NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

THÈSES CANADIENNES

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilming. Nous avons tout fait pour assurer une qualité supérieure de réproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.
A FORM MANIPULATION SYSTEM FOR RELATIONAL DATABASES

by

Christina To

A thesis
presented to the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Master of Science
in
Systems Science
Department of Computer Science

© Christina To, Ottawa, Canada, 1987.
Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.
SUMMARY

A lot of activities in an office involve the processing of forms. One way to automate an office is therefore to computerize its processes for form manipulation.

Our major work is the design and development of a computerized form manipulation system FMS-UO and its integration with a relational database management system. FMS-UO can be used in a small office or a home environment, where the computer resources are very limited. It runs under the Disk Operating System DOS 3.3 in an APPLE II computer having at least 64K RAM and two disk drives. Through a friendly user-interface, FMS-UO provides the user with the facilities of designing the formats for forms, inputting data into the forms, storing data for forms in diskettes and outputting them onto the screen or the printer. To enhance the capabilities of FMS-UO, we integrate it with an inexpensive commercial relational database management system, RQL. The codes of RQL are modified and a data file is set up for transferring data between RQL and FMS-UO.

In this thesis, we first describe the basics of electronic forms and their operations. We then review some relevant works in the literature (Chapters 1 and 2). Description of FMS-UO from the user's point of view and the implementational details of the system are included in Chapters 3 and 4. An example of creating a letter by means of the integrated systems is provided. The example shows how a letter can be generated for different clients (Chapter 5).
ACKNOWLEDGEMENTS

The author wishes to express her deepest gratitude to her thesis supervisor Dr. T.Y. Cheung for his constant advice, valuable suggestions and guidance. He always found time for productive discussions and encouragement to her work.

She also would like to thank the help of all her professors and colleagues. The invaluable help of the university staff, especially those of the Department of Computer Science, is also appreciated.

The financial supports from the University of Ottawa and Natural Sciences and Engineering Research Council of Canada are acknowledged with gratitude.

Lastly, but not the least, the author is indebted to the very special supports she received from her parents Mr. and Mrs. Y. S. To and her husband Mr. Wayne Woo.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUMMARY</td>
<td>iv</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>v</td>
</tr>
<tr>
<td><strong>Chapter</strong></td>
<td></td>
</tr>
<tr>
<td><strong>1. ELECTRONIC FORMS</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Fundamentals of form manipulation systems</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1 Elements of electronic forms</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Basic operations on forms</td>
<td>6</td>
</tr>
<tr>
<td><strong>2. REVIEWS ON FORM MANIPULATION SYSTEMS</strong></td>
<td>9</td>
</tr>
<tr>
<td>2.1 Business Definition Language</td>
<td>9</td>
</tr>
<tr>
<td>2.2 System for Business Automation (SBA)</td>
<td>10</td>
</tr>
<tr>
<td>2.3 OfficeTalk-Zero and OfficeTalk-D</td>
<td>12</td>
</tr>
<tr>
<td>2.4 Toronto Latest Acronym (TLA)</td>
<td>13</td>
</tr>
<tr>
<td>2.5 DBASE III</td>
<td>16</td>
</tr>
<tr>
<td>2.6 Comparison of functions among the form manipulation systems</td>
<td>18</td>
</tr>
<tr>
<td><strong>3. A USER'S PERSPECTIVE OF THE FORM MANIPULATION SYSTEM FMS-UO</strong></td>
<td>19</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>19</td>
</tr>
<tr>
<td>3.2 The operations of FMS-UO</td>
<td>20</td>
</tr>
<tr>
<td>3.2.1 Create a frame</td>
<td>23</td>
</tr>
<tr>
<td>3.2.2 Modify/Delete block coordinates</td>
<td>25</td>
</tr>
<tr>
<td>3.2.3 Create/Modify/Delete data from a form</td>
<td>27</td>
</tr>
<tr>
<td>3.2.4 Display/Print data using a frame</td>
<td>28</td>
</tr>
<tr>
<td>3.2.5 Display frame layout</td>
<td>29</td>
</tr>
<tr>
<td>3.2.6 Quit</td>
<td>29</td>
</tr>
<tr>
<td>3.3 RGL</td>
<td>31</td>
</tr>
<tr>
<td>3.4 Interface between FMS-UO and RGL</td>
<td>31</td>
</tr>
<tr>
<td><strong>4. DESIGN AND IMPLEMENTATIONAL DETAILS OF THE FORM MANIPULATION SYSTEM FMS-UO</strong></td>
<td>34</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>34</td>
</tr>
<tr>
<td>4.2 The functions of FMS-UO</td>
<td>35</td>
</tr>
<tr>
<td>4.2.1 Create a frame</td>
<td>36</td>
</tr>
<tr>
<td>4.2.2 Modify/Delete block coordinates</td>
<td>37</td>
</tr>
<tr>
<td>4.2.3 Create/Modify/Delete data from a form</td>
<td>40</td>
</tr>
<tr>
<td>4.2.4 Display/Print data using a frame</td>
<td>43</td>
</tr>
<tr>
<td>4.2.5 Display frame layout</td>
<td>44</td>
</tr>
<tr>
<td>4.2.6 Quit</td>
<td>45</td>
</tr>
<tr>
<td>4.3 System integration of FMS-UO and RGL</td>
<td>45</td>
</tr>
<tr>
<td><strong>5. AN EXAMPLE AND SOME CONCLUDING REMARKS</strong></td>
<td>49</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>49</td>
</tr>
</tbody>
</table>
5.2 The process of generating similar letters - an example ........................................... 49
5.3 Generating letters with same contents but different clients ....................................... 55
5.4 Outputting the letter by a different format ................................................................. 55
5.5 Some concluding remarks ............................................................................................... 57

APPENDIX A ERROR MESSAGES FOR FMS-UO ................................................................. 58
APPENDIX B THE PROGRAM OF FMS-UO (TH2) ............................................................. 60
REFERENCES ...................................................................................................................... 85

List of Figures:
Figure 1.1 Comparison of system requirements of DB III and FMS-UO ................................. 3
Figure 1.2 A form instance is presented to different output media through different templates ... 6
Figure 1.3 Basic operations on form instances and templates ............................................... 6
Figure 2.1 An example of data retrieval by QBE ................................................................. 11
Figure 2.2 A typical SBA program ................................................................................... 12
Figure 2.3 A precondition sketch with constraints in TLA ................................................... 14
Figure 2.4 An action sketch in TLA .................................................................................. 14
Figure 2.5 An origin pseudoform sketch in TLA ................................................................. 15
Figure 2.6 Creation of database INVENTORY by DBASE III .................................................. 16
Figure 2.7 Comparison of functions of different form manipulation systems ...................... 18
Figure 3.1 The MAIN menu of FMS-UO ........................................................................... 20
Figure 3.2 A form instance displayed through the frame above it ........................................ 20
Figure 3.3 The same form instance displayed through another frame .................................... 21
Figure 3.4 FMS-UO prompts for a new frame name ........................................................... 21
Figure 3.5 The user has the opportunity to view the frame .................................................. 22
Figure 3.6 Coordinates of the top left cgrner of a block are defined ...................................... 22
Figure 3.7 Coordinates of the bottom right corner of a block are defined ............................... 22
Figure 3.8 The user may redefine the block ....................................................................... 23
Figure 3.9 The menu - MODIFY BLOCK COORDINATES ............................................ 23
Figure 3.10 The prompts displayed when option 1 or 2 or 3 is selected ................................. 24
Figure 3.11 The message displayed after deletion of Block X .............................................. 24
Figure 3.12 The system prompts the user to confirm his intention of deleting the whole frame ... 25
Figure 3.13 The menu - CREATE/MODIFY/DELETE DATA IN A FORM ............................... 25
CHAPTER 1
ELECTRONIC FORMS

1.1 INTRODUCTION

Most of the data processing activities in an office are related to the preparation, distribution, relocation, filing, retrieval and checking of documents, such as reports, letters, minutes, memos, etc (DESO81). These documents are manipulated essentially in the format of forms (GEHA81). A form can be considered as a structured set of slots containing textual or graphical data. These slots of data are functionally related and can each be treated as a unit for operations.

It is more efficient to handle computerized forms than paper forms. Forms, if computerized, reduce drastically their storage space requirement. They can be stored, retrieved, processed and transmitted at an electronic speed. For example, it will be much faster to test data conformance with constraints and do numerical calculation in a computerized system (BAIR78) (BURN77) (COCH80).

In the past, many form manipulation systems with primitive capabilities have been built for facilitating individual office procedures. The current trend of system development gears towards integrating these tools into a single system. Office services, such as typing, copying and telephone communications which were provided by different means in the past, can now be supported by a stand-alone computerized office system. This integration reduces the complexity of human interfaces with different office machines. Integration can also greatly enhance
the capabilities of many single form manipulation systems. The facilities of an isolated form manipulation system are usually limited to a few functions, such as inputting of data from the console, printing of the forms, etc. If the system is integrated with other systems designed particularly for data manipulation, such as database or file management systems, additional services can be obtained. In particular, recent developments tend to integrate office automation systems with relational database management systems.

In this thesis, we first design and develop a form manipulation system (FMS-UO) and then integrate it with an existing relational database management system called RQL. Most of the existing systems which can process forms require quite sophisticated hardware or software facilities. FMS-UO, however, aims at users in a home environment with a limited amount of computing resources. To illustrate this point, Figure 1.1 compares the well-known database management system DB. III with our FMS-UO in terms of hardware and software requirements, target users and the ease of use.
The rest of this chapter describes the elements of electronic forms and their operations. The second chapter reviews some relevant works in the literature. Some form manipulation systems are described. In the third chapter, the functions and operations of FMS-UO are presented. It also briefly describes the system RQL and its integration with FMS-UO. Chapter 4 presents the details of implementing FMS-UO, such as its system architecture, the data structures used for its text files, the processes for system integration and data retrieval, etc. Lastly, an example of creating business letters using the integrated system described in Chapter 5. This example also illustrates the integration of the two systems.

1.2 FUNDAMENTALS OF FORM MANIPULATION SYSTEMS

In Section 1.2.1 the different elements of forms, such as form fields, form instances, form templates, etc., will be
described from a user's point of view. Section 1.2.2 stipulates the different operations on forms, such as creation, insertion, deletion, modification, storing, copying and outputting. Forms may also be routed or traced in a network environment (MAZE83).

1.2.1 ELEMENTS OF ELECTRONIC FORMS

Hereafter, "electronic forms" are simply called "forms". In the following, we explain the three basic elements of forms.

(A) Fields

A form is composed of various fields which a user fills with relevant information. For example, a letter, the most commonly used form in the office, may have five fields: letter heading, date, name of the receiver, contents of the letter and name of the sender.

A field is a unit of information for processing and with independent characteristics. Certain constraints that govern the characteristics, processing order and sources of data may be imposed upon the fields. Data entered into or removed from them must be checked for consistency and conformance with these constraints. Some examples follow:

a. Restrictions on insertion

For some fields, data can be entered only at the time when a form is created. Other fields allow delayed data entry but cannot be changed afterwards.

b. Automatic insertion

For certain fields, data are automatically entered as the consequence of a specific computation or operation.
c. Mandatory insertion

Entry of data into some fields may be mandatory when other related fields are filled. For instance, in a form containing employee information, once the value "married" is entered into the field MARITAL STATUS, the system will require the fields SPOUSE'S NAME and NUMBER OF CHILDREN to be filled with appropriate data as well.

d. Order of insertion

The order of entering data into a number of fields may be mandatory in certain applications. For instance, the field SIGNATURE, which indicates the approval or authentication of the form at different administrative levels, must be filled by the managers in the order of increasing responsibility.

e. Restricted access

A field can be made accessible or inaccessible to certain users. Some systems require a user to enter a password before accessing a read-only, or a write-only field.

(B) Form Instances

A collection of values for the fields at a certain moment constitutes an instance of the form. A form instance may be stored in various kinds of media.

(C) Templates

A template is a structural skeleton through which form instances are presented to the user. The same form instance may be presented via different templates to different output media (see Figure 1.2). For example, in a display system, a form instance is mapped onto a text template in order to create a textual output. Similarly, if a graphic template is used to
display a form instance, a picture is created. In more sophisticated systems, a voice message may be obtained via a speech template.

A special kind of text template, called FRAME, is developed in this thesis for displaying form instances. It will be described in Chapter 3.

---

**FORM INSTANCE**

---

<table>
<thead>
<tr>
<th>text</th>
<th>graphics</th>
<th>speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>template</td>
<td>template</td>
<td>template</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>textual</th>
<th>two dimensional</th>
<th>voice</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>picture</td>
<td>message</td>
</tr>
</tbody>
</table>

---

Figure 1.2 A form instance is presented to different output media through different templates

1.2.2 BASIC OPERATIONS ON FORMS

The following table lists some of the basic operations on form instances and form templates.

---

<table>
<thead>
<tr>
<th>OPERATIONS ON</th>
<th>OPERATIONS ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORM INSTANCES</td>
<td>FORM TEMPLATES</td>
</tr>
<tr>
<td>CREATION</td>
<td>CREATION</td>
</tr>
<tr>
<td>MODIFICATION</td>
<td>MODIFICATION</td>
</tr>
<tr>
<td>DELETION</td>
<td>DELETION</td>
</tr>
<tr>
<td>STORING</td>
<td>STORING</td>
</tr>
<tr>
<td>COPYING</td>
<td>COPYING</td>
</tr>
<tr>
<td>OUTPUTTING</td>
<td></td>
</tr>
<tr>
<td>ROUTING</td>
<td></td>
</tr>
<tr>
<td>TRACING</td>
<td></td>
</tr>
</tbody>
</table>

---

Figure 1.3 Basic operations on form instances and templates.
(A) OPERATIONS ON FORM INSTANCES

a. Creation/Modification of form instances

Data can be entered into a form through the keyboard or from other file systems. The data must conform with the specified restrictions.

b. Storing/Deletion of form instances

Form instances can be stored for future applications. The method of storing is system and implementation dependent. In some systems, for example, form instances are stored as tuples of a relational database (FERR82). Form instances can be removed from the text file only if they are not protected against deletion.

c. Copying form instances

Copying reproduces an exact copy of the original. One important aspect in the manipulation of forms, as often required in legal or business transactions, is to distinguish the duplicated copies from their original. To achieve this, copies are usually each given a copy number. In an office environment, keeping the consistency of data among the different copies and the original is an important management issue.

d. Outputting form instances (RABI81)

As described in Section 1.2.3, form instances can be outputted in various formats by using different templates. Hence, the same form instance may appear as a textual image, or as a two dimensional graphic image, or as a voice message.

e. Routing/Tracing form instances (LADD80)

In an environment where there are many workstations, form
Instances may have to be routed from one workstation to another. A trace command provides the processing history of a form instance. For example, in a network environment where forms are processed through different workstations, the route along which the form flows through the workstations can be traced. In TLA (TSIC82), a method is developed for determining whether the form is circulating infinitely in a loop within the network.

