Order Number 9304742

Design and development of a document imaging system: IMAGE TRAC™

Tse, Stephen Kam Hong, Ph.D.

City University of New York, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Design and Development of

A Document Imaging System: IMAGE TRAC (TM)

by

Stephen Kam Hong Tse

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1992

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

6/30/92 — Date

Michael Anshel — Chair of Examining Committee

7/1/92 — Date

Stanley Habib — Executive Officer

STANLEY HABIB, CCNY

STEVE LUCCI, CCNY

IRA RUDOWSKY, BANKERS' TRUST

Supervisory Committee

The City University of New York

ABSTRACT

Design and Development of

A Document Imaging System: IMAGE TRAC (TM)

by

Stephen Kam Hong Tse

Adviser: Professor Michael Anshel

The problem with standard data processing of information stored as text is quite simply overload. Today's office generates and processes text from memos, fax, mail, and numerous other media. In such a context , it becomes quite difficult to locate and retrieve information in the form desired by the user. The current state of affairs is well discussed in [1].

The typical user is involved with such typical processes as making a copy of the document to send elsewhere, filing the document, or signing and commenting on the

document. ALL these tedious but standard office procedures are now done manually.

It is the thesis of this dissertation that digitizing the text and computerizing the output eliminates these manual chores. A system for document image management (IMAGE TRAC™) that we have designed and developed works quite well at these tasks. This system allows users to scan in documents and then digitizes them, thus preparing it for storage on an optical disk. The techniques used for digitizing documents center around the use of the Gray code [2] for data compression.

[1] C. L. Benier and A. N. Yerkey, "Problems of Information Science," "Reading Overload," and "Condensed Surrogates," in *Cogent Communication: Overcoming Reading Overload*, Greenwood Publishing Company, Westport, CT, 1979, pp.18-22, 31-51, and 64-117.

[2] Z. Kohavi, "Binary Codes" in *Switching and Finite Automata Theory*, McGraw Hill, New York, NY, 1978, pp. 10-14.

ACKNOWLEDGMENTS

No words can express my gratitude to my mentor, Professor Michael Anshel. Throughout my efforts on this dissertation, he has been my source of encouragement and inspiration. His guidance has given significant direction to my research and career, for which I will forever be indebted. Thanks also to members of the supervisory committee, Professor Habib, Professor Steven Lucci, and Dr. Ira Rudowsky, for their time in carefully reading the thesis and in giving valuable criticism. I also appreciate the assistance of Dr. Joseph Fulda, for his warm and resourceful editing of my manuscript which spanned the area of writing style, focus, and structure.

I owe an immeasurable debt to my parents, brothers, and sisters, for their constant support and encouragement so my life can be fully explored.

Most of all, my gratitude goes to my beautiful, loving wife, Regina Chen-Te Lee Tse. Her love and spiritual nourishment can never be returned in full measure. Without her, my life would never be as fulfilled.

Many thanks also go to Wai-Ki Hui, Bell Laboratories, for his valuable technical comments and suggestions. To Stephen Schobel, Bell Laboratories, and Brian Grant, AT&T Federal Systems Advanced Technology, for their encouragement and valuable

suggestions. To Dr. Chuck C. T. Cheng, CTC Systems Inc., for his valuable hardware and networking information. To James K. Tse, NYNEX, for his valuable information on communication. To Dr. Mark Eng, Glamour Look Ltd., for his detailed information on cosmetics and manufacturing. To Pauline Chang Tse, Manhattan Saving Bank, for her valuable banking information. To Susana K. Luke, Bankers Trust Co., for her financial and loan information. To Richard Seto, BIG D Oilfield Equipment Co., for his valuable information on the petroleum industry. To Peter Tse, Korean Airlines, for his airline and flight schedule information. To Sheila Yi-Fang Lee, for her constant update on Japanese technology and the insurance industry. Also, to Sabina M. Rosenthal, C.P.A., for her tax and financial information. To Terry Morris, Captronic International, for sharing her valuable secretarial experience and her knowledge of office operations. To Dr. Hollington Y. Tong, Summit Podiatry Group, for medical industry information. To Rosaleen Light, Cowell Banker Real Estate, for sharing her knowledge of real estate needs.

TABLE OF CONTENTS

I. OVERVIEW

One picture is worth more than ten thousand words.

--Anonymous, Chinese proverb.

We begin by placing our work within the context of modern science and engineering. With the discovery of the right-brain/left-brain dichotomy [11], the former more visual and holistic and the latter more verbal and sequential, it is clear that the distinction between pictures and words, especially with respect to information-bearing, is not a mental artifact but is rather a true representation of fundamentally different types of cognition. Because both pictorial and textual data are critical to human decision-making, the conversion process shows up in the oddest places. (Lawyers, for example, know that an overall situation--a picture of sorts--can usually be reduced to the answers to a sequence of yes-or-no questions.) At the date of this writing, no system exists which can use and interpret data holistically without internal conversion to digital form. The alternative, conversion to digitized form, is a task for which computers are very well-equipped. Digitization makes computation possible for a whole range of visual data otherwise inaccessible by computers.

IMAGE TRAC™, the subject of this dissertation, is an image processing system: it takes images as inputs and stores them in compressed, digital form for easy access and manipulation. The purpose of having a document image system such as IMAGE TRAC is to minimize office paperwork and the concomitant waste of employees' time and office

space. Moreover, paper is not a good medium for the long-term storage of information. Optical storage, on the other hand, is guaranteed by manufacturers to last a century and is both simple and efficient. This thesis describes IMAGE TRAC in the context of existing imaging systems and with an eye towards still greater functionality. The research for this thesis thus had three components: (1) Bibliographic research, (2) Field work, and (3) The design and development of a complex software system which took a full man-year to complete. It is important to note, therefore, that unlike a mathematics dissertation where the thesis contains the results, this engineering thesis reports on the research which went into IMAGE TRAC.

We briefly outline the purpose of each of the subsequent chapters. Chapter 2 presents a large number of imaging systems--already on the market--and concisely sums up the advantages and disadvantages of each. This bibliographic research was of substantial assistance in designing and developing a superior system as *users* see it. Potential users who were interviewed for this project include accountants, bankers, insurance professionals, real estate brokers, and secretaries. Together with an understanding of what the weaknesses of current systems are, this field work served as an invaluable guide in designing and developing IMAGE TRAC.

Chapter 3 presents IMAGE TRAC from the user's perspective, but contains enough documentation to allow systems' professionals a glimpse of its novel internal workings (especially the use of masking). In line with the theme of this study, Chapter 3 is presented simultaneously in pictographic form and in textual form. The 31 figures in the chapter provide a hands-on feel to the exposition, and are an integral part of this work.

Chapter 4 suggests that imaging systems may be evaluated along several dimensions and shows that IMAGE TRAC is, by and large, well above the current standards for such systems. More field work confirmed the veracity of the claims here put forth: A prototype of IMAGE TRAC was given an extensive trial run and performed splendidly, surpassing expectations. The trial run was carried out at New Era Electric Cooperative in Athens, Texas. Finally, Chapter 5 suggests future enhancements to IMAGE TRAC and also discusses future systems that IMAGE TRAC may spawn.

II. SURVEY OF PREVIOUSLY DEVELOPED SYSTEMS

A. STANDARD DOCUMENT IMAGE MANAGEMENT AND PROCESSING

In a nation-wide survey [1], it was found that 95% of information stored in the United States is stored on paper. Since the 1930's, people have been trying to store documents in pictorial form, but the usefulness of such systems has largely been limited to the banking industry and libraries.

To enhance office productivity, automation in retrieving and storing documents--a reduction in paper processing--is essential. Document image management systems were the theme of a recent issue of ***COMPUTER*** [2]. Related problems of building intelligent subsystems for document image database systems are discussed in [3].

A view of existing document image database management environments is given in Figure 1. The typical document image management system performs the following functions:

(1) CAPTURE - Digitize the image into computer-recognizable format.

(2) STORE - Perform compression of digitized image at the outset to save space and allow efficient handling.

(3) INDEX - Provide an appropriate data structure for efficient data retrieval.

(4) RETRIEVAL - Display the image in various media (print, fax, graphic monitors, and on networks) using device-oriented algorithms.

B. SURVEY OF DOCUMENT IMAGE AND OPTICAL STORAGE SYSTEMS

Currently, some systems are available for retrieval and storage of computer-generated data and image capturing. Their strengths and weaknesses follow:

INFOTREEV (now known as TREEV) - Legend Technologies [4].

Strengths - (a) First image-management system (marketed November 1987).
(b) Focused on the financial industry.
(c) Good existing dealer base.

Weaknesses - (a) Only runs on an IBM System/36 environment.
(b) Product has no flexibility: modifications for user-specific requests cannot be performed.
(c) Only used for computer-generated reports; no pictorial/graphics capability.

(d) Limited indexing of data, thus slowing data retrieval.

(e) Requires heavy mainframe disk space.

(f) Not user friendly: very limited help screens and poor menus.

(g) Little or no installation and training support and inferior customer service.

(h) Poor system documentation.

TRIMDEX - Microsize [5].

Strengths - (a) Written in C with an eye to potential portability and flexibility.

(b) Data compression on menu.

(c) Customer choice of record fields for indexing.

(d) Supports both 80- and 132-column display: allows switching between display formats with only one key-stroke.

(e) Focused on the financial industry.

(f) Publicly held company.

Weaknesses - (a) Product offering is available only on a 244MB optical drive.

(b) Limited flexibility in cross-indexing document images and computer-generated reports.

(c) Menus and screens are too complex and cumbersome for users: both menu items and screen function keys are poorly designed, and function keys are not defined consistently among screens.

(d) Poor user documentation.

(e) Poor color coordination. When some messages flash with non-white lettering on a green background, it is difficult to read and can be confusing.

RECALL - Datawork Inc. [6]

Strengths - (a) Strong resemblance to InfoTreev, but more advanced and less expensive.

(b) Separate package for signature verification, which will eventually be combined with the document image system, thus enabling the processing of official documents, and the like.

Weaknesses - (a) Very limited flexibility: cannot easily fix errors or update data.

(b) Available only on IBM mainframes.

(c) Imaging system allows only computer-generated output to be digitized and stored on an optical disk.

(d) Cannot add user-specific functions.

(e) Not user-friendly: fixed and inflexible records.

COIN - Acctex Info Systems. [7]

Strengths - (a) Strong software development team: the product, developed using COBOL, has been entirely rewritten in C to enhance portability and flexibility.

(b) Applications include spreadsheets, signature verification, loan document preparation, teller terminal software, and any other PC-based application: these applications can be run in tandem with COIN.

(c) Excellent software documentation for systems programmers.

Weaknesses - (a) Poor user documentation.

(b) Limited feature functionality: Neither automatic system backup or restore, nor the ability to handle user-specific requests, nor the replacement or addition of pages is provided for.

(c) No specific industry focus, e.g., banking, brokerage or insurance. This "something for every market" entails heavy product overhead.

LASERTRIEVE - BankSystems Company. [8]

Strengths - (a) Flexible: some modifications for user-specific requests can be performed.

(b) User-friendly.

(c) Excellent user documentation.

(d) Good selection of optical disk hardware (Panasonic 5-1/4" optical drive), and interface software (Corel optical driver).

Weaknesses - (a) Only used for computer-generated reports; no pictorial/graphics capability.

(b) Poor selection of PC hardware (LaserTrieve's PC clone is quite unreliable).

(c) No product installation and maintenance support for either hardware or software.

(d) No version that can run on a network.

(e) Limited indexing of data, thus slowing data retrieval.

SCANBASE - Scanbase Graphics Inc. [9]

Strengths - (a) Endorsed by AT&T.

(b) Flexible: Provides customization templates with seven choices of database and screen formats, which can be tailored to individual requirements.

(c) Variety of mass storage devices support.

(d) The system is entirely menu-driven.

(e) The system utilizes standard hardware, available from most retailers and distributors.

(f) Software is dBASE III-compatible.

(g) Automatic backup of databases to portable media (e.g., diskettes and platters) and monitoring of backups.

(h) Ease of recovery in the event of a system failure.

Weaknesses - (a) Captured document images are converted to PCX (i.e., PC Paintbrush format), which doesn't conform to industry standards.

(b) Customization templates are limited to seven (industries) which may not include the needs of the many diverse small businesses, which might use an image-processing system to great advantage.

(c) Backup on several different media creates situations when locating the proper medium for a restore is often delayed and sloppy.

(d) SCANBASE users have to know system implementation details. For example, if a 20MB hard disk is included in the system, the user must specify the storage capacity that is required for storing the image. Since that depends on the medium involved, the system makes demands of users which are unreasonable.

(e) A cumbersome procedure to define the drives and volumes

for storing images is required.

(f) Backups can be performed on several media, but those media have to be "prepared" by the user at the time of creation or backup will prove impossible. This is another example of inappropriately requiring users to interface with the system's internal workings.

(g) A cumbersome procedure to prepare a new platter for use in image storage is also required.

(h) There is no procedure for preparing a partially used optical platter for image storage.

III. OUR DOCUMENT IMAGE MANAGEMENT SYSTEM: IMAGE TRAC™

A. THE IMAGE MANAGEMENT ENVIRONMENT

To improve office management and increase productivity, the control of office paperwork, documents, and other sources of information is the paramount task of modern office management systems.

Different sources of information produce different information presentation forms (Figure 1). Thus, information sources such as paper, microfilm, or microfiche can be viewed through scanners or projectors and presented in a screen image form. If offices are using electronic equipment, such as terminals, word processor systems, and personal computers, then the presentation forms will be changed to text format. For large institutions, information sources may include a mainframe computer, in which case the information may be a part of a database and presented in structured data forms. In the answering service industry, the information sources are various voice machines and the presentation forms are in voice "text".

The best control of all this information is to digitize it and store the result as an electronic image. With the advances in electronic hardware and with sophisticated software, these images can be enhanced, updated, retrieved, and transmitted just as with the original documents, only more quickly and accurately.

Figure 1

The Image Management Environment

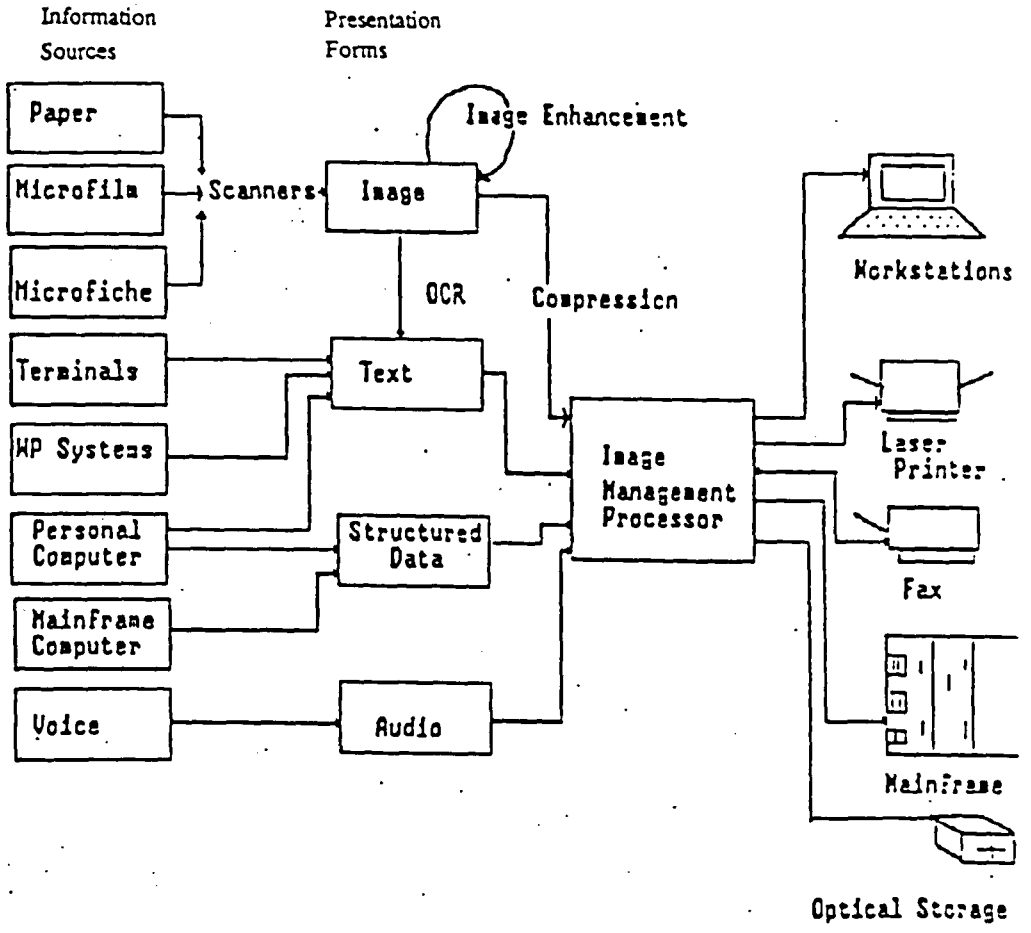


IMAGE TRAC is an electronic document image processing and management system that can eliminate sluggishness and clogging of paperwork in a tangled office environment. The user inputs a document through a scanner. The system then captures the document image by digitizing the input and displaying it in an electronic format. The original document then can be safeguarded in a remote location while the document image will be compressed, transmitted onto a network, or stored for later processing. IMAGE TRAC directs and processes the document image throughout the life of the document (work-flow process). The system's purpose is to change the office work flow from a paper-oriented environment to an image-oriented environment. These changes will lead toward a paperless office and should yield substantial economies.

B. OVERVIEW OF THE IMAGE MANAGEMENT SYSTEM

Any image management system includes four major subsystems. The first of these is the image capture and display subsystem. Once the information sources are determined, the subsystem will use appropriate equipment--a scanner or a modem--to input the source data and digitize them into a computer-recognized image-display format. The input source may be from papers or from electronic data information (EDI).

Once the image is captured, the second subsystem can be applied to compress the digitized display image into a much smaller file with the result that storage is saved and that transmission of the digitized image (and the corresponding file) can be accomplished in much less time. This subsystem highlights some of the principal advantages of using

IMAGE TRAC: space- and time-efficiency.

The third subsystem provides a database with indexing for efficient and rapid data retrieval. All current systems on the market, including IMAGE TRAC™, use relational databases with indexing to the compressed files for efficient and rapid data retrieval.

The fourth subsystem decompresses the retrieved data in a device-dependent manner using device-oriented algorithms. The image can then be displayed on various media, i. e., graphic display, printer, and fax as well as transmitted over a network.

C. THE WORKINGS OF IMAGE TRAC™

1. System Hardware Requirements

The IMAGE TRAC™ document image management system is a PC-based system that takes advantage of the Intel 80386 microprocessor architecture. The system needs a minimum of two (2) megabytes of random access memory (RAM) in order to scan and display an image. To display the original document in image form, a Video Graphics Array (VGA) graphic adapter, which provides screen resolution of a 640-by-480 pixels, must be used. A scanner is needed for document image input. An optical disk and an 40MB (minimum) hard disk are needed for document image storage, and directory- and database-structure information storage. If hard copy is required, a laser printer is highly desirable for preparing attractive paper output.

