DATA BASE FOR ADMINISTRATIVE SYSTEM
AN INTERACTIVE DIALOGUE FOR THE CREATION,
MAINTENANCE AND QUERYING OF A DATA BASE
REPRESENTATIVE OF THE ANATOMY
OF AN ADMINISTRATIVE SYSTEM

by

PRATIMA MATHUR, B.E. (HONS.)

A Project
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Science

McMaster University
February 1980
MASTERS OF SCIENCE (1980)  McMaster UNIVERSITY  Hamilton, Ontario

TITLE: An Interactive Dialogue for the Creation, Maintenance and Querying of a Data Base Representative of the Anatomy of an Administrative System

AUTHOR: Pratima Mathur, B.E. (Hons.) (Pilani, India)

SUPERVISOR: J. S. Masterson

NUMBER OF PAGES: vii, 127
ABSTRACT

The logical structure of a database is given, and is representative of the anatomy of an administrative system, and of the personnel organization by which it is operated. This project will adopt an appropriate host computer, terminal, and physical representation of the database, for the main purpose of developing a user-terminal dialogue by which the database can be created and maintained. The dialogue should be as easy and flexible to use by an audit analyst as is possible, while at the same time imposing a systematic and disciplined approach to the task.
ACKNOWLEDGEMENTS

I would like to thank my supervisor Mr. J. S. Masterson for his guidance, supervision and encouragement during the course of this work and preparation of this thesis.

I would also like to thank Antoinette Spinosa for her excellent typing.

Finally, I would like to thank my husband, Praveen, for his patience, understanding and continuous encouragement throughout the preparation of this work.
Dedicated to my son, Anubhav, who decided to arrive in this world during this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter I: Problem Definition and Background</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Background of the auditing process</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Background of the graphic auditing system</td>
<td>3</td>
</tr>
<tr>
<td>1.3 General capabilities of the HP2647A</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Mode of use of the graphical auditing system</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Objectives of this project</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter II: Problem Analysis</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Specific features of the HP2647A</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Data base contents</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Remote data base</td>
<td>14</td>
</tr>
<tr>
<td>2.4 Processing requirements</td>
<td>15</td>
</tr>
<tr>
<td>2.5 Choice of Terminal/Computer Language/ File/Organization/Data Communication Protocols</td>
<td>18</td>
</tr>
<tr>
<td>2.6 Design of forms and processing features</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter III: System Definition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 System configuration</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Data specification</td>
<td>30</td>
</tr>
<tr>
<td>3.3 Program specification</td>
<td>38</td>
</tr>
<tr>
<td>3.4 Sample test cases</td>
<td>59</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter IV: Result and Conclusion</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Evaluation of results</td>
<td>65</td>
</tr>
<tr>
<td>4.2 Were the objectives achieved?</td>
<td>66</td>
</tr>
<tr>
<td>4.3 Possible improvements</td>
<td>68</td>
</tr>
</tbody>
</table>

APPENDIX A: PROGRAM LISTINGS 71

APPENDIX B: SAMPLE OUTPUTS 111

BIBLIOGRAPHY 127
LIST OF FIGURES

1. System Directory Short Form 01  
   2. System Directory Long Form 02  
   3. Schema Diagram  
   4. System Flow Chart  
   5. Title Page  
   6. Menu

Page
23
24
26
29
40
41
CHAPTER 1
PROBLEM DEFINITION AND BACKGROUND

The general objective of this project is to develop a user-terminal dialogue for creating, maintaining and querying a data base for analytical auditing within an Administrative System.

1.1 Background of the auditing process

An audit is an examination of a company's financial statements by a firm of independent public accountants. The audit consists of a searching investigation of the accounting records and the evidence supporting these financial statements. Auditing has been practised in one form or another for an extremely long period of time. It has been known from the time of the Babylonian and the Assyrian civilizations that society was concerned for the safety of assets entrusted to public officials. One of the main contributions of the Grecian civilization to the development of auditing theory was the concept of public audits. For example, during the building of a temple in Athens, interested persons could review the cost of construction by examining the marble placed in front of the construction, showing the cumulative amount expended. An auditing function is further found in the history of feudal England. From the twelfth to seventeenth centuries, the lord of the manor, who was not a totally independent person, used to perform the function somewhat similar to that performed by auditors now-a-days.
During the middle ages, the Industrial Revolution began and the English manor lost its self-sustaining status. The auditing profession was well established in England by the end of the nineteenth century due to the rise in commercial and industrial firms.

In North America, very little auditing was done before 1900. At that time auditing was mainly concerned with the detection of fraud. In the first half of the twentieth century, the direction of audit work had moved away from fraud detection towards determining whether the financial statements give a fair picture of financial position, operating results and changes in financial position. This shift of emphasis was a response to the needs of the millions of new investors in corporate securities. In the last fifteen years, the detection of large scale management fraud is again of growing importance as a result of a dramatic increase in the number of lawsuits charging that management fraud has gone undetected by the independent auditors.

The modern business world is changing; business units are growing larger and accounting systems are becoming more and more complex. To suit this changed environment, auditors need a system-oriented auditing technique. A system-oriented auditing technique based on flow chart analysis and limited procedural tests is called Analytical Auditing.
Analytical Auditing is a very recent development and is now used all over North America.

There are two major objectives of Analytical Auditing.

The primary objective is to determine through an analysis of accounting systems and internal control, the fairness of financial statements.

Secondly, it is to give the client timely suggestions for strengthening the system of internal controls, for increasing the efficiency of the accounting system, and for improving the clients financial and tax planning.

1.2 Background of the graphics auditing system

A logical component for meeting the above objectives of Analytical Auditing is the flow chart technique for auditing. Flow charts are the means to analyze the system. This is the most commonly used technique because a visual review of flow charts provides a faster and more reliable indication of the existence of weakness in the internal controls. The flow chart technique is also a faster way to obtain information during the system review, because it is a form of shorthand. It also helps the auditor to visualize the system under review and to see immediately what loose ends have been left unexplained. These may include unnecessary handling of certain documents, inefficient routing of certain documents, unnecessary document copies or records.
Drawing flow charts manually is very tedious and time consuming. The technology of computer graphics has been developed in the last several years and is being rapidly applied and expanded. Terminals are available with a microprocessor, and these provide interactive graphics features under user or program control. Terminals can be operated in stand-alone or on-line mode. Computer graphics are new, practical, cost effective and readily available. Drawing flow charts automatically on the display screen using graphics features, is feasible and economically efficient. There are several graphics terminals available in the market, including the HP2647A.

1.3 General Capabilities of the HP 2647A

This intelligent graphics terminal uses a microprocessor under firmware control. It has a local BASIC language which can be loaded from a cartridge tape. It has two tape units which provide mass storage capability. The terminal can be used for either stand-alone or on-line operations. Two separate memories are provided in this terminal; one is for alphanumeric display and the other is for graphics display. It has the ability to store three alternate character sets provided that an optional display enhancement feature is added. It has inverse video, half bright, underline and blinking field display features. Blinking field is good for displaying error messages,
inverse video and half bright fields are good for constructing the forms. There are some firmware programs residing in ROM which provide some powerful features to the terminal such as dynamic memory allocation, self-test transparent control codes and off-screen storage. Data can be read into the BASIC work space from the data communication line, from cassette tapes, from the display, and from the keyboard.

The terminal can be connected in a variety of network configurations. Data can be transmitted character by character or it can be transmitted as a variable length block over the communication line. The terminal can operate in full duplex, half duplex, synchronous or asynchronous transmission mode. Several other data communication facilities are also provided.

1.4 Mode of use of the Graphical Auditing System

The data base for the graphics auditing system will reside on a supporting remote computer and interaction with the terminal will generate flow charts and alphanumeric displays. The data base will contain all the parameters necessary to define the administrative system for Analytical Auditing. This data base will be created once and will be maintained and queried from time to time.

The end users of the Graphics Auditing System would be administrative staff, auditors, and clerical staff. The commands to maintain the data base would be to delete or change the previously stored information or add new and/or
extend the existing information. This job would be performed by clerical staff. Administrative staff or auditors can query the data base to get information on systems, sub-systems or procedures, which are currently in operation. Hence they will have the access and ability to recommend changes to improve the efficiency of internal control.

Keeping the end users in mind, the data entry phase will adopt a form filling approach on the display screen. The queries will be small and simple.

McMaster University is developing a Graphics Auditing System for analytical auditing. The data entry phase of the project will be utilized in the McMaster Graphics Auditing System.

1.5 **Objectives of this project**

1) To develop a user dialogue with the HP2647A Graphics terminal in the context of the Graphics Auditing System, which will:-

(a) provide for data entry to a local tape cassette

(b) provide for bulk updating of the data base on a remote computer from transactions on the local data cassette.

(c) provide for batch or interactive querying of the remote data base, and the optional generation of hard-copy.
2) To emphasize ease of use in all aspects of the user-terminal dialogue, and especially in the case of local data entry which should be operable by clerical staff.

3) To emphasize robustness of the system, especially in forgiving erroneous key depressions.
CHAPTER II

PROBLEM ANALYSIS

Available facilities have to be taken into consideration to achieve the objectives. The objectives of the system are explained in chapter I.

2.1 Specific features of the HP 2647/A

The HP2647/A is a microprocessor-based graphics terminal and has certain unique features which may be utilized in the development of the project. These unique features of the terminal are discussed below:

(i) Forms Mode: There are three types of fields defined in 'forms mode'.

a) Protected field.

b) Unprotected field.

c) Transmit-only field.

The 'on' and 'off' control of the 'forms mode' can be handled through programming the computer itself, or by commands recorded on cartridge tapes.

a) Protected field: Fields are protected from being overwritten, and inhibited from transmitting data from the display screen. When the 'forms mode' is turned 'on' the complete display screen is 'protected' unless any field is specifically defined as 'unprotected' or 'transmit-only'.

b) Unprotected field: Fields can be specifically defined as 'unprotected fields'. Data can be
written into and transmitted from the 'unprotected field'. After reaching the end of one unprotected field, the cursor moves to the next 'unprotected field'. The cursor can be placed at the beginning of the previous 'unprotected field' by using the back tab.

c) Transmit-only field: The transmit-only field is similar to the protected field except that the data can be transmitted to the computer. This field has to be defined explicitly.

d) Data checking: While in forms mode there is a facility provided for data checking in 'unprotected' and 'transmit-only' fields. Fields can be defined for the following types of data

1) Alphabetic field: A thru Z, a thru z, and space.

2) Numeric field: 0 thru 9, -, +, ., and .

3) Alphanumeric field: All keyboard characters.

(ii) Special Character Sets: Four different sets of characters can be displayed on the screen. Each character set can be up to 128 characters or symbols. These are a) math symbols, b) line drawing symbols, c) roman symbols, and d) large character set. The line drawing symbols and the roman symbols are the most helpful in this project. Fairly complex forms can be drawn on the screen using the line drawing symbols. Switching from one character set to another
character set can be done on a character by character basis i.e. a character from one set can be displayed just next to a character from another set. This is done by defining one or more character positions in a line to be from the selected alternate character set.

(iii) Display enhancement: The display enhancement feature enables the terminal to have extra display features apart from the standard normal and inverse video. Characters can be displayed in a) half bright, b) underline, c) inverse video, d) blinking, and e) any combination of the above. The display enhancement feature can be used to improve the cosmetics of the data entry form.

(iv) Tape units: There are two cassette tape units on the terminal. Forms, and edit programs for forms can reside on these tapes. Specific forms can be recalled for display on the screen. After filling out the form, data can be stored locally on the tape for subsequent bulk update of a data base.

(v) Display Workspace: There are four display workspaces in the terminal. Information can be displayed and can be read from any one of the display workspace. Only one display workspace can be displayed on the screen. This feature can be helpful in transferring common or global data from one program to another program.

(vi) Data Communications: The terminal can be connected
in a variety of network configurations such as a) hardwired to a computer, b) hardwired through other terminals to a computer, c) connected to a computer through a modem, d) connected to a computer through other terminals to a modem.

The terminal can be used with a variety of interfaces. Once the interface has been selected, the terminal can be configured to operate with a variety of protocols, parities, and data formats. Data can be sent in full or half duplex mode.

Protocols, or sets of rules and procedures, have to be established for orderly transfer of data between the computer and the terminal. The protocol will determine the sender and receiver of the data. It also would recover data lost by communication error. The following protocols are provided in the terminal:

a) Character Protocol: Character protocol transmits a single character at a time. Data checking is done on individual characters.

b) Block Protocol: Block protocol transmits a block of characters at a time. Data checking is performed on an entire block of data. Block transfer allows the user to verify and correct data before sending it to the computer.

The COPY command copies 7 bit ASCII data from a
specified source to a specified destination. Data can be a file, a line or all on the source device.

(vii) BASIC Language: The terminal provides the BASIC programming language which may be loaded into the terminal from the cartridge tape. BASIC programs for displaying the data entry form and editing it can be written and run on the terminal locally. The following functions in BASIC would be helpful in writing the edit program for the forms.

(a) GETKBD(X): This function returns the keycode in X associated with the key struck. The function value is 1 if a keycode is returned or Ø if no keycode is returned. This function can help in editing keyed data and printing out only those characters which are needed in the form.

(b) DISPINS$(L,X): Returns the string from the display screen starting from current cursor position. The length of the string is the absolute value of L. The value of X will be set to Ø if the end of the data field, or the end of the current line, or the end of the display is not encountered before the specified number of characters are read. X will be set to -1 if the end of the data field, or the end
of the current line is reached, while X will be set to 1 if the end of the display is reached.

(c) MOVCA(R,C): This function moves the cursor to row R and column C where row and column indicate the absolute display memory address.

(d) Execute File: This facility is provided for the loading and execution of BASIC programs using the terminal execute file. Several programs can reside on tape. Selected programs can be loaded from the tape to the terminal memory and executed using the terminal's execute file.

2.2 Data base contents

Information content of the data base is the System Directory. It is the hierarchy of the following entities:

i) System.

ii) Sub-System.

iii) Procedure.

Each entity is described by the following:

i) ID: Identifying code for the entities

ii) Name: Short descriptive term for the entities

iii) Description lines: Ultimate text which identifies the entities.
2.3 **Remote Data Base**

The remote data base will contain those parameters and relationships which are needed for the Analytical Auditing System. Physical representations of the data base could be handled through a data base management system, or through customized data file organization. Choice between the two will depend on several factors such as

i) Choice of computer.

ii) Limits on computer memory.

iii) Choice of programming language.

iv) Time taken to retrieve the data base.

There are several advantages and disadvantages to both approaches. Customized data file organization could be relatively simple to handle, whereas data-bases handled by DBMS could be very complex. Retrieval of information through DBMS would be more time consuming than from a customized data file. Even though there is no great duplication of data in the data base, during creation, update, and retrieval, the loading of a DBMS software package may need memory in excess of what is available.

One of the objectives of this project is to retrieve information from the data base interactively and it is impossible to use the DBMS available on the CDC 170/730 at McMaster University because only 60K of memory is allowed to interactive users. Hence, it is feasible only to use customized data file organization as the remote data base.
Indexed sequential file organization will be used for the data base.

Record Manager is used to perform execution-time input/output, blocking, and record description. Record Manager routines are used commonly by all compilers and can be accessed through the program.

2.4 Processing Requirements

The following are the main phases in the design of the system:

i) Local data entry.

ii) Data communication.

iii) Creation of data base.

iv) Bulk update of data base.

v) Retrieval of information from data base

(on-line and/or batch mode.)

i) Local data entry: Local data entry programs can be run locally at the terminal. Selected forms can be displayed on the screen. Information for the maintenance of the data base can be entered into these forms. These forms also can be edited by a local edit program. After completion of the form, information on the form can be saved in a convenient manner on a cartridge tape. This information can be then used as a working sub-set of the data base. The local operability of this
phase of the system provides easy and quick data entry, minimizes the demand for remote computer communication, and hence improves the input/output performance of the system.

ii) Data Communication: This phase of the system will send the information stored on the cartridge tape to the remote computer and create a permanent transaction file on the disk storage. This transaction file would contain "z" type records and would be used to update the data base.

iii) Creation of Data Base: This phase of the system is used only once at the beginning, to create the data base. The data base will comprise one indexed sequential file (Master file). Input to the Master file updating process will be the transaction file, stored on the disk. The first record of the master file will indicate the number of records in the master file.

iv) Update of Data Base: This phase of the system is run whenever records have to be added, changed, deleted or information has to be extended in the data base. Whenever a record is deleted, all its descendent records will be deleted too. A record is added
only if its parent record exists in the data base.

Input to the update program is the transaction file, stored on the disk, containing all the transaction records. There will be several types of transaction records which will be discussed subsequently. Records in the transaction file would not be sorted at any stage. Output of the update program would be

a) updated master file

b) Print out of error messages describing bad transaction records and update codes.

c) Print out of the number of successful transaction records of each type.

d) Total number of records in the updated master file.

This program can be run on-line or in batch mode.

v) Retrieval of information from data base (on-line and/or batch mode): Queries for retrieving information from the data base can be accepted by this program interactively or in batch mode. An option to get a hard copy would be provided while running the program interactively. The program will prompt for the input and the user will have to supply the input data. If the program is running in batch mode, then the first input card should be blank, otherwise the
information on the first card would be lost.

Information can be retrieved such as

a) List all the systems currently being used.

b) List all the sub-systems and their systems currently being used.

c) List all the sub-systems of a given system currently being used.

d) List all the sub-systems and procedures of a given system currently being used.

e) List detailed information of a given system or sub-system or procedure currently being used etc.

2.5 Choice of Terminal/Computer/Language/ File Organization/ Data Communication Protocols

i) Terminal: HP2647/A graphics terminal is an ideal choice because of its unique features as mentioned previously i.e. forms mode, protected field, unprotected field, transmit-only field, data communications facility, BASIC language etc.

2) Computer: Following supporting computers can be used to implement the system

a) CDC 170/173 NOS.

b) IBM 370/155 OS/MVT.

c) Stand-alone mini computer.