(B) OPERATIONS ON FORM TEMPLATES

a. Creation of form templates

Users can specify the characteristics of templates, such as their data domains, the positions and display formats of the data, etc.

b. Modification of form templates

The characteristics of existing templates can be changed by different operations.

c. Storing/Deletion of form templates

In general, templates to be used for a set of form instances can be stored for future applications, but they can also be deleted from a system unless they are protected by the user.

d. Copying form templates

Reproducing form templates with similar characteristics as an existing template can be performed easily by copying the existing templates.
CHAPTER 2
REVIEWS ON SOME FORM MANIPULATION SYSTEMS

As there exist numerous form manipulation systems, it is impossible to give an exhaustive review on all of them. Instead, we review only some of them with the objective of illustrating the useful features of an ideal system. These systems are:

- Business Definition Language (BDL)
- System for Business Automation (SBA)
- OfficeTalk-Zero & OfficeTalk-D
- Toronto Latest Acronym (TLA)
- DBASE III

Readers are referred to <FERR82>, <SHU81>, and <EMBL80> for more complete reviews.

Form has been used quite loosely in the literature as a generic name to mean form instance and template.

2.1 BUSINESS DEFINITION LANGUAGE (BDL) <KRUS74> <FURU82>

BDL is a form-based office automation language developed by IBM. Its special feature is the built-in structure that enables the user to build programs without performing any translation of his interpretation of the problem to the language available. This is achieved through the three components of the language: the Form Definition Component (FDC), the Document Flow Component (DFC), and the Document Transformation Component (DTC).

(A) Form Definition Component

In FDC, a user defines the forms using an interactive graphic system. The following characteristics of a form can be defined:

- name
- width
- height
preprinted information
properties of the forms, such as data type definitions, data formats, etc.

(B) Document Flow Component

This component is used to define the basic structure of an application by describing the office and its flow of documents. The BDL user uses an interactive language for drawing data flow graphs whose nodes correspond to objects, such as departments, sections, people, or functions within an office. Each arc between two nodes is labelled with the title of the document that flows between them. Actions will be carried out when their triggering conditions are satisfied (HOWE75).

(C) Document Transformation Component

This component expresses the precise functional relationships among the documents of the system. It is here that the actual computations are specified. DTC also describes how output documents are constructed.

2.2 SYSTEM FOR BUSINESS AUTOMATION (SBA) (ZLO077) (LU081)

In this system, information is expressed in tables, business forms, reports, etc. The language supported by the system has three components:

- the language (QBE) Query-By-Example for database management
- a programming language for writing application programs
- a business system specification language

In Query-By-Example (QBE) (ZLO075) (ZLO077) (ZLO081), the basic data objects are tables (relations). In a uniform way, the user is provided with the skeleton of a table for querying, updating, defining or maintaining a database. The user formulates his query by filling in a table which is associated with examples
of possible answers in some of the fields. For example, if the user wants to find the red items from his stocks, he specifies the query as shown in Figure 2.1.

```
TYPE
ITEM : COLOR : SIZE
P.PEN : RED
```

Figure 2.1 An example of data retrieval by QBE

When the user enters 'TYPE' as the name of the requested table, the system automatically generates the headings: ITEM, COLOR, and SIZE. In order to print all the items in the stock that have a red color, he enters RED below the heading COLOR and P.PEN below the heading ITEM as an example to the system. P.PEN here stands for "printing an item such as a PEN".

An SBA program is composed of a set of applications of Query-By-Example on those data represented in tables or more complex data types, such as forms, charts, reports, etc. Its significant departure from the language Query-By-Example is its inclusion of the INPUT and OUTPUT sections used for establishing a means of communications with other application programs. A typical SBA program is shown in Figure 2.2.
<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>NAME</th>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPPLIER</td>
<td>REPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>DEPT</td>
<td>SUPPLIER</td>
<td></td>
</tr>
<tr>
<td>PARKER</td>
<td>TOY</td>
<td>PARKER</td>
<td></td>
</tr>
<tr>
<td>SALES</td>
<td>SUPPLY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT</td>
<td>ITEM</td>
<td>SUPPLIER</td>
<td></td>
</tr>
<tr>
<td>TOY</td>
<td>ROD</td>
<td>PARKER</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.2 A typical SBA program

2.3 OFFICETALK-ZERO and OFFICETALK-D 〈ELLI80A〉 〈ELLI80B〉

OfficeTalk-Zero was developed at Xerox Palo Research Centre (PARC) in 1976. Its most attractive feature is the abundance of tools to be utilized by the users. OfficeTalk-Zero is integrated with many other Xerox systems, such as:

- text editors
- graphic packages
- communications facilities
- file management facilities
- forms data entry facilities

To implement a particular OfficeTalk application, a tailored set of blank forms must be designed and entered into the
database. OfficeTalk-Zero provides a form editor which allows one to specify the graphical design of a form and the style of each of the fields of the form. The editor restricts the data types to match the form's field definitions. The forms can be transmitted to another workstation and traced by the user. OfficeTalk-D (ELLI80) was another experimental office information system developed at PARC. As an extension to OfficeTalk-Zero, the D in OfficeTalk-D refers to the emphasis of this experiment on the system aspects of distributed control and cooperation among a group of users. The central element is an entity-relationship database. An information control net model (ELLI79) is used to provide a global graphical description of all the activities and tasks currently in progress.

2.4 TORONTO LATEST ACRONYM (TLA) (TSIC79)

TLA is an extension of the electronic Office Form System (OFS) (TSIC80A) (TSIC80B). Both TLA and OFS were developed at the University of Toronto for the PDP-11 series of mini-computers and the LSI-11 micro-computers (CHEU80). TLA is integrated with a relational database management system called MRS. The query-oriented language SLQ is used to specify form procedures in terms of form sketches. Form sketches are images of forms in which user can input his requests for the operations. It is through these form sketches that the form procedures are modified. Three different form sketches and their functions are briefly described below.

(A) Precondition Sketches

Using a precondition sketch, the user can find a form
instance whose values satisfy the specific constraints. Figure 2.3 shows an example of a precondition sketch. The underlined portion against the field NAME indicates that this field of the requested form instance must contain the characters "U OF Q". The symbol "?" indicates a field name is being specified. The field DOWN must contain a value less than the value in the field TOTAL. The field ACCOUNT must match the field ACCOUNT of another form called LIMIT.

```
CREDIT REQUEST

NAME U OF Q
ACCOUNT ? LIMIT ACCOUNT
TOTAL < 10000 OLD LIMIT
DOWN < ? TOTAL NEW LIMIT
DUE
APPROVED .
```

Figure 2.3 A precondition sketch with constraints in TLA

(B) Action Sketches

A user can specify how values can be assigned to the fields by specifying procedures in three ways:

a. A constant value is inserted into a field. For example, in Figure 2.4, the value "YES" is inserted into the field APPROVED when the procedure is executed.

```
CREDIT REQUEST

NAME
ACCOUNT
TOTAL OLD LIMIT ? LIMIT CREDIT
DOWN NEW LIMIT LIMIT CREDIT
DUE # EXPR # TOTAL = ? DOWN
APPROVED YES
```

Figure 2.4 An action sketch in TLA
b. A value is copied from a field of another sketch. The requested sketch and field are entered into the field in the format "?SKETCH.FIELD". As illustrated in Figure 2.4, the value in the field CREDIT of another sketch LIMIT is copied into the field OLD LIMIT of this sketch.

c. A function assigns a value to a field. A function call is indicated by an initial "#" followed by the function's name and its arguments. This is demonstrated in the field DUE in Figure 2.4. The function is EXPR, with arguments TOTAL, - and DOWN.

(C) Pseudoform Sketches

A pseudoform sketch is used to specify the actions, or preconditions which cannot be expressed in the fields of the forms. For example, in the case where there is no field on the form for indicating its origin, a pseudoform sketch can be used by the procedure to specify that only those forms from certain workstations can be processed. The pseudoform sketch shown in Figure 2.5 indicates only those forms from the finance department are accepted for processing. Similarly, a pseudoform sketch can be used to define the dispersement information of the forms such as the destination, the copies shipped, etc.

```
ORIGIN PSEUDO-SKETCH

NOT : .......
STATION :

FINANCE DEPARTMENT

```

Figure 2.5 An origin pseudoform sketch in TLA
2.5 DBASE III

DBASE III is a relational database management system to be used in microcomputers. Four main functions are supported by the system:

- database manipulation
- querying
- report/form generation
- procedure creation

(A) Database manipulation

DBASE III can be used to create and modify databases. By using the CREATE instruction as shown in Figure 2.6, the database INVENTORY is set up and its field names, data types, field widths and data formats are defined.

New data records can either be appended at the end or inserted in the middle of the database file.

```
: B:INVENTORY
: field name type width dec
: ---------------------------
: 1 LIQUOR Char/text 10
: 2 BRAND Char/text 20
: 3 SIZE Char/text 7
: 4 QUANTITY Numeric 3
: 5 COST Numeric 6 2
: 6 PRICE Numeric 6 2
```

Figure 2.6 Creation of database INVENTORY by DBASE III

Changing databases in DBASE III can be categorized into 3 areas:

- changing the database structure, e.g., adding and deleting attributes, data type of attributes, etc.
- adding and deleting records
- changing the contents of records
(B) Querying

The user can issue requests for information to DBASE III through its query language called Application Development Language (ADL). ADL has procedural and non-procedural commands. With the procedural commands, the user instructs the system to produce the desired results step by step. With non-procedural commands, the user can state the macro-type commands. For example, the user may try to determine the total quantity of a liquor called BOURBON by issuing the following ADL command:

```
SUM QUANTITY FOR LIQUOR = 'BOURBON'
```

Other non-procedural commands such as DISPLAY, LIST, COUNT, etc., are also available.

(C) Report/Form Generation

The REPORT command can be used to produce reports/forms. These forms are of rather rigid format and have a heading and several columns of data. When the REPORT command is initiated, the system will allow the page heading of the report to be inputted on a screen and the attribute name and heading of each column of the report on another screen.

(D) Procedure Creation

DBASE III uses procedures to extend its data processing capabilities. For example, the user can use a procedure to produce a function selection menu for applications, such as database update. It will display the different possible update choices to the user and prompt for a selection. Procedures can be used to produce special reports with formats specified by the user which cannot be obtained by using the REPORT command in the normal way.
2.6 COMPARISON OF FUNCTIONS AMONG THE FORM MANIPULATION SYSTEMS

The following table summarizes the important functions that are supported by the different form manipulation systems.

<table>
<thead>
<tr>
<th>FUNCTIONS</th>
<th>BDL</th>
<th>SBA</th>
<th>OFFICETALK-D</th>
<th>TLA</th>
<th>DB III</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORM INSTANCE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CREATION</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>MODIFICATION</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>DELETION</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>STORING</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>COPYING</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OUTPUTTING</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ROUTING</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRACING</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>FORM TEMPLATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATION</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>MODIFICATION</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DELETION</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>STORING</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>COPYING</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OUTPUTTING</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>ROUTING</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACING</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEFINE CHARACTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OF FIELDS</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISTRIBUTED DB</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>QUERY FACILITY</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM SPECIFICAT.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.7 Comparison of functions of different form manipulation systems
CHAPTER 3

A USER'S PERSPECTIVE OF THE FORM MANIPULATION SYSTEM FMS-UO

3.1 INTRODUCTION

In this chapter, the form manipulation system FMS-UO we designed and implemented at the University of Ottawa is described from the user's point of view. With the limitation of time and resources, this thesis emphasizes only on the design and the implementation of some basic functions of a form manipulation system and its interface with a relational database management system. FMS-UO aims at users in a home environment using very little computing resources. It runs under the Disk Operating System DOS 3.3 in an APPLE II computer having at least 64K RAM and two disk drives.

The operations of FMS-UO are enhanced because of its integration with the commercially available relational database management system RQL. Data retrieved from RQL can be inserted into the forms in FMS-UO. A subset of the basic operations on forms described in Section 1.2.2 is implemented. Operations of FMS-UO are described in details in Section 3.2. The system RQL is described in Section 3.3. Their integration is explained in Section 3.4. Error messages for the different operations of FMS-UO are included in Appendix A.

3.2 THE OPERATIONS OF FMS-UO

The main operations of FMS-UO are displayed as five options in the MAIN menu (see Figure 3.1) when the system is initiated or after any operation has been completed. These options will be
explained in the following sub-sections. When information is required from the user during the form operations, the system will prompt the user with a message starting with "FMS>".

```
FMS> INDICATE YOUR CHOICE BY ENTERING THE NUMBER

1. CREATE A FRAME
2. MODIFY/DELETE BLOCK COORDINATES
3. CREATE/MODIFY/DELETE DATA FROM A FORM
4. DISPLAY/PRINT DATA USING A FRAME
5. DISPLAY FRAME LAYOUT
6. QUIT
```

Figure 3.1 The MAIN menu of FMS-UO

### 3.2.1 CREATE A FRAME

A special kind of text template, called FRAME, is used in FMS-UO for displaying or outputting form instances. A frame is composed of rectangular areas called BLOCKS. Figures 3.2 and 3.3 show how the same form instance appears differently through different frames.

```

\[ \text{Figure 3.2 A form instance displayed through the frame above it} \]
```
NOTICE: THE FOLLOWING EMPLOYEES ARE TO BE TRANSFERRED TO TORONTO

JOHN SMITH
1234

DAVID PATTEN
3641

Figure 3.3 The same form instance displayed through another frame.

A frame is created by specifying the layout of its blocks. As shown in Figure 3.4, the user is first prompted to enter a name for the new frame. If the name is already used for another frame, the user will be prompted for another one. After a valid name has been entered, the user enters the coordinates of its blocks. At this point, he is given the opportunity to have a view of the frame. (See Figure 3.5.)

CREATE A FRAME

FMS> PLEASE GIVE A NEW FRAME NAME:

Figure 3.4 FMS-UO prompts for a new frame name
Figure 3.5 The user has the opportunity to view the frame.

If requested, the frame is displayed with the X and Y scales along the top and the left side of the screen. (Details of this operation are explained in Section 3.2.5.)

Next, the system will prompt for the X and Y coordinates of the top left corner of the block. (See Figure 3.6.) The user is given the option to see the frame before defining the coordinates of the bottom right corner of the block. (See Figure 3.7.)

Figure 3.6 Coordinates of the top left corner of a block are defined.

Figure 3.7 Coordinates of the bottom right corner of a block are defined.

The system checks the validity of the data. The user is required to re-enter the data if the X coordinate is not within
the range of 1 to 40 or the Y coordinate is not within the range of 1 to 22 or the blocks are overlapping. Before completion, the system prompts for any alterations, as shown in Figure 3.8. If the response is 'Y', the whole block has to be re-defined. Otherwise, the user proceeds to define another block.

---

FMS> DO YOU WANT TO MAKE ANY CHANGES FOR THIS BLOCK? (Y OR N):

---

Figure 3.8 The user may redefine the block

3.2.2 MODIFY/DELETE BLOCK COORDINATES

The coordinates of an existing block can be modified using Option 2 of the MAIN menu. First, the system will prompt for the frame name. If the input matches one of the existing frame names, the menu MODIFY BLOCK COORDINATES will be displayed, as shown in Figure 3.9. When Option 1 or 2 or 3 is selected, the existing block numbers are displayed and the user can choose to see the frame before modifying the block number. (See Figure 3.10.)