2. System Design Requirements

The system is designed to store documents using their original structure. Each document has internal indexing on four fields. Every page of the document has a unique internal number assigned by the system. This provides access to documents quickly and with partial information. The system also helps process documents throughout their life cycles. Thus, each document can have multiple pages but has only one "Note Pad" area to keep comments and work-flow information. The "Note Pad" area allows the document to be used by multiple users at different stages of the document's life cycle. In a Local Area Network (LAN) environment, documents can be remotely accessed on the network, allowing simultaneous viewing of the document by parties at different locations.

An optical platter is reserved for the master directory that serves as the table of contents for all documents in the system. If a page in any document is modified, the newer version replaces the old one. Consider, for example, the banking industry. For auditing purposes, all previous versions of a document must be kept on record. Such a storage structure is known as the audit trail. Thus, if an auditor wants to examine the changes made to the original documents, he can follow the audit trail and trace back each change in each document. Audit trail is a feature that is required for information-history look-up and for legal action support.

When optical disks are used in the system, each platter has its directory information kept on the hard disk. This directory information is also contained in the master

directory platter. Likewise, for each user in the system, there is a user document directory.

3. System Architecture

The system's architecture must satisfy all the requirements given in section 2. The system's main module controls the user information database, the Document Image Database, and the Image Manager (see Figure 2).

The Image Manager provides for the manipulation of images via scanner interface functions, printer interface functions, and screen display functions.

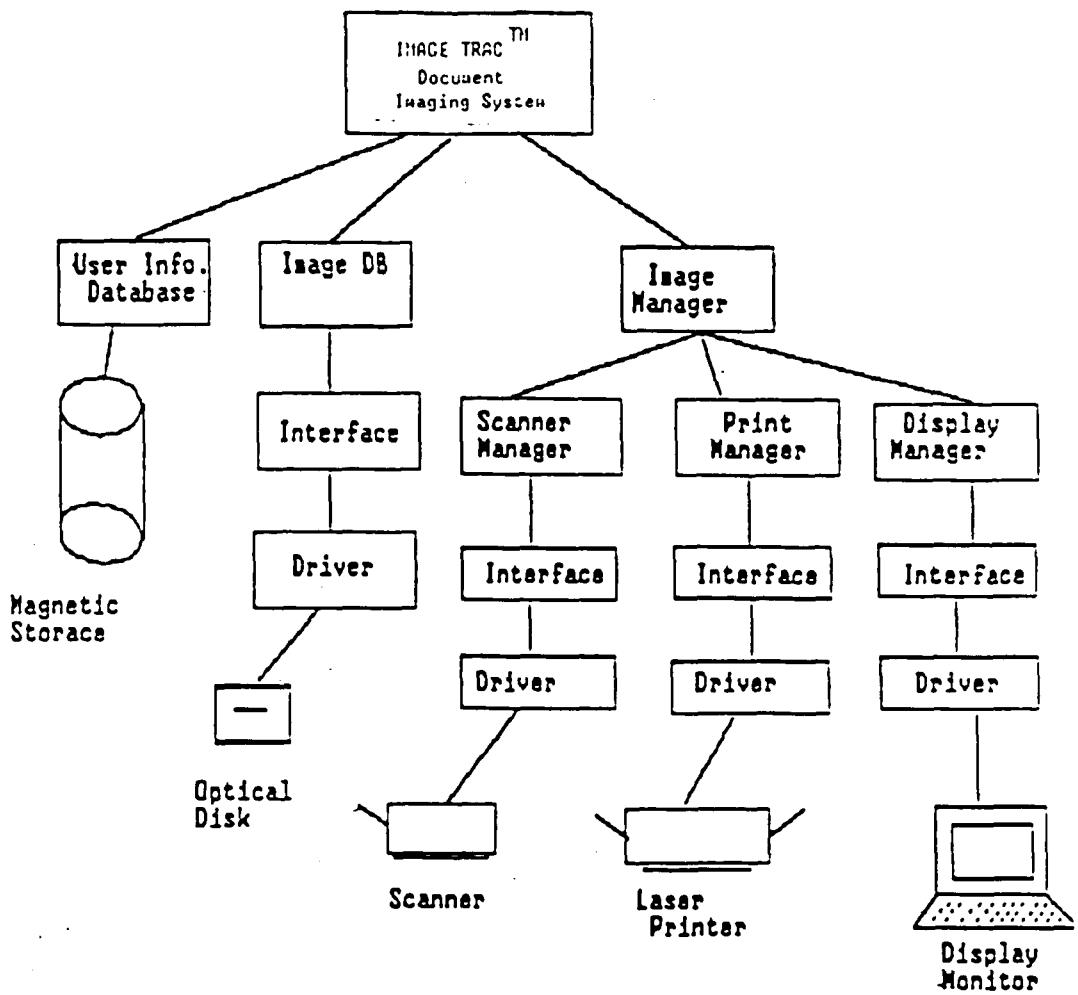
The Document Image Database is the area where all the documents are kept. IMAGE TRAC is dBASE III-compatible but does not use dBASE III in order to avoid the substantial overhead of dBASE III. All database files are custom-designed and indexed. With optical disks as storage media, multiple platter and master-directory management is implemented. This module also provides all the optical-drive interface functions.

The user information database contains all the user information created by the system administrator when the user obtains a login-id from the administrator.

4. Image Manager

Figure 2

System Architecture



We will discuss screen display functions, printer interface functions, and scanner interface functions.

a. Screen Display Design

IMAGE TRAC is a window- and menu-driven system. When the system comes up, a window with a title and login menu is displayed on the screen. The user then types in his name and password in order to login to the system. After the user logs in, another window opens and contains a menu of actions for the user to choose from. The design of this window and menu are explained in the following paragraphs.

The basic menu window contains three elements, i.e. the QUICK HELP sub-window, the action menu sub-window, and the escape (ESC) and function keys sub-window (Figure 3).

The action menu sub-window can be any width (but no wider than the screen) and any length (but no longer than the screen). It contains heading, borders, and menu items. The individual items of the menu respond to arrow keys and the ENTER key. The up or down arrow keys will move and highlight the menu items. At the same time the QUICK HELP sub-window will display an appropriate explanation of the menu item. The ENTER key will activate the highlighted menu item. The action sub-window provides the user with a choice of actions, dependent on the application.

The QUICK HELP sub-window is the first line after the window title. The sub-

window changes to offer help for each item of the action menu as the arrow key moves up and down through the menu. This provides user friendly, immediate help or system/user instruction interface.

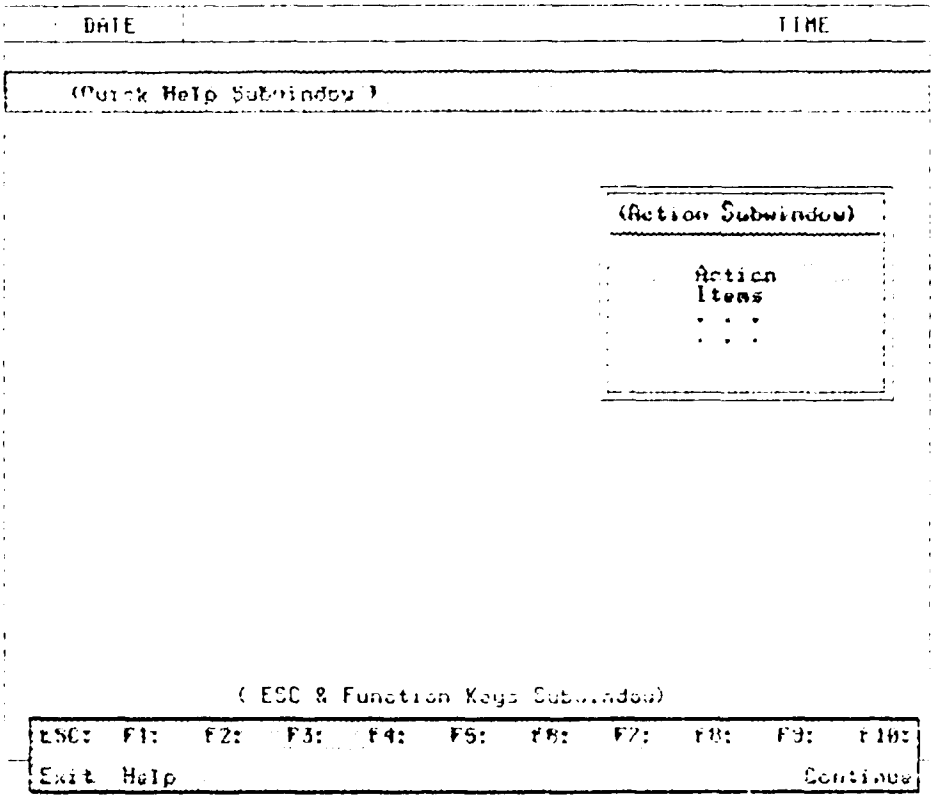
The ESC sub-window, for both the escape (ESC) and function keys, is displayed on the bottom lines of the screen. This sub-window displays action labels corresponding to the ESC, F1, ..., F10 function keys. For example, the EXIT label always corresponds to the ESC key, the HELP label always corresponds to the F1 function key. If the ESC key is pressed, the user exits from the current menu sub-window and returns to the previous menu window or, if at the highest level menu, exits from the system. If the F1 function key is pressed, a help sub-window is displayed. A beeping sound is produced if a function key that has no label is pressed.

This window layout is used for every menu. When a menu item is activated, a second menu sub-window is displayed, and the previous menu window will be masked out. The QUICK HELP sub-window will overlay the previous QUICK HELP sub-window, the ESC sub-window will overlay the previous ESC sub-window, and the action sub-window will overlay all, part or none of the previous action sub-window. This allows the simultaneous display of multiple menu sub-windows. The masking technique provides both faster window display and a uniform user interface.

The majority of people are more video-minded than audio- minded. The famous memory power teacher and TV personality, Harry Lorayne, always emphasizes that if an idea or information is presented to a person in a clear image, the person will

Figure 3

Basic Menu Window Layout



remember the idea easily. If color is added into the picture, the information will be remembered with even greater frequency. IMAGE TRAC uses color, not just text, to convey information. For example, in the system setup window, if a user chooses to re-initialize the database, the first warning sub-window is white-on-lavender. If the user says "yes" to initialize the system, the "ARE YOU SURE?" sub-window is white-on-bright red to indicate urgency. All windows of type **FORM** are double-lined rectangles with white-on-royal blue, while **MENU** sub-windows are double-lined rectangles with turquoise banners and lavender windows. The **SELECT** windows are white-on-turquoise with no borders, while all **REPORT** and **HELP** windows are white-on-green with double-lined borders and are centered in the middle of the screen for easy reading. The background color is royal blue and the date and time are in digital format and are displayed respectively on the left and right upper corners of every screen with white-on-black. Input fields are always in reverse video. I.e., if the background of the window is X-on-Y, the fields accepting input are Y-on-X.

Color also provides a sense of depth to the system. Windows with banners and shades will serve as nodes fanning out to many windows, i.e., each of its menu items will activate a new window. These actions can be overlaid for up to 5 levels. When a solid-colored window is reached, this is always a **FORM** or **SELECT** window; the user has now reached the bottom level. The user can choose to make a selection and back-fill the **FORM** or type in the information directly.

b. Menu Window Masking Example

For effectively displaying windows, all application menu- windows should be created and placed in core memory beforehand. When needed, a menu window can be laid on top of others--like putting on a mask. Whatever is not covered will still be displayed on the screen. To create a masking window, first a window menu file and a masking key file are created. These two ASCII files are called and compiled by a C program, creating a binary masking module. Many different masking window modules are created for the application, efficiently loaded into memory, and rapidly displayed on the screen when needed.

Figure 4 depicts an ADMIN-menu window file. The first line provides the sub-window heading information. It starts with a flag, *S*, to signify that this provides sub-window screen information. The possible flags are: *S*=screen information, *ST*=screen attribute information, *T*=text information, and *F*=input field information. The columns in sequence are as follows: the sub-window mask-id code, e.g. *301*, range from 1 to 999; the title or masking name column, e.g. "admin"; the sub-window masking type, e.g. **MENU**. There are four types of sub-window structures, viz., the **MENU**, the **FORM**, the **REPORT**, and the **REPORT-SELECT** structures. The **MENU** structure allows users to choose menu items for action. This type of structure leads the user to the next sub-window action. The **FORM** structure provides an input form for the user to input information to the system. The system expects information to be entered on the screen, and also responds when an error occurs in the input. The **REPORT** structure, like the help sub-window discussed earlier, is a window that contains information for reading only. No user action is needed. The **REPORT-SELECT** structure, e.g. a

customer name list for an application, is a sub-window that contains a list of names for user selection. For example, in filling out a customer information form, there are two possibilities. One is to directly type in all the necessary data, while the other is to open a sub-window that consists of a list of items for selection. The user can highlight the selected information and simply press the enter key to transfer the information directly into the previous form. This type of structure also allows backward movement to previous sub-window action. The next two columns are reserved for status (0-99) and field-length checking. However, in the present application these two columns are not checked. The last two columns in this line contain the sub-window row and column range (last row, last column), and the sub-window color area range (last row, last column).

The second line in Figure 4 provides the color attribute and screen-layout information. It starts with the flag *ST*. The second column gives the sub-window color code which ranges from 0 to 127, with a -1 (default) indicating the last-mentioned color code. The third column specifies the banner color attribute. The fourth column is the sub-window starting coordinates given in the traditional way as (x,y). The fifth column is reserved for a code to continue the action in progress on the next masking window. In the present application, this field is not used. The sixth column is the name of a database of attributes or departments or any other necessary information items for the file. The seventh column is reserved for item codes and is not used in the present application. The eighth column gives the box-type, i.e. 0=no border, 1=single-line border, 2=double-line border. The ninth column gives the shade code: 0=no shade, 1=shade. The last column specifies the banner size (starting row, ending row).

The third line is the masking key file name line and is specified by the flag "KEY". The key file name follows the flag, here "admin.key".

The lines following the key in the file depicted in Figure 4 are the menu-item specification lines. The initial *T* flag means that this line is a text line. The second column specifies the color attribute of this line, with *-1* indicating use of the most recent color. The third column gives the relative starting position of this text item: For example, (0,4) is an indication to start the text at the top of the sub-window and four positions to the right. The last column is the subject of the menu item, except for the top line. For the top line, this column is the title of this sub-window.

After the description of the last menu item line, input-field description lines follow. Input-field description lines specify the attributes of the input information from the screen. The initial *F* flag means that this is such a line. The numbers in the second column correspond to the items in the menu. The third column is the input field identifier. The s_i's in this column correspond to the subjects in the menu. Then follows (x,y), which specifies the display coordinates of this input field. The *c* specifies that this input field is character data; other valid values in this field are *d* (date format), *f* (floating point), and *i* (integer). The *n* in the sixth column means that the data cannot have varying types; the user is constrained by the type specified in the fifth column. A *y* specified in this field would allow heterogeneous data. The following field specifies the maximum length of the input line (here, 23). In the next column, the *Y* indicates that echoing back the user's input to the screen is desired, while an *N* would mean no

Figure 4 - Admin Window Masking File

```
S   301   admin   MENU    6   7    11,36   11,36
ST  95    62      10,42   0   0    0       2       1    0,1
KEY  "ADMIN.KEY"
T   -1    0,4     "System Administration"
T   -1    2,4     "User Maintenance"
T   -1    3,4     "Change ADMIN Password"
T   -1    4,4     "Security Control"
T   -1    5,4     "Folder Subject Maintenance"
T   -1    6,4     "Re-index Database"
T   -1    7,4     "Pack Database"
F   1     s1      2,4     c   n    23      y    n
F   2     s2      3,4     c   n    23      y    n
F   3     s3      4,4     c   n    23      y    n
F   4     s4      5,4     c   n    26      y    n
F   5     s5      6,4     c   n    26      y    n
F   6     s6      7,4     c   n    22      y    n
```

Figure 5 - Admin Key File

```
D:  EXIT   Help
F:ESC:QUIT
F:F1:Fhelp
S:1:Fpasswd
S:2:Fcpasswd
S:3:Fat_access
S:4:Fdeptm
S:5:index.msk
S:6:pack.msk
I:1"Add, Modify and Delete Users"
I:2"Change Password and Security Levels"
I:3"Security Control to Different Subjects"
I:4"Add and Modify Subject IDs and Names"
I:5"Re-index Individual Database"
I:6"Pack Individual Database and Index the Database"
```

echoing. In some cases, for example when asking for a user password, there is no echoing back when the user is typing or responding to the menu item. The last column is the protection field. The *n* means that the input field is not protected and that, as a consequence, a user can change this field at any time. An *r* means restricted, and that, as a consequence, only a superuser or a user with administration permission can change any of the input fields of that record.

Figure 5 depicts an Admin Key file. This file specifies the appropriate action to be taken when an item or function key is chosen by the user. The file begins by specifying the appropriate action to be taken when a function key is pressed, along with a label, such as *Exit* or *Help*. The next part of the file corresponds to the menu items (actions). The file continues with an explanation of the menu items for display. The "D:" in the first line indicates that this line is to display the labels used in the ESC sub-window. The next line starts with "F:" and is a function-key line. The "ESC:" or "F1:" corresponds to the function-key name. The *QUIT* or *Fhelp* is the action taken in response to a given function key being pressed. The line that starts with "S:" corresponds to a selection from the menu item sub-window. The succeeding "1:" or "2:" corresponds to the first or second menu items. The "Fpasswd" in the next field will open the user password sub-window and wait for user input, while "Fcpsswd" in the field will open the system password sub-window and allow the system password to be changed. This latter action requires system permission. The lines that start with "I:" provide the QUICK HELP explanation line corresponding to the menu items that are indicated in the next column, e. g. *1* or *2*. The rest of the line (surrounded by quote marks) is the explanation itself.

If any field has a *-1*, this means that the last-mentioned value (the default) is used.

c. User Interface and Screen Layout of System Windows

Simple window layout and efficient user interface is the primary design philosophy for the IMAGE TRAC system. When the system is activated, the login window is displayed (see Figure 6). This is a window that has type FORM, because it expects user input only. When user login and password inputs are both correct, the system goes to the next window - the system main function window. If the user login or password is incorrect, an error message is displayed on the bottom line of the screen. In the following sections we describe the various system windows.

