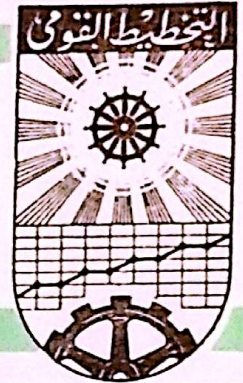


# ARAB REPUBLIC OF EGYPT

THE INSTITUTE OF  
NATIONAL PLANNING

م ت د



Memo. No. 1498

AN INFORMATION SYSTEM FOR  
INVENTORY MANAGEMENT

(A) Computer-Oriented

by

Dr. Abdalla El-Daoushy  
Dr. M. Yehia Youssef

April, 1989

CAIRO  
SALAH SALEM St-NASR CTTY



## PREFACE

-----

The proposed system will be designed to allow a keep track of all goods in stock, goods on order, and the location of the goods in the store. When the store decides to receive a new item, the item will be assigned a unique part-number and added to the master dataBase file.

The system will provide the user with reports of all goods in stock, items that need to be reordered, and items on order, and it will automatically produce purchase orders.

The system will also allow the user to keep track of all individual sales transactions: item sold, to whom sold, salesperson, date of sale, and invoice number. The user will be able to keep track of all incoming goods as stock is filled up again.

The individual transaction records from sales and goods received will be used to update the stock status records in the master dataBase file automatically.

The proposed system generally requires the use of several data files. The most important data files are the master file (MASTER.DBF) and two transaction files. One of these transaction files is called sales transaction file (SALES.DBF), and the second is called the new stock transaction file (NEWSTOCK.DBF).

The Inventory Information System developed in this memo is fairly complex from a programming point of view, but very powerful and easy to use.

- . The system provides a great deal of power for storing data about sales and incoming stock transactions and for printing useful reports.
- . It also allows the store manager to update the master file from these transactions files with a simple menu choice.
- . When a user discover an error in either the sales or newstock file, it is easy to edit the transaction, and the system automatically takes care of making all the proper corrections to the master file.

صدقة جارية على روح  
أ.د. / مخرج الحداد

### The MASTER dataBase File :

---

This file keeps track of :

- The quantity of each item presently in stock ,
- The location of each item in the warehouse .
- The cost of each item ,
- The reorder point for each item ,
- The quantity currently on order of each item, and
- The name and address of the vendor the item is purchased from.

Accordingly, this file allows the user of the system to locate an item quickly, get a list of goods that need to be ordered, automatically create and print new orders, and perform other similar stock control tasks.

### SALES Transactions dataBase File :

---

This file keeps track of the following :

- To whom items were sold, when, for how much, and the invoice or receipt number.

### NEUSTOCK Received Transaction dataBase File :

---

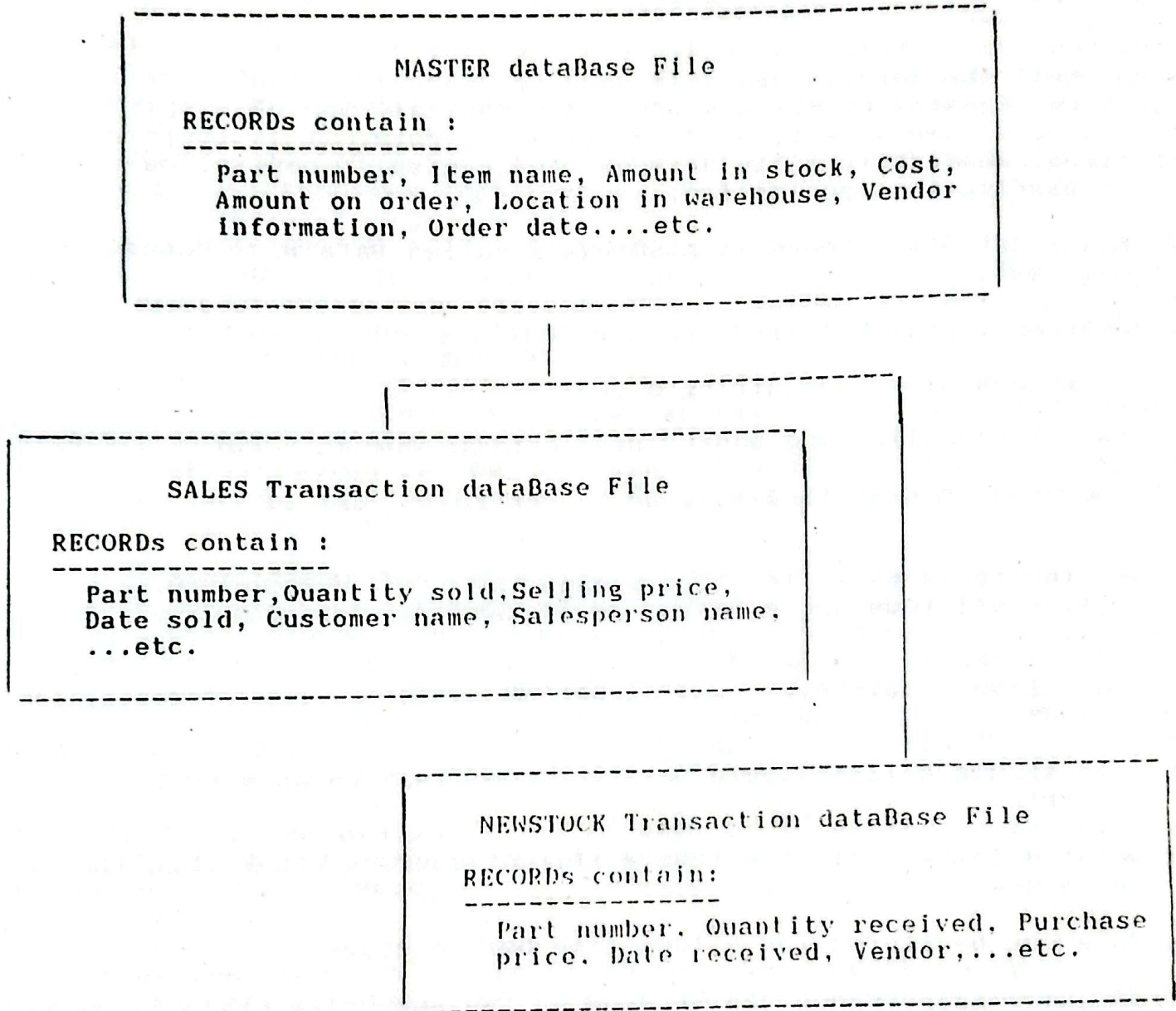
This file might maintain a record of all new stock received.

The transaction files maintain a history of all individual transactions including the sale and receipt of goods, while the master file maintains the current status of goods in stock at any given moment by using information supplied directly from the two transaction files.

In a sense, the master file is an up-to-date summary of all activities in the two transaction files.



The relationship between the master file and two such transaction files can be illustrated as follows:



( The Relationship of Typical Inventory System DataBase Files )



## SELECTED SOFTWARE AND DESIGN

---

### Why DataBase ? :

---

Sometimes, the word `dataBase` is used to mean simply any very large collection of data, but it has precise technical meaning, implying that attempts have been made to avoid duplicating data. It is therefore a mechanized, shared, formally defined, and centrally controlled collection of data used in an organization.

With any `dataBase`, there is a SOFTWARE called Data Base Management System ( DBMS ).

The `dataBase` approach is :

- . more than simply acquiring a DBMS
- . more than collecting and storing data in one integrated `dataBase`
- . more than designating someone to centrally control the data

It involves tools to collect and manage the data, a responsible and cooperative attitude among users, and a `dataBase` Administrator.

The `dataBase` should satisfy a number of technical requirements, these are :

- . Data should be accessible "at the same time" to as many users as possible.
- . Data could be collected from different sources other than that of the user.
- . Data can be stored for a long time before usage
- . `dataBase` maintenance can be done at the same time while it is used.
- . Data within the `dataBase` is organized in order to achieve a relation between different sets of data and to be able to accept new relations



As a conclusion, dataBase can have a variety of advantages, some of which are broadly explained in the following :

- Redundancy can be reduced : In a non-database system, each application has its own private files. This can often lead to considerable redundancy in stored data, with resultant waste in storage space.
- Inconsistency can be avoided ( to some extent ) : An inconsistent database system supplies incorrect or conflicting information.
- The data can be shared : It means not only that existing applications can share the data in the database, but also that new applications can be developed to operate on the same stored data. In other words, the data requirements of new applications may be satisfied without having to create any new stored data files.
- Standards can be enforced : With the central control of the database, the DBA ( Database Administrator ) can ensure that all applicable standards are followed in the representation of the data. Standardizing stored data formats is particularly desirable as an aid to data interchange or migration between systems.
- Security restrictions can be applied : The DBA can ensure that the only means of access to the database system is through the proper channels. The DBA also can define authorization checks to be carried out whenever access to sensitive data is attempted. Different checks can be established for each type of access ( retrieve, modify, delete, etc. ) to each piece of information in the database. [ note that without such checks, the security of the data may actually be more at risk in a database system than in a traditional filing system ].
- Integrity can be maintained : The problem of integrity is the problem of ensuring that the data in the database is accurate. Inconsistency between two entries representing the same "fact" is an example of lack of integrity.
- Conflicting requirements can be balanced : Knowing the overall requirements of the Enterprise ---- as opposed to the requirements of any individual user --- the DBA can structure the database system to provide an overall service that is "best for the Enterprise". For example, a representation can be chosen for the data in storage that gives fast access for the most important applications at the cost of poor performance in some other applications.



## THE INVENTORY SYSTEM DATABASE DESIGN

There are three fundamental dataBase Files ( Operational Data ) to manage the inventory system :

1. The MASTER DataBase file, called MASTER.DBF ,
2. The SALES Transaction file, called SALES.DBF , and
3. The NEW STOCK Transaction file, called NEWSTOCK.DBF.

### 1. The Structure of the Master dataBase file ( MASTER.DBF ) :

The structure of MASTER.DBF with a brief description of each field's contents is shown as follows:

| Field name     | Type      | Width | Dec | Contents                     |
|----------------|-----------|-------|-----|------------------------------|
| 1. PART_NO     | CHARACTER | 7     |     | Part number ( Key field )    |
| 2. TITLE       | CHARACTER | 20    |     | Item description             |
| 3. QTY         | NUMERIC   | 4     |     | Quantity in stock (On-hand)  |
| 4. COST        | NUMERIC   | 8     | 2   | Purchase price               |
| 5. SEL_PRICE   | NUMERIC   | 8     | 2   | Selling price                |
| 6. REORDER     | NUMERIC   | 4     |     | Reorder point                |
| 7. ON_ORDER    | NUMERIC   | 4     |     | Quantity on order            |
| 8. LOCATION    | CHARACTER | 7     |     | Location in warehouse        |
| 9. VENDOR      | CHARACTER | 25    |     | Vendor name                  |
| 10. VENDOR_ADD | CHARACTER | 25    |     | Vendor address               |
| 11. VENDOR_CSZ | CHARACTER | 25    |     | Vendor city, state, zip code |
| 12. DATE       | DATE      | 8     |     | Date of last update          |
| 13. ORDER_DATE | DATE      | 8     |     | Date of last order           |
| 14. NEW_ORDER  | NUMERIC   | 4     |     | Quantity to order            |



2. The Structure of the Sales dataBase file ( SALES.DBF ) :

The structure of the SALES.DBF is shown as follows

| Field name    | Type      | Width | Dec | Contents                       |
|---------------|-----------|-------|-----|--------------------------------|
| 1. PART_NO    | CHARACTER | 7     |     | Part number ( Key field )      |
| 2. INVOICE_NO | NUMERIC   | 6     |     | Invoice number                 |
| 3. SALSPERSON | CHARACTER | 25    |     | Salesperson's name ( or Code ) |
| 4. CUSTOMER   | CHARACTER | 25    |     | Customer name                  |
| 5. QTY        | NUMERIC   | 4     |     | Quantity sold                  |
| 6. PRICE      | NUMERIC   | 8     | 2   | Selling price                  |
| 7. DATE       | DATE      | 8     |     | Date sold                      |
| 8. POSTED     | LOGICAL   | 1     |     | Posted to MASTER.DBF yet ?     |

3. The Structure of the New Stock dataBase File ( NEWSTOCK.DBF ) :

The structure of the NEWSTOCK.DBF is shown as follows :

| Field name | Type      | Width | Dec | Contents                   |
|------------|-----------|-------|-----|----------------------------|
| 1. PART_NO | CHARACTER | 7     |     | Part number ( Key field )  |
| 2. QTY     | NUMERIC   | 4     |     | Quantity received          |
| 3. COST    | NUMERIC   | 8     | 2   | Purchase price             |
| 4. DATE    | DATE      | 8     |     | Date received              |
| 5. VENDOR  | CHARACTER | 25    |     | Vendor name                |
| 6. POSTED  | LOGICAL   | 1     |     | Posted to MASTER.DBF yet ? |

Structure for database: B:sales.dbf

Number of data records: 11

Date of last update : 03/29/89

| Field       | Field Name | Type      | Width | Dec |
|-------------|------------|-----------|-------|-----|
| 1           | PART_NO    | Character | 7     |     |
| 2           | INVOICE_NO | Numeric   | 6     |     |
| 3           | SALSPERSON | Character | 25    |     |
| 4           | CUSTOMER   | Character | 25    |     |
| 5           | QTY        | Numeric   | 4     |     |
| 6           | PRICE      | Numeric   | 8     | 2   |
| 7           | DATE       | Date      | 8     |     |
| 8           | POSTED     | Logical   | 1     |     |
| ** Total ** |            |           | 85    |     |

Structure for database: B:newstock.dbf

Number of data records: 6

Date of last update : 03/30/89

| Field       | Field Name | Type      | Width | Dec |
|-------------|------------|-----------|-------|-----|
| 1           | PART_NO    | Character | 7     |     |
| 2           | QTY        | Numeric   | 4     |     |
| 3           | COST       | Numeric   | 8     | 2   |
| 4           | DATE       | Date      | 8     |     |
| 5           | VENDOR     | Character | 25    |     |
| 6           | POSTED     | Logical   | 1     |     |
| ** Total ** |            |           | 54    |     |

Structure for database: B:newstock.dbf

Number of data records: 6

Date of last update : 03/30/89

| Field       | Field Name | Type      | Width | Dec |
|-------------|------------|-----------|-------|-----|
| 1           | PART_NO    | Character | 7     |     |
| 2           | QTY        | Numeric   | 4     |     |
| 3           | COST       | Numeric   | 8     | 2   |
| 4           | DATE       | Date      | 8     |     |
| 5           | VENDOR     | Character | 25    |     |
| 6           | POSTED     | Logical   | 1     |     |
| ** Total ** |            |           | 54    |     |



## THE INVENTORY SYSTEM SOFTWARE DESIGN

---

Now that we have designed the dataBase files for the Inventory System, we need to design the Software Structure. This is going to be a fairly large system, so preplanning is very important. The proposed system allows different individuals to perform the various major tasks :

The store manager will be responsible for managing the master inventory, placing orders, checking the status of goods, and so forth.