---

MODIFY/DELETE BLOCK COORDINATES

FMS> PLEASE SPECIFY FRAME NAME: STUDENT

FMS> PLEASE INDICATE YOUR CHOICE BY ENTERING THE NUMBER

1. MODIFY AN EXISTING BLOCK
2. ADD A NEW BLOCK
3. DELETE AN EXISTING BLOCK
4. DELETE THE WHOLE FRAME

---

Figure 3.9 The menu - MODIFY BLOCK COORDINATES
FMS> BLOCKS DEFINED SO FAR ARE:
   1 2 3

FMS> DO YOU WANT TO SEE THE FRAME
   FIRST? (Y OR N) : N

FMS> PLEASE SPECIFY THE BLOCK NUMBER
   TO BE MODIFIED:

Figure 3.10 The prompts displayed when option 1 or 2 or 3 is selected.

The following paragraphs explain the options in the menu —

MODIFY BLOCK COORDINATES.

Option 1 - MODIFY AN EXISTING BLOCK

The block number specified by the user for modification is
checked against those already defined. If the block exists, the
coordinates of the block can be re-defined following the same
procedure as described in the function - CREATE A FRAME. (See
Section 3.2.1.)

Option 2 - ADD A NEW BLOCK

The user has to input a unique number for the new block. The
block is defined by following the procedure as described in the
function - CREATE A FRAME. (See Section 3.2.1.)

Option 3 - DELETE AN EXISTING BLOCK

A block can be removed from a frame by this option. The
block number X specified by the user is checked against those
already defined. If it exists, the block will be deleted and the
following message will be displayed:

FMS> BLOCK NUMBER X HAS BEEN DELETED

Figure 3.11 The message displayed after deletion of Block X
Option 4 - DELETE THE WHOLE FRAME

If this option is selected, the user is asked to confirm his intention by the following message.

FMS> YOU HAVE CHOSEN TO DELETE THE WHOLE FRAME, DO YOU STILL WANT TO GO AHEAD? (Y OR N)

Figure 3.12 The system prompts the user to confirm his intention of deleting the whole frame.

3.2.3 CREATE/MODIFY/DELETE DATA FROM A FORM

When this option is selected from the MAIN menu, the user can create, modify or delete data in each of the fields in the form. After the system prompts for a form name, the menu CREATE/MODIFY/DELETE DATA IN A FORM is presented. (See Figure 3.13.)

CREATE/MODIFY/DELETE DATA FROM A FORM

FMS> PLEASE SPECIFY THE FORM NAME : CLASS

FMS> PLEASE INDICATE YOUR CHOICE BY ENTERING THE NUMBER

1. CREATE/MODIFY DATA IN A FIELD
2. DELETE DATA FROM ONE FIELD
3. DELETE DATA FROM ALL FIELDS

Figure 3.13 The menu - CREATE/MODIFY/DELETE DATA IN A FORM

Option 1 - CREATE/MODIFY DATA IN A FIELD

The user is prompted for the number of the field whose data he wants to create or modify. (See Figure 3.14.) If the specified field already contains data, the user is given the
Figure 3.14 The system prompts for the field number choice of modifying it or leaving it as it is. Two ways are available for inserting data into a field, as shown in Figure 3.15. If "C" is entered, the system will prompt for the entry of data through the console. (See Figure 3.16.) Otherwise, data will come from RQL.

Figure 3.15 Two methods for inputting data

Figure 3.16 The system prompts for data input through the console

Option 2 - DELETE DATA FROM ONE FIELD

In this option, as shown in Figure 3.14, the user is also prompted for the field number whose data is to be deleted. After the user enters the number, the data is removed from the field and the messages in Figure 3.17 are displayed.

Figure 3.17 Messages displayed after data has been deleted from a field
3.2.4 DISPLAY/PRINT DATA USING A FRAME

The data of a form can be displayed or printed (See Figure 3.18) through frames with different layouts. The system then prompts for the names of the form and the frame. (See Figures 3.19 and 3.20.) The user may output part or the entire form by answering the question shown in Figure 3.21. If a

```
DISPLAY/PRINT DATA USING A FRAME

FMS> PLEASE INDICATE YOUR CHOICE BY ENTERING THE NUMBER

1. DISPLAY DATA ON THE SCREEN
2. PRINT DATA ON THE PRINTER
```

Figure 3.18 The user is given a choice of methods for outputting data

```
FMS> PLEASE SPECIFY THE FORM NAME :
```

Figure 3.19 The system prompts for the form name

```
FMS> PLEASE SPECIFY THE FRAME NAME :
```

Figure 3.20 The system prompts for the frame name

```
FMS> HOW MANY FIELDS DO YOU WANT TO OUTPUT? (O MEANS ALL):
```

Figure 3.21 The system prompts for the output options

non-zero number is input, the system will prompt for the numbers of the specified fields before outputting the data. (See Figure
3.2.5 DISPLAY FRAME LAYOUT

This function displays the layout of the blocks of a frame. It can be initiated under 3 of the 5 options of the MAIN menu. (See Figure 3.1.):

- Option 1 CREATE A FRAME
- Option 2 MODIFY/DELETE BLOCK COORDINATES
- Option 5 DISPLAY FRAME LAYOUT

Scales are displayed along the top and left edges of the screen. A block with dotted perimeters is displayed as shown in Figure 3.23. The block number is displayed inside the top left corner of the block. In the course of defining a block (Option 1 or 2 of the MAIN menu), if only its top left corner has been defined, the coordinate is represented by an asterisk "*". Display of the frame can be terminated by pressing any key.
3.2.6 QUIT

This option allows the user to leave FMS-UO and return to DOS.

3.3 RQL

In order to fully utilize FMS-UO, a user should also be familiar with RQL. Following is a summary of its main operations. See <CHEN83> for more details. RQL is a relational database management system developed by HELLO Software Ltd. It operates under DOS 3.3 in APPLE II micro-computers with at least 48K RAM and two floppy disk drives. The user manipulates the databases interactively by using a query language. RQL offers some useful data processing capabilities such as searching, sorting and merging of data.

In the system, a relation is described in terms of a table. Each database in RQL is composed of at least one table (see Figure 3.24). There can be more than one database in RQL.

<table>
<thead>
<tr>
<th>STUDENT NAME</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEVE MORGAN</td>
<td>102468</td>
</tr>
<tr>
<td>SUSAN GRAND</td>
<td>103357</td>
</tr>
<tr>
<td>BILL LAYTON</td>
<td>124420</td>
</tr>
<tr>
<td>STEVE ZOOM</td>
<td>124789</td>
</tr>
</tbody>
</table>

Figure 3.24 The student names and numbers are stored in the table "CLASSA" of the database "STUDENT".

Five operations are available for database manipulations:

a. Define a new database
b. Access an existing database
c. Display or print the names of all tables in a database
d. Destroy a database
e. Rename a database
Eleven operations are available for table manipulations:

a. Define a new table within an existing database
b. Manipulate (insert, delete, update) the rows of a table
c. Create a new table from an existing table
d. Create a new table by joining two existing tables
e. Display or print the format of a table
f. Display or print the content of a table
g. Destroy a table
h. Sort a table on a particular column
i. Compute aggregate functions: Average, Count, Sum, Minimum, Maximum
j. Rename a column of a table
k. Rename a table

Most commands begin with a key word, such as "DEFINE" or "DISPLAY", followed by the key word "DATABASE" or "TABLE" and then the actual name of the database or the table. We shall give an example of storing the names and numbers of students in a new table "CLASSA" within a new database "STUDENT". The command "DEFINE DB STUDENT" is used to define the database. The name and description of the database have to be specified. After the database is defined, we can create a table within this database by using the command "DEFINE TABLE CLASSA". In this command, the name, width, data type and description of each attribute are defined. Data can then be inserted into the table using the command "INSERT CLASSA".

To retrieve data, the user has to specify the name of the database and the table in which the data is stored. For example, to retrieve the names of the students whose number is greater than "12345", the following commands have to be used:

    USE DB STUDENT
    DISPLAY TABLE CLASSA ON NAME WHERE NUMBER > "12345"

The first command is for specifying the database "STUDENT". The second command is to display from the table "CLASSA" the names of
the students whose number is greater than "12345". The content of "CLASSA" is shown in Figure 3.24 and the names of those students who satisfy the specified condition are shown in Figure 3.25.

---

NAME

BILL LAYTON
STEVE ZOOM

---

Figure 3.25 Names of students whose number is greater than "12345"

3.4 INTERFACE BETWEEN FMS-UO AND RQL

The data management capability of a form manipulation system such as FMS-UO is quite limited. It can be greatly enhanced if integrated with a relational database management system such as RQL. The user can manipulate data stored in the databases before passing it on to the forms prepared in FMS-UO.

The user can switch between the two systems at three operational points:

(A) After booting the computer

The system prompts the user for a choice between FMS-UO and RQL. (See Figure 3.26.) If "F" is selected, the MAIN menu of FMS-UO is displayed. Otherwise, RQL is executed.

---

DO YOU WANT TO RUN FMS-UO OR RQL?
(F OR R):

---

Figure 3.26 The user may initiate FMS-UO or RQL.
(B) After completion of an RQL command

The user can continue operating under RQL or switch to FMS-UO after having completed a RQL command by responding to the question in Figure 3.27.

```
DO YOU WANT TO STAY IN RQL OR SWITCH TO FMS-UO ? (R OR F) :
```

Figure 3.27 The user may switch systems after completing an RQL command

(C) When creating or modifying data in Option 3 of MAIN menu under FMS-UO

After the user has specified the form name and the field number, the question in Figure 3.28 is displayed. If "R" is

```
FMS DO YOU WANT TO INPUT DATA FROM CONSOLE OR FROM RQL ? (C OR R)
```

Figure 3.28 Choice of the source of input data

selected, the RQL system is executed and a message is displayed (see Figure 3.29).

```
** NOTE : IF YOU HAVE A TABLE WHOSE DATA YOU WANT TO MOVE TO FMS, USE THE RQL COMMAND TO DISPLAY THAT TABLE, OTHERWISE, DEFINE YOUR TABLE NOW.
```

Figure 3.29 The message displayed after RQL is selected as the source of input data

Data transfer from RQL

The data has to be collected in a single table in order to be transferred to FMS-UO. The operations JOIN and DISPLAY are particularly useful for this purpose. When the operation DISPLAY
is used to display a table, the user is given the option of transferring the data in the table to FMS-U0. (See Figure 3.30.) If the answer is "Y", the data is moved to the field of the form in FMS-U0 and FMS-U0 will be invoked. If the answer is "N", the user continues with other RGL operations.

: DO YOU WANT TO MOVE THIS DATA TO
: THE FORM ? (Y OR N):

Figure 3.30 The user has the option of moving the data to a form
CHAPTER 4

DESIGN AND IMPLEMENTATIONAL DETAILS
OF THE FORM MANIPULATION SYSTEM FMS-UO

4.1 INTRODUCTION

This chapter presents the details of designing and implementing FMS-UO. This includes its integration with RQL. An overview of the functional hierarchy of the system is first depicted in Figure 4.1, together with the hardware and software requirements. In Section 4.2, the functions of FMS-UO are described in details. The record layouts of the files are explained wherever they are used. In Section 4.3, the system integration of FMS-UO and RQL is described.

---------------------------------------------

FMS-UO

---------------------------------------------

: FRAME : DELETE : MODIFY/ : PRINT : FRAME : U:
: BLOCK : DELETE : DATA : LAYOUT : I:
:COORDINATES : DATA FROM : USING A : USING : T:
: A FORM : FRAMES : :

---------------------------------------------

Figure 4.1 Functional hierarchy of FMS-UO

Hardware and software requirements of FMS-UO

(A) Hardware requirements

Computer : APPLE II
RAM : at least 64K
DISK DRIVE : two
(B) Software requirements

The Disk Operating System DOS 3.3 and the following programs are needed in order to run FMS-UO.

- **a. HELLO** - greeting program of FMS-UO
- **b. APPLESOFT** - language program of APPLE II
- **c. FPBASIC** - language program of APPLE II
- **d. RDB.OPT.HALF** - RQL program
- **e. SORT** - RQL program
- **f. SET.OPT** - RQL program
- **g. TH2** - FMS-UQ program

The programs of the systems RQL and FMS-UO are both written in the language APPLESOFT BASIC. The program TH2 is composed of a control block and many subroutines. The control block contains a section in which functions are selected. If a certain function is selected, the system branches to the subroutines for handling that function. After processing the function, control is returned to the control block.

Random access and sequential files are used to store information of the forms and frames. The file commands, OPEN, WRITE, READ and CLOSE, are used for both kinds of files. Other file commands, POSITION and APPEND, are used only for sequential files.

4.2 THE FUNCTIONS OF FMS-UO

The functions listed in Figure 4.1 are described in the following subsections according to the following aspects:

- the purpose of the function;
- how the function is initiated;
- the input files used in the function;
- the output files used in the function;
- coding reference (Refer to Appendix B for actual codes); and
- the procedures (steps) used for achieving its purpose.
4.2.1 CREATE A FRAME

Purpose : To define the coordinates of the blocks in a new frame.

Initiated by : Option 1 of the Main menu

Input file : TFRAME

Output files : TFRAME, framename.COR

Code reference : Lines 400 to 910 in TH2

Procedures :

a. If the frame name specified by the user is not found in the file TFRAME, a new frame name has to be re-entered.

b. Before defining the coordinates, user can choose to look at the frame. If some blocks have already been defined, the coordinates are read from the file framename.COR for displaying the frame.

c. After the coordinates of a block are defined, they are checked against the coordinates of the existing blocks in the file framename.COR. If any blocks overlap, go to step b.

d. If modification is not needed for the block which has just been defined, go to step f.

e. User can re-define the coordinates of the block by going to step b or return to the MAIN menu.

f. The coordinates are stored with a new record number in the file framename.COR. The total number of blocks defined and the block numbers are updated in the first record of the file.

g. The new framename is stored in the file TFRAME and the total number of frame defined is incremented by one in the first record of the file.

Figures 4.2 and 4.3 show the file layouts of framename.COR and TFRAME.
4.2.2 MODIFY/DELETE BLOCK COORDINATES

This function consists of four sub-functions as depicted in Figure 4.4.
MODIFY/DELETE BLOCK COORDINATES

MODIFY AN EXISTING BLOCK

Purpose

- To update the coordinates of an existing block.

Initiated by

- Option 2 of the MAIN menu and Option 1 of the sub-menu MODIFY/DELETE BLOCK COORDINATES

Input files

- TFRAME, framename.COR

Output files

- TFRAME, framename.COR

Code reference

- Lines 900-1950 in TH2

Procedures:

a. If the frame name specified by the user is found among the frame name in the file TFRAME, the block numbers in the first record of the file framename.COR are read. These block numbers are displayed with a message "FMS> BLOCKS DEFINED SO FAR ARE : ".

b. If the block number specified by the user matches one of the existing blocks, the coordinates of the block are read from the file framename.COR.

c. The coordinates of the block are re-defined by the same procedure as in the function - CREATE A FRAME.

ADD A NEW BLOCK

Purpose

- To add a block to a frame

Initiated by

- Option 2 of the MAIN menu and Option 2 of the sub-menu MODIFY/DELETE BLOCK COORDINATES

Input files

- TFRAME, framename.COR
Output files : TFRAME, framename.COR

Code reference : Lines 930 to 1950 in TH2

Procedures:

a. If the frame name specified by the user is found among the ones stored in the file TFRAME, the block numbers stored in the first record of the file framename.COR are read. These block numbers are displayed with a message:

"FMS > BLOCKS DEFINED SO FAR ARE : "

b. If the block number specified by the user does not match any of the existing ones, the coordinates of the block can be defined.

