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

MASTER OF SCIENCE (1980)
(Computation)

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

L I S T O F F I G U R E S

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) DISPIN$(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 key board. 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 - SHORT FORM 01 UPDATE CODE CURRENCY STATUS

SYSTEM	SUB-SYSTEM	PROCEDURE	NAME

LEGEND

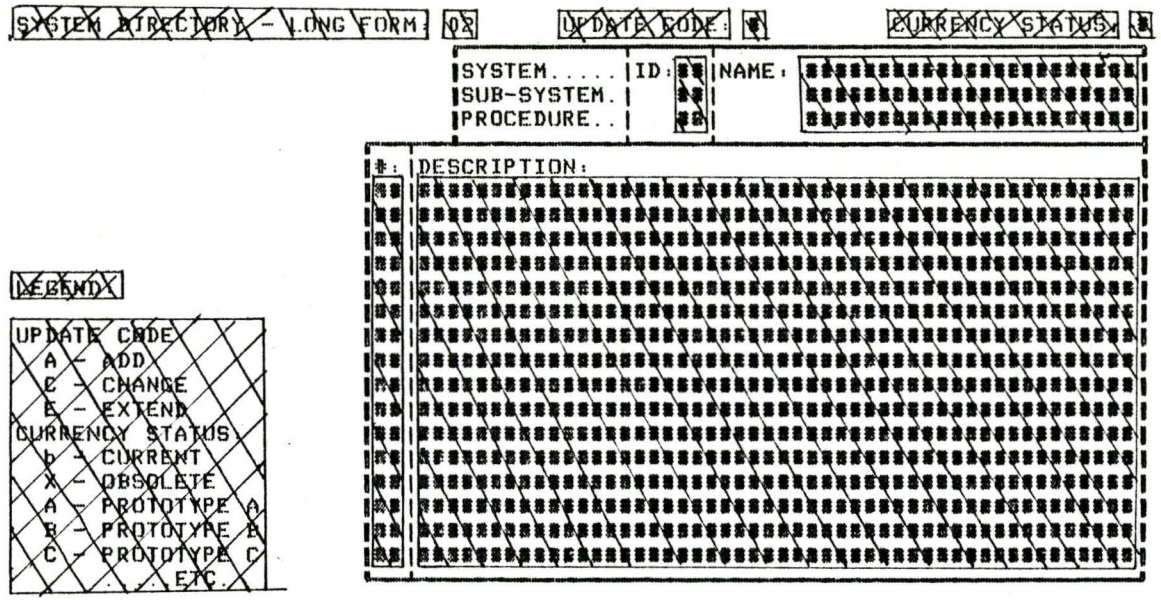
UPDATE CODE:
A - ADD
C - CHANGE
D - DELETE

CURRENCY STATUS
C - CURRENT
X - OBSOLETE
A - PROTOTYPE A
B - PROTOTYPE B
C - PROTOTYPE C
....ETC.

KEY: INVERSE-VIDEO

HALF-BRIGHT AND INVERSE VIDEO

Figure [1]



KEY: INVERSE-VIDEO

HALF-BRIGHT AND INVERSE-VIDEO

Figure [2]

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:

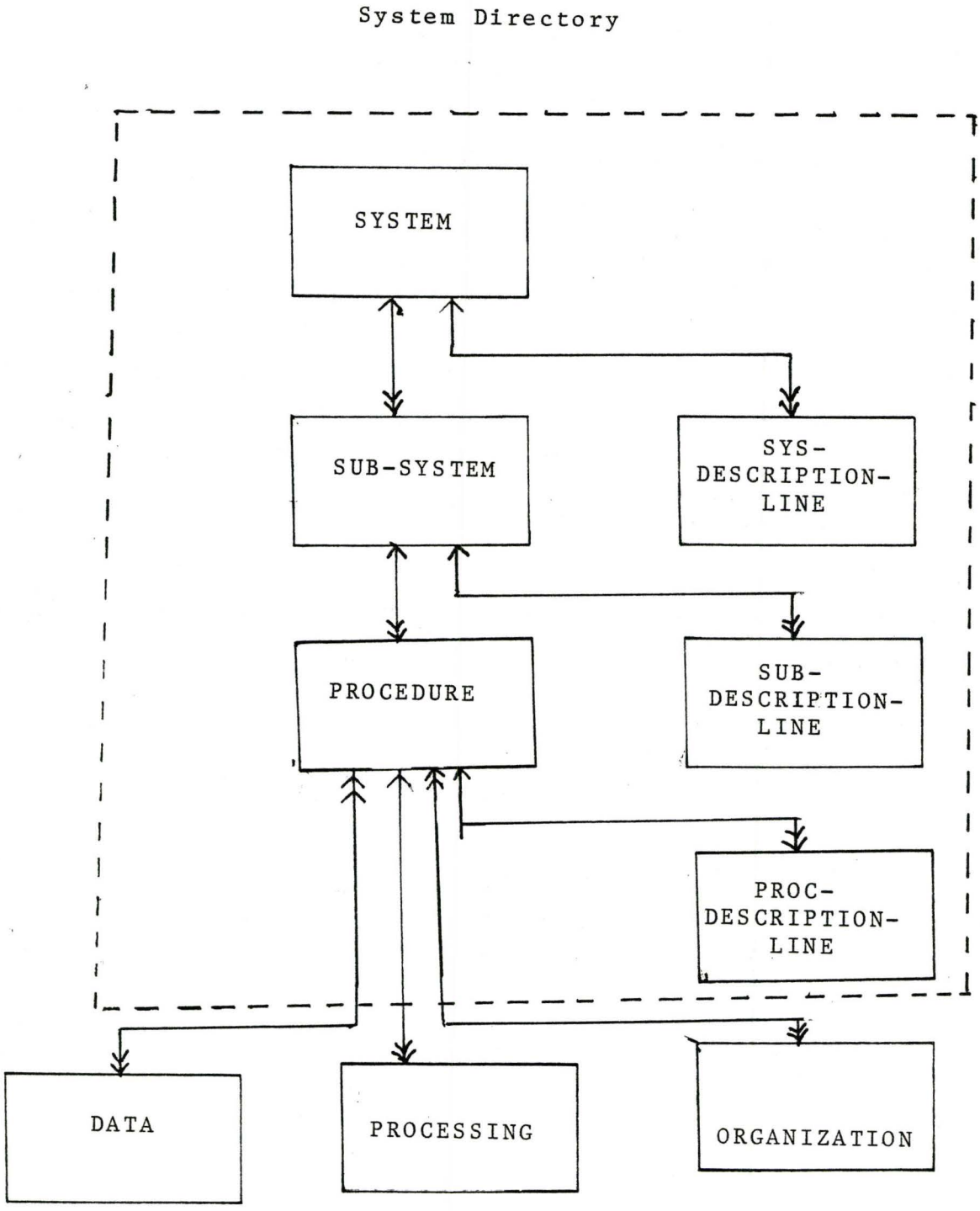


Figure [3]

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 data base. (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:

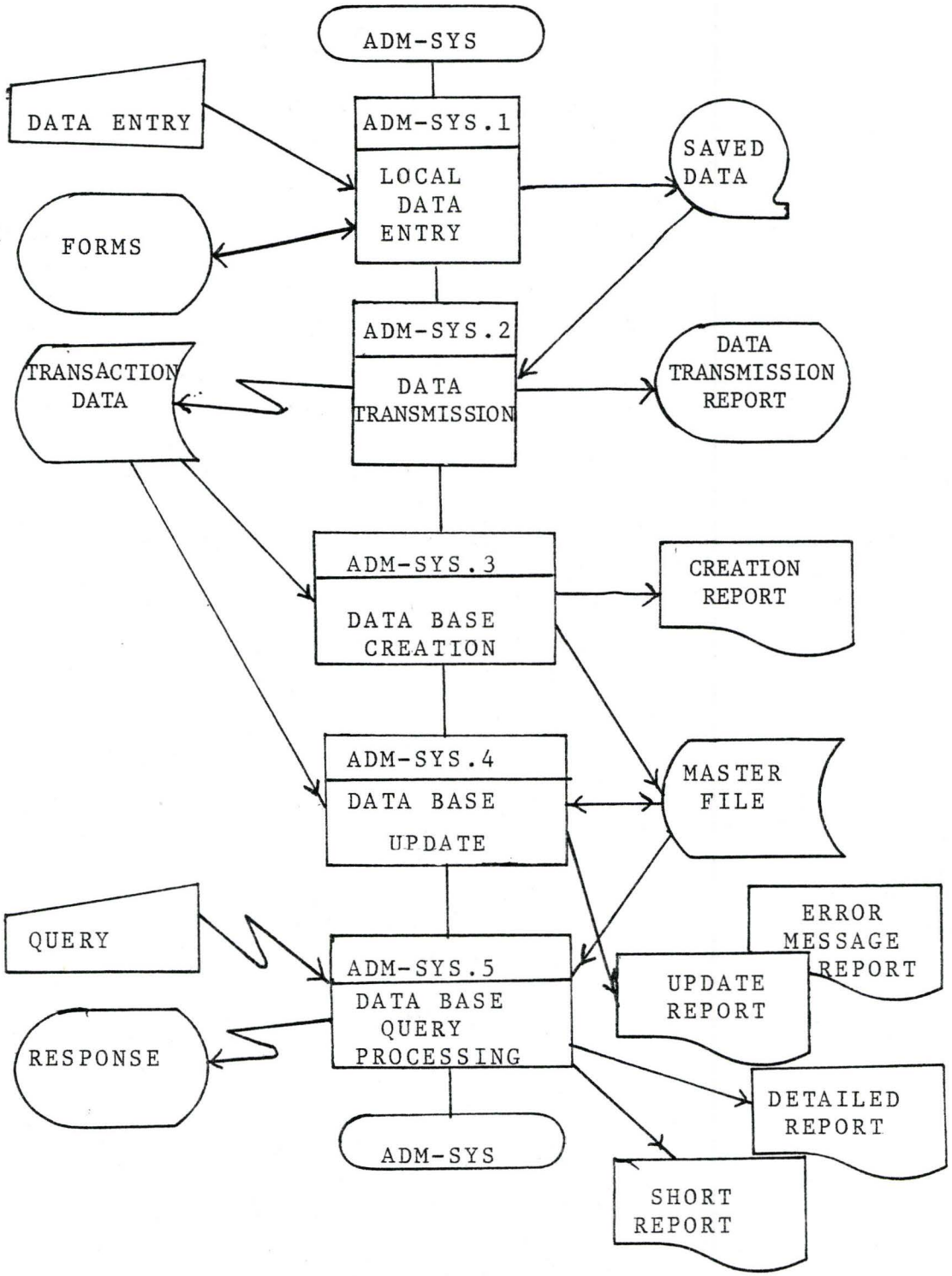


Figure [4]

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

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

**X stands for alphanumeric character.

*9 stands for numeric character.