Individual sales transactions will be entered by salespersons at the point of sale, and

Incoming stock will be recorded by a stock room manager.

Three separate but related systems will be developed with the overall structure as shown on the next page:

-----  
I Menu.PRG ( Inventory System Main Menu ) :  
-----

1. Manage the MASTER.DBF Inventory file
2. Manage the SALES.DBF Information file
3. Manage the NEWSTOCK.DBF Information file, or
4. Exit

-----  
1. MMenu.PRG :  
-----

Add new part numbers,  
Print reports,  
Make changes, or  
Update the master file

( SECTION 1 )

-----  
2. SMenu.PRG :  
-----

Record sales,  
Print reports, or  
Make changes

( SECTION 2 )

-----  
3. NMenu.PRG :  
-----

Record items received,  
Print reports, or  
Make changes

( SECTION 3 )

-----  
Note that :

- The IMenu.PRG file will be the overall Inventory System Menu. Depending on the user's choice, it will branch to either:
  - 1.0 The MMenu.PRG to manage the master file ,
  - 2.0 The SMenu.PRG to handle individual sales transactions, or
  - 3.0 The NMenu.PRG to handle incoming stock (NewStock) transactions
- Each of the sub menus in the figure above will further branch to and execute other programs.



Menu.PRG ( Inventory System Main Menu ) :

\* Program...:IMenu.PRG  
\* Date.....: March, 1989  
\* Notes.....: Main Menu for the INVENTORY Management System.

SET STATUS OFF  
CLEAR ALL  
SET TALK OFF  
SET BELL OFF  
SET SAFETY OFF  
SET HEADING OFF  
CLEAR

Uline = REPLICATE( "\_", 70 )  
Date = Date()  
15, 5 SAY "Today's Date = " GET T\_Date PICT "99/99/99"  
17, 5 SAY "-----Enter new date, and/or press ENTER"  
READ

Choice = 0  
DO WHILE IChoice # 4  
CLEAR  
@ 2, 5 SAY "Inventory Management System MAIN Menu"  
@ 2,50 SAY DTOC( T\_Date ) + " " + Time()  
@ 3, 5 SAY Uline  
?  
?  
TEXT

Would you like to :

1. Manage the MASTER Inventory File ,
2. Handle individual SALES transactions ,
3. Handle NEW STOCK transactions , or
4. Exit

ENDTEXT  
@ 24, 5 SAY "Enter Your choice: " GET IChoice PICT "9" RANGE 1,4  
READ

```
DO CASE
  CASE IChoice = 1
    DO MMenu
  CASE IChoice = 2
    DO SMenu
  CASE IChoice = 3
    DO NMenu
ENDCASE
```

```
ENDDO
CLEAR
RETURN
eof()
```

Today's Date = 04/03/89  
-----Enter new date, and/or press ENTER

Inventory Management System MAIN Menu

04/03/89

01:43:09

Would you like to :

1. Manage the MASTER Inventory File ,
2. Handle individual SALES transactions ,
3. Handle NEW STOCK transactions , or
4. Exit

No  
--

Enter Your choice: 1



## SECTION 1

### THE SOFTWARE STRUCTURE FOR MANAGING THE MASTER.DBF FILE

MMenu.PRG is a submenu for managing the master dataBase file. It presents additional options to the user as follows:

#### 1. MMenu.PRG ( Master File Menu )

- 1.1. Add new part numbers
- 1.2. Print reports
- 1.3. Make changes
- 1.4. Update (will be discussed in section 4)

#### 1.5. Return to Main Menu

#### 1.1. AddNumbs.PRG

to add new part numbers

#### 1.1.a IScreen1.FMT

to create custom screen

#### 1.2. Print Reports

#### 1.2.a MReports.PRG

to print reports.

#### 1.2.b Orders.PRG:

to print the orders:

- 1.2.b1 The Reorder Report
- 1.2.b2 The On-Order Report
- 1.2.b3 The Purchase Orders

#### 1.3. MEdit.PRG

to edit and change data

#### 1.4. Updater.PRG

to update MASTER.DBF  
from Sales and NewStock  
(discussed in section 4)

(inv1.txt)

1. MMenu.PRG ( Master File Menu ) :

---

\* Program...: MMenu.PRG  
\* Date.....: March, 1989  
\* Notes.....: Menu to manage the MASTER.DBF file  
\*  
\*

MChoice = 0  
DO WHILE MChoice # 5  
CLEAR

@ 2,5 SAY "Managing MASTER.DBF File"  
@ 2,50 SAY DTOC(T\_Date) + " " + TIME()  
@ 3,5 SAY Uline  
?  
?  
TEXT

Would you like to :

1. ADD New Items with new Part Numbers ,
2. PRINT Reports ,
3. Make CHANGES ,
4. UPDATE from SALES.DBF and/or NEWSTOCK.DBF , or
5. RETURN to MAIN Menu

ENDTEXT

@ 21,5 SAY "Enter Choice(1-5)" GET MChoice PICT "9" RANGE 1,5  
READ

DO CASE  
CASE MChoice = 1  
DO ADDNUMBS  
CASE MChoice = 2  
DO MREPORTS  
CASE MChoice = 3  
DO MEDIT  
CASE MChoice = 4  
DO UPDATER

ENDCASE

ENDDO (WHILE MChoice # 5)

RETURN

\* eof()



Managing MASTER.DBF File

04/03/89

01:45:21

---

Would you like to :

1. ADD New Items with new Part Numbers ,
2. PRINT Reports ,
3. Make CHANGES ,
4. UPDATE from SALES.DBF and/or NEWSTOCK.DBF , or
5. RETURN to MAIN Menu

Enter Choice(1-5) 2

Inventory MASTER.DBF File Report Options

04/03/89

01:45:56

---

1. Entire Inventory
2. Re-order report
3. On-order report
4. Purchase orders
5. Return to Master Menu

Enter choice(1-5) 1

1.1 AddNumbs.PRG ( Adding Unique Part Numbers ) :

-----  
Program Algorithm:  
-----

- Use MASTER INDEX MASTER
- Create loop for adding new PART\_NOs
  - . Clear screen
  - . Ask user for new PART\_NO
  - . Check to see if this PART\_NO already exists
    - . If exists
      - notify user and allow another try
    - Otherwise,
      - let user add it using the custom screen format
  - . Continue loop for adding PART\_NOs
- Return to MMenu.PRG.

\* Program...: AddNumbs.PRG  
\* Date.....: March, 1989  
\* Notes.....: To add new items to the MASTER.DBF file  
\*-----

USE MASTER INDEX MASTER  
\*-----

PartNumb = "X"

DO WHILE PartNumb # " "

CLEAR

@ 2, 5 SAY "Add New Items with new Part Numbers"

@ 2,50 SAY DTOC( T\_Date ) + " " + Time()

@ 3, 5 SAY Uline

?

?

-----  
PartNumb = SPACE(7)

@ 15, 5 SAY "Enter part number (or press RETURN to EXIT)" GET Par  
READ

-----  
PartNumb = UPPER(PartNumb)

SEEK PartNumb  
\*-----

DO CASE

CASE PartNumb = " "

CLEAR

CASE FOUND()

@ 20, 10 SAY PartNumb + " already exists."

? CHR(7)

WAIT

\*



```
CASE .NOT. FOUND()  
  SET FORMAT TO IScreen1  
  APPEND BLANK  
  REPLACE PART_NO WITH PartNumb  
  REPLACE DATE WITH T_Date  
  READ  
  SET FORMAT TO
```

```
ENDCASE
```

```
ENDDO
```

```
RETURN
```

```
*eof()
```

Note :

PART\_NO is a Key field, therefore, each part number must be unique. To ensure this, the AddNumbs.PRG checks to make sure the new part-number is not duplicate. If it is duplicate, the program tells the user about this duplication ; for example:

X-111 already exists !

(where X-111 is an existing number) and gives a warning beep. Then it allows the user to try another number.

#### 1.1.a Creating a Custom Screen :

When the user does enter a unique part number, a custom data-entry screen like the one that follows appears on the screen. The required program to display this screen will be written on the next page.

MASTER.DBF File of Inventory Information System

Part number: A-111

Date 04/04/89

Part name:

Selling Price

Quantity in stock

Unit cost

Quantity on order

Reorder point

Storage location

Vendor

Name

Address

City, State, Zip

Cursor: Up, Down, Right, and Left arrow keys

INSERT MODE: Ins DELETE CHARACTER: Del DELETE FIELD: ^Y

DONE/SAVE: ^E

MASTER.DBF File of Inventory Information System

Part number: AT-4000

Date 03/27/89

Part name: TANDY AT/4000

Selling Price 7000.00

Quantity in stock 50

Unit cost 500.00

Quantity on order 10

Reorder point 10

Storage location AT-4000

Vendor

Metra Computers  
6, Ibn Malek street  
Giza, Cairo, Egypt

Name

Address

City, State, Zip

Cursor: Up, Down, Right, and Left arrow keys

INSERT MODE: Ins DELETE CHARACTER: Del DELETE FIELD: ^Y

DONE/SAVE: ^E



IScreen1.FMT ( Format File ) Program :

```
* Program...: IScreen1.FMT
* Date.....: March, 1989
* Notes....: A program to display the above custom screen
*
@ 2,18 SAY "MASTER.DBF File of Inventory Information System"
@ 5, 2 SAY "Part number: "
@ 5,15 GET MASTER->PART_NO
@ 5,45 SAY " Date"
@ 5,52 GET MASTER->DATE
*-----
@ 7, 2 SAY " Part name: "
@ 7,15 GET MASTER->TITLE
*-----
@ 9, 2 SAY "Quantity in stock"
@ 9,21 GET MASTER->QTY
@ 9,38 SAY " Unit cost"
@ 9,52 GET MASTER->COST
*-----
@ 11, 2 SAY "Quantity on order"
@ 11,21 GET MASTER->ON_ORDER
@ 11,38 SAY "Reorder point"
@ 11,52 GET MASTER->REORDER
*-----
@ 13, 2 SAY " Storage location"
@ 13,21 GET MASTER->LOCATION
*-----
@ 15, 9 SAY " Vendor"
@ 15,21 GET MASTER->VENDOR
@ 15,48 SAY "Name"
*-----
@ 16,21 GET MASTER->VENDOR_ADD
@ 16,48 SAY "Address"
*-----
@ 17,21 GET MASTER->VENDOR_CSZ
@ 17,48 SAY "City, State, Zip"
*-----
@ 21, 2 SAY "Cursor: Up, Down, Right, and Left arrow keys"
*-----
@ 22, 2 SAY "INSERT MODE: Ins DELETE CHARACTER: Del DELETE FIELD: ^Y;
DONE/SAVE: ^End"
*-----
@ 1, 0 TO 3,79 DOUBLE
@ 20, 0 TO 23,79 DOUBLE
*eof()
```

## 1.2 The Inventory MASTER.DBF File Reports :

---

The primary output from the MASTER.DBF file will be :

- . Reports concerning the status of the inventory and purchase orders for restocking.
- . The current inventory report will display for each item :
  - . Part number
  - . Item discription
  - . Quantity in stock
  - . Purchase price
  - . Reorder point
  - . Quantity on order
  - . Location in warehouse
  - . Vendor name, Vendor address
  - . Date of last update
  - . Date of last order
  - . Quantity to order



1.2.a MReports.PRG (The Inventory Master File Reports) :  
-----

\* Program...: MReports.PRG  
\* Date.....: March, 1989  
\* Notes.....: Present report options for Master File.  
\*

USE MASTER INDEX MASTER  
\*

RepChoice = 0

DO WHILE RepChoice # 5

CLEAR

@ 2, 5 SAY "Inventory MASTER.DBF File Report Options"

@ 2.50 SAY DTOC( T\_Date ) + " " + Time()

@ 3, 5 SAY Uline

?

