER, 2001

COMPUTERIZED REPAIR AND MAINTENANCE SYSTEM

A CASE STUDY OF SENSOR REPAIR STATION,

CURRENCY PROCESSING OFFICE (ENGINEERING),

CENTRAL BANK OF NIGERIA, ABUJA

BY

ABEGUNDE JULIUS OLUKAYODE

PGD/MCS/098/1993/94

A PROJECT SUBMITTED TO THE DEPARTMENT OF

MATHEMATICS / COMPUTER SCIENCE,

FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGER STATE.

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD

OF POST GRADUATE DIPLOMA IN COMPUTER SCIENCE

SEPTEMBER, 2001

CERTIFICATION

I certify this project work was carried out by MR. ABEGUNDE J.O., of the Department of Mathematics/Computer Science, Federal University of Technology, Minna.

PRINCE BADAMOSI, R.O. (PROJECT SUPERVISOR)	DATE
MR L.N. EZEAKO (HEAD OF DEPARTMENT)	DATE
EXTERNAL EXAMINER	DATE

DEDICATION

I dedicate this project to Almighty God.

ACKNOWLEDGEMENT

First and foremost, I wish to express my heartfelt gratitude and appreciation to the LORD GOD ALMIGHTY, for the life, energy and wisdom He bestowed on me to enable me undertake this programme.

My gratitude also goes to my project supervisor, Mr. R.O BADAMOSI, who patiently went through this work and saw to it that it was a success.

Further appreciation goes to the Head of Department of Mathematics/Computer Science Department, MR. L. N. EZEAKO for this contributions and also all the other lecturers, Dr. Y. M. Aiyesimi, Mallam Isah Audu, Prince Badmus and DR S. A. Reju

My heartfelt gratitude and appreciation also goes to my darling wife Mrd. O.O Abegunde for her assistance and encouragement, and for coping with the demands of the home whenever I am the programme. My children Mary, Emmanuel and Deborah are wonderful for their co-operation and understanding during the course of this programme.

I also wish to thank colleagues and friends who rendered immense assistance during and after the programme, especially Mr. Peter Adaji, Mr. Bayo Falekulo, Mr. M.H. Musa, Ayo Lawrence, Kabiru for their help and co-operation.

I wish to thank Mr. Shola Alabi who typed this work, My thanks goes to Pastor Charles Nwokpara and members of Christian Pentecostal Mission, Graki Branch for their prayers and encouragement.

ABSTRACT

Any organisation that does repair, replacement or maintenance of machinery for its branch offices should have a way of keeping track of the information about the reception of such goods, the return of repaired ones and the replacement of irreparable parts.

This project work entails the way a large organisation like CBN can keep track of their spare parts stock using computers. Useful information was gathered as regards the traditional method of handling the operation. With the manual system which is obsolete, slow and error prone, there is urgent need for this project to come in.

TABLE OF CONTENT

CHAPTER ONE

1.0 INTRODUCTION

Repair and maintenance is the operation whereby pieces of equipment are kept in good condition as much as possible, and the putting in order those that may have some faults. The operation makes the factory equipment to be available in quantity and quality as and when required, with regard to the economy comparison of repair to replacement.

It is necessary to keep records of such repairs or replacement and decide how much information is to be shown on the records. All these processes require the closest scrutiny because of the large number of – maintenance, repair, and replacement – handled by the maintenance/repair department and therefore the smallest economies are worth pursuing.

Such records are expected to show particulars of receipts, issues and balances remaining in stock for each individual item held in storehouse, from day to day, because a system of records of this kind indicates at any time the number of items on hand for repair, number on stock, number repairable etc.

Mechanical sorting and recording of parts are appropriate to every business venture be reason of the volume and magnitude of the items treated and the facilities for centralisation of the work itself.

1.1 OBJECTIVES OF THE STUDY

The project focused towards the store control system of CURRENCY PROCESSING OFFICE (ENGINEERING), CENTRAL BANK OF NIGERIA, ABUJA, the branch locations, the delivery receipt, issues of delivery, accounts and control in the storage of spare parts in a way to achieve the following:

(i) The prevention of overstocking and understocking.

(ii) The prevention of the accumulation of irreparable parts.

(iii) Reduction in the capital allocated for stores.

(iv) Greater departmental control.

(v) Improvement in the standard of services of the Currency Processing Office.

(vi) The economy in floor space, labour and clerical services.

(vii) Reduction of the number of defected parts from the branch locations.

(viii) Reduction of overhead cost or expenses.

1.2 SCOPE OF STUDY

The project scope and limitation is based on CURRENCY PROCESSING OFFICE – CPO - (ENGINEERING) CENTRAL BANK OF NIGERIA, ABUJA. The project limits its research to repair and maintenance control as practised in CPO, CBN.

Furthermore, some of the principles adopted in this study would be useful to any establishment encountered with the same kind of problems in handling their respective stock control.

1.3 RESEARCH METHODOLOGY

In order to carry out the study successfully, it is of paramount importance to follow the following approaches:

i) Direct information concerning the establishment is collected. This assisted very much in thinking of the best software package to be developed. The package developed should be able to handle well their operations and generates a full-detailed reports.

ii) Gathering of more facts from the CPO workshop and collection of all documented details concerning their operations. How they collect the defected parts and distribute the repaired ones to their various branch locations.

iii) The department directly involved would be examined in order to be able to decide the kind of computer system that would handle their problems at hand and next few years to come.

iv) The determination of what kind of programming language that would be most appropriate to handle the operations effectively.

v) The staff directly involved would be educated on how to operate and make use of the computer to generate valuation reports.

vi) Before the software package being introduced their will be a test run of the newly introduced package in order to confirm its efficiency and parallel running method adopted for software introduction.

CHAPTER TWO

2.0 BACKGROUND OF STUDY

2.1 DESCRIPTION OF THE EXISTING SYSTEM

The function of the company is to:

(i) distribute spare parts and machinery to the zonal branch locations. The zonal locations are Minna, Kaduna, Kano, Maiduguri, Jos, Bauchi, Sokoto, Yola, Makurdi, Ilorin and also Abuja holden store.

(ii) collect defected parts that need sophisticated machinery for repair – e.g. sensors (0-8), electronics, printer mechanisms, and compressor – and return the repaired parts to the locations.

(iii) Taking stock of the quantity of each machinery/spare parts needed to be in a particular location, the number available there, thereby calculating the quantity to be sent for replacement.

At present, the storekeeping is in charge of a storekeeper who reports every transaction to the CPO manager. The store is basically a warehouse which keeps stock of parts and equipment. The manual system of keeping stock is very obsolete, slow and error prone.

The current system in the workshop of the CPO includes a system which allows one to keep a close check on the repair items at hand. Each item is allocated a bin number and associated with this is a card index system. This card is laid out in such a way that whenever an item is repaired in the workshop and returned to the central store the card index is updated by writing in the date and the quantity returned to store in the quantity repaired column.

2.2 COLLECTION, HANDLING AND PROCESSING OF INFORMATION

BIN CARDS

It is sometimes thought desirable to keep a record with the physical stock itself by attaching bin cards to the bins or racks containing the materials, using a separate card for each item of part. These cards are usually very simple, giving a description of the item including its vocabulary number, the unit of issue, the quantities received and issued, and the balance remaining in the bin.

COLLECTION AND HANDLING

Whenever a new set of parts arrived, each item of equipment is given a stock code and the bin card is attached to each stock being stored.

MINIMUM LEVEL

This is the amount expressed in unit of issue for each item/spare part at the branch locations below which the stock of the commodity should not be allowed to fall.

When this level is reached, it triggers off urgent action to bring forward delivery of the next order. It is known as the "re-order level" and often referred to as the danger level.

In fixing this level, the main factor considered is the effect which a run-out of equipment stock would have upon the affected branch location.

ORDERING STOCK LEVEL

This is the amount expressed in unit of issue at which ordering actions is indicated in-time for the spare parts to be delivered before stock falls below the proper usage level and re-order cycle after inspection and receipt is also considered. When re-ordering level is reached for any item i.e. when the re-order point is reached, a check is always made to see if there are outstanding deliveries in respect of any existing order before arrangement are made to buy a fresh supply.