Field Name	Picture	Description
CURR-STATUS	X	Currency status
SYS-ID	XX	System identification code
SUB-ID	XX	Sub-system identification code
PROC-ID	XX	Procedure identification code
LINE-NO	99	Line-number
NAME	X(22)	Short descriptive term
FILLER	X(28)	Blanks

Field Name	Picture	Description
CURR-STATUS	X	Currency status
SYS-ID	XX	System identification code
SUB-ID	XX	Sub-system identification code
PROC-ID	XX	Procedure identification code
LINE-NO	99	Line number of description
DESCR	X(50)	One line of description

A pictorial layout of the record is shown below:-

0	1	3	5	7	9	10	32 60
CURR-STATUS	SYS-ID	SUB-ID	PROC-ID	LINE-NO	FILLER	NAME	
X	XX	XX	XX	99	X	X(22)	

0	1	3	5	7	9	10 60
CURR-STATUS	SYS-ID	SUB-ID	PROC-ID	LINE-NO	FILLER	DESCR
X	XX	XX	XX	99	X	X(50)

0	1	3	5	7	9	10	14 60
CURR-STATUS	SYS-ID	SUB-ID	PROC-ID	LINE-NO	FILLER	TOTAL-NO -REC	
X	XX	XX	XX	99	X	9(4)	

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

Field	Picture	Description
SP-CHAR	X	Special character '*' to identify the new form.
FORM-CODE	99	Form number.
UPDATE-CODE	X	Update code which identifies the type of transactions.
CURRENCY-STATUS	X	Currency status.

Record name: FORM-01-REC

SYS-ID1	XX	System identification code.
SUB-ID1	XX	Sub-system identification code.
PROC-ID1	XX	Procedure identification code.
NAME1	X(22)	Short descriptive term.

Record name: FORM-02-HEADER

FILLER	X	Special character '$' to identify the header part of the form 02.
SYS-ID2	XX	System identification code.
SYS-NAME2	X(22)	Short descriptive term for the Sub-system.

SUB-ID2	XX	Sub-system identification code
SUB-NAME2	X(22)	Short descriptive term for the Sub-System
PROC-ID2	XX	Procedure identification code
PROC-NAME2	X(22)	Short descriptive term for procedure

Record name: FORM-02-TRAILER

LINE-NO2	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

0	1	3	4	5
SP-CHAR '*' X	FORM-CODE 99	UPDATE-CODE X	CURRENCY-STATUS X	

FORM-01-REC

0	2	4	6	28
SYS-ID1 XX	SUB-ID1 XX	PROC-ID1 XX	NAME1 X(22)	

FORM-02-HEADER

0	1	3	25	27	49	51	73
FILLER '$'	SYS- ID2 XX	SYS- NAME2 X(22)	SUB- ID2 XX	SUB- NAME2 X(22)	PROC- ID2 XX	PROC- NAME2 X(22)	

FROM-02-TRAILER

0	2	52
LINE-NO2 99	DESCR2 X(50)	

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.

FIELD	PICTURE	DESCRIPTION
FILLER	X(8)	Blanks
P-CURR-STATUS	X	Currency Status
FILLER	X(11)	Blanks
P-SYS-ID	XX	System identification code.
FILLER	X(8)	Blanks
P-SUB-ID	XX	Sub-System identification code
FILLER	X(9)	Blanks
P-PROC-ID	XX	Procedure identification code.
FILLER	X(6)	Blanks
P-NAME	X(22)	Short descriptive term.
Record name:	PRINT-DESCR-REC	
FILLER	X(8)	Blanks
P-DESCR	X(50)	One line of detailed description.
FILLER	X(8)	Blanks

Pictorial view of the record is given below:-

0–8	9–20	20–22	22–30	30–32	32–41	41–43	43–49	49–71	
FILLER X(8)	P-CURR-STATUS X	FILLER X(11)	P-SYS-ID XX	FILLER X(8)	P-SUB-ID XX	FILLER X(8)	P-PROC-ID XX	FILLER X(6)	P-NAME X(22)

0–8	8–58	58–66
FILLER X(8)	P-DESCR X(50)	FILLER X(8)

The format of the query is given below:-

$$[C/][SYS\text{-}ID[.SUB\text{-}ID[.PROC\text{-}ID]]],\left\{\begin{matrix}0\\1\\2\\3\end{matrix}\right\}[,P]$$

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.

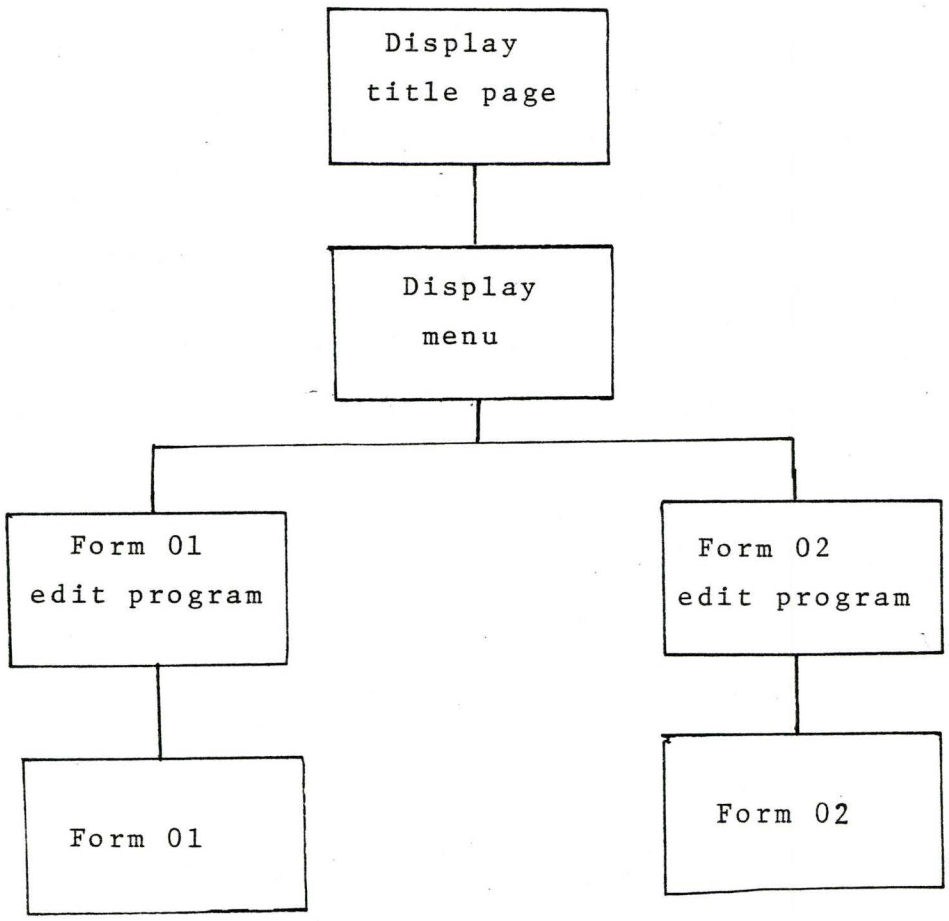
3: Procedure level summary of specified key.