?

TEXT

1. Entire Inventory

2. Re-order report

3. On-order report

4. Purchase orders

5. Return to Master Menu

ENDTEXT

@21, 5 SAY "Enter choice(1-5) " GET RepChoice PICT "9" RANGE 1,5

READ

\*

CLEAR

STORE " " TO YN . Printer

IF RepChoice < 4

@ 5, 5 SAY "Send report to printer(Y/N) ? " GET YN PICT "!"

READ

CLEAR

\*

IF YN = "Y"

Printer = "TO PRINT"

ENDIF

ENDIF  
\*

```
DO CASE
  CASE RepChoice = 1
    REPORT FORM AllMAST &Printer    && All data from MASTER.DBF
  CASE RepChoice = 2
    REPORT FORM ReOrders FOR (QTY + ON_ORDER) <= ReOrder &Printer
  CASE RepChoice = 3
    REPORT FORM OnOrder FOR ON_ORDER > 0 &Printer
  CASE RepChoice = 4
    DO Orders
ENDCASE
*
*
*
IF YN # "Y" .AND. RepChoice # 5
  ?
  ?
  WAIT "Press any key to return to the Reports Menu . . ."
ENDIF
*
ENDDO
RETURN
* eof()
```

Notice that:

- 
- . The first CASE displays all data from the MASTER.DBF.
  - . The second CASE displays only those records with on-hand quantity (QTY) plus on-order quantity (ON\_ORDER) is < the REORDER quantity
  - . The third CASE displays only those items already on order (ON\_ORDER > 0).
  - The fourth option, placing orders, is handled by a separate program called Orders.PRG .



Inventory MASTER.DBF File Report Options

04/04/89

00:20:32

1. Entire Inventory
2. Re-order report
3. On-order report
4. Purchase orders
5. Return to Master Menu

Enter choice(1-5) 1

ge No. 1  
/03/89

Current Stock

| rt-<br>umber | Discription   | On-<br>hand | Unit-<br>Cost | Selling-<br>Price | On-<br>Order | Re-<br>Order | Loc.    | Last-<br>update |
|--------------|---------------|-------------|---------------|-------------------|--------------|--------------|---------|-----------------|
| -3000        | TANDY AT/3000 | 20          | 4000.00       | 6000.00           | 10           | 50           | AT-3000 | 03/27/89        |
| -4000        | TANDY AT/4000 | 50          | 500.00        | 7000.00           | 10           | 10           | AT-4000 | 03/27/89        |
| -4000        | CASIO FP/4000 | 0           | 1500.00       | 3000.00           | 1000         | 50           | FP-4000 | 03/31/87        |
| -4100        | CASIO FP/4100 | 0           | 2000.00       | 3500.00           | 1000         | 50           | FP-4100 | 01/01/88        |
| -1000        | TANDY PC/1000 | 0           | 1500.00       | 3000.00           | 100          | 50           | PC-1000 | 01/01/88        |
| -30          | IBM PS/30     | 30          | 5000.00       | 8000.00           | 50           | 10           | PS-30   | 01/01/89        |
| -50          | IBM PS/50     | 10          | 6000.00       | 10000.00          | 100          | 50           | PS-50   | 01/01/89        |
| -60          | IBM PS/60     | 100         | 8000.00       | 18000.00          | 150          | 50           | PS-60   | 01/01/89        |
| Total        | ***           | 210         | 28500.00      | 58500.00          | 2420         | 320          |         |                 |

### 1.2.a The current-Stock Report :

You can create the report by typing at the dBASE dot prompt :

```
. USE MASTER  
. CREATE REPORT A11MAST && === > A11MAST.FRM (all data from  
MASTER.DBF)
```

and assigning the general format parameters as shown :

|                        | Current Stock |
|------------------------|---------------|
| Page title             |               |
| Page width (positions) | 80            |
| Left margin            | 1             |
| Right margin           | 0             |
| Lines per page         | 58            |
| Double space report    | No            |

Then include the information in column format as of the following figure

| Column No. | Contents | Heading     | Width | Decimals | Total? |
|------------|----------|-------------|-------|----------|--------|
| 1          | PART_NO  | Part        | 7     |          |        |
| 2          | TITLE    | Description | 20    |          |        |
| 3          | QTY      | On-Hand     | 5     | 0        | N      |
| 4          | COST     | Unit Cost   | 8     | 2        | N      |
| 5          | REORDER  | Reorder     | 5     | 0        | N      |
| 6          | LOCATION | Loc.        | 7     |          |        |
| 7          | DATE     | Last Update | 8     |          |        |



By using the " REPORT FORM " Command through the MReport.PRG, a report as the one follows appears :

e No. 1  
04/89

### Current Stock

| Item-<br>Number | Description   | On-<br>hand | Unit-<br>Cost | Selling-<br>Price | On-<br>Order | Re-<br>Order | Loc.    | Last-<br>update |
|-----------------|---------------|-------------|---------------|-------------------|--------------|--------------|---------|-----------------|
| 11              |               | 0           | 0.00          | 0.00              | 0            | 0            |         | 04/04/89        |
| 3000            | TANDY AT/3000 | 20          | 4000.00       | 6000.00           | 22           | 50           | AT-3000 | 03/27/89        |
| 4000            | TANDY AT/4000 | 50          | 500.00        | 7000.00           | 10           | 10           | AT-4000 | 03/27/89        |
| 4000            | CASIO FP/4000 | 0           | 1500.00       | 3000.00           | 1000         | 50           | FP-4000 | 03/31/87        |
| 4100            | CASIO FP/4100 | 0           | 2000.00       | 3500.00           | 1000         | 50           | FP-4100 | 01/01/88        |
| 1000            | TANDY PC/1000 | 0           | 1500.00       | 3000.00           | 100          | 50           | PC-1000 | 01/01/88        |
| 30              | IBM PS/30     | 30          | 5000.00       | 8000.00           | 50           | 10           | PS-30   | 01/01/89        |
| 50              | IBM PS/50     | 10          | 6000.00       | 10000.00          | 100          | 50           | PS-50   | 01/01/89        |
| 60              | IBM PS/60     | 100         | 8000.00       | 18000.00          | 150          | 50           | PS-60   | 01/01/89        |
| Total ***       |               | 210         | 28500.00      | 58500.00          | 2432         | 320          |         |                 |

## 1.2.b The Orders.PRG :

### 1.2.b1 The Reorder Report :

If the user selects option 2 from the Reports Menu, the program displays the reorder report like the one shown:

Page No. 1  
04/05/89

#### Goods to be Reordered

| Part-<br>number | Description   | On-<br>hand | On-<br>order | Re-<br>order | Vendor Name     |
|-----------------|---------------|-------------|--------------|--------------|-----------------|
| A-111           |               | 0           | 0            | 0            |                 |
| AT-3000         | TANDY AT/3000 | 20          | 22           | 50           | Metra Computers |
| *** Total ***   |               | 20          | 22           | 50           |                 |

For this to happen, you must first design the Reorder Report using dBASE dot prompt command:

- . USE MASTER
- . CREATE REPORT Reorders

with the parameters shown :

|                        |                       |
|------------------------|-----------------------|
| Page title             | Goods to be Reordered |
| Page width (positions) | 80                    |
| Left margin            | 1                     |
| Right margin           | 0                     |
| Lines per page         | 58                    |
| Double space report    | No                    |

and the column format as in the following figure :

| Column No. | Contents | Heading     | Width | Decimals | Total? |
|------------|----------|-------------|-------|----------|--------|
| 1          | PART_NO  | Part        | 7     |          |        |
| 2          | TITLE    | Description | 20    |          |        |
| 3          | QTY      | On-Hand     | 5     | 0        | N      |
| 4          | ON_ORDER | On-Order    | 5     | 0        | N      |
| 5          | REORDER  | Reorder     | 5     | 0        | N      |
| 6          | VENDOR   | Vendor Name | 25    |          |        |



### 1.2.b2 The ON\_ORDER Report :

When the user selects option 3 from the Reports Menu, a report of the goods currently on order is displayed.

The necessary dBASE dot prompt commands are as follows :

- . USE MASTER
- . CREATE REPORT OnOrder

with the parameters shown in the following figure:

|                        |                          |
|------------------------|--------------------------|
| Page title             | Items Currently on Order |
| Page width (positions) | 80                       |
| Left margin            | 1                        |
| Right margin           | 0                        |
| Lines per page         | 58                       |
| Double space report    | No                       |

and the column format from the following figure:

| Column No. | Contents        | Heading     | Width | Decimals | Total? |
|------------|-----------------|-------------|-------|----------|--------|
| 1          | ORDER_DATE      | Order Date  | 8     |          |        |
| 2          | PART_NO         | Part No.    | 7     |          |        |
| 3          | TITLE           | Description | 16    | 0        | N      |
| 4          | ON_ORDER        | On-Order    | 5     | 0        | N      |
| 5          | COST            | Unit Cost   | 8     | 2        | N      |
| 6          | COST * ON_ORDER | Total Cost  | 8     | 2        | Y      |
| 7          | VENDOR          | Vendor Name | 20    |          |        |

( Notice the mathematical expression to create the values of the Total Cost column )

Page No. 1  
4/05/89

Items Currently on Order

| Order Date | Part-number | Description   | On-Order | Unit Cost | Total Cost | Vendor Name     |
|------------|-------------|---------------|----------|-----------|------------|-----------------|
| 4/03/89    | AT-3000     | TANDY AT/3000 | 22       | 4000.00   | 88000.00   | Metra Computer  |
| / /        | AT-4000     | TANDY AT/4000 | 10       | 500.00    | 5000.00    | Metra Computer  |
| / /        | FP-4000     | CASIO FP/4000 | 1000     | 1500.00   | 1500000    | Cairo Trading   |
| / /        | FP-4100     | CASIO FP/4100 | 1000     | 2000.00   | 2000000    | Cairo Trading   |
| / /        | PC-1000     | TANDY PC/1000 | 100      | 1500.00   | 150000.0   | Metra Computer  |
| / /        | PS-30       | IBM PS/30     | 50       | 5000.00   | 250000.0   | Egypt for Compu |
| / /        | PS-50       | IBM PS/50     | 100      | 6000.00   | 600000.0   | Egypt for Compu |
| / /        | PS-60       | IBM PS/60     | 150      | 8000.00   | 1200000    | Egypt for Compu |
| Total      | ***         |               | 2432     | 28500.00  | 5793000    |                 |



### 1.2.b3 The Purchase Orders :

Option 4 from the Reports Menu allows the user to place orders and have the purchase orders printed automatically.

When the user selects this option, the screen displays some basic information about each item in stock that has fallen below the reorder point as follows:

Part number: AT-3000 TANDY AT/3000  
On hand: 20  
On order: 22  
Reorder-Point: 50  
Unit Cost: 4000.00

Order how many ? 5 ↵

To reorder more, the user simply fills the quantity to order at the "Order how many ?" prompt.

After the program has gone through the entire list of items that need to be reordered, it prints the purchase orders automatically. The following figure shows a sample purchase order created by the Inventory System:

Metra Computers  
6, Ibn Malek street  
Giza, Cairo, Egypt

Please send us the following items...

|                 |         |          |
|-----------------|---------|----------|
| 5 TANDY AT/3000 | 4000.00 | 20000.00 |
| Total cost:     |         | 20000.00 |

Mail to: Abdalla El-Daoushy, Inc.  
24 Abu Hayaan El-Tawhidi  
Nasr-city, Cairo

Orders.PRG ( The Purchase Orders Program ) :

-----  
This program performs several tasks :

- . Finds and determines which items are due to for reordering ( by checking whether the amount on-hand + the amount on-order is less than the reorder-point ).
  - . Asks the user how many to order.
  - . Prints the orders and updates the on-order field in the MASTER.DBF file.
- \* Program...: Orders.PRG  
\* Date.....: March, 1989  
\* Notes.....: Create purchase orders for reordering.  
\*

CLEAR

USE MASTER INDEX MASTER

\*

- \* Loop through and display goods below reorder point and ask user
- \* how many of each to order.
- \*

GO TOP

DO WHILE .NOT. EOF()

\*

- \* Find out if on-hand + on-order quantity is < reorder point
- \*

IF ( QTY + ON\_ORDER ) < REORDER

CLEAR

@ 5, 5 SAY " Part number:" + PART\_NO + " " + TITLE  
@ 6, 5 SAY " On hand:" + STR( QTY , 4 )  
@ 7, 5 SAY " On order:" + STR( ON\_ORDER , 4 )  
@ 8, 5 SAY "Reorder-Point:" + STR( REORDER , 4 )  
@ 9, 5 SAY " Unit Cost:" + STR( COST , 9 , 2 )

- \* Ask user how many to order.
- \*

@12, 5 SAY "Order how many ? " GET NEW\_ORDER PICT "9999"  
REPLACE ORDER\_DATE WITH T\_Date  
READ

ENDIF

SKIP

ENDDO

\*

- \* When all orders have been placed, make a temporary file ( TEMP
- \* of items to be ordered.
- \*

CLEAR

@10, 5 SAY "Preparing Files . . . Please Wait"

@13,10 SAY "( Please Prepare PRINTER while Waiting )"



```
COPY TC TEMP FOR NEW_ORDER > 0
*
* Update ON_ORDER field in MASTER.DBF file with new orders, then
* set the NEW_ORDERS field back to Zero.
*
REPLACE ALL ON_ORDER WITH ON_ORDER + NEW_ORDER
REPLACE ALL NEW_ORDER WITH 0
*
* Use TEMP File ( which contains new orders ), Indexed by VENDOR.
*
USE TEMP
INDEX ON UPPER(VENDOR) TO TEMP
*
CLEAR
? CHR(7)
@ 23, 5 SAY "Ready Printer to print orders"
WAIT
SET PRINT ON
GO TOP
*
* Loop through TEMP File
*
DO WHILE .NOT. EOF()
*
* for each vendor, print name and address
*
    THIS_LOOP = VENDOR
    MTOTAL = 0
    ? VENDOR
    ? VENDOR_ADD
    ? VENDOR_CSZ
    ?
    ? "    Please send us the following items... "
    ?
    ?
    *
    * for each item to be ordered from this Vendor,
    * print quantity, item, and price
    *
    DO WHILE VENDOR = THIS_LOOP .AND. .NOT. EOF()
        ? NEW_ORDER , TITLE , COST , NEW_ORDER * COST
        MTOTAL = MTOTAL + NEW_ORDER * COST
        SKIP
    ENDDO
    ?
```

```
*  
* When done with this Vendor, print total cost and  
* shipping name and address.  
?  
? "Total cost: " , MTOTAL  
?  
?  
? "Mail to: Abdalla El-Daoushy, Inc."  
? " 24 Abu Hayaan El-Tawhidi"  
? " Nasr-city, Cairo"  
EJECT  
ENDDO ( continue for each Vendor in TEMP File )  
*  
SET PRINT OFF  
RETURN  
* eof()
```



1.3. The MEdit.PRG ( Editing and Changing Data in MASTER.DBF File ) :

-----

Option 3 from the Master Menu allows the user to edit and change the Master file.