(C) DELETE AN EXISTING BLOCK

Purpose : To remove a block from a frame

Initiated by : Option 2 of the MAIN menu and Option 3 of the menu MODIFY/DELETE BLOCK COORDINATES

Input files : TFRAME, framename.COR

Output files : TFRAME, framename.COR

Code reference : Lines 930 to 1950 in TH2

Procedures:

a. If the frame name specified by the user is among the ones stored in the file TFRAME, the system will prompt for a block number.

b. If the input block number is found in the file framename.COR, the block is deleted.

c. The total number of blocks defined is reduced by one and the block number is removed from the first record of the file framename.COR.

(D) DELETE THE WHOLE FRAME

Purpose : To remove a frame from the system

Initiated by : Option 1 of the MAIN menu and Option 4 of the sub-menu MODIFY/DELETE BLOCK COORDINATES

Input files : TFRAME, framename.COR
Output file : TFRAME

Code reference : Lines 930 to 1950 in TH2

Procedures :

a. If the frame name specified by the user is among the ones in the file TFRAME, the system confirms the deletion process by prompting the user.

b. The frame name is deleted from the first record of TFRAME and the total number of frames defined is reduced by one. The file framename.COR is deleted.

4.2.3 CREATE/MODIFY/DELETE DATA FROM A FORM

This function consists of three sub-functions depicted in Figure 4.5.

CREATE/MODIFY/DELETE DATA FROM A FORM

.................

: CREATE/MODIFY : : DELETE DATA : : DELETE DATA :
: DATA IN A FIELD: : FROM ONE FIELD: : FROM ALL FIELDS:

--------------

Figure 4.5 Sub-functions of CREATE/MODIFY/DELETE DATA FROM A FORM

(A) CREATE/MODIFY DATA IN A FIELD

Purpose : To add or modify data in a field

Initiated by : Option 3 of the MAIN menu and Option 1 of the sub-menu CREATE/MODIFY/DELETE DATA IN A FORM

Input files : TFORM, formname.DAT, PASS

Output files : TFORM, formname.DAT, PASS

Code reference : Lines 1960 to 3550 in TH2

Procedures :

a. If the form name specified by the user is among the ones stored in the file TFORM then go to step d.

b. Initialize a data file with a name formname.DAT.

c. The form name is stored into the file TFORM. The total
The number of forms created is incremented by one.

The system prompts for the field number in which data is to be input.

The data flag in the file formname.DAT is read to determine whether the field contains any data.

If no data exists, go to step j, otherwise the user has a choice to modify the data.

If the user chooses to modify the data then go to step j.

The system prompts the user whether to modify other fields.

If the answer is "Y", the system prompts for a field number. Otherwise, the system returns to the MAIN menu.

User has a choice of inputting data from the console or the RQL.

If console is chosen, user inputs data through the keyboard. Otherwise, the system RQL is invoked. (Refer to Section 4.3 for details.) The input data is stored in a file formname.DAT.

The file layouts of the TFORM, formname.DAT, PASS are shown in Figures 4.6, 4.7 and 4.8.

---

**Filename** : TFORM  
**Function** : stores names of forms  
**File type** : random access  
**Record type** : 20 bytes

---

**RECORDS**

<table>
<thead>
<tr>
<th>Field #</th>
<th># in record</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>total number of forms created</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>form name #1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>form name #2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>form name #3</td>
</tr>
</tbody>
</table>

*Figure 4.6 Layout of file TFORM*
Filename : formname.DAT
Function : stores control information & data of form
File type : sequential

Field: Position

<table>
<thead>
<tr>
<th>#</th>
<th>from</th>
<th>length</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>Number of non-empty fields</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>Initial location of new data in the file</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>A flag for field #1 (empty or non-empty)</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>1</td>
<td>A flag for field #2</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>1</td>
<td>A flag for field #10</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>1</td>
<td>Number of string variables for field #1</td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>1</td>
<td>Number of string variables for field #2</td>
</tr>
<tr>
<td>22</td>
<td>25</td>
<td>1</td>
<td>Number of string variables for field #10</td>
</tr>
<tr>
<td>23</td>
<td>26</td>
<td>3</td>
<td>Initial location of field #1</td>
</tr>
<tr>
<td>24</td>
<td>29</td>
<td>3</td>
<td>Initial location of field #2</td>
</tr>
<tr>
<td>32</td>
<td>53</td>
<td>3</td>
<td>Initial location of field #10</td>
</tr>
<tr>
<td>33</td>
<td>56</td>
<td>length: Data of field #1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.7 Layout of file formname.DAT.

Filename : PASS
Function : stores information for data transfer
File type : random access
Record type : 300 bytes

RECORDS

<table>
<thead>
<tr>
<th>Field #</th>
<th>in record</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>A flag - data transfer request in FMS-UO</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>A form name in which data is stored</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>A field # in which data is stored</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>A flag - data from RQL is transferred</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>The # of string variables required</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Data from RQL</td>
</tr>
</tbody>
</table>

Figure 4.8 Layout of file PASS
(B) DELETE DATA FROM ONE FIELD

Purpose : To remove data from one field

Initiated by : Option 3 of the MAIN menu and Option 2 of the sub-menu CREATE/MODIFY/DELETE DATA IN A FIELD.

Input files : TFORM, formname.DAT

Output files : TFORM, formname.DAT

Code reference : Lines 1960-3550 in TH2

Procedures :

a. If the form name specified by the user is found among the ones in the file TFORM, the system prompts for the field number in which data is to be deleted.

b. The flag which indicates the existence of data in the field is set to zero in the file formname.DAT.

(C) DELETE DATA FROM ALL FIELDS

Purpose : To remove data from all fields in a form

Initiated by : Option 3 of the MAIN menu and Option 3 of the sub-menu CREATE/MODIFY/DELETE DATA IN A FORM

Input files : TFORM, formname.DAT

Output files : TFORM

Code reference : Lines 1960 to 3550 in TH2

Procedures :

a. If the form name specified by the user is among the ones in the file TFORM, the system confirms the deletion action by prompting the user.

b. If deletion is not confirmed, the system returns to the MAIN menu. Otherwise, data is deleted by erasing the file formname.DAT. The formname is removed from the file TFORM.

4.2.4 DISPLAY/PRINT DATA USING A FRAME

This function allows user to output a form using a frame. Two types of output can be obtained on screen or on printer as
shown in Figure 4.9.

---

**DISPLAY/PRINT DATA USING A FRAME**

---

:DISPLAY DATA ON : PRINT DATA ON :
THE SCREEN : THE PRINTER

---

**Figure 4.9 Sub-functions of DISPLAY/PRINT DATA USING A FRAME**

**Purpose**: Output a form on the screen or on the printer

**Initiated by**: Option 4 of the MAIN menu and Option 1 of the sub-menu DISPLAY/PRINT DATA USING A FRAME

**Input files**: TFRAME, TFORM, framename.COR, formname.DAT

**Code reference**: Lines 3560 to 4640 in TH2

**Procedures**:

a. If the formname specified by the user is found among the ones in the file TFORM, user will be prompted for the frame name.

b. If the framename is found in the file TFRAME, user can specify whether to output the whole form or part of the form.

c. If part of the form is specified, the system prompts for the field number.

d. If the field contains data and the corresponding block have been defined, the form is outputted by positioning the data at the coordinates.

---

**4.2.5 DISPLAY FRAME LAYOUT**

**Purpose**: To display the coordinates of the blocks in the frame

**Initiated by**: Option 5 of the MAIN menu or in the function CREATE A FRAME or MODIFY/DELETE BLOCK COORDINATES
Input file : frame.name.COR

Code reference : Lines 4650 to 5030 in TH2

Procedures :

a. If the frame name specified by the user is found in the file frame.name.COR, the coordinates of the blocks are read and are displayed with scales on the sides of the screen.

4.2.6 QUIT

The user can exit from the system and return to DOS on completing a function of FMS-UO.

4.3 SYSTEM INTEGRATION OF FMS-UO and RQL

One of the obstacles we encountered during integration was to find a suitable relational database management system which runs on micro-computers. Among the very few existing systems, RQL was the only one which we were able to obtain permission from its developer HELLO Software Ltd. for copying and modifying the codes so that integration is possible.

The limitation on the size of the memory prohibits loading both systems, FMS-UO and RQL, into memory. Instead, one system is loaded at a time. When the user initiates the transfer of data from one system to the other, the DOS command RUN is used to execute the other system. This command RUN clears the memory before the new system is loaded. Thus, a data file PASS is used to store data that is passed from one system to the other. Because of this obstacle, longer processing time for system integration is needed and the number of functions that can be implemented on form is limited.

A lot of effort is spent on designing the features of system
Integration such as how system integration is initiated and the entrance and exit points of this function in both systems. Data transfer is initiated in the option CREATE/MODIFY/DELETE DATA FROM A FORM in FMS-UO. When the user specifies the transfer of data from RQL, the flag in record 1 of the file PASS is set to "1". The name of the form where data is to be input is stored in record 2, and the field number is stored in record 3 of PASS. The system control is passed to RQL. Data resulted from the RQL command DISPLAY is transferred to FMS-UO on the user's request. The system control is then returned to FMS-UO to continue with the process of creating or modifying data in a form. If the value of the record 4 of PASS is "1", data is retrieved from record 6 and is stored in the field of the form as specified in records 2 and 3 of PASS.

RQL consists of a Main program RDB.OPT.HALF and two subprograms SORT and SET.OPT. The program RDB.OPT.HALF consists of a section for identifying keywords in the command input by the user. The system then branches off to the subroutine of a specific function. The system control is passed to and from the Main program and two subprograms SORT and SET.OPT using the BASIC command RUN.

The original codes in the program RDB.OPT.HALF are modified in order to integrate RQL with FMS-UO. Codes are added at the beginning of the program for reading the flag in record 1 of the file PASS to determine whether data transfer is requested. When a command DISPLAY is input, the subroutine for the command is executed to search the tables for data that meet the specification in the command. If data transfer is requested, an
additional prompt is displayed to ask the user whether data resulted from the command DISPLAY be transferred to FMS-UO. If the answer to the prompt is "Y", the data selected to be transferred is stored in record 6 of the file PASS. The flag in record 4 of PASS is set to "1" to indicate the transfer of data from RQL. The system control is returned to FMS-UO using the BASIC command RUN. If the user responds "N" to the prompt for data transfer, the system control goes to a section in RQL which prompts for an command. Upon completion of an RQL command, a prompt is added to ask the user if he wants to continue with RQL or execute FMS-UO. Figure 4.10 describes the logic flow of RQL after the modification for system integration.
Figure 4.10 The logic flow of RQL after the modification for system integration
CHAPTER 5
AN EXAMPLE AND SOME CONCLUDING REMARKS

5.1 INTRODUCTION
In business, companies often send their clients and employees circulars or letters with identical contents but different names and addresses. It will be very tedious and time-consuming to have the same contents typed repeatedly. In this chapter, we illustrate how time and effort can be saved by generating the letters through the operations and interface of the two systems FMS-UO and RGL. Different formats for outputting the letters are also considered. The limitations of FMS-UO and areas for future research are discussed in Section 5.5.

5.2 THE PROCESS OF GENERATING SIMILAR LETTERS - AN EXAMPLE
The process consists of four steps:

(A) The names and addresses of the clients are stored in a relation in RQL.
(B) A frame for the output format of the letter is created by FMS-UO.
(C) Data of the letter are stored in a data file in FMS-UO.
(D) The letters are generated.

We shall explain these steps using the clients' names and addresses listed in Figure 5.1. A sample letter is shown in Figure 5.2.

<table>
<thead>
<tr>
<th>LNAME</th>
<th>FNAME</th>
<th>STREET</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWN</td>
<td>MARY</td>
<td>20 MAIN ST.</td>
<td>TORONTO</td>
</tr>
<tr>
<td>STEWART</td>
<td>JACK</td>
<td>111 LAURIER ST.</td>
<td>OTTAWA</td>
</tr>
<tr>
<td>SMITH</td>
<td>JOHN</td>
<td>10 DUNDAS ST.</td>
<td>HAMILTON</td>
</tr>
</tbody>
</table>

Figure 5.1 Names and addresses of clients
20 MAIN ST.
TORONTO

DEAR MARY :

I WILL BE ON VACATION FROM DECEMBER 20
TO JANUARY 10. PLEASE CONTACT JIM RAY
FOR SYSTEM SUPPORT.

VPDP RELEASE 1.1 IS INSTALLED!

JOSEPH

---

Figure 5.2 A sample letter

(A) The names and addresses of the clients are stored in a
relation in RGL.

The data in Figure 5.1 is stored in a table (relation) INFORM
of a database CLIENT in RGL using the following commands:

. DEFINE DB CLIENT
  (To create a database called CLIENT)

. USE DB CLIENT
  (To specify the database)

. DEFINE TABLE INFORM
  (To create a table called INFORM, the description, data type
   and width of each column are described)

. INSERT INFORM
  (To input data into the table INFORM)

(B) A frame for the output format of the letter is created

The letter shown in Figure 5.2 is divided into different
fields according to the functions and locations of the data. (See
Figure 5.3.) Some guidelines for the division are:

a. A unit of data to be manipulated independently of the others
   is considered as a field, e.g., the name of a client.

50
b. Any part of the data requiring special layout can be assigned as a field. For example, the line which starts with "COFFEE TABLE : : " in Figure 5.2 is labeled as field #8. These words have special starting and ending positions which are different from those in the other lines.

---

1(20 MAIN ST. )
2(TORONTO )
3(DEAR)(MARY )(:)
4
5
6(I WILL BE ON VACATION FROM DECEMBER 20 TO JANUARY 10. PLEASE CONTACT JIM RAY FOR SYSTEM SUPPORT.)
7(VPDP RELEASE 1.1 IS INSTALLED !)
8(JOSEPH)

---

Figure 5.3. The different fields of the letter

The frame "ALETTER" is defined under Option 1 of the MAIN menu. The coordinates of its 9 blocks are listed in the following table.

<table>
<thead>
<tr>
<th>BLOCK NUMBER</th>
<th>X</th>
<th>Y</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>6</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>10</td>
<td>40</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>14</td>
<td>40</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>17</td>
<td>8</td>
<td>17</td>
</tr>
</tbody>
</table>

The frame ALETTER as shown in Figure 5.4 is displayed using Option 5 DISPLAY FRAME LAYOUT of the MAIN menu.

51
Figure 5.4 The frame ALETTER as displayed on the screen

(C) Data for the letter are stored in a data file in FMS-ÚO

For each letter, data is input into its fields #1, #2 and #4 from RQL and into the remaining fields from the console. The form name "SALE" is input under Option 3 CREATE/MODIFY/DELETE DATA IN A FORM of the MAIN menu. Option 1 CREATE/MODIFY DATA IN A FIELD is chosen from the sub-menu.

a. Data input from RQL

To input the street address of the client, Mary Brown for example, from RQL, field #1 is specified when the system prompts for the field number of the input data and answer with an "R" to the question in Figure 5.5. The message in Figure 5.6 will appear on the screen.
FMS> DO YOU WANT TO INPUT DATA FROM
CONSOLE OR RQL? (C OR R)

Figure 5.5 User can input data from console or RQL

** IF YOU HAVE A TABLE WHOSE DATA
YOU WANT TO MOVE TO FMS-UO, USE
THE RQL COMMAND TO DISPLAY THAT
TABLE. OTHERWISE, DEFINE YOUR TABLE NOW

Figure 5.6 The displayed message after user inputs data from RQL

In response to the prompt RQL>, the following commands should be typed in order to retrieve the street address of the client, Mary Brown.