MAXIMUM STOCK LEVEL

This is the amount expressed in unit of issue at which each spare part/item should be allowed to rise. The purpose of this level is to curb excess machinery and parts in locations. In fixing a maximum level, consideration is given to availability of finance, rate of consumption to avoid excessive stock holding, possibility of item becoming obsolete in nature and the danger of deterioration in perishable commodities.

When the level is reached, it is a signal to defer or cancel outstanding deliveries (if any).

PROCESSING OF INFORMATION

With the aid of all the information being gathered, any stock code being entered searched the database and brings out the description of item and more so, any location code being entered will also search the database to bring out the branch description.

DISADVANTAGES OF EXISTING SYSTEM

The manual system of keeping stock is very obsolete, slow and prone to errors and in a situation whereby the supplier company in Germany continually changes the specification of the parts that they produced which led to the present situation today where a very large number of different parts have to be stocked.

The present system is not efficient because the store supervisor (i.e. manager CPA) issue order for the store man to check for items requiring ordering when he updates the store cards. As he is human, he may forget or he may mislay his list of items for re-ordering. Whenever new pieces of equipment are delivered, the stock records must be updated by writing in the date and the quantity of goods put into stock. Again the quantity on hand must be updated.

CHAPTER THREE

3.0 SYSTEM DESIGN

3.1 OBJECTIVES

The system is designed to demonstrate the feasibility of developing a repair and maintenance control system. The system is designed to interact with the user, collect relevant information about the situation of the quantities in store and the repairs in the workshop, and updates the database.

3.2 INPUT DESIGN

The careful study of every input design in any system development is to make data entry as easy, logical and free from errors as possible. The user needed to know the following data entry.

(a) The space allocated to each field.

(b) Field sequence, which must correspond to that in the source document.

(c) The format in which the fields are entered.

Before the data being captured via the keyboard into the system, the user need to supply answer to the question specified. The user needs to answer the questions in concise and correct manner to ease processing.

Different table structures are used to keep record of different operations performed either in the store or workshop. For better handling of these operations a database is used for keeping the tables, the relationship between them.

The database (MAINDATA) contains the tables

a) Maintenance Table: which has all the information about the type and operations performed in the workshop of the Currency Processing Office. The field names in the table together with their field types and width are as stated below:

Field Name	Type	Width
Type1	Character	15
Operation	Character	30
Date	Date	8

b) Requirement Data Table: This contains information concerning the maximum and minimum requirement of each of the spare parts. The content of requirement table determines the amount of spare parts and/or equipment to send to a location for re-ordering. The table has the following fields.

Field Type	Type	Width
Locname	Character	20
Sensor	Numeric	3
Sensor2	Numeric	3
Electronics	Numeric	3
Electronics2	Numeric	3
Prn_mech	Numeric	3
Prn_mech2	Numeric	3
Compressor	Numeric	3
Compressor2	Numeric	3
Belts	Numeric	3
Belts2	Numeric	3
Rollers	Numeric	3
Rollers2	Numeric	3

c) Repair Data Table: The relation data table holds data of each repair. It has the tables has shown below

Field Name	Type	Width
Name	Character	30
Location	Character	10
Date_repaired	Date	8

Serial_num	Character	15

d) Shread Data Table: keeps data of shreading done, the time it was done and the reason and mode of shreading.

Field Name	Type	Width
reason	Character	11
Mode	Numeric	3
Date	Date	8

e) Defective Data Table: The fields here are:

Field Name	Type	Width
Code	Character	10
Name	Character	20
Nbrought	Numeric	3

CHAPTER FOUR

4.0 REPAIR AND MAINTENANCE SIMULATION

4.1 INTRODUCTION

Program development entails taking cognisance of all the activities contributing to program initiation. It is concerned with adequate scrutiny of the programming languages to know the particular one that is most appropriate for the problem context.

This chapter talks about the development and implementation of Repair and Maintenance software system. It explains how this can be used to produce a program that is highly operational and effective. That is implementing it and putting it into work by applying it in the required workshop and stores to know whether it serves the purpose for which it is developed. The process involves the development of quality procedures for the data security.

4.2 CHOICE OF PROGRAMMING LANGUAGE.

The choice of the appropriate programming language to use will depend on the features of the program the project is trying to solve and the nature of result expected to be generated.

Studying the problem at hand, it is clearly noted that the programming language to be adopted must possess the following qualities:

1. Adequate keeping of data records and the interrelationships between files/tables.
2. Perfect handling of enhanced graphics.
3. Access time and handle time management.
4. Setting of relationships between data and activity.
5. A user captured interface.
6. Multi-user environment.

Therefore, any programming language that can handle these would be absolute for the problem.

In this case, a record-keeping relational database will satisfy the first, third and fourth points. That is, languages of the database series such as FoxPro, Clippers and their likes. Meanwhile, to pay justice to the second, fifth and last points, there should be a need for any of these languages that operates under Windows operating system.

In view of this observation, I opt for the use of visual FoxPro that can adequately satisfy the conditions in the best possible way.

4.3 FEATURES OF CHOICE PROGRAMMING LANGUAGE

The efficiency and effectiveness of any programming language lies in the ease with which it handles problems and the flexibility of its usage. Visual FoxPro is one of the most efficient and effective programming language. As a relational database language which has to do with data keeping than computation on such data it is referred to as a transpute-bound programming language.

Visual FoxPro, due to its English-like nature and the enhanced end-user facility can be regarded as a fourth generation programming. Among the characteristics which make FoxPro outstanding, and the best choice for this software development are:

1. Efficiency in multiple data management.
2. High support of OOP.
3. Perfect handling of enhanced graphics.

1. Efficiency in multiple data management.

In VFP, the databases are used to organise and relate tables (the concepts; database and table are not synonymous in Visual FoxPro. The term database refers to a relational database that is a container of information about one or more tables)

The databases provide the architecture for storing data and have additional benefits as well. Among the benefits are the creation of table-level rules such as field- and record-level rules, default field values, and triggers. Stored procedures and persistent table relationships can be created.

Using an established VFP database design process, a well-defined database that provides convenient access to the information needed can be quickly and effectively created. With a solid design, the software developer will spend less time constructing the database and end up either faster, more accurate result.

Having one-to-many and many-to-many relationships is a common attribute to all the relational database languages but setting file pointers to the right records of the open tables as the pointer to the current file in use is moved is another important factor that makes VFP so outstanding.

2. High Support of OOP.

While VFO still supports standard procedure programming, new extensions to the language give the power and flexibility of OOP.

Object oriented design and OOP represent a change in focus from standard procedural programming. Instead of thinking about program flow the first line of code to the last line of code, one needs to think about creating objects. These objects are self-contained components of an application that have private functionality as well as functionality that can be exposed to the user.

In VFP, forms and controls are objects that can be included in applications. These objects can be manipulated through their properties, events and methods. The object oriented VFP language extensions provide you with a great deal of control over the objects in your applications. These extensions also make it easier to create and maintain libraries of reusable code, giving you:

- More compact code.
- Easier incorporation of code in applications without elaborate naming schemes.
- Less complexity when integrating code from different files into an application.

The objects available in VFP are as classified into two viz:

1. Controls – the lowest level of particular type.
2. Containers – these can contain other objects (controls) in them.

The objects can also be

1. Visual: visible at run time.
2. Non-visual: Invisible at run time.

Some of the objects are now as stated below.

Control		Container
Checkbox	Shape	Formset
Combobox	Spinner	Form
Editbox	Textbox	Grid
Command Button		Column

Header

Label

Line

OLE Bound Control

OLE Container

Control

Page Frames

Page

Toolbar

Option Button Group

Command Button

Group

Another term that is of great importance in OOP is classes. Classes and objects are closely related, but they are not the same. A class contains information about how two objects should look and behave. All of the properties, events and methods for an object are specified in the class definition. In addition, classes have the following characteristics that make them especially useful for creating reusable, easily maintainable code:

Encapsulation

Subclasses

Inheritance.

3. Perfect handling of enhanced graphics

VFP allows programmers to extend the power of their applications by employing the strengths of other OLE-enabled applications. Specific functionality or data such as text, sound, pictures and videos from other applications. You can view or manipulate this data visibly by using the application that created it. Other applications can also tap into the power visual FoxPro through automation. OLE servers can also be created for use for application to access.