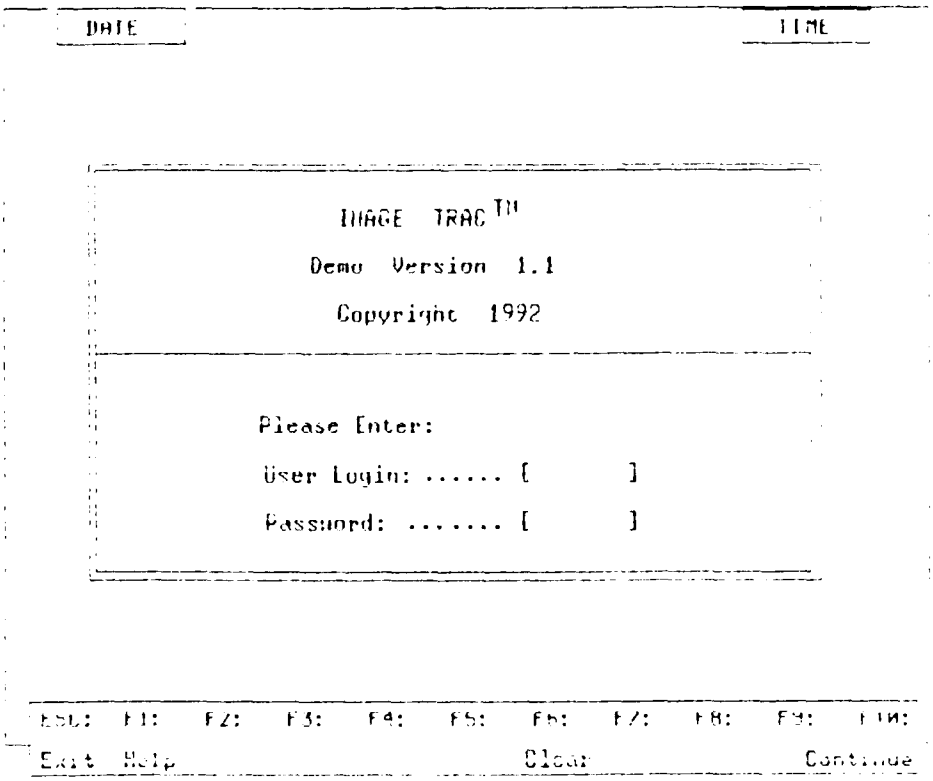
i. The *Login* Window

The *login* window layout is similar to the basic window layout (see Figure 3) except that there is no QUICK HELP sub-window. The action sub-window is enlarged, moved to the center of the screen, and given a multiple-line heading. The ESC sub-window has labels:

Exit -- corresponding to the ESC key and for exit from the system,
Help -- corresponding to the F1 function key and for login help explanation,
Clear -- corresponding to the F6 function key and for the user to clear the screen and re-try his login and password, and

Figure 6

Login Window



Continue -- corresponding to the F10 function key and for advancing to the next window.

This login window is shown in Figure 6. The corresponding login window file and the login key file are shown in Figures 6a and 6b. In Figure 6a, there is no line beginning with "I:" and there is no QUICK HELP sub-window.

ii. The *System Main Function* Window

After a user successfully logs in, the system's main function window is displayed (Figure 7). This simple window layout illustrates the three main system functions, viz. *Retrieve*, *Store*, and *Utilities*. A user can move the arrow keys to indicate his selection and press the ENTER key to begin the action. A quick way to access these functions is to type in the leading character--(*R*, *S*, *U*)--to activate the function. The *Retrieve* function allows users to recover documents previously stored on an optical disk or hard disk. The *Store* function allows users to scan and save documents on either optical disk or hard disk. The utilities are a set of functions, available to both users and the system administrator, for maintaining user information and performing system maintenance. The corresponding masking window file and masking key file are shown in Figures 7a and 7b.

iii. The *Retrieve* Window and Subsequent Functions

Figure 6a - Login Window Files

```
Flag Mask-  Mask-  Type  Status  Field-   Mask-    Color-
     ID     Name        (0-9)   Length   Range    Range
S    301    login FORM    5      20      15,60    15,60

Flag Color Banner Posi- Next-  Database Item Box- Shade Banner
     Attr  Color  tion  Msk-ID Name       Type (0/1) (st,n)
ST   95    63     4,9    0      0        0    2    1     1,6

Flag"Key-File-Name"
KEY  "login.key"

Flag Color-     Relative "Text:
     Attribute  Position  Title or Subject"
T    -1         1,14     "I M A G E  T R A C ™"
T    -1         3,14     "Copyright (C) 1992"
T    -1         4,14     "Elite Design Systems"
T    -1         7,14     "Please Enter:"
T    -1         9,14     "User Login:....."
T    -1         10,14    "Password:....."

Flag Sequence Field  Relative Type Heteroge- Max- Echo Protect
     ID       Name   Position      neous(y/n) len  (y/n)  (y/n)
F    1        login  9,34     c    n          8    y      n
F    2        passwd 10,34    c    n          8    n      n
```

Figure 6b - Login Key File

```
Flag ESC  Lbl- Lbl- Lbl- Lbl- Lbl- Lbl- Lbl- Lbl- Lbl- Lbl-
           1    2    3    4    5    6    7    8    9   10

D:  Exit Help                Clear                    Continue

Flg:Function-Key:Function
F:ESC:QUIT
F:F1:Fhelp
F:F6:Fmask_clear
F:F10:Fchkpass
F:CRL:Fchkpass
```

The processing of documents in an image management system is similar to the filing systems of an office environment. Correct marking of folders and cabinets is required before documents can be stored or fetched. Usually, a folder is assigned to each customer and an identification number, name, address, affiliation, and other information are kept as part of the folder header information for easy filing. This arrangement of the header information facilitates the retrieval process when a particular document is needed. Such an arrangement is developed in IMAGE TRAC as elaborated below.

Four levels of indexing fields are included in the folder header information, of which only the primary indexing field is required. Consider a typical application: the customer ID is used as the primary index, the customer name, customer address, and affiliation are used as the second, third, and fourth level of indexing. When storing a customer document, the unique customer ID field must be filled, because this is the primary indexing field. Thus, using the previous example, the name, the address, and affiliation may be totally or partially omitted when the form is first completed. (Position checking is also implemented so that queries on part of a field can successfully match those in the customer database. Note that position-checking matches column-by-column.) A later customer information modification/update is allowed for all secondary indexing fields. If only partial information is provided in an index field, a list of records that match the given information will be displayed. The user can then select the appropriate record for document retrieval.

In IMAGE TRAC, if the *Retrieve* option is selected, a folder header information window will be displayed (see Figure 8). The folder header title and sub-titles in this

Figure 7

Main Function Window

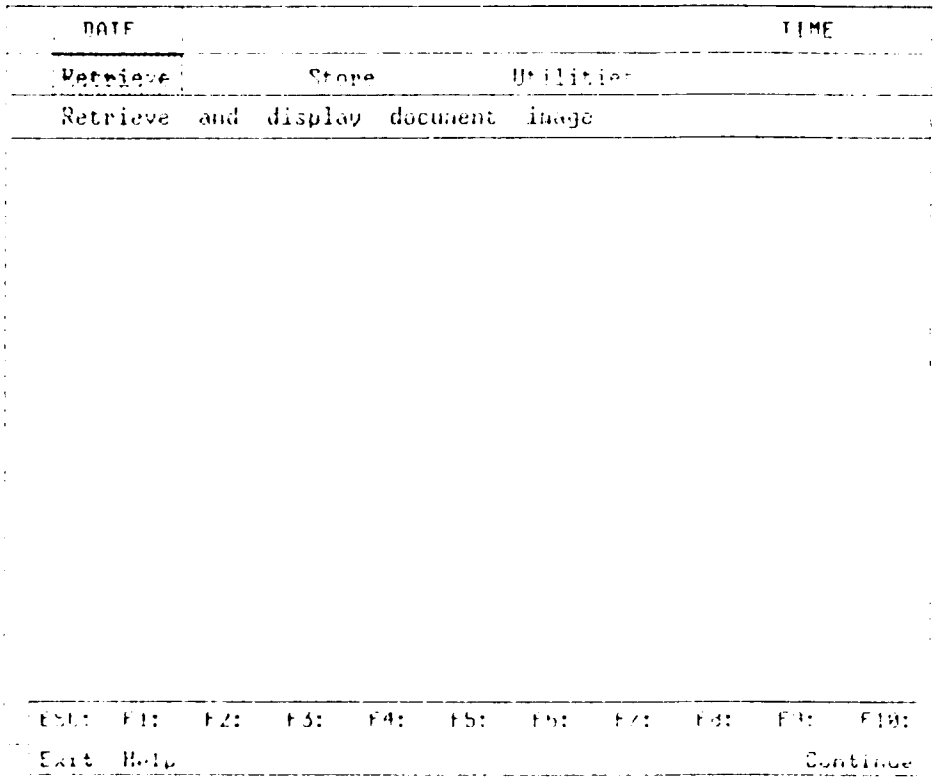


Figure 7a - Mainmenu Window File

```
S    0   main   MENU  6   6   2,80   2,80
ST   63   -1    1,0   0   0   0      0    0   -1
KEY  "mainmenu.key"
T    -1  0,1    "Retrieve"
T    -1  0,22   "Store"
T    -1  0,41   "Utility"
T    95  1,0    "                                  "
F    1   s1     0,1   c   n   8      y
F    2   s2     0,22  c   n   5      y
F    3   s3     0,41  c   n   7      y
```

Figure 7b - Mainmenu Key File

```
D: Exit  Help
F:ESC:QUIT
F:F1:Fhelp
S:1:Fretri_op
S:2:Fstore
S:3:util.msk
I:1  "Retrieve and display document image"
I:2  "Store: Scan in and save new document image in database"
I:3  "Utilities"
```

window can be customized by the system administrator using system utilities. To enhance the interface's user friendly feature and to eliminate unnecessary typing of data, pressing the F9 function key displays a list of folder information (see Figure 8a). As shown, the original window is partially masked. The user can select any folder and the system will then automatically complete the folder header form. If only the primary index field is filled, pressing the F10 function key leads to the next field.

Many useful office functions interface with the folder header information window as we shall now detail. The F2 function key is a search key that allows a user to type in just a few leading characters in any field and will then display all folders that match the information in that field. The F3 function key allows a user to add a new folder into the system. The F4 function key permits a user to modify/update the folder information (excepting the folder ID). The F6 function key clears the entire folder header and lets the user re-enter the information. After the folder information is entered, pressing the F8 function key displays all the document information under that folder ID.

After the folder header information is entered, attention is turned to the folder subject information window. In an office, folders with the same subject are placed in the same drawer or cabinet. For example, customer mortgage folders are placed in the mortgage file, while insurance folders are placed in the insurance file. The folder subject window (see Figure 9), has two levels of indexing, the subject ID primary field and the subject name field. The folder subject title and sub-title can be customized using system utilities, e.g., "Department", "Department ID", and "Department name", in place of "Folder Subject", "Folder ID", and "Folder Name". The F9 function key, in addition to its use

Figure 8

Folder Header Information Window

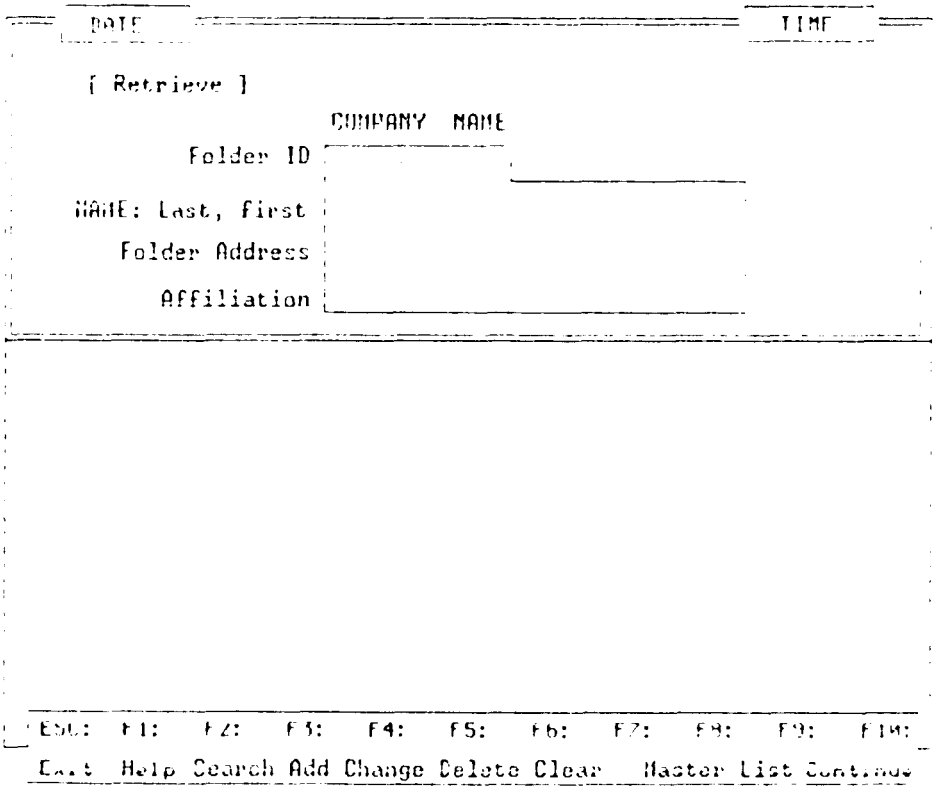


Figure 8a

Folder List Window

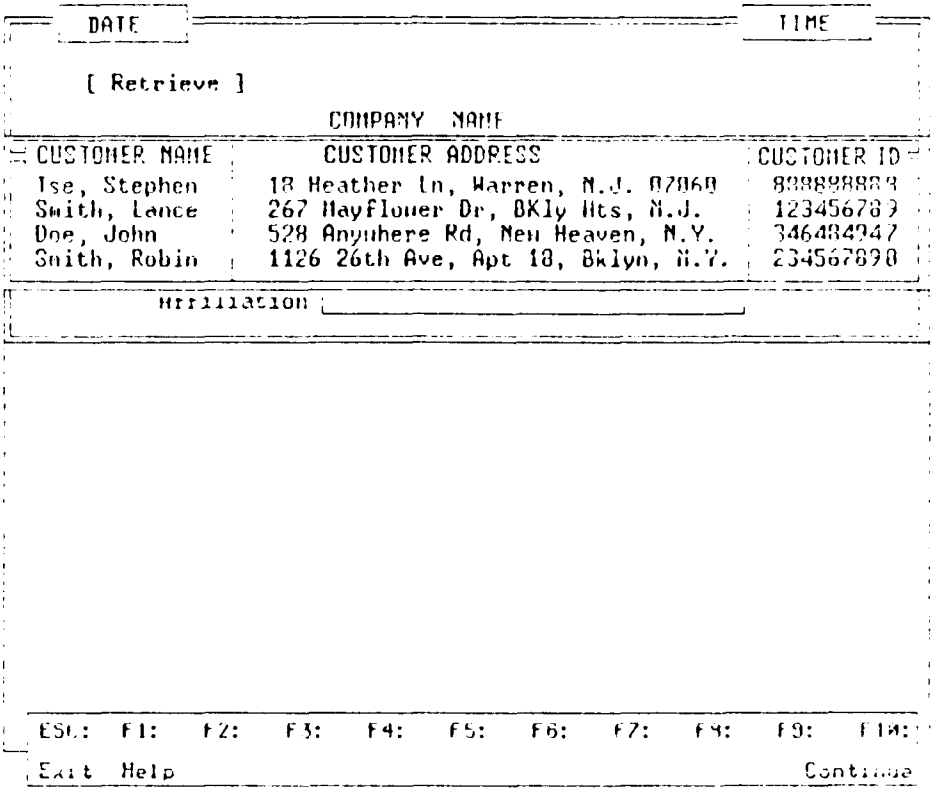
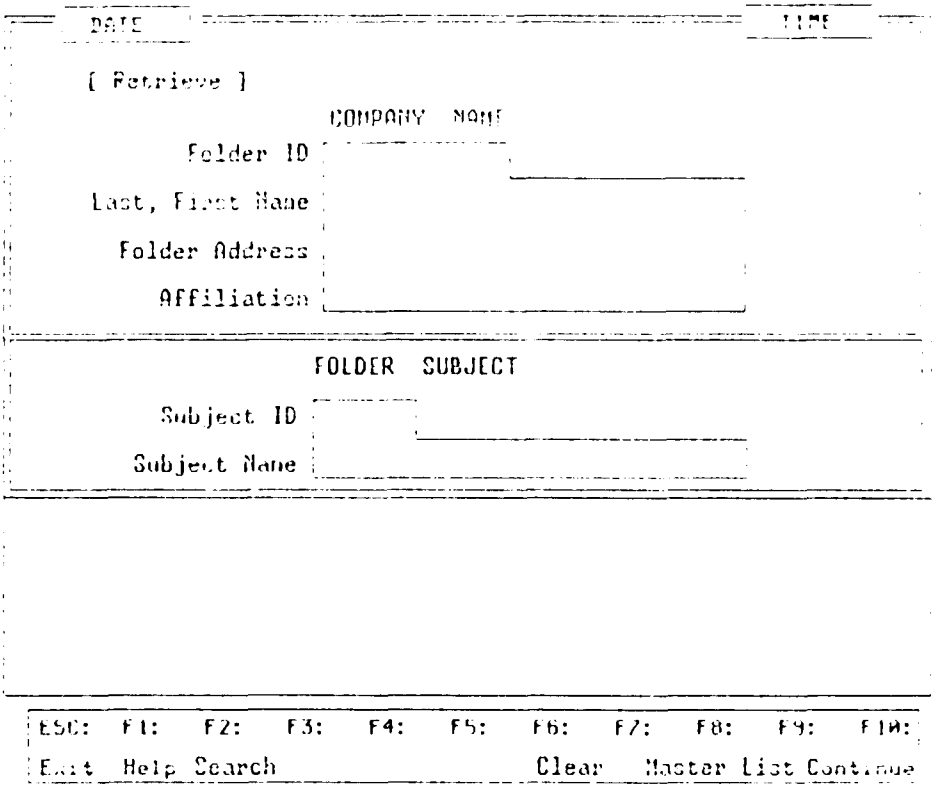


Figure 9

Folder Subject Window



in the folder header information window, also provides a list of folder subjects for the folder subject information window. A user can select and fill out this form as before. The F10 function key (or the Enter key) advances the user to the document information window.

The *document information* window (see Figure 10) has eight identification fields that the user needs in his document management system. Each document must have a unique identification number as well as a name. A survey of the banking industry (which included allied profession) undertaken for this dissertation indicated that the following fields are needed:

(1) The date and time the document was received; the document must have a stamped date to identify it.
(2) The party responsible for (or owner of) the document.
(3) The document's residence within the system--in a magnetic hard disk drive or in an optical disk platter (these comprise the "Platter" and "Drive" sub-fields). If the optical platter number is 000, it means the document is in a hard disk drive.
(4) The document type.
(5) The size (in pages) of the document--important when printing and file transfer are involved.