Program Algorithm:

-----

- USE MASTER INDEX MASTER
- Construct loop for performing edits:
  - . Find out what PART\_NO to edit
  - . Try to find that PART\_NO
  - . If found,
    - Edit the RECORD, using the custom screen format
  - . If not found,
    - Warn user and allow another try
  - . Continue Editing until user requests EXIT
- Return to MMenu.PRG (Master Menu Program).

```
*
* Program...: MEdit.PRG
*
* Edit the Inventory Master File.
* Called from Master menu, MMenu.PRG.
*
USE MASTER INDEX MASTER
*
PartNumb = "X"
DO WHILE PartNumb # " "
  *
  * find out what part number to edit
  *
  CLEAR
  @ 2, 1 SAY "Edit Inventory Master File"
  @ 2,60 SAY DTOC( T_Date ) + " " + Time()
  @ 3, 0 SAY Uline
  ?
  ?
  PartNumb = SPACE( 7 )
  @15, 5 SAY "Edit for what part number ? GET PartNumb"
  READ
  *
  * Try to find that part number.
  *
  PartNumb = UPPER( PartNumb )
  SEEK PartNumb
  DO CASE
    * if no part number entered, return to Master Menu.
    CASE PartNumb = " "
```

04/05/89

00:25:35

Edit Inventory Master File

Edit for what part number ? PS-30

MASTER.DBF File of Inventory Information System

Part number: PS-30

Date 01/01/89

Part name: IBM PS/30

Selling Price 8000.00

Quantity in stock 30

Unit cost 5000.00

Quantity on order 50

Reorder point 10

Storage location PS-30

Vendor Egypt for Computers

Arab Ligue street

Mohandiseen, Cairo, Egypt

Name

Address

City, State, Zip

Cursor: Up, Down, Right, and Left arrow keys

INSERT MODE: Ins DELETE CHARACTER: Del DELETE FIELD: ^Y

DONE/SAVE: ^Z



- 33 - SECTION

CLEAR

\*

CASE FOUND()

SET FORMAT TO IScreen1

READ

SET FORMAT TO

\* otherwise, warn User and allow another try.

CASE .NOT. FOUND()

@17, 5 SAY "There is no part " + PartNumb

@24, 5 SAY "Please any key to try again..."

WAIT " "

ENDCASE

ENDDO

\*

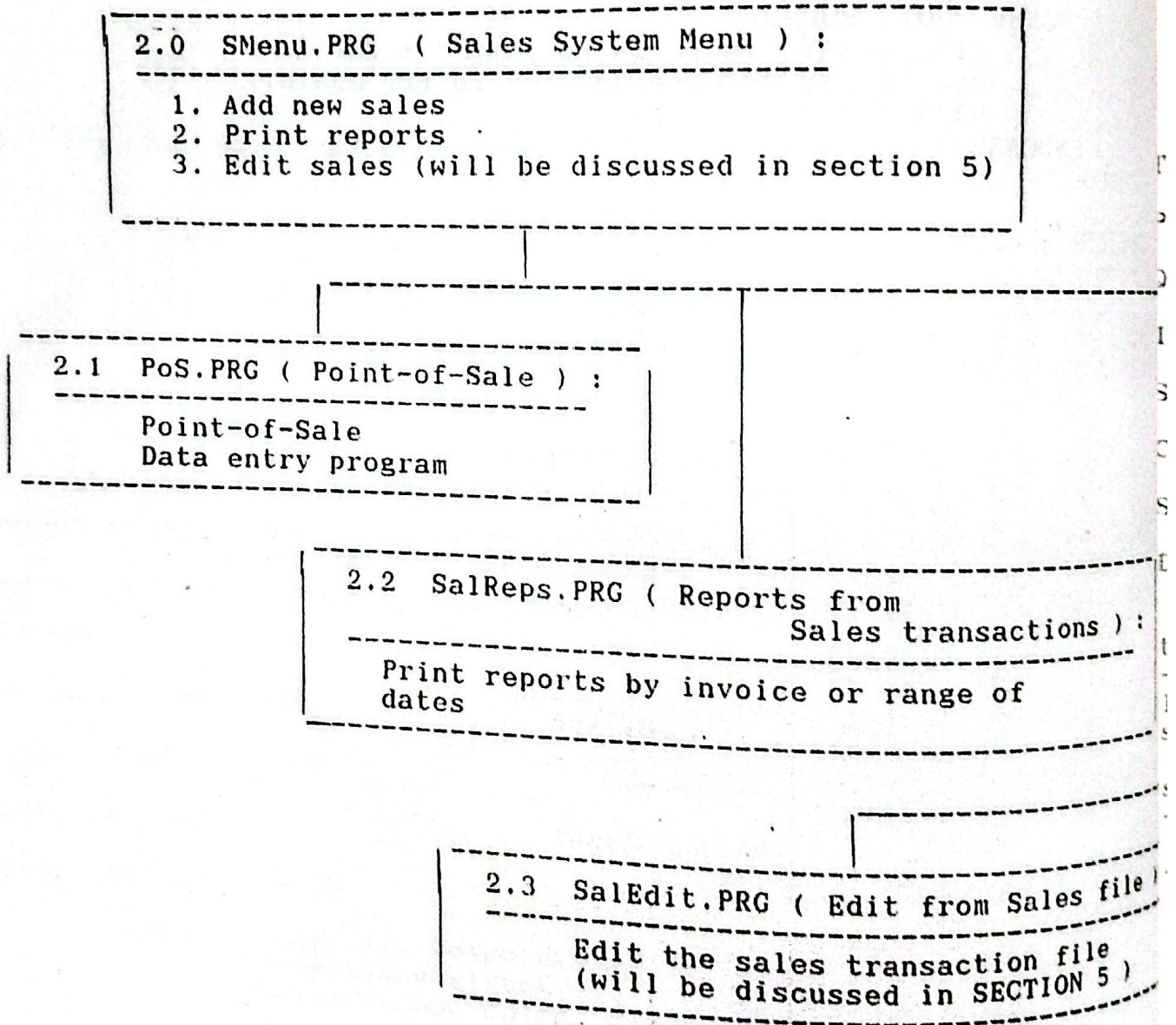
RETURN

\* eof()

## SECTION 2

### 2. SALES SYSTEM SOFTWARE STRUCTURE

Consider the following structure:





The structure above is created so that three programs are linked to Menu.PRG program. These three programs are :

PoS.PRG, which allows the user's to add new data to the Sales.DBF at the point-of-sale.

SalReps.PRG, which prints reports from the sales transaction file based on invoice number or date.

SalEdit.PRG, which allows the user to edit from the sales files.

The following items could be produced from the sales transaction file :

Part number,

Quantity,

Invoice ( Receipt ) number,

Salesperson's name,

Customer's name,

Selling price, and

Date.

Note:

-----  
In this section, SMenu.PRG, PoS.PRG, and SalReps.PRG programs are designed and developed, but Edit.PRG program will be created at the

-----  
last section ( SECTION 5 ) because editing procedures will be affected  
-----  
by the updating program developed in section 4.  
-----

## 2.0 SMenu.PRG ( Sales System Menu ):

When the user chooses option 2 from the Inventory Main Menu, Menu appears on the screen and the program then branches to the indicated by the user's selection.

\* Program...: SMenu.PRG  
\* Date.....: March, 1989  
\* Notes....: Menu for Sales portion of the Inventory System.

```
SChoice = 0
DO WHILE SChoice # 4
  CLEAR
  @ 2, 5 SAY "Sales System Menu"
  @ 2,50 SAY DTOC( T_Date ) + "      " + Time()
  @ 3, 5 SAY Uline
  ?
  ?
  TEXT
```

1. Enter Point-of-Sale routine
2. Print Sales Reports
3. Edit Sales data
4. Return to Main Menu

ENDTEXT

```
@20, 5 SAY "Enter choice(1-4) " GET SChoice PICT "9"
READ
```

```
DO CASE
  CASE SChoice = 1
    DO Pos
  CASE SChoice = 2
    DO SalReps
  CASE SChoice = 3
    DO SalEdit
ENDCASE
```

```
ENDDO ( WHILE SChoice # 4 )
```

```
RETURN
* eof()
```



## 2.1 PoS.PRG ( Point-of-Sale Data Entry ) :

### ----- Program Algorithm: -----

- USE SALES
- Get last-used INVOICE\_NO from the SALES.DBF
- Create TEMorary file for new transactions
- USE MASTER , TEMPorary
- Set up relationship between MASTER.DBF and SALES.DBF
- Construct loop for displaying Invoice forms
  - # Set up top portion of the Invoice on the screen
  - # Set up loop for each item on the Invoice
    - . Set up loop to check validity of PART\_NO
      - . Set up Invoice memory variables
      - . Ask for PART\_NO
      - . Make sure PART\_NO exists
        - . If not, warn user
        - . If exists, display it
          - . GET QTY and Selling price
          - . Display total
          - . Increment grand total
      - . Store transaction on TEMPorary file
    - . Continue loop for checking for valid PART\_NO
    - . Increment row counter
    - . If nearing bottom of screen, Scroll everything up a line
  - # Continue loop for adding transactions
    - . Display grand total
    - . Ask about printing Invoice
    - . If print requested, print Invoice
  - # Continue loop to add transactions
- Close DataBases and update the SALES.DBF file
- Clear out the TEMPorary file
- Return to SMenu.PRG

```
* Program...: Pos.PRG
* Date.....: March, 1989
* Notes.....: Point-of-sale data entry program for Sales transaction
*
USE SALES
GO BOTT
Minvoice = Invoice_No    && the last invoice # in the SALES.DBF file
*
* Add new transactions to a TEMPorary File.
*
SET SAFETY OFF
COPY STRUCTURE TO TEMPInv
*
USE
*
SELECT A
USE MASTER INDEX MASTER
SELECT B
USE TEMPInv
SET RELATION TO PART_NO INTO MASTER
*
StartTrans = 1
Ulinehere = REPLICATE("_" , 74 )
*
Again = "Y"
DO WHILE Again = "Y"
    * set up top portion of invoice on the screen
    CLEAR
    Minvoice = Minvoice + 1
    STORE SPACE(25) TO MClerk, MCust    && Clerk means Salesperson
    MTotals = 0
    @ 1, 1 SAY T_Date
    @ 1,30 SAY "Invoice number:" + STR(Minvoice, 5)
    @ 3, 1 SAY "Salesperson:" GET MClerk
    @ 3,40 SAY "Customer:" GET MCust
    @ 4, 1 SAY Ulinehere
    ? " Part # Name" , SPACE(20)
    ?? "Qty Price Total"
    READ
    *
    Row = 7
    Adding = .T.
    DO WHILE Adding
        PartNumb = SPACE(8)
        OK = .F.
        *
        * check validity of part number
```



```
DO WHILE .NOT. OK
  Quantity = 0
  * -----ask for part number.
  PartNumb = SPACE(6)
  @ Row, 2 GET PartNumb
  READ
  * -----make sure part number exists.
  PartNumb = UPPER( TRIM( PartNumb ) )
  *
  SELECT A      && MASTER.DBF
  SEEK PartNumb
  *
DO _CASE
  CASE LEN( PartNumb ) = 0 ( no part number was entered )
    OK = .T.
    Adding = .F.
  CASE .NOT. FOUND()
    @ Row,10 SAY "No such Part ! ! !"
    OK = .F.
  CASE FOUND()
    @ Row,10 SAY TITLE      && Part discription
    @ Row,35 GET Quantity PICT "999"
    @ Row,40 SAY Sel_Price PICT "99999.99"
    READ
    * -- If Quantity is zero, loop. Else, display Total.
    IF Quantity = 0
      LOOP
    ELSE
      Sel_Price = Sel_Price
      @ Row,50 SAY Quantity + Sel_Price PICT "##,###.##"
      MTotal = MTotal + Quantity * Sel_Price
      OK = .T.
    ENDIF
  *
  * Add a Blank Record to TEMPInv and fill in the fields.
  *
  SELECT B      && TEMPInv
  APPEND BLANK
  REPLACE DATE WITH T_Date
  REPLACE SALSPERSON WITH MClerk
  REPLACE INVOICE_NO WITH Minvoice
  REPLACE CUSTOMER WITH MCust
  REPLACE PART_NO WITH PartNumb
  REPLACE QTY WITH Quantity
  REPLACE PRICE WITH Sel_Price
  REPLACE POSTED WITH .F.
ENDCASE
*
ENDDO ( While .NOT. OK )
*
Row = Row + 1
```

```

* -----Scroll screen if nearing end
IF Row >= 19
  @ 24, 1
  ?
  Row = 19
ENDIF
ENDDO ( While Adding )
* -----Display Grand total, and pause before next Invoice.
@ Row+2,40 SAY "Total: "
@ Row+2,50 SAY MTotal PICT "$##,###.##"
Pinvoice = "Y"
@ 23, 2 SAY "Print invoice ? (Y/N) " GET Pinvoice PICT "!"
READ
* -----Print invoice, reset StartTrans.
IF Pinvoice = "Y"
SET PRINT ON
? "          Date:" , T_Date
? "Invoice number:" , STR( MInvoice , 6 )
? "          Customer:" , MCust , SPACE( 3 )
?? "Salesperson:" , MClerk
? Ulinehere
?
SELECT B      && TEMPInv
GO StartTrans
LIST OFF WHILE .NOT. EOF() PART_NO , A->TITLE , QTY , PRICE ,
QTY * PRICE
?
?
? "Total price:" , SPACE( 2 ) , MTotal
EJECT
SET PRINT OFF
StartTrans = RECCOUNT() + 1
ENDIF
wait
CLEAR
@ 23, 5 SAY "Do another transaction (Y/N)?" GET Again PICT "!"
READ
ENDDO ( While Adding )
*
CLOSE DATABASES
CLEAR
? "Updating transaction file, please wait..."
SET TALK ON
USE SALES
APPEND FROM TEMPInv
USE TEMPInv
ZAP
USE SALES INDEX SALES
REINDEX
SET TALK OFF
CLOSE DATABASES
RETURN
* eof()