4.4 IMPLEMENTATION

The whole program developed, was put in a project named REPAIR under the directory "REPAIR". The method of using one special directory for all the files i.e. databases, forms, codes, etc. is

(1) to create a direct path for VFP to check in case of the needs to generate a setup disk or an executable file.

(2) for faster compilation at runtime.

1. FORMS

Besides giving users a familiar interface for viewing and entering data into a database, forms also provide a rich set of objects that can respond to users events so as to enable them to accomplish their information management task as easily as possible.

Forms are objects with their own properties, events and methods which can be set in the designer.

Although databases are used in this program tremendously, their importance compared to forms is very minimal.

Forms and their controls is the central object which repair and maintenance system solely depends upon. They are used exclusively for the following operations

1. Data entry
2. Simulations of the movement of the client to show his present location.

Detailed explanation of the forms used for data entry are as stated below

FORM MAINTENANCE

The form MAINTENANCE is used for data input and record keeping of the record table "maintenance". It allows records of operations performed either daily, quarterly etc. to be kept and also the description of what is actually performed.

It uses Combobox for type and operation because they are previously defined and a textbox for date.

Also included on the form are two command boxes for editing and navigation.

The editing Button

This contains the command buttons Add, Revert, Delete, Save and Exit.

When the form is first loaded, revert and save buttons are invisible. The operation performed by the editor are so tailored that any command button not needed at any time is not available for use it is either invisible or disabled.
When "Add "button is clicked, it appends a blank record to the end of table maintenance and refresh the form to display the current status. This now allows data to be entered in the empty controls.
Changes made on maintenance table and the form can be undone. This is performed by the "Revert" button which is only enabled and visible when between the Add and Save process (i.e. after Add is clicked and Save is clicked).

The same process is performed immediately after enter key is pressed. Data duplicity is controlled using two distinct methods:

1. No record can be entered twice.
2. Any data input cannot be stored in the personal table twice since "Add" button is disabled whenever " Save" is enabled and vice versa.

In deleting the record display on form all you need to do is to click delete. A message box will appear to ask you about the validity of what you are doing .if you pick yes the record will be absolutely deleted from the disk.

<u>Navigator button</u>

Navigator buttons are used to display records on form maintenance. Following is the list of navigator buttons and their functions:

1. << To display record in data table
2. < To display the previous record to the record currently on screen
3. > To display the next record to the record currently on screen
4. >> To display the last record in the table

If the records displayed on the screen is the first record, then < and << are invisible, also if the last record is displayed then > and >> are invisible. The navigator buttons are such that you cannot access the beginning if file or end of file.

FORM LOCATION

The form LOCATION is used for data input and record keeping of the record table "location". It allows records of operations performed either daily, quarterly etc. to be kept and also the description of what is actually performed.

It uses textboxes to collect information about the locations and provide avenue for new locations to be input into the location table.

Also included on the form are two command boxes for editing and navigation.

<u>The editing Button</u>

This contains the command buttons Add, Revert, Delete, Save and Exit.

When the form is first loaded, revert and save buttons are invisible. The operation performed by the editor are so tailored that any command button not needed at any time is not available for use it is either invisible or disabled. When "Add "button is clicked, it appends a blank record to the end of table maintenance and refresh the form to display the current status. This now allows data to be entered in the empty controls.

Changes made on maintenance table and the form can be undone. This is performed by the "Revert" button which is only enabled and visible when between the Add and Save process (i.e. after Add is clicked and Save is clicked).

The same process is performed immediately after enter key is pressed. Data duplicity is controlled using two distinct methods:

1 No record can be entered twice.

2 Any data input cannot be stored in the personal table twice since "Add" button is disabled whenever " Save" is enabled and vice versa.

In deleting the record display on form all you need to do is to click delete. A message box will appear to ask you about the validity of what you are doing .if you pick yes the record will be absolutely deleted from the disk.

Navigator button

Navigator buttons are used to display records on form maintenance. Following is the list of navigator buttons and their functions:

1 << To display record in data table

2 < To display the previous record to the record currently on screen

3 > To display the next record to the record currently on screen

4 >> To display the last record in the table

If the records displayed on the screen is the first record, then < and << are invisible, also if the last record is displayed then > and >> are invisible. The navigator buttons are such that you cannot access the beginning if file or end of file.

FORM REPAIR

The form REPAIR is used for data input and record keeping of the record table "repair". It allows information of operations performed on a spare part, the name of the part, its serial number and the date the spare part is repaired to be kept.

It uses comboboxes for location and name of spare part since they are previously known and textboxes for date repaired and serial number.

Also included on the form are two command boxes for editing and navigation.

The editing Button

This contains the command buttons Add, Revert, Delete, Save and Exit.

When the form is first loaded, revert and save buttons are invisible. The operation performed by the editor are so tailored that any command button not needed at any time is not available for use it is either invisible or disabled. When "Add "button is clicked, it appends a blank record to the end of table maintenance and refresh the form to display the current status. This now allows data to be entered in the empty controls.

Changes made on maintenance table and the form can be undone. This is performed by the "Revert" button which is only enabled and visible when between the Add and Save process (i.e. after Add is clicked and Save is clicked).

The same process is performed immediately after enter key is pressed. Data duplicity is controlled using two distinct methods:

1 No record can be entered twice.

2 Any data input cannot be stored in the personal table twice since "Add" button is disabled whenever " Save" is enabled and vice versa.

In deleting the record display on form all you need to do is to click delete. A message box will appear to ask you about the validity of what you are doing .if you pick yes the record will be absolutely deleted from the disk.

Navigator button

Navigator buttons are used to display records on form maintenance. Following is the list of navigator buttons and their functions:

1 << To display record in data table

2 < To display the previous record to the record currently on screen

3 > To display the next record to the record currently on screen

4 >> To display the last record in the table

If the records displayed on the screen is the first record, then < and << are invisible, also if the last record is displayed then > and >> are invisible. The navigator buttons are such that you cannot access the beginning if file or end of file.

FORM SHREAD

The form SHREAD is used for data input and record keeping of the record table "shread". It makes avenue for input operations about shreading, reason for shreading and the mode used in shreading.

It uses a Combobox for "reason for shreading", a spinner is used for mode. The spinner gives opportunity for mode of shreading within the range of 30 – which is the minimum value and 90 – which is the maximum.

Also included on the form are two command boxes for editing and navigation.

The editing Button

This contains the command buttons Add, Revert, Delete, Save and Exit.

When the form is first loaded, revert and save buttons are invisible. The operation performed by the editor are so tailored that any command button not needed at any time is not available for use it is either invisible or disabled. When "Add "button is clicked, it appends a blank record to the end of table maintenance and refresh the form to display the current status. This now allows data to be entered in the empty controls.

Changes made on maintenance table and the form can be undone. This is performed by the "Revert" button which is only enabled and visible when between the Add and Save process (i.e. after Add is clicked and Save is clicked).

The same process is performed immediately after enter key is pressed. Data duplicity is controlled using two distinct methods:

1 No record can be entered twice.

2 Any data input cannot be stored in the personal table twice since "Add" button is disabled whenever " Save" is enabled and vice versa.

In deleting the record display on form all you need to do is to click delete. A message box will appear to ask you about the validity of what you are doing .if you pick yes the record will be absolutely deleted from the disk.

Navigator button

Navigator buttons are used to display records on form maintenance. Following is the list of navigator buttons and their functions:

1 << To display record in data table

2 < To display the previous record to the record currently on screen

3 > To display the next record to the record currently on screen

4 >> To display the last record in the table

If the records displayed on the screen is the first record, then < and << are invisible, also if the last record is displayed then > and >> are invisible. The navigator buttons are such that you cannot access the beginning if file or end of file.

FORM REQUIREMENT

The form REQUIREMENT is used to input the required maximum and minimum values for each spare part/equipment so that that can be used to monitor the amount of each part to send to each location in case of replacement.

It uses a textboxes for all its input.

The command group buttons for editing and navigating are also included.

<u>The editing Button</u>

This contains the command buttons Add, Revert, Delete, Save and Exit.

When the form is first loaded, revert and save buttons are invisible. The operation performed by the editor are so tailored that any command button not needed at any time is not available for use it is either invisible or disabled. When "Add "button is clicked, it appends a blank record to the end of table maintenance and refresh the form to display the current status. This now allows data to be entered in the empty controls.