CDC 170/730 NOS is preferred over the IBM 370/155
or stand-alone mini-computer, as it is readily accessible. It has time-sharing, choice of several programming languages, communications interface, and a data base management system. The major disadvantage in this computer is the limited available memory of 60K for interactive users. But the memory constraint can be accommodated if the data base is handled through Record Manager instead of the DBMS software package. Apart from this, programs have to be compiled in batch mode, and permanent files on disk storage have to be created for the compiled object programs, which can be done quite easily.

iii) Language: Record Manager routines can be accessed through COBOL and FORTRAN extended compilers. Both the languages are available on the CDC 170/730. COBOL is more suited to the handling of character data and file organization options, and is widely used in the business world while FORTRAN is not. Hence COBOL is the better choice.

iv) File Organization: Record Manager supports four file organizations

a) Sequential files: It can be extended across any number of volumes. Records, are placed in order of presentation. They may be accessed sequentially only.
b) Word addressable files: Word addressable files are mass storage files. They contain continuous data or spaces for data. They may be accessed either randomly by word address or sequentially.

c) Indexed sequential files: Indexed sequential files have fixed length data blocks and index blocks. Data blocks and index blocks need not be of the same size. Records are stored in data blocks in order of increasing key value. Keys are stored in index blocks according to a collating sequence which may be specified by the user. Records may be accessed randomly by key, or sequentially.

d) Direct access files: Direct access files contain fixed length blocks. Records may be accessed randomly by key or sequentially. Records accessed sequentially are not logically ordered. To implement a randomly organized file in optimal manner, an analysis of the record's key range is desirable. Indexed sequential file organization is chosen for this system as it facilitates both random and sequential access of records and yields reasonably efficient storage utilization.
v) Data Communication Protocol: Block protocol is chosen for data communications while sending file from cartridge tape to the computer for bulk update. It is decided to create only one file on each cartridge tape and transfer it to the computer at a time. The reason is, if the cartridge tape is half filled and all the information from the tape is sent to the computer, then unwanted data would be transferred too.

Character protocol is used while sending the queries to the retrieval program.

2.6 Design of forms and processing features

The System Directory short form 01 and long form 02 are designed as in figure 1 and figure 2. Forms may be drawn using non-graphics features of the terminal and may be saved on a cartridge tape.

A menu would be displayed on the screen to select the form. Selected forms would be displayed on the screen by pressing the appropriate key on the keyboard. Information would be entered into the form. Each line of the form would be edited for its correctness, and an arrow would be pointed to the given entity e.g. in short form 01 the arrow will show whether the name given in the line is a system's name or a sub-system's name or a procedure's name. Any information which is entered by the user may be changed by moving the cursor to that location and
overwriting the information. Once the form is filled up, the information may be saved on cartridge tape by pressing the ENTER key on the keyboard and the form would be cleared for the next entry. The menu would be displayed again by pressing the appropriate key.

The following update codes are used to maintain the data base.

A - Information filled in the form is added to the data base.
C - Information filled in the form is changed in the data base.
D - Information filled in the form is deleted from the data base.
E - Extend the information in the data base by the description filled in the form.

Currency status shows which part of the data base is updated by the update code.

Ø - Update the system which is currently in use.
X - Update the system which is obsolete.
A - Update the prototype A system.
B - Update the prototype B system.
C - Update the prototype C system.

Prototype systems are experimental systems which may be subject to analytical auditing. This will help to do experiments with a system before implementing it.
## SYSTEM DIRECTORY

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>SUB-SYSTEM</th>
<th>PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### UPDATE CODES

- A - ADD
- C - CHANGE
- D - DELETE

### CURRENCY STATUS

- B - CURRENT
- X - OBSOLETE
- P - PROTOTYPE A
- Q - PROTOTYPE B
- C - PROTOTYPE C
- ETC.

**KEY:**

- INVERSE-VIDEO
- HALF-BRIGHT AND INVERSE VIDEO

**Figure [1]**
### Figure [2]

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>ID</th>
<th>NAME</th>
<th>CURRENCY STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**KEY:**
- [Inverse-Video](#)
- [Half-Bright and Inverse-Video](#)

**Legend**
- **Update Code**
  - A - Add
  - C - Change
  - E - Extend

**Currency Status**
- P - Current
- O - Obsolete

**Prototype**
- A - Prototype A
- B - Prototype B
- C - Prototype C
CHAPTER III
SYSTEM DEFINITION

After analysing the problem, it is found feasible to create, the Systems Directory subset of a data base representing administrative systems. It was decided to use COBOL5 interactive programs on the CDC CYBER 170/730 for maintaining and accessing this data base. The local terminal will execute BASIC programs and the data entry forms will be saved on cartridge tape at the terminal. Thus the system will be implemented as above.

3.1 System Configuration

The implemented data base is a hierarchical data base. Hierarchical data base means that any record can have zero or more descendant records but each descendant record can have one and only one parent record, except for each record of the type at the root of the hierarchy which have no parent records.

The schema diagram of the administrative system is shown in fig. [3]. The logical entities and the data items are discussed below. Data items named ID represent identifying codes, NAME represents a short descriptive term for each of the entities.

SYSTEM: This is the highest level grouping of the administrative system.

data-items: ID; NAME
SCHEMA DIAGRAM:

System Directory

Figure [3]
SYS-DESCRIPTION-LINE: This is the detailed text which describes the SYSTEM. Each record contains one line of description.

data-items: LINE-NO; DESCRIPTION.

SUB-SYSTEM: This is a subset of a SYSTEM, which is recorded in a self contained set of flow charts.

data-items: ID; NAME

SUB-DESCRIPTION-LINE: This is the descriptive text of the SUB-SYSTEM.

data-items: LINE-NO; DESCRIPTION.

PROCEDURE: A subset of a SUB-SYSTEM which is a natural grouping of the processing sequence.

data-items: ID; NAME.

PROC-DESCRIPTION-LINE: This is the text which describes the PROCEDURE.

data-items: LINE-NO; DESCRIPTION.

DATA: This is a logical set of data such as a document, book or ledger, permanent or temporary file, and all necessary information in machine readable form. There is a many to many relationship between PROCEDURE and DATA.

PROCESSING: This is a subset of a PROCEDURE which can be performed to produce a specified output from a specified input.
ORGANIZATION: This is a unit which contains Departments, job positions, etc. There is a many to many relationship between PROCEDURE and ORGANIZATION. It is already mentioned before that this project involves the implementation of the System Directory part of the database. (refer to fig. [3]).

The system flow chart which provides a graphical representation of the implemented design for this project is shown in fig. [4]).
SYSTEM FLOWCHART:

Figure [4]
3.2 Data Specification:

The data base is represented as an indexed sequential file i.e. master file. This file contains fixed length type records. Each record is 60 characters long. The primary key is 10 characters long. There are three different kinds of records.

File name: MASTER

Record name: MASTER-PIC

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR-STATUS</td>
<td>X</td>
<td>Currency status</td>
</tr>
<tr>
<td>SYS-ID</td>
<td>XX</td>
<td>System identification code</td>
</tr>
<tr>
<td>SUB-ID</td>
<td>XX</td>
<td>Sub-system identification code</td>
</tr>
<tr>
<td>PROC-ID</td>
<td>XX</td>
<td>Procedure identification code</td>
</tr>
<tr>
<td>LINE-NO</td>
<td>99</td>
<td>Line number</td>
</tr>
<tr>
<td>TOTAL-NO-REC</td>
<td>9(4)</td>
<td>Total number of records in the master file.</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(18)</td>
<td>Blanks</td>
</tr>
</tbody>
</table>

**X stands for alphanumeric character.

*9 stands for numeric character.
<table>
<thead>
<tr>
<th>Field Name</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR-STATUS</td>
<td>X</td>
<td>Currency status</td>
</tr>
<tr>
<td>SYS-ID</td>
<td>XX</td>
<td>System identification code</td>
</tr>
<tr>
<td>SUB-ID</td>
<td>XX</td>
<td>Sub-system identification code</td>
</tr>
<tr>
<td>PROC-ID</td>
<td>XX</td>
<td>Procedure identification code</td>
</tr>
<tr>
<td>LINE-NO</td>
<td>99</td>
<td>Line-number</td>
</tr>
<tr>
<td>NAME</td>
<td>X(22)</td>
<td>Short descriptive term</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(28)</td>
<td>Blanks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR-STATUS</td>
<td>X</td>
<td>Currency status</td>
</tr>
<tr>
<td>SYS-ID</td>
<td>XX</td>
<td>System identification code</td>
</tr>
<tr>
<td>SUB-ID</td>
<td>XX</td>
<td>Sub-system identification code</td>
</tr>
<tr>
<td>PROC-ID</td>
<td>XX</td>
<td>Procedure identification code</td>
</tr>
<tr>
<td>LINE-NO</td>
<td>99</td>
<td>Line number of description</td>
</tr>
<tr>
<td>DESCR</td>
<td>X(50)</td>
<td>One line of description</td>
</tr>
</tbody>
</table>
A pictorial layout of the record is shown below:

<table>
<thead>
<tr>
<th>CURR-STATUS</th>
<th>SYS-ID</th>
<th>SUB-ID</th>
<th>PROC-ID</th>
<th>LINE-NO</th>
<th>FILLER</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X</td>
<td>X(22)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CURR-STATUS</th>
<th>SYS-ID</th>
<th>SUB-ID</th>
<th>PROC-ID</th>
<th>LINE-NO</th>
<th>FILLER</th>
<th>DESCRIPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X</td>
<td>X(50)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CURR-STATUS</th>
<th>SYS-ID</th>
<th>SUB-ID</th>
<th>PROC-ID</th>
<th>LINE-NO</th>
<th>FILLER</th>
<th>TOTAL-NO-REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X</td>
<td>9(4)</td>
</tr>
</tbody>
</table>
To update the data base, a transaction file is used which is a sequential file. It contains CYBER Record Manager 'Z' type records. There are four kinds of records in this file. The number of characters in a record varies from 5 to 73 depending upon the kind of record.

File name: TRANS

Record name: T-REC

<table>
<thead>
<tr>
<th>Field</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP-CHAR</td>
<td>X</td>
<td>Special character '*' to identify the new form.</td>
</tr>
<tr>
<td>FORM-CODE</td>
<td>99</td>
<td>Form number.</td>
</tr>
<tr>
<td>UPDATE-CODE</td>
<td>X</td>
<td>Update code which identifies the type of transactions.</td>
</tr>
<tr>
<td>CURRENCY-STATUS</td>
<td>X</td>
<td>Currency status.</td>
</tr>
</tbody>
</table>

Record name: FORM-01-REC

<table>
<thead>
<tr>
<th>Field</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS-ID1</td>
<td>XX</td>
<td>System identification code.</td>
</tr>
<tr>
<td>SUB-ID1</td>
<td>XX</td>
<td>Sub-system identification code.</td>
</tr>
<tr>
<td>PROC-ID1</td>
<td>XX</td>
<td>Procedure identification code.</td>
</tr>
<tr>
<td>NAME1</td>
<td>X(22)</td>
<td>Short descriptive term.</td>
</tr>
</tbody>
</table>

Record name: FORM-02-HEADER

<table>
<thead>
<tr>
<th>Field</th>
<th>Picture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILLER</td>
<td>X</td>
<td>Special character '$' to identify the header part of the form 02.</td>
</tr>
<tr>
<td>SYS-ID2</td>
<td>XX</td>
<td>System identification code.</td>
</tr>
<tr>
<td>SYS-NAME2</td>
<td>X(22)</td>
<td>Short descriptive term for the Sub-system.</td>
</tr>
</tbody>
</table>
Record name: FORM-02-TRAILER

LINE-N02  99  Line number of the description

DESCR2  X(50)  One line of detailed text.

A pictorial view of the record is shown below:

T-REC

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP-CHAR</td>
<td>FORM-CODE</td>
<td>UPDATE-CODE</td>
<td>CURRENCY-STATUS</td>
<td></td>
</tr>
<tr>
<td>'X'</td>
<td>99</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
### FORM-01-REC

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYS-ID1</td>
<td>SUB-ID1</td>
<td>PROC-ID1</td>
<td>NAME1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>X(22)</td>
<td></td>
</tr>
</tbody>
</table>

### FORM-02-HEADER

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>3</th>
<th>25</th>
<th>27</th>
<th>49</th>
<th>51</th>
<th>73</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FILLER</td>
<td>SYS-ID2</td>
<td>SYS-NAME2</td>
<td>SUB-ID2</td>
<td>SUB-NAME2</td>
<td>PROC-ID2</td>
<td>PROC-NAME2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'S'</td>
<td>XX</td>
<td>X(22)</td>
<td>XX</td>
<td>X(22)</td>
<td>XX</td>
<td>X(22)</td>
<td></td>
</tr>
</tbody>
</table>

### FORM-02-TRAILER

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>52</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LINE-N02</td>
<td>DESCR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>X(50)</td>
<td></td>
</tr>
</tbody>
</table>
Print-file contains the print-out information which is retrieved from the data base, and is also a sequential file. It contains 'Z' type records. There are two kinds of records in the print file (a) short description (b) detailed description.

File: PRNT.

Record name: PRINT-REC.

<table>
<thead>
<tr>
<th>FIELD</th>
<th>PICTURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILLER</td>
<td>X(8)</td>
<td>Blanks</td>
</tr>
<tr>
<td>P-CURR-STATUS</td>
<td>X</td>
<td>Currency Status</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(11)</td>
<td>Blanks</td>
</tr>
<tr>
<td>P-SYS-ID</td>
<td>XX</td>
<td>System identification code.</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(8)</td>
<td>Blanks</td>
</tr>
<tr>
<td>P-SUB-ID</td>
<td>XX</td>
<td>Sub-System identification code.</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(9)</td>
<td>Blanks</td>
</tr>
<tr>
<td>P-PROC-ID</td>
<td>XX</td>
<td>Procedure identification code.</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(6)</td>
<td>Blanks</td>
</tr>
<tr>
<td>P-NAME</td>
<td>X(22)</td>
<td>Short descriptive term.</td>
</tr>
</tbody>
</table>

Record name: PRINT-DESCR-REC

<table>
<thead>
<tr>
<th>FIELD</th>
<th>PICTURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILLER</td>
<td>X(8)</td>
<td>Blanks</td>
</tr>
<tr>
<td>P-DESCR</td>
<td>X(50)</td>
<td>One line of detailed description.</td>
</tr>
<tr>
<td>FILLER</td>
<td>X(8)</td>
<td>Blanks</td>
</tr>
</tbody>
</table>
Pictorial view of the record is given below:-

The format of the query is given below:-

\[ [C/][SYS-ID[.SUB-ID[.PROC-ID]]],{\frac{1}{2}}[P] \]

COBOL notation is used to define the query.

[ ]: This is an optional portion. All of the format inside the bracket can be omitted.

{ }: Only one of the vertically stacked items inside the braces can be used.

Any thing which is not in any bracket is essential.

C: Currency status.

SYS-ID: System identification code.

SUB-ID: Sub-system identification code.

PROC-ID: Procedure identification code.
0, 1, 2 and 3 are level numbers

0: Detailed description for specified key.
1: System level summary of specified key.
2: Sub-system level summary of specified key.
3: Procedure level summary of specified key.

P: Option for remote hard copy while in interactive mode.

3.3 Program Specification

ADM-SYS.1: Local data entry

The logical structure of the local data entry phase is shown below:
The local data entry phase consists of four routines: All routines are independent programs written in the BASIC language. The terminal execute file is used to exit from one program. It will also load and run another program. The cartridge tape which contains all the programs should be placed on the left tape drive.

a) Display title page: This program first clears the display screen and then will display the title page on the screen for two seconds. The background of the page is a user-defined pattern. This program then locates the 'display menu' program on cartridge tape, loads it in the BASIC workspace and executes it. (Fig. 5)

b) Display menu: This program clears the display screen and displays the menu which shows all the user's options. The menu is shown in Fig. [6]. After displaying the menu on the screen, this program waits for the appropriate key to be pressed. Only the keys shown on the 'menu' are honoured, all other keys are disregarded. The selected program is then located on the cartridge tape, loaded in the work space, and run, e.g. if soft key f2 is pushed, the form-01 edit program will be loaded and executed.

c) Form-01 edit program: This program clears the display window #2 and displays the System Directory Short-Form-01 (Fig. 1) on the screen and turns 'on' the forms mode. The program reads the key board and prints only
GRAPHICS
AUDITING DATA ENTRY SYSTEM
MODULE
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Display Menu</td>
<td>52</td>
</tr>
<tr>
<td>22</td>
<td>Short Form 01</td>
<td>64</td>
</tr>
<tr>
<td>23</td>
<td>Long Form 02</td>
<td>67</td>
</tr>
<tr>
<td>64</td>
<td>NULL</td>
<td>68</td>
</tr>
</tbody>
</table>

**Enter** Enter data on right cassette.

**Note:** Place data cassette in right drive to enter data.

**Key:** 

![Inverse-Video](image)  

**Figure [6]**
those characters which are allowed on the form e.g. only 'A', 'D', 'C' can be accepted for update code. Space, 'X', 'A', 'B', 'C' can be accepted for currency status and A-Z, a-z, 0-9 and spaces can be entered in Sys-ID, SUB-ID and PROC-ID and NAME fields. Sub-routine 'Printtno' automatically generates the line numbers on the form. Sub-routine 'Check' checks every line of information for its correctness, e.g. a line with SYS-ID and PROC-ID without having any SUB-ID is incorrect. Sub-routine 'Err' displays the error message on display window #5, and clears the incorrect line and error message with a hit of <CR> key. Sub-routine 'Clrtn' is used to clear one line on the form. The cursor can be moved to any location and any field can be changed. Once the form is filled up or even only partially filled, the data can be saved on the right cartridge tape by pressing the ENTER key on the keyboard. Sub-routine "Entr' is used to pack one line of information in variable A$[28] and record it as a record on the right cartridge tape. Line numbers are not recorded on the cartridge tape. A special character '*' is added at the beginning of the first record. The record layout is given below:

```
0  1  3  5
   '*'  01  UPDATE CODE  CURRENCY STATUS
```
After data is recorded on the right cartridge tape the form is cleared, and the next set of data can be entered. The menu can be displayed again by pressing the 'fl' soft key.

d) Form-02 edit program: This program works exactly as the form-01 edit program, except that it displays the System Directory Long Form-02 (refer to fig. [2]). This program also generates line numbers, which can be changed by overwriting. Data on the forms can be saved on the right cartridge tape by pressing the 'enter' key. There are three kinds of record. The record layouts are given below:

- Form-02 layout:

  | 0 | 2 | 4 | 6 | 28 |
  |------------------|
  | SYS-ID | SUB-ID | PROC-ID | NAME |

- Update Code layout:

  | 0 | 1 | 3 | 4 | 5 |
  |---------|
  | 'x' | 02 | UPDATE CODE | CURRENCY STATUS |

- Currency Status layout:

  | 0 | 1 | 3 | 25 | 27 | 49 | 51 | 73 |
  |------------------|
  | 'š' | SYS-ID | SYS-NAME | SUB-ID | SUB-NAME | PROC-ID | PROC-NAME |
After data is recorded on the right cartridge tape, the form is cleared for entering the next set of data. The menu can be displayed again by pressing the 'fl' soft key.