```



This program adds new data to SALES.DBF by displaying the following screen which looks like an invoice :

04/05/89

Invoice number: 2

Salesperson:

Customer:

| Part # | Name | Qty | Price | Total |
|--------|------|-----|-------|-------|
|--------|------|-----|-------|-------|

The user types the clerk ( Salesperson ) name and the customer's name. Then a highlight for entering the part number appears on the screen.

The user enters the part number of the item sold. If that part number does not exist, the computer provides a warning and the user can try again. When the valid number is entered, the name of the part appears automatically along with prompts for filling in the quantity and selling price.

04/05/89

Invoice number: 2

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

| Part #  | Name          | Qty | Price   | Total     |
|---------|---------------|-----|---------|-----------|
| AT-4000 | TANDY AT/4000 | 5   | 7000.00 | 35,000.00 |

- If the user wants to enter another valid transaction with the same part number, the program calculates the total and displays it on the screen as follows:

04/05/89

Invoice number: 2

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

| Part #  | Name          | Qty | Price   | Total     |
|---------|---------------|-----|---------|-----------|
| AT-4000 | TANDY AT/4000 | 5   | 7000.00 | 35,000.00 |
| AT-3000 | TANDY AT/3000 | 2   | 6000.00 | 12,000.00 |

Total: \$47,000.00

Print invoice ? (Y/N) Y

The PoS.PRG program is complex, but it provides a great deal of convenience for the user. It also minimizes the likelihood of errors by checking the validity of the part numbers entered, as well as by immediately performing and displaying calculations for each transaction. It is a very general Point-of-Sale program, and it would probably need to be modified to suit each business's exact needs.

Note :

Automatic selling price: The PoS.PRG assumes the user enters the selling price for each transaction. As an alternative, you can store the selling price for each item on the MASTER.DBF file and have it come to the screen automatically when the user enters the appropriate part number. You need only to change the line that reads:

@ row,40 GET Sel\_Price PICT "999.99"

to @ row,40 SAY Sel\_Price PICT "999.99"



Inventory Management System MAIN Menu

04/05/89

00:47:36

-----  
Would you like to :

1. Manage the MASTER Inventory File ,
2. Handle individual SALES transactions ,
3. Handle NEW STOCK transactions , or
4. Exit

Enter Your choice: 2

Sales System Menu

04/05/89

00:48:43

- 
1. Enter Point-of-Sale routine
  2. Print Sales Reports
  3. Edit Sales data
  4. Return to Main Menu

Enter choice(1-4) 1

04/05/89

Invoice number: 2

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

| Part #  | Name          | Qty | Price   | Total     |
|---------|---------------|-----|---------|-----------|
| AT-4000 | TANDY AT/4000 | 5   | 7000.00 | 35,000.00 |
| AT-3000 | TANDY AT/3000 | 2   | 6000.00 | 12,000.00 |

Date: 04/05/89

Invoice number: 2

Customer: Dr. M. Y. Youssef

Salesperson: Eng. Magdi

|         |               |   |         |          |
|---------|---------------|---|---------|----------|
| AT-4000 | TANDY AT/4000 | 5 | 7000.00 | 35000.00 |
| AT-3000 | TANDY AT/3000 | 2 | 6000.00 | 12000.00 |

Total price: 47000.00



## 2.2 SalReps.PRG ( Sales System Reports ) :

### Program Algorithm:

- USE SALES ., MASTER
- Set up relationship
- Construct loop for menu options
  - 1. By Invoice number
  - 2. By Dates
  - 3. Return to SMenu.PRG
- Get user's option
  - 1. Search by INVOICE\_NO
    - Ask for INVOICE\_NO
    - Locate first transaction with that INVOICE\_NO
    - If found, print Invoice
  - 2. Search by Dates
    - Ask for starting and ending dates
    - Clear screen
    - Print report for those dates, using SALES.FRM
- Continue to loop through menu
- Return to SMenu.PRG.

\* Program...: SalReps.PRG  
\* Date.....: March, 1989  
\* Notes.....: Print reports from the SALES.DBF File.  
\*

SELECT 1      && see relational dataBases.....  
USE SALES

SELECT 2  
USE MASTER INDEX MASTER  
\*

SELECT 1      && SALES.DBF  
SET RELATION TO PART\_NO INTO MASTER  
,

RepChoice = 0

DO WHILE RepChoice # 3

CLEAR

@ 2, 5 SAY "Sales Report Options"

@ 2,50 SAY DTOC( T\_Date ) + " " + Time()

@ 3, 5 SAY Uline

?

?

TEXT

1. By Invoice Number

2. By Dates

### 3. Return to Sales Menu

```

ENDTEXT
@ 18, 5 SAY "Enter Choice(1-3)" GET RepChoice PICT "9" RANGE 1,3
READ
*
@ 5, 0 CLEAR
STORE " " TO YN , Printer
IF RepChoice # 3
    @ 15, 5 SAY "Send to Printer? " GET YN PICT "!"
    READ
    IF YN = "Y"
        Printer = "TO PRINT"
    ENDIF
ENDIF
ENDIF
*
@ 5, 0 CLEAR
DO CASE
    CASE RepChoice = 1
        @ 15, 5
        INPUT "Look for what Invoice Number? " TO ISearch
        CLEAR
        *
        LOCATE FOR INVOICE_NO = ISearch
        *
        IF FOUND()
            IF YN = "Y"
                SET PRINT ON
            ENDIF
            * ---- Print header info from first record with that
            * ---- Invoice number,
            ? "Invoice Number:" , INVOICE_NO
            ?? "      Date:" , DATE
            ?
            ? "Salseperson:" , SALSPERSON , "      Customer:" , CUSTOMER
            ?
            ?
            ? "Part #      Part Name                      Qty      Price          To
            *
            * ---- Print Date for all records with that invoice no.
            *
            LIST OFF WHILE INVOICE_NO = ISearch;
            PART_NO , B->TITLE , QTY , PRICE , ( QTY * PRICE )
            IF YN = "Y"
                EJECT
                SET PRINT OFF
            ENDIF
        ENDIF (found)
        *

```



```
CASE RepChoice = 2
  STORE SPACE(8) TO Start , End
  @ 15, 5 SAY "Enter start date " GET Start PICT "99/99/99"
  @ 17, 5 SAY "Enter End date   " GET End PICT "99/99/99"
  READ
  Start = CTOD( Start )
  End = CTOD( End )
  CLEAR
  *
  * ----- Print the Report.
  *
  REPORT FORM SALES;
    FOR DATE >= Start .AND. DATE <= End &Printer
```

```
ENDCASE
```

```
* --- if not going to printer, pause the screen.
```

```
IF YN # "Y" .AND. RepChoice # 3
```

```
  ?
  ?
  WAIT "Press any key to return to the Reports Menu..."
```

```
ENDIF
```

```
NDDO
```

```
---- when done, return to Sales Menu.
```

```
ET RELATION TO
LOSE DATABASES
RETURN
eof()
```

- This program allows the user to print reports from the SALES.DBF. Reports can be based on either a single invoice number or sales transactions for a range of dates; e.g. all sales transactions from date() to another date().
- When the user decides to print a report of sales, a menu like the following menu appears :

Sales Report Options

date() time()

1. By invoice number
2. By dates
3. Return to Sales menu

Enter choice (1-3) |||

- A report for a specified invoice number will be as follows :

Invoice number : 12345  
Clerk: ACS

Date 03/17/88  
Customer : D. Jones

| Part # | Qty | Part name     | Price | Total |
|--------|-----|---------------|-------|-------|
| Z-999  | 2   | Penny loafers | 12.00 | 24.00 |
| A-113  | 3   | Sandals       | 15.00 | 45.00 |
| A-111  | 1   | Wingtips      | 90.00 | 90.00 |

- You can create the report format by using the SALES.DBF, the d MODIFY REPORT command, and the REPORT FORM parameters :

| Page title            | Sales Transactions |
|-----------------------|--------------------|
| Page width (position) | 80                 |
| Left margin           | 1                  |
| Right margin          | 0                  |
| Lines per page        | 58                 |
| Double space report   | No                 |



and the information in column format :

| Column | Contents   | Heading     | Width | Decimals | Total? |
|--------|------------|-------------|-------|----------|--------|
| 1      | DATE       | Date        | 8     |          |        |
| 2      | INVOICE_NO | Invoice     | 7     |          |        |
| 3      | SALSPERSON | Salesperson | 20    |          |        |
| 4      | CUSTOMER   | Customer    | 15    |          |        |
| 5      | PART_NO    | Part number | 7     |          |        |
| 6      | QTY        | Qty         | 3     | 0        | N      |
| 7      | PRICE      | Sale Price  | 8     | 2        | Y      |

- A summary of all the sales transactions will look like those on the following pages.

1. By Invoice Number
2. By Dates
3. Return to Sales Menu

Enter Choice(1-3) 1

Sales Report Options

04/05/89 00:57:0

Look for what Invoice Number? 1

Invoice Number: 1

Date: 04/03/89

Salseperson: Eng. Hosam

Customer: Dr. Abdalla

| Part #  | Part Name     | Qty | Price   | Total    |
|---------|---------------|-----|---------|----------|
| AT-4000 | TANDY AT/4000 | 5   | 7000.00 | 35000.00 |
| AT-4000 | TANDY AT/4000 | 1   | 7000.00 | 7000.00  |



Sales Report Options

04/05/89

00:59:47

---

Look for what Invoice Number? 2

Invoice Number: 2 Date: 04/05/89

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

| Item # | Part Name     | Qty | Price   | Total    |
|--------|---------------|-----|---------|----------|
| -4000  | TANDY AT/4000 | 5   | 7000.00 | 35000.00 |
| -3000  | TANDY AT/3000 | 2   | 6000.00 | 12000.00 |

Edit Sales Transactions

04/05/89

01:01

Enter Invoice number to edit (0 if none): 1

Edit Sales Transactions

04/05/89

01:01

| RecNo         | Part #  | QTY | Price   | Salesperson | Customer    |
|---------------|---------|-----|---------|-------------|-------------|
| 1<br>04/03/89 | AT-4000 | 5   | 7000.00 | Eng. Hosam  | Dr. Abdalla |
| 2<br>04/03/89 | AT-4000 | 1   | 7000.00 | Eng. Hosam  | Dr. Abdalla |

Edit which RECORD: 2



Edit Sales Transactions

04/05/89

01:03:14

Invoice number: 2

Date 04/05/89

Part number: AT-4000

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

Quantity: 5

Selling Price: 7000.00

Delete this RECORD? (Y/N)

Edit Sales Transactions

04/05/89

01:04:10

Invoice number: 2

Date 04/05/89

Part number: AT-4000

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

Quantity: 5

Selling Price: 7000.00

Delete this RECORD? (Y/N)