Changes made on maintenance table and the form can be undone. This is performed by the "Revert" button which is only enabled and visible when between the Add and Save process (i.e. after Add is clicked and Save is clicked).

The same process is performed immediately after enter key is pressed. Data duplicity is controlled using two distinct methods:

1 No record can be entered twice.

2 Any data input cannot be stored in the personal table twice since "Add" button is disabled whenever " Save" is enabled and vice versa.

In deleting the record display on form all you need to do is to click delete. A message box will appear to ask you about the validity of what you are doing .if you pick yes the record will be absolutely deleted from the disk.

Navigator button

Navigator buttons are used to display records on form maintenance. Following is the list of navigator buttons and their functions:

1 << To display record in data table

2 < To display the previous record to the record currently on screen

3 > To display the next record to the record currently on screen

4 >> To display the last record in the table

If the records displayed on the screen is the first record, then < and << are invisible, also if the last record is displayed then > and >> are invisible. The navigator buttons are such that you cannot access the beginning if file or end of file.

4.4 SPECIFICATIONS

A. HARDWARE

Due to the nature of software used, in developing this software, it will require at least a personal computer in the following system configuration:

1. 8MB RAM
2. at least Pentium 233 processor
3. 3.5" floppy DD
4. Coloured VDU
5. A well grounded stabiliser

B. SOFTWARE REQUIREMENT

The software requirements are:

1. Windows '95 OS or a higher version
2. Visual FoxPro version 5.0 or 6.0
3. Microsoft Office for word and Excel application packages.

4.5 PROGRAM MAINTENANCE

A well-designed package, should be flexible and adaptable. It should allow changes to be made without any elaborate programmatic tasking. The opportunity for this is provided in the program by putting the codes in the necessary events of the controls used in the program. As such, the entire program allows for changes to be made even after installation.

The least hardware requirement is also used to allow conformity with the use of the program on virtually any machine.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

The relevance of computer in business enterprises cannot be over emphasised. It will help a lot in any repair and/or maintenance departmental store. It will be very ideal if all business enterprises are computerised, for only then is non-disputable summary report at required time, at no extra cost and no doubtful result possible.

Although computers are said to be very costly, but when the work it does and the efficiency of its application on business activities is compared to the cost, it would be said that the advantages outweigh the disadvantages so mentioned.

5.2 RECOMMENDATION

It is worthwhile for every organization dealing with repair and maintenance to automate its operations for proper documentation of the operation they perform every now and then. This will give the organization the opportunity to refer to the particular type of operation performed at which time and for which branch location (as in the case of CPO).

For this it is recommended a that complete automation of Currency Processing Office of the Central Bank of Nigeria for proper handling of their repair and maintenance job.

REFERENCES

- ALBERTH BATTERSHY (1982), A guide to stock control, Pitman Publishing, London.
- BOODMAN, DAVID M. & MAGRE, JOHN F. (1963) Production Planning & Inventory System, McGrawHill Inc. New York,
- CARY N. PRAGUE & JAMES E. HAMIT (1986) Dbase Programming Tips & Techniques, Ashton Tate.
- MORRISON (1982) Storage and Control of Stock for Industry and Public Undertakings, Third Edition, ELBS.
- STARR, MARTIN K. (1962) Inventory Control Theory and Practice, Prentice-Hall Inc., Eaglewood Cliffs, New Jersey.
- VISUAL FOXPRO DEVELOPER'S GUIDE (1996)Relational Database Development System for Windows, Microsoft Corporations™.