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

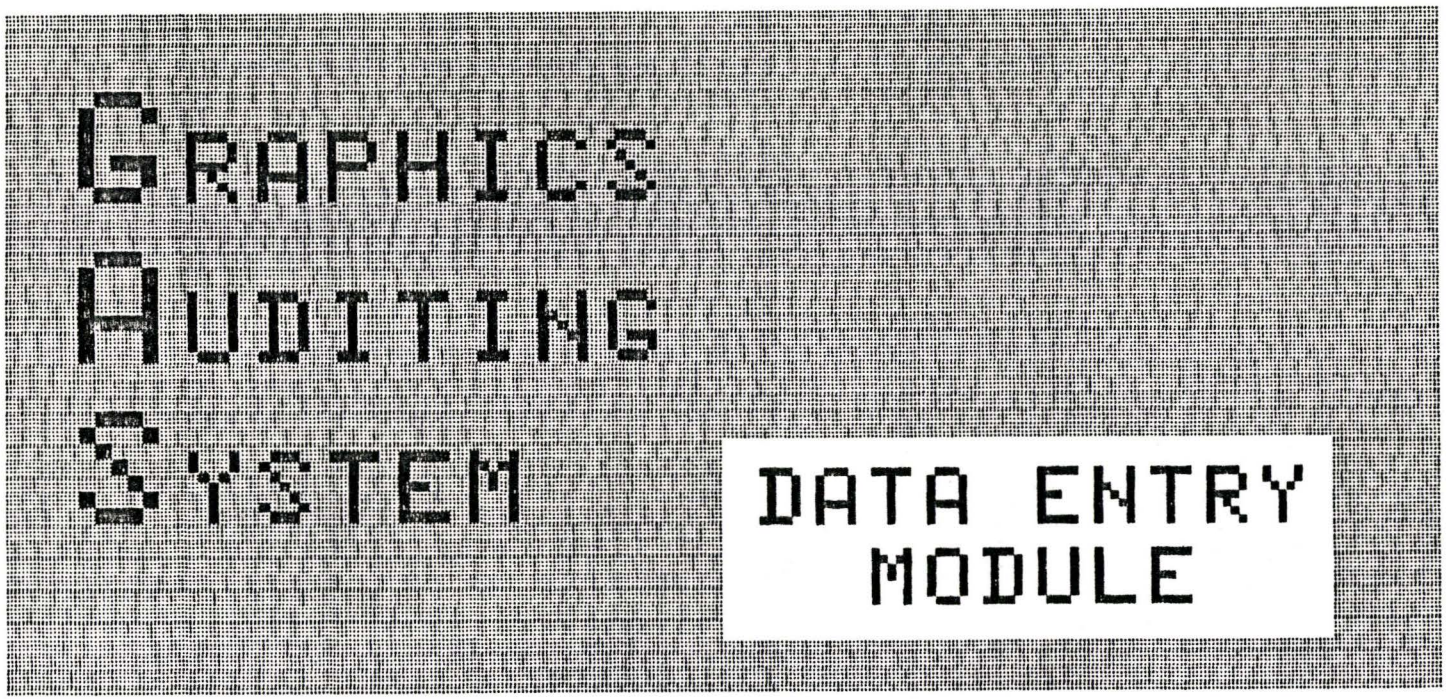


Figure [5]

MENUE: OPTION TO SELECT SYSTEM-FORMS PAGE 1

PRESS "F1" TO "F8 TO SELECT THE DESIRED OPTION

F1	DISPLAY MENU	F5	NULL
F2	SHORT FORM 01	F6	NULL
F3	LONG FORM 02	F7	NULL
F4	NULL	F8	EXIT

ENTER ENTER DATA ON RIGHT CASSETTE

NOTE: Place Data Cassette in right drive to enter data.

KEY: INVERSE-VIDEO

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 'Printℓno' 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 'Clrℓn' 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	4	5
'*'	01	UPDATE CODE	CURRENCY STATUS	

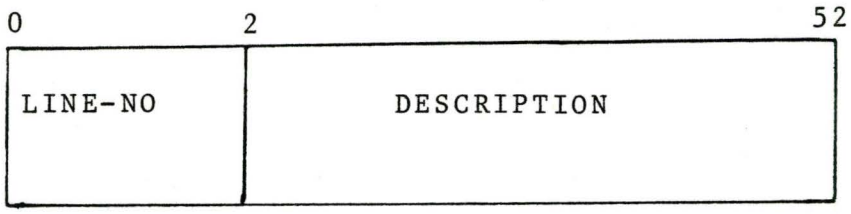
0–2	2–4	4–6	6–28
SYS-ID	SUB-ID	PROC-ID	NAME

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 'f1' 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:-

0–1	1–3	3–4	4–5
'*'	02	UPDATE CODE	CURRENCY STATUS

0–1	1–3	3–25	25–27	27–49	49–51	51–73
'$'	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 'f1' 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

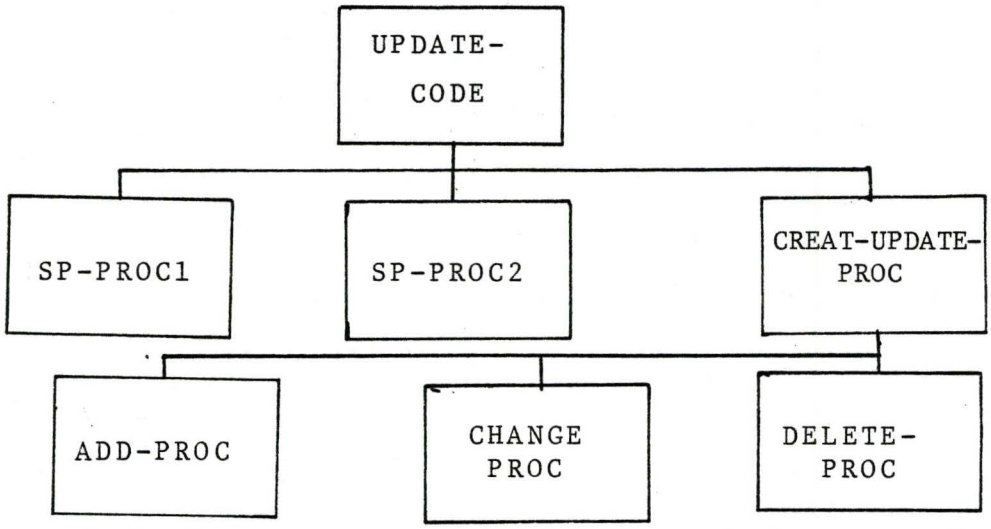
As already described, the data base consists of one indexed sequential file i.e. master file. The first record in this file shows how many records there are in the master file. The primary key value for this record is spaces. At the time of creation of the indexed sequential file, it needs records sorted 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:



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

0	1	3	4 5
'*' X	FORM-CODE 99	UPDATE-CODE X	CURRENCY-STATUS X

Following is the list of update codes:-

Update code	Description
A	Addition of records
C	Change of records
D	Deletion of records
E	Extension of existing information

Following is the list of form codes:-

Form-Code	Description
01	System Directory Short form.
02	System Directory Long form.

Following is the list of currency status values:-

Currency-Status	Description
Ø	current system
X	obsolete system
A	prototype A system
B	prototype B system
C	prototype C system

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

0	1	3	25	27	49	51	73
'$' X	SYS-ID XX	SYS-NAME X(22)	SUB-ID XX	SUB-NAME X(22)	PROC-ID XX	PROC-NAME X(22)	

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

0–2	2–4	4–6	6–28
SYS-ID XX	SUB-ID XX	PROC-ID XX	NAME X(22)

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

FORM-02-TRAILER

0–2	2–52
LINE-NO 99	DESCRIPTION X(50)

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:-

0–1	1–3	3–5	5–7	7–9	9–10	10–32	32–60
CURRENCY-STATUS X	SYS-ID XX	SUB-ID XX	PROC-ID XX	LINE-NO 99		NAME X(22)	

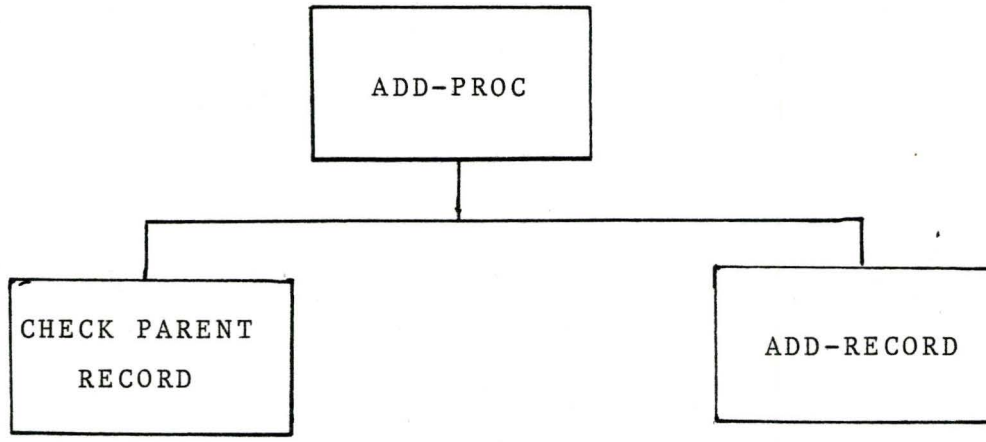
Line number is set to zero.

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:-

0	1	3	5	7	9 10	60
CURRENCY-STATUS X	SYS-ID XX	SUB-ID XX	PROC-ID XX	LINE-NO 99		DESCRIPTION X(50)

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 heirarchical 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:-

0	1	3	5	7	9	10	32	60
CURRENCY-STATUS X	SYS-ID XX	SUB-ID XX	PROC-ID XX	LINE-NO 99		NAME X(22)		

OR

0	1	3	5	7	9	10	60
CURRENCY-STATUS X	SYS-ID XX	SUB-ID XX	PROC-ID XX	LINE-NO 99		DESCRIPTION X(50)	

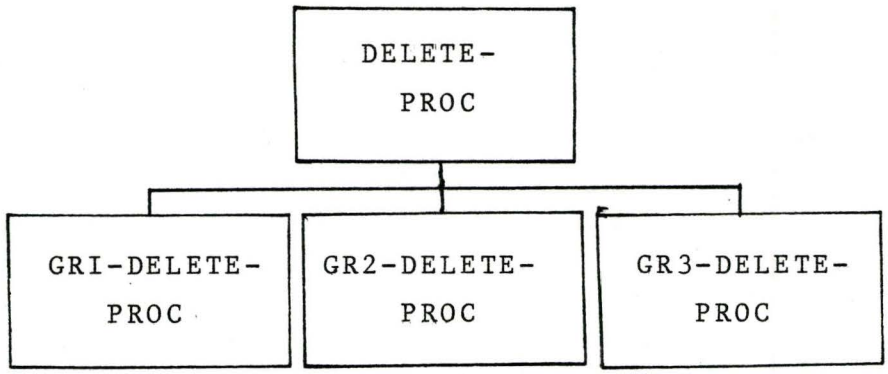
Primary Key

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) Can not add / extend: no parent record exists for 'RECORD'

d) Can not change: no existing record for 'RECORD'

e) Can not 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:-

$$[C/]\ [SYS\text{-}ID[.SUB\text{-}ID].PROC\text{-}ID]]],\left\{\begin{matrix}0\\1\\2\\3\end{matrix}\right\}\ [,P]$$