ADM-SYS.2: Data Transmission

This program logs in the user to the CDC CYBER 170/730 automatically and displays a message on the terminal to indicate that the user is 'logged in'. Then it rewinds the right cartridge tape and copies one file from the right cartridge tape to the CDC CYBER 170/730 and creates a permanent transaction file on the computer disk storage. The program displays messages to indicate that the transmission of data has started and the data is saved in permanent file 'TRANS' on CDC CYBER 170/730 disk storage.

ADM-SYS.3: Data Base Creation

As already described, the data base consists of one indexed sequential file i.e. master file. The first record in this file shows how many records there are in the master file. The primary key value for this record is spaces. At the time of creation of the indexed sequential file, it needs records sorted in ascending order of primary key. Input to the creation of the data base is
the transaction file which is saved on disk storage. The transaction file should be empty at the time of creation of the data base, since the first record on the master file is the last to be written in this initial dummy version. Output of the data base creation program is

'TOTAL NUMBER OF RECORDS IN THE FILE IS ø'.

**ADM-SYS.4: Data Base Update**

Logical layout of the update program is shown below:

```
+-----------------+
| UPDATE-CODE     |
+-----------------+
    |                |
    | SP-PROC1       |
    |                |
    | ADD-PROC       |
|-----------------+-----------------+
    |                |
    | SP-PROC2       |
    | CHANGE-PROC    |
|-----------------+-----------------+
    |                |
    | CREAT-UPDATE-PROC |
    | DELETE-PROC    |
```

Updating of data base consists of three routines.

a) Add records or extend existing information in the data base.

b) Change records in the data base.

c) Delete records from the data base.

Input to the update program is the transaction file. It is already described in the data specifications that the transaction file consists of four kinds of records. The update program detects the type of record by examining the
first character of the input record.

SP-PROC1: If the first character is '*' then the rest of the record is T-REC and contains form code, update code and currency status. These values are saved in working storage for further reference.

T-REC

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>'X'</td>
<td>FORM-CODE</td>
<td>UPDATE-CODE</td>
<td>CURRENCY-STATUS</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>99</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Following is the list of update codes:

<table>
<thead>
<tr>
<th>Update code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Addition of records</td>
</tr>
<tr>
<td>C</td>
<td>Change of records</td>
</tr>
<tr>
<td>D</td>
<td>Deletion of records</td>
</tr>
<tr>
<td>E</td>
<td>Extension of existing information</td>
</tr>
</tbody>
</table>

Following is the list of form codes:

<table>
<thead>
<tr>
<th>Form-Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>System Directory Short form.</td>
</tr>
<tr>
<td>02</td>
<td>System Directory Long form.</td>
</tr>
</tbody>
</table>
Following is the list of currency status values:

<table>
<thead>
<tr>
<th>Currency-Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>¥</td>
<td>current system</td>
</tr>
<tr>
<td>X</td>
<td>obsolete system</td>
</tr>
<tr>
<td>A</td>
<td>prototype A system</td>
</tr>
<tr>
<td>B</td>
<td>prototype B system</td>
</tr>
<tr>
<td>C</td>
<td>prototype C system</td>
</tr>
</tbody>
</table>

SP-PROC2: If the first character is "¥" in the input record then the record is a FORM-02-HEADER record and contains system-ID, system name, sub-system ID sub-system name, procedure - ID and procedure name. This information is saved in the program working storage for further use.

FORM-02-HEADER

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>3</th>
<th>25</th>
<th>27</th>
<th>49</th>
<th>51</th>
<th>73</th>
</tr>
</thead>
<tbody>
<tr>
<td>¥</td>
<td>X</td>
<td>XX</td>
<td>X(22)</td>
<td>XX</td>
<td>X(22)</td>
<td>XX</td>
<td>X(22)</td>
<td>XX</td>
</tr>
</tbody>
</table>

CREATE-UPDATE-PROC: If the first character in the input record is neither '*' nor '¥' then the input record can be either FORM-01-REC record or FORM-02-TRAILER record. FORM-01-REC contains System-ID, Sub-system - ID, Procedure - ID and Short descriptive name. Name is the system's name if sub-system - ID and procedure - ID are blanks, it is sub-system's name if only Procedure - ID is blank and procedure's name if none of the identifying code fields is blank.
FORM-01-REC

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS-ID XX</td>
<td>SUB-ID XX</td>
<td>PROC-ID XX</td>
<td>NAME X(22)</td>
<td></td>
</tr>
</tbody>
</table>

FORM-02-TRAILER record contains line number and one line of detailed description.

FORM-02-TRAILER

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>52</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE-NO 99</td>
<td>DESCRIPTION X(SO)</td>
<td></td>
</tr>
</tbody>
</table>

FORM-01-REC and FORM-02-TRAILER records are identified by examining the form code saved in working storage. If the form code is 01 then the input record is FORM-01-REC else if the form code is 02 then the input record is FORM-02-TRAILER record.

While FORM-01-REC is read from the input transaction file, information is combined from this record and previously saved values from T-REC in the variable SAVED-VALUE in working storage.

A pictorial view of variable SAVED-VALUE is given below:-

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>10</th>
<th>32</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCY-STATUS</td>
<td>SYS-ID</td>
<td>SUB-ID</td>
<td>PROC-ID</td>
<td>LINE-NO</td>
<td>NAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X(22)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Line number is set to zero.
While FORM-02-TRAILER is read from the input transaction file, information is again combined from this record and previously saved value from T-REC and FORM-02-HEADER records in the variable SAVED-VALUE. Pictorial view is given below:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>10</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCY-STATUS</td>
<td>SYS-ID</td>
<td>SUB-ID</td>
<td>PROC-ID</td>
<td>LINE-NO</td>
<td>DESCRIPTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X(50)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Saved value for update code is examined and the appropriate branching takes place.

ADD-PROC: Addition of record takes place when the update code is 'A' or 'E'.

As the data base is a hierarchical data base, a record can be added to it if and only if its parent record exists in the data base, e.g. a sub-system can be added to the data base only if its parent system already exists. A
detailed description of an entity can be extended if and only if that entity exists in the data base. Hence this routine has two parts. One part checks whether the parent record exists in the data base and displays an error message if it does not. The second part writes the record in the data base from variable 'saved value'. As it is already mentioned that the data base consists of an indexed sequential master file, the records on the master file are as shown below:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>10</th>
<th>32</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCY-STATUS</td>
<td>SYS-ID</td>
<td>SUB-ID</td>
<td>PROC-ID</td>
<td>LINE-NO</td>
<td>NAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X(22)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OR

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>10</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCY-STATUS</td>
<td>SYS-ID</td>
<td>SUB-ID</td>
<td>PROC-ID</td>
<td>LINE-NO</td>
<td>DESCRIPTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
<td>99</td>
<td>X(50)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Primary Key

Primary key is 10 characters long. Every time a record is added to the master file, a counter for addition of records is updated.

CHANGE-PROC: Change of records takes place when update code is 'C'. This routine changes the name field and/or
description field of an existing record in the data base. If the record does not exist then an error message is displayed on the terminal. Every time a record field is changed in master file, a counter for changing of records is updated.

DELETE-PROC: Deletion of records from the data base takes place when the update code is 'D'. Once a record is deleted from the data base all its descendent records should be deleted too e.g. if a system record is deleted then all its sub-system and procedure records should be deleted.

Delete routine has three procedures. If a description line record is deleted then no other records will be deleted as it has no descendent records. DELETE-PROC deletes the first record which matches the input transaction record and all the descendent records of the input transaction record are deleted by the GRI-DELETE-PROC, GR2-DELETE-PROC or GR3-DELETE-PROC. If the record does not exist then an error messages in displayed.
GR1-DELETE-PROC: If the input transaction record is a system record then all its description records, its sub-system and their description records, procedures and their description records are deleted by this procedure.

GR2-DELETE-PROC: If the input transaction record is a sub-system record then all its description records and its procedure and their description records were deleted by this procedure.

GR3-DELETE-PROC: If the input transaction record is a procedure record then all its description records are deleted by this procedure.

Every time a record is deleted from the master file, the counter for deletion of records is updated.

A counter for the total number of records in the master file is updated after all the transaction records are read from the transaction file and appropriate action has been taken.

Outputs of the update program

(1) Update report
   a) Number of records deleted.
   b) Number of records added.
   c) Number of records changed.
   d) Total number of records in the file.
(2) Error messages

a) No record in transmitted file
b) Form code is not recognized
c) Cannot add / extend: no parent record exists for 'RECORD'
d) Cannot change: no existing record for 'RECORD'
e) Cannot delete: record does not exist for 'RECORD'

(3) Updated master file on disk

The update program is compiled in batch mode and the compiled object program is saved on the disk storage for interactive use.

ADM-SYS.5: Data base query processing.

Query can be asked in the following manner: 

\[ [C/\{SYS-ID\.SUB-ID\.PROC-ID\}]\{P\} \]

or END

0, 1, 2, and 3 are level numbers. Format of query is explained in the data specifications section.

Query is answered in the following manner: 

a) If currency status is left out then it is assumed blank.

b) If any identifying code is left out then all the short descriptive records of that entity is assumed.
c) Level-No

0       Short description line and detailed description of the given currency status, SYS-ID, SUB-ID and PROC-ID.

1       Systems' short descriptive line.

2       Sub-systems' and their parents' short descriptive line.

3       Procedures' and their parents' short descriptive line.

d) 'P' is an option for hard copy when this program is running interactively. A message is displayed on the terminal saying how many lines are printed on hard copy.

Examples:

Query

  ,1       Display all the systems' short descriptive line whose currency status is blank.

  X/,2     Display all the systems' and their sub-systems short descriptive line whose currency status is X.

  01,1     Display system's short descriptive line whose currency status is blank and identifying code is 01.
X/02,0 Display detailed description of the system whose currency status is X and identifying code is 02.
END Exit from the program.

Logical layout of the data base query processing phase is shown below:

```
  READ-QUERY
    INITIALIZE-PROC
    UNSTRING-PROC
      LEVEL0-PROC
      LEVEL1-PROC
      LEVEL2-PROC
      LEVEL3-PROC
```

a) Initially the program prompts to find out whether the program is running interactively or not. If the program is running in batch mode then the first data card should be blank, otherwise the information on the first card would be lost.
b) READ-QUERY:

If the program is running interactively then the program prompts for input and the user must supply the input, which is read by the program as one string.

c) INITIALIZE-PROC:

After reading the query all the variables are initialized.

d) UNSTRING:

This routine separates the data items from the query string into separate variables for further use, and then branches to the appropriate routine according to the level number. This routine also prints an error message if the query syntax is incorrect.

e) LEVEL0-PROC: This routine prints the short descriptive line for the given key and all the detailed description lines for that entity. Key is currency status, System-ID, Sub-System-ID and Procedure-ID.

f) LEVEL1-PROC: This routine prints the short descriptive line at the system's level.

g) LEVEL2-PROC: This routine prints the short descriptive line at the sub-system's level.

h) LEVEL3-PROC: This routine prints the short descriptive line at the procedure's level.

If the program is running interactively then after displaying 22 lines this program asks the user if he wants to continue or terminate the query. The heading
line is displayed every time a new query is asked or 22 lines of answer for a query is displayed. In batch mode, or with the hard copy option of interactive mode, all queries start from the top of the new page and the query is printed at the top of the page. Heading line is printed on every page.

The database query processing program is compiled in batch mode and the compiled object program is saved on disk for interactive use.

Output:

Messages

a) Query not recognized.
b) Number of records printed 'count'.
c) Record does not exist.

Reports

a) Short descriptive report at system, sub-system and/or procedure level.
b) Detailed descriptive report of any entity.
c) Hard copy of (a) and (b) on demand.

Application program to construct forms: Forms can be constructed directly through the keyboard, but it is very time consuming especially when several forms have to be constructed. Every time a special forms-drawing character has to be printed, then that character set has to be turned 'on' and 'off'.

Hence a utility program is written to construct forms. Alphanumeric characters can be written on the form, after constructing it, by the keyboard, and then it can be saved on a cartridge tape.

This program can draw thick and thin lines. Corner characters are printed by the program. This program can also define several fields which are discussed below:

Field Code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVE</td>
<td>Inverse video field</td>
</tr>
<tr>
<td>HBF</td>
<td>Half bright field</td>
</tr>
<tr>
<td>IHF</td>
<td>Inverse half bright field</td>
</tr>
<tr>
<td>UPF</td>
<td>Unprotected field</td>
</tr>
<tr>
<td>TOF</td>
<td>Transmit-only field.</td>
</tr>
</tbody>
</table>

Line Code

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKL</td>
<td>Thick horizontal line</td>
</tr>
<tr>
<td>TNL</td>
<td>Thin horizontal line</td>
</tr>
<tr>
<td>TKC</td>
<td>Thick vertical line</td>
</tr>
<tr>
<td>TNC</td>
<td>Thin vertical line</td>
</tr>
</tbody>
</table>

Input format

```
{Field code}
{TKL, starting column, ending column, starting row, ending row, steps in row.}
{TNL}
e.g. TKL, 2, 10, 6, 10, 2
```

```
{TKC} starting row, ending row, starting column, ending column, steps in column.
{TNC}
```

END to exit from the program.
3.4. **Sample test cases:**

A few test cases of the update program and query program are given below.

a) **update report from the update program**

<table>
<thead>
<tr>
<th>NUMBER OF RECORDS DELETED</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER OF RECORD CHANGED</td>
<td>3</td>
</tr>
<tr>
<td>NUMBER OF RECORD ADDED</td>
<td>41</td>
</tr>
<tr>
<td>TOTAL NUMBER OF RECORDS IN THE FILE</td>
<td>40</td>
</tr>
<tr>
<td>JOB PERFORMED</td>
<td></td>
</tr>
</tbody>
</table>

b) **Typical error messages from the update program**

```plaintext
*** CAN NOT ADD/EXTEND RECORD ALREADY EXISTS FOR 001 00 ACCOUNT PABLE
*** CAN NOT DELETE RECORD DOES NOT EXISTS FOR 06 00
```

A record can be deleted from the data base by giving only the primary key value. The name or description line is not necessary in this transaction record. In the second error message, the system ID is 06 and the input record is the
short descriptive record because the line number is zero.
The print out record in the error messages is in the same
form as in the master file.

Terminal print out for query program is shown below:

```
FIBIN
ARE YOU ON THE TERMINAL? (TYPE "Y" OR "YES")
? YES
TYPE NEXT QUERY
? +1^P
NUMBER OF RECORDS PRINTED = 2
TYPE NEXT QUERY
? +2^P
NUMBER OF RECORDS PRINTED = 10
TYPE NEXT QUERY
? +3^P
NUMBER OF RECORDS PRINTED = 31
TYPE NEXT QUERY
? X/1^P
NUMBER OF RECORDS PRINTED = 1
TYPE NEXT QUERY
? X/2
CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
X 01 01 STUDENT INFO SYSTEM
X 01 03 RECEIVING
X 01 05 ACCOUNT PAYMENT
TYPE NEXT QUERY
? 01:1
CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
01 STUDENT INFO SYSTEM
TYPE NEXT QUERY
? 01:03:3
CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
01 STUDENT INFO SYSTEM
DESCRIPTION
TYPE NEXT QUERY
? 01:01:3
CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
01 STUDENT INFO SYSTEM
TYPE NEXT QUERY
? 01:03:3
CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
01 03 STUDENT INFO SYSTEM
01 03 ADMISSIONS
01 03 DAC INPUT
01 03 GRADUATE ADMISSION
01 03 PART-TIME ADMISSION
TYPE NEXT QUERY
? END
JOB REPIEVED.
```
Optional hard copy print out examples are shown below:

<table>
<thead>
<tr>
<th>THE QUERY IS</th>
<th>1,6</th>
<th>CURRENCY STATUS</th>
<th>SYSTEM PROCEEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>STUDENT INFO SYSTEM</td>
<td>PAYROLL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>THE QUERY IS</th>
<th>2,8</th>
<th>CURRENCY STATUS</th>
<th>SYSTEM PROCEEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 03 05 07 11</td>
<td>02 01 03 05</td>
<td>STUDENT INFO SYSTEM</td>
<td>ADMISSIONS</td>
<td>PAYROLL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REGISTRATION</td>
<td>EXAMINATIONS</td>
<td>SALARIED PAYROLL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GRADUATION TRANSPORT</td>
<td></td>
<td>HOURLY PAYROLL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYROLL</td>
<td></td>
<td>TEMPORARY PAYROLL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>THE QUERY IS</th>
<th>X+1,6</th>
<th>CURRENCY STATUS</th>
<th>SYSTEM PROCEEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>X 01</td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CURRENCY STATUS SYSTEM</td>
<td>SUB SYSTEM</td>
<td>PROCEDURE</td>
<td>NAME</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>01</td>
<td>STUDENT INFO SYSTEM</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>03</td>
<td>ADMISSION</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>05</td>
<td>GRADUATE ADMISSION</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>04</td>
<td>01</td>
<td>PART-TIME ADMISSION</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>05</td>
<td>01</td>
<td>UNDER-GRADUATE REGISTRATION</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>05</td>
<td>03</td>
<td>LATE REGISTRATION</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>07</td>
<td>05</td>
<td>GRADUATE REGISTRATION EXAMINATIONS</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>07</td>
<td>03</td>
<td>WINTER</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>07</td>
<td>05</td>
<td>SPRING</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>09</td>
<td>01</td>
<td>GRADUATION</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>09</td>
<td>03</td>
<td>SPRING</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>11</td>
<td>01</td>
<td>FALL</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>01</td>
<td>01</td>
<td>TRANSCRIPT</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>01</td>
<td>03</td>
<td>PAYROLL</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>01</td>
<td>SALARIED PAYROLL</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>01</td>
<td>04</td>
<td>EMPLOYEE HIRE</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>01</td>
<td>PAY</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>04</td>
<td>07</td>
<td>YEAR-END</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>01</td>
<td>HOURLY PAYROLL</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>03</td>
<td>TIME REPORT</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>05</td>
<td>PAY</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>07</td>
<td>YEAR-END</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>01</td>
<td>TEMPORARY PAYROLL</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>03</td>
<td>PAY</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>03</td>
<td>YEAR-END</td>
<td></td>
</tr>
</tbody>
</table>
In a query only two characters are allowed for each of System-ID, Sub-System-ID, and Procedure-ID. One character is allowed for the currency status and hard copy option, and one digit is allowed for level number.