# Sales Report Options

04/05/89

00:57:31

Enter Start date     /   /

Enter End date       /   /

Enter Start date     01/01/89  
Enter End date       04/31/89

Page No.       1  
04/05/89

## Sales Transactions

| Date          | Invoice | Salesperson      | Customer          | Part number | Qty |
|---------------|---------|------------------|-------------------|-------------|-----|
| 04/03/89      | 1       | Eng. Hosam       | Dr. Abdalla       | AT-4000     | 5   |
| 04/03/89      | 1       | Eng. Hosam       | Dr. Abdalla       | AT-4000     | 1   |
| 04/05/89      | 2       | Eng. Magdi Maher | Dr. M. Y. Youssef | AT-4000     | 5   |
| 04/05/89      | 2       | Eng. Magdi Maher | Dr. M. Y. Youssef | AT-3000     | 2   |
| *** Total *** | 6       |                  |                   |             | 13  |



### SECTION 3

#### 3. SOFTWARE STRUCTURE FOR THE NEWSTOCK SYSTEM

This system consists of 4 programs :

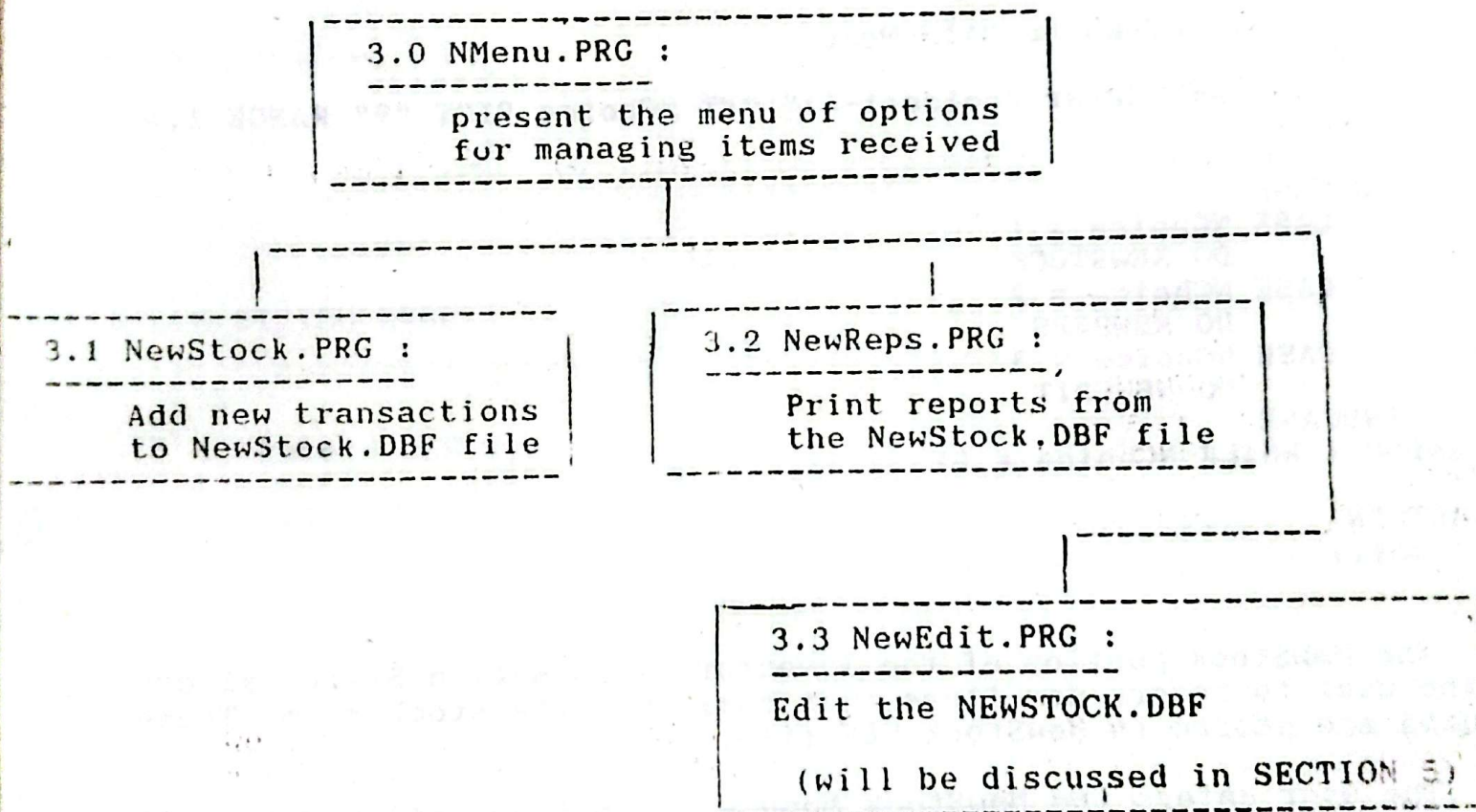
NMenu.PRG , which presents the opening menu

NewStock.PRG , which allows the user to add new data

NewReps.PRG , to print reports

NewEdit.PRG ., allows the user to edit existing data. This program will be discussed in SECTION 5 and will be labeled by 5.2.

The following figure shows this structure :



### 3.0 NMenu.PRG ( New Stock System Menu ) :

\* Program...: NMenu.PRG  
\* Date.....: March, 1989  
\* Notes.....: Menu for managing NEWSTOCK portion of the Inventory  
\*

SET CONFIRM ON

NChoice = 0

DO WHILE NChoice # 4

CLEAR

@ 2,5 SAY "NEW STOCK SYSTEM MENU"

@ 2,50 SAY DTOC(T\_DATE) + " " + TIME()

@ 3,5 SAY ULINE

?

?

TEXT

1. Record NEW Items

2. Print NEWSTOCK Reports

3. Edit NEWSTOCK Data

4. RETURN to MAIN Menu

ENDTEXT

@ 18,5 SAY "Enter choice(1-4)" GET NChoice PICT "9" RANGE 1,  
READ

\*

DO CASE

CASE NChoice = 1

DO NEWSTOCK

CASE NChoice = 2

DO NEWREPS

CASE NChoice = 3

DO NEWEDIT

ENDCASE

ENDDO ( WHILE NChoice # 4)

\*

RETURN

\* eof()

The NewStock portion of the Inventory Information System allows the user to record new items that come into the stock-room. These data are stored in NewStock.DBF file.

The user enters the NewStock Menu by selecting option 3 from the Inventory System Main Menu. This brings up the screen of the NEWSTOCK SYSTEM MENU shown on the next page.



Inventory Management System MAIN Menu

04/05/89

00:02:59

---

Would you like to :

---

1. Manage the MASTER Inventory File ,
2. Handle individual SALES transactions ,
3. Handle NEW STOCK transactions , or
4. Exit

Enter Your choice: 3

NEW STOCK SYSTEM MENU

04/05/89

00:03:33

---

1. Record NEW Items
2. Print NEWSTOCK Reports
3. Edit NEWSTOCK Data
4. RETURN to MAIN Menu

Enter choice(1-4) 1

3.1 NewStock.PRG ( NewStock Data-Entry ) :

```
* Program...: NEWSTOCK.PRG
* Date.....: March, 1989
* Notes.....: Data entry program for goods received.

SELECT A
USE MASTER INDEX MASTER
SELECT B
USE NEWSTOCK INDEX NEWSTOCK
SELECT A
+
PartNumb = "X"
DO WHILE PartNumb # " "
  @ 5,0 CLEAR
  PartNumb = SPACE (7)
  @ 10,5 SAY "Enter DATA for goods received"
  @ 12,5 SAY "Part number" GET PartNumb PICT "!!!!!!!"
  READ
  +
  IF PartNumb # " "
    SEEK PartNumb
    DO CASE
      CASE .NOT. FOUND()
        @ 12,25 SAY "NO SUCH PART!!!"
        ? CHR(7)
      CASE FOUND()
        @ 12,25 SAY TITLE
        +
        SELECT B      && NEWSTOCK.DBF
        +
        APPEND BLANK
        REPLACE PART_NO WITH PartNumb
        REPLACE DATE WITH T_Date
        REPLACE VENDOR WITH A->VENDOR
        @ 14,5 SAY "Quantity" GET QTY
        @ 14,25 SAY "Price" GET COST PICT "99999.99"
        @ 16,5 SAY "Date" GET DATE PICT "99/99/99"
        @ 16,22 SAY "Vendor" GET VENDOR
        READ
        +
        SELECT A      && MASTER.DBF
        +
        ENDCASE
      ENDIF (PartNumb # " ")
    ENDDO (WHILE PartNumb # " " )    && i.e., while User Does not ask
  CLOSE DATABASES
  RETURN
  + eof()
```



When the user selects option 1 from the New Stock System Menu, the screen clears and the following prompt appears :

Enter data for goods received  
Part number : \_

If a part number is valid, the screen will display the name of the part, the date, and the vendor ( which can be changed ,if necessary ), and prompts the user to fill in the quantity received and the purchase price, as follows :

NEW STOCK SYSTEM MENU

04/05/89

00:03:33

---

Enter DATA for goods received

Part number FP-4000 CASIO FP/4000

Quantity 20 Price 1500.00

Date 04/05/89 Vendor Cairo Trading

When the user has filled in the blank fields, the program asks for the next transaction. After adding the transactions, the user just presses the return key rather than typing another number. This brings back the New Stock System Menu.

NEW STOCK SYSTEM MENU

---

04/05/89

00:03:33

Enter DATA for goods received

Part number FP-4000



NEW STOCK SYSTEM MENU

04/05/89

00:03:33

---

Enter DATA for goods received

Part number FP-4000 CASIO FP/4000

Quantity Price

Date 04/05/89 Vendor Cairo Trading

### 3.2 NewReps.PRG ( NewStock System Reports ) ;

```
* Program...: NewReps.PRG
* Date.....: March, 1989
* Notes.....: Print Reports from the NEWSTOCK.DBF file.
*
SELECT 1
USE NEWSTOCK
SELECT 2
USE MASTER INDEX MASTER
*
SELECT 1
SET RELATION TO PART_NO INTO MASTER
*
RepChoice = 0
DO WHILE RepChoice # 3
  CLEAR
  @ 2, 5 SAY "NEW STOCK REPORT OPTIONS"
  @ 2,50 SAY DTOC(T_DATE) + " " + TIME()
  @ 3, 5 SAY Uline
  ?
  ?
  TEXT
    1. By Part number
    2. By Dates
    3. RETURN to NEWSTOCK Menu
  ENDTEXT
  @ 18,5 SAY "Enter choice(1-3) " GET RepChoice PICT "9"
  READ
  @ 5,0 CLEAR
  STORE " " TO YN,Printer
  IF RepChoice # 3
    @ 15,5 SAY "Send to Printer? "GET YN PICT "!"
    READ
    IF YN = "Y"
      Printer = "TO PRINT"
    ENDIF
  ENDIF
  @ 5,0 CLEAR
  DO CASE
    CASE RepChoice = 1
      SET INDEX TO NEWSTOCK
      NSearch = SPACE(7)
      @ 15,5 SAY "Look for what Part number?: ";
      GET NSearch PICT "!!!!!!!"
      READ
      CLEAR
      SEEK NSearch
      REPORT FORM NEWSTOCK WHILE PART_NO = NSearch &PRINTER
    *
  END CASE
```

```
*
CASE RepChoice = 2
  SET INDEX TO
  STORE SPACE(5) TO Start , End
  @ 15,5 SAY "Enter Start Date" GET Start PICT "99/99/99"
  @ 17,5 SAY "Enter End Date  " GET End PICT "99/99/99"
  READ
  Start = CTOD(Start)
  End = CTOD(End)
  CLEAR
  *
  REPORT FORM NEWSTOCK FOR DATE >= Start .AND. Date <= End ;
                                     &Printer
ENDCASE
IF YN # "Y" .AND. RepChoice # 3
  ?
  ?
  WAIT "Press any KEY to RETURN TO the REPORTS Menu..."
ENDIF
ENDDO (WHILE RepChoice # 3 ) && i.e.,While User does not request EXIT
SET RELATION TO
CLOSE DATABASES
RETURN
eof()
```

This program allows the user to review , in report form , all stock received by part number or for a range of dates.

These reports are useful for resolving differences with vendors or between the computer's report of stock and on what is actually on hand. Also, allows a store manager to review the number of orders for a particular item over a given period of time and adjust reorder points accordingly.