APPENDIX

PROGRAM LISTING

```
**************************************************
*-- Form:         form1 (c:\program files\devstudio\vfp\repair\location.scx)
*-- ParentClass:  form
*-- BaseClass:    form
*
DEFINE CLASS form1 AS form


        Top = 4
        Left = 73
        DoCreate = .T.
        Caption = "Form1"
        Name = "Form1"


        ADD OBJECT txtcode AS textbox WITH ;
                ControlSource = "location.code", ;
                Height = 23, ;
                Left = 254, ;
                MaxLength = 10, ;
                TabIndex = 2, ;
                Top = 69, ;
                Width = 79, ;
                Comment = "", ;
                Name = "txtCode"


        ADD OBJECT lblcode AS label WITH ;
                AutoSize = .T., ;
                BackStyle = 0, ;
                Caption = "CODE", ;
                Left = 25, ;
                Top = 207, ;
                TabIndex = 1, ;
                Name = "lblCode"


        ADD OBJECT txtname AS textbox WITH ;
                ControlSource = "location.name", ;
                Height = 23, ;
                Left = 80, ;
                MaxLength = 15, ;
                TabIndex = 4, ;
                Top = 70, ;
        Width = 113, ;
        Comment = "", ;
        Name = "txtName"


ADD OBJECT lblname AS label WITH ;
        AutoSize = .T., ;
        BackStyle = 0, ;
        Caption = "NAME", ;
        Left = 15, ;
        Top = 70, ;
        TabIndex = 3, ;
        Name = "lblName"


ADD OBJECT txtaddress AS textbox WITH ;
        ControlSource = "location.address", ;
        Height = 23, ;
        Left = 80, ;
        MaxLength = 35, ;
        TabIndex = 6, ;
        Top = 107, ;
        Width = 253, ;
        Comment = "", ;
        Name = "txtAddress"


ADD OBJECT lbladdress AS label WITH ;
        AutoSize = .T., ;
        BackStyle = 0, ;
        Caption = "ADDRESS", ;
        Left = 15, ;
        Top = 107, ;
        TabIndex = 5, ;
        Name = "lblAddress"


ADD OBJECT txtphone AS textbox WITH ;
        ControlSource = "location.phone", ;
        Height = 23, ;
        Left = 80, ;
        MaxLength = 13, ;
        TabIndex = 8, ;
        Top = 138, ;
        Width = 100, ;
        Comment = "", ;
        Name = "txtPhone"


ADD OBJECT lblphone AS label WITH ;
```

```
        AutoSize = .T., ;
        BackStyle = 0, ;
        Caption = "PHONE", ;
        Left = 15, ;
        Top = 138, ;
        TabIndex = 7, ;
        Name = "lblPhone"


ADD OBJECT cmdgrpeditor AS commandgroup WITH ;
        AutoSize = .T., ;
        ButtonCount = 5, ;
        BackStyle = 1, ;
        Value = 1, ;
        Height = 33, ;
        Left = 10, ;
        Top = 204, ;
        Width = 178, ;
        TabIndex = 18, ;
        BackColor = RGB(192,192,192), ;
        Name = "cmdgrpeditor", ;
        Command1.Top = 5, ;
        Command1.Left = 5, ;
        Command1.Height = 23, ;
        Command1.Width = 42, ;
        Command1.Caption = "\<Add", ;
        Command1.Name = "cmdadd", ;
        Command2.Top = 5, ;
        Command2.Left = 47, ;
        Command2.Height = 23, ;
        Command2.Width = 42, ;
        Command2.Caption = "\<Save", ;
        Command2.ColorScheme = 2, ;
        Command2.Name = "cmdsave", ;
        Command3.Top = 5, ;
        Command3.Left = 89, ;
        Command3.Height = 23, ;
        Command3.Width = 42, ;
        Command3.Caption = "\<Delete", ;
        Command3.Name = "cmddelete", ;
        Command4.Top = 5, ;
        Command4.Left = 131, ;
        Command4.Height = 23, ;
        Command4.Width = 42, ;
        Command4.Caption = "E\<xit", ;
        Command4.Name = "cmdexit", ;
        Command5.Top = 5, ;
        Command5.Left = 5, ;
        Command5.Height = 23, ;
        Command5.Width = 42, ;
        Command5.Caption = "\<Revert", ;
        Command5.Name = "cmdrevert"


ADD OBJECT cmdgrpnavigator AS commandgroup WITH ;
        AutoSize = .T., ;
        ButtonCount = 4, ;
        BackStyle = 1, ;
        Value = 1, ;
        Height = 33, ;
        Left = 186, ;
        Top = 204, ;
        Width = 178, ;
        TabIndex = 19, ;
        BackColor = RGB(192,192,192), ;
        Name = "cmdgrpnavigator", ;
        Command1.Top = 5, ;
        Command1.Left = 5, ;
        Command1.Height = 23, ;
        Command1.Width = 42, ;
        Command1.Caption = "<<", ;
        Command1.Name = "cmdtop", ;
        Command2.Top = 5, ;
        Command2.Left = 47, ;
        Command2.Height = 23, ;
        Command2.Width = 42, ;
        Command2.Caption = "<", ;
        Command2.Name = "cmdprevious", ;
        Command3.Top = 5, ;
        Command3.Left = 89, ;
        Command3.Height = 23, ;
        Command3.Width = 42, ;
        Command3.Caption = ">", ;
        Command3.Name = "cmdnext", ;
        Command4.Top = 5, ;
        Command4.Left = 131, ;
        Command4.Height = 23, ;
        Command4.Width = 42, ;
        Command4.Caption = ">>", ;
        Command4.Name = "cmdbottom"


ADD OBJECT label9 AS label WITH ;
        AutoSize = .T., ;
        FontBold = .F., ;
        FontItalic = .T., ;
        BackStyle = 0, ;
        Caption = "Navigator", ;
        Height = 17, ;
        Left = 189, ;
        Top = 190, ;
        Width = 55, ;
        Name = "Label9"


ADD OBJECT label10 AS label WITH ;
        AutoSize = .T., ;
        FontBold = .F., ;
        FontItalic = .T., ;
        BackStyle = 0, ;
        Caption = "Editor", ;
        Height = 17, ;
        Left = 12, ;
        Top = 190, ;
        Width = 34, ;
        Name = "Label10"


ADD OBJECT lblcode2 AS label WITH ;
        AutoSize = .T., ;
        BackStyle = 0, ;
        Caption = "CODE", ;
        Left = 216, ;
        Top = 71, ;
        TabIndex = 22, ;
        Name = "lblCode2"


ADD OBJECT label1 AS label WITH ;
        AutoSize = .T., ;
        FontSize = 18, ;
        Caption = "BRANCH LOCATION INFO", ;
        Height = 30, ;
        Left = 9, ;
        Top = 5, ;
        Width = 292, ;
        ForeColor = RGB(128,128,128), ;
        Name = "Label1"


ADD OBJECT line1 AS line WITH ;
        Height = 0, ;
        Left = 7, ;
        Top = 40, ;
        Width = 359, ;
        Name = "Line1"


ADD OBJECT shape1 AS shape WITH ;
        Top = 41, ;
        Left = 8, ;
        Height = 1, ;
        Width = 358, ;
        BorderStyle = 1, ;
        BorderColor = RGB(255,255,255), ;
        Name = "Shape1"


ADD OBJECT line2 AS line WITH ;
        Height = 0, ;
        Left = 8, ;
        Top = 185, ;
        Width = 358, ;
        Name = "Line2"


ADD OBJECT shape2 AS shape WITH ;
        Top = 184, ;
        Left = 8, ;
        Height = 1, ;
        Width = 359, ;
        BorderStyle = 1, ;
        BorderColor = RGB(255,255,255), ;
        Name = "Shape2"


PROCEDURE Activate

        thisform.cmdgrpeditor.cmdadd.visible = .T.
        thisform.cmdgrpeditor.cmdrevert.visible = .F.
        thisform.cmdgrpeditor.cmdsave.enabled = .F.
ENDPROC


PROCEDURE cmdgrpeditor.cmdadd.Click
        APPEND BLANK
        THISFORM.REFRESH()

        thisform.cmdgrpnavigator.enabled = .F.
        thisform.cmdgrpeditor.cmdadd.visible = .F.
        thisform.cmdgrpeditor.cmdrevert.visible = .T.
        thisform.cmdgrpeditor.cmdsave.visible = .T.
        thisform.cmdgrpeditor.cmdsave.enabled = .T.
ENDPROC


PROCEDURE cmdgrpeditor.cmddelete.Click
        DELETE
        PACK
        THISFORM.REFRESH()
        *thisform.txtidnum.value = ""

        if this.parent.cmdadd.visible == .F.
                this.parent.cmdadd.visible = .T.
                this.parent.cmdrevert.visible = .F.
                this.parent.cmdsave.enabled = .F.
        endif
ENDPROC


PROCEDURE cmdgrpeditor.cmdexit.Click
        THISFORM.RELEASE()
ENDPROC


PROCEDURE cmdgrpeditor.cmdrevert.Click

        GOTO BOTTOM
        DELETE
        PACK

        THISFORM.REFRESH()
        thisform.cmdgrpeditor.cmdadd.visible = .T.
        thisform.cmdgrpeditor.cmdrevert.visible = .F.
        thisform.cmdgrpeditor.cmdsave.enabled = .F.
ENDPROC


PROCEDURE cmdgrpnavigator.cmdtop.Click
        GOTO TOP
        THISFORM.REFRESH()


        this.parent.cmdprevious.enabled = .F.
        this.enabled = .F.

        this.parent.cmdnext.enabled = .T.
        this.parent.cmdbottom.enabled = .T.
ENDPROC


PROCEDURE cmdgrpnavigator.cmdprevious.Click
        IF !BOF()
                SKIP -1
                this.parent.cmdbottom.enabled = .T.
                this.parent.cmdnext.enabled = .T.
                IF BOF()
                        GOTO TOP
                        this.parent.cmdtop.enabled = .F.
```

```
                    this.enabled = .F.

                    this.parent.cmdnext.enabled = .T.
                    this.parent.cmdbottom.enabled = .T.
                else

                    this.parent.cmdtop.enabled = .T.
                    this.enabled = .T.
                ENDIF
        ENDIF
        THISFORM.REFRESH()
ENDPROC


PROCEDURE cmdgrpnavigator.cmdnext.Click
        IF !EOF()
                SKIP
                this.parent.cmdtop.enabled = .T.
                this.parent.cmdprevious.enabled = .T.
                IF EOF()
                        GOTO BOTTOM
                        this.parent.cmdbottom.enabled = .F.
                        this.enabled = .F.

                        this.parent.cmdprevious.enabled = .T.
                        this.parent.cmdtop.enabled = .T.
                ENDIF
        ENDIF
        THISFORM.REFRESH()
ENDPROC


PROCEDURE cmdgrpnavigator.cmdnext.MouseDown
        LPARAMETERS nButton, nShift, nXCoord, nYCoord
ENDPROC


PROCEDURE cmdgrpnavigator.cmdbottom.Click
        GOTO BOTTOM
        THISFORM.REFRESH()


        this.parent.cmdnext.enabled = .F.
        this.enabled = .F.

        this.parent.cmdtop.enabled = .T.
        this.parent.cmdprevious.enabled = .T.
ENDPROC
```

```
ENDDEFINE
*
*-- EndDefine: form1
'**************************************************************
```