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:

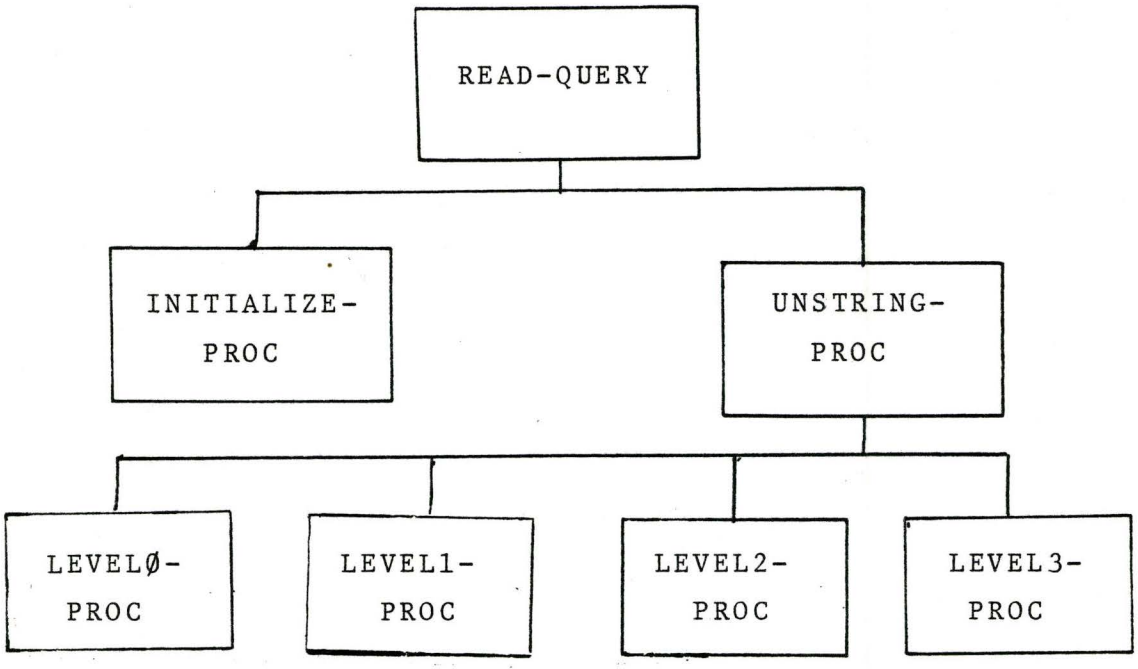
Query

,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:-



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) LEVELØ-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) LEVEL 1-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 date base 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

IVF	Inverse video field
HBF	Half bright field
IHF	Inverse half bright field
UPF	Unprotected field
TOF	Transmit-only field.

Line Code

TKL	Thick horizontal line
TNL	Thin horizontal line
TKC	Thick vertical line
TNC	Thin vertical line

Input format

{Field code / TKL / TNL}, starting column, ending column, starting row, ending row, steps in row.

e.g. TKL, 2, 10, 6, 10, 2

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

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

```
NUMBER OF RECORDS DELETED      1
NUMBER OF RECORD CHANGED       3
NUMBER OF RECORD ADDED        41
TOTAL NUMBER OF RECORDS IN THE FILE      40
JOB REPRIEVED.
```

b) Typical error messages from the update program

```
*** CAN NOT ADD/EXTEND RECORD ALREADY EXISTS FOR   X01    00 ACCOUNT PA
YABLE
*** 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:

```
RTBIN
 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                            STUDENT INFO SYSTEM
          X          01        01                  PURCHASE
          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
YPE NEXT·QUERY
 ? 01.03,3
  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                     01                             STUDENT INFO SYSTEM
                     01        03                   ADMISSIONS
                     01        03        01         OAC INPUT
                     01        03        03         GRADUATE ADMISSION
                     01        03        05         PART-TIME ADMISSION
 TYPE NEXT QUERY
? END
 JOB REPRIEVED.
```

Optional hard copy print out examples are shown below:

```
THE QUERY IS    ,1,P
CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME
                  01                              STUDENT INFO SYSTEM
                  02                              PAYROLL
```

```
THE QUERY IS    ,2,P
CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME
                  01
                  01        03                    STUDENT INFO SYSTEM
                  01        05                    ADMISSIONS
                  01        07                    REGISTRATION
                  01        09                    EXAMINATIONS
                  01        11                    GRADUATION
                  02                              TRANSCRIPT
                  02        01                    PAYROLL
                  02        03                    SALARIED PAYROLL
                  02        05                    HOURLY PAYROLL
                                                  TEMPORARY PAYROLL
```

```
THE QUERY IS    X/,1,P
CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME
       X          01                              STUDENT INFO SYSTEM
```

THE QUERY IS ,3,P

CURRENCY STATUS	SYSTEM	SUB SYSTEM	PROCEDURE	NAME
	01			STUDENT INFO SYSTEM
	01	03		ADMISSIONS
	01	03	01	OAC INPUT
	01	03	03	GRADUATE ADMISSION
	01	03	05	PART-TIME ADMISSION
	01	05		REGISTRATION
	01	05	01	UNDER-GRADUATE REGISTR
	01	05	03	LATE REGISTRATION
	01	05	05	GRADUATE REGISTRATION
	01	07		EXAMINATIONS
	01	07	01	WINTER
	01	07	03	SUMMER
	01	07	05	SPRING
	01	09		GRADUATION
	01	09	01	SPRING
	01	09	03	FALL
	01	11		TRANSCRIPT
	02			PAYROLL
	02	01		SALARIED PAYROLL
	02	01	01	EMPLOYEE HIRE
	02	01	03	CHANGE PROCEDURE
	02	01	05	PAY
	02	01	07	T4 YEAR-END
	02	03		HOURLY PAYROLL
	02	03	01	TIME REPORTIN
	02	03	03	CHANGE PROCEDURE
	02	03	05	PAY
	02	03	07	T4 YEAR-END
	02	05		TEMPORARY PAYROLL
	02	05	01	PAY
	02	05	03	T4 YEAR-END

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.

```
 ARE YOU ON THE TERMINAL? (TYPE "Y" OR "YES")
? YES
 TYPE NEXT QUERY
? ,4
 QUERY NOT RECOGNIZED ,4
 TYPE NEXT QUERY
      ,? 01.03,7,P
 QUERY NOT RECOGNIZED 01.03,7,P
 NUMBER OF RECORDS PRINTED =      0
 TYPE NEXT QUERY
? END
 JOB REPRIEVED.
```

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:-

Programs	Execution time
To process 4 queries through the query processing program	3.54 secs.
To update 30 records through the update program.	2.95 secs

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

```
**********************************
*  PROGRAM TO DISPLAY TITLE PAGE  *
**********************************

BASIC
GET "L"
10 REM
20 REM TITLE PAGE
30 REM
40 COMMAND "SUSPEND COMMAND FILE"
50 PRINT CHR$(27);",cDI 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);"*dk30,250osZG"
120 PRINT "A"
130 PRINT "S"
140 PRINT CHR$(27);"*m5mZ";CHR$(27);"*d88,256oZRAPHICS"
150 PRINT CHR$(27);"*d88,176oZUDITING"
160 PRINT CHR$(27);"*d88,96oZYSTEM"
170 PRINT CHR$(27);"*m1a1b361,46,665,145e2a1mZ";CHR$(27);"*d375,106oZ"

180 PRINT CHR$(27);"*m4mZDATA ENTRY"
190 PRINT "  MODULE"
200 REM
210 REM PRINT TITLE PAGE FOR TWO SECONDS.
220 REM
230 A$=CHR$(27)&"@"
240 PRINT RPT$(A$,2);
250 PRINT CHR$(27);"*da";CHR$(27);"*dD";CHR$(27);"*dD";CHR$(27);"*mrZ";
260 PRINT CHR$(27);"*da1Z";CHR$(27);"h";CHR$(27);"J";
270 READ F$
280 DATA 3
290 COMMAND "F F "&F$&" L"
300 COMMAND "RESUME COMMAND FILE"
310 END
RUN
```

```
*****************************
*  PROGRAM TO DISPLAY MENU  *
*****************************

GET "L"
5 REM
6 REM DISPLAY MENU
7 REM
10 COMMAND "SUSPEND COMMAND FILE"
20 PRINT CHR$(27);",c DI W#1"
30 REM
40 REM HOME THE CURSOR, CLEAR THE GRAPHIC AND THE ALPHANUMERIC MEMORY
50 REM
130 PRINT CHR$(27);"*da";CHR$(27);"*dD";CHR$(27);"*dD";CHR$(27);"*mrZ";
135 PRINT CHR$(27);"*dalZ";CHR$(27);"h";CHR$(27);"J";
136 REM
137 REM DISPLAY THE MENU
138 REM
140 PRINT CHR$(27);"&dNMENUE:";CHR$(27);"&dJ  OPTION TO SELECT SYSTEM-FORMS ";
150 PRINT CHR$(27);"&d@";MOVCA(0,72);"PAGE 1"
160 PRINT CHR$(27);")B";CHR$(14);RPT$(";",79);CHR$(15)
170 PRINT CHR$(27);"&dDPRESS ";CHR$(34);"f1"&CHR$(34);" TO ";CHR$(34);"f8";
180 PRINT " TO SELECT THE DESIRED OPTION";CHR$(27);"&d@"
190 PRINT MOVCA(5,4);CHR$(27);"&dB  f1  ";CHR$(27);"&d@  DISPLAY MENU";
200 PRINT MOVCA(5,40);CHR$(27);"&dB  f5  ";CHR$(27);"&d@  NULL"
210 PRINT MOVCA(9,4);CHR$(27);"&dB  f2  ";CHR$(27);"&d@  SHORT FORM 01";
230 PRINT MOVCA(9,40);CHR$(27);"&dB  f6  ";CHR$(27);"&d@  NULL"
250 PRINT MOVCA(13,4);CHR$(27);"&dB  f3  ";CHR$(27);"&d@  LONG FORM 02";
260 PRINT MOVCA(13,40);CHR$(27);"&dB  f7  ";CHR$(27);"&d@  NULL"
270 PRINT MOVCA(17,4);CHR$(27);"&dB  f4  ";CHR$(27);"&d@  NULL"
280 PRINT MOVCA(17,40);CHR$(27);"&dB  f8  ";CHR$(27);"&d@  EXIT"
285 PRINT MOVCA(20,22);CHR$(27);"&dB  ENTER  ";CHR$(27);"&d@  ENTER DATA ON ";
286 PRINT "RIGHT CASSETTE";
290 PRINT MOVCA(22,0);CHR$(27);")B";CHR$(14);RPT$(";",79);CHR$(15)
300 PRINT CHR$(27);"&dNNOTE:";CHR$(27);"&d@ Place Data Cassette in right";
310 PRINT " drive to enter data."
313 REM FILE ON THE TAPE
314 REM
315 REM CHECK FOR THE KEY PUSHED 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 F$="5"\ GOTO 370
351 IF X=247 THEN COMMAND "EXIT"
360 GOTO 330
370 GETKBD OFF
380 COMMAND "F F "&F$&" L"
390 COMMAND "RESUME COMMAND FILE"
400 END
RUN
```