RQL> USE DB CLIENT
RQL> DISPLAY TABLE INFORM ON STREET WHERE LNAME = "BROWN"

Figure 5.7 shows the street address retrieved from the database. The system then prompts if the data displayed is to be moved to the field in FMS-UO. (See Figure 5.8.) If the answer

<table>
<thead>
<tr>
<th>STREET</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 MAIN ST.</td>
</tr>
</tbody>
</table>

Figure 5.7 The address of Mary Brown retrieved from database

DO YOU WANT TO MOVE THIS DATA TO
THE FORM? (Y OR N)

Figure 5.8 The system prompts if the data is to be moved to the form

is "Y", the street address "20 MAIN ST." will be inserted into field #1 of the letter.
b. Data input from the console

The literal "DEAR" is input into field #3 of the letter through the console by specifying a "C" to the question as shown in Figure 5.5.

(D) The letters are generated

After having created the contents and frame of the letter, we choose Option 4 DISPLAY/PRINT DATA USING A FRAME of the MAIN menu to output the letter. Suppose we want to display the whole letter, we will select the Option DISPLAY DATA ON THE SCREEN and answer with "0" to the following question:

```
FMS> HOW MANY FIELDS DO YOU WANT TO OUTPUT? (0 MEANS ALL):
```

Figure 5.9 The system prompts for the fields to be output

If we want to output only the street, city and name of the client, we answer with a "3" to the above question. As shown in Figure 5.10, we specify Fields #1, #2 and #4 to be output. The result is shown in Figure 5.11.

```
FMS> PLEASE SPECIFY THE FIELD NUMBERS:
1 2 4
```

Figure 5.10 The field numbers specified for output

```
20 MAIN ST.
TORONTO
MARY
```

Figure 5.11 The address and name of the client are output
5.3 Generating letters with same contents but different clients

To generate similar letters for the other clients, Jack Stewart and John Smith, their names, street addresses and cities have to be retrieved from the database in RQL. The letter to Jack Stewart, as shown in Figure 5.12, is generated by selecting Option 4 DISPLAY/PRINT DATA USING A FRAME of the MAIN menu and specifying the frame "ALETTER" and the form "SALE".

```
111 LAURIER ST.
OTTAWA

DEAR JACK:

I WILL BE ON VACATION FROM DECEMBER 20
TO JANUARY 10. PLEASE CONTACT JIM RAY
FOR SYSTEM SUPPORT:

VPDP RELEASE 1.1 IS INSTALLED!

JOSEPH
```

Figure 5.12 The letter addressed to Jack Stewart

5.4 Outputting the letter by a different format

The format of the letter with the same contents can be changed if a different frame is used. If, the frame "BLETTER" shown in Figure 5.13 is created under Option 1 of the MAIN menu, the generated letter is shown in Figure 5.14.
Figure 5.13 The frame BLETTER

VPDP RELEASE 1.1 IS INSTALLED!

111 LAURIER ST.
OTTAWA

DEAR JACK

I WILL BE ON VACATION FROM DECEMBER 20 TO JANUARY 10. PLEASE CONTACT JIM RAY FOR SYSTEM SUPPORT.

JOSEPH

Figure 5.14 The letter addressed to Jack Stewart
5.5 SOME CONCLUDING REMARKS

In this thesis, a computerized form manipulation system FMS-UO and its integration with a commercial relational database management system are designed and developed. FMS-UO is implemented in a very restrictive computing environment - microcomputer APPLE II using the Disk Operating System DOS 3.3. Many form manipulation features, such as routing and tracing of forms cannot be implemented because of the limitation of computing resources. Its random access memory is only 64K which is very small for implementing system integration. For example, data shared between the two systems cannot be stored in the memory, instead, a data file is used. The speed of executing the functions is slow because of the micro-processor used. The screen is limited to 20 lines by 40 characters which is quite unlikely to represent a normal work paper. More sophisticated hardware and software facilities will enable the development of a more marketable system. The data processing capability of FMS-UO can be extended in the area where data retrieved from SQL for more than one form instance can be updated to the form instances in FMS-UO respectively.

As the cost of hardware is dropping considerably, automated systems for the home environment can become more sophisticated. It not only supports the inputting and outputting of voice, image, text, data and graphics, but can also be connected through the local area network to the banking system, grocery system and other commercial systems.
APPENDIX A
ERROR MESSAGES FOR FMS-UO

The system can detect certain errors during input or output operations. Following is a list of the error messages and their explanations.

CREATING A FRAME, MODIFY/DELETE BLOCK COORDINATES

*** ERROR! DUPLICATE FRAME - TRY AGAIN ***

If the input name for the new frame already exists, this error message will be displayed.

*** ERROR! X MUST BE BETWEEN 1 AND 40, REDEFINE THE X COORDINATE ***

The X coordinate is outside the range allowed by the system.

*** ERROR! Y MUST BE BETWEEN 1 AND 22, REDEFINE THE Y COORDINATE ***

The Y coordinate is outside the range allowed by the system.

*** ERROR! COORDINATES OF THE BOTTOM POINT MUST BE GREATER THAN THE TOP COORDINATES, REDEFINE THE BLOCK ***

The bottom right corner must be on the right and below of the top left corner.

*** ERROR! OVERLAPPING BLOCKS. REDEFINE THE BLOCK ***

The block overlaps with other blocks of the frame.

*** ERROR! FRAME HAS NOT BEEN DEFINED ***

*** ERROR! BLOCK HAS BEEN DEFINED, CANNOT BE ADDED ***

*** ERROR! BLOCK DOES NOT EXIST BEFORE, CANNOT BE DELETED ***

CREATE/MODIFY/DELETE DATA IN A FORM

*** ERROR! FIELD NUMBER CANNOT BE <= 0 ***
*** ERROR! FIELD NUMBER CANNOT BE > 10 ***
Field number must be between 1 and 10, inclusive.

*** ERROR! FORM HAS NOT BEEN DEFINED ***
The form specified by the user has not been created.

DISPLAY/PRINT DATA USING A FRAME

*** ERROR! FRAME HAS NOT BEEN DEFINED ***
The frame requested for outputting does not exist.

*** ERROR! FORM HAS NOT BEEN DEFINED ***
The form specified for output does not exist.

*** ERROR! BLOCK #X HAS NOT BEEN DEFINED ***
The block #X in the specified frame for outputting the form has not been defined yet.

*** FMS> PLEASE INSERT THE DISKETTE WHICH CONTAINS THE FILE FORM.DAT INTO DISK DRIVE 2 THEN PRESS ANY KEY ***
The data file is not in disk drive 2.

*** FMS> PLEASE CONNECT THE PRINTER THEN PRESS ANY KEY ***
The printer is not hooked up to the computer.

DISPLAY FRAME LAYOUT

*** ERROR! FRAME NAME DOES NOT EXIST ***
APPENDIX B

THE PROGRAM OF FMS-uo (TH2)

CODES

.COM
ﺚ$t\$ THIS FORM MANIPULATION
SYSTEM (FMS) CONSISTS OF TW O PARTS:

110 REM PART I. FACILITIES
FOR CREATING, MODIFYING, DEL ETING, DISPLAYING AND PRINTING FORMS.

120 REM PART II. FACILITIES
FOR INTERFACING WITH AN EXIS TING RELATIONAL DATABASE SYS TEM (RGL).

125 D$ = CHR$(4): PRINT D$; "MAX FILES 4"

130 DIM X1%(10), X2%(10), Y1%(10), Y2%(10), A%(100), P%(10 ,100), PS%(100), N%(10), C%(100), E%(22, 10, 10)

140 Z% = 0: W% = 1: G% = Z%: G$ = CHR$(7): D$ = CHR$(4)

150 REM CHECK IF THERE IS A NY DATA PASSED FROM RGL TO F MS

160 PRINT D$: PRINT D$; "OPEN PASS L300,D1,S6": PRINT D$: "READ PASS,R1": INPUT L$: PRINT D$: "WRITE PASS,R1": PRINT D$: "CLOSE PASS": IF FB% = 0 THEN GOTO 180

170 PRINT D$: "OPEN PASS L300,D1": PRINT D$: "READ PASS,R4": INPUT L$: PRINT D$: "WRITE PASS,R1": PRINT D$: "CLOSE PASS"

180 ONERR GOTO 200

190 PRINT D$: "UNLOCK TFRAME,D1": POKE 216, 0: GOTO 220

200 POKE 216, 0

210 PRINT D$: "OPEN TFRAME,L20,D1 ": PRINT D$: "WRITE TFRAME,R0 ": PRINT Z%: PRINT D$: "CLOSE " : PRINT D$: PRINT D$: "OPEN TFORM,L20,D1": PRINT D$: "WRITE TFORM,R0": PRINT Z%: PRINT D$: PRINT D$: "CLOSE"

220 IF FB% = Z% THEN GOTO 250

230 IF L$ = W% THEN GOTO 3030

250 CALL -936: REM MAIN MENU

260 PRINT "FMS) INDICATE YOUR CHOICE BY ENTERING THE NUMBER"

-Declaration of arrays

-Check if any data is passed from RGL in the file PASS

Check if the file TFRAME exists.

Data is passed from RGL

Display the Main Menu
270 PRINT : PRINT "1. CREATE A FRAME" : PRINT : PRINT "2. MODIFY/DELETE BLOCK COORDINATES"

280 PRINT : PRINT "3. CREATE/MODIFY/DELETE DATA IN A FORM" : PRINT

290 PRINT "4. DISPLAY/PRINT DATA USING A FRAME" : PRINT

300 PRINT "5. DISPLAY FRAME LAYOUT" : PRINT

310 PRINT "6. QUIT" : PRINT : INPUT ANS

320 IF ANS = "1" THEN GOTO 430

330 IF ANS = "2" THEN GOTO 930

340 IF ANS = "3" THEN GOTO 1970

350 IF ANS = "4" THEN GOTO 3590

360 IF ANS = "5" THEN GOTO 4680

370 IF ANS = "6" THEN GOTO 5030

380 GOTO 250

400 REM

410 REM CREATE A FRAME

420 REM

430 CALL - 936: RO% = W%: RE% = Z%

440 PRINT "CREATE A FRAME": PRINT "==============": PRINT : INPUT "FMS> PLEASE GIVE A NEW FRAME NAME: " : IFP$

450 GOSUB 1900 : REM CHECK FOR DUPLICATE FRAME NAME

460 IF Y% = Z% THEN PRINT G$: "** ERROR! DUPLICATE FRAME NAME - TRY": PRINT " AGAIN ***": FOR PAUSE = 1 TO 1500 : NEXT PAUSE: GOTO 430

470 I = W%: NE% = W%: TP% = Z%: BI = I: REM: NO DATA IN THE COORDINATE FILE YET

480 GOSUB 1380 : REM THE USER WANTS TO SEE THE FRAME DISPLAY

485 I = BI

490 PRINT : PRINT "BLOCK# "; I: BI = I

500 PRINT : PRINT "FMS> PLEASE SPECIFY THE X,Y COORDINATES": PRINT "OF THE TOP LEFT CORNER OF BLOCK# "; I
CODES

220 WX% = Z%: C1% = W%
530 I = BI
540 PRINT
550 INPUT "X, (1-40) : ":;X1%(I):
560 INPUT "Y, (1-22) : ":;Y1%(I):
595 GOSUB 1380: X% = X1%(BI): Y% = Y1%(BI) = Y%: REM IF USER WANTS TO SEE THE FRAME DISPLAY
600 I = BI: PRINT : PRINT "FNS> P LEASE SPECIFY THE X, Y COORDINATES": PRINT " OF THE BOTTOM RIGHT CORNER OF" : PRINT " BLOCK# ":;I
620 WX% = Z%: C1% = Z%: I = BI: WB% = W%
630 PRINT
640 INPUT "X, (1-40) : ":;X2%(I):
650 INPUT "Y, (1-22) : ":;Y2%(I):
680 IF X2%(I) < X1%(I) OR Y2%(I) < Y1%(I) THEN PRINT G$: PRINT "*** ERROR! COORDINATES OF THE BOTTOM POINT MUST BE GREATER THAN THE TOP COORDINATES."
690 PRINT " REDEFINE THE BLOCK ***": GOTO 490
700 IF I = W% AND RE% = W% THEN GOTO 720
710 IF I = W% THEN X%(I) = 1: GOTO 750
720 IF RO% = W% THEN X%(I) = I: K = I
730 GOSUB 1430: REM CHECK IF THE BLOCKS OVERLAP
740 IF ER% = W% THEN PRINT : PRINT G$: "*** ERROR! OVERLAPPING BLOCKS." : PRINT " REDEFINE THE BLOCK ***": ER% = Z%: GOTO 490
750 PRINT : PRINT "FNS> DO YOU WANT TO MAKE ANY CHANGES?" : INPUT " FOR THIS BLOCK ? (Y OR N) : "; AN$
CODES

760 IF ANS = "Y" THEN GOTO 764
762 IF ANS = "N" THEN GOTO 780
763 IF ANS < > "Y" AND ANS < > "N" THEN GOTO 750
764 PRINT : PRINT "FMS> DO YOU WANT TO LEAVE THIS OPTION?": INPUT " (Y OR N) : "; CE$
767 IF CE$ = "Y" THEN PRINT D$:
768 IF CE$ = "N" THEN GOTO 490
769 IF CE$ < > "Y" AND CE$ < > "N" THEN GOTO 764
780 GOSUB 1700: REM STORE THE COORDINATES INTO THE FILE
790 IF RE% = 1 THEN GOTO 850
800 GOSUB 1830: REM RECORD THE BLOCK# BEING DEFINED
810 IF RO% < > W% THEN GOTO 850
820 PRINT : INPUT "FMS> DO YOU WANT TO DEFINE ANOTHER BLOCK? (Y OR N) : "; ANS$
830 IF ANS = "Y" THEN I = I + 1:
840 IF ANS < > "Y" AND ANS < > "N" THEN GOTO 820
850 IF RO% = Z% THEN RETURN
860 REM PUT DATA INTO TFRAME TO RECORD THE FRAME NAME
870 PRINT D$; "OPEN TFRAME,L20,D1 "; PRINT D$; "READ TFRAME,RO": INPUT K%
880 PRINT D$; "WRITE TFRAME;RO": PRINT K% + 1
890 PRINT D$; "WRITE TFRAME,R": K% + 1: PRINT FS$
900 PRINT D$; "CLOSE TFRAME";
910 GOTO 250
920 REM
930 CALL - 936: RO% = Z%
940 PRINT "MODIFY/DELETE BLOCK COORDINATES": PRINT "=": PRINT
942 RE% = Z%: REM BLOCK EXIST BEFORE?
944 PRINT : INPUT "FMS> PLEASE SPECIFY FRAME NAME : "; FF$
946 PRINT D$; "OPEN TFRAME,L20,D1 "; TEMP = Z%
948 PRINT DS; "READ TFRAME,RO"; INPUT K%: PRINT DS; IF K% = Z% THEN!
PRINT: PRINT G$: "*** ERROR!
FRAME HAS NOT BEEN DEFINED
***": PRINT DS; PRINT DS; "CLOSE": FOR PAUSE = 1 TO 1500: NEXT
GOTO 250
950 FOR I = 1 TO K%: PRINT DS; "R
READ TFRAME,R"; I: INPUT FS$: IF
FS$ = FS$ THEN TEMP = I
952 NEXT : PRINT DS
954 IF TEMP = Z% THEN PRINT G$: "*** ERROR!
FRAME HAS NOT BEEN DEFINED ***": PRINT DS; PRINT
D$: "CLOSE": FOR PAUSE = 1 TO
1500: NEXT PAUSE; GOTO 250
960 PRINT: PRINT: PRINT "PMS>
PLEASE INDICATE YOUR CHOICE
BY": PRINT "ENTERING TH
E NUMBER"
965 PRINT: PRINT "1. MODIFY AN
EXISTING BLOCK": PRINT: PRINT
"2. ADD A NEW BLOCK": PRINT
: PRINT "3. DELETE AN EXISTI
NG BLOCK"
970 PRINT: PRINT "4. DELETE THE
WHOLE FRAME": PRINT: INPUT
CE$.
975 CALL - 936
980 IF CE$ <> "1" AND CE$ <> "2" AND CE$ <> "3" AND CE$ < > "4" THEN GOTO 950
1060 IF CE$ <> "4" THEN GOTO
1120
1070 PRINT: PRINT "PMS> YOU HAV
E CHOSEN TO DELETE THE": PRINT
"WHOLE FRAME, DO YOU ST
ILL": INPUT ": WANT TO GO
AHEAD? (Y OR N) : ": ANS$
1080 IF ANS$ <> "Y" AND ANS$ <> "N" THEN GOTO 1070
1090 IF ANS$ = "Y" THEN GOTO 111
1100 IF ANS$ = "N" THEN GOTO 250
1103 GOTO 1070
1110 PRINT DS; "WRITE TFRAME,R"; TEMP: PRINT "": PRINT DS; PRINT
DS; "CLOSE TFRAME": PRINT DS; "DELETE"; FY$: ".COR,D1": GOTO
250
1120 PRINT DS; "OPEN"; FS$; ".COR,L" 50*