MAINTENANCE

```
**************************************************
*-- Form:         form1 (c:\program files\devstudio\vfp\repair\maintenance.scx)
*-- ParentClass:  form
*-- BaseClass:    form
*
DEFINE CLASS form1 AS form


        Top = 12
        Left = 66
        Height = 250
        Width = 373
        DoCreate = .T.
        Caption = "Form1"
        Name = "Form1"


        ADD OBJECT cbotype AS combobox WITH ;
                RowSourceType = 1, ;
                RowSource = "DAILY,QUARTERLY,OVERHAULING,PERIODIC", ;
                ControlSource = "maintenance.type1", ;
                Height = 24, ;
                Left = 111, ;
                TabIndex = 2, ;
                Top = 68, ;
                Width = 147, ;
                Comment = "", ;
                Name = "cboType"


        ADD OBJECT lbltype AS label WITH ;
                AutoSize = .T., ;
                WordWrap = .T., ;
                BackStyle = 0, ;
                Caption = "TYPE", ;
                Left = 25, ;
        Top = 69, ;
        Width = 32, ;
        TabIndex = 1, ;
        Name = "lblType"


    ADD OBJECT cbooperation AS combobox WITH ;
        RowSourceType = 5, ;
        ControlSource = "maintenance.operation", ;
        Height = 24, ;
        Left = 110, ;
        TabIndex = 4, ;
        Top = 93, ;
        Width = 147, ;
        Comment = "", ;
        Name = "cboOperation"


    ADD OBJECT lbloperation AS label WITH ;
        AutoSize = .T., ;
        WordWrap = .T., ;
        BackStyle = 0, ;
        Caption = "OPERATION", ;
        Left = 25, ;
        Top = 97, ;
        Width = 71, ;
        TabIndex = 3, ;
        Name = "lblOperation"


    ADD OBJECT txtdate AS textbox WITH ;
        ControlSource = "maintenance.date", ;
        Height = 23, ;
        Left = 110, ;
        TabIndex = 6, ;
        Top = 118, ;
        Width = 71, ;
        Comment = "", ;
        Name = "txtDate"


    ADD OBJECT lbldate AS label WITH ;
        AutoSize = .T., ;
        WordWrap = .T., ;
        BackStyle = 0, ;
        Caption = "DATE", ;
        Left = 25, ;
        Top = 119, ;
        Width = 33, ;
        TabIndex = 5, ;
        Name = "lblDate"


ADD OBJECT lblcode AS label WITH ;
        AutoSize = .T., ;
        BackStyle = 0, ;
        Caption = "CODE", ;
        Left = 27, ;
        Top = 195, ;
        TabIndex = 1, ;
        Name = "lblCode"


ADD OBJECT cmdgrpeditor AS commandgroup WITH ;
        AutoSize = .T., ;
        ButtonCount = 5, ;
        BackStyle = 1, ;
        Value = 1, ;
        Height = 33, ;
        Left = 12, ;
        Top = 192, ;
        Width = 178, ;
        TabIndex = 18, ;
        BackColor = RGB(192,192,192), ;
        Name = "cmdgrpeditor", ;
        Command1.Top = 5, ;
        Command1.Left = 5, ;
        Command1.Height = 23, ;
        Command1.Width = 42, ;
        Command1.Caption = "\<Add", ;
        Command1.Name = "cmdadd", ;
        Command2.Top = 5, ;
        Command2.Left = 47, ;
        Command2.Height = 23, ;
        Command2.Width = 42, ;
        Command2.Caption = "\<Save", ;
        Command2.ColorScheme = 2, ;
        Command2.Name = "cmdsave", ;
        Command3.Top = 5, ;
        Command3.Left = 89, ;
        Command3.Height = 23, ;
        Command3.Width = 42, ;
        Command3.Caption = "\<Delete", ;
        Command3.Name = "cmddelete", ;
        Command4.Top = 5, ;
        Command4.Left = 131, ;
        Command4.Height = 23, ;
        Command4.Width = 42, ;
        Command4.Caption = "E\<xit", ;
        Command4.Name = "cmdexit", ;
```

```
		Command5.Top = 5, ;
		Command5.Left = 5, ;
		Command5.Height = 23, ;
		Command5.Width = 42, ;
		Command5.Caption = "\<Revert", ;
		Command5.Name = "cmdrevert"


	ADD OBJECT cmdgrpnavigator AS commandgroup WITH ;
		AutoSize = .T., ;
		ButtonCount = 4, ;
		BackStyle = 1, ;
		Value = 1, ;
		Height = 33, ;
		Left = 188, ;
		Top = 192, ;
		Width = 178, ;
		TabIndex = 19, ;
		BackColor = RGB(192,192,192), ;
		Name = "cmdgrpnavigator", ;
		Command1.Top = 5, ;
		Command1.Left = 5, ;
		Command1.Height = 23, ;
		Command1.Width = 42, ;
		Command1.Caption = "<<", ;
		Command1.Name = "cmdtop", ;
		Command2.Top = 5, ;
		Command2.Left = 47, ;
		Command2.Height = 23, ;
		Command2.Width = 42, ;
		Command2.Caption = "<", ;
		Command2.Name = "cmdprevious", ;
		Command3.Top = 5, ;
		Command3.Left = 89, ;
		Command3.Height = 23, ;
		Command3.Width = 42, ;
		Command3.Caption = ">", ;
		Command3.Name = "cmdnext", ;
		Command4.Top = 5, ;
		Command4.Left = 131, ;
		Command4.Height = 23, ;
		Command4.Width = 42, ;
		Command4.Caption = ">>", ;
		Command4.Name = "cmdbottom"


	ADD OBJECT label1 AS label WITH ;
		AutoSize = .T., ;
		FontSize = 20, ;
		Caption = "MAINTENANCE", ;
```