The layout of all these fields is shown in Figure 10. The titles in this window cannot be changed. Pressing the F9 function key provides a list of documents from which the user

Figure 10

Document Information Window

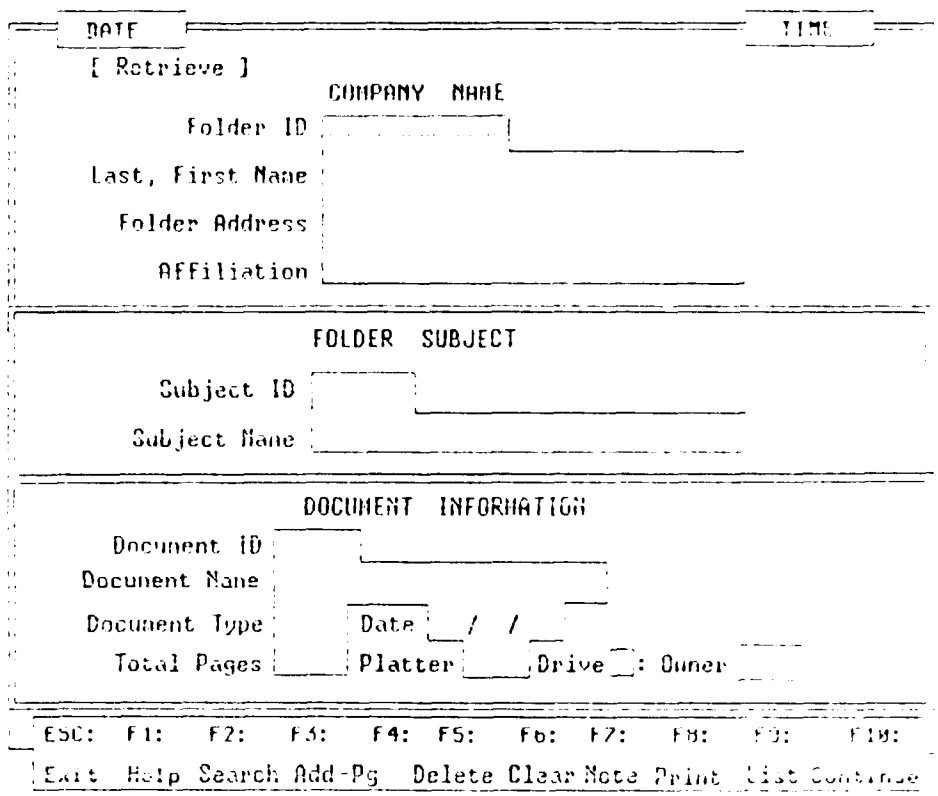
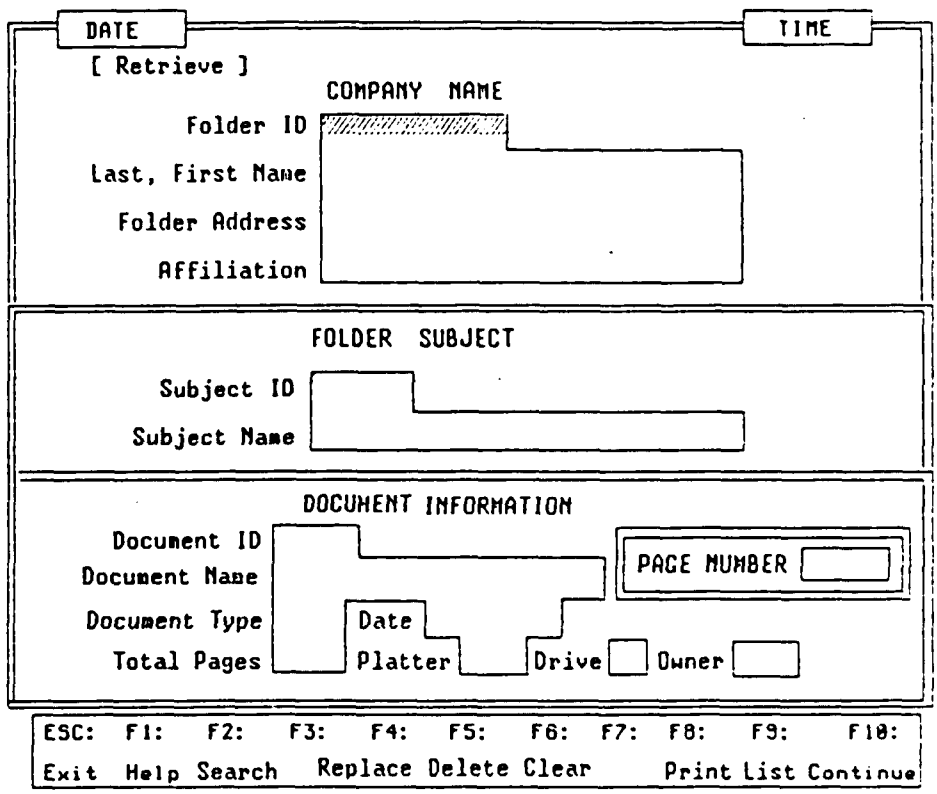


Figure 11

Page Information Window



may choose.

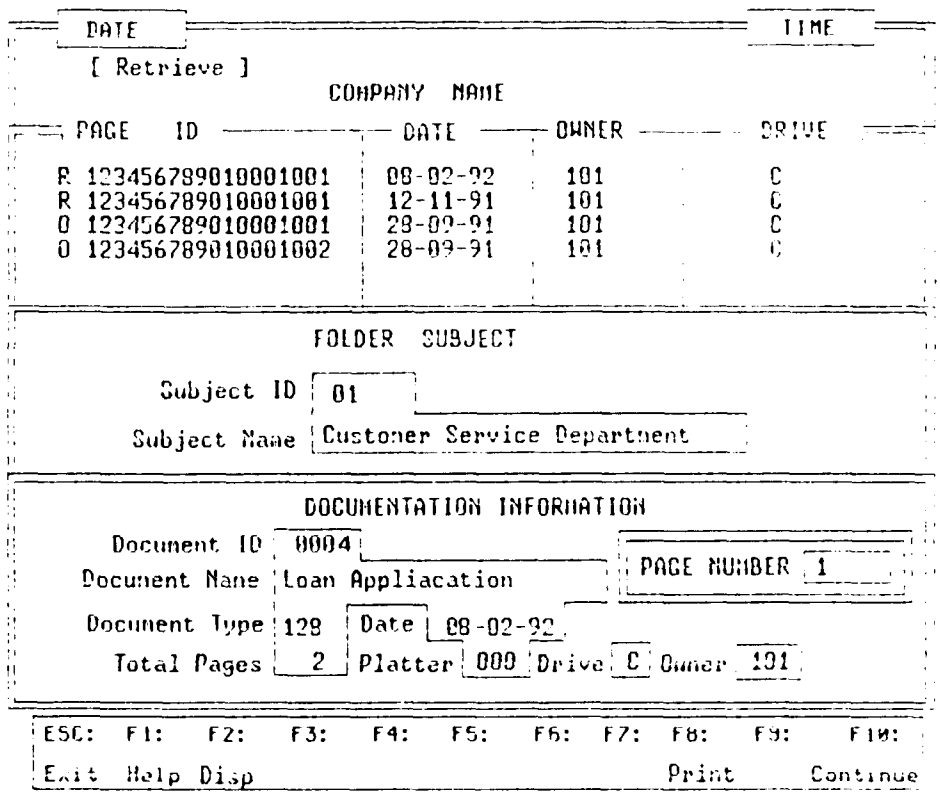
The final window, the *page information* window (see Figure 11), specifies which page to display. After the page number is specified, pressing the F10 function key (or the Enter key) retrieves the page. The user can also perform the following functions: replace, delete, or print the page. If the screen label key Replace (function key F4) is pressed, a window for the user to enter the page (and to place the document page in the scanner) appears. The user can then perform the *Store* procedure (see the following section). If the screen label key Delete is pressed, the *current* version of the page is deleted. If the screen label key Print is pressed, a copy of the page is printed out on a HP-LaserJet printer. In addition, the superuser and system administrators can press the F9 function key which provides an audit trail, viz., a page-history listing as shown in Figure 12. Different versions of the page exist with different date stamps and sometimes with different information. The superuser and system administrators can highlight the desired page (see Figure 12) and press the F2 function key to display any of the previous versions of the page.

iv. The *Store* Window and Subsequent Functions

In IMAGE TRAC, if the *Store* option is selected, a folder header information window is displayed (see Figure 13). The workings of the `Store option are much the same as the` *Retrieve* option explained above. The user first has to input a folder ID. A new account requires creation of a new ID. If the folder information

Figure 12

Audit Page Information Window



already exists in the database, pressing the screen label key List (F9 function key) displays a list of folder ID's and related information. The user can highlight the desired folder ID and press the Enter key to fill this window. However, this list contains all the folder ID's in the system, and it may prove too long to manually search for the record with the desired ID. To shorten this process, the following search procedure can be used (using a procedure similar to that below, the folder subject window can be automatically filled):

(1) Input the leading digits of the ID field (or leading characters in the other index fields) and then press the screen label key Search (F2 function key).
(2) A list of customer ID's that match the given information will be displayed.
(3) Highlight a particular customer ID and press the Enter key.
(4) All the customer information will automatically fill out the folder header information window.
(5) Pressing the screen label key Continue (F10 function key) causes the folder subject window to be opened.

The third window is the document information window (see Figure 13). Under the *Store* option, the document ID field is pre-filled by the system. A user must enter the document name, document type, and total number of pages. Pressing the screen label key *Store* will cause the system to display a window, which requires that a document page be fed into the optical scanner (see Figure 14). When the user is satisfied with the scanning and storage of the page, the system will display the window again and ask for the next page. Until the number of pages that the system has received matches the

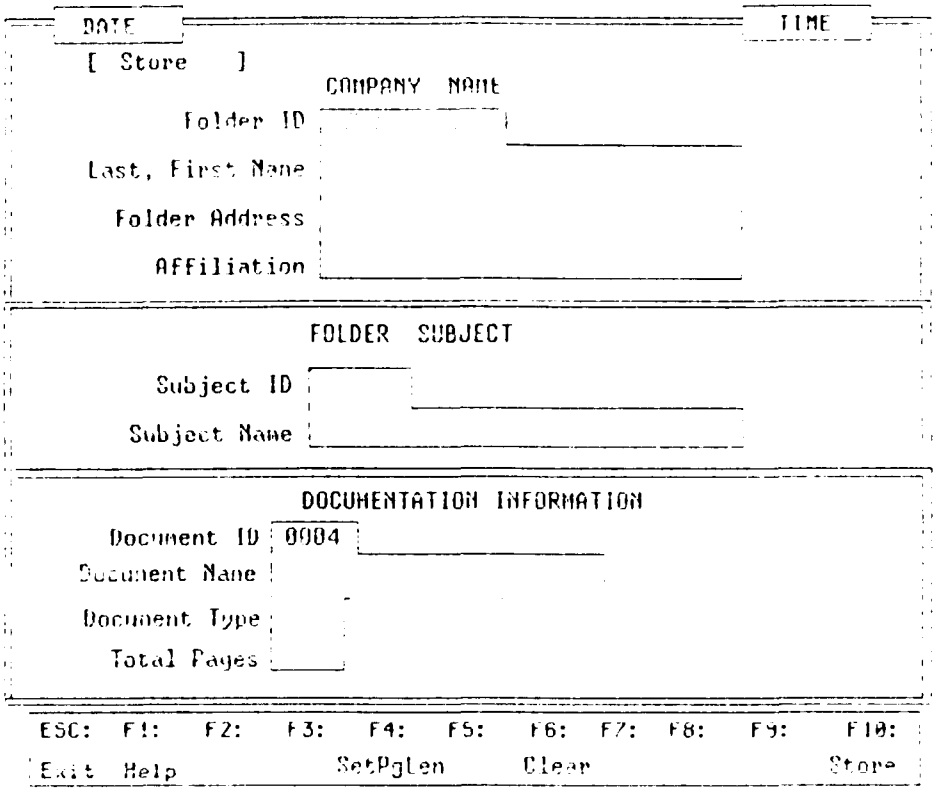
number that the user specified in the document information window (see Figure 13), the system will display a prompt for another document.

To identify every document, IMAGE TRAC assigns a unique internal number to every image page kept in the system. These internal numbers are maintained by the system, and depend on the user ID, folder subject ID, and the document ID. When the user replaces a document page (in *Retrieve* mode), a new internal page number is generated, but the user can only retrieve the latest version of the document page. Earlier versions (each page can have up to 99 versions) of the same page can be retrieved by the superuser and system administrators. This document history implements the audit trail, as discussed in section III.C.2.

The generation and assignment of unique internal page numbers follows dBASE III's database structure architecture. This allows use of dBASE III utilities for maintenance of the system. Every document page (i.e. image file) is associated with a unique nine-digit page address number (with one byte per digit). This number can be broken down into one group of three digits and three groups of two digits, e.g. 001 52 AF 04. The first three digits represent a decimal number and refers to the optical platter number (exception: 000 represents the magnetic hard disk drive). The next six digits are hexadecimal and constitute the page address within a particular optical platter. Hence, this system can handle a maximum of 999 optical disk platters, providing close to 939 gigabytes (10^{12}) in storage. (Each platter has 940 megabytes (10^{6}) of storage capacity.) This is a very liberal allowance, far more than needed for typical applications. For example, an electric utility company serving the entire town of Athens, Texas, used a

Figure 13

[Store] Document Information Window

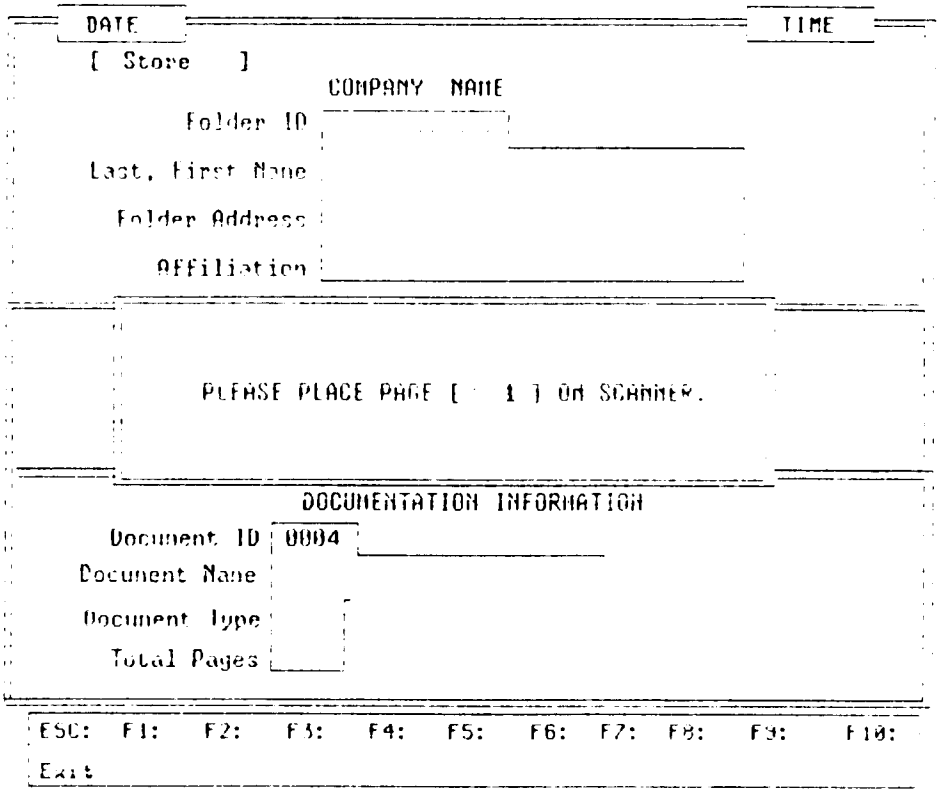


prototype of our imaging system to capture their customer's application forms for records. In a three-month period, they could not fill up even one optical platter. At the rate of four optical platters per year, this imaging system can serve this company for the next 250 years!

To generate a new page number, the system examines the last address and increments it by one to generate the new page number, here 52AF05. To avoid hashing and exhausting the number of files and sub-directories that a single directory can create, a multi-level directory address-mapping technique is used. The maximum number of files/sub-directories (MS-DOS limit) that a single directory can create is 512. The following example will demonstrate how multi-level directory addressing works. The address 52AF05 is mapped to a file in the directory 52\AF. The first two hex digits are mapped to a directory named "52", the next two hex digits, AF, are mapped to a sub-directory named "AF". The last two hex digits, 05, constitute the page address within this sub-directory. Therefore, the page addresses start at 00 and go up to FF for each sub-sub-directory, i.e. 52\AF\(from 00 to FF). When page FF within the sub-sub-directory 52\AF is reached, the directory, AF, is increased by one to B0 with page addresses again starting from 00 to FF, e. g., 52\B0\(from 00 to FF). Under this implementation, the total page address space is $(256)^3 = 16,777,216$. Hence sixteen million page addresses are distributed in 256x256 directories. The benefits of using multi-level directory mapping for document-page addressing and storage are that the system software allows the system hardware to start with a small capacity hard disk and increase to a larger capacity hard disk as the system requires extension. The other benefit

Figure 14

Scan Window



is that the multi-level-directory mapping will limit the number of files in a directory to a level that the operating system can run without too much degradation. If a directory contains more than 300 files and the average file size is over 100K bytes (the average size of a compressed image file is about 70K bytes), file manipulation and access in this directory will be slow and the overall system performance will be degraded by a factor of 3 or 4. So 250 files in a directory is the approximate upper limit that the operating system can handle.

To accelerate access to image files in the database, the dBASE III indexed database structure is used. This structure provides each database in the system with an indexed file for rapid address reference. For example, in the user information database, each record contains various information fields, e.g. user ID, user name, user address, user affiliation, owner, and date of record creation. In order to access a record quickly, a user ID-indexed file, uid.ndx, is created. In this file, each entry contains only the user ID and the record address, and when the user ID is found, the record can be retrieved immediately.

When the Replace function is invoked in the *Retrieve* mode under the document page window, the system will assign a new internal number for the new page and mark the old page with either "O" for original or "R" for revised. All the previous versions of this page are kept for later audit-trail reference. In *Retrieve* mode, in the document page window, if the screen label key List is pressed (provided the user is a superuser or has administration permission), a window with a list of all the different versions of that page is displayed (see Figure 12). The auditor can highlight any page version and either press

the screen-labeled key Disp to display the page on the screen or the screen-labeled key Print to print out the page.

V. The *Utilities* Window and Subsequent Functions

The *Utility* menu (see Figure 15) provides two categories of utility functions. One category is for users to maintain their own password and directories, and also perform some low-level system functions, while the other category is reserved for system administrators (the users in group number 100) to maintain the system.

The privileged functions come under the ADMIN Utility, System Setup, Modify System Setup, and Configuration headings as shown in Figure 15. The ADMIN Utility function provides a set of six routines to maintain the system. The ADMIN Utility window (see Figure 16) includes the following functions: User Maintenance, Change ADMIN Password, Security Control, Folder Subject Maintenance, Re-index Database, and Pack Database.

The User Maintenance function contains a *USER MANAGEMENT* window (see Figure 17) to provide new USER ID's and USER LOGIN's, change user passwords, keep records of user names, and assign user group numbers. (A given group number is assigned to users who work in the same area.)

The second function is Change ADMIN Password, which is similar to the change user password function except that the user ID of 101, the user name of the

Administrator, and the group number of 100 are pre-filled and cannot be changed. The password field is the only field that can be changed.

The third function is Security Control. Every folder and folder subject has three kinds of permissions, viz., read, write, and delete. When a new user login is setup, the system administrator has to assign appropriate permissions to that user to access each folder subject ID's (see Figure 18). The administrator can assign permissions to specific subject folders by providing read only, read and write (but not delete), or read, write, and delete permissions. If no permission is provided, the user cannot access the folder at all. This security measure is especially required by the banking industry as my field research (including interviewing several bankers, insurance agents, an oil industrial executive, and an air-line executives) for this dissertation has demonstrated.

The fourth function is the Folder Subject Maintenance. Sometimes new folder subjects may have to be added to the system or old folder subjects may have to be changed, because a new department is being established or because the company is reorganizing. As shown in Figure 19, this function will provide a window in which the administrator can change the folder subject name. Any field can be modified, but if the primary key field (here, an ID number) is changed, the system treats the folder as new.