Check if the frame name has been defined

Display options in submenu

The system prompts for confirmation of deleting the whole frame

Delete the frame name from the file TFRAME
1130 PRINT D$;"READ";FF$;".COR,R0"; INPUT K; FOR J = 1 TO K: INPUT X%(J): NEXT
1140 PRINT D$;"CLOSE";FF$;".COR"

1142 IF CES = "1" THEN PRINT "MODIFY AN EXISTING BLOCK'S COORDINATES": PRINT "="
1144 IF CES = "2" THEN PRINT "ADD A NEW BLOCK": PRINT "="
1146 IF CES = "3" THEN PRINT "DELETE AN EXISTING BLOCK'S COORDINATES": PRINT "="

1150 PRINT: PRINT "FMS> BLOCKS DEFINED SO FAR ARE:".
1160 FOR J = 1 TO K: PRINT X%(J); SPC(1): NEXT
1170 PRINT: GOSUB 1380: PRINT: REM SEE DISPLAY
1180 PRINT "FMS> PLEASE SPECIFY THE BLOCK NUMBER": INPUT "TO BE MODIFIED": ;I
1185 RE% = 0
1190 FOR J = 1 TO K
1200 IF I = X%(J) THEN RE% = 1: GOTO 1360: REM THIS BLOCK EXIST
1210 NEXT
1215 IF CES = "2" AND RE% = 1 THEN PRINT G$;"***ERROR! BLOCK HAS BEEN DEFINED, ": PRINT "CANNOT BE ADDED ***": FOR PAUSE = 1 TO 1500: NEXT PAUSE: GOTO 250
1220 IF CES = "1" OR CES = "2" THEN GOTO 1320: REM ATTACH AND MODIFY BLOCKS' COORDINATES
1230 IF CES = "3" THEN CALL -936: PRINT "DELETE BLOCK": PRINT "="
1240 IF RE% < > .W% THEN PRINT G$;"***ERROR! BLOCK DOES NOT EXIST BEFORE,": PRINT "CANNOT BE DELETED ***": FOR PAUSE = 1 TO 1700: NEXT PAUSE: GOTO 250
1250 PRINT D$; "OPEN"; FS$; ".COR, L50"
1260 PRINT D$; "WRITE"; FS$; ".COR, RO"; PRINT (K - 1)
1270 FOR J = 1 TO K: IF X%(J) < 
> GE% THEN PRINT X%(J)
1280 NEXT
1290 PRINT D$; "CLOSE"; FS$; ".COR"
1300 PRINT "FM$> BLOCK NUMBER "; 
: PRINT GE%; PRINT " HAS BE 
EN DELETED"; FOR PAUSE = 1 TO 
1500: NEXT PAUSE
1310 GOTO 250
1320 ER% = Z%; PRINT D$; "OPEN"; FS 
$: ".COR, L50, D1"; FOR J = 1 TO 
K: PRINT D$; "READ"; FS$; ".COR 
,R"; J: INPUT X%(J): INPUT X 
2%(J): INPUT Y%(J): INPUT Y 
2%(J): NEXT: PRINT D$
1330 GOSUB 490: PRINT D$
1340 PRINT D$: PRINT: PRINT "FM 
$> DO YOU WANT TO MODIFY OTH 
ER BLOCKS"; INPUT " OF T 
HIS FRAME? (Y OR N) : "; AN$
1350 IF AN$ = "Y" THEN CALL - 
936: GOTO 960
1360 IF AN$ < > "Y" AND AN$ < 
"N" THEN GOTO 1340
1370 PRINT D$: PRINT D$; "CLOSE"; 
: GOTO 250
1375 REM SUBROUTINE TO ASK THE 
USER IF HE WOULD LIKE TO SEE 
THE FRAME
1380 PRINT D$: PRINT: PRINT "FM 
$> DO YOU WANT TO SEE THE FR 
AME"; INPUT " FIRST? (Y 
OR N) : " ; AN$
1390 IF AN$ = "Y" THEN SU% = W%; 
GOSUB 4750: REM SEE THE FR 
AME DISPLAY
1400 IF AN$ < > "Y" AND AN$ < 
"N" THEN GOTO 1380
1410 IF AN$ = "Y" THEN CALL - 
936
1420 RETURN
1430 REM SUBROUTINE TO CHECK 
OVERLAPPING
1440 L = W%
CODES

1450 J = X%(L)
1460 IF RE% = W% THEN GOTO 1550

1470 IF Y1%(I) < Y1%(J) AND Y2%(I) < Y1%(J) THEN GOTO 1520
1480 IF X1%(I) < X1%(J) AND X2%(I) < X1%(J) THEN GOTO 1520
1490 IF Y1%(I) > Y2%(J) AND Y2%(I) > Y2%(J) THEN GOTO 1520
1500 IF X1%(I) > X2%(J) AND X2%(I) > X2%(J) THEN GOTO 1520
1510 ER% = W%; RETURN
1520 IF RE% = W% THEN GOTO 1560

1530 L = L + 1: IF L < K THEN GOTO 1450
1540 RETURN
1550 IF I < > J THEN GOSUB 1470
1560 IF ER% = W% THEN RETURN
1570 L = L + 1: IF L < = K THEN J = X%(L): IF I < > J THEN
GOTO 1470
1580 IF I = J AND L < = K THEN
GOTO 1570
1590 RETURN
1600 REM CHECK COORDINATES LIMIT
1610 WY% = Z%
1620 IF X% < 1 OR X% > 40 THEN PRINT GS; "***ERROR! X MUST BE BETWEEN 1 AND 40."; PRINT " REDEFINE THE X COORDINATE ***"
WY% = W%
1623 IF WY% = W% AND FR% = 1 THEN
GOTO 550
1625 IF WY% = W% AND FR% = 2 THEN
GOTO 640
1627 IF WY% < > W% AND FR% = 1 THEN
GOTO 560
1627 IF WY% < > W% AND FR% = 2 THEN
GOTO 650
1628 WY% = Z%
1630 IF Y% < 1 OR Y% > 22 THEN PRINT GS; "***ERROR! Y MUST BE BETWEEN 1 AND 22."; PRINT " REDEFINE THE Y COORDINATE ***"
WY% = W%
1632 IF WY% = W% AND FR% = 1 THEN
GOTO 560
1633 IF WY% = W% AND FR% = 2 THEN
GOTO 650
CODES

1634 IF WX% < > W% AND FR% = 1 THEN
GOTO 595
1635 IF WX% < > W% AND FR% = 2 THEN
GOTO 690
1700 REM STORE DATA INTO FRAME NAME COR
1710 PRINT D$: "OPEN"; FF$; ".COR", L
50"
1720 IF RE% = 1 THEN GOTO 1780
1730 IF NE% = W% THEN PRINT D$: "WRITE"; FF$; ".COR", RO": PRINT I
1: PRINT D$: GOTO 1780
1740 PRINT D$: "READ"; FF$; ".COR", RO": INPUT NB%
1750 PRINT D$
1760 PRINT D$: "WRITE"; FF$; ".COR", RO": PRINT NB% + 1
1770 PRINT D$
1780 PRINT D$: "WRITE"; FF$; ".COR", RO": I
1790 PRINT X1%(I): PRINT X2%(I):
PRINT Y1%(I): PRINT Y2%(I)
1800 PRINT D$: "CLOSE"; FF$; ".COR"

1810 NE% = Z%
1820 RETURN
1830 REM PUT BLOCK# INTO RO OF FF$..COR
1840 PRINT D$: "OPEN"; FF$; ".COR", L
50"
1850 PRINT D$: "READ"; FF$; ".COR", RO": INPUT NB%
1860 PRINT D$: "WRITE"; FF$; ".COR", RO":
1870 PRINT D$: "POSITION"; FF$; ".COR", RO": NB%
1880 PRINT D$: "WRITE"; FF$; ".COR": PRINT I
1890 PRINT D$: "CLOSE"; FF$; ".COR": RETURN
1900 ONERR GOTO 1920: REM CHEC K FOR DUPLICATE FRAME NAME
1910 Y% = Z%: PRINT D$: "UNLOCK"; FF$; ".COR", D1": POKE 216, O: RETURN

1920 Y% = W%: POKE 216, 0: GOTO 46
0: REM NOT EXIST BEFORE

1930 ONERR GOTO 1950: REM CHEC K IF FRAME NAME EXIST
1940 Y% = Z%: PRINT D$: "UNLOCK"; FF$; ".COR", D1": POKE 216, 0: RETURN
CODES

1950 Y% = W%: POKE 216,0: GOTO 2100: REM NOT EXIST BEFORE
1960 REM
1970 REM ENTER, MODIFY OR DELETE DATA IN FRAME
1980 REM
1990 CALL = 936
2000 PRINT "CREATE/MODIFY/DELETE DATA IN A FORM": PRINT "==
2001 ==
2002 PRINT : INPUT "FMS> PLEASE SPECIFY THE FORM NAME : " ;FF$
2003 PRINT D$: PRINT D$;"OPEN TF ORM.L20,D1":TEMP = Z%
2004 PRINT D$:"READ.TFORM,RO": INPUT K%: PRINT D$
2005 IF K% = 0 THEN GOTO 2008
2006 FOR I = 1 TO K%: PRINT D$:"READ TFORM,R";I: INPUT FYS: IF
2007 FYS = FF$ THEN TEM = I: REM DATA FILE EXIST
2008 NEXT : PRINT D$: PRINT D$;"CLOSE TFORM"
2010 PRINT : PRINT : PRINT "FMS> PLEASE INDICATE YOUR CHOICE BY: PRINT ENTERING T
2011 HE NUMBER : ": PRINT : PRINT 
2012 1. CREATE/MODIFY DATA IN A FIELD"
2013 PRINT " 2. DELETE DATA FROM ONE FIELD": PRINT " 3. DELETE DATA FROM ALL FIELDS"
2020 INPUT AS$;
2040 IF AS$ < > "1" AND AS$ < > "2" AND AS$ < > "3" THEN GOTO 2030
2050 IF TEMP < > Z% THEN GOTO 2220
2120 IF AS$ = "3" OR AS$ = "2" THEN PRINT G$:"***ERROR! FORM HAS NOT BEEN DEFINED***": FOR
2130 PAUSE = 1 TO 1500: NEXT PAUSE
2140 ONERR GOTO 2160
2150 PRINT D$: PRINT D$;"OPEN";F
2155 PRINT "$;."DAT,D2": POKE 216,0: GOTO 2180: REM INITIALIZE FILE

COMMENTS

Option 3. CREATE/MODIFY/DELETE DATA IN A FORM

The system prompts for the form name.

Check if the form exists.

Display the options in the submenu.

69
2160 Y = PEEK (222): IF Y = 9 THEN
   PRINT "\"FMS\" DISK FULL! PLEASE INSERT AN\""; PRINT "\"INITIALIZED DISKETTE INTO DISK\"; PRINT "\"DRIVE 2 THEN PRESS ANY KEY\""; GET A$; GOTO 2140
2170 IF Y = 8 THEN PRINT G$;"FMS > PLEASE INSERT A DISKETTE\"; PRINT "\"INTO DISK DRIVE 2 THEN\"; PRINT "\"PRESS ANY KEY\""; GET A$; GOTO 2140
2180 PRINT D$;"WRITE\";FF$;".DAT"; REM INITIALIZE THE DATA FILE
2190 PRINT "\"00\""; PRINT "\"000\""; FOR I = 1 TO 20: PRINT "\"0\""; NEXT
2200 FOR I = 1 TO 10: PRINT "\"000\""; NEXT
2210 PRINT D$;"CHEC\";FF$;".DAT": PRINT D$;"POSITION\";FF$;"D AT\"; PRINT D$;"WRITE\";FF$;".DAT": PRINT D$;"CLOSE\";
2215 PRINT D$;"OPEN TFORM,L20,D1\"; PRINT D$;"READ TFORM,RO\"; INPUT K%; PRINT D$; PRINT D$; "WRITE TFORM,RO\"; PRINT K% + 1: PRINT D$; PRINT D$; "WRITE TFORM,R\";K% + 1: PRINT F$; PRINT D$; PRINT D$; "CLOSE TFORM\";
2220 ONERR GOTO 2240: REM DATA FILE EXISTS
2230 PRINT D$; PRINT D$;"VERIFY\";FF$;".DAT,D2\"; POKE 216,0: GOTO 2250
2240 PRINT : PRINT "\"FMS\" PLEASE INSERT THE DISKETTE WHICH CONTAINS THE FILE \";FF$;".DAT INTO\"; PRINT "\"DISK DRIVE 2 THEN PRESS ANY KEY\""; GET A$: POKE 216,0: GOTO 2220
2250 IF A$ = "\"3\" THEN GOTO 327
2270 PRINT : INPUT "\"FMS\" PLEASE GIVE THE FIELD NUMBER : \";BB$
   IF BB$ > 10 THEN PRINT G$ "***ERROR! FIELD NUMBER CANNOT BE > 10 ***"; GOTO 2270
2290 IF BB$ < = 0 THEN PRINT G$
***ERROR! FIELD NUMBER CA
NOT BE <=0 ***": GOTO 2270
2360 IF AS$ = "2" THEN GOTO 350
70
2370 PRINT D$;"OPEN";FF$;".DAT";
D2".
2380 PRINT D$;"POSITION";FF$;".DA
AT,R":BB% + 1: PRINT D$;"REA
D";FF$;".DAT": INPUT ANS: PRINT
DS: IF VAL (ANS) = 0 THEN M
0% = Z%: PRINT D$;"CLOSE";FF
$;".DAT": GOTO 2460: REM
ENTER DATA FROM THE CONSOLE
OR FROM RQL
2390 PRINT : PRINT "FMS> THE BL
CK CONTAINS DATA, DO YOU" : INPUT
" WANT TO MODIFY THE DAT
A ? (Y OR N) " : ANS
2400 IF ANS < "Y" AND ANS < "N" THEN GOTO 2390
2410 IF ANS = "Y" THEN MO% = W%:
GOTO 2460: REM C OR R
2415 IF ANS < "Y" AND ANS < "N" THEN GOTO 2390
2420 PRINT : INPUT "FMS> CREATE
OR MODIFY OR DELETE ANOTHER
FIELD ? (Y OR N) " :
; ANS
2430 IF ANS < "Y" AND ANS < "N" THEN GOTO 2420
2440 IF ANS = "Y" THEN GOTO 200
4
2450 GOTO 250: REM IF ANS = "Y"
2460 PRINT : PRINT "FMS> DO YOU
WANT TO INPUT DATA FROM": INPUT
" CONSOLE OR FROM RQL ?
(C OR R) " : ANS
2470 IF ANS < "C" AND ANS < "R" THEN GOTO 2460
2480 IF ANS = "R" THEN GOTO 298
0: REM DATA FROM RQL
2490 PRINT : PRINT "FMS> PLEASE
INPUT DATA." : CO% = Z%: SC% =
W%: FB% = Z%: REM SC% = NO:
OF STRING
2500 BS = ""
2510 GET A$/: IF A$/ = CHR$(8)
THEN GOTO 2540: REM BACK
SPACE