```
*********************************
*  SYSTEM DIRECTORY SHORT FORM 01  *
*          EDIT PROGRAM            *
*********************************
```

```
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$(27);",cDI W#2"
110 PRINT CHR$(27);",cF F. ";F;"L"
120 PRINT CHR$(27);"H";CHR$(27);"J";
130 PRINT CHR$(27);",cC F L DI"
140 PRINT CHR$(27);"W";
150 F=0
160 CALL Prtlno
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$>="0" AND X$<="9" THEN PRINT CHR$(X);
370 IF X=9 OR X$=" " THEN PRINT X$;
380 B=CSENA(R1,C1)
390 IF X<>239 AND C<>79 AND C1<>22 THEN 170
400 CALL Check(Flag,R)
410 IF Flag=0 THEN 160
420 IF GETKBD(X)=0 THEN 420
430 REM
440 REM ONLY CURSOR CAN BE MOVED OR DATA CAN BE ENTERED ON THE
450 REM RIGHT TAPE OR IT CAN BE IGNORED WHILE REACHED AT THE END
460 REM OF THE FORM.
470 REM
480 IF (X>=193 AND X<=196) THEN PRINT CHR$(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);",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$;
640 RETURN
650 F=0
660 A=CSENA(R1,C1)
670 IF C1<22 THEN PRINT RPT$(CHR$(9),3);\ GOTO 700
680 IF C1<25 OR C1>30 AND C1<36 OR C1>41 AND C1<47 THEN PRINT RPT$(CHR$(9),2);
690 Z=C1>24 AND C1<28 OR C1>35 AND C1<39 OR C1>46 AND C1<50
700 IF Z THEN PRINT CHR$(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 11
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$=DSPIN$(-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$=DSPIN$(-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
1032 REM DISPLAY THE ERROR MESSAGE IN DISPLAY WINDOW #5.
1033 REM
1040 PRINT CHR$(27);",cDI W#5"
1050 PRINT CHR$(27);"&dC ERROR DETECTED ";CHR$(27);"&d@"
1060 IF GETKBD(X)=0 THEN 1060
1070 IF X=239 THEN PRINT CHR$(27);",cCLO W#5"\ ELSE GOTO 1060
1080 REM
1090 REM CLEAR ONE LINE OF THE FORM
1100 REM
1110 CALL Clrln(R)
1120 SUBEND
1130 SUB Clrln(R)
1140 PRINT MOVCA(R,22);"      ";RPT$("##   ",2);RPT$("#",24);
1150 SUBEND
1190 SUB Prtlno
1191 REM
1192 REM PRINT LINE NUMBER ON THE FORM
1193 REM
1200 A=CSENA(R,C)
1210 IF R=1 THEN 1250
```

```
1220 I=(R-4) DIV 2+1
1230 PRINT MOVCA(R,22);
1240 IF I=10 THEN PRINT "10";\ ELSE PRINT " ";CHR$(48+I);
1250 SUBEND
1290 SUB Entr
1291 REM
1292 REM ENTER DATA FROM FORM TO RIGHT CASSETTE TAPE.
1293 REM
1300 ASSIGN "RIGHT TAPE" TO #2
1310 DIM A$[28]
1320 PRINT CHR$(27);"H";
1330 A$[1;1]="*"
1340 A$[2;2]=DSPIN$(-2,X)
1350 A$[4;1]=DSPIN$(-1,X)
1360 A$[5;1]=DSPIN$(-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]=DSPIN$(-2,X)
1430 IF (A$[J;2]="  " OR A$[J;2]="##") AND C=29 THEN 1490
1440 J=J+2
1450 NEXT C
1460 PRINT MOVCA(R,58);
1470 A$[J;22]=DSPIN$(-22,X)
1480 PRINT #2;A$[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
```

```
*************************
*  FORM 02 EDIT PROGRAM  *
*************************
```

```
GET "L"
10 REM
20 REM  SYSTEM DIRECTORY LONG FORM 02 EDIT PROGRAM
30 REM
40 COMMAND "SUSPEND 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 DI"
113 PRINT CHR$(27);"W";
114 Flag=0\Row=2\Ro=7
116 REM
118 REM  JUMP TO THE APPROPRIATE ROUTINE ACCORDING TO THE RECORD
119 REM  TYPE
120 REM
122 A=CSENA(R,C)
125 IF R=1 THEN GOSUB 300
130 IF R>=2 AND R<6 THEN GOSUB 500
140 IF R>=6 THEN GOSUB 800
150 REM
152 REM  PRINT CURSOR MOVEMENT
154 REM
158 IF X>=193 AND X<=196 THEN PRINT CHR$(X);\Flag=0
160 IF X=152 THEN CALL Entr\ 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
396 REM
410 IF GETKBD(X)=0 THEN 410
415 IF (X)=193 AND X(=196) OR X=152 OR (X)=240 AND X(=247) THEN GOTO 470
420 X$=UPC$(CHR$(X))
430 Chk=C(=52 AND (X$="A" OR X$="C" OR X$="E")
440 IF Chk THEN PRINT X$;\ GOTO 410
450 Chk=C)52 AND C(=79 AND (X$=" " OR X$="X" OR (X$)="A" AND X$(="C"))
460 IF Chk THEN PRINT X$;
470 RETURN
500 REM
504 REM  PRINT AND EDIT THE SECOND TYPE OF RECORD WHICH IS THE NEXT
506 REM  THREE LINE IN THE FORM
508 REM
520 IF GETKBD(X)=0 THEN 520
530 REM
532 REM PRINT ALLOWABLE CHARECTERS
534 REM
535 A=CSENA(R,C)
536 IF R)Row+1 THEN GOTO 710
537 IF X=152 OR (X)=240 AND X(=247) OR (X)=193 AND X(=196) THEN GOTO 740
539 IF X=239 THEN GOTO 590
540 X$=UPC$(CHR$(X))
550 Chk=(X$)="A" AND X$(="Z") OR X$=" " OR (X$)="0" AND X$(="9")
560 IF Chk AND R(=Row+1 AND C(=78 THEN PRINT CHR$(X);
570 IF C(78 THEN GOTO 520
572 REM
573 REM CHECK THE LINE
574 REM
590 PRINT MOVCA(R,48);
600 FOR I=1 TO 24
605 A=CSENA(R,C)
610 A$=DSPIN$(-1,X)
620 IF A$="#" THEN PRINT MOVCA(R,C);" ";
630 NEXT I
640 PRINT MOVCA(R,48);
650 A$=DSPIN$(-2,X)
660 PRINT MOVCA(R,57);
665 B$=DSPIN$(-2,X)
666 Chk=A$="  " OR B$="  "
668 IF Chk THEN PRINT MOVCA(R,48);RPT$(" ",24*(6-R));\ GOTO 710
670 REM
672 REM  PRINT THE ARROW
674 REM
678 IF R()Row+1 THEN PRINT MOVCA(Row,48);\ GOTO 710
680 PRINT CHR$(27);"X";
690 IF R()3 THEN PRINT MOVCA(Row,28);"   ";
```

```
700 PRINT MOVCA(R,28);">>>";CHR$(27);"W";
703 Row=R
705 R=R+1
707 IF R<=5 THEN PRINT MOVCA(R,48);\ GOTO 520
710 R=8\C=26
720 PRINT MOVCA(R,C);
740 RETURN
800 REM
802 REM  PRINT AND EDIT THE DESCRIPTION PORTION OF THE FORM
804 REM
820 IF (R=8 AND C=26) THEN PRINT USING "2A";"05" ELSE GOTO 830
825 PRINT MOVCA(8,29);
830 IF GETKBD(X)=0 THEN 830
840 REM
842 REM PRINT ALLOWABLE CHARECTERS
843 REM
844 IF X=239 THEN GOTO 940
845 IF (X)=193 AND X<=196) OR (X)=240 AND X<=247) OR X=152 THEN GOTO 1120
846 A=CSENA(R,C)
847 IF R>Ro+1 THEN R=Ro+1\ GOTO 1030
850 X$=UPC$(CHR$(X))
870 IF R<=Ro+1 AND C>27 AND Flag=0 THEN PRINT CHR$(X);
880 IF X$>="0" AND X$<="9" AND C<=27 AND R<=Ro+1 THEN PRINT X$;
900 IF C<78 THEN GOTO 830
910 REM
920 REM  CHECK THE LINE
930 REM
940 PRINT MOVCA(R,29);
950 A$=DSPIN$(-1,X)
960 IF A$="#" THEN PRINT RPT$("#",50);MOVCA(R,29);\ GOTO 830
970 FOR I=1 TO 49
974 A=CSENA(R,C)
975 A$=DSPIN$(-1,X)
980 IF A$="#" THEN PRINT MOVCA(R,C);" ";
990 NEXT I
992 REM
994 REM  PRINT LINE NUMBER
996 REM
1000 IF R=23 THEN Ro=R\Flag=1\ GOTO 1120
1010 IF R=Ro+1 THEN Ro=Ro+1\R=R+1
1020 IF R=Ro THEN R=R+1
1025 IF R<Ro THEN R=Ro+1
1030 PRINT MOVCA(Ro,26);
1040 N$[1;2]=DSPIN$(-2,X)
1050 N=VAL(N$[1;2])
1060 IF (N MOD 5)=0 THEN N=N+5\ ELSE N=N+1
1070 PRINT MOVCA(R,26);
1072 PRINT USING "2D";N
1075 PRINT MOVCA(R,29);
1080 GOTO 830
1120 RETURN
```