When the user decides to print reports, another menu of options appears:

1. By part number
2. By dates
3. Return to New Stock Menu

NEW STOCK REPORT OPTIONS

---

1. By Part number
2. By Dates
3. RETURN to NEWSTOCK Menu

Enter choice(1-3) 1

04/05/89 00:09:12

NEW STOCK SYSTEM MENU

---

1. Record NEW Items
2. Print NEWSTOCK Report
3. Edit NEWSTOCK Data
4. RETURN to MAIN Menu

Enter choice(1-4) 2

The following shows a report of items received by part number :

NEW STOCK REPORT OPTIONS

04/05/89

00:10:28

Look for what Part number?: FP-4000

Page No. 1  
04/05/89

Inventory Items Received

| Part No       | Part name     | Qty | Purch.-<br>Price | Date     | Vendor name   |
|---------------|---------------|-----|------------------|----------|---------------|
| FP-4000       | CASIO FP/4000 | 20  | 1500.00          | 04/05/89 | Cairo Trading |
| *** Total *** |               | 20  | 1500.00          |          |               |

The user can also view only transactions that occurred between two dates as follows :

### NEW STOCK REPORT OPTIONS

1. By Part number
2. By Dates
3. RETURN to NEWSTOCK Menu

Enter choice(1-3) 2

Enter Start Date 01/01/89  
Enter End Date 04/31/89

Page No. 1  
04/05/89

### Inventory Items Received

| Part No | Part name     | Qty | Purch. Price | Date     | Vendor name   |
|---------|---------------|-----|--------------|----------|---------------|
| FP-4000 | CASIO FP/4000 | 20  | 1500.00      | 04/05/89 | Cairo Trading |
| Total   |               | 20  | 1500.00      |          |               |



You can create a REPORT FORM file called NewStock.FRM by typing the following commands at the dBASE dot prompt :

```
. CLEAR ALL
. SELECT 1
. USE NEWSTOCK
. SELECT 2
. USE MASTER INDEX MASTER
. SELECT 1
. SET RELATION TO PART_NO INTO MASTER
. MODIFY REPORT NEWSTOCK
```

and filling out the questionnaire with the following information :

|                            |                          |
|----------------------------|--------------------------|
| Page title                 | Inventory Items Received |
| Page width (position)      | 80                       |
| Left margin                | 0                        |
| Right margin               | 0                        |
| Lines per page             | 58                       |
| Double space report        | No                       |
| Page eject before printing | Yes                      |
| Page eject after printing  | Yes                      |
| Plain page                 | No                       |

| Column | Contents | Heading        | Width | Decimals | Total? |
|--------|----------|----------------|-------|----------|--------|
| 1      | PART_NO  | Part No        | 7     |          |        |
| 2      | B->Title | Part name      | 25    |          |        |
| 3      | QTY      | Qty            | 4     | 0        | N      |
| 4      | COST     | Purchase price | 12    | 2        | N      |
| 5      | DATE     | Date           | 8     |          |        |
| 6      | VENDOR   | Vendor name    | 25    |          |        |

#### SECTION 4

### INVENTORY SYSTEM UPDATING

At this point, we have created all three major portions of the Inventory System. However, the final program, the one that gives the system its real power, remains to be written. This program can take data from both SALES.DBF and NEWSTOCK.DBF files and update the MASTER.DBF file based on those data. The system also needs to allow editing from the SALES.DBF and NewStock.DBF files after an update taken place. This is the purpose of this section.

#### Updater.PRG ( Updating the MASTER.DBF File ) :

The purpose of this program is to handle MASTER.DBF updates. The program needs to subtract the quantities in the SALES.DBF from the MASTER.DBF, since these items have been sold. Then it needs to add the quantities from the NEWSTOCK.DBF to those in the MASTER.DBF, since these items are now in stock.

Also, the updating program must subtract the quantities in the NEWSTOCK.DBF from the ON\_ORDER quantities in the MASTER.DBF, since these have now been received, and replace the purchase price of the items in the MASTER.DBF with the purchase price in the NEWSTOCK.DBF, since this is the most recent price.

Finally, the program should replace the dates in the MASTER.DBF with the dates in the transaction files, as means of keeping track of the last update occurred.



Program Algorithm :

- Clear the screen.
- Display message that MASTER.DBF is being updated from SALES.DBF.
- USE SALES INDEX SALES.
- COPY all non-updated records to a TEMPorary file, TEMP:
  - . USE TEMP
  - . Make sure there are records in TEMP.
  - . If there are records, then
    - . USE MASTER INDEX MASTER.
    - . Update from TEMP, by subtracting quantities and replacing dates.
    - . CLOSE DATABASES for saftey
    - . USE SALES
    - . Make all items in the POSTED field .TRUE.
    - . CLOSE DATABASES to free work areas.
- Display message that MASTER.DBF is being updated from NEWSTOCK.DBF.
- USE NEWSTOCK INDEX NEWSTOCK.
- COPY all non-updated records to TEMP file:
  - . USE TEMP
  - . Make sure there are records in TEMP.
  - . If there are records, then
    - . USE MASTER INDEX MASTER.
    - . Udate from TEMP, by adding quantities, replacing cost and date, and subtracting quantities from ON\_ORDER field.
    - . CLOSE DATABASES for saftey.
    - . USE NEWSTOCK.
    - . Make all items in the POSTED field .TRUE.
    - . CLOSE DATABASES to free areas.
- Return to MMenu.PRG.



Updater.PRГ :

```
-----
* Program...: Updater.PROG
* Date.....: March, 1989
* Notes....: Updates MASTER.DBF from SALES.DBF and NEWSTOCK.DBF
*
* Ask User if sure about UPDATING
*
YN = "Y"
@ 5,0 CLEAR
@ 15,5 SAY "UPDATE MASTER FILE FROM SALES AND NEWSTOCK? (Y/N)";
GET YN PICT "!"
READ
*
IF YN = "N"
    RETURN
ENDIF
@ 5,0 CLEAR
@ 12,5 SAY "WAIT -----> Updating from the SALES file"
*
USE SALES INDEX SALES
COPY STRUCTURE TO TEMP
COPY TO TEMP FOR .NOT. POSTED
*
SELECT 2
USE TEMP
IF RECCOUNT() > 0
    SELECT 1
    USE MASTER INDEX MASTER
    UPDATE ON PART_NO FROM TEMP;
    REPLACE QTY WITH (QTY-TEMP->QTY) , DATE WITH TEMP->DATE
    CLOSE DATABASES
    *
    USE SALES
    REPLACE ALL POSTED WITH .T.
ENDIF (RECCOUNT > 0)
*
CLOSE DATABASES
*
@12,5 SAY "WAIT -----> Updating from the NEWSTOCK file"
*
USE NEWSTOCK INDEX NEWSTOCK
COPY STRUCTURE TO TEMP
COPY TO TEMP FOR .NOT. POSTED
*
&& SET SAFTEY OFF is used in IMenu.PRГ
```

```
SELECT 2  
USE TEMP  
RECCOUNT() > 0  
SELECT 1  
USE MASTER INDEX MASTER  
UPDATE ON PART_NO FROM TEMP;  
    REPLACE QTY WITH (QTY + TEMP->QTY) , DATE WITH TEMP->DATE ,;  
    COST WITH TEMP->COST , ON_ORDER WITH (ON_ORDER - TEMP->QTY)  
CLOSE DATABASES  
*  
USE NEWSTOCK  
REPLACE ALL POSTED WITH .T.  
IF (RECCOUNT > 0)  
CLOSE DATABASES  
TURN  
eof()
```

## SECTION 5

### EDITING THE SALES.DBF and NEWSTOCK.DBF AFTER UPDATING PROCESS

#### 2.3 SalEdit.PRG ( Editing the Sales File ) :

The problem here is as follows:

After an updating process to the MASTER.DBF from the SALES.DBF, the user discovers, for example, that:

1. The user accidentally records 100 sales of PART\_NO x, and later he discovers it was only 10 ( not 100 ).
2. The user records a sale of 10 sales of PART\_NO x, but he discovers that they were PART\_NO y ( not x ).

Accordingly, when the user makes the correction to the SALES.DBF, the MASTER.DBF will be incorrect unless it too is changed.

The solution is to write a program (SalEdit.PRG) that allows the user to make any changes he wishes to the SALES.DBF and automatically makes the same corrections to the MASTER.DBF.

The idea of this program is as follows :

- Allow the user to change data for a RECORD in the SALES.DBF file
- If the user changed a PART\_NO, add the QTY to the QTY for the old PART\_NO (the one updated incorrectly), and subtract it from the QTY for the new PART\_NO in the MASTER.DBF (the one corrected).
- If the user changed a QTY in the SALES.DBF, find the difference between the old QTY (100 in the example above) and the new QTY (10) and subtract that difference (90) from the QTY in the MASTER.DBF.
- If the user changed the DATE, it could also be changed in the MASTER.DBF, but it is not necessary to do so. The DATE will still reflect the date of the last update, rather than the date of the edit.



Program Algorithm :

```
. Set up memory variables
. Set up loop for invoice numbers
Get INVOICE_NO for data to edit
If user does not request exit,
  USE SALES INDEX SALES
  Count RECORDs with that INVOICE_NO
  IF INVOICE_NO .NOT. FOUND()
    Warn user and allow another try
  IF several RECORDs have that INVOICE_NO
    Display them and GET Rec.# from the user
  Otherwise, just go to the RECORD
  Locate RECORD and store original values to memory variables
  Display data on edit screen and allow edit
  After editing, adjust MASTER.DBF
    If RECORD POSTED but now deleted
      Mark RECORD for deletion
      Locate PART_NO in MASTER.DBF
      Increment QTY in SALES.DBF
    If RECORD POSTED and PART_NO changed
      Locate old PART_NO in MASTER.DBF
      Add old QTY back into MASTER.DBF
      Find new PART_NO in MASTER.DBF
      Subtract new QTY from MASTER.DBF
    If RECORD POSTED and QTY changed
      Calculate difference between old and new QTYies
      Find PART_NO in MASTER.DBF
      Subtract difference from MASTER.dbf's QTY
  Continue loop to allow user more edits
  If RECORDs deleted, PACK SALES.DBF
  Return to SMenu.PRG.
```

```
* Program...: SalEdit.PRG
* Date.....: March, 1989
* Notes....: Edit the SALES.DBF and Update MASTER.DBF
* Remark....: Called from SMenu.PRG (Sales Menu)
```

```
CLEAR
@15,5 SAY "Take care..Do not use MODIFY COMMAND with this program"
WAIT
CLEAR
```

```
No_Dels = 0
Search = 1
```

```
* Loop for Invoice numbers....
```

```
DO WHILE Search # 0
```

```
  CLEAR
```

```
  @ 2, 5 SAY "Edit Sales Transactions"
```

```
  @ 2,50 SAY DTOC(T_Date) + " " + TIME()
```

```
  @ 3, 5 SAY Uline
```

```
  ?
```

```
  ?
```

```
  @15, 5 SAY "Enter Invoice number to edit (0 if none):";
                                     GET Search PICT "9999"
```

```
  READ
```

```
  * If the user does not request EXIT, continue with Edit.
```

```
  IF Search > 0
```

```
    USE SALES INDEX SALES
```

```
    * count RECORDs with that Invoice number
```

```
    COUNT FOR INVOICE_NO = Search TO HowMany
```

```
    DO CASE
```

```
      CASE HowMany = 0    && Invoice not found, so warn user.
        @ 20, 5 SAY "No such Invoice number !!"
        ? CHR(7)
```

```
      CASE HowMany > 0    && Invoice number found, so proceed.
```

```
        * If several RECORDs have that Invoice number,
        * display them and get correct record number.
```

```
        IF HowMany > 1
```

```
          @ 5, 0 CLEAR
```

```
          ? " "
```

```
          ?? " RecNo    Part #  QTY    Price Salesperson"
          ?                                     Customer"
```

```
LIST FOR INVOICE_NO = Search;
PART_NO , QTY , PRICE , SALSPERSON , CUSTOMER , DATE
?
INPUT "Edit which RECORD: " TO RecNO
GOTO RecNO
;
; Otherwise, just go to the RECORD
;
ELSE
LOCATE FOR INVOICE_NO = Search
ENDIF
;
; Store original field values to variables.
;
Old_PART = PART_NO
Old_QTY = QTY
;
; Display data on edit screen and allow edit.
;
@ 5, 0 CLEAR
Deleted = " "
@ 7, 5 SAY "Invoice number:" GET INVOICE_NO
@ 7,50 SAY "Date " GET DATE PICT "99/99/99"
@ 10, 5 SAY "Part number:" GET PART_NO PICT "!!!!!!!"
@ 12, 5 SAY "Salesperson:" GET SALSPERSON
@ 12,45 SAY "Customer:" GET CUSTOMER
@ 15, 5 SAY "Quantity:" GET QTY
@ 15,30 SAY "Selling Price:" GET PRICE
@ 23, 5 SAY "Delete this RECORD? (Y/N) ";
GET Deleted PICT "!"
READ
;
; After editing, adjust MASTER.DBF.
;
DO CASE
; If transaction to be deleted,delete it and
; add its QTY back to the MASTER.DBF.
;
CASE POSTED .AND. Deleted = "Y"
DELETE
No_Dels = No_Dels + 1
;
USE MASTER INDEX MASTER
SEEK Old_PART
IF FOUND()
REPLACE QTY WITH QTY + Old_QTY
ENDIF
```



```
*
* If Part number changed, add QTY to old PART_NO,
* and subtract from new PART_NO.
*
CASE POSTED .AND. PART_NO # Old_PART
  New_QTY = QTY
  New_PART = PART_NO
  *
  USE MASTER INDEX MASTER
  SEEK Old_PART
  IF FOUND()
    REPLACE QTY WITH QTY + Old_QTY
  ENDIF
  *
  SEEK New_PART
  IF FOUND()
    REPLACE QTY WITH QTY - New_QTY
  ENDIF
*
* If QTY changed, adjust the MASTER>DBF QTY.
*
CASE POSTED .AND. QTY # Old_QTY
  Diff = QTY - Old_QTY
  *
  USE MASTER INDEX MASTER
  SEEK Old_PART
  *
  IF FOUND()
    REPLACE QTY WITH QTY - Diff
  ENDIF
  ENDCASE
  ENDCASE ( HowMany > 0)
  ENDIF (Search > 0)
ENDDO ( Search # 0)
*
* If RECORDs deleted, PACK the SALES.DBF.
*
IF No_Dels > 0
  @ 5,0 CLEAR
  ? "Deleting unwanted RECORDs from the SALES.DBF...."
  *
  USE SALES INDEX SALES
  PACK
ENDIF
*
CLOSE DATABASES
RETURN
* eof()
```

When the user chooses to edit the SALES.DBF file, the screen clears and displays the prompt :

Enter invoice number to edit ( 0 in none ):

as shown as follows:

Sales System Menu

---

1. Enter Point-of-Sale routine
2. Print Sales Reports
3. Edit Sales data
4. Return to Main Menu

Enter choice(1-4) 3

Edit Sales Transactions

---

04/05/89

02:08:01

---

Enter Invoice number to edit (0 if none): 1

. When the user enters the invoice number, the program displays all the transactions on that invoice; for example :

Edit Sales Transactions

04/05/89

02:08:01

| RecNo         | Part #  | QTY | Price   | Salesperson | Customer    |
|---------------|---------|-----|---------|-------------|-------------|
| 1<br>04/03/89 | AT-4000 | 5   | 7000.00 | Eng. Hosam  | Dr. Abdalla |
| 2<br>04/03/89 | AT-4000 | 1   | 7000.00 | Eng. Hosam  | Dr. Abdalla |

Edit which RECORD:



Once the user selects a single transaction, the program displays the data for that transaction on the custom screen for editing or deleting as follows :

The user can change whatever data are incorrect or remove the transaction.