Delete data in one field

Check whether the field contains data

The system prompts if the data in the field is to be modified

The system prompts if data in another field is to be modified

Return to Main Menu

The system prompts whether data is input from RQL or console

The system prompts for data from the console
2520 PRINT AA$: IF AA$ = CHR$(13) THEN GOTO 2650: REM CARRIAGE RETURN
2530 GOTO 2580
2540 IF LEN(B$) = 0 GOTO 2510
2550 IF ASC (RIGHT$(B$, 1)) > 31 THEN PRINT AA$""AA$;
2560 IF LEN(B$) = 1 THEN BS = "": GOTO 2510
2570 BS = LEFT$(B$, LEN(B$) - 1): GOTO 2510
2580 IF AA$ = CHR$(21) THEN 2510: REM FORWARD
2590 IF AA$ = ":" THEN BS = BS + "-+1": GOTO 2510: REM HAVE TO CONVERT THE CHARACTERS IF IT IS
2600 IF AA$ = CHR$(34) THEN BS = BS + "-+2": GOTO 2510: REM ":", OR "" OR ":", BECAUSE
2610 IF AA$ = "": THEN BS = BS + "-+3": GOTO 2510: REM THEY ARE NOT VALID CHARACTERS IN THE DATA FILE
2620 BS = BS + AA$
2630 IF LEN(B$) = 240 THEN A$(SC%) = B$: B$ = "": SC% = SC% + 1: GOTO 2510
2640 GOTO 2510
2650 A$(SC%) = B$
2660 REM SEE IF ANY COMMA, COL ON OR BACK SPACE IN THE INPUT
2670 FOR U = 1 TO SC%; ZA$ = A$(U): FOR J = 1 TO 100: C$(J) = "": NEXT J
2680 M% = W%
2690 FOR J = 1 TO LEN(ZA$)
2700 C1$ = MID$(ZA$, J, 1)
2710 IF C1$ > "": THEN C$(M%) = C$(M%) + C1$: M% + Z$: GOSUB 2730
2720 M% = M% + 1
2730 NEXT J: GOSUB 2800
2740 NEXT U: GOTO 3060
2750 S = LEN(C$(M%)): IF S > 3 THEN GOSUB 2800
2760 IF RIGHT$(C$(M%), 3) = "-+1" THEN C$(M%) = "": RETURN
CODES

2770 IF RIGHT$(C$(M%),3) = "++
2" THEN C$(M%) = CHR$(34):
RETURN
2780 IF RIGHT$(C$(M%),3) = "++
3" THEN C$(M%) = ":" : RETURN
2790 GOTO 2830
2800 IF RIGHT$(C$(M%),3) = "++
1" THEN S = S - 3:SS = LEFT$(
C$(M%),S):C$(M%) = S$ + ",",
: RETURN
2810 IF RIGHT$(C$(M%),3) = "++
2" THEN S = S - 3:SS = LEFT$(
C$(M%),S):C$(M%) = S$ + CHR$(
34): RETURN
2820 IF RIGHT$(C$(M%),3) = "++
3" THEN S = S - 3:SS = LEFT$(
C$(M%),S):C$(M%) = S$ + ":",
: RETURN
2830 RETURN
2970 REM ENTER DATA FROM RQL
2980 FB% = W%: CALL - 936:
2990 PRINT : PRINT "** NOTE: IF
YOU HAVE A TABLE WHOSE DATA"
: PRINT "YOU WANT TO MOVE TO RMS, USE THE": PRINT "RQL COMMAND TO DISPLAY
THAT TABLE": PRINT "OTHERWISE, DEFINE YOUR TABLE NO"
W" : X = FRE (0)
3000 PRINT DS;"OPEN PASS,L300,D1"
: PRINT DS;"WRITE PASS,R1":
PRINT FB%: PRINT DS; PRINT
DS;"WRITE PASS,R2": PRINT FF.
$: PRINT DS; PRINT DS;"WRITE
PASS,R3": PRINT BB%
3010 PRINT DS; PRINT DS;"WRITE P
ASS,R4": PRINT 0: PRINT DS; PRINT
DS;"CLOSE PASS"
3020 PRINT DS;"RUN RDB.OPT.HALF,
S6,D1": REM INPUT COMMAN
D OF RQL BUT FOR FORM DIS
LAY
3030 PRINT DS;"OPEN PASS,L300,D1"
: PRINT DS;"READ PASS,R2": INPUT
FF$
CODES

3040 PRINT DS$;"READ PASS,R3"; INPUT BB%: PRINT DS$; PRINT DS$;"READ PASS,R5"; INPUT SC%: PRINT DS$: PRINT DS$;"READ PASS,R6": FOR I = 1 TO SC%: INPUT A$(I): NEXT: PRINT DS$;"CLOSE PASS"

3050 REM CHECK IF THE RIGHT DISKETTE IS IN THE DISK DRIVE

3060 ONERR GOTO 3080

3070 PRINT DS$;"VERIFY";FF$;".DAT,D2": POKE 216,0: GOTO 3090

3080 POKE 216,0: PRINT "FMS> PLE ASE INSERT THE DISKETTE WHICH CONTAINS THE FILE "FF$".DAT INTO": PRINT "DISK DRIVE 2. THEN PRESS ANY KEY": GET ANS$: POKE 216,0: GOTO 3060

3090 IF MO% = W% GOTO 3350: REM MODIFY THE DATA

3100 PRINT DS$;"OPEN";FF$;".DAT,D2": PRINT DS$;"READ";FF$;".DAT"

3110 INPUT NB$: NB% = VAL (NB$) + 1: PRINT DS$;"OPEN";FF$;".DAT";

3120 PRINT DS$;"WRITE";FF$;".DAT"; IF NB% < 10 THEN NB$ = "0" + STR$(NB%)

3130 PRINT NB$: PRINT DS$: PRINT DS$;"READ";FF$;".DAT": INPUT LFS$: TES% = LFS$

3140 PRINT DS$;"OPEN ";FF$;".DAT": PRINT DS$;"POSITION";FF$;".DAT",R1"

3150 PRINT DS$;"WRITE";FF$;".DAT"

3160 LFS% = VAL (LFS$) + SC%: IF LFS% < 10 THEN LFS% = "0" + STR$(LFS%)

3170 IF LFS% < 10 THEN LFS% = "0" + STR$(LFS%)

3180 PRINT LFS$

3190 PRINT DS$;"POSITION";FF$;".DAT,R": BB% = 1: PRINT DS$;"WRITE";FF$;".DAT": PRINT 1

3200 PRINT DS$;"POSITION";FF$;".DAT,R9": PRINT DS$;"WRITE";FF$;".DAT": PRINT SC%
3210 PRINT D$;"POSITION";FFS;".D AT,R9"; PRINT D$;"WRITE";FFS; ";.DAT";TE% = VAL (TES$): IF 
TE% < 10 THEN TES$ = "00" + STRS (TE%) 
3220 IF TE% < 100 THEN TES$ = "0" + 
STRS (TE%) 
3230 PRINT TES$: PRINT D$ 
3240 PRINT D$;"OPEN";FFS;".DAT"; 
PRINT D$;"POSITION";FFS;".D AT,R"; VAL (TES$) - 1: PRINT 
D$;"WRITE";FFS;".DAT"; FOR I 
= 1 TO SC%; PRINT A$(I): NEXT 
: PRINT D$ 
3250 FOR I = 1 TO SC%;A$(I) = " 
: NEXT : GOTO 3520 
3260 REM DELETE ALL BLOCKS' DA TA 
3270 PRINT : PRINT "FMS> YOU HAV E CHOSEN TO DELETE DATA FROM 
"; PRINT "ALL THE FIELD 
S. DO YOU STILL WANT"; INPUT 
"TO GO AHEAD ? (Y OR N) 
";ANS 
3280 IF ANS < > "Y" AND ANS < > "N" THEN GOTO 3270 
3290 IF ANS = "Y" THEN GOTO 330 
0 
3300 GOTO 250 
3310 PRINT D$;"DELETE";FFS;".DAT 
,D2" 
3320 PRINT D$;"OPEN TFORM,L20,D1 

3330 PRINT D$;"WRITE TFORM,R";TEM P; PRINT " 
3340 PRINT D$: PRINT D$;"CLOSE T FORM"; PRINT "FMS> D ATA IN ALL THE FIELDS HAVE 
PRINT " BEEN DELETED"; FOR 
PAUSE = 1 TO 1500: NEXT PAUS 
E: GOTO 250 
3350 REM MODIFY THE DATA 
3360 PRINT D$;"OPEN";FFS;".DAT,D 
2"; PRINT D$;"POSITION";FFS; ";.DAT,R1" 
3370 PRINT D$;"READ";FFS;".DAT"; 
INPUT LFS;TES = LFS 
3380 PRINT D$;"OPEN";FFS;".DAT"; 
PRINT D$;"POSITION";FFS;".D AT,R";BB% + 21: PRINT D$;"WR ITE";FFS;".DAT"; PRINT LFS 

Store the number of strings used in the field 

Store the data retrieved from RGL into the file 

The system prompts for confirmation for deleting data from all the fields 

The data file is deleted 

Delete the form name from the file TFORM 

Modify data in the field 

Store the start address of the field
3390 PRINT D$;"OPEN";FFS;".DAT": PRINT D$;"POSITION";FFS;".DAT",RI":
3400 PRINT D$;"WRITE";FFS;".DAT": LF$ = VAL (LF$) + SC%
3410 IF LF% < 10 THEN LF$ = "00"
3420 IF LF% < 100 THEN LF$ = "0"
3430 PRINT LF$: PRINT D$: PRINT D$;"OPEN";FFS;".DAT": PRINT D$:"POSITION";FFS;".DAT,R"; VAL (TE$) - 1: PRINT D$;"WRITE": FFS;".DAT": REM CHANGE LENGTH AND ENTER DATA
3440 FOR I = 1 TO SC%: PRINT A$(I): NEXT I: PRINT D$
3450 PRINT D$;"OPEN";FFS;".DAT": PRINT D$:"POSITION";FFS;".DAT,R";BB% + 11
3460 PRINT D$;"WRITE";FFS;".DAT": PRINT SC%; GOTO 3520: REM CHANGE # OF STRING
3470 REM DELETE DATA IN A FIELD
3480 INPUT "FMS> PLEASE GIVE THE FIELD NUMBER TO BE DELETED ": BB%
3490 IF BB% > 10 OR BB% < 1 THEN PRINT G$;"***ERROR! FIELD NUMBER HAVE TO BE": PRINT "BETWEEN 1 AND 10 ***": GOTO 3480
3500 PRINT D$;"OPEN";FFS;".DAT,D2": PRINT D$:"POSITION";FFS;".DAT,R";BB% + 1: PRINT D$:"WRITE";FFS;".DAT": PRINT 0: PRINT D$: PRINT D$;"CLOSE"
3510 PRINT D$: PRINT: PRINT "FMS> DATA IN FIELD# "BB%": HAS BEEN DELETED"
3520 PRINT D$: PRINT: PRINT "FMS> CREATE OR MODIFY OR DELETE DATA": INPUT " IN ANOTHER FIELD? (Y OR N) "ANS$
3530 IF ANS < > "Y" AND ANS < > "N" THEN GOTO 3520
3540 IF ANS = "Y" THEN GOTO 200
CODES

3550 PRINT "DS:";"CLOSE";"FFS";".DAT"; GOTO 250
3560 REM
3570 REM DISPLAY DATA IN FRAME
3580 REM
3590 CALL - 936: PRINT "DISPLAY /
PRINT DATA USING A FRAME": PRINT
"="*"=="="=="="*"=="="=="="="

3600 A% = Z%
3610 PRINT : PRINT "FMS> PLEASE
INDICATE YOUR CHOICE BY": PRINT
"ENTERING THE NUMBER": PRINT
"1. DISPLAY DATA ON
THE SCREEN": PRINT "2. PRINT
DATA ON THE PRINTER": INPUT
ANS
3620 IF ANS < > "1" AND ANS < > 
"2" THEN GOTO 3610
3630 IF ANS = "2" THEN PR% = W%:
REM PRINT THE FORM ON THE
PRINTER
3632 PRINT : INPUT "FMS> PLEASE
SPECIFY THE FORM NAME : ";FFS
S:FO$ = FF$S
3633 PRINT DS:""OPEN TFORM,L20,D1
";TEMP = Z%
3634 PRINT DS:""READ TFORM,RO"": INPUT
K%: PRINT DS: IF K% = Z% THEN
PRINT : PRINT G$:"*** ERROR
! FORM HAS NOT BEEN": PRINT
"DEFINED YET ***": PRINT
DS: PRINT DS"CLOSE": FOR PAU
SE = 1 TO 1500: NEXT PAUSE: GOTO
250
3635 FOR I = 1 TO K%: PRINT DS;"
READ TFORM,R":I: INPUT FYS: 
FYS = FF$S THEN TEMP = I
3636 NEXT: PRINT DS
3637 IF TEMP = Z% THEN PRINT : PRINT
G$:"*** ERROR! FORM HAS NOT
BEEN": PRINT "DEFINED
YET ***": PRINT DS: PRINT DS
"CLOSE": FOR PAUSE = 1 TO 1
500: NEXT: GOTO 3632
3640 PRINT : INPUT "FMS> PLEASE
SPECIFY THE FRAME NAME : ";FFS
3650 PRINT DS:""OPEN TFRAME,L20,D1"
";TEMP = Z%
3660 PRINT D$; "READ TFRAME,RO": INPUT KS: PRINT D$: IF KS = Z$ THEN PRINT G$; "*** ERROR.
   ! FRAME HAS NOT BEEN DEFINED
**": PRINT D$: PRINT D$; "CLOSE": FOR PAUSE = 1 TO 1500: NEXT PAUSE: GOTO 250
3670 FOR I = 1 TO KS: PRINT D$; "READ TFRAME,R": I: INPUT FY$: IF FY$ = FF$ THEN TEMP = I
3680 NEXT I: PRINT D$
3690 IF TEMP = Z$ THEN PRINT: PRINT G$; "*** ERROR! FRAME HAS NOT BEEN DEFINED **": PRINT D$; "CLOSE": FOR PAUSE = 1 TO 15 CO: NEXT I: GOTO 3640
3710 ON ERR GOTO 3730: REM CHECK IF THE RIGHT DISKETTE IS IN THE DISK DRIVE
3720 PRINT D$; "UNLOCK": FO$: "DAT,D2": POKE 216,0: GOTO 3740
3730 PRINT: PRINT "FMS> PLEASE INSERT THE DISKETTE WHICH CONTAINS THE FILE "; FO$: D AT INTO": PRINT " DISK DRIVE 2 THEN PRESS ANY KEY": GET AN$: POKE 216,0: PRINT D$: GOTO 3740
3740 PRINT D$: "OPEN": FF$: ".COR,L 50,D1": PRINT D$: "READ": FF$: ".COR,RO": INPUT NB$: NB$ = VAL <NB$: FOR J = 1 TO NB$: INPUT N$(J): N%(J) = VAL (N$(J)): NEXT
3750 FOR J = 1 TO 10: FOR M = 1 TO 100: P$(J,M) = "": NEXT M: NEXT J
3760 PRINT D$: PRINT: INPUT "FMS> HOW MANY FIELDS DO YOU WANT TO OUTPUT? (0 MEANS ALL) ": J$
3770 IF J$ = 0 THEN, GOTO 3900: REM ALL BLOCKS
3780 PRINT: PRINT "FMS> PLEASE SPECIFY THE FIELD NUMBERS"
3790 FOR I = 1 TO J$: INPUT 'X%(I)": NEXT I
3800 L = 1
3810 A% = Z$
3820 FOR I = 1 TO NB$