```
1140 SUB Enti
1142 REM
1144 REM  PACK  RECORDS IN A$ AND ENTER IT IN THR RIGHT TAPE
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;22]=DSPIN$(-22,X)
1270 A$[26;2]=DSPIN$(-2,X)
1280 IF A$[2;2]="  " OR A$[2;2]="**" THEN GOTO 1410
1290 A$[28;22]=DSPIN$(-22,X)
1292 A$[50;2]=DSPIN$(-2,X)
1300 A$[52;22]=DSPIN$(-22,X)
1315 IF A$[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
```

```
*****************************
*  DATA TRANSMISSION PROGRAM  *
*****************************

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$&"&k0R"
80 PRINT Loc$
90 COMMAND "RE R"
100 ASSIGN "RIGHT TAPE" TO #2
110 COMMAND "A S R"
120 PRINT Rmote$
130 ASSIGN "DATACOMM" TO #1
140 PRINT #1;CHR$(13)
150 GOSUB 220
160 PRINT #1;"7826078,PRAT"
170 GOSUB 220
180 GOSUB 220
190 PRINT #1;"JSGRAD"
200 GOSUB 220
210 GOTO 290
220 GETDCM ON \D$=""
230 IF GETDCM(D$)=0 THEN GOTO 230
240 PRINT D$;
250 IF D$<>":" AND D$<>"?" AND D$<>"/" 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 STRORAGE
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$)=0 THEN 590
600 IF D$<>"/" THEN 590
610 GETDCM OFF
620 RETURN
630 END
>
```

```
 1          IDENTIFICATION DIVISION.
 2
 3
 4         *            ******************************
 5         *            *  PROGRAM TO CREATE DATA BASE  *
 6         *            ******************************
 7
 8
 9          PROGRAM-ID. UP-DATE.
10          AUTHOR. PRATIMA MATHUR.
11
12
13          ENVIRONMENT DIVISION.
14          INPUT-OUTPUT SECTION.
15          FILE-CONTROL.
16              SELECT TRANSMITTED-FILE ASSIGN TO TRANS
17                  USE "RT=Z".
18              SELECT MASTER-FILE ASSIGN TO MASTER
19                  ORGANIZATION IS INDEXED
20                  ACCESS MODE IS DYNAMIC
21                  RECORD KEY IS REC-KEY.
22
23
24          DATA DIVISION.
25          FILE SECTION.
26
27
28         ******************************************
29         *  TRANSMITTED FILE CONTAINS TRANSECTION   *
30         *  RECORDS. THERE ARE FOUR TYPES OF        *
31         *  TRANSECTION RECORDS WHICH ARE DESCRIBED *
32         *  BELOW                                   *
33         ******************************************
34
35
36          FD  TRANSMITTED-FILE
37              LABEL RECORD IS OMITTED
38              DATA RECORDS ARE T-REC, FORM-01-REC, FORM-02-HEADER,
39              FORM-02-TRAILER
40              RECORD CONTAINS 5 TO 73 CHARACTERS.
41
42
43          01  T-REC.
44              03  SP-CHAR                    PIC X.
45              03  FORM-CODE                  PIC 99.
46              03  UPDATE-CODE                PIC X.
47              03  CURRENCY-STATUS            PIC X.
48
49          01  FORM-01-REC.
50              03  SYS-ID1                    PIC XX.
51              03  SUB-ID1                    PIC XX.
52              03  PROC-ID1                   PIC XX.
53              03  NAME1                      PIC X(22).
54
```

```
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
 55          01  FORM-02-HEADER.
 56              03  FILLER                          PIC X.
 57              03  SYS-ID2                         PIC XX.
 58              03  SYS-NAME2                       PIC X(22).
 59              03  SUB-ID2                         PIC XX.
 60              03  SUB-NAME2                       PIC X(22).
 61              03  PROC-ID2                        PIC XX.
 62              03  PROC-NAME2                      PIC X(22).
 63
 64          01  FORM-02-TRAILER.
 65              03  LINE-NO2                        PIC 99.
 66              03  DESCR2                          PIC X(50).
 67
 68
 69         ******************************************
 70         *  MASTER-FILE IS THE DATA BASE FILE WHICH *
 71         *  CONTAINS THREE TYPES OF RECORDS AS      *
 72         *  DESCRIBED BELOW                         *
 73         ******************************************
 74
 75
 76          FD  MASTER-FILE
 77              LABEL RECORDS ARE OMITTED.
 78
 79
 80          01  MASTER-PIC.
 81              02  REC-KEY.
 82                  03  KEY3.
 83                      04  KEY2.
 84                          05  KEY1.
 85                              06  CURR-STATUS PIC X.
 86                              06  SYS-ID      PIC XX.
 87                          05  SUB-ID          PIC XX.
 88                      04  PROC-ID             PIC XX.
 89                  03  LINE-NO                 PIC 99.
 90                  03  FILLER                  PIC X.
 91              02  NAME-OR-DESCR.
 92                  03  DESCR.
 93                      04  NAME                PIC X(22).
 94                      04      FILLER REDEFINES NAME.
 95                          05  TOTAL-NO-REC    PIC 9(4).
 96                          05  FILLER              PIC X(18).
 97                      04  FILLER              PIC X(28).
 98
 99
100          WORKING-STORAGE SECTION.
101
102
103         ******************************************
104         *  FOLLOWING FLAGS ARE USED IN THE PROGRAM *
105         *  MORE-DATA-REMAINS:  TO INDICATE END OF  *
106         *  TRANSECTION DATA                        *
107         *  END-DELETE: TO INDICATE DELITION OF     *
108         *  RECORDS COMPLETED                       *
```