The fifth function is the Re-index Database. There can be many index files associated with a particular database file, with up to seven open at any time. All open index files are updated as the contents of the associated database changes. Were index files closed when the database was updating, these index files would no longer

Figure 15

Utility Menu Window

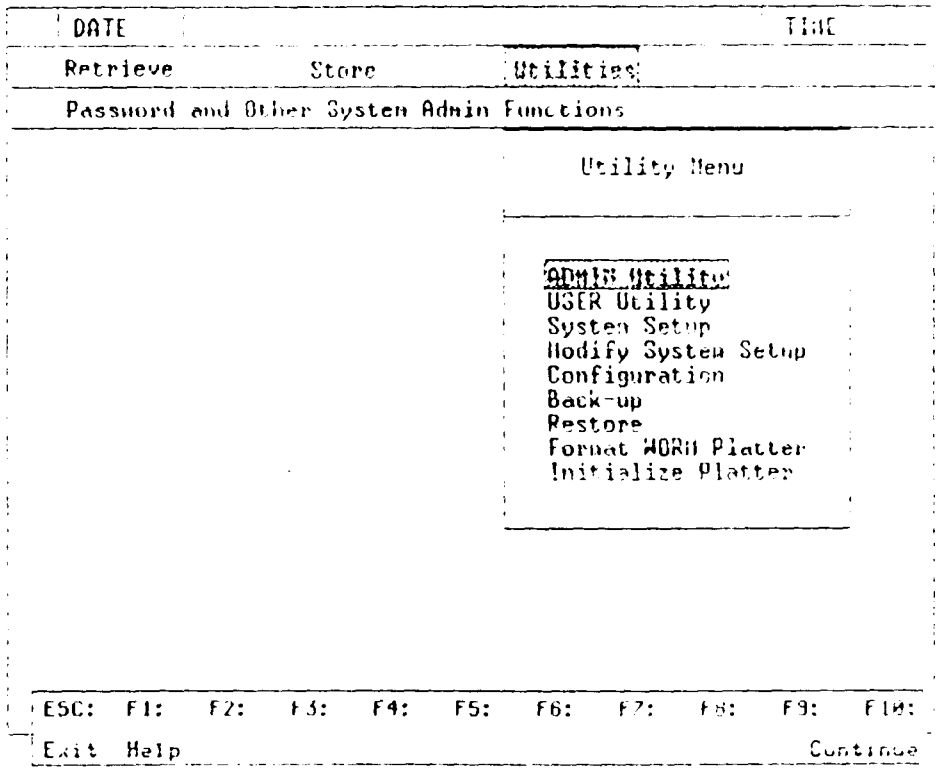


Figure 16

System Administration Window

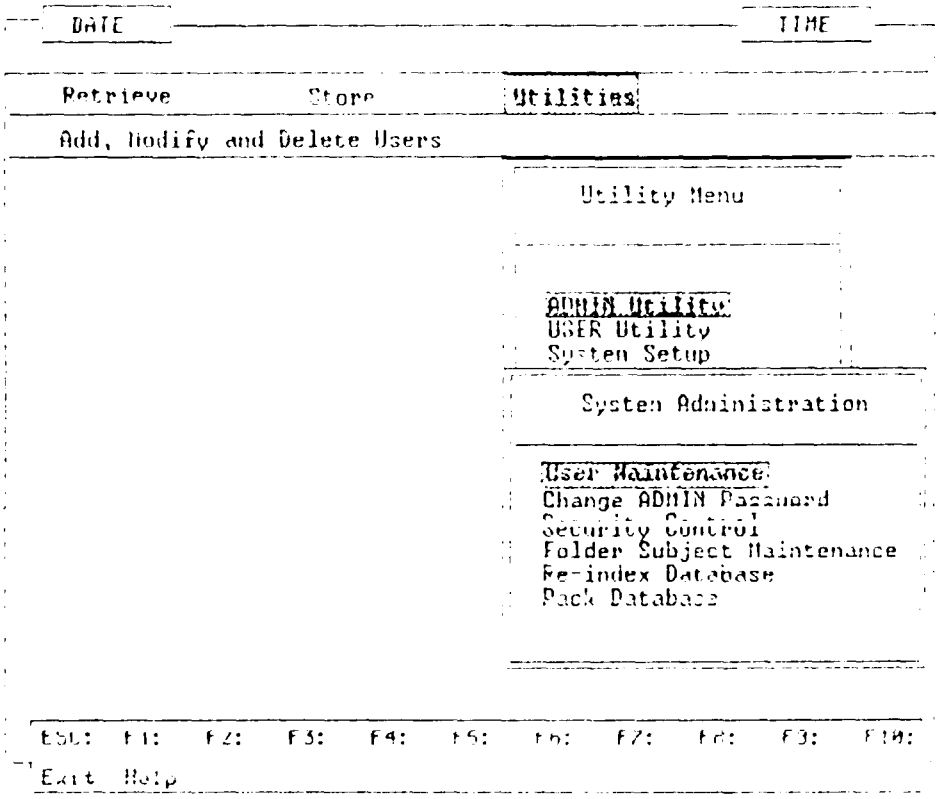


Figure 17

User Management Window

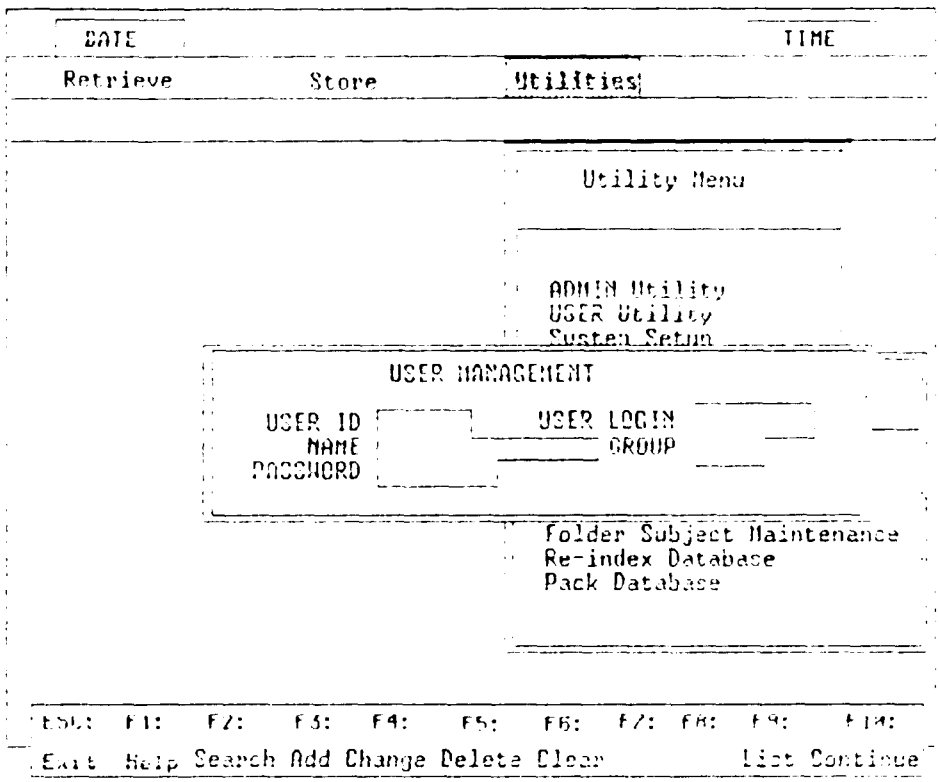
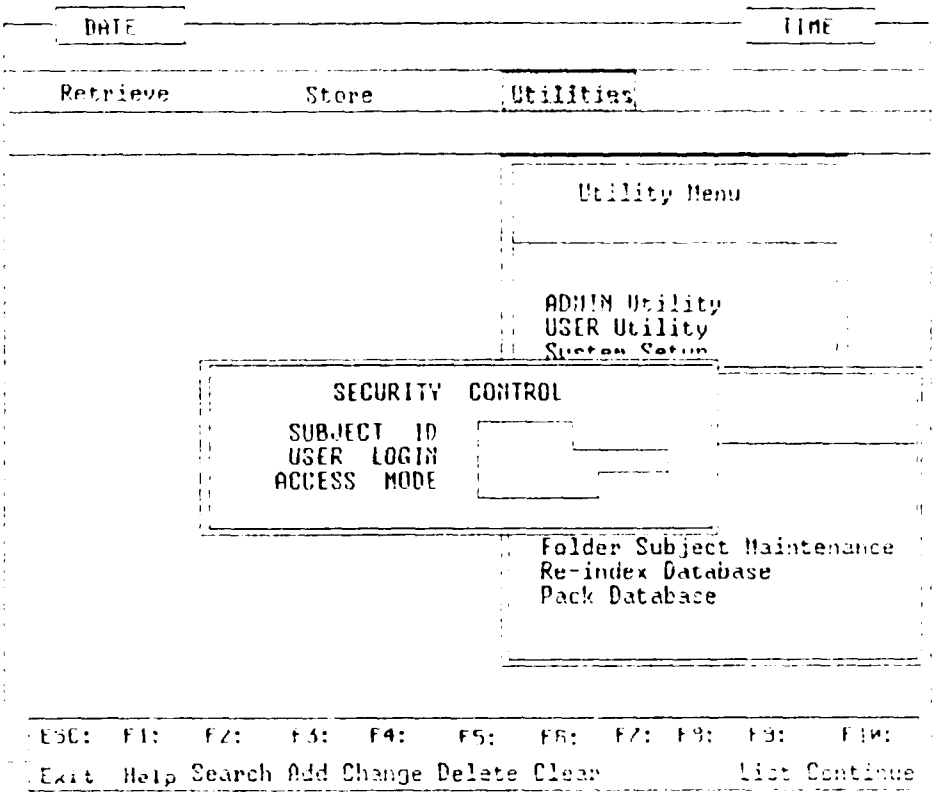


Figure 18

Security Control Window



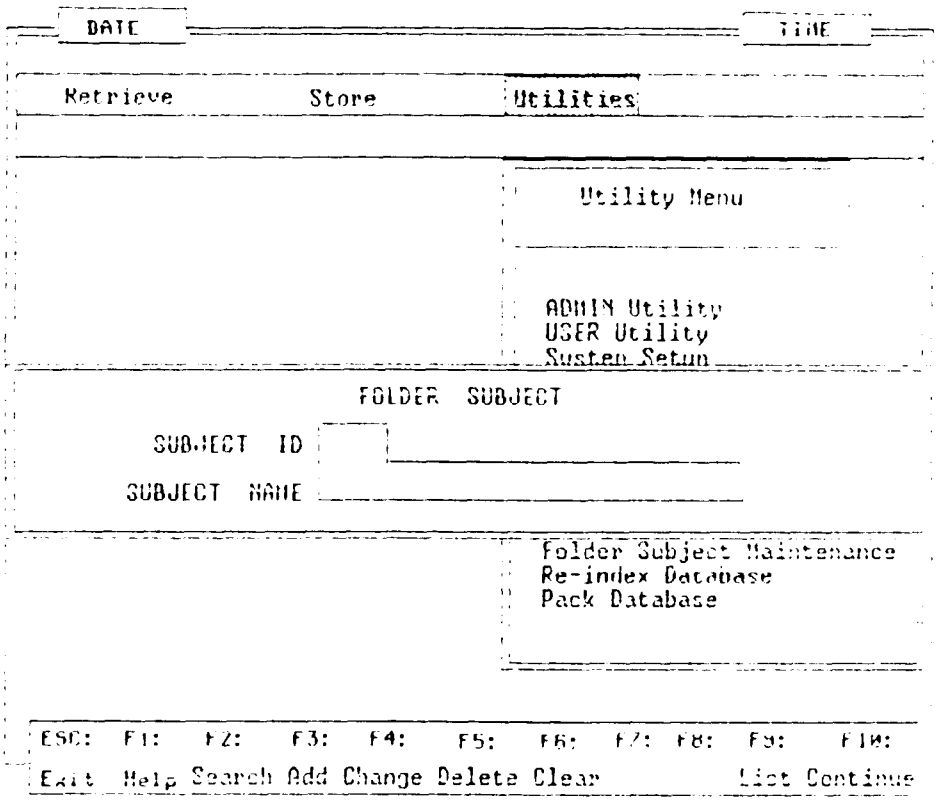
correspond to the database. As a result, when the index files are used later, a data error leading to a corrupted database may occur. Under these circumstances, a re-index of the .ndx file (index file) will fix the problem. The Re-index Database function will make sure that the index files are updated correctly and properly correspond to the database file. This can provide a quick fix for data corruption of the folder, document, page, password, subject, or access databases, as the case may be. The system administrator can also highlight the ALL option and re-index all of the above databases' index files.

The sixth function is the Pack Database. After deletions from or updates of the database, items that are marked "delete" still remain in the database. This function performs housekeeping chores (such as garbage collection) of the folder, document, page, password, subject, and access databases. System performance is thereby enhanced as the space needed for the system's many databases is reduced. The system administrator should frequently reassess the availability of storage.

The next two privileged functions comprise the System Setup utility. As one might expect, this utility is used when a new system is set up for the first time. The first task a system administrator handling a new system must perform is to use the *System Setup* window to set up folder parameters (see Figure 20). Before the main setup window is opened, if the system was used before, a warning message will be sent to the system administrator "A folder database exists, do you want to overwrite?" If the answer is yes, a second warning message will be issued "Are you sure? This will erase all existing data!" If the answer is yes again, the main window will open. It can be used to enter a **FOLDER MAIN TITLE** name (e.g., the company name) which cannot be more than

Figure 19

Folder Subject Maintenance Window



24 characters, and the name and size of the four **FOLDER SUB-TITLE FIELDs** (e.g., user ID, user name, user address, and user affiliation). The maximum field width is 40 characters or digits. Then the system administrator must go to the *Admin Utilities* window and set up the folder subjects and the system configurations. He can use the *Folder Subject Setup* window (see Figure 21) to setup the different departments and use the *Configuration* window to setup the storage device(s), the printer port, and the scanner. If the administrator wants to change the **Folder Subject Titles**, he can use the *Modify System Setup* window to change the **SUBJECT MAIN TITLE** (e.g., department) as well as the two **SUBJECT SUB-TITLE FIELDs** (e.g., the numeric ID and the name). After the above system setup, the system is ready for general use. for general use.

After the system has been set up, if some *System Setup* information has to be modified, the *Modify System Setup* window (see Figure 22) will allow the system administrator to modify the *Folder Title Setup* and the *Subject Title Setup*. The system administrator can modify both the main title and the four folder fields, as well as the subject main title and the two subject title fields. However, the size of these fields can only be fixed during system initialization (compare Figure 22 with Figure 21).

The final privileged system function is the System Configuration function. The system administrator can open the *Configuration* window (see Figure 23) and choose the appropriate hardware for the system to use. The menu contains four types of hardware that a system administrator can designate as peripheral devices. The *Storage Devices*

Figure 20

System Setup Window

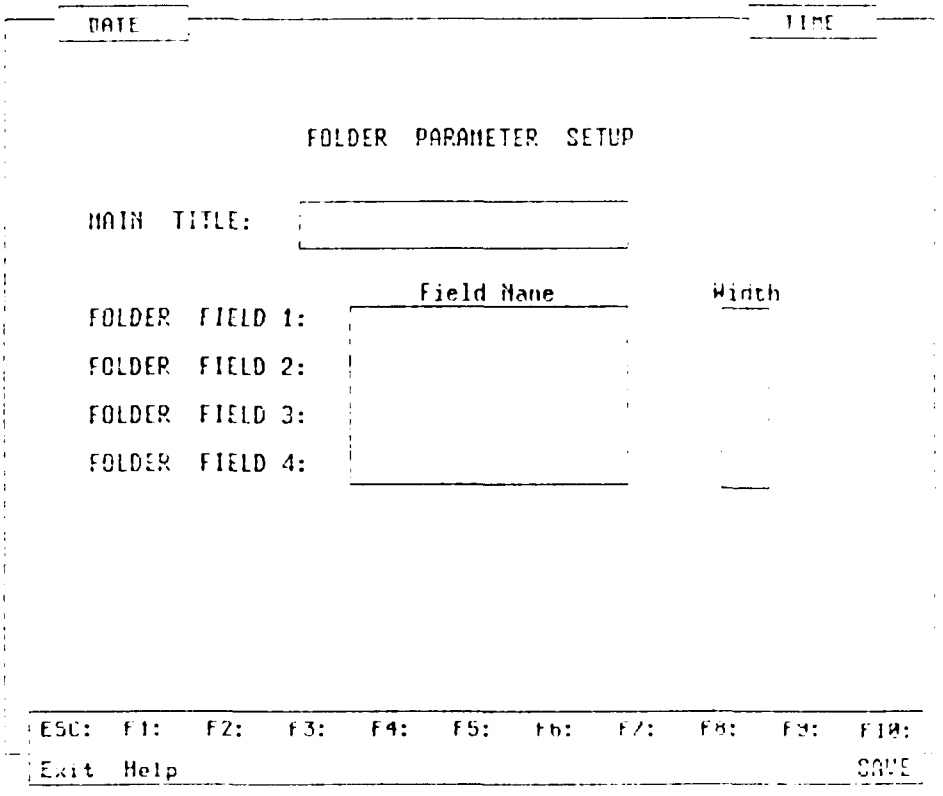
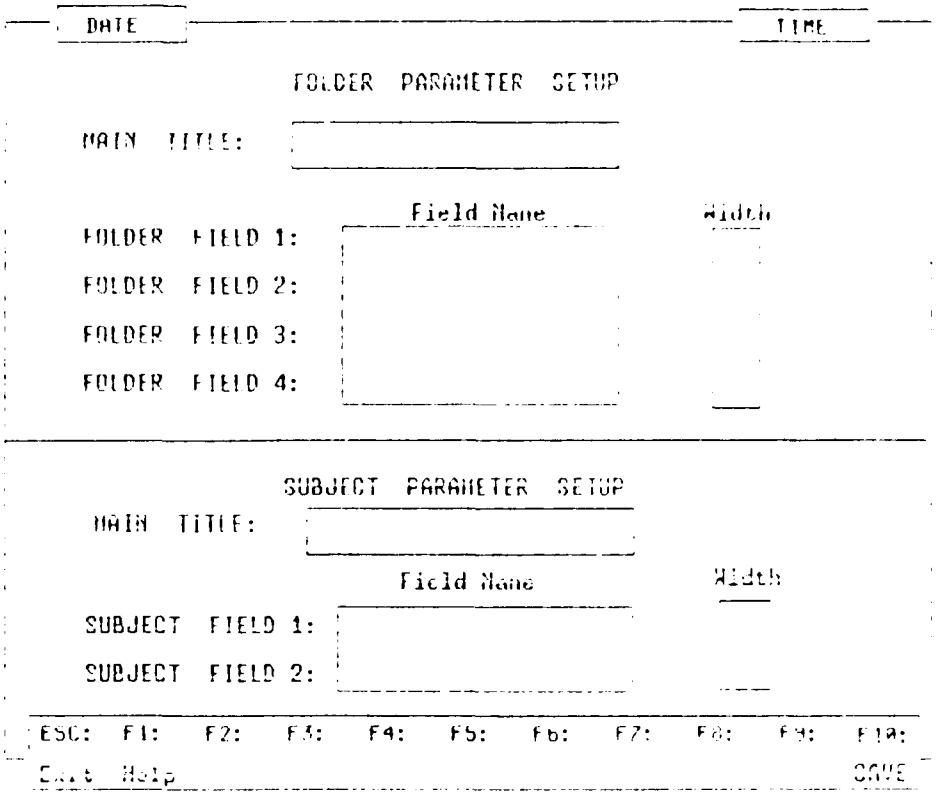


Figure 21

Subject Parameter Setup Window



window (see Figure 24) allows the administrator to specify the type of storage devices the system should have for its document storage. By default, the system assumes all logical devices are mapped to the same physical device, the magnetic hard disk drive. The system administrator either assigns physical device drive letters to the Write Once and Read Many (WORM) optical drive, to the Read/Write Magnetic Storage Device, to what we have termed "Temporary Storage Device" or allows the default value to be used (e.g., the "C" drive). The *Printer Port* window (see Figure 25) allows the system administrator to specify what printer port should be used for image printing (the a default is printer port 1). The *Choose Scanner* window (see Figure 26) allows the system administrator to specify what scanner should be used for image capture. Presently, IMAGE TRAC™ supports both the Panasonic RS307U and the Canon IX-12 scanners. The final configuration assignment is the *Display Device* window (see Figure 27). The administrator must assign an IBM VGA-compatible color display monitor for image display.