The program then handles all aspects of adjusting the MASTER.DBF, if necessary, and allows the user to edit more sales transactions.

Edit Sales Transactions

04/05/89

02:08:01

| RecNo    | Part #  | QTY | Price   | Salesperson | Customer    |
|----------|---------|-----|---------|-------------|-------------|
| 1        | AT-4000 | 5   | 7000.00 | Eng. Hosam  | Dr. Abdalla |
| 04/03/89 |         |     |         |             |             |
| 2        | AT-4000 | 1   | 7000.00 | Eng. Hosam  | Dr. Abdalla |
| 04/03/89 |         |     |         |             |             |

lit which RECORD: 1

Edit Sales Transactions

04/05/89

02:08:01

Invoice number: 1

Date 04/03/89

Part number: AT-4000

Salesperson: Eng. Hosam

Customer: Dr. Abdalla

Quantity: 5

Selling Price: 7000.00

Delete this RECORD? (Y/N) N

Edit Sales Transactions

04/05/89

02:10:36

| RecNo    | Part #  | QTY | Price   | Salesperson | Customer    |
|----------|---------|-----|---------|-------------|-------------|
| 1        | AT-4000 | 5   | 7000.00 | Eng. Hosam  | Dr. Abdalla |
| 04/03/89 |         |     |         |             |             |
| 2        | AT-4000 | 1   | 7000.00 | Eng. Hosam  | Dr. Abdalla |
| 04/03/89 |         |     |         |             |             |

Edit which RECORD: 2

Edit Sales Transactions

04/05/89

02:10:36

Invoice number:

1

Date 04/03/89

Part number: AT-4000

Salesperson: Eng. Hosam

Customer: Dr. Abdalla

Quantity: 1

Selling Price: 7000.00

Delete this RECORD? (Y/N) N

Edit Sales Transactions

04/05/89

02:11:41

| RecNo         | Part #  | QTY | Price   | Salesperson      | Customer          |
|---------------|---------|-----|---------|------------------|-------------------|
| 4<br>04/05/89 | AT-3000 | 2   | 6000.00 | Eng. Magdi Maher | Dr. M. Y. Youssef |
| 3<br>04/05/89 | AT-4000 | 5   | 7000.00 | Eng. Magdi Maher | Dr. M. Y. Youssef |

it which RECORD: 3

Edit Sales Transactions

04/05/89

02:11:41

Invoice number: 2

Date. 04/05/89

Part number: AT-4000

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

Quantity: 5

Selling Price: 7000.00

Delete this RECORD? (Y/N) N



Edit Sales Transactions

04/05/89

02:13:14

| RecNo    | Part #  | QTY | Price   | Salesperson      | Customer          |
|----------|---------|-----|---------|------------------|-------------------|
| 4        | AT-3000 | 2   | 6000.00 | Eng. Magdi Maher | Dr. M. Y. Youssef |
| 04/05/89 |         |     |         |                  |                   |
| 3        | AT 4000 | 5   | 7000.00 | Eng. Magdi Maher | Dr. M. Y. Youssef |
| 04/05/89 |         |     |         |                  |                   |

Edit which RECORD: 4

Edit Sales Transactions

04/05/89

02:13:14

Invoice number: 2

Date 04/05/89

Part number: AT-3000

Salesperson: Eng. Magdi Maher

Customer: Dr. M. Y. Youssef

Quantity: 2 Selling Price: 6000.00

Delete this RECORD? (Y/N) N

### 3.3 NewEdit.PRG ( Editing the NewStock File ) ;

This program handles edits to the NEWSTOCK.DBF. It looks very similar to the SalEdit.PRG, except that it asks the user for the PART\_NO, rather than the INVOICE\_NO, of the transaction to edit.

```
* Program...: NewEdit.PRG
* Date.....: March, 1989
* Notes.....: Edit the NEWSTOCK.DBF and update MASTER.DBF.
* Remark....: Called from NMenu.PRG.

CLEAR
@14,10 SAY "NewEdit.PRG....."
@15,10 SAY "Take care...Dot use MODIFY COMMAND with this program....."
@17, 5
wait
CLEAR

* Memory variables..
*
*
No_Dels = 0
Search = "1"
HowMany = 0

* Get PART_NO for data to edit..
*
DO WHILE Search # "0"
*
USE NEWSTOCK INDEX NEWSTOCK
*
CLEAR
@ 2, 5 SAY "Edit NEWSTOCK Transactions"
@ 2,50 SAY DTOC(T_DATE) +" " + TIME()
@ 3, 5 SAY Uline
*
@ 15, 5
ACCEPT "      Enter Part number to EDIT (0 if none) " TO Search
Search = UPPER(Search)
*
* If user did not request EXIT, continue with edit.
*
IF Search # "0"
*
* Count the RECORDs with that PART_NO.
*
SEEK Search
COUNT WHILE PART_NO = Search TO HowMany
```

DO CASE

\*  
\* IF PART\_NO not found, warn user.  
\*

CASE HowMany = 0  
? " No such Part number !!"  
? CHR(7)

WAIT

CASE HowMany > 0

IF HowMany > 1

\* If several RECORDs have that PART\_NO,  
\* display them and get correct RECORD number

CLEAR

SEEK Search

LIST WHILE PART\_NO = Search;

PART\_NO, QTY, COST, DATE, VENDOR

?

INPUT " Edit which RECORD (Enter Record number)?";  
TO RecNumb

GOTO RecNumb

ELSE

\*  
\* If only one RECORD in the NEWSTOCK.DBF has the  
\* requested PART\_NO, that RECORD is located and its  
\* Record Number is stored to the variable RecNumb.  
\*

SEEK Search

RecNumb = RECNO()

ENDIF (HowMany > 1)

Old\_PART = PART\_NO

Old\_QTY = QTY

Old\_COST = COST

CLEAR

Deleted = " "

@ 2, 5 SAY "Edit NEWSTOCK Transaction..."

@ 5, 5 SAY "Part number:" GET PART\_NO PICT "@!"

@ 7, 5 SAY " Quantity:" GET QTY

@ 7,26 SAY "Purchase price:" GET COST

@ 9, 5 SAY " Date:" GET DATE PICT "99/99/99"

@ 9,26 SAY " Vendor:" GET VENDOR

@15, 5 SAY "DELETE THIS RECORD? (Y/N) "GET Deleted PICT "!"

READ

\* ----- After editing, adjust MASTER.DBF

\* First, handle change in COST



```
IF COST # Old_COST
  New_PART = PART_NO
  New_COST = COST
  *
  USE MASTER INDEX MASTER
  SEEK New_PART
  *
  IF FOUND()
    REPLACE COST WITH New_COST
    *
    USE NEWSTOCK INDEX NEWSTOCK
    GOTO RecNumb
  ENDIF
ENDIF (COST # Old_COST)

DO CASE
  *
  * If NEWSTOCK transaction to be deleted,
  * subtract its QTY from the MASTER.DBF,
  * and add it to the ON_ORDER field.
  *
  CASE POSTED .AND. Deleted = "Y"
    DELETE
    No_Dels = No_Dels + 1
    *
    USE MASTER INDEX MASTER
    SEEK Old_PART
    *
    IF FOUND()
      REPLACE QTY WITH QTY - Old_QTY
      REPLACE ON_ORDER WITH ON_ORDER + Old_QTY
    ENDIF
    *
    * If PART_NO changed, subtract the QTY from
    * the old PART_NO, and add to the new.
    * Do the opposite for the ON_ORDER field.
    *
    CASE POSTED .AND. PART_NO # Old_PART
      New_QTY = QTY
      New_PART = PART_NO
      *
      USE MASTER INDEX MASTER
      SEEK Old_PART
      *
      IF FOUND()
        REPLACE QTY WITH QTY - Old_QTY
        REPLACE ON_ORDER WITH ON_ORDER + Old_QTY
      ENDIF
      *
```

```
SEEK New_PART
IF FOUND()
    REPLACE QTY WITH QTY + New_QTY
    REPLACE ON_ORDER WITH ON_ORDER - New_QTY
ENDIF
```

```
*
* If user just changed QTY, adjust the MASTER.DBF's QTY.
*
```

```
CASE POSTED .AND. QTY # Old_QTY
    Diff = QTY - Old_QTY
```

```
*
    USE MASTER INDEX MASTER
    SEEK Old_PART
```

```
*
    IF FOUND()
        REPLACE QTY WITH QTY + Diff
        REPLACE ON_ORDER WITH ON_ORDER - Diff
    ENDIF
```

```
ENDCASE
```

```
ENDCASE
```

```
ENDIF (Search # 0)
```

```
ENDDO (WHILE Search # 0)
```

```
IF No_Dels > 0
```

```
    CLEAR
```

```
    ? "Deleting unwanted RECORDs from the NEWSTOCK.DBF "
```

```
    *
```

```
    USE NEWSTOCK INDEX NEWSTOCK
```

```
    PACK
```

```
ENDIF
```

```
*
```

```
CLOSE DATABASES
```

```
RETURN
```

```
* eof()
```

NEW STOCK SYSTEM MENU

04/05/89

00:49:40

---

1. Record NEW Items
2. Print NEWSTOCK Reports
3. Edit NEWSTOCK Data
4. RETURN to MAIN Menu

Enter choice(1-4) 3



Edit NEWSTOCK Transactions

04/05/89

00:59:06

Enter Part number to EDIT (0 if none) FP-4000

Edit NEWSTOCK Transaction...

Part number: FP-4000

Quantity: 20 Purchase price: 1500.00

Date: 04/05/89 Vendor: Cairo Trading

DELETE THIS RECORD? (Y/N)

REFERENCES

1. Alan Simpson,  
Advanced Techniques in dBASE III Plus  
San francisco . Paris , Dusseldorf . London
2. Ashton-Tate,  
dBASE III Plus
3. Abdalla El-Daoushy,  
First course in dataBase Analysis and Programming with  
dBASE III Plus, Memo. no. 880, 1988  
Institute of National Planning, Cairo, Egypt
4. Abdalla El-Daoushy  
INP dataBase Management System ,Memo no. 1475, 1988  
Institute of National Planning, Cairo, Egypt
5. C. J. Date,  
An Introduction to Database Systems, Third Edition  
Addison-Wesley Publishing Company  
Reading,Massachusetts . Amsterdam . London  
Manila . Singapore . Sydney . Tokyo
6. James Martin,  
Principles of Data-Base Management  
Printic-Hall, Inc . Englewood Cliffs . New Jersey



ان الحياة الاقتصادية لم تشهد تقدما ملحوظا فى علم التخزين الا فى الربع قرن الاخير .  
والتخزين هو الوظيفة التى تقوم بالحفاظ على السلع والبضائع (أو تخزينها ) خلال الفترة  
ما بين إنتاجها أو توريدها والوقت التى تطلب فيه تلبية للحاجة اليها .

ولما كانت فلسفة الانتاج بكميات كبيرة للاستفادة من مزايا الانتاج الكبير ( تخفيض تكلفة  
الوحدة ) وتخزين هذه الكميات الضخمة فى مراكز التخزين هى السائدة فى الوقت الحالى  
ولما كان الاتجاه العام للمعروض من البضائع أن يتعادل غالبا مع حجم الطلب عليها خاصة  
فى الاحوال الاقتصادية العادية ، ولما كان من المألوف لدى المستهلك توقع تسليم ما  
يطلبه من بضائع أو سلع فى أسرع وقت ممكن ، أصبح موعد هذا التسليم أداة أساسية من أدوات  
التسويق .

ولقد استخدم أسلوب ضغط فترة التسليم بدلا من سياسة تخفيض أسعار السلع وذلك  
بغية جذب العملاء الى هذه السلع ومن ثم أصبح هذا المنهج فى أعمال التسويق —  
الاسباب الجوهرية التى دعت الى التوسع السريع فى مجالات التخزين .

والمبحث الحالى من خلال عمل نظام المعلومات الخاصة بالمخزون وقابليته للتطبيق مباشرة  
فى مجال مراقبة ومتابعة المخزون ، يعتبر من العوامل المساعدة فى حل المشاكل التالية :

#### ١ - مراقبة المخزون :

- خطة مراقبة المخزون
- مسئولية مراقب المخزون
- جرد المخازن
- تسعير الاصناف المخزونة



— ٨٤ —

٢ — مشكلات الاجراءات ونظام العمل بالمخازن

٣ — مشكلات المخزون الراكد

٤ — مشكلات الرقابة على الاعمال المخزنية

٥ — مشكلات الموارد البشرية العاملة بالمخازن