If less than the allowed number of characters is given in a query for any data item, then blanks are inserted in the remaining right locations. If more than the allowed number of characters are given for any data item in a query then the excess characters are truncated from the right.

In the above examples there is a query 01,03,3. Here in the level position we have 03. The level 0 is assumed because the first digit in level position is 0, and the 3 is truncated.

The detailed report is displayed on the terminal because anything other than 'P' in the optional hard copy position causes the report to be displayed on the terminal. In this case 3 is in the optional hard copy position in the query. A description heading is displayed to show that the report is a detailed report even though there is no detailed description in the data base for the system 01.

The query 01,01,3 displays only the system record whose ID is 01 because there is no sub-system 01 under system 01, and therefore no procedure under system 01 and sub-system 01.
All other examples work the way they are expected to work.

As has already been mentioned before, the allowed level numbers are 0, 1, 2 and 3. Hence the above error messages are displayed when the first digit of the level number position is greater than 3 or less than zero.
CHAPTER IV
RESULT AND CONCLUSION

The new system is designed and implemented as discussed in the previous chapters. The system is evaluated to ensure that it is operating as intended and that the original objectives are accomplished.

4.1 Evaluation of results

The system can be evaluated in terms of the objectives achieved as well as the performance of the system and supporting software.

The data base is capable of handling a large number of records but to test the update and query processing programs a data base with a small set of test data records is created. The update program is capable of updating any field of the data base records. This program can be run in interactive or in batch mode and thus is capable of bulk update. The query processing program can be run interactively or in batch mode. A simple query is capable of producing several types of records in response. Examples are given in section 3.4 and Appendix B.

Data entry and data transmission modules are local to the HP2647A terminal. The data entry module needs more memory than is available on the terminal; hence the module is split into several programs and an overlay technique is used through the terminal execute file. A menu is displayed
on the terminal which helps the user to select the appropriate form easily. Data entry forms are legible and simple.

All programs are written in structured manner. Meaningful names are given to the data-items and procedures. Comments are inserted to improve the readability of programs.

Performance of the system can be observed by examining the execution time of the programs. For instance:-

<table>
<thead>
<tr>
<th>Programs</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>To process 4 queries through the query processing program</td>
<td>3.54 secs.</td>
</tr>
<tr>
<td>To update 30 records through the update program.</td>
<td>2.95 secs</td>
</tr>
</tbody>
</table>

The above execution time depends on the following factors.

a) How many disk accesses are made during execution of a program? e.g. one query may need to access more records than another.

b) How busy is the system?

c) How many times is the program swapped in and out of the memory?

It is difficult to observe the execution time of the data entry module and data transmission module.

Performance of the data entry module is reasonable because
the user does not have to wait too long while a program is loaded and starts execution, while an edit program edits one line of the form. Performance of the data transmission module depends upon the following:

a) How busy is the remote computer?

b) Length of the file which has to be transmitted.

As the system is not tested in a real environment, it is hard to judge its performance. But it is reasonable for the small data base of 40 records.

4.2 Were the objectives achieved?

The local data entry module is tested and found capable of storing data on a local tape cartridge. The data transmission module transmits data efficiently to the remote CDC Cyber 170/730 computer and successfully saves the transmitted file on computer disk storage. The update program reads the transaction data from the transmitted file and updates the data base thus providing the facility of bulk updating of data base.

Examples in section 3.4 and Appendix B show that the query processing program can be run interactively or in batch mode and the option is provided to generate remote hard copy. This shows that the first objective is achieved.

The second objective is also achieved. The data is entered through filling up the displayed forms. The user need not have to position the cursor manually to the required
location in the form. The cursor automatically moves to the location where the data has to be entered. The forms are cleared automatically after recording the data on a cartridge tape.

The query format is very simple. A single query format is capable of producing several types of reports. Examples are shown in section 3.4 and Appendix B.

The edit programs for the forms first edit the key which is pressed. If this key character is allowed on the forms then that is displayed else ignored. This emphasizes the robustness of the system.

The above discussion shows that all the objectives which are laid out before in Chapter I are achieved fairly well.

4.3 Possible improvements

Even though the objectives are achieved this system is not perfect. There are several areas where improvements can be made.

a) The update program can be partially changed to improve the performance. In the existing program, while a record is added to the data base, the existence of its parent record is checked by reading the parent record from the disk. This takes disk access time which is in the order of milli-seconds. This access time can be reduced
by saving the last record written in the data base, in working storage. Existence of the parent record can be checked against the previously saved record. The disk access should be made only if the existence of the parent record can not be checked by the previously saved record. As it is observed that most of the time the last added record in the data base is a parent record or a twin record. A twin record also contains the key of the parent record. Thus the existence of the parent record can be checked very easily from the saved record. This technique will save a significant amount of disk access time during the addition of records.

b) The query process can be made simpler if the program prompts for entry of every data item in a query. This way, the user need not have to remember the format of a query.

c) Edit programs for data entry forms can be improved. In the existing program, if a line of a form is being edited and the user types some characters which are valid for the form, but not valid for the next line to be entered, then those characters will be entered in the form. This problem can be solved by emptying the input buffer after editing
a line and before accepting data for the next line.

Further improvements could no doubt be identified once this system is merged into the complete administrative system and run in an everyday environment.
APPENDIX A: Program Listings
BASIC
GET "L"
10 REM
20 REM TITLE PAGE
30 REM
40 COMMAND "SUSPEND COMMAND FILE"
50 PRINT CHR$(27);","DI W$1"
60 PRINT CHR$(27);"*df";CHR$(27);"*dc";CHR$(27);"*da";
70 REM
80 REM PRINT USER DEFINE PATTERN.
90 REM
100 PRINT CHR$(27);"*m3b0,85,0,85,0,85,0,85d0,20,719,359e8mZ";
110 PRINT CHR$(27);"*dk30,250esZG"
120 PRINT "A"
130 PRINT "S"
140 PRINT CHR$(27);"*m5mZ";CHR$(27);"*d88,256ezRAPHICS"
150 PRINT CHR$(27);"*d88,176ezUDITING"
160 PRINT CHR$(27);"*d88,96ezYSTEM"
170 PRINT CHR$(27);"*m1a1b361,46,665,145e2amZ";CHR$(27);"*d375,106ez"
180 PRINT CHR$(27);"*m4mZDATA ENTRY"
190 PRINT " MODULE"
200 REM
210 REM PRINT TITLE PAGE FOR TWO SECONDS.
220 REM
230 AS=CHR$(27)&"@"
240 PRINT RPT$(A$2);
250 PRINT CHR$(27);"*da";CHR$(27);"*D";CHR$(27);"*D";CHR$(27);"*mZ";
260 PRINT CHR$(27);"*daz";CHR$(27);"h";CHR$(27);"J";
270 READ F$
280 DATA 3
290 COMMAND "F F "AF$A" L"
300 COMMAND "RESUME COMMAND FILE"
310 END
RUN
GET "L"
5 REM
6 REM DISPLAY MENU
7 REM
10 COMMAND "SUSPEND COMMAND FILE"
20 PRINT CHR$(27);"c DI W"
30 REM
40 REM HOME THE CURSOR, CLEAR THE GRAPHIC AND THE ALPHANUMERIC MEMORY
50 REM
130 PRINT CHR$(27);"da",CHR$(27);"*dD",CHR$(27);"*dD",CHR$(27);"*mZ";
135 PRINT CHR$(27);"dalD",CHR$(27);"h",CHR$(27);"J"
136 REM
137 REM DISPLAY THE MENU
138 REM
140 PRINT CHR$(27);"&dA",CHR$(27);"&dA OPTION TO SELECT SYSTEM-FORMS "
150 PRINT CHR$(27);"&dA",MOVCA(0,72),"PAGE 1"
160 PRINT CHR$(27);"&dA",CHR$(14),RPT$("",79);CHR$(15)
170 PRINT CHR$(27);"&dD",CHR$(34),"f1"&CHR$(34)," TO ",CHR$(34),"fB"
180 PRINT " TO SELECT THE DESIRED OPTION";CHR$(27);"&dA"
190 PRINT MOVCA(5,4);CHR$(27);"&dA f1 ",CHR$(27);"&dA DISPLAY MENU"
200 PRINT MOVCA(5,40);CHR$(27);"&dA f5 ",CHR$(27);"&dA NULL"
210 PRINT MOVCA(9,4);CHR$(27);"&dA f2 ",CHR$(27);"&dA SHORT FORM 01"
220 PRINT MOVCA(9,40);CHR$(27);"&dA f6 ",CHR$(27);"&dA NULL"
230 PRINT MOVCA(13,4);CHR$(27);"&dA f3 ",CHR$(27);"&dA LONG FORM 02"
240 PRINT MOVCA(17,4);CHR$(27);"&dA f7 ",CHR$(27);"&dA NULL"
250 PRINT MOVCA(17,40);CHR$(27);"&dA f8 ",CHR$(27);"&dA EXIT"
260 PRINT MOVCA(22,4);CHR$(27);"&dA ENTER ",CHR$(27);"&dA ENTER DATA ON ";
270 PRINT "RIGHT CASSETTE"
280 PRINT MOVCA(22,40);CHR$(27);"&dA ENTER DATA ON ",CHR$(14),RPT$("",79);CHR$(15)
290 PRINT MOVCA(22,4);CHR$(27);"&dA NULL"
310 PRINT "drive to enter data."
313 REM FILE ON THE TAPE
314 REM
315 REM CHECK FOR THE KEY PULLED AND FIND THE APPROPRIATE
316 File ON TheTape
317 REM
320 GETKBD ON
330 IF GETKBD(X)=0 THEN 320
340 IF X=240 THEN 130
350 IF X=241 THEN FS="5"\ GOTO 370
351 IF X=247 THEN COMMAND "EXIT"
360 GOTO 330
370 GETKBD OFF
380 COMMAND "F F 
390 COMMAND "RESUME COMMAND FILE"
400 END
RUN
GET "L"
10 REM
20 REM SYSTEM DIRECTORY SHORT FORM 01 EDIT PROGRAM
30 REM
40 COMMAND "SUSPEND COMMAND FILE"
50 GETKBD ON
60 REM
70 REM CLEAR DISPLAY WINDOW #2 AND DISPLAY THE FORM
80 READ F
90 DATA 4
100 PRINT CHR$(27);",CD I W$2"
110 PRINT CHR$(27);",C F F",F,"L"
120 PRINT CHR$(27);","CHR$(27);",J"
130 PRINT CHR$(27);",C F L D I"
140 PRINT CHR$(27);",W"
150 F=0
160 CALL Prtno
170 A=CSENA(R,C)
180 IF C=25 OR C=36 OR C=47 THEN PRINT CHR$(9);" "
190 REM
200 REM READ A CHARACTER AND JUMP TO THE APPROPRIATE ROUTINE
210 REM
220 IF GETKBD(X)=0 THEN 220
230 IF X=152 THEN CALL Entr \ GOTO 160
240 IF X=240 THEN 510
250 REM
260 REM GO TO PRINT AND EDIT FIRST LINE OF THE FORM
270 REM
280 IF R=1 THEN GOSUB 590 \ GOTO 160
290 IF X=193 AND X=196 THEN PRINT CHR$(X)\"F=1\" GOTO 220
300 IF X=239 THEN PRINT
310 IF F=1 THEN GOSUB 650
320 REM
330 REM PRINT REST OF THE FORM
340 REM
350 X$=UPC$(CHR$(X))
360 IF X$="A" AND X$="Z" OR X$="O" THEN PRINT CHR$(X);
370 IF X=9 OR X$=" " THEN PRINT X$;
380 B=CSEA(R1,C1)
390 IF X(239 AND C(79 AND C1C22 THEN 170
400 CALL Check(Flag,R)
410 IF Flag=0 THEN 160
420 IF GET$=X)=0 THEN 420
430 REM
440 REM ONLY CURSOR CAN BE MOVED OR DATA CAN BE ENTERED ON THE
450 REM RIGHT TAPE OR IT CAN BE IGNORED WHILE REACHED AT THE END
460 REM OF THE FORM.
470 REM
480 IF (X)=193 AND X(=196 THEN PRINT CHR$(X)\" GOTO 170
490 IF X=240 AND X=247 OR X=152 THEN GOTO 230
500 GOTO 420
510 PRINT CHR$(27);"X";
520 PRINT CHR$(27);"C",CDI W#1"
530 COMMAND "F F 3 L"
550 END
560 REM
570 REM PRINT AND EDIT FIRST LINE OF THE FORM.
580 REM
590 X$=UPC$(CHR$(X))
600 IF C=79 THEN 630
610 IF X$="A" OR X$="C" OR X$="D" OR X$="P" THEN PRINT X$;
620 GOTO 640
630 IF X$=" " OR X$="X" OR X$="A" AND X$="C" THEN PRINT X$;
640 RETURN
650 F=0
660 A=CSEA(R1,C1)
670 IF C1C22 THEN PRINT RPT$(CHR$(9),3)\" GOTO 700
680 IF C1C25 OR C1C30 AND C1C36 OR C1C41 AND C1C47 THEN PRINT RPT$(CHR$(9),2);
690 Z=C1C24 AND C1C28 OR C1C35 AND C1C39 OR C1C46 AND C1C50
700 IF Z THEN PRINT CHR$(9);
710 RETURN
720 REM
730 REM CHECK EACH LINE OF THE FORM AND CALL THE ERR ROUTINE
740 REM
750 SUB Check(Flag,R)
760 IF R=23 THEN Flag=1\" ELSE Flag=0
770 FOR C=25 TO 47 STEP 15
780 PRINT MOVCA(R,C)," ",