All users in the system can use the User Utility, Back-up, Restore, Format WORM Platter, and Initialize Platter utility functions. The *User Utility* window (see Figure 28) has functions for a user to change his password and to move his documents from one non-optical drive to another or to an optical drive. When a user chooses the Change Login User Password function, a window (see Figure 29) will open. All the user information is pre-filled except for the password field. The user can type in a new password and press the screen label key F4 to change the password. When a user chooses the Move Document function the system will open the *Search Document By Date*

Figure 22

Modify System Setup Window

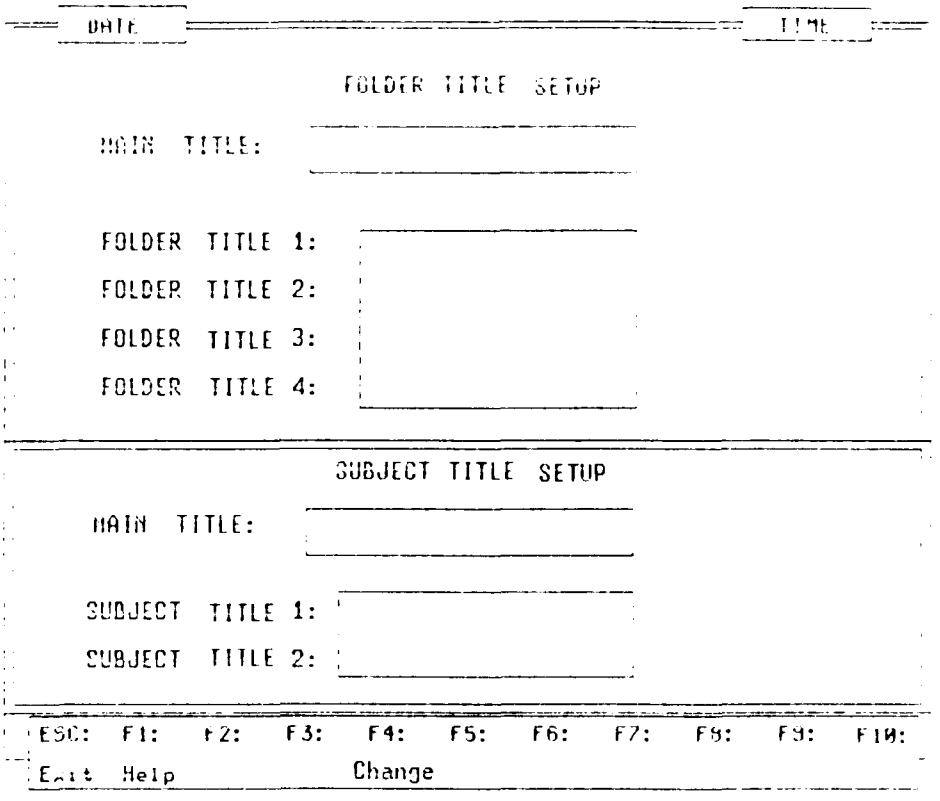


Figure 23

Configuration Window

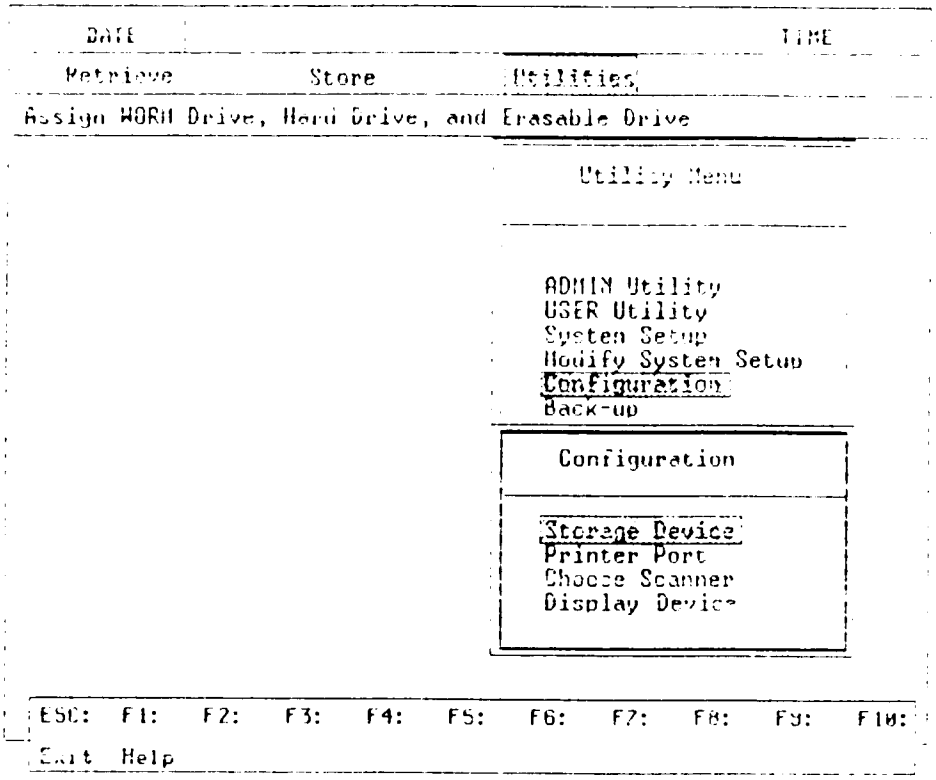
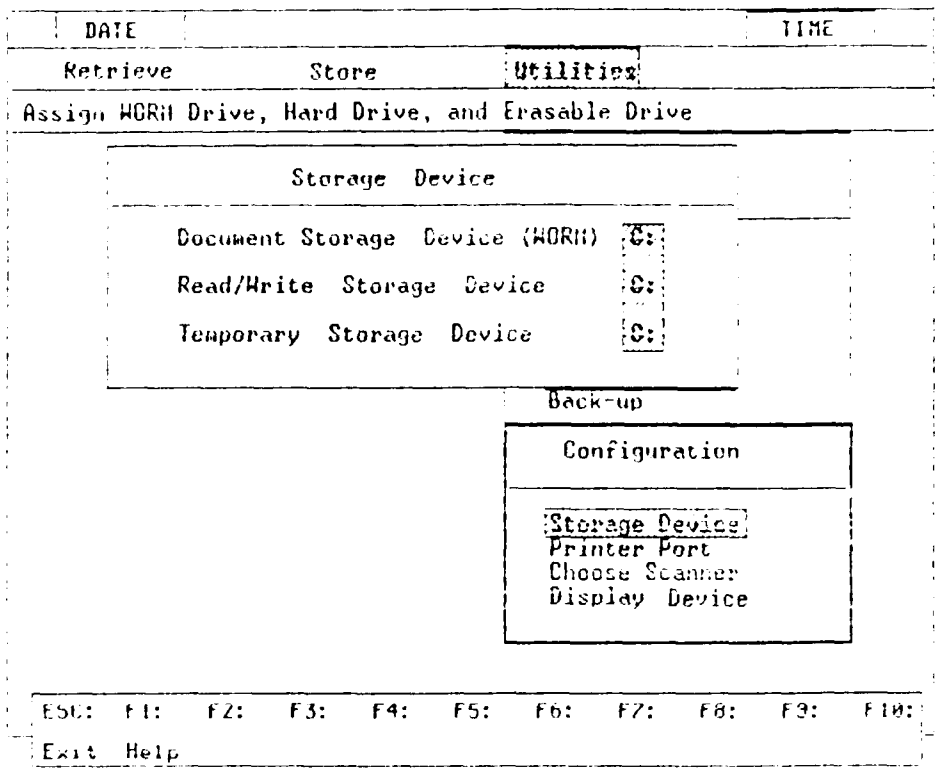


Figure 24

Storage Device Window



window (see Figure 30). This window allows for multiple searches for documents created on a given date. Hence, when the user fills in the date, the system will find all the documents created on that date by the user and will display them in a *Move Document By Date* window (see Figure 31). The user can then use the directional keys to find the document that he wants to move, using the "+" key to mark the document and using the "-" key to delete the mark. After this, depressing the screen label key Move (function key F4) causes the system to respond with an information window which asks "Do you want to move marked DOCs?" If the user responds yes, the system displays a second information window and responds with the implicit query, "Move to drive _: ". The user then enters a drive letter and presses the screen label key Continue (function key F10). At this point, the documents have been successfully moved from one drive to another.

The other four system functions that a user can invoke are the Back-up, Restore, Format WORM Platter, and Initialize Platter. When a user invokes the Back-up function, the system asks "Do you want to backup?" If the user answers yes, the system goes into the directory YY/MM (YY=year, MM=month) and creates a today's date, or DD, directory. If either YY or MM does not exist as a directory, the system creates it. The system then copies all the database structure files, and all the databases into the DD directory. If the WORM drive is not available, the system asks "Document drive C: is not a WORM drive. Do you want to continue?" If the answer is yes, the procedure above is performed.