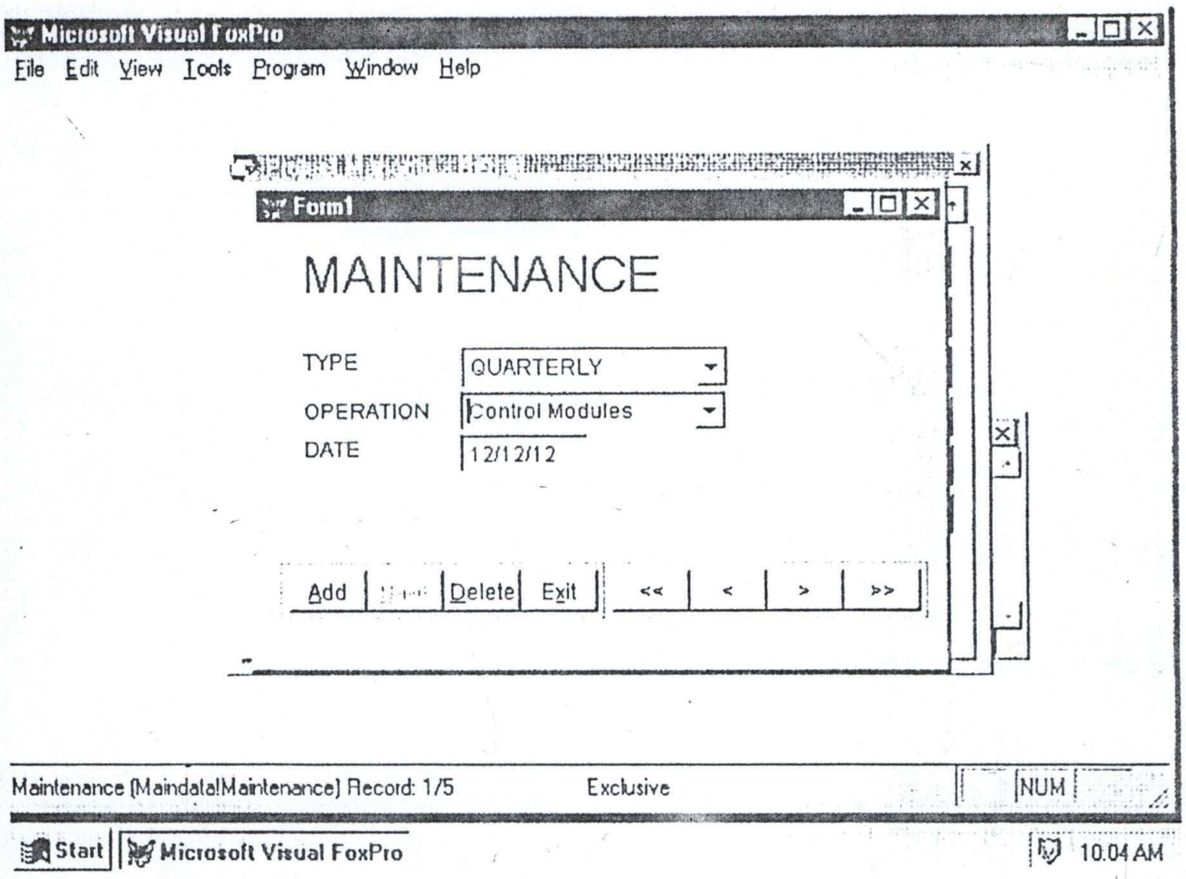
```
            Height = 35, ;
            Left = 24, ;
            Top = 12, ;
            Width = 198, ;
            Name = "Label1"


      PROCEDURE Activate
            DIMENSION
DAILYARR(5),QUARTERARR(5),OVERHAULARR(5),PERIODIC(5)


            thisform.cmdgrpeditor.cmdadd.visible = .T.
            thisform.cmdgrpeditor.cmdrevert.visible = .F.
            thisform.cmdgrpeditor.cmdsave.enabled = .F.



            quarterarr = " "
            dailyarr = " "
            overhaul = " "
            periodic = " "

            dailyarr(1) = "Belt Adjustment"
            dailyarr(2) = "Tightening of Loose Screw"
            dailyarr(3) = "Cleaning of machine"
            dailyarr(4) = "Filter Removal"

            quarterarr(1) = "Check Modules"
            quarterarr(2) = "Control Modules"
            quarterarr(3) = "Singulator Modules"
            quarterarr(4) = "Stacker Modules"

            overhaularr(1) = "Check all Modules"
            overhaularr(2) = "Replace Belts"
            overhaularr(3) = "Replace Worn Out Rollers"
            overhaularr(4) = "Sensor Callibration"
            overhaularr(5) = "Adjustment of Printer Mechanism"
      ENDPROC


      PROCEDURE cbotype.InteractiveChange
            thisval = ALLTRIM(this.value)


                  if thisval = "DAILY"
                        thisform.cbooperation.rowsourcetype = 5
                        thisform.cbooperation.rowsource = "dailyarr"
                  else
                  if thisval = "QUARTERLY"
```

```
                thisform.cbooperation.rowsourcetype = 5
                thisform.cbooperation.rowsource = "quarterarr"
            else
            if thisval = "OVERHAUL"
                thisform.cbooperation.rowsourcetype = 5
                thisform.cbooperation.rowsource = "overhaularr"
            else
            if thisval = "PERIODIC"
                thisform.cbooperation.rowsourcetype = 1
            endif
            endif
            endif
            endif
ENDPROC


PROCEDURE cmdgrpeditor.cmdadd.Click
    APPEND BLANK
    THISFORM.REFRESH()

    thisform.cmdgrpnavigator.enabled = .F.
    thisform.cmdgrpeditor.cmdadd.visible = .F.
    thisform.cmdgrpeditor.cmdrevert.visible = .T.
    thisform.cmdgrpeditor.cmdsave.visible = .T.
    thisform.cmdgrpeditor.cmdsave.enabled = .T.
ENDPROC


PROCEDURE cmdgrpeditor.cmdsave.Click
    REPLACE TYPE1 WITH THISFORm.cbotype1.value
    REPLACE operation WITH THISFORm.cbooperation.value
    REPLACE date WITH THISFORm.txtdate.value
ENDPROC


PROCEDURE cmdgrpeditor.cmddelete.Click
    DELETE
    PACK
    THISFORM.REFRESH()


    if this.parent.cmdadd.visible == .F.
        this.parent.cmdadd.visible = .T.
        this.parent.cmdrevert.visible = .F.
        this.parent.cmdsave.enabled = .F.
    endif
ENDPROC


PROCEDURE cmdgrpeditor.cmdexit.Click
```

```
        THISFORM.RELEASE()
ENDPROC


PROCEDURE cmdgrpeditor.cmdrevert.Click

        GOTO BOTTOM
        DELETE
        PACK

        THISFORM.REFRESH()
        thisform.cmdgrpeditor.cmdadd.visible = .T.
        thisform.cmdgrpeditor.cmdrevert.visible = .F.
        thisform.cmdgrpeditor.cmdsave.enabled = .F.
ENDPROC


PROCEDURE cmdgrpnavigator.cmdtop.Click
        GOTO TOP
        THISFORM.REFRESH()


        this.parent.cmdprevious.enabled = .F.
        this.enabled = .F.

        this.parent.cmdnext.enabled = .T.
        this.parent.cmdbottom.enabled = .T.
ENDPROC


PROCEDURE cmdgrpnavigator.cmdprevious.Click
        IF !BOF()
                SKIP -1
                this.parent.cmdbottom.enabled = .T.
                this.parent.cmdnext.enabled = .T.
                IF BOF()
                        GOTO TOP
                        this.parent.cmdtop.enabled = .F.
                        this.enabled = .F.

                        this.parent.cmdnext.enabled = .T.
                        this.parent.cmdbottom.enabled = .T.
                else

                        this.parent.cmdtop.enabled = .T.
                        this.enabled = .T.
                ENDIF
        ENDIF
        THISFORM.REFRESH()
ENDPROC
```

```
	PROCEDURE cmdgrpnavigator.cmdnext.Click
		IF !EOF()
			SKIP
			this.parent.cmdtop.enabled = .T.
			this.parent.cmdprevious.enabled = .T.
			IF EOF()
				GOTO BOTTOM
				this.parent.cmdbottom.enabled = .F.
				this.enabled = .F.

				this.parent.cmdprevious.enabled = .T.
				this.parent.cmdtop.enabled = .T.
			ENDIF
		ENDIF
		THISFORM.REFRESH()
	ENDPROC


	PROCEDURE cmdgrpnavigator.cmdnext.MouseDown
		LPARAMETERS nButton, nShift, nXCoord, nYCoord
	ENDPROC


	PROCEDURE cmdgrpnavigator.cmdbottom.Click
		GOTO BOTTOM
		THISFORM.REFRESH()


		this.parent.cmdnext.enabled = .F.
		this.enabled = .F.

		this.parent.cmdtop.enabled = .T.
		this.parent.cmdprevious.enabled = .T.
	ENDPROC

ENDDEFINE
*
*-- EndDefine: form1
**************************************************
```