790 NEXT C
800 F=0
810 FOR C=29 TO 52 STEP 11
820 PRINT MOVCA(R,C);
830 A$=DSPINS(-2,X)
840 IF (A$="" OR A$="###") AND C=29 THEN CALL Err(R)\ GOTO 980
850 IF (A$="" AND A$="###") AND F=1 THEN CALL Err(R)\ F=0\ GOTO 980
860 IF (A$="" OR A$="###") AND F=0 THEN Col=C-15\ F=1
870 IF (A$="" OR A$="###") THEN PRINT MOVCA(R,C); " ";
880 NEXT C
890 IF F=0 THEN Col=47
900 PRINT MOVCA(R,Col);">>>";
910 F=0
920 FOR C=58 TO 79
930 PRINT MOVCA(R,C);
940 A$=DSPINS(-1,X)
950 IF A$="" OR A$="" THEN PRINT MOVCA(R,C); " "\ ELSE F=1
960 NEXT C
970 IF F=0 THEN CALL Err(R)
980 IF F=0 THEN Flag=0\ PRINT MOVCA(R,22);\ ELSE R1=R+1\ PRINT MOVCA(R1,22),
990 SUBEND
1030 SUB Err(R)
1031 REM DISPLAY THE ERROR MESSAGE IN DISPLAY WINDOW $5.
1032 REM
1040 PRINT CHR$(27);",cDI W5"
1050 PRINT CHR$(27);",dC ERROR DETECTED ",CHR$(27);","d@"
1060 IF GETKB(X)=0 THEN 1060
1070 IF X=239 THEN PRINT CHR$(27);",cCLO W5"\ ELSE GOTO 1060
1080 REM
1090 REM CLEAR ONE LINE OF THE FORM
1100 REM
1110 CALL Clnln(R)
1120 SUBEND
1130 SUB Clnln(R)
1140 PRINT MOVCA(R,22);" ",RPT$("####",2);RPT$(",",24);
1150 SUBEND
1160 SUB Prtno
1170 REM
1180 REM PRINT LINE NUMBER ON THE FORM
1190 REM
1200 A=CSENA(R,C)
1210 IF R=1 THEN 1250
1220 **1=(R-1) DIV 2+1**
1230 PRINT MOVCA(R,22);
1240 IF I=10 THEN PRINT "10"
1250 SUBEND
1260 SUB Ent
1270 REM
1280 REM ENTER DATA FROM FORM TO RIGHT CASSETTE TAPE.
1290 REM
1300 ASSIGN "RIGHT TAPE" TO $2
1310 DIM A$[28]
1320 PRINT CHR$(27>"H");
1330 A$[1]"="
1350 A$[4]=DSPINS(-1,X)
1360 A$[5]=DSPINS(-1,X)
1370 PRINT $2,A$[1,5]
1380 FOR R=5 TO 23 STEP 2
1390 J=1
1400 FOR C=29 TO 51 STEP 11
1410 PRINT MOVCA(R,C);
1420 A$[J,2]=DSPINS(-2,X)
1440 J=J+2
1450 NEXT C
1460 PRINT MOVCA(R,58),
1470 A$[J,22]=DSPINS(-22,X)
1480 PRINT $2,A$[1,22],
1490 NEXT R
1500 PRINT CHR$(27)"h",""
1510 FOR R=5 TO 23
1520 CALL Clrln(R)
1530 NEXT R
1540 PRINT CHR$(27)"h",
1550 SUBEND
RUN
GET "L"
10 REM
20 REM SYSTEM DIRECTORY LONG FORM 02 EDIT PROGRAM
30 REM
40 COMMAND "SUPEPEND COMMAND FILE"
50 GETKBD ON
55 DIM N#(2)
60 REM
70 REM CLEAR THE DISPLAY WINDOW $2 AND DISPLAY THE FORM 02
80 REM
90 PRINT CHR$(27);",cDI W#2"
100 PRINT CHR$(27);"h",CHR$(27);"J";
110 COMMAND "F F 7 L"
112 COMMAND "C F L D I"
113 PRINT CHR$(27);"W";
114 Flag=0\Row=2\R=7
116 REM
118 REM JUMP TO THE APPROPRIATE ROUTINE ACCORDING TO THE RECORD
119 REM TYPE
120 REM
122 A=CSENA(R,C)
125 IF R=1 THEN GOSUB 300
130 IF R=2 AND R<6 THEN GOSUB 500
140 IF R>=6 THEN GOSUB 500
150 REM
152 REM PRINT CURSOR MOVEMENT
154 REM
158 IF X>=193 AND X<=196 THEN PRINT CHR$(X);\Flag=0
160 IF X=152 THEN CALL Enter\ GOTO 114
170 IF X=240 THEN GOTO 122
180 REM
184 REM DISPLAY THE MENU ON DISPLAY WINDOW #1
186 REM
190 PRINT CHR$(27);"X",CHR$(27);",cDI W#1";
200 COMMAND "RESUME COMMAND FILE"
210 COMMAND "F F 3 L"
220 END
300 REM
302 REM PRINT AND EDIT THE FIRST TYPE
304 REM OF RECORD WHICH IS THE FIRST LINE OF THE FORM
306 REM
310 IF GETKBD(X)=0 THEN 410
315 IF (X)=193 AND X(=196) OR X=152 OR (X)=240 AND X(=247) THEN GOTO 470
320 X$=UPC$(CHR$(X))
330 Chk=C(-52 AND X$="A" OR X$="C" OR X$="E")
340 IF Chk THEN PRINT X$; GOTO 410
350 Chk=C(-52 AND C(=79 AND (X$=" " OR X$="X" OR (X$)="A" AND X$="C"))
360 IF Chk THEN PRINT X$;
370 RETURN
390 REM
400 REM PRINT AND EDIT THE SECOND TYPE OF RECORD WHICH IS THE NEXT
402 REM THREE LINE IN THE FORM
404 REM
410 IF GETKBD(X)=0 THEN 520
420 REM
424 REM PRINT ALLOWABLE CHARACTERS
426 REM
428 A=CSENA(R,C)
430 IF R>Row+1 THEN GOTO 710
432 IF X=152 OR (X)=240 AND X(=247) OR (X)=193 AND X(=196) THEN GOTO 740
434 IF X=239 THEN GOTO 590
436 X$=UPC$(CHR$(X))
438 Chk=C(-52 AND X$="A" AND X$="Z") OR X$=" " OR (X$)="0" AND X$="9")
440 IF Chk THEN PRINT CHR$(X);
450 IF C(78 THEN GOTO 520
452 REM
454 REM CHECK THE LINE
456 REM
458 PRINT MOVCA(R,48);
460 FOR 1=1 TO 24
462 A=CSENA(R,C)
464 A$=DSPIN$(-1,X)
466 IF A$=" " THEN PRINT MOVCA(R,C);" ";
468 NEXT 1
470 PRINT MOVCA(R,48);
472 A$=DSPIN$(-2,X)
474 PRINT MOVCA(R,57);
476 B$=DSPIN$(-2,X)
478 Chk=A$=" " OR B$=" "
480 IF Chk THEN PRINT MOVCA(R,48),RPT$=" ",24*(6-R));\ GOTO 710
482 REM
484 REM PRINT THE ARROW
486 REM
488 IF R()>Row+1 THEN PRINT MOVCA(Row,48);\ GOTO 710
490 PRINT CHR$(27);"X";
492 IF R()>3 THEN PRINT MOVCA(Row,28);" ";
760 PRINT MOVCA(R, 28); >>;
761 RC=27:
762 PRINT MOVCA(R, C);
763 RETURN
764 REM
765 REM PRINT AND EDIT THE DESCRIPTION PORTION OF THE FORM
766 REM
767 IF (R=8 AND C=26) THEN PRINT USING "2A"; O5
768 GOTO 830
769 PRINT MOVCA(8, 29);
770 IF GETKBD(X)=0 THEN 830
771 REM
772 REM PRINT ALLOWABLE CHARACTERS
773 REM
774 IF X=239 THEN GOTO 940
775 IF (X=193 AND X=196) OR (X=240 AND X=247) OR X=152 THEN GOTO 1120
776 A=CSENA(R, C)
777 IF R>Ro+1 THEN R=Ro+1 \ GOTO 1030
778 X$=UPC$(CHR$(X))
779 IF R($)=Ro+1 AND C>=27 AND Flag=0 THEN PRINT CHR$(X);
780 IF X$=" " AND C=27 AND R($)=Ro+1 THEN PRINT X$;
781 IF C=70 THEN GOTO 830
782 REM
783 REM CHECK THE LINE
784 REM
785 PRINT MOVCA(R, 29);
786 A$=DSPIN$(1, X)
787 IF A$=" " THEN PRINT RPT$(" ", 50); MOVCA(R, 29); \ GOTO 830
788 FOR I=1 TO 49
789 A$=CSENA(R, C)
790 A$=DSPIN$(1, X)
791 IF A$=" " THEN PRINT MOVCA(R, C); " 
792 NEXT I
793 REM
794 REM PRINT LINE NUMBER
795 REM
796 IF R=23 THEN Ro\Flag=1 \ GOTO 1120
797 IF R=Ro+1 THEN Ro=Ro+1 \ R=R+1
798 IF R=Ro THEN R=R+1
799 IF R(Ro) THEN R=Ro+1
800 PRINT MOVCA(Ro, 26);
801 N$[1:2]=DSPIN$(0, X)
802 N$=VAL(N$[1:2])
803 IF (N MOD 5)=0 THEN N=N+5 \ ELSE N=N+1
804 PRINT MOVCA(R, 26);
805 PRINT USING "2D"; N
806 PRINT MOVCA(R, 29);
807 GOTO 930
808 RETURN
809
REM PRINT RECORDS IN A$ AND ENTER IT IN THE RIGHT TAPE
1144 REM
1146 REM
1150 ASSIGN "RIGHT TAPE" TO $2
1160 DIM A$(73)
1170 PRINT CHR$(27);"H";
1172 REM
1174 REM ENTER FIRST RECORD
1176 REM
1180 A$(1,1)="*
1190 A$(2,2)=DSPIN$(-2,X)
1200 A$(4,1)=DSPIN$(-1,X)
1205 IF A$(4,5)="*" THEN GOTO 1410
1210 A$(5,1)=DSPIN$(-1,X)
1220 PRINT $2,A$(1,5)
1230 REM
1234 REM ENTER SECOND RECORD
1236 REM
1240 A$(1,1)="*
1250 A$(2,2)=DSPIN$(-2,X)
1260 A$(4,2)=DSPIN$(-2,X)
1270 A$(6,2)=DSPIN$(-2,X)
1280 IF A$(2,2)=" " OR A$(2,2)="*" THEN GOTO 1410
1290 A$(8,2)=DSPIN$(-22,X)
1292 A$(50,2)=DSPIN$(-22,X)
1300 A$(52,2)=DSPIN$(-22,X)
1310 IF A$(2,2)=" " OR A$(2,2)="*" THEN GOTO 1410
1320 PRINT $2,A$(1,73)
1322 REM
1324 REM ENTER ALL THE DESCRIPTION LINES
1326 REM
1330 FOR I=1 TO 16
1350 A$(1,2)=DSPIN$(-2,X)
1370 A$(3,50)=DSPIN$(-50,X)
1380 IF A$(3,1)="*" OR A$(3,1)=" " THEN GOTO 1410
1390 PRINT $2,A$(1,52);
1400 NEXT I
1410 REM
1412 REM CLEAR THE FORM
1414 REM
1420 PRINT CHR$(27);"X";
1422 FOR R=3 TO 5
1430 PRINT MOVCA(R,28);" ";
1440 NEXT R
1450 PRINT CHR$(27);"W";
1460 FOR I=1 TO 874
1470 PRINT "*";
1480 NEXT I
1485 PRINT CHR$(27);"h";
1490 SUBEND
RUN
10 REM
20 REM LOG IN TO CDC 730
30 REM
40 DIM A$(60)
50 Ch$=CHR$(27)
60 Rmote$=Ch$&"k1R"
70 Loc$=Ch$&"kOR"
80 PRINT Loc$
90 COMMAND "RE R"
100 ASSIGN "RIGHT TAPE" TO $2
110 COMMAND "ASR"
120 PRINT Rmote$
130 ASSIGN "DATACOMM" TO $1
140 PRINT $1;CHR$(13)
150 GOSUB 220
160 PRINT $1;"702607B,PRAT"
170 GOSUB 220
180 GOSUB 220
190 PRINT $1;"JSGRAD"
200 GOSUB 220
210 GOTO 290
220 GETDCM ON \D$=""
230 IF GETDCM(D$)=0 THEN GOTO 230
240 PRINT D$;
250 IF D$(1)="" AND D$(2)="?" AND D$(3)="" THEN GOTO 230
260 PRINT
270 GETDCM OFF
280 RETURN
290 PRINT "YOU ARE LOGGED IN"
300 REM
310 REM CREAT FILE ON CDC 730 DISK STORAGE
320 REM
330 COMMAND "A D DA"
340 PRINT #1, "A"
350 GOSUB 580
360 PRINT #1, "NEW,TRANS"
370 GOSUB 580
380 PRINT #1, "ASCII"
390 GOSUB 580
400 PRINT #1, "TEXT"
410 PRINT "DATA ENTRY STARTS"
420 READ #1, A$
430 COMMAND "C"
440 PRINT #1, CHR$(3)
450 GOSUB 540
460 PRINT #1, "pack"
470 READ #1, A$
480 PRINT #1, "REWIND,TRANS"
490 READ #1, A$
500 PRINT #1, "REPLACE,TRANS"
510 PRINT "DATA STORED IN FILE TRANS"
520 GOSUB 580
530 GOTO 630
540 READ #1, A$
550 IF A$(1;14)="EXIT TEXT MODE" THEN 540
560 PRINT "DATA ENTRY COMPLETE."
570 RETURN
580 GETDCM ON \D$=" "
590 IF GETDCM(D$)=0 THEN 590
600 IF D$(1)="/" THEN 590
610 GETDCM OFF
620 RETURN
630 END
IDENTIFICATION DIVISION.

* PROGRAM TO CREATE DATABASE

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT TRANSMITTED-FILE ASSIGN TO TRANS
USE "PT.ZM"
SELECT MASTER-FILE ASSIGN TO MASTER

ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS REC-KEY.

DATA DIVISION.

FILE SECTION.

TRANSMITTED-FILE CONTAINS TRANSECTION
RECORDS THERE ARE FOUR TYPES OF
TRANSECTION RECORDS WHICH ARE DESCRIBED

FD TRANSMITTED-FILE
LABEL RECORD IS OMITTED
DATA RECORDS ARE T-REC, FORM-01-REC, FORM-02-HEADER,
FORM-02-TRAILER
RECORD CONTAINS 5 TO 73 CHARACTERS.

01 T-REC.
03 SP-CHAR PIC X.
03 FORM-CODE PIC 99.
03 UPDATE-CODE PIC X.
03 CURRENCY-STATUS PIC X.
01 FORM-01-REC.
03 SYS-ID PIC X(2).
03 SUR-ID PIC X.
03 PRG-ID PIC X.
03 NAME PIC X(22).

COLUMN 1 2 3 4 5 6 7 8
**MASTER-FILE IS THE DATA BASE FILE WHICH**
**CONTAINS THREE TYPES OF RECORDS AS**
**DESCRIBED BELOW**

**FD MASTER-FILE**

**LABEL RECORDS ARE OMITTED.**

```plaintext
01 MASTER-REC.
  02 REC-KEY.
    03 KEY.
      04 KEY2.
        05 CURR-STATUS PIC X.
      06 SYS-ID PIC XX.
      07 SUB-IND PIC XX.
      08 LINE-NO PIC 99.
      09 FILLER PIC X.
  02 NAME-04-DESC.
    03 NAME.
      04 FILLER REDEFINES NAME PIC X.
    05 TOTAL-NO-REC PIC 99.
    06 FILLER PIC X.
  04 NAME-04-DESC.
    03 NAME.
      04 FILLER REDEFINES NAME PIC X.
    05 TOTAL-NO-REC PIC 99.
    06 FILLER PIC X.

**WORKING-Storage SECTION.**
```

**FOLLOWING FLAGS ARE USED IN THE PROGRAM**
**MORE-RESTAINS TO INDICATE END OF**
**TRANSACTION DATA**
**END-DELETE TO INDICATE DELITION OF**
**RECORDS COMPLETED**

<table>
<thead>
<tr>
<th>COLUMN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**REC-SUCCE TO INDICATE SUCCESSFUL TRANSACTION**

```
01 FLAGS.
  03 MORE-DATA-REMAINS PIC XXX VALUE "YES".
  03 END-DELETE PIC XXX VALUE "NO".
  03 REC-SUCCE PIC XXX VALUE "YES".
```

**INFORMATION IS SAVED FROM TRANSACTION**

```
01 SAVED-INFO.
  03 W-FORM-CODE PIC 99.
  03 W-UPDATE-CODE PIC X.
```

**FOLLOWING COUNTERS ARE USED FOR COUNTING**

```
01 REC-DELETED PIC S999 VALUE ZEROS.
01 REC-CHANGED PIC S999 VALUE ZEROS.
01 REC-ADDED PIC S999 VALUE ZEROS.
01 REC-DELETE PIC S999.
```

**PROCEDURE DIVISION.**

```
OPEN-FILES.
```
OPEN INPUT TRANSMITTED-FILE,
OPEN OUTPUT MASTER-FILE,
READ-REC
READ TRANSMITTED-FILE
AT END PERFORM NO-REC,
PERFORM UPDATE-PROC
UNTIL MORE-DATA-REMAINS = "NO".

***************
PRINT THE STATISTICS OF TRANSACTIONS*********
***************

DISPLAY "NUMBER OF RECCHOS DELETED ", REC-DELETED,
DISPLAY "NUMBER OF RECCHOS CHANGED ", REC-CHANGED,
DISPLAY "NUMBER OF RECCHOS ADDED ", REC-ADDED,
MOVE SPACES TO MASTER-PIC,
MOVE ZERO TO TOTAL-NO-REC,
MOVE TOTAL-NO-REC TO REC-TOTAL,
SUBTRACT REC-ADDED FROM REC-TOTAL,
MOVE REC-TOTAL TO TOTAL-NO-REC,
WHITE MASTER-PIC,
DISPLAY "TOTAL NUMBER OF RECORDS IN THE FILE ", REC-TOTAL,
CLOSE TRANSMITTED-FILE, MASTER-FILE,
STOP RUN.

NO-REC,
MOVE "NO" TO MORE-DATA-REMAINS,
DISPLAY "*** NO RECORD IN TRANSMITTED-FILE".

***************
ANALYZE THE INPUT TRANSECTION RECORD AND
PERFORM THE APPROPRIATE PROCEDURE***************

UPDATE-PROC
IF SP-CHAR = "+"
PERFORM SP-PROC1
ELSE IF SP-CHAR = "$" PERFORM SP-PROC2 THRU SP2-PROC-EXIT
ELSE PERFORM CREAT-UPDATE-PROC THRU CREAT-UPDATE-EXIT,
MOVE SPACES TO MASTER-PIC,
READ TRANSMITTED-FILE
AT END MOVE "NO" TO MORE-DATA-REMAINS.
SAVE INPUT INFORMATION WHEN THE TRANSACTION RECORD IS T-REC TYPE.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SP-PROC1.
MOVE FORM-CODE TO W-FORM-CODE.
MOVE UPDATE-CODE TO W-UPDATE-CODE.
MOVE CURRENCY-STATUS TO W-CURREN-STATUS.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SAVE INPUT INFORMATION WHEN THE TRANSACTION RECORD IS FORM-02 HEADER TYPE.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

SP-PROC2.
MOVE SYS-ID2 TO W-SYS-ID.
MOVE SUR-ID2 TO W-SUR-ID.
MOVE PROC-ID2 TO W-PROC-ID.
MOVE ZEROS TO W-LINE-NO.
MOVE W-REC-KEY TO REC-KEY.
IF PROC-NAME2 NOT = SPACES
    MOVE PROC-NAME2 TO W-DESCR
ELSE
    IF SUR-NAME2 NOT = SPACES
        MOVE SUR-NAME2 TO W-DESCR
    ELSE
        MOVE SYS-NAME2 TO W-DESCR
    IF W-DESCR NOT = SPACES

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

ADD SHORT DESCRIPTIVE RECORD TO THE MASTER-FILE IF IT IS NOT ALREADY PRESENT.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

READ MASTER-FILE RECORD
KEY IS REC-KEY
INVALID KEY PERFORM ADD-PROC THRU ADD-PROC-EXIT.

SP2-PROC-EXIT.
EXIT.

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

ASSEMBLE ALL INFORMATION IN WORKING-

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

COLUMN 123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
CREAT-UPDATE-PROC
IF W-FORM-CODE = 01
  MOVE SYS-1D1 TO W-SYS-1D
  MOVE SUB-1D1 TO W-SUB-1D
  MOVE PROG-1D1 TO W-PROG-1D
  MOVE ZEROS TO W-1LNE-NO
  MOVE NAME1 TO W-DESC
ELSE
  IF W-FORM-CODE = 02
    MOVE LINE-NO2 TO W-LINE-NO
  ELSE
    MOVE DESC2 TO W-DESC
    DISPLAY ** ERROR CODE NOT RECOGNIZED **, W-FORM-CODE
    GO TO CREAT-UPDATE-EXIT.
ENDIF

**********************************************
* APPROPRIATE PROCEDURE, IF UPDATE CODE *
* IS NOT RECOGNIZED THEN DISPLAY THE ERROR* *
**********************************************

IF W-UPDATE-CODE = "MN" OR "MD"
PERFORM ADD-PROC THRU ADD-PROC-EXIT
ELSE IF W-UPDATE-CODE = "MC"
PERFORM CHANGE-PROC
ELSE IF W-UPDATE-CODE = "MD"
PERFORM DELETE-PROC THRU DELETE-PROC-EXIT
ELSE
  DISPLAY ** UPDATE CODE IS NOT RECOGNIZED **
  MOVE 'YES' TO REC-SUCC.
ENDIF

CREAT-UPDATE-EXIT.
EXIT.