When a user chooses the Restore function, the system asks "Do you want to

Figure 25

Printer Port Window

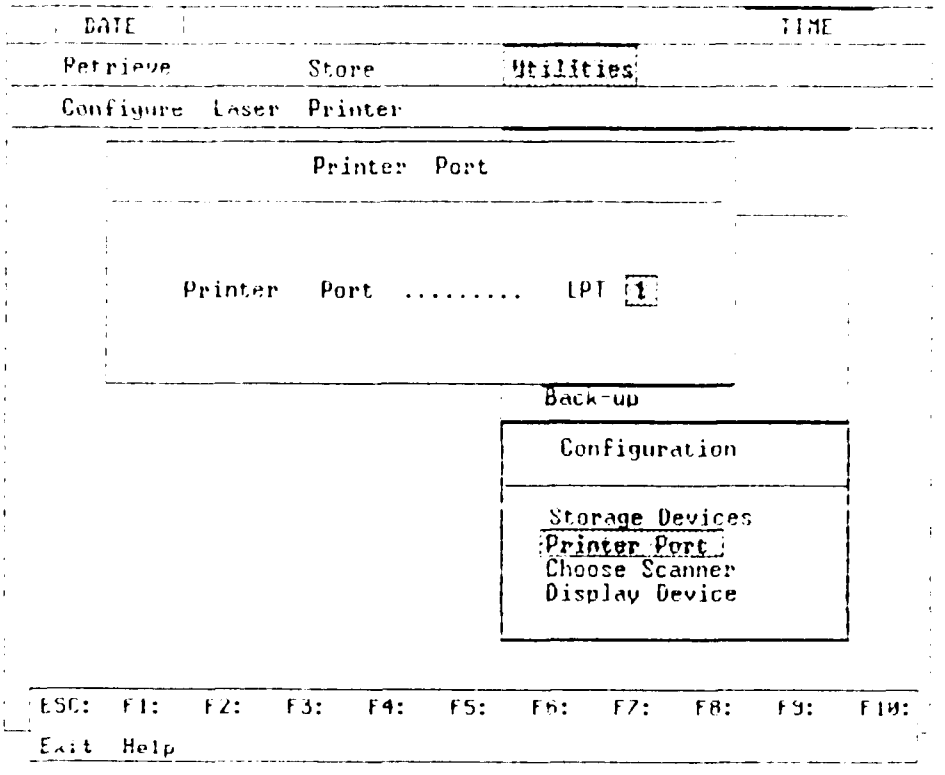


Figure 26

Choose Scanner Window

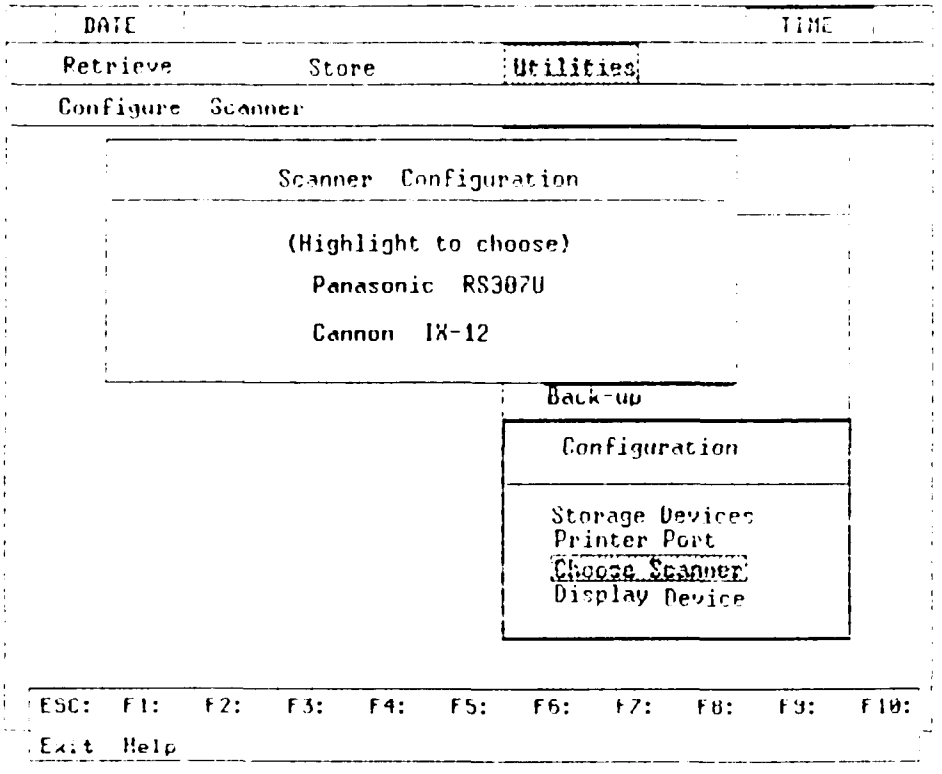


Figure 27

Display Device Window

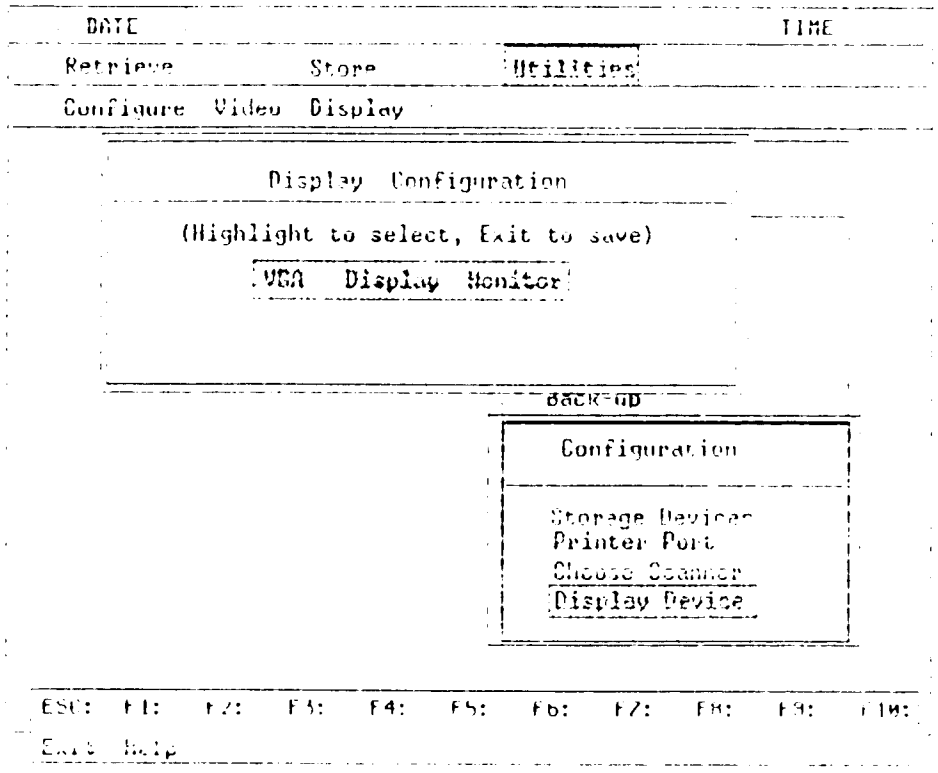


Figure 28

User Utility Window

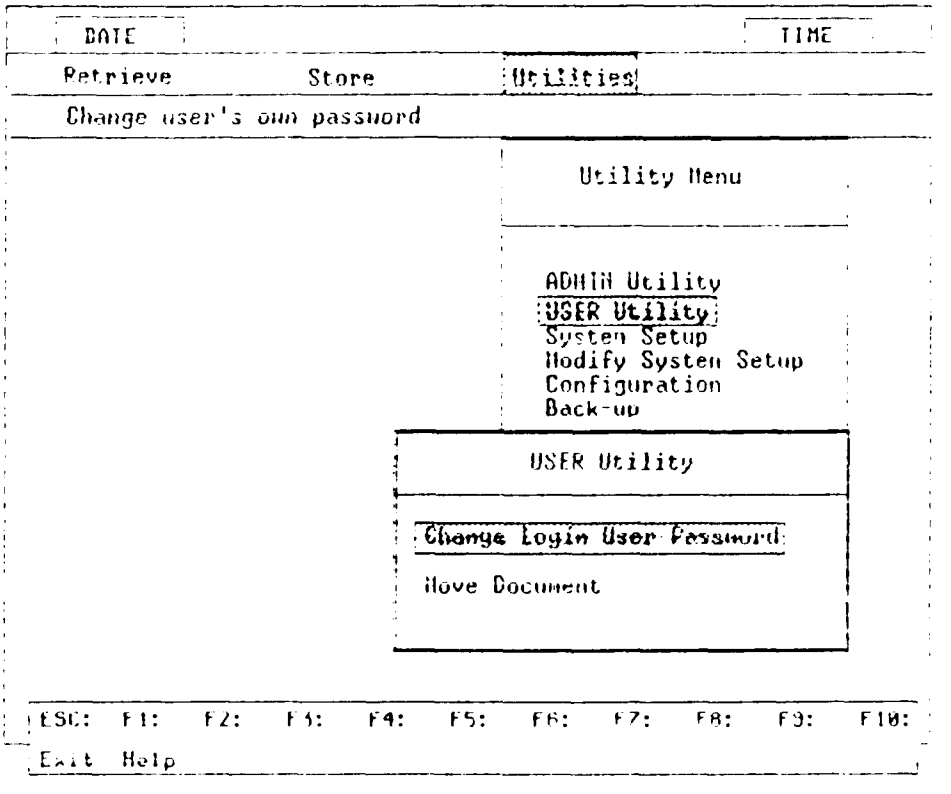


Figure 29

Change Password Window

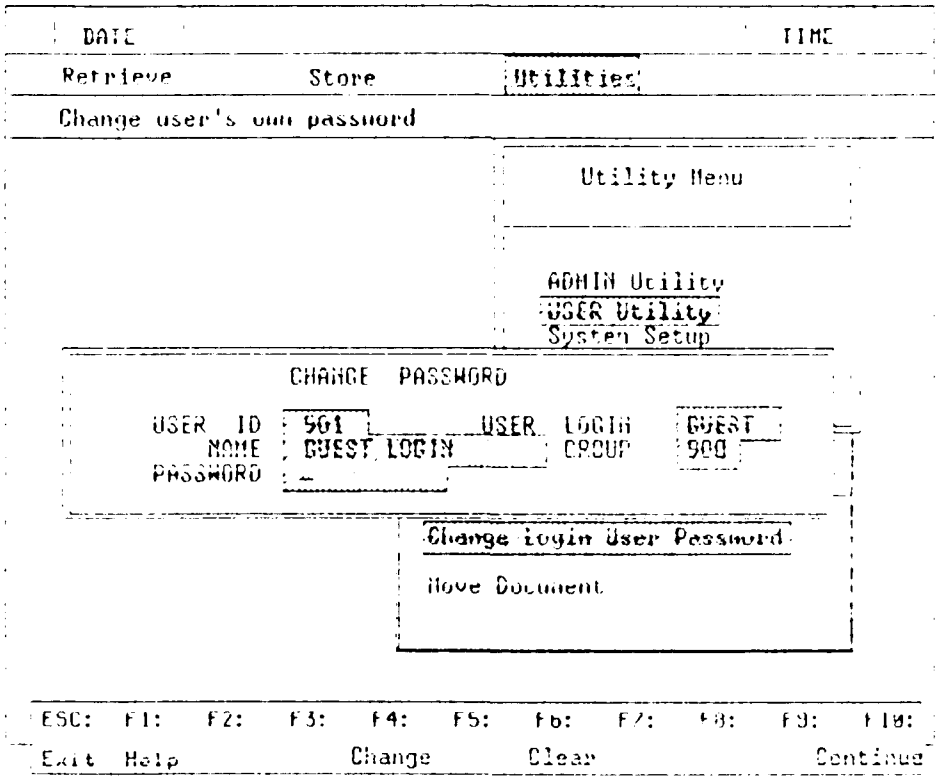
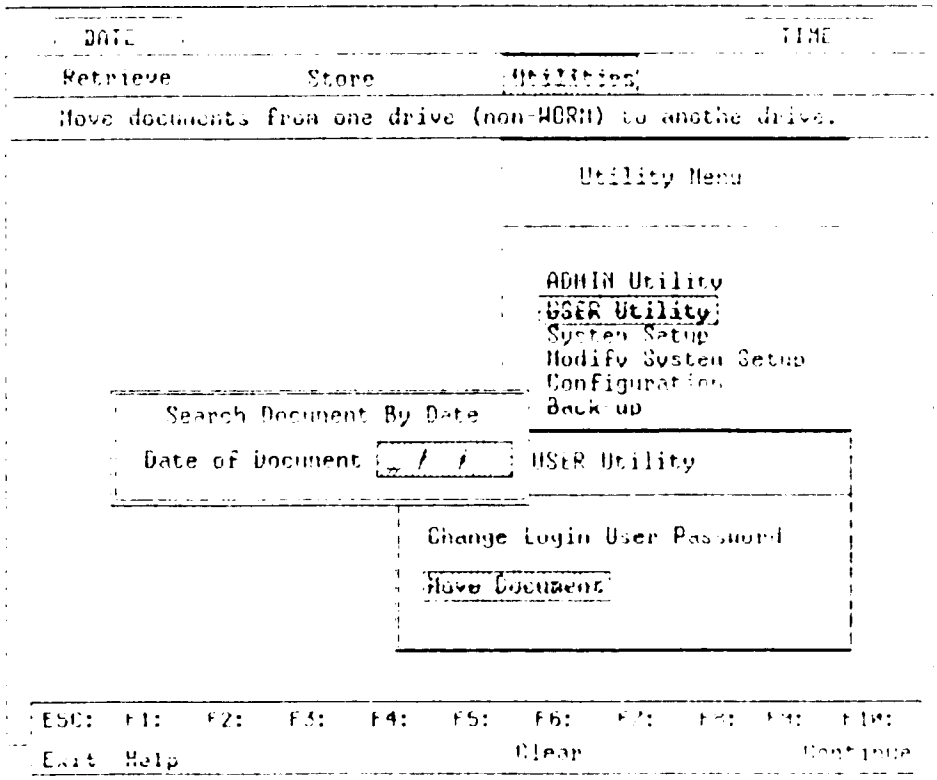


Figure 30

Move Document Window



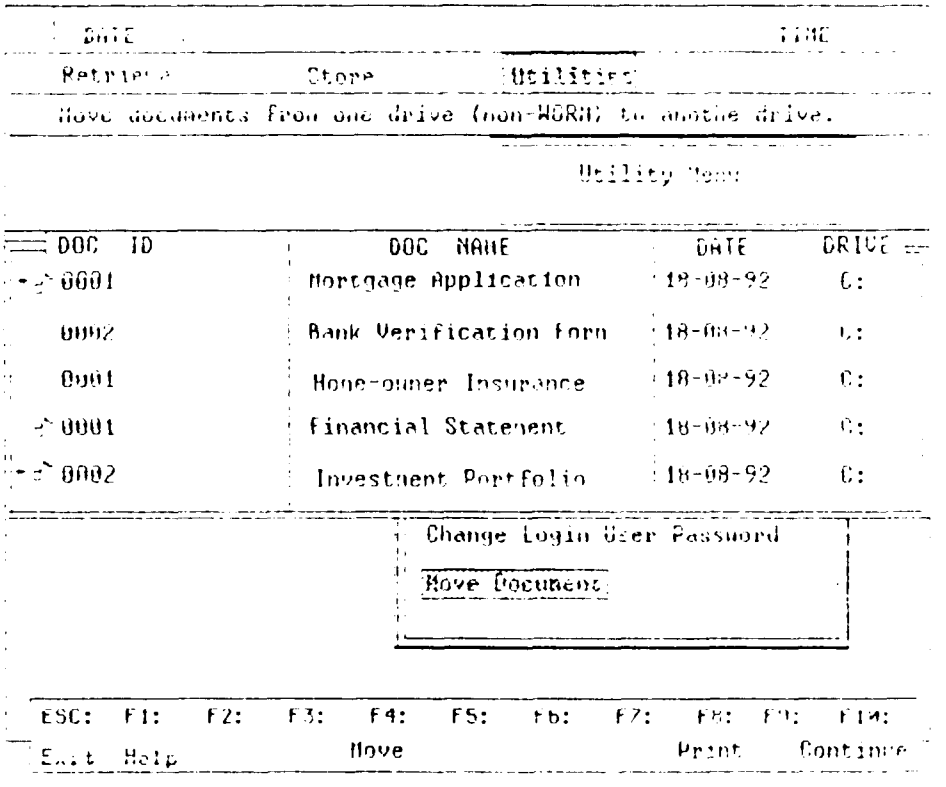
restore?" If the user answers yes, the system will ask "Date to restore from?" When the user has provided the date information and has the backup optical platter in the optical drive, the system performs the restore function.

When a user chooses the Format WORM Platter function, the system responds with the implicit query "Format drive __:". The user then enters an optical drive letter and the system performs the WORM platter formatting.

The Initialize Platter function initializes a new or used (but not entirely full) platter so that the system will recognize it as one of the system's document platters. IMAGE TRAC provides unique internal numeric ID's to every document. Therefore, document platters must start with IMAGE TRAC-assigned documents. If any foreign files exist in the platter, the system will mark them "non-usable", and move the initial header storage mark to the next available empty space.

Figure 31

Move Document By Date Window



IV. CONTRIBUTIONS OF THE PRESENT SYSTEM

A. INTRODUCTION

IMAGE TRAC combines features in such a way as to improve upon all other existing software systems with the same general purpose. Performance is measured for this class of systems along several dimensions:

(1) conformance with the theoretical framework of data-object logic,

(2) quality of user interface,

(3) speed of window display and data retrieval,

(4) flexibility including ease of system expansion,

(5) price,

(6) availability of technical support.

(7) space efficiency,

With weights assigned to the components of this vector based on the needs of the commercial enterprises we surveyed for this research (see Chapter III), the dot product shows that our system outperforms all existing competitors and constitutes an advance in the engineering of imaging systems. We elaborate.

B. CONFORMANCE WITH THE THEORETICAL FRAMEWORK OF DATA-OBJECT LOGIC[1]

Designing data objects is as important as writing programs. If data are formatted as logical data objects, programs can then manipulate them efficiently without knowing their physical structure. This allows for portability and greatly contributes to speed and ease of use. Data are, finally--to a computer scientist--anything that a program manipulates, while programming is the art of taking some real-world system and modeling it as a collection of data objects. Indeed, the central difficulty faced by systems developers is that modeling is a task that requires consummate skill.

In IMAGE TRAC, each of the screen layouts, i.e., the ".msk" files, is a carefully designed and formatted data object. Screens are data objects of structure type MASK (see Appendix under mask.h) with operations defined in the window-type and the ".key" files, as explained in Chapter III, section 4.b.

C. SUPERIOR USER INTERFACE[2]

1. Color Coordination and Standard Function Definition Conventions

Fine color coordination creates an environment which is comfortable for users.

---

[1] Component (1), above.

[2] Component (2), above.

IMAGE TRAC has such a color arrangement built-in on which immediate elaboration. One example of the elegance of using color is the way it can be used to elicit responses from the user. For example, the system uses color to indicate trouble whenever it occurs. Presented with a red screen the user senses the importance of his response to the menu. Presented with a green screen, the user can choose to skip it for now and return later, because it is always--in our system--an explanation or report screen. Even sub-windows are color-coordinated. For example, screen-label-key sub-windows are always blue on turquoise. Colors used by the IMAGE TRAC system include black, gray, green, lavender, red, royal blue, sky blue, turquoise, white, and yellow. The layout of each screen is carefully arranged with multiple colors used in the various windows. The impact that the color coordination of our system has on those using the system can only be fully appreciated with a demonstration of its use.

To ease learning to use the system, all function keys follow the same definition conventions. If a user knows how to work and manipulate functions in one window, he will be able to manipulate those functions in all other windows. This is because all screen label keys and keyboard function keys are in the same position and have the same definition in every screen. Specifically, the screen label keys are always at the bottom of the screen with consistent color and function arrangement. Likewise, the F1 key is always a help key, while the F10 key is always a continue or logical return-to or next-to key, and so on. Indeed, the entire system was designed with ease of use in mind.

2. Almost All Functions Are One Keystroke Away

The user interface is comprised of 48 windows with pull-down menu and screen-label keys. The user can access menu selections, help, and system functions with only one or two keystrokes (see Chapter III). A user can select an item anywhere in the window by just typing the initial character of the item. The function will then be performed. Even to a novice who has only experimented with the system for a few sessions, all of IMAGE TRAC's features are just a keystroke away.

3. Limited Typing and Reduced Typing Errors by Providing Selection Menus

For ease of data entry and to minimize typing errors, we have chosen to rely heavily on menus and on highlighting our selections. Thus, in all input **FORM** windows, a LIST and/or MASTER list command (one of the screen label keys) is available for the user to open a **SELECT** list which provides input selections. If the needed input information is not new to the system, the user will find it in the **SELECT** list. Otherwise, instead of entering all the relevant data, the user simply highlights the selected information and presses the return key to back-field the **FORM**. This **SELECT** list user interface was developed during many interviews with actual potential customers including professional accountants, bankers, real estate brokers, and secretaries. Our field work guided the design and development of IMAGE TRAC and partly explains its superiority to existing image management systems.

4. User Help Is in Every Screen

When a screen item is highlighted, a screen explanation line (QUICK help line)

appears on the screen. If a user needs more help, he can press the F1 key for more details. There is a help function key in every screen, and we have labored to make these functions truly helpful.

Ease of use--always keeping the user in mind--was the primary focus as the system was developed. This focus on the user's comfort makes the system easy to learn and use.

D. RAPID WINDOW DISPLAY AND RAPID DATA RETRIEVAL FROM DATABASE[3]

When an image system is designed it must handle the display of windows efficiently. Many of the existing packages treat window manipulation and other sophisticated functions as overhead. Thus, these windows are always disk-resident and there is a consequent delay in transferring files from disk to memory. (The delay time can be compensated for with a machine that has a faster CPU, i.e., the Intel 60386/33 mega-hertz or Intel 60486. The user, however, must then pay a higher price for his machine.)

In our system, all the ".msk" (i.e., the binary window masking files) and ".key" (i.e., the screen label key builder files) files are memory-resident to give maximum performance. They are all bound in one run-module, a file with extension ".m". When a particular screen is called, the corresponding masking module will immediately be

---

[3] Component (3), above.

displayed on the screen. No redrawing of the screen or downloading from the disk is involved. Instead, a new window will be laid out on top of the existing window and will mask out that part of the window needed for the new display. The user will still see some portion of the old window behind the new window. Overlaying windows in this manner--masking--provides for rapid display of windows, while most existing window systems--which do not use masking--are slow and cumbersome in this regard.

To locate data in a huge database, the key is to have an effective addressing scheme. The arrangement of page addresses in IMAGE TRAC is precise and short. The length of page addresses is 9 bytes. From the first 3 bytes we know the platter on which the data reside, while from the next two pairs of 2 bytes we can locate the sub- and sub-sub-directory within that platter, and from the last 2 bytes we know exactly where the page resides (see Chapter III, section C.4.c.iv). The clear advantage of this addressing scheme is that the actual search of the data is limited to one platter only, since from the master listing the user knows which platter to insert, leaving a maximum search length of about ten thousand pages--the average number of pages that can be stored on one side of an optical platter with capacity of 470K bytes per side and an average page (file) size of 50K bytes. Many existing systems, however, have addressing schemes that range from 32 bytes to over one hundred bytes. Since the speed of data retrieval is directly related to the address length and the database size, a carefully designed addressing scheme focused on speed of data retrieval is much to be desired. Indeed in a large database, each increase of a single byte in the search path can increase the search time by an order of magnitude.

E. THE SYSTEM IS HIGHLY FLEXIBLE[4]

To protect the customer's initial investment, a system must be able to change computing environments and to allow for the easy addition of peripherals. IMAGE TRAC can run on any IBM-PC clone with Intel 60386 architecture. This class of machines currently dominates the PC market and yet is modestly priced. (Many faster machines are available and IMAGE TRAC can be upgraded for even better performance). Many customers want an imaging system but don't wish to upgrade to a new machine; IMAGE TRAC is just right for them. IMAGE TRAC may start as a stand-alone system, but can be upgraded easily to include multiple storage devices, placement on a network, the addition of a file server to the existing (image) server, and I/O, including electronic mail, fax machines, and other office equipment. For example, the ***Configuration*** window allows for the connection of new devices to the system. The **Help** function is easily expanded when new devices are added to the system.

The **Folder Title** and the **Folder Subject Title** can be modified by customers. Finally, creating a new window is as easy as creating the corresponding ".msk" and ".key" files.

---

[4] Component (4), above.

F. MODEST PRICING[5]

Most existing image systems cost over fifteen thousand dollars; the IBM imaging system costs close to a million dollars! IMAGE TRAC, on the other hand, is a stand-alone system aimed at small offices and small businesses with good growth potential. Customers can run the image system on their own hardware (assuming that they have optical drive hardware--which costs approximately three thousand dollars), provided it is IBM-compatible. To keep the office orderly, to conserve paper, and to prevent the loss of originals, every office should have a document image system at an affordable price. Approximate prices for such a system are:

(1) Operating System (e.g., MS-DOS)	$250.00
(2) Optical Disk Driver (e.g., Corel with Compression/Decompression)	$700.00
(3) Imaging System (IMAGE TRAC™)	$500.00

Between the reduction in cluttered office space and time spent on filing, purchase of a document image system is an almost trivial investment that will be recovered in no time at all. With the use of such a system, the company can look forward to such desiderata as better customer service, efficient office management, safer record-keeping, faster file and message transfer between offices, and better record updates.

---

[5] Component (5), above.

G. AVAILABILITY OF TECHNICAL SUPPORT[6]

Service for much-used machines is available almost everywhere. Hence, since IBM-PC clones are very popular machines, hardware technical support is available across the country, at PC Warehouse, Computer Land, Business Land, Radio Shack, and numerous other computer stores.

Maintenance of and customer service for the IMAGE TRAC system can be maintained with ***PC-Anywhere*** software. With any PC, a telephone line, and a modem connection, the IMAGE TRAC distributor will be able to communicate with the customer. The distributor can examine the customer's software, take over the console (remote control), establish the setup, and help the customer correct errors.

IMAGE TRAC distributors can even establish automatic system backup (with on-site customer cooperation for mounting and dismounting the backup platter), routine database check-up, and the generation of statistical reports about the system. Bulletin boards can be set up among the user community allowing free information exchange and timely distribution of system updates. As a result, users and distributors form a closely knit environment, with each aware of the other's perspective and needs.

---

[6] Component (6), above.

H. SPACE EFFICIENCY[7]

Many existing imaging systems store the updated directory as well as the image file on the optical drive. The advantage of this arrangement is that the optical platter is self-contained. For a WORM drive, however, the directory information is not erasable, and each time a file is updated, a new directory will have to be created. In systems such as Infotreev and Recall, for example, more than half the optical platter storage consists of directory- and database-structure information. For industries that require WORM technology such as banking and finance, the above systems simply waste too much space on non-image data. In contrast, IMAGE TRAC uses the magnetic hard disk to store the directory- and database-structure information, while the optical disk is dedicated to image storage.

I. A SUCCESSFUL TRIAL RUN

A prototype of IMAGE TRAC was installed in New ERA Electric Cooperative in Athens, Texas. The company keeps the original customer application forms for future reference. If a customer wants changes, i.e., address, name, or meter changes, the original document must be fetched and modified to reflect the new information. Before the imaging system was installed, it took a clerk about two weeks to complete a job; with the system installed, it only takes a clerk one to two days to complete all the paper work. In an emergency, documents can be retrieved within an hour. Before the imaging system

---

[7] Component (7), above.

was installed, the cooperative's basement was called the "dungeon" and was used to keep all the original documents. There were file cabinets from wall to wall and many documents were piled up in the hallway waiting to be filed. After six months with the imaging system, most of the file cabinets were gone, a conference room was set up in the basement, and the papers in the hallway are gone!

V. FUTURE RESEARCH DIRECTIONS

A. RETRIEVING MANUSCRIPTS BY SUBJECT

In document management systems, retrieval of documents is now limited to word or Boolean search. Hence, database searches look for documents containing a specific word or phrase. A search by subject--that spans the entire extent of the database--is much to be desired, since the typical user would like to retrieve information on a topic within a given subject. The system should provide him with a list of related articles weighted so that the most related article is presented first, and so on. Naturally, this raises the possibility of a revolution in bibliographic research and should be of great interest to not only those in the academy but also library science professionals and market research professionals.