```
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

```
109          *  REC-SUCCI TO INDICATE SUCCESSFUL TRANS- *
110          *  ECTION OF A TRANSECTION RECORD          *
111          ******************************************
112
113
114           01  FLAGS.
115               03  MORE-DATA-REMAINS          PIC XXX VALUE "YES".
116               03  END-DELETE                 PIC XXX VALUE "NO".
117               03  REC-SUCC                   PIC XXX VALUE "YES".
118
119
120          ******************************************
121          *  INFORMATION IS SAVED FROM TRANSECTION   *
122          *  RECORD FOR FURTHER USE                  *
123          ******************************************
124
125
126           01  SAVED-INFO.
127               03  W-FORM-CODE                PIC 99.
128               03  W-UPDATE-CODE              PIC X.
129
130
131           01  SAVED-VALUE.
132               02  W-REC-KEY.
133                   03  GR3.
134                       04  GR2.
135                           05  GR1.
136                               06  W-CURR-STATUS PIC X.
137                               06  W-SYS-ID   PIC XX.
138                           05  W-SUB-ID       PIC XX.
139                       04  W-PROC-ID          PIC XX.
140                   03  W-LINE-NO              PIC 99.
141                   03  FILLER                 PIC X.
142               02  W-DESCR                        PIC X(50).
143
144
145          ******************************************
146          *  FOLLOWING COUNTERS ARE USED FOR COUNTI- *
147          *  NG THE NUMBER OF RECORDS ADDED,CHANGED  *
148          *  AND DELETED DURING THE RUN AND TOTAL    *
149          *  NUMBER OF RECORDS IN THE MASTER-FILE    *
150          ******************************************
151
152
153
154
155           01  REC-DELETED                    PIC S999 VALUE ZEROS.
156           01  REC-CHANGED                    PIC S999 VALUE ZEROS.
157           01  REC-ADDED                      PIC S999 VALUE ZEROS.
158           01  REC-TOTAL                      PIC S9999.
159
160
161           PROCEDURE DIVISION.
162           OPEN-FILES.
```

```
COLUMN    1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
163                 OPEN INPUT TRANSMITTED-FILE.
164                 OPEN OUTPUT MASTER-FILE.
165             READ-REC.
166                 READ TRANSMITTED-FILE
167                     AT END PERFORM NO-REC.
168                 PERFORM UPDATE-PROC
169                     UNTIL MORE-DATA-REMAINS = "NO".
170
171
172             ******************************************
173             *  PRINT THE STATISTICS OF TRANSECTIONS  *
174             ******************************************
175
176
177                 DISPLAY "NUMBER OF RECORDS DELETED  ", REC-DELETED.
178                 DISPLAY "NUMBER OF RECORD CHANGED   ", REC-CHANGED.
179                 DISPLAY "NUMBER OF RECORD ADDED     ", REC-ADDED.
180                 MOVE SPACES TO MASTER-PIC.
181                 MOVE ZERO TO  TOTAL-NO-REC.
182                 MOVE TOTAL-NO-REC TO REC-TOTAL.
183                 ADD REC-ADDED TO REC-TOTAL.
184                 SUBTRACT REC-DELETED FROM REC-TOTAL.
185                 MOVE REC-TOTAL TO TOTAL-NO-REC.
186                 WRITE MASTER-PIC.
187                 DISPLAY "TOTAL NUMBER OF RECORDS IN THE FILE   ", REC-TOTAL.
188                 CLOSE TRANSMITTED-FILE, MASTER-FILE.
189                 STOP RUN.
190
191
192             NO-REC.
193                 MOVE "NO" TO MORE-DATA-REMAINS.
194                 DISPLAY "*** NO RECORD IN TRANSMITTED-FILE".
195
196
197             ******************************************
198             *  ANALYZE THE INPUT TRANSECTION RECORD AND*
199             *  PERFORM THE APPROPRIATE PROCEDURE       *
200             ******************************************
201
202
203             UPDATE-PROC.
204                 IF SP-CHAR = "*"
205                     PERFORM SP-PROC1
206                 ELSE
207                     IF SP-CHAR = "$"
208                         PERFORM SP-PROC2 THRU SP2-PROC-EXIT
209                     ELSE
210                         PERFORM CREAT-UPDATE-PROC THRU CREAT-UPDATE-EXIT.
211                 MOVE SPACES TO MASTER-PIC.
212                 READ TRANSMITTED-FILE
213                     AT END MOVE "NO" TO MORE-DATA-REMAINS.
214
215
216             ******************************************

        COLUMN   1         2         3         4         5         6         7         8
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
217             *  SAVE INPUT INFORMATION WHEN THE TRANS-  *
218             *  ECTION RECORD IS T-REC TYPE             *
219             ******************************************
220
221
222              SP-PROC1.
223                  MOVE FORM-CODE TO W-FORM-CODE.
224                  MOVE UPDATE-CODE TO W-UPDATE-CODE.
225                  MOVE CURRENCY-STATUS TO W-CURR-STATUS.
226
227
228             ******************************************
229             *  SAVE INPUT INFORMATION WHEN THE TRANS-  *
230             *  ECTION RECORD IS FORM-02 -HEADER TYPE   *
231             ******************************************
232
233
234              SP-PROC2.
235                  MOVE SYS-ID2 TO W-SYS-ID.
236                  MOVE SUB-ID2 TO W-SUB-ID.
237                  MOVE PROC-ID2 TO W-PROC-ID.
238                  MOVE ZEROS TO W-LINE-NO.
239                  MOVE W-REC-KEY TO REC-KEY.
240                  IF PROC-NAME2 NOT = SPACES
241                      MOVE PROC-NAME2 TO W-DESCR
242                  ELSE
243                      IF SUB-NAME2 NOT = SPACES
244                          MOVE SUB-NAME2 TO W-DESCR
245                      ELSE
246                          MOVE SYS-NAME2 TO W-DESCR
247                  IF W-DESCR NOT = SPACES
248
249
250             ******************************************
251             *  ADD SHORT DESCRIPTIVE RECORD IN THE     *
252             *  MASTER-FILE IF IT IS NOT ALREADY PRESENT*
253             ******************************************
254
255
256                  READ MASTER-FILE RECORD
257                      KEY IS REC-KEY
258                  INVALID KEY PERFORM ADD-PROC THRU ADD-PROC-EXIT.
259
260
261               SP2-PROC-EXIT.
262                  EXIT.
263
264
265             ******************************************
266             *  ASSEMBLE ALL INFORMATION IN WORKING-    *
267             *  STORAGE AND UPDATE THE MASTER-FILE      *
268             ******************************************
269
270
COLUMN    1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

```
271          CREAT-UPDATE-PROC.
272              IF W-FORM-CODE = 01
273                  MOVE SYS-ID1 TO W-SYS-ID
274                  MOVE SUB-ID1 TO W-SUB-ID
275                  MOVE PROC-ID1 TO W-PROC-ID
276                  MOVE ZEROS TO W-LINE-NO
277                  MOVE NAME1 TO W-DESCR
278              ELSE
279                  IF W-FORM-CODE = 02
280                      MOVE LINE-NO2 TO W-LINE-NO
281                      MOVE DESCR2 TO W-DESCR
282                  ELSE
283                      DISPLAY "*** FORM CODE NOT RECOGNIZED  ", W-FORM-CODE
284                      GO TO CREAT-UPDATE-EXIT.
285
286
287          ******************************************
288          *  ANALYZE THE UPDATE CODE AND PERFORM THE *
289          *  APPROPRIATE PROCEDURE. IF UPDATE CODE   *
290          *  IS NOT RECOGNIZED THEN DISPLAY THE ERROR*
291          *  MESSAGE.                                *
292          ******************************************
293
294
295              IF W-UPDATE-CODE = "A" OR "E"
296                  PERFORM ADD-PROC THRU ADD-PROC-EXIT
297              ELSE
298                  IF W-UPDATE-CODE = "C"
299                      PERFORM CHANGE-PROC
300                  ELSE
301                      IF W-UPDATE-CODE = "D"
302                          PERFORM DELETE-PROC THRU DELETE-PROC-EXIT
303                      ELSE
304                          DISPLAY "*** UPDATE CODE IS NOT RECOGNIZED   "
305          -       , W-UPDATE-CODE.
306              MOVE "YES" TO REC-SUCC.
307
308
309          CREAT-UPDATE-EXIT.
310              EXIT.
311
312
313          ******************************************
314          *  FIRST CHECK IF THE PARENT RECORD EXISTS *
315          *  IN THE MASTER-FILE. IF IT EXISTS THEN   *
316          *  ADD THE INPUT RECORD IN THE MASTER-FILE *
317          ******************************************
318
319
320          ADD-PROC.
321              IF W-LINE-NO NOT = ZERO
322                  MOVE W-REC-KEY TO REC-KEY
323                  MOVE ZERO TO LINE-NO
324                  READ MASTER-FILE RECORD
```

```
COLUMN    1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
                    KEY IS REC-KEY
                    INVALID KEY GO TO ADD-ERROR-PROC.
                MOVE ZERO TO LINE-NO.
                IF W-PROC-ID NOT = SPACE
                    MOVE SPACES TO KEY3
                    MOVE GR2 TO KEY2
                    READ MASTER-FILE
                        KEY IS REC-KEY
                        INVALID KEY GO TO ADD-ERROR-PROC.
                IF W-SUB-ID NOT = SPACES
                    MOVE SPACES TO KEY3
                    MOVE GR1 TO KEY1
                    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.
                    ADD 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.
                    ADD 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.


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

      COLUMN    1         2         3         4         5         6         7         8
      12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
379             *  DELETE RECORDS FROM THE MASTER-FILE      *
380             ********************************************
381
382
383              DELETE-PROC.
384                  MOVE W-REC-KEY TO REC-KEY.
385                  DELETE MASTER-FILE RECORD
386                      INVALID KEY PERFORM INV-KEY.
387                  IF REC-SUCC = "YES"
388                      ADD 1 TO REC-DELETED
389                  ELSE
390                      GO TO DELETE-PROC-EXIT.
391
392
393             ********************************************
394             *  CHECK THE KIND OF INPUT RECORD E.G.      *
395             *  SYSTEM,SUB-SYSTEM OR PROCEDURE RECORD    *
396             *  AND PERFORM THE APPROPRIATE PROCEDURE    *
397             ********************************************
398
399
400                  IF W-LINE-NO NOT = ZERO
401                      GO TO DELETE-PROC-EXIT.
402                  IF W-SUB-ID = SPACES
403                      PERFORM GR1-DELETE-PROC
404                          UNTIL END-DELETE = "YES"
405                  ELSE
406                          IF W-PROC-ID = SPACES
407                          PERFORM GR2-DELETE-PROC
408                              UNTIL END-DELETE = "YES"
409                      ELSE
410                          IF W-LINE-NO = ZEROS
411                              PERFORM GR3-DELETE-PROC
412                                  UNTIL END-DELETE = "YES".
413                  MOVE "NO" TO END-DELETE.
414
415              DELETE-PROC-EXIT.
416                  EXIT.
417
418
419              INV-KEY.
420                  MOVE "NO" TO REC-SUCC.
421                  DISPLAY "*** CAN NOT DELETE RECORD DOES NOT EXISTS FOR   "
422                  , MASTER-PIC.
423
424
425             ********************************************
426             *  DELETE ALL THE DESCEDANT RECORD OF THE   *
427             *  GIVEN SYSTEM                             *
428             ********************************************
429
430
431              GR1-DELETE-PROC.
432                  READ MASTER-FILE NEXT RECORD
```