**********************************************
* FIRST CHECK IF THE PARENT RECORD EXISTS *
* IN THE MASTER-FILE, IF IT EXISTS THEN *
* ADD THE INPUT RECORD IN THE MASTER-FILE *
**********************************************

ADD-PROC.
IF W-LINE-NO NOT = ZERO
  MOVE W-REC-KEY TO REC-KEY
  MOVE ZERO TO LINE-NO
  READ MASTER-FILE RECORD
KEY IS REC·KEY
INVALID KEY 00 TO ADD-ERROR-PROC.
MOVE ZERO TO LINE-N0.
IF W·PROC-ID NOT = SPACE
MOVE SPACES TO KEY3
READ MASTER·FILE
KEY IS REC·KEY
INVALID KEY 00 TO ADD-ERROR-PROC.
IF W·PROC-ID NOT = SPACE
MOVE SPACES TO KEY3
READ MASTER·FILE
KEY IS REC·KEY
INVALID KEY 00 TO ADD-ERROR-PROC.
WRITE MASTER·PIC FROM SAVED·VALUE.
INVALID KEY PERFORM DUP·KEY,
ADD I TO REC·ADDED.
ADD·PROC·EXIT.
EXIT.
ADD·ERROR·PROC:
DISPLAY "***CAN NOT ADD/EXTEND NO PARENT RECORD EXISTS FOR "
+ SAVED·VALUE
GO TO ADD·PROC·EXIT.
***************************************************************
CHANGE THE RECORD IN THE MASTER·FILE
***************************************************************
CHANGE·PROC:
REWRITE MASTER·PIC FROM SAVED·VALUE.
INVALID KEY PERFORM BAD·REC.
ADD I TO REC·CHANGED.
DUP·KEY:
SUBTRACT I FROM REC·ADDED.
DISPLAY "***CAN NOT ADD/EXTEND RECORD ALREADY EXISTS FOR "
- MASTER·PIC.
BAD·REC:
SUBTRACT I FROM REC·CHANGED.
DISPLAY "***CAN NOT CHANGE NO EXISTING RECORD FOR "
- MASTER·PIC.
***************************************************************
DELETPROC

374 DELETE-PROC

375 * DELETE RECORDS FROM THE MASTER-FILE

376 ***------------------------------------------------------------------------***

DELETPROC

378

379 DELETE-PROC

380 * GIVE W-REC-KEY TO REC-KEY

381 * DELETE MASTER-FILE RECORD

382 IF REC-SUC = "YES" AND 1 TO REC-DELETED

383 ELSE GO TO DELETE-PROC-EXIT.

384******************************************************************************

385 IF W-LINE-NO NOT = ZERO GO TO DELETE-PROC-EXIT.

386 IF W-SUB-1D = SPACES PERFORM GR1-DELETE-PROC UNTIL END-DELETE = "YES"

387 ELSE IF W-PROC-10 = SPACES PERFORM GR1-DELETE-PROC UNTIL END-DELETE = "YES"

388 ELSE IF W-LINE-NO = ZEROS PERFORM GR1-DELETE-PROC UNTIL END-DELETE = "YES"

389 MOVE HNOH TO END-DELETE.

390 DELETE-PROC-EXIT.

391 EXIT.

392 INV-KEY

393 MOVE HNOH TO REC-SUC

394 DISPLAY "*** CAN NOT DELETE RECORD DOES NOT EXISTS FOR ***

395 * MASTER-FILE.

396 ******************************************************************************

397 * DELETE ALL THE DESCENDANT RECORD OF THE *

398 * GIVEN SYSTEM

399 ******************************************************************************

400 GR1-DELETE-PROC

401 READ MASTER-FILE NEXT RECORD

COLUMN 1 2 3 4 5 6 7 8 9 10
123456789012345678901234567890123456789012345678901234567890
AT END MOVE SPACES TO W-REC-KEY.
  IF KEY1 = GR1
    DELETE MASTER-FILE RECORD
    AND 1 TO REC-DELETE
  ELSE
    MOVE "NO" TO END-DELETE.

**************************************
* DELETE ALL THE DESCEDANT RECORD OF THE *
* GIVEN SUB-SYSTEM                       *
**************************************

GR2-DELETE-PROC.
  READ MASTER-FILE NEXT RECORD
  AT END MOVE SPACES TO W-REC-KEY.
  IF KEY2 = GR2
    DELETE MASTER-FILE RECORD
    AND 1 TO REC-DELETE
  ELSE
    MOVE "YES" TO END-DELETE.

**************************************
* DELETE ALL THE DESCEDANT RECORD OF THE *
* GIVEN PROCEDURE                        *
**************************************

GR3-DELETE-PROC.
  READ MASTER-FILE NEXT RECORD
  AT END MOVE SPACES TO W-REC-KEY.
  IF KEY3 = GR3
    DELETE MASTER-FILE RECORD
    AND 1 TO REC-DELETE
  ELSE
    MOVE "YES" TO END-DELETE.
IDENTIFICATION DIVISION.

** PROGRAM TO UPDATE DATA FILE **

PROGRAM-ID. UP-DATE.
AUTHOR. PRATIMA MATHUR.

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT TRANSMITTED-FILE ASSIGN TO TRANS
LINE "HAT2M".
SELECT MASTER-FILE ASSIGN TO MASTER
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS REC-KEY.

DATA DIVISION.
FILE SECTION.

** TRANSMITTED FILE CONTAINS TRANSECTION RECORDS WHICH ARE DESCRIBED BELOW **

FD TRANSMITTED-FILE
LABEL RECORD IS OMITTED
DATA RECORDS ARE T-REC, FORM-01-REC, FORM-02-HEADER,
FORM-02-TRAILER.
RECORD CONTAINS 5 TO 73 CHARACTERS.

01 T-REC.
03 CHAMP PIC X.
03 FORM-CODE PIC 99.
03 UPDATE-CODE PIC X.
03 CURRENCY-STATUS PIC X.

01 FORM-01-REC.
03 KYO-101 PIC XX.
03 RNH-101 PIC XX.
03 PHOC-101 PIC XX.
03 NAME1 PIC X(22).

COLUMN
MASTER-FIELD IS THE DATA BASE FILE WHICH CONTAINS THREE TYPES OF RECORDS AS DESCRIBED BELOW.

FD MASTER-FILE LABEL RECORDS ARE OMITTED.

WORKING-STORE SECTIONS.

FOLLOWING FLAGS ARE USED IN THE PROGRAM:
- MORE-DATA-REMAINS: TO INDICATE END OF RECORDS COMPLETED
- TRANSACTION DATA
- END-DELETE TO INDICATE DELETION OF
**CDC COBOL 5.1 • LEVEL 460**  
SOURCE LISTING OF UP-DATE

**SOURCE LISTING OF UP-DATE**  
AOP# 66/CDC/CDCS

01 FLAG,<
  03 MORE-REMAINS PIC XXX VALUE "YES",
  03 FND-DELETE PIC XXX VALUE "NO",
  03 REC-SUCC PIC XXX VALUE "YES",

******************************************************************************

**INFORMATION IS SAVED FROM TRANSACTION**  
**RECORD FOR FURTHER USE**

******************************************************************************

01 SAVE-INFO,
  03 W-FORM-CODE PIC 99.
  03 W-UPDATE-CODE PIC X,

01 SAVE-VALUE,
  02 W-REC-KEY,
    03 GR3,
      04 GR2,
        05 GR1:
    05 W-CURR-STATUS PIC X,
    06 W-SYS-ID PIC XX.
  05 W-PROC-ID PIC XX.
  04 W-PROC-ID PIC XX.
  03 W-LINE-NO PIC 99.
  03 FILLER PIC X.
  02 W-DESCR PIC X(50).

******************************************************************************

**FOLLOWING COUNTERS ARE USED FOR COUNTING**  
**NUMBER OF RECORDS ADDED, CHANGED**
**AND DELETED DURING THE RUN AND TOTAL**
**NUMBER OF RECORDS IN THE MASTER-FILE**

******************************************************************************

01 REC-DELETED
  01 REC-CHANGED
  01 REC-ADDED
  01 REC-TOTAL

PROCEDURE DIVISION.

OPEN FILES.

COLUMN 1 2 3 4 5 6 7 R
1234567890123456789012345678901234567890123456789012345678901234567890
OPEN INPUT TRANSMITTED-FILE.
READ-REC.
READ TRANSMITTED-FILE
AT END PERFORM NO-REC.
PERFORM UPDATE-PROC
UNTIL MORE-DATA-RE mains = "NO".

******************************************************

DISPLAY "NUMBER OF RECORDS DELETED ", REC-DELETED,
DISPLAY "NUMBER OF RECORDS CHANGED ", REC-CHANGED,
DISPLAY "NUMBER OF RECORDS ADDED ", REC-ADDED,
MOVE SPACES TO MASTER-REC.
READ MASTER-FILE RECORD
KEY IS REC-KEY
INVALID KEY DISPLAY "***ERROR IN READ MASTER-FILE".
MOVE TOTAL-NO-REC TO REC-TOTAL.
READ TRANSMITTED-FILE RECORD
KEY IS REC-KEY
INVALID KEY DISPLAY "***ERROR***
DISPLAY "TOTAL NUMBER OF RECORDS IN THE FILE ", REC-TOTAL.
CLOSE TRANSMITTED-FILE, MASTER-FILE.
STOP RUN.

NO-REC.
MOVE "NO" TO MORE-DATA-RE mains.
DISPLAY "*** NO RECORD IN TRANSMITTED-FILE".

******************************************************

UPDATE-PROC.
IF SP-CHAR = "N"
PERFORM SP-PROC1
ELSE IF SP-CHAR = "S"
PERFORM SP-PROC2 THRU SP-PROC-EXIT
ELSE
PERFORM CREAT-UPDATE-PROC THRU CREAT-UPDATE-EXIT.
MOVE SPACES TO MASTER-REC.
READ TRANSMITTED-FILE
AT END MOVE "NO" TO MORE-DATA-RE mains.

COLUMN 1 12345678901234567890123456789012345678901234567890123456789012345678901234567890
SP=PROC1
MOVE FORM-CODE TO W-FORM-CODE.
MOVE UPDATE-CODE TO W-UPDATE-CODE.
MOVE CURRENCY-STATUS TO W-CURRENCY-STATUS.

* SAVE INPUT INFORMATION WHEN THE TRANSACTION RECORD IS FORM-02 HEADER TYPEF

SP=PROC2
MOVE SYS-ID TO W-SYS-ID.
MOVE SUB-ID TO W-SUB-ID.
MOVE PROC-ID TO W-PROC-ID.
MOVE FD-ID TO W-FD-ID.
MOVE FNAME TO W-FNAME.
IF PROC-NAME NOT = SPACES
MOVE PROC-NAME TO W-PROC-NAME.
ELSE IF SUB-NAME NOT = SPACES
MOVE SUB-NAME TO W-SUB-NAME.
ELSE MOVE SYS-NAMF TO W-SYS-NAMF.
IF W-SYS-NAMF NOT = SPACES
READ MASTER-FILE RECORD
KEY IS REC-KEY.
INVALID KEY PERFORM ADD-PROC THRU ADD-PROC-EXIT.

SP2=PROC-EXIT.
EXIT.

* ASSEMBLE ALL INFORMATION IN WORKING-STORAGE.
* STORAGE AND UPDATE THE MASTER-FILE.
CREAT-UPDATE-PROC

IF W-FORM-CODE = '01
MOVE SYS-ID TO W-SYS-ID
MOVE SUR-ID TO W-SUR-ID
MOVE PROC-ID TO W-PROC-ID
MOVE ZEROS TO W-LINE-NO
MOVE NAMEI TO W-NAME
ELSE
IF W-FORM-CODE = '02
MOVE LINE-NO TO W-LINE-NO
MOVE DESCZ TO W-DESC
ELSE
DISPLAY *** FORM CODE NOT RECOGNIZED "", W-FORM-CODE
GO TO CREAT-UPDATE-EXIT.

IF W-LINE-NO NOT = ZERO
DO ADD-PROC THRU ADD-PROC-EXIT
ELSE
DO CHANGE-PROC
ELSE
DO DELETE-PROC THRU DELETE-PROC-EXIT
ELSE
DISPLAY *** UPDATE CODE IS NOT RECOGNIZED "".
DO UPDATE-CODE.
MOVE YES TO REC-SUC.

CREAT-UPDATE-EXIT.
EXIT.

ADD-PROC.

IF W-LINE-NO NOT = ZERO
MOVE W-REC-KEY TO REC-KEY
MOVE ZERO TO LINE-N0
READ MASTER-FILE RECORD
KEY IS REC-KEY
INVALID KEY GO TO ADD-ERROR-PROC.
MOVE ZERO TO LINE-N0.
IF W-PROC-ID NOT = SPACE
MOVE SPACES TO KEY3
MOVE GRI TO KEY2
READ MASTER-FILE
KEY IS REC-KEY
INVALID KEY GO TO ADD-ERROR-PROC.
IF W-SUR-ID NOT = SPACES
MOVE SPACES TO KEY3
MOVE GRI TO KEY2
READ MASTER-FILE
KEY IS REC-KEY
INVALID KEY GO TO ADD-ERROR-PROC.
WRITE MASTER-PIC FROM SAVED-VALUE
INVALID KEY PERFORM DUP-KEY.
AND 1 TO REC-ADDED.
ADD-PROC-EXIT.
EXIT.
ADD-ERROR-PROC.
DISPLAY "### CAN NOT ADD/EXTEND NO PARENT RECORD EXISTS FOR "
+ SAVED-VALUE.
GO TO ADD-PROC-EXIT.

******************************************************
* CHANGE THE RECORD IN THE MASTER-FILE              *
******************************************************

CHANGE-PROC.
REWRITE MASTER-PIC FROM SAVED-VALUE
INVALID KEY PERFORM BAD-REC.
AND 1 TO REC-CHANGED.

DUP-KEY
SUBTRACT 1 FROM REC-ADDED
DISPLAY "### CAN NOT ADD/EXTEND RECORD ALREADY EXISTS FOR "
- MASTER-PIC.

BAD-REC
SUBTRACT 1 FROM REC-CHANGED
DISPLAY "### CAN NOT CHANGE NO EXISTING RECORD FOR "
- MASTER-PIC.
DELETE-PROC
MOVE W-REC-KEY TO REC-KEY.
DELETE 'MASTER-FILE RECORD INVALID RECORD PERFORM INV-KEY.
IF REC-SUCCEED "YES" AND 1 TO REC-DELETED
ELSE GO TO DELETE-PROC-EXIT.

DELETE-PROC-EXIT.

* CHECK THE KIND OF INPUT RECORD E.G.
* SYSTEM SUR SYSTEM ON PROCEDURE RECORD
* AND PERFORM THE APPROPRIATE PROCEDURE
*

IF W-LINE-NO NOT = ZERO
GO TO DELETE-PROC-EXIT.
IF W-SUB-ID = SPACES
PERFORM GR1-DELETE-PROC
UNTIL END-DELETE = "YES"
ELSE IF W-PROC-ID = SPACE
PERFORM GR2-DELETE-PROC
UNTIL END-DELETE = "YES"
ELSE
IF W-LINE-NO = ZEROS
PERFORM GR3-DELETE-PROC
UNTIL END-DELETE = "YES"
MOVE "NO" TO END-DELETE.
DELETE-PROC-EXIT.

INV-KEY.
MOVE "NO" TO REC-SUCCEED
DISPLAY "*** CAN NOT DELETE RECORD DOES NOT EXISTS FOR ***"
MMASTER-PIC.

DELETE ALL THE DESCEDANT RECORD OF THE
* GIVEN SYSTEM
* ** ***
GR1-DELETE-PROC
READ MASTER-FILE NEXT RECORD
    AT END MOVE SPACES TO W-REC-KEY.
        IF KEY1 = GR1
        DELETE MASTER-FILE RECORD
        AND 1 TO REC-DELETED
        ELSE
        MOVE "YES" TO END-DELETE.

GR2-DELETE-PROC
READ MASTER-FILE NEXT RECORD
    AT END MOVE SPACES TO W-REC-KEY.
        IF KEY2 = GR2
        DELETE MASTER-FILE RECORD
        AND 1 TO REC-DELETED
        ELSE
        MOVE "YES" TO END-DELETE.

GR3-DELETE-PROC
READ MASTER-FILE NEXT RECORD
    AT END MOVE SPACES TO W-REC-KEY.
        IF KEY3 = GR3
        DELETE MASTER-FILE RECORD
        AND 1 TO REC-DELETED
        ELSE
        MOVE "YES" TO END-DELETE.
IDENTIFICATION DIVISION.

* "QUERY PROCESSING PROGRAM"

* ***********************

PROGRAM-ID. RETREV.

AUTHOR. PRATIMA MATHUR.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT PRNT-FILE ASSIGN TO PRNT
LIKE PRT = 44.

SELECT MASTER-F FILE ASSIGN TO MASTER
ORGANIZATION IS INDEXED
ACCESS MODE IS DYNAMIC
RECORD KEY IS REC-KEY.

DATA DIVISION.

FILE SECTION.

************************************

* MASTER FILE IS THE DATA BASE FILE
* MASTER-FILE RECORDS ARE DESCRIBED BELOW
* ************************************

FD MASTER-FILE
LABEL RECORDS ARE OMITTED.

01 MASTER-REC.
  02 REC-KEY.
    03 KEY1.
    04 KEY2.
    05 KEY3.
    06 CURR-STATUS PIC X.
    07 SYS-ID PIC XX.
    08 PROC-ID PIC XX.
    09 SUB-ID PIC XX.
    10 LINE NO PIC X9.
    11 NAME PIC X(2).
    12 NAME-ON-DESCR PIC X.
    13 DESC PIC X(2).
    14 FILLER PIC X.

COLUMN 123456789012345678901234567890123456789012345678901234567890
USING FILE IS USED TO PRINT THE OUTPUT
* RECORDS

**FD PRINT-FILE
01 PRINTER-REC.
  02 FILLER PIC X.
  02 P-CURR-STATUS PIC X.
  02 P-SYS-NO PIC X.
  02 FILLER PIC X.
  02 P-SUB-ID PIC X.
  02 FILLER PIC X.
  02 P-PROC-ID PIC X.
  02 FILLER PIC X.
  02 P-NAME PIC X.
01 PRINT-DESCR-REC.
  02 FILLER PIC X.
  02 P-DESCR PIC X.
  02 FILLER PIC X.

WORKING-STORAGE SECTION
01 HEADINGS.
  02 SHORT-HEAD-LINE.
    03 FILLER PIC X VALUE SPACE.
    03 CURR PIC X(16) VALUE "CURRENCY STATUS ".
    03 SYS PIC X(16) VALUE " SYSTEM ".
    03 SUB PIC X(16) VALUE " SUB SYSTEM ".
    03 FILLER PIC X(16) VALUE SPACES.
    03 NAME PIC X(16) VALUE " NAME ".
    03 FILLER PIC X(16) VALUE SPACES.
  02 DESC-HEAD-LINE.
    03 FILLER PIC X(16) VALUE " DESCRIPTION ".
    03 FILLER PIC X(16) VALUE SPACES.

**FOLLOWING FLAGS ARE USED IN THE PROGRAM
END-FLAG REACHED END OF MASTER-FILE
WHILE READING IT SEQUENTIALLY
END OF CURRNCY STATUS
PRINT-LINE IF LINE OF INFORMATION IS
PRINTEN
CONT I WHATHEVER USER WANTS TO SEE THE
OUTPUT INFORMATION AFTER 22 LINES
OF INFORMATION IS DISPLAYED ON THE
SCREEN
MSG1 WHETHER THE PROGRAM IS RUNNING INT-
FECTIVELY OR NOT

COLUMN
01 FLAGS.
02 FMD-FILE PIC XXX VALUE "NO".
02 FN0-SEC PIC XXX VALUE "NO".
02 PRINT-LINE PIC XXX VALUE "YES".
02 MSG PIC X VALUE "N".