B. INTEGRATION OF IMAGE TRAC WITH OBJECT CHARACTER RECOGNITION SYSTEMS

With image documents, the user cannot edit or modify the document with the word processor of his choice, nor can he copy or extract text from the document. An image-to-text converter is much to be desired: The user can then extract and move image information to his word processor. Object Character Recognition (OCR), an emerging technology, provides the means for transforming pixel-by-pixel (bit map) input to ASCII output. Hence, OCR provides a specific mechanism for users to incorporate image

information with text as the need arises.

IMAGE TRAC can be combined with a scanner that has OCR capability. When the price and performance (the fidelity of many current OCR scanners is extremely low) of OCR scanners reach the level currently desired by the computing community, IMAGE TRAC can be linked with an OCR scanner using the **CONFIGURATION** screen.

C. INTEGRATION OF IMAGE TRAC WITH INFORMATION SERVICES

As telecommunication becomes more and more popular, and large databases are more easily accessible (say the Dow Jones financial database or the CompuServe consumer database), a growing number of users will subscribe to such services. The integration of such information services with IMAGE TRAC is highly desirable because users can then incorporate their own information with the information provided by outside databases. As Alice said, "and what is the use of a book without pictures or conversations?"[10] Alice is, of course, right. We now know that the right hemisphere of the brain is guided largely by images, while the left hemisphere is guided largely by words.[11] With IMAGE TRAC, someone with optical platters of John Tenniel's fine illustrations can combine them with a straight-text edition of the Alice stories.

D. INCORPORATION OF HIGH-RESOLUTION MONITORS, COLOR IMAGES, AND COLOR PRINTERS

Many users would like to have color display images, which compare favorably to

ordinary color photographs. For example, high-resolution monitors are required for draftsmanship and architectural drawing. Unfortunately, high-resolution monitors, color scanners, and color printers remain very expensive. However, as they become affordable, they would be natural enhancements of imaging systems like IMAGE TRAC.

VI APPENDIX

A. MASK.H

```
/*****************************
structure of the screen:
*****************************/
#ifdef MSC
#include "types.h"
#include "stat.h"
#define  O_RAW     O_BINARY
#endif
#define MSKNAME  16
#define DBNAME   8
#define FIELDNAME      8
#define LINESIZ     256
#define MAXLINE   256
#define TEXTMAX  256     /* max. char per text item  */
#define MAXFLD    32
#define NARG        20
#define M-FORM     1
#define M_REPORT 2
#define M_REP_SEL      3
#define M_MENU    4
#define M_FHLP    5
#define M_LMENU  6
#define M_BFORM  7

/*BOX TYPE */
#define        NO_BDR     0       /* NO BOX */
#define        SNG_BDR    1       /* single line BOX */
#define        DBL_BDR    2       /* double line BOX */
#define        REV_BDR    3       /* reverse vidio BOX */
#define        DOT_BDR    4       /* dot  BOX */

/* STATUS MODE */

#define MEDIT        101    /* edit input mode */
#define FLD_VR     0x0001 /* verify field */
#define FLD_NEH   0x0002 /* not echo field input data */
#define FLD_PROT  0x0004 /* protected mode - field input data not allow */
#define FLD_COMPUTE    0x0008 /* compute mode - field input data not allow */
/* file name sizes */
```

```
#define KFN_SIZE  32      /* key definitions file name size*/
#define KDEF_SIZE 32      /* key definition  size*/
#define FFN_SIZE  32      /* field file name size */

/* how to handle input field */

#define FILL        1
#define NFILL       2
#define UPDATE      0x4
#define SEL_RET     0x1
#define REPORT      0x2
#define END_SAVE    0x4
#define END_QUIT    0x8
#define RET_MAIN    0x10
#define UNDO        0x20
#define REP_SEL     0x40
#define RET_FUN     0x80
#define EXIT_ACT    0x100

/* follwoing are definitions of keys */

#define BACK        '\b'
#define BTAB        6
#ifndef K_TAB
#define K_TAB       '\t'
#endif
#define BACKFLD     8
#define ENTER       '\027'  /* ^W as save */
#define QUIT        '\004'  /*  as quit */
#define ESC         '\033'  /*  as quit */
#define PAGE_NEXT       '\006'  /* ^F as next page for UNIX */
#define PAGE_PRV    '\002'  /* ^B as previous page for UNIX */

 /* function key function table index */
#define FNF1            1
#define FNF2            2
#define FNF3            3
#define FNF4            4
#define FNF5            5
#define FNF6            6
#define FNF7            7
#define FNF8            8
#define FNF9            9
#define FNF10           10
#define FNESC           11
#define FNCR            FNESC+1
#define FNLEFT          FNESC+2
```

```
#define FNRIGHT   FNESC+3
#define FNPNEXT   FNESC+4 /* NextPage */
#define FNPPRV    FNESC+5 /* PrvPage */
#define FNALT_D   FNESC+6 /* Date insertion */
#define FNCRL     FNESC+7 /* CR at Last field */
#define MAX_KEY   20

#define CRL       -99

#define DATE_LEN 8
#define PHONE_LEN     13

#define T_DATE    'd'
#define T_PHONE   'p'
#define T_DOUBLE  'b'
#define T_INT     'i'
#define T_CHAR    'c'
#define T_TEXT    's'
#define T_FLOAT   'f'
#define T_MONEY   'm'

typedef struct _pos {
        unsigned row;
        unsigned col;
        unsigned int index;   /* index of that field in the row*/
} POS;

typedef struct _text {
unsigned row;
unsigned col;
char    *txptr;
int     len;                    /* length of that text field */
char    att;                  /* text attribute */
char    res;                    /* reserved */
}S_TEXT;

#define FPAD        3               /* padding for FIELD must be 0-3 */
typedef struct _field{
        char id;                /* ?? check informix for the type */
        char dbid;            /* database table id */
        char fid;               /* field id within the db table */
        char type;            /* type of the field   char, long,                    int,
double, float, money, date,                         phone.      ...      */
        char name[FIELDNAME+1];
        char att;             /* field attribute */
        char update;          /*  update flag          */
        char dlen;            /* max string size of the field */
```

```
        char dsize;          /* size of field data     */
        char opt;            /* option (reserved)      */
        char *dbuf;          /* pointer to the field data */
        short status;        /* status: verify, echo, protected,
                                compute  */
        unsigned int offset; /* offset from the beginning of fld data block */
        POS  pos;            /* position within the window */
        struct _field  *next; /* next field */
        struct _field  *prv;  /* previous  field */
        struct _field  *up;    /* up one row field */
        struct _field  *down; /* down one row  field */
} FIELD;

typedef struct _mask{
        char name[MSKNAME];
        char dbase[DBNAME];   /* (needed ???) */
        char id;
        char fnumber;          /* number of fields */
        char fmax;           /* max number of fields */
        char type;           /* type of screen: TDYNAMIC or
                             TSTATIC dynamic: len of screen
                             will vary as result of query */
        char srow;           /* # of row in the screen */
        char scol;           /* # of col in the screen */
        char r_row;          /* mask size of row */
        char r_col;          /* mask size of col */
        char att;            /* attributes */
        char box;            /* box type */
        POS   pos;           /* position of the mask should be
placed */
        POS   cur;           /* current cursor position of the
mask */
        int stat;            /* status */
        S_TEXT *tx_hd;       /* text header tables */
        char *tex_ptr;       /* pointer to the text section */
        int  tex_item;       /* number of the text items in the
mask  */
        unsigned int tex_size; /* size of the text section    */
        int  item;           /* numbet of select items on the
screen */
        FIELD *fptr;         /* pointer to the fields */
        FIELD *update;       /* pointer to the update fields */
                             /* use by input(), query(), update(),..(?) */
        unsigned int fld_size; /* size of the total field data        */
        struct _mask *next;  /* next mask */
        struct _mask *prv;   /* prv mask */
        struct _mask *np;    /* next page */
```

```
        struct _mask *pp;     /* prv page */
        struct _mask *smptr;  /* status(or key) mask */
        char patt;            /* partial att */
        char prow0;
        char prow1;
        WINDOW *win;      /* window pointer */
        char con_mask[MSKNAME]; /* continue (next ) mask_id */
        int con_flag;         /* continue flag (if there is next
                              page) */
        char keydef[KFN_SIZE]; /* file name of key defintion */
        char fldhelp[FFN_SIZE]; /* file name of field help */
} MASK;

typedef struct _mfun{
        char name[18];
        int  (*func)();
} MFUN;

#define MFUN_MAX       50

#ifdef MSC
#define M_NOOP    -1
#else
#define M_NOOP    255
#endif
#define mlogic(i)     ((i)?1:0)
MASK *mask_free();
```

B. IMG.H

```
/*************************
    img.h
*************************/


#define  FOLDER      0
#define  DOC         1
#define  PAGE        2
#define  PASSWD      3
#define  DEPT        4
#define  ACCESS      5
#define  PLATER      6
#define  NUM_DBS     7

#define  FLDTX_SIZ   24
```

```
#define     LEN_DEPTID   2
#define     LEN_DEPTNAME 16
#define     LEN_AMODE    3
#define     LEN_CITY     16
#define     LEN_DATE     8
#define     LEN_DOCID    4
#define     LEN_DOCNAME  35
#define     LEN_ADDR     36
#define     LEN_DOCTYPE  3
#define     LEN_DRIVETP  1
#define     LEN_FID      8
#define     MAXLEN_FID   21
#define     LEN_FNAME    16
#define     LEN_GROUP    3
#define     LEN_LNAME    18
#define     LEN_LOGIN    8
#define     LEN_NAME     36
#define     LEN_NOTE     10
#define     LEN_NUMPAGE  3
#define     LEN_PAGEID   4
#define     LEN_PADDR    9
#define     LEN_PASSWD   16
#define     LEN_PLTNUM   3
#define     LEN_PNAME    8
#define     LEN_SMASK    1
#define     LEN_STATE    2
#define     LEN_UNAME    16
#define     LEN_USRID    3
#define     LEN_ZIP      9
#define     MAXKEY_LEN   41 /* must > MAXLEN_FID + LEN_DEPTID +
                               LEN_DOCID + LEN_P_PAGEID + 1 */

#define  LEN_DRIVE    8

/* FOLDER DB field offsets */
#define  F_FID        0
#define  F_NAME       1
#define  F_ADDR       2
#define  F_ADDR1      3
#define  F_SMASK      4
#define  F_OWNER      5
#define  FOLDER_FLDS  6

/* DOC DB field offsets */
#define  D_DOCID      0
#define  D_DOCNAME    1
#define  D_DOCTYPE    2
```

```
#define  D_NUMPAGE  3
#define  D_DATE     4
#define  D_PLTNUM   5
#define  D_DRIVE    6
#define  D_SMASK    7
#define  D_OWNER    8
#define  D_NOTEPAD  9

#define  DOC_FLDS   10

/* PAGE DB field offsets */
#define  P_PAGEID    0
#define  P_PADDR     1
#define  P_DRIVETP   2
#define  P_CDATE     3
#define  P_DDATE     4
#define  P_OWNER     5
#define  P_MODIFIER  6
#define  PAGE_FLDS   7

/* USER PASSWORD DB field offsets */
#define  U_USRID     0
#define  U_LOGIN     1
#define  U_UNAME     2
#define  U_GROUP     3
#define  U_PASSWD    4
#define  PASSWD_FLDS 5

/* DEPT DB field offsets */
#define  S_DEPTID    0
#define  S_DEPTNAME  1

#define  DEPT_FLDS   2

/* Access control  DB field offsets */
#define  A_DEPTID    0
#define  A_LOGIN     1
#define  A_MODE      2
#define  ACCESS_FLDS 3

/* PLATER DB field offsets */
#define  L_PLTNUM    0
#define  L_PNAME     1
#define  L_PADDR     2
#define  TER_FLDS    3
```

```
/* FOLDER DB index field */
#define  I_FFID      0
#define  I_NAME      1
#define  I_ADDR      2
#define  F_INDEXNUM  3

/* DOC DB index field */
#define  I_DOCID     0
#define  I_DOCNAME   1
#define  I_DOCDATE   2
#define  D_INDEXNUM  3

/* PAGE DB index field */
#define I_PAGEID  0
#define I_PADDR   1
#define I_PDTYPE  2
#define P_INDEXNUM    3

/* USER PASSWORD DB index field */
#define I_USRID    0
#define I_LOGIN    1
#define U_INDEXNUM    2

/* DEPT DB index field */
#define I_DEPTID  0
#define S_INDEXNUM    1

/* DEPT ACCESS DB index field */
/* #define I_DEPTID       0 */
#define A_INDEXNUM    1

/* PLATER DB index field */
#define I_PLTNUM  0
#define L_INDEXNUM    1

#define NUMINDEX
F_INDEXNUM+D_INDEXNUM+P_INDEXNUM+U_INDEXNUM+S_INDEXN
UM+A_INDEXNUM+L_INDEXNUM

#define MAXNUMINDEX  8  /* max number of index per db is */
typedef struct disp{
      int fld_id;      /* e.g. D_COMP, if -1 means LAST+FIRST */
      int idx_id;      /* e.g. I_COMP, if -1 means not a index fld
*/
      char *str;       /* description of the field */
}DISP;
```

```
#define MARK_POS 7+1
#define MARK_CHR        '*'
#define NOTE_CHR 0x0d
#define DB_NEW          0x1ff

typedef struct _skey{
        char dbuf[MAXKEY_LEN];
        char  name[32];       /* name of the index fld */
        int  size;
        struct _skey *next;
}S_KEY;

/*****************************************************************
Searching Parameters
*****************************************************************/
#define S_LNAME 11
#define NAME_SIZE  S_LNAME+1
#define SI_SIZE 200          /* number of str in each array */
#define SI_MAX  3000         /* Total MAX size of search arrays */
#define SI_NUMB SI_MAX/SI_SIZE /* number of array */

/*****************************************************************
MACRO definitions
All macor must starting with Upper case.
*****************************************************************/
/* if the ptr is the  last one in search array */
#define S_bottom(ptr)    ((ptr+1) > &s_index[n_match-1])
#define S_top(ptr)       (ptr < = &s_index[0])
#define Is_zom_msk(mptr) (strncmp((mptr)->name,ZOOM_MSK,
          strlen(CHG_MSK) - 4) ? 0:1)
#define Is_ch_msk(mptr)  (strncmp((mptr)->name,CHG_MSK,
        strlen(CHG_MSK) - 4) ? 0:1)
#define Is_add_msk(mptr) (strncmp((mptr)->name,ADD_MSK,
         strlen(ADD_MSK) - 4) ? 0:1)
#define Is_srch_msk(mptr) (strncmp((mptr)->name,DSP_SRCH_MSK,
           strlen(DSP_SRCH_MSK) - 4) ? 0:1)
#define Is_brw_msk(mptr) (strncmp((mptr)->name,BROWSE_MSK,
          strlen(BROWSE_MSK) - 4) ? 0:1)


#define LENGTH   160
#define ADD      0
#define UPD      1
#define DEL      2
#define PROTECT  3
#define WIN_SIZE 17
#define FR_SIZE (1 * WIN_SIZE)
```

```
#define FR_WIDTH 82

#define FLINE_SIZ 128

typedef struct{
        char *dbid;      /* ptr to the db */
        dBFIELD *dbfld;  /* ptr to the field */
        char *fp;        /* pointer to the ndx file */
        int type;        /* index type */
        int fld_idx;     /* index to the fields array */
} I_FILE;

#define IDX_PRIM  1
#define IDX_SECOND      2

#define EXIT_MSK        "exit.msk"
#define BOX_MSK         "db_box.msk"
#define DEMO_MSK        "demo.msk"
#define DEMO_INTRO_MSK "demo_int.msk"
#define INTRO_MSK       "intro.msk"
#define LOGIN_MSK       "login.msk"
#define DEL_MSK         "delete.msk"
#define DEL_MSG_MSK     "del_msg.msk"
#define CHNG_MSG_MSK    "chng_msg.msk"
#define DIAL_MSK        "dial.msk"
#define UTIL_MSK        "util.msk"

/** Application Specific Parameter Definitions **/

#define FOLDER_DB "folder.dbf"
#define DOC_DB    "doc.dbf"
#define PAGE_DB   "page.dbf"
#define PASSWD_DB "passwd.dbf"
#define DEPT_DB   "dept.dbf"
#define ACCESS_DB "access.dbf"


#define DB_BAK          "img.bak"

#define BROWSE_MSK     "db_out.msk"
#define DSP_SRCH_MSK  "db_srch.msk"

#define ADD_MSK         "inf_add.msk"
#define CHG_MSK         "inf_ch.msk"
#define ZOOM_MSK        "inf_zom.msk"
#define SEARCH_MSK      "i_search.msk"
#define OUT_MSK         "inf_out.msk"
```

```
#define SETUP_MSK     "inf_set.msk"
#define KEYWD_MSK     "keyword.msk"
#define SAV_DB_MSK    "db_save.msk"
#define APPN_DB_MSK       "db_appn"
#define CREATE_DB_MSK "create.msk"
#define FULLHANDLE_MSK "fullp.msk"
#define DB_EXP_MSK    "db_exp.msk"
#define SET_ADD_MSK   "set_add.msk"
#define PHONE_MSK     "phone.msk"
#define DATE_MSK      "date.msk"

#define LOGFILE       "img.log"

#define NON_IDX   -1

#define NUM_SKEY          12      /* number od search canned keywords */


#define LEN_PATH  64      /* path where the database is */
#define LEN_DBNAME      32      /* Name of the dbase */


typedef struct{
        char text[FLDTX_SIZ];
        int len;
} IMGFLD;

typedef struct{
        char dbname[LEN_PATH+LEN_DBNAME+1];
        int  num_fields;
        dBFIELD *fields;
        int  num_idx;
        I_FILE          *idx;   /* array store offset */
        char    *fti;           /* field to index table */
        int  *d_offset;         /* array store offset */
}IMGDB;

#define TCH         '^'
#define PAN_SCAN    'P'
#define CANON_SCAN 'C'
```

VII BIBLIOGRAPHY

[1] *Dataquest*, September 1988.

[2] W. I. Grosky, R. Mehrotra eds, "Image Database Management", *COMPUTER*, IEEE Computer Society, December 1989.

[3] S. K. Chang, C. W. Yan, D. C. Dimiroff, and T. Arndt, "An Intelligent Image Database System", *IEEE Transactions on Software Engineering*, Vol. 14, Number 5, March 1988, pp. 681-688.

[4] Legend Technologies-supplied documentation and demonstration diskette, 1989.

[5] Microsize-supplied documentation and demonstration diskette, 1989.

[6] Dataworks-supplied documentation and demonstration diskette, 1989.

[7] Acctex-supplied documentation and demonstration diskette, 1989.

[8] BankSystems-supplied documentation and demonstration diskette, 1989.

[9] SCANBASE system diskette and documentation, 1990.

[10] Lewis Carroll, *ALICE's Adventures in Wonderland* in Edward Guiliano, Ed., *The Complete Illustrated Works of Lewis Carroll,* New York; Crown Publishers, Inc., 1982.

[11] Thomas R. Blakeslee, *The Right Brain,* New York: Berkeley Books, 1980.