The system prompts for the number of the fields to be output.
3830 IF X%(L) = N%(I) THEN A% = W%
3840 NEXT I
3850 IF A% = .Z% THEN PRINT G$;
   ***ERROR! BLOCK ";BC%(L)" HAS NOT BEEN DEFINED ***: GOTO 3760
3860 L = L + 1: IF L < = J% THEN GOTO 3810
3870 NB% = J%
3880 FOR I = 1 TO NB%: N%(I) = X%(I): NEXT
3890 REM READ COORDINATES
3900 FOR I = 1 TO NB%
3910 PRINT D$;"READ";FF$;".COR", R%;N%(I);K% = N%(I)
3920 INPUT X1%(K%); INPUT X2%(K%);
   INPUT Y1%(K%); INPUT Y2%(K%): NEXT: PRINT D$:
3930 ON ERR: GOTO 3950: REM READ DATA
3940 PRINT D$;"VERIFY";FOS;".DAT
   ,D2": POKE 216,0: GOTO 3960
3950 PRINT "FMS> PLEASE INSERT THE DISKETTE WHICH CONTAINS THE FILE ";FFS;".DATA INTO": PRINT "DISK DRIVE 2 THEN PRESS ANY KEY": GET AN$: POKE 216,0: GOTO 3930
3960 I = 1
3970 IF I > NB% THEN GOTO 4120
3980 K = N%(I)
3990 PRINT D$; PRINT D$;"OPEN";FOS;".DAT,D2": PRINT D$;"POSITION";FOS;".DAT,R%;N%(I) + 1
4000 PRINT D$;"READ";FOS;".DAT": INPUT AN$: IF AN$ = "O" GOTO 4110: REM NO DATA IN THE FIELD
4010 PRINT D$;"POSITION";FOS;.DAT,R9": PRINT D$;"READ";FOS;".DAT": INPUT CC$: PRINT D$;"POSITION";FOS;".DAT,R9": PRINT D$;"READ";FOS;".DAT": INPUT OS$: PRINT D$:
4020 M% = 1: W1% = W%: CC% = VAL (CC$): REM W1%=W WHEN SPACE
4030 OS$ = VAL (OS$) - 1

Check if the coordinates of the block is defined

Read the coordinates of the blocks

Check if there is any data in the field

Read the # of strings and the start address of the data in the field

79
CODES

4040 PRINT D$:"OPEN";FOS$:".DAT";
4050 FOR L = 1 TO CC%; INPUT A$(L)
4060 FOR J = 1 TO LEN(ZA$): C1$ = MID$(ZA$,J,1)
4070 IF C1$ < > " " THEN P$ (K,M%) = P$ (K,M%) + C1$: W1% = Z%
4100 M% = M% + 1
4110 NEXT J: GOSUB 4190
4120 NEXT L: P$% (K) = M%
4130 S = LEN (P$% (K,M%)): IF S > 3 THEN GOTO 4190
4140 IF RIGHTS (P$% (K,M%),3) = " -1" THEN P$% (K,M%) = ":": GOTO 4220
4150 IF RIGHTS (P$% (K,M%),3) = " +2" THEN P$% (K,M%) = CHR$(34): GOTO 4220: REM CHR$(34) IS A QUOTATION MARK
4160 IF RIGHTS (P$% (K,M%),3) = " +3" THEN P$% (K,M%) = " "
4170 GOTO 4220
4180 REM CONVERT THE SYMBOLS INTO ACTUAL CHARACTERS
4190 IF RIGHTS (P$% (K,M%),3) = " -1" THEN S = S - 3: S$ = LEFT$(P$% (K,M%),S):P$% (K,M%) = S$ + ":": GOTO 4220
4200 IF RIGHTS (P$% (K,M%),3) = " +2" THEN S = S - 3: S$ = LEFT$(P$% (K,M%),S):P$% (K,M%) = S$ + CHR$(34): GOTO 4220
4210 IF RIGHTS (P$% (K,M%),3) = " +3" THEN S = S - 3: S$ = LEFT$(P$% (K,M%),S):P$% (K,M%) = S$ + ":":
4220 RETURN
4230 IF PR% = 2% THEN GOTO 4560: REM DISPLAY THE FORM ON THE SCREEN
4240 REM CONVERT THE DATA LINE WISE FOR THE LINE PRINTER

COMMENTS

Read the data in the field

Convert the symbols into comma, quotation mark and colon
4250 YY% = Z%: FOR P = 1 TO 22: FOR Q = 1 TO NB%: K = N% (Q): FOR S = 1 TO 10: ES (P, K, S) = \"\": NEXT S: NEXT Q: NEXT P
4260 FOR L = 1 TO NB%: K = N% (L)
4270 WW% = X2% (K) - X1% (K) + 1: YY%
% = Z%: D = Y1% (K): S = Z%
4280 FOR M = 1 TO PS% (K)
4290 YY% = YY% + LEN (PS (K, M))
4300 IF YY% < WW% THEN YY% = YY%
 + 1: S = S + W%: ES (D, K, S) =
 PS (K, M) + \"\": GOTO 4340
4310 IF YY% = WW% THEN S = S + W%
%: ES (D, K, S) = PS (K, H): GOTO 4340.
4320 D = D + 1: S = W%: ES (D, K, S) =
 PS (K, M) + \"\",
4330 YY% = LEN (PS (K, M)) + 1
4340 NEXT M
4350 YY% = Z%: NEXT L: CALL 936
4360 IF PR% = W% THEN PRINT \"FM S\" PLEASE CONNECT THE PRINTER R THEN \"\": PRINT \"PRESS A NY KEY\": GET A$: PRINT D$: PRINT D$: \"PR#1\".
4370 REM SORT THE BLOCK IN INC REASING TOP LEFT X COORDINATE FOR THE LINE
4380 FOR D = 1 TO 22: H = Z%
4390 FOR L = 1 TO NB%: K = N% (L)
4400 IF ES (D, K, 1) = \"\" THEN GOTO 4420
4410 H = H + 1: F% (H) = K
4420 NEXT L: IF H = Z% THEN GOTO 4520
4430 IF H = 1 THEN GOTO 4500
4440 G = H - 1: H1 = H - 1: FOR I = 1 TO G
4450 FOR J = 1 TO H1: REM PRINT THE DATA
4460 IF X1% (P% (J)) > X1% (F% (J) +
1) THEN TEMP = F% (J): F% (J) =
F% (J + 1): F% (J + 1) = TEMP
4470 NEXT J
4480 H1 = H1 - 1
4490 NEXT I
4500 FOR N = 1 TO H
CODES

4510 HTAB X1%(F%(N)): VTAB D:K =  
F%(N): FOR S = 1 TO 10: PRINT  
E$(D,K,S)): NEXT S: NEXT N  
4520 PRINT "": NEXT D  
4530 IF PR% < > W% THEN FLASH  
: HTAB 1: VTAB 23: PRINT "PR  
ESS ANY KEY AFTER VIEWING": GET  
ANS: NORMAL  
4540 FOR I = 1 TO CC%A$(I) = "":  
: NEXT I  
4550 PRINT D$: PRINT D$:"PR#0":P  
R% = Z%: GOTO 250  
4560 YY% = Z%: CALL - 936  
4570 FOR L = 1 TO NB%:K = N%(L):  
REM KEEP THE WHOLE WORD T  
OGETHER  
4580 HTAB X1%(K): VTAB Y1%(K):W%  
%= X2%(K) - X1%(K) + 1:YY% =  
Z%  
4590 FOR M = 1 TO PS%(K):YY% = Y  
% + LEN (P$(K,M))  
4600 IF YY% < W% THEN PRINT P$(  
K,M): SPC( 1):YY% = YY% + 1  
: GOTO 4640  
4610 IF YY% = W% THEN PRINT P$(  
K,M): GOTO 4640  
4620 Y1%(K) = Y1%(K) + W%: HTAB X  
1%(K): VTAB Y1%(K)  
4630 YY% = LEN (P$(K,M)) + 1: PRINT  
P$(K,M): SPC( 1)  
4640 NEXT M:YY% = Z%: NEXT L: GOTO  
4530  
4650 REM  
4660 REM DISPLAY FRAME LAYOUT  
4670 REM  
4680 SU% = Z%: REM FROM SUBROUT  
INE SU%=W%  
4690 CALL - 936: PRINT "DISPLAY  
FRAME LAYOUT": PRINT "====  
====="  
4700 PRINT : INPUT "FMS> PLEASE  
SPECIFY FRAME NAME : ":FF$:F  
A$ = FF$  
4710 ONERR GOTO 4730  
4720 Y% = Z%: PRINT D$:"UNLOCK":F  
FS$; ".COR,D1": POKE 216,0: GOTO  
4740  
4730 Y% = W%: POKE 216,0: GOTO 47  
40: REM NOT EXIST

COMMENTS

Keep the whole word together

Option 5. DISPLAY FRAME LAYOUT

The system prompts for the frame name

Check if the frame exists
CODES

4740 IF Y% = W% THEN PRINT : PRINT G$: "**ERROR! FRAME NAME DOES NOT EXIST **" : FOR PAUSE = 1 TO 1500: NEXT PAUSE: GOTO 250

4750 CALL ' - 936: FOR I = 1 TO 4 0: HTAB I: VTAB 1: IF I < 10 THEN PRINT I

4760 IF I > 10 THEN I$ = STR$(I): I$ = RIGHT$(I$, 1): PRINT I$

4770 NEXT I

4780 FOR J = 1 TO 22: VTAB J: HTAB 1: IF J < 10 THEN PRINT J

4790 IF J > 10 THEN J$ = STR$(J): J$ = RIGHT$(J$, 1): PRINT J$

4800 NEXT

4810 IF NE% = 1 AND C1% = 1 THEN HTAB X1%(1): VTAB Y1%(1): PRINT "*

4820 IF NE% = 1 THEN HTAB 1: VTAB 23: FLASH: PRINT "PRESS ANY KEY AFTER VIEWING" : GET AS: NORMAL : RETURN

4830 PRINT DS$; "OPEN"; FF$: ".COR,L 50.D1"

4840 PRINT DS$; "READ"; FF$: ".COR,R 0": INPUT NB%: FOR I = 1 TO NB%: INPUT N%$: NEXT I

4850 FOR I = 1 TO NB%


4865 IF X1%(L) = X2%(L) AND Y1%(L) = Y2%(L) THEN HTAB X1%(L): VTAB Y1%(L): PRINT L: GOTO 4990

4870 IF Y1%(L) = Y2%(L) THEN HTAB X1%(L): VTAB Y1%(L): PRINT L: FOR J = X1%(L) + 1 TO X2%(L): VTAB Y1%(L): HTAB J: PRINT ".": NEXT: GOTO 4990

4880 IF X1%(L) = X2%(L) THEN HTAB X1%(L): VTAB Y1%(L): PRINT L: FOR J = Y1%(L) + 1 TO Y2%(L): HTAB X1%(L): VTAB J: PRINT "": NEXT: GOTO 4990

4890 IF X1%(L) = 1 AND Y1%(L) = 1 THEN CO% = W%: GOTO 4930

COMMENTS

Display the scale on the top of the screen

Display the scale on the left of the screen

Display a "*" if only the coordinates of the top left corner of the block is defined

Read the block numbers that have been defined

Read the coordinates of the block numbers that have been defined

Display the boundary of the blocks
4900 IF X1%(L) = 1 THEN CO% = 2:
  GOTO 4920
4910 IF Y1%(L) = 1 THEN CO% = 3:
  GOTO 4930.
4920 VTAB Y1%(L): FOR J = X1%(L)
  + 1 TO X2%(L): HTAB J: PRINT
  "": VTAB Y1%(L): NEXT
4930 VTAB Y2%(L): FOR J = X1%(L)
  + 1 TO X2%(L): HTAB J: PRINT
  ":": VTAB Y2%(L): NEXT: IF
  CO% = W% THEN GOTO 4970
4940 IF CO% = Z% THEN HTAB X1%(L):
  FOR J = Y1%(L) TO Y2%(L)
  : VTAB J: PRINT ":": HTAB
  X1%(L): NEXT
4950 IF CO% = 3 THEN HTAB X1%(L):
  FOR J = Y1%(L) + 1 TO Y2%
  (L): VTAB J: PRINT ":": HTAB
  X1%(L): NEXT
4960 IF CO% = Z% OR CO% = 2 THEN
  HTAB X2%(L): FOR J = Y1%(L)
  TO Y2%(L): VTAB J: PRINT ":
  HTAB X2%(L): NEXT
4970 IF CO% = W% OR CO% = 3 THEN
  HTAB X2%(L): FOR J = Y1%(L)
  + 1 TO Y2%(L): VTAB J: PRINT
  ":": HTAB X2%(L): NEXT
4980 HTAB \(X1%(L) + 1)\): VTAB \(Y1%
  (L) + 1)\): PRINT L
4990 \HTAB 1: VTAB 23: CO% = Z%: NEXT
1: IF C1% = W% THEN HTAB X4%
  : VTAB Y%: PRINT "x"
4995 PRINT D$: "CLOSE"
5000 PRINT D$: VTAB 23: HTAB 1: FLASH
  : PRINT "PRESS ANY KEY AFTER
  VIEWING": IF SU% = Z% THEN
  GET AN$: NORMAL : GOTO 250
5010 PRINT D$: "CLOSE"
5020 GET A$: NORMAL : RETURN
5030 END

Display the block number inside the boundary of the block.
REFERENCES