01 COUNTERS.
02 RFC-CNT PIC 999 VALUE ZERO.
02 UNPL-CNT PIC 99 VALUE LOW-VALUES.
02 PRINT-CNT PIC 99 VALUE LOW-VALUES.

01 SEPARATORS.
02 RUNW-SEP PIC x.
02 SRTW-SEP PIC x.
02 PROC-SEP PIC x.
02 LEVEL-SEP PIC x.

01 OPTIONS.
02 PRINT-OUT PIC x.
02 W-LEVEL PIC X.

**************************************************************
* SAVED-KEY-VALUE IS USED TO HOLD THE RECORD WHICH HAS TO BE PRINTED *
**************************************************************

01 SAVED-KEY-VALUE.
   02 W-REC-KEY.
      03 GR1.
      04 GR2.
      05 GR3.
   06 W-CURR-STATUS PIC X.
   07 W-MSID PIC 99.
   08 W-PROC-ID PIC X.
   09 W-LINE-ID PIC X.
   10 W-NAME PIC X(22).

**************************************************************
* QUERY-REC IS USED TO HOLD THE QUERY *
**************************************************************

01 QUERY-REC.
   02 W-QUERY PIC X(16), VALUE "THE QUERY IS ".
   03 W-STATUS PIC X(14).

PROCEDURE DIVISION.
OPEN-FILES.
   OPEN I-O MASTER-FILE.
   OPEN OUTPUT PRINT-FILE.

DISPLAY-MSG.
   DISPLAY "ARE YOU ON THE TERMINAL? (TYPE "Y" OR "YES")?".
   ACCEPT MSG.

READ-QUERY.
   IF MSG = "Y".
      DISPLAY "TYPE NEXT QUERY ".
      ACCEPT QUERY.
   END-IF.
   IF QUERY = "END".
      GO TO FINISH.
   PERFORM INITIALIZE-PROC THRU INSTRING-PROC-EXIT.
   GO TO READ-QUERY.
FINISH.
  CLOSE MASTER-FILE, PRINT-FILE.
  STOP RUN.

***************************************************************
  INITIALIZE-PROC.
  MOVE NOO TO DISPL-CNT, PRINT-CNT, REC-CNT.
  MOVE NOO TO END-FILE, END-SEC, PRINT-LINE.
  MOVE NOO TO CON.
  MOVE SPACES TO PRINT-REC, W-STATUS, SAVED-KEY-VALUE, OPTIONS.

***************************************************************
  BREAK THE QUERY IN SEPARATE DATA ITEMS.
  ***************************************************************

UNSTRING-PROC.
  UNSTRING QUERY
    DELIMITED BY ALL "/" OR ALL "," OR ALL "*", INTO W-STATUS, DELIMITER IN CURR-SEP
    W-SYS-ID, DELIMITER IN SYS-SEP
    W-PROC-ID, DELIMITER IN PROC-SEP
    W-LEVEL, DELIMITER IN LEVEL-SEP
    PRINT-OUT.

***************************************************************
  MOVE THE VALUES IN CORRECT VARIABLES BY
  ***************************************************************

IF CURR-SEP = "/", MOVE W-SUB-ID TO PRINT-OUT
  MOVE W-SYS-ID TO W-LEVEL
  MOVE SPACES TO W-SUB-ID, W-SYS-ID
ELSE
  IF SYS-SEP = "/", MOVE W-PROC-ID TO PRINT-OUT
    MOVE W-SUB-ID TO W-LEVEL
    MOVE SPACES TO W-PROC-ID, W-SUB-ID
  ELSE
    IF SUB-SEP = "/", MOVE W-LEVEL TO PRINT-OUT
MOVE W-PROC-ID TO W-LEVEL
MOVE SPACES TO W-PROC-ID.

IF CURR-SEP NOT = "V"
  MOVE W-SUR-ID TO W-PROC-ID
  MOVE W-SYS-ID TO W-SUR-ID
  MOVE W-STATUS TO W-SYS-ID
  MOVE SPACES TO W-CURR-STATUS
ELSE
  MOVE W-STATUS TO W-CURR-STATUS.

IF MSG NOT = "Y"
  MOVE "P" TO PRINT-OUT.
  MOVE KFY TO KEY
  IF W-SYS-ID NOT = SPACE
    START MASTER-FILE
    KEY = KFY
    INVALID KEY GO TO RECORD-ERROR-PROC.

IF W-SYS-ID = SPACE
  START MASTER-FILE
  KEY > KFY
  INVALID KEY GO TO RECORD-ERROR-PROC.
  READ MASTER-FILE NEXT RECORD

** *******************************************************************************
* EXAMINE THE LEVEL NUMBER AND PERFORM THE*
* APPROPRIATE PROCEDURES *
** *******************************************************************************

IF W-LEVEL = "0"
  PERFORM LEVEL0-PROC THRU LEVEL0-EXIT
ELSE
  IF W-LEVEL = "1"
    PERFORM LEVEL1-PROC
  ELSE
    IF W-LEVEL = "2"
      PERFORM LEVEL2-PROC
    ELSE
      IF W-LEVEL = "3"
        PERFORM LEVEL3-PROC
      ELSE
        DISPLAY "QUERY NOT RECOGNIZED", QUERY.
        IF PRINT-OUT = "P" AND MSG = "Y"
          DISPLAY "NUMBER OF RECORDS PRINTED = ", REC-CNT,

    UNSTRING-PROC-EXIT.
  EXIT.

  RECORD-ERROR-PROC:
  DISPLAY "** RECORD DOES NOT EXIST FOR KEY ", GR3,
  GO TO UNSTRING-PROC-EXIT.

COLUMN
123456789123456789123456789123456789123456789123456789123456789123456789
***PROCESS THE DETAILED DESCRIPTION OUTPUT***

LEVEL0-PROC

IF W-SYS-ID = SPACES
   GO TO LEVEL0-EXIT.
   MOVE GR3 TO KEY1.
   MOVE ZERO TO LINE-NO.

***POSITION THE MASTER-FILE IN SHORT***

* DESCRIPTIVE RECORD OF THE GIVEN KEY AND *

* PERFORM THE APPROPRIATE PROCEDURE *

READ MASTER-FILE RECORD
   KEY IS REC-KEY.
   INVALID KEY GO TO LEVEL0-EXIT.
   PERFORM LEVEL0-CHECK-PROC
   UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".

LEVEL0-EXIT.

EXIT.

***OUTPUT SHORT DESCRIPTIVE RECORD AND THE***

* DETAILER DESCRIPTION OF THE GIVEN KEY *

LEVEL0-CHECK-PROC.

IF KEY3 = GR3
   IF LINE-NO = ZERO
      MOVE NAME TO W-NAME
      PERFORM PRINT-SHORT-LINE-PROC
      PERFORM PRINT-DESCR-HEAD-PROC
      ELSE
         PERFORM PRINT-DESCR-PROC
   ELSE
      MOVE "YES" TO END-SEC.
      READ MASTER-FILE NEXT RECORD
      AT END MOVE "YES" TO END-FILE.

***PRINT ALL THE RECORDS OF A GIVEN AT***

* SYSTEMS LEVEL *
LEVEL1-PROC
PERFORM LEVEL1-CHECK-PROC
UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".

LEVEL1-CHECK-PROC
IF W-CURR-STATUS = CURR-STATUS
   IF SUB-ID = SPACE AND PROC-ID = SPACE AND LINE-NO = ZER
      MOVE "YES" TO END-SEC
      PERFORM PRINT-SHORT-LINE-PROC
   ELSE IF W-SYS-ID = SYS-ID
      MOVE "YES" TO END-SEC
      PERFORM PRINT-SHORT-LINE-PROC
   ELSE
      MOVE "YES" TO END-SEC
      FALSE
      NEXT SENTENCE
   ELSE
      MOVE "YES" TO END-SEC
      IF NEXT RECORD
         AT END MOVE "YES" TO END-FILE.

LEVEL2-PROC
PERFORM LEVEL2-PROC
UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".

LEVEL2-PROC
PERFORM LEVEL1-CHECK-PROC
UNTIL PRINT-LINE = "YES" OR END-SEC = "YES" OR END-FILE = "YES" OR CONT = "YES".
   IF PRINT-LINE = "YES" AND CONT = "YES"
      MOVE "NO" TO PRINT-LINE END-SEC
   PERFORM LEVEL2-CHECK-PROC
   UNTIL END-SEC = "YES" OR END-FILE = "YES" OR CONT = "NO"
   MOVE "NO" TO END-SEC. PRINT-LINE.
   IF (W-CURR-STATUS = CURR-STATUS AND W-SYS-ID = SPACE) OR
      COLUMN
         1234567890
         1234567890
         1234567890
GRL = KEY1
IF PROC-ID = SPACES AND LINE-NO = ZERO
IF W-SUB-ID = SPACE
PERFORM PRINT-SHORT-LINE-PROC
ELSE
IF W-SUB-ID = SUB-SEC
MOVE "YES" TO END-SEC
PERFORM PRINT-SHORT-LINE-PROC
ELSE NEXT SENTENCE
ENDIF

ELSE NEXT SENTENCE
ELSE
MOVE "YES" TO END-SEC.
READ MASTER-FILE NEXT RECORD
AT END MOVE "YES" TO END-FILE.

***********
* PRINT ALL THE PARENT RECORD AND THE *
* RECORD OF A GIVEN KEY AT PROCEDURE LEVEL *
***********

LEVEL3-PROC
PERFORM LEVEL3-PROC
UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".

LEVEL3-PROC1
PERFORM LEVEL1-CHECK-PROC
UNTIL PRINT-LINE = "YES" OR END-SEC = "YES" OR END-FILE = "YES" OR CONT = "NO".
IF PRINT-LINE = "YES" AND CONT = "YES"
MOVE "NO" TO PRINT-LINE, END-SEC
PERFORM LEVEL3-PROC2
UNTIL END-SEC = "YES" OR END-FILE = "YES" OR CONT = "NO"
MOVE "NO" TO END-SEC, PRINT-LINE.

LEVEL3-PROC2
PERFORM LEVEL2-CHECK-PROC
UNTIL PRINT-LINE = "YES" OR END-SEC = "YES" OR END-FILE = "YES" OR CONT = "NO".
IF PRINT-LINE = "YES" AND CONT = "YES"
MOVE "NO" TO PRINT-LINE, END-SEC
PERFORM LEVEL3-CHECK-PROC
UNTIL END-SEC = "YES" OR END-FILE = "YES" OR CONT = "NO"
MOVE "NO" TO END-SEC, PRINT-LINE.

LEVEL3-CHECK-PROC.

COLUMN 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
123456789012345678901234567890123456789012345678901234567890

"NO"
IF (W-CURR-STATUS = CURR-STATUS AND (W-SYS-ID = SPACE OR
   W-SYS-ID = SPACE}) OR GR2 = "KEY?"
   IF W-LINE-NO = "LINE"
      IF W-PROC-ID = "SPACE"
        PERFORM PRINT-SHORT-LINE-PROC
      ELSE
        IF W-PROC-ID = "PROC-ID"
          MOVE "YES" TO END-SEC
          PERFORM PRINT-SHORT-LINE-PROC
        ELSE
          NEXT SENTENCE
      END-ELSE
   ELSE
      NEXT SENTENCE
   END-ELSE
ELSE
   MOVE "YES" TO END-SEC.
   READ MASTER-FILE NEXT RECORD
   AT END MOVE "YES" TO END-FILE.

******************************************************************************************

* PRINT THE HEADING ON THE HARD COPY

******************************************************************************************

PRINT-HEAD-PROC
   WRITE PRINT-REC FROM QUERY-REC
   WRITE PRINT-REC FROM SHORT-HEAD-LINE
   WRITE PRINT-REC FROM SHORT-HEAD-LINE
   WRITE PRINT-REC...
   MOVE SPACES TO PRINT-REC.
   MOVE 2 TO PRINT-CNT.

******************************************************************************************

* DISPLAY THE HEADING ON THE SCREEN

******************************************************************************************

DISPLAY-HEAD-PROC
   DISPLAY SHORT-HEAD-LINE
   MOVE SPACES TO PRINT-REC.
   DISPLAY PRINT-REC.
   MOVE 2 TO DISPL-CNT.

******************************************************************************************

* PRINT THE DESCRIPTION HEADING in THE
* HARD COPY OR DISPLAY IT ON THE SCREEN

******************************************************************************************

PRINT-DESCR-HEAD-PROC
   IF PRINT-OUT NOT = "" AND DISPL-CNT = 3
      DISPLAY PRINT-REC
DISPLAY DESCRIPTIVE-HEAD-LINE
AND 1 TO PRINT-CNT.

IF PRINT-OUT = "HP" AND PRINT-CNT = 3
WRITE PRINT-REC
WRITE PRINT-REC FROM DESCRIPTIVE-HEAD-LINE
MOVE SPACES TO PRINT-REC.
AND 1 TO PRINT-CNT.

*************************************************************

PRINT-SHORT-LINE-PROC.
IF PRINT-OUT NOT = "HP" AND DISPL-CNT = ZERO
PERFORM DISPLAY-PROC.
IF PRINT-OUT = "HP" AND PRINT-CNT = ZERO
PERFORM DISPLAY-PROC.

*************************************************************

* ASSEMBLE THE INFORMATION IN PRINT-REC *

*************************************************************

MOVE "YES" TO PRINT-LINE.
MOVE CURR-STATUS TO P-CURR-STATUS.
MOVE SYS-ID TO P-SYS-ID.
MOVE SUR-ID TO P-SUR-ID.
MOVE PROC-ID TO P-PROC-ID.
MOVE NAME TO P-NAME.
IF PRINT-OUT NOT = "HP"
DISPLAY PRINT-REC
AND 1 TO DISPL-CNT
IF DISPL-CNT = 22
MOVE ZER0 TO DISPL-CNT
PERFORM DISPLAY-CONT-PROC.
IF PRINT-OUT = "HP"
WRITE PRINT-REC
AND 1 TO REC-CNT
AND 1 TO PRINT-CNT
IF PRINT-CNT = 4
MOVE ZER0 TO PRINT-CNT.
MOVE SPACES TO PRINT-REC.

*************************************************************

* PRINT ON HARD COPY OR DISPLAY ON SCREEN *

*************************************************************

COLUMN
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
PRINT-DESCR-PROC.
IF PRINT-CNT = 0 AND PRINT-OUT = "P"
PERFORM PRINT-HEAD-PROC.
IF DISPL-CNT = 0 AND PRINT-OUT NOT = "P"
PERFORM DISPL-HEAD-PROC.
IF PRINT-CNT = 2 OR DISPL-CNT = 2

******************************************************************************
******************************************************************************

MOVE CURR-STATUS TO P-CURR-STATUS
MOVE SYS-ID TO P-SYS-ID
MOVE SUB-ID TO P-SUB-ID
MOVE PHOC-ID TO P-PHOC-ID
MOVE W-NAME TO P-W-NAME
IF DISPL-CNT = 2 AND PRINT-OUT NOT = "P"
DISPLAY PRINT-REC.
AND 1 TO DISPL-CNT.
IF PRINT-CNT = 2 AND PRINT-OUT = "P"
WRITE PRINT-REC
AND 1 TO PRINT-CNT.
IF PRINT-CNT = 3 OR DISPL-CNT = 3
MOVE SPACES TO PRINT-REC.
PERFORM PRINT-DESCR-HEAD-PROC.
MOVE DESC TO P-DESC.
IF PRINT-OUT NOT = "P"
DISPLAY PRINT-REC.
AND 1 TO DISPL-CNT.
IF DISPL-CNT = 2
MOVE ZERO TO DISPL-CNT.
PERFORM DISPL-CONT-PROC.
IF PRINT-OUT = "P"
WRITE PRINT-REC
AND 1 TO REC-CNT.
AND 1 TO PRINT-CNT.
IF PRINT-CNT = 5
MOVE ZERO TO DISPL-CNT.
MOVE SPACES TO PRINT-REC.

******************************************************************************
******************************************************************************

DISPL-CONT-PROC.
DISPLAY "DO YOU WANT TO SEE MORE? (TYPE "YES" OR "NO"),
ACCEPT CONT.