PROGRAM OUTPUT SCREEN



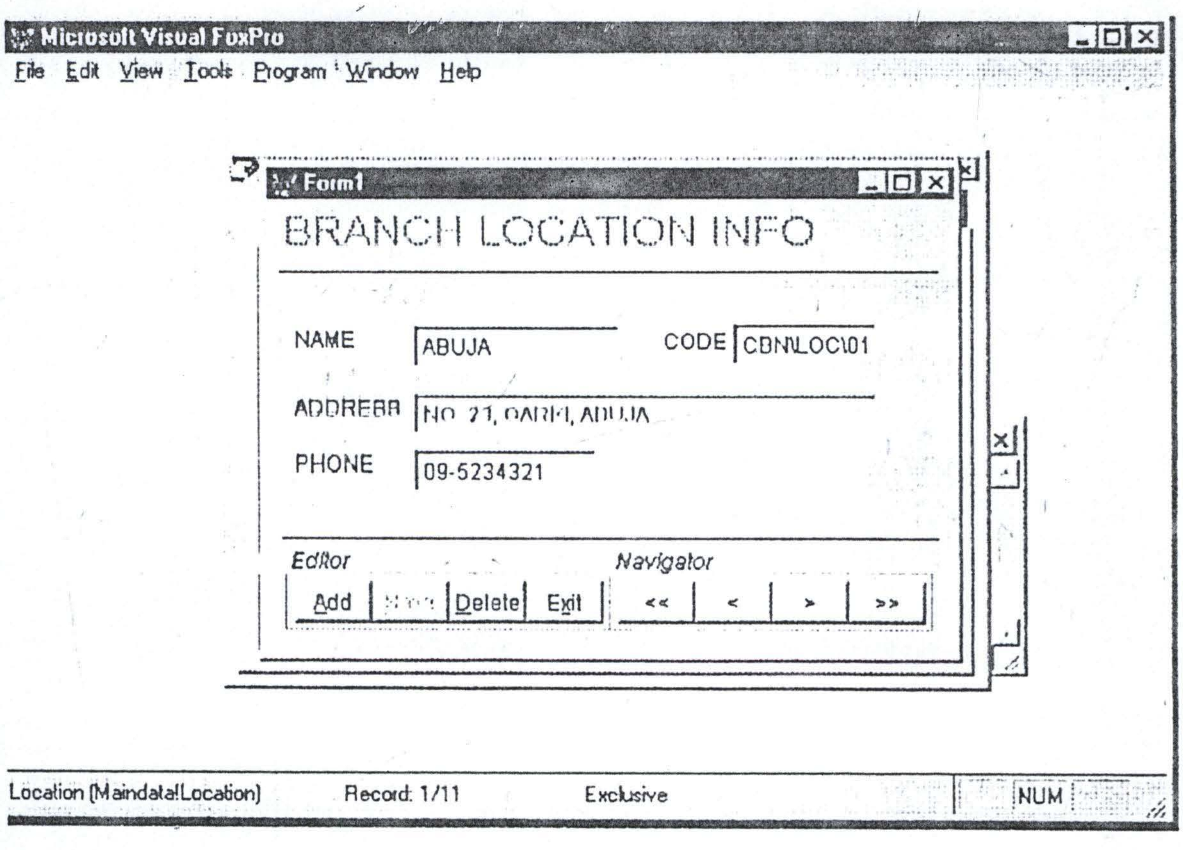
MAINTENANCE

TYPE QUARTERLY

OPERATION Control Modules

DATE 12/12/12

Maintenance (Maindata!Maintenance) Record: 1/5 Exclusive



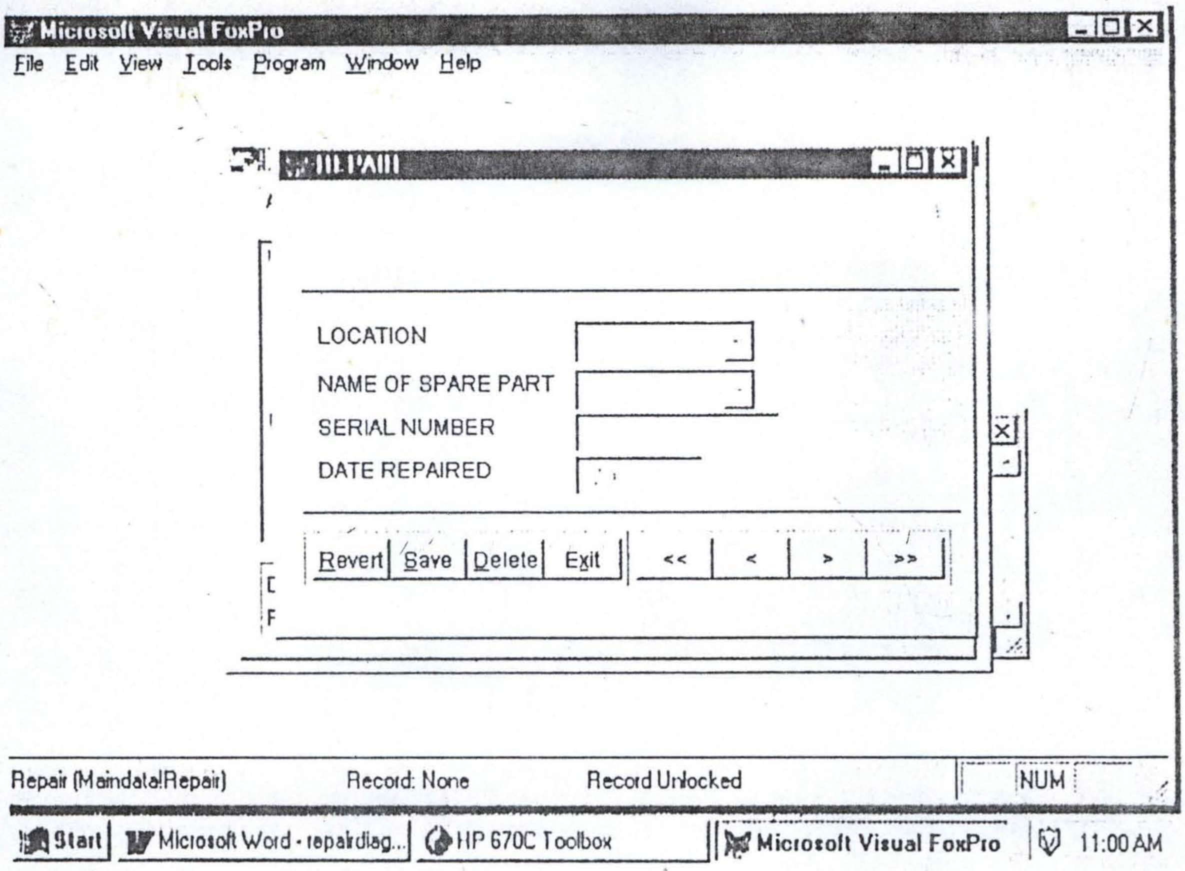
BRANCH LOCATION INFO

NAME ABUJA CODE CBN\LOC\01

ADDRESS NO. 21, GARKI, ABUJA

PHONE 09-5234321

Location (Maindata!Location) Record: 1/11 Exclusive



Microsoft Visual FoxPro

File Edit View Tools Program Window Help

REPAIR

LOCATION

NAME OF SPARE PART

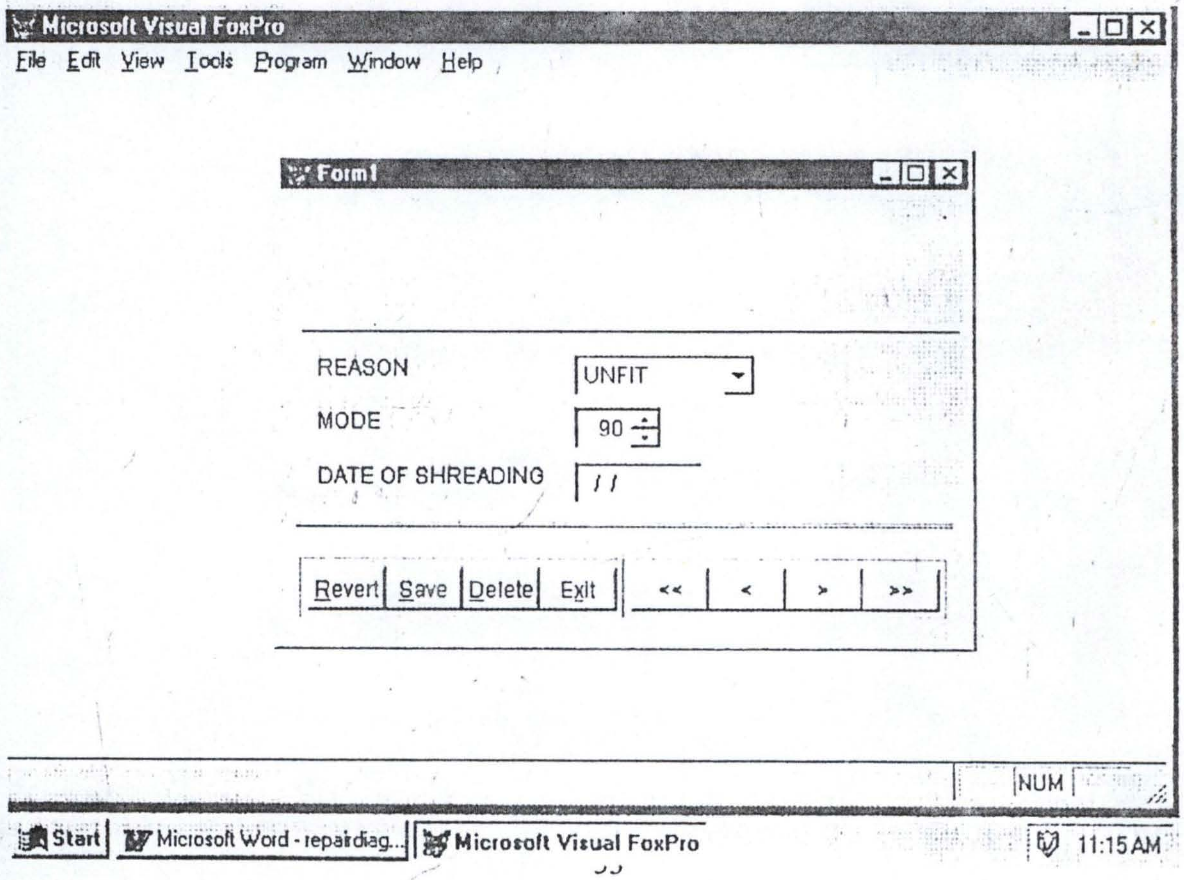
SERIAL NUMBER

DATE REPAIRED

Revert Save Delete Exit << < > >>

Repair (Maindata!Repair) Record: None Record Unlocked NUM

Start | Microsoft Word - repairdiag... | HP 670C Toolbox | Microsoft Visual FoxPro | 11:00 AM



Microsoft Visual FoxPro

File Edit View Tools Program Window Help

Form1

REASON UNFIT

MODE 90

DATE OF SHREADING / /

Revert Save Delete Exit << < > >>

NUM

Start | Microsoft Word - repairdiag... | Microsoft Visual FoxPro | 11:15 AM