```
COLUMN     1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
                    AT END MOVE SPACES TO W-REC-KEY.
                IF KEY1 = GR1
                    DELETE MASTER-FILE RECORD
                    ADD 1 TO REC-DELETED
                ELSE
                    MOVE "YES" 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
                    ADD 1 TO REC-DELETED
                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
                    ADD 1 TO REC-DELETED
                ELSE
                    MOVE "YES" TO END-DELETE.
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
 1          IDENTIFICATION DIVISION.
 2
 3
 4         *            ********************************
 5         *            *  PROGRAM TO UPDATE DATA BASE  *
 6         *            ********************************
 7
 8
 9          PROGRAM-ID. UP-DATE.
10          AUTHOR. PRATIMA MATHUR.
11
12
13          ENVIRONMENT DIVISION.
14          INPUT-OUTPUT SECTION.
15          FILE-CONTROL.
16              SELECT TRANSMITTED-FILE ASSIGN TO TRANS
17                  USE "RT=Z".
18              SELECT MASTER-FILE ASSIGN TO MASTER
19                  ORGANIZATION IS INDEXED
20                  ACCESS MODE IS DYNAMIC
21                  RECORD KEY IS REC-KEY.
22
23
24          DATA DIVISION.
25          FILE SECTION.
26
27
28         ******************************************
29         *  TRANSMITTED FILE CONTAINS TRANSECTION  *
30         *  RECORDS. THERE ARE FOUR TYPES OF       *
31         *  TRANSECTION RECORDS WHICH ARE DESCRIBED *
32         *  BELOW                                  *
33         ******************************************
34
35
36          FD  TRANSMITTED-FILE
37              LABEL RECORD IS OMITTED
38              DATA RECORDS ARE T-REC, FORM-01-REC, FORM-02-HEADER,
39              FORM-02-TRAILER
40              RECORD CONTAINS 5 TO 73 CHARACTERS.
41
42
43          01  T-REC.
44              03  SP-CHAR                        PIC X.
45              03  FORM-CODE                      PIC 99.
46              03  UPDATE-CODE                    PIC X.
47              03  CURRENCY-STATUS                PIC X.
48
49          01  FORM-01-REC.
50              03  SYS-ID1                        PIC XX.
51              03  SUB-ID1                        PIC XX.
52              03  PROC-ID1                       PIC XX.
53              03  NAME1                          PIC X(22).
54
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
 55          01  FORM-02-HEADER.
 56              03  FILLER                          PIC X.
 57              03  SYS-ID2                         PIC XX.
 58              03  SYS-NAME2                       PIC X(22).
 59              03  SUB-ID2                         PIC XX.
 60              03  SUB-NAME2                       PIC X(22).
 61              03  PROC-ID2                        PIC XX.
 62              03  PROC-NAME2                      PIC X(22).
 63
 64          01  FORM-02-TRAILER.
 65              03  LINE-NO2                        PIC 99.
 66              03  DESCR2                          PIC X(50).
 67
 68
 69        ******************************************
 70        *  MASTER-FILE IS THE DATA BASE FILE WHICH *
 71        *  CONTAINS THREE TYPES OF RECORDS AS      *
 72        *  DESCRIBED BELOW                         *
 73        ******************************************
 74
 75
 76          FD  MASTER-FILE
 77              LABEL RECORDS ARE OMITTED.
 78
 79
 80          01  MASTER-PIC.
 81              02  REC-KEY.
 82                  03  KEY3.
 83                      04  KEY2.
 84                          05  KEY1.
 85                              06  CURR-STATUS PIC X.
 86                              06  SYS-ID      PIC XX.
 87                          05  SUB-ID          PIC XX.
 88                      04  PROC-ID             PIC XX.
 89                  03  LINE-NO                 PIC 99.
 90                  03  FILLER                  PIC X.
 91              02  NAME-OR-DESCR.
 92                  03  DESCR.
 93                      04  NAME                PIC X(22).
 94                      04      FILLER REDEFINES NAME.
 95                          05  TOTAL-NO-REC    PIC 9(4).
 96                          05  FILLER              PIC X(18).
 97                      04  FILLER              PIC X(28).
 98
 99
100          WORKING-STORAGE SECTION.
101
102
103        ******************************************
104        *  FOLLOWING FLAGS ARE USED IN THE PROGRAM *
105        *  MORE-DATA-REMAINS: TO INDICATE END OF   *
106        *  TRANSECTION DATA                        *
107        *  END-DELETE: TO INDICATE DELITION OF     *
108        *  RECORDS COMPLETED                       *
```

```
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

```
109          *  REC-SUCC TO INDICATE SUCCESSFUL TRANS- *
110          *  ECTION OF A TRANSECTION RECORD         *
111          ******************************************
112
113
114           01  FLAGS.
115               03  MORE-DATA-REMAINS          PIC XXX VALUE "YES".
116               03  END-DELETE                 PIC XXX VALUE "NO".
117               03  REC-SUCC                   PIC XXX VALUE "YES".
118
119
120          ******************************************
121          *  INFORMATION IS SAVED FROM TRANSECTION  *
122          *  RECORD FOR FURTHER USE                 *
123          ******************************************
124
125
126           01  SAVED-INFO.
127               03  W-FORM-CODE                PIC 99.
128               03  W-UPDATE-CODE              PIC X.
129
130
131           01  SAVED-VALUE.
132               02  W-REC-KEY.
133                   03  GR3.
134                       04  GR2.
135                           05  GR1.
136                               06  W-CURR-STATUS PIC X.
137                               06  W-SYS-ID     PIC XX.
138                           05  W-SUB-ID         PIC XX.
139                       04  W-PROC-ID            PIC XX.
140                   03  W-LINE-NO                PIC 99.
141                   03  FILLER                   PIC X.
142               02  W-DESCR                          PIC X(50).
143
144
145          ******************************************
146          *  FOLLOWING COUNTERS ARE USED FOR COUNTI- *
147          *  NG THE NUMBER OF RECORDS ADDED,CHANGED  *
148          *  AND DELETED DURING THE RUN AND TOTAL    *
149          *  NUMBER OF RECORDS IN THE MASTER-FILE    *
150          ******************************************
151
152
153
154
155           01  REC-DELETED                    PIC S999 VALUE ZEROS.
156           01  REC-CHANGED                    PIC S999 VALUE ZEROS.
157           01  REC-ADDED                      PIC S999 VALUE ZEROS.
158           01  REC-TOTAL                      PIC S9999.
159
160
161           PROCEDURE DIVISION.
162           OPEN-FILES.
```

```
COLUMN    1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
163                 OPEN INPUT TRANSMITTED-FILE.
164                 OPEN I-O MASTER-FILE.
165             READ-REC.
166                 READ TRANSMITTED-FILE
167                     AT END PERFORM NO-REC.
168                 PERFORM UPDATE-PROC
169                     UNTIL MORE-DATA-REMAINS = "NO".
170
171
172             ******************************************
173             *  PRINT THE STATISTICS OF TRANSECTIONS  *
174             ******************************************
175
176
177                 DISPLAY "NUMBER OF RECORDS DELETED  ", REC-DELETED.
178                 DISPLAY "NUMBER OF RECORD CHANGED   ", REC-CHANGED.
179                 DISPLAY "NUMBER OF RECORD ADDED     ", REC-ADDED.
180                 MOVE SPACES TO MASTER-PIC.
181                 READ MASTER-FILE RECORD
182                     KEY IS REC-KEY
183                     INVALID KEY DISPLAY "***ERROR IN READ MASTER-FILE".
184                 MOVE TOTAL-NO-REC TO REC-TOTAL.
185                 ADD REC-ADDED TO REC-TOTAL.
186                 SUBTRACT REC-DELETED FROM REC-TOTAL.
187                 MOVE REC-TOTAL TO TOTAL-NO-REC.
188                 REWRITE MASTER-PIC
189                     INVALID KEY DISPLAY "***ERROR***".
190                 DISPLAY "TOTAL NUMBER OF RECORDS IN THE FILE   ", REC-TOTAL.
191                 CLOSE TRANSMITTED-FILE, MASTER-FILE.
192                 STOP RUN.
193
194
195             NO-REC.
196                 MOVE "NO" TO MORE-DATA-REMAINS.
197                 DISPLAY "*** NO RECORD IN TRANSMITTED-FILE".
198
199
200             ******************************************
201             *  ANALYZE THE INPUT TRANSECTION RECORD AND*
202             *  PERFORM THE APPROPRIATE PROCEDURE       *
203             ******************************************
204
205
206             UPDATE-PROC.
207                 IF SP-CHAR = "*"
208                     PERFORM SP-PROC1
209                 ELSE
210                     IF SP-CHAR = "$"
211                         PERFORM SP-PROC2 THRU SP2-PROC-EXIT
212                     ELSE
213                         PERFORM CREAT-UPDATE-PROC THRU CREAT-UPDATE-EXIT.
214                 MOVE SPACES TO MASTER-PIC.
215                 READ TRANSMITTED-FILE
216                     AT END MOVE "NO" TO MORE-DATA-REMAINS.

        COLUMN    1         2         3         4         5         6         7         8
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
217
218
219          ******************************************
220          *  SAVE INPUT INFORMATION WHEN THE TRANS-  *
221          *  ECTION RECORD IS T-REC TYPE             *
222          ******************************************
223
224
225           SP-PROC1.
226               MOVE FORM-CODE TO W-FORM-CODE.
227               MOVE UPDATE-CODE TO W-UPDATE-CODE.
228               MOVE CURRENCY-STATUS TO W-CURR-STATUS.
229
230
231          ******************************************
232          *  SAVE INPUT INFORMATION WHEN THE TRANS-  *
233          *  ECTION RECORD IS FORM-02 -HEADER TYPE   *
234          ******************************************
235
236
237           SP-PROC2.
238               MOVE SYS-ID2 TO W-SYS-ID.
239               MOVE SUB-ID2 TO W-SUB-ID.
240               MOVE PROC-ID2 TO W-PROC-ID.
241               MOVE ZEROS TO W-LINE-NO.
242               MOVE W-REC-KEY TO REC-KEY.
243               IF PROC-NAME2 NOT = SPACES
244                   MOVE PROC-NAME2 TO W-DESCR
245               ELSE
246                   IF SUB-NAME2 NOT = SPACES
247                       MOVE SUB-NAME2 TO W-DESCR
248                   ELSE
249                       MOVE SYS-NAME2 TO W-DESCR
250               IF W-DESCR NOT = SPACES
251
252
253          ******************************************
254          *  ADD SHORT DESCRIPTIVE RECORD IN THE     *
255          *  MASTER-FILE IF IT IS NOT ALREADY PRESENT*
256          ******************************************
257
258
259               READ MASTER-FILE RECORD
260                   KEY IS REC-KEY
261               INVALID KEY PERFORM ADD-PROC THRU ADD-PROC-EXIT.
262
263
264            SP2-PROC-EXIT.
265               EXIT.
266
267
268          ******************************************
269          *  ASSEMBLE ALL INFORMATION IN WORKING-    *
270          *  STORAGE AND UPDATE THE MASTER-FILE      *
```

COLUMN 1 2 3 4 5 6 7 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

```
271             ******************************************
272
273
274              CREAT-UPDATE-PROC.
275                  IF W-FORM