COLUMN
10 REM MAIN PROGRAM
20 DIM H$(20),H(20,5),V$(20),V(20,5),F$(20),F(20,5)
30 I=0,J=0,K=0
40 INPUT A$,B,C,D,E,G
50 IF A$="END" THEN 120
60 IF A$="TKL" AND A$="TNL" THEN 80
70 I=I+1\H$(I)=A$\H(I,1)=B\H(I,2)=C\H(I,3)=D\H(I,4)=E\H(I,5)=G\ GOTO 40
80 IF A$="TKC" AND A$="TNC" THEN 100
90 J=J+1\V$(J)=A$\V(J,1)=B\V(J,2)=C\V(J,3)=D\V(J,4)=E\V(J,5)=G\ GOTO 40
100 IF A$="UPF" OR A$="IVF" OR A$="HBF" OR A$="IHF" OR A$="TOF" THEN 110
105 GOTO 40
110 K=K+1\F$(K)=A$\F(K,1)=B\F(K,2)=C\F(K,3)=D\F(K,4)=E\F(K,5)=G\ GOTO 40
120 PRINT CHR$(27),"cDI W$2"\ PRINT CHR$(27),"h";CHR$(27);"J";
130 FOR A=I TO J
135 FOR L=V(A,3) TO V(A,4) STEP V(A,5)
140 CALL VI(V$(A),V(A,1),V(A,2),L)
145 NEXT L
150 NEXT A
152 FOR A=1 TO I
153 FOR L=H(A,3) TO H(A,4) STEP H(A,5)
154 CALL HI(H$(A),H(A,1),H(A,2),L)
155 NEXT L
156 NEXT A
160 FOR A=1 TO I
170 FOR B=1 TO J
175 FOR L=H(A,3) TO H(A,4) STEP H(A,5)
177 FOR M=V(B,3) TO V(B,4) STEP V(B,5)
180 CALL SUP(H$(A),H(A,1),H(A,2),L,V$(B),V(B,1),V(B,2),M)
185 NEXT M
187 NEXT L
190 NEXT B
200 NEXT A
220 FOR A=1 TO K
225 FOR L=F(A,3) TO F(A,4) STEP F(A,5)
230 CALL FIELD(F$(A),F(A,1),F(A,2),L)
235 NEXT L
240 NEXT A
270 END
500 REM DRAW HORIZONTAL LINE
510 SUB H$(F$,S,E,L)
520 IF $="TKL" THEN X$=";
530 IF $="TNL" THEN X$=";"
535 FOR I=S TO E
540 PRINT MOVCAL(I,);CHR$(27);"B";CHR$(14);X$;CHR$(15);
545 NEXT I
550 SUBEND
610 SUB V$(F$,S,E,L)
620 IF $="TKC" THEN X$="•";
630 IF $="TNC" THEN X$=".";
640 FOR I=S TO E
650 PRINT MOVCAL(L,);CHR$(27);"B";CHR$(14);X$;CHR$(15);
660 NEXT I
670 SUBEND
1000 SUB Sup$(H$,S,E,L,V$,P,Q,C)
1005 REM SUPERIMPOSE CORNER CHARACTERS
1010 IF S=C OR E=C OR P=L OR Q=L THEN 1610
1020 A=P
1030 IF S=C AND L=P THEN A=1\ GOTO 1110
1040 IF S=C AND L=Q THEN A=2\ GOTO 1110
1050 IF E=C THEN A=3\ GOTO 1110
1060 IF E=C AND L=P THEN A=4\ GOTO 1110
1070 IF E=C AND L=Q THEN A=5\ GOTO 1110
1080 IF E=C THEN A=6\ GOTO 1110
1090 IF L=P THEN A=7\ GOTO 1110
1100 IF L=Q THEN A=8
1110 IF H$="TKL" AND V$="TKC" THEN B=1\ GOTO 1150
1120 IF H$="TKL" AND V$="TNC" THEN B=2\ GOTO 1150
1130 IF H$="TNL" AND V$="TKC" THEN B=3\ GOTO 1150
1140 IF H$="TNL" AND V$="TNC" THEN B=4\ GOTO 1150
1150 ON B GOSUB 1250,1290,1190,1380,1420,1330,1460,1510,1560
1160 PRINT MOVCA(L,C),X$;
1170 GOTO 1610
1190 ON B GOTO 1210,1220,1230,1220
1195 REM LEFT MIDDLE CORNER
1210 X$="1"\ GOTO 1240
1220 X$="5"\ GOTO 1240
1230 X$="4"\ GOTO 1240
1240 RETURN
1250 ON B GOTO 1260,1260,1260,1270
1255 REM UPPER LEFTMOST CORNER
1260 X$="Q"\ GOTO 1280
1270 X$="R"\ GOTO 1280
1280 RETURN
1290 ON B GOTO 1300,1300,1300,1310
1295 REM LOWER LEFTMOST CORNER
1300 X$="A"\ GOTO 1320
1310 X$="F";
1320 RETURN
ON B GOTO 1340,1350,1360,1350
REM RIGHT MIDDLE CORNER
X$="2" GOTO 1370
X$="6" GOTO 1370
X$=CHR$(34)
RETURN
REM UPPER RIGHTMOST CORNER
ON B GOTO 1390,1390,1390,1400
REM UPPER RIGHTMOST CORNER
X$="W" GOTO 1410
X$="T"
RETURN
ON B GOTO 1430,1430,1430,1440
REM UPPER MOST CORNER
X$="S" GOTO 1450
X$="G"
RETURN
ON B GOTO 1470,1480,1490,1490
REM MIDDLE CORNER
X$="0" GOTO 1605
X$="+" GOTO 1605
X$="/" GOTO 1605
RETURN
ON B GOTO 1520,1530,1540,1540
REM LOWER MIDDLE CORNER
X$="4" GOTO 1550
X$="$"
RETURN
ON B GOTO 1570,1580,1590,1600
REM MIDDLE INTERSECTION
X$="0" GOTO 1605
X$="+$ GOTO 1605
X$="*" GOTO 1605
RETURN
SUB Field(*,S,E,L)
REM DISPLAY FIELD
IF *="UPF" THEN X$=CHR$(27)"\"Y$=CHR$(27)\" GOTO 1770
IF *="IVF" THEN X$=CHR$(27)&"dB"Y$=CHR$(27)&"&d" GOTO 1770
IF *="HBF" THEN X$=CHR$(27)&"dH"Y$=CHR$(27)&"&d" GOTO 1770
IF *="IFBF" THEN X$=CHR$(27)&"dI"Y$=CHR$(27)&"&d" GOTO 1770
PRINT MOVCA(L,S);X$;RPT("\",E-S+1);Y$;
SUBEND
SUB Field(*,S,E,L)
REM DISPLAY FIELD
IF *="UPF" THEN X$=CHR$(27)"\"Y$=CHR$(27)\" GOTO 1770
IF *="IVF" THEN X$=CHR$(27)&"dB"Y$=CHR$(27)&"&d" GOTO 1770
IF *="HBF" THEN X$=CHR$(27)&"dH"Y$=CHR$(27)&"&d" GOTO 1770
IF *="IFBF" THEN X$=CHR$(27)&"dI"Y$=CHR$(27)&"&d" GOTO 1770
PRINT MOVCA(L,S);X$;RPT("\",E-S+1);Y$;
SUBEND
APPENDIX B:  Sample Outputs
Error messages from update program

UPBIN

*** CAN NOT DELETE RECORD DOES NOT EXISTS FOR 04 00
*** CAN NOT CHANGE NO EXISTING RECORD FOR 03 00 ACCOUNTING SYSTEM
*** CAN NOT ADD/EXTEND RECORD ALREADY EXISTS FOR 01 00 STUDENT IN FO SYSTEM
*** CAN NOT ADD/EXTEND NO PARENT RECORD EXISTS FOR 0305 00 TEMPORARY FAYROLL

Update report

NUMBER OF RECORDS DELETED 20
NUMBER OF RECORD CHANGED 1
NUMBER OF RECORD ADDED 3
TOTAL NUMBER OF RECORDS IN THE FILE 50
JOB REPRIEVED.
**Outputs from query processing program.**

```
RTBIN
ARE YOU ON THE TERMINAL? (TYPE 'Y' OR 'YES')
? Y
TYPE NEXT QUERY
? 01.1

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM SUB SYSTEM PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
</tr>
</tbody>
</table>

TYPE NEXT QUERY
? 01.03.2

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM SUB SYSTEM PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
</tr>
<tr>
<td>01 03</td>
<td></td>
<td>ADMISSIONS</td>
</tr>
</tbody>
</table>

TYPE NEXT QUERY
? 01.03.05.3

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM SUB SYSTEM PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 03 05</td>
<td></td>
<td>PART-TIME ADMISSION</td>
</tr>
</tbody>
</table>

TYPE NEXT QUERY
? 01.2

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM SUB SYSTEM PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 03</td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
</tr>
<tr>
<td>01 05</td>
<td></td>
<td>ADMISSIONS</td>
</tr>
<tr>
<td>01 07</td>
<td></td>
<td>REGISTRATION</td>
</tr>
<tr>
<td>01 09</td>
<td></td>
<td>GRADUATION</td>
</tr>
<tr>
<td>01 11</td>
<td></td>
<td>TRANSCRIPT</td>
</tr>
</tbody>
</table>

TYPE NEXT QUERY
? 01.3

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM SUB SYSTEM PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 03 01</td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
</tr>
<tr>
<td>01 03 03</td>
<td></td>
<td>ADMISSIONS</td>
</tr>
<tr>
<td>01 03 05</td>
<td></td>
<td>PART-TIME ADMISSION</td>
</tr>
<tr>
<td>01 05 01</td>
<td></td>
<td>UNDER-GRADUATE REGISTR</td>
</tr>
<tr>
<td>01 05 03</td>
<td></td>
<td>LATE REGISTRATION</td>
</tr>
<tr>
<td>01 05 05</td>
<td></td>
<td>GRADUATE REGISTRATION</td>
</tr>
<tr>
<td>01 07 01</td>
<td></td>
<td>EXAMINATIONS</td>
</tr>
<tr>
<td>01 07 03</td>
<td></td>
<td>WINTER</td>
</tr>
<tr>
<td>01 07 05</td>
<td></td>
<td>SUMMER</td>
</tr>
<tr>
<td>01 07 09</td>
<td></td>
<td>SPRING</td>
</tr>
<tr>
<td>01 09 01</td>
<td></td>
<td>GRADUATION</td>
</tr>
<tr>
<td>01 09 03</td>
<td></td>
<td>FALL</td>
</tr>
<tr>
<td>01 11 03</td>
<td></td>
<td>TRANSCRIPT</td>
</tr>
</tbody>
</table>
```
### Currency Status System Sub System Procedure Name

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM</th>
<th>SUB SYSTEM</th>
<th>PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>PAYROLL</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>01</td>
<td>03</td>
<td>SALARIED PAYROLL</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>02</td>
<td>05</td>
<td>EMPLOYEE HIRE</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>07</td>
<td>01</td>
<td>CHANGE PROCEDURE</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>02</td>
<td>07</td>
<td>T4 YEAR-END</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>03</td>
<td>01</td>
<td>HOURLY PAYROLL</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>03</td>
<td>03</td>
<td>TIME REPORTIN</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>03</td>
<td>05</td>
<td>CHANGE PROCEDURE</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>05</td>
<td>01</td>
<td>PAY</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>05</td>
<td>07</td>
<td>T4 YEAR-END</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>05</td>
<td>03</td>
<td>TEMPORARY PAYROLL</td>
</tr>
</tbody>
</table>

### Description

The scope of this system includes the production of pay cheques and the payroll journal for all salaried and hourly-paid employees, the withholding of income tax, pension, and other deductions from pay, the consequent processing of such deductions on a quarterly and annual basis. This entails the maintenance of appropriate employee records from the point of hiring to the end of the year in which the employee terminates, in which are carried the current and year-to-date pay and deductions data, and all necessary personal information.

### Type Next Query

? 02.3

? 02.0

### Currency Status System Sub System Procedure Name

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM</th>
<th>SUB SYSTEM</th>
<th>PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>PAYROLL</td>
</tr>
</tbody>
</table>

### Description

This is the sub-system of the payroll system which deals with salaried employees.
TYPE NEXT QUERY
? 02.01.01,0

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
02 01 01 EMPLOYEE HIRE

DESCRIPTION

TYPE NEXT QUERY
? 02.01.05,0

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
02 01 05 PAY

DESCRIPTION

THIS PROCEDURE PRODUCES THE MONTHLY SALARY CHEQUES
THE CORRESPONDING PAYROLL JOURNAL, AND SUMMARY
LISTINGS FOR VARIOUS DEDUCTIONS.

TYPE NEXT QUERY
? 02.01.03,0

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
02 01 03 CHANGE PROCEDURE

DESCRIPTION

WHENEVER AN EMPLOYEE CHANGES HIS/HER ADDRESS, OR
OTHER PERSONAL DATA, OR THERE IS A CHANGE IN HIS
OR HER EMPLOYMENT STATUS, OR THERE IS AN
ADJUSTMENT TO PAY, THIS PROCEDURE IS INVOKED.

TYPE NEXT QUERY
? 02.01.07,0

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME
02 01 07 T4 YEAR-END

DESCRIPTION

THIS PROCEDURE PRODUCES THE ANNUAL T4 STATEMENT
OF EARNINGS AND DEDUCTIONS FOR INCOME TAX, PENSION
AND CHARITABLE DONATIONS, AND PROVIDES FOR
DISTRIBUTION TO THE EMPLOYEES AND THE FEDERAL
GOVERNMENT.
TYPE NEXT QUERY
? X/1,3

<table>
<thead>
<tr>
<th>CURRENCY STATUS</th>
<th>SYSTEM</th>
<th>SUBSYSTEM</th>
<th>PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>01</td>
<td>01</td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>PURCHASE REQUEST</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>01</td>
<td>03</td>
<td>PURCHASE ORDER</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>03</td>
<td></td>
<td>RECEIVING</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>03</td>
<td>01</td>
<td>RECEIVING ADVICE</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>05</td>
<td>03</td>
<td>BACK ORDER</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>05</td>
<td>01</td>
<td>ACCOUNT PAYMENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CHEQUES</td>
</tr>
</tbody>
</table>

TYPE NEXT QUERY
? *1,P

NUMBER OF RECORDS PRINTED = 2

TYPE NEXT QUERY
? *2,P

NUMBER OF RECORDS PRINTED = 10

TYPE NEXT QUERY
? *3,P

NUMBER OF RECORDS PRINTED = 31

QUERY NOT RECOGNIZED X/P

TYPE NEXT QUERY
? X/1,P

NUMBER OF RECORDS PRINTED = 9

TYPE NEXT QUERY
? END

JOB REPRINTED.
THE QUERY IS \( \text{1P} \)

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME

01 01
02 01

STUDENT INFO SYSTEM PAYROLL

THE QUERY IS \( \text{2P} \)

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME

01 01 03
01 01 05
01 01 07
01 01 09
01 01 11
02 02 01
02 02 03
02 02 05

STUDENT INFO SYSTEM
ADMISSIONS
REGISTRATION
EXAMINATIONS
GRADUATION
TRANSCRIPT
PAYROLL
SALARIED PAYROLL
HOURLY PAYROLL
TEMPORARY PAYROLL

THE QUERY IS \( \text{3P} \)

CURRENCY STATUS SYSTEM SUB SYSTEM PROCEDURE NAME

X 01 01 01
X 01 01 03
X 01 01 03
X 01 03 01
X 01 05 03
X 01 05 01

STUDENT INFO SYSTEM
PURCHASE
REQUEST
PURCHASE ORDER
RECEIVING
RECEIVING ADVICE
ACCOUNT PAYMENT
CHEQUES
### The Query is 3P

<table>
<thead>
<tr>
<th>Currency Status</th>
<th>System</th>
<th>Sub System</th>
<th>Procedure</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>03</td>
<td>01</td>
<td>Student Info System</td>
<td>ADMISSIONS</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>03</td>
<td>OAC Input</td>
<td>GRADUATE ADMISSION</td>
</tr>
<tr>
<td>01</td>
<td>03</td>
<td>05</td>
<td>Part-Time Admission</td>
<td>REGISTRATION</td>
</tr>
<tr>
<td>01</td>
<td>05</td>
<td>03</td>
<td>Late Registration</td>
<td>UNDER-GRADUATE REGISTRATION</td>
</tr>
<tr>
<td>01</td>
<td>05</td>
<td>03</td>
<td>Graduate Registration</td>
<td>EXAMINATIONS</td>
</tr>
<tr>
<td>01</td>
<td>07</td>
<td>01</td>
<td>Winter</td>
<td>SUMMER</td>
</tr>
<tr>
<td>01</td>
<td>07</td>
<td>03</td>
<td>Spring</td>
<td>GRADUATION</td>
</tr>
<tr>
<td>01</td>
<td>09</td>
<td>01</td>
<td>Fall</td>
<td>STUDENT INFO SYSTEM</td>
</tr>
<tr>
<td>01</td>
<td>09</td>
<td>03</td>
<td>Transcript</td>
<td>PAYROLL</td>
</tr>
<tr>
<td>02</td>
<td>01</td>
<td>03</td>
<td>Salaries</td>
<td>PAYROLL</td>
</tr>
<tr>
<td>02</td>
<td>01</td>
<td>01</td>
<td>Employee Hire</td>
<td>PAY</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>01</td>
<td>Change Procedure</td>
<td>YEAR-END PAY</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>01</td>
<td>TIME REPORT</td>
<td>TEMPORARY PAYROLL</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>03</td>
<td>Change Procedure</td>
<td>YEAR-END PAY</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>03</td>
<td>Temporary Payroll</td>
<td>YEAR-END PAY</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>03</td>
<td>T4 Year-End</td>
<td>PAY</td>
</tr>
<tr>
<td>02</td>
<td>05</td>
<td>03</td>
<td>T4 Year-End</td>
<td>PAY</td>
</tr>
</tbody>
</table>
### SYSTEM DIRECTORY - SHORT FORM: 01

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>SUB-SYSTEM</th>
<th>PROCEDURE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>STUDENT INFO SYSTEM</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td></td>
<td>ADMISSIONS</td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>03</td>
<td>OAC INPUT</td>
</tr>
<tr>
<td>4</td>
<td>01</td>
<td>05</td>
<td>REGISTRATION</td>
</tr>
<tr>
<td>5</td>
<td>01</td>
<td>05</td>
<td>UNDER GRADUATE REGISTRATION</td>
</tr>
<tr>
<td>6</td>
<td>01</td>
<td>05</td>
<td>LATE REGISTRATION</td>
</tr>
<tr>
<td>7</td>
<td>01</td>
<td>05</td>
<td>GRADUATE REGISTRATION</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>01</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- **A** - ADD
- **C** - CHANGE
- **D** - DELETE
- **P** - PURGE

**Currency Status:**

- **B** - CURRENT
- **X** - OBSOLETE
- **A** - PROTOTYPE A
- **B** - PROTOTYPE B
- **C** - PROTOTYPE C
- .......ETC.
<table>
<thead>
<tr>
<th>UPDATE CODE</th>
<th>CURRENCY STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - ADD</td>
<td>b - CURRENT</td>
</tr>
<tr>
<td>C - CHANGE</td>
<td>x - OBSOLETE</td>
</tr>
<tr>
<td>E - EXTEND</td>
<td>a - PROTOTYPE A</td>
</tr>
<tr>
<td></td>
<td>b - PROTOTYPE B</td>
</tr>
<tr>
<td></td>
<td>c - PROTOTYPE C</td>
</tr>
</tbody>
</table>

**DESCRIPTION:**

The scope of this system includes the production of pay cheques and the payroll journal for all salaried and hourly-paid employees, the withholding of income tax, pension, and other deductions from pay, the consequent processing of such deductions on a quarterly and annual basis. This entails the maintenance of appropriate employee records from the point of hiring to the end of the year in which the employee terminates, in which are carried the current and year-to-date pay and deductions data, and all necessary personal information.

---

1. **SYSTEM:** ID: 02 | **NAME:** PAYROLL
2. **SUB-SYSTEM:**
3. **PROCEDURE:**

---

**LEGEND:**

- **A** - ADD
- **C** - CHANGE
- **E** - EXTEND
- **b** - CURRENT
- **x** - OBSOLETE
- **a** - PROTOTYPE A
- **b** - PROTOTYPE B
- **c** - PROTOTYPE C
- **etc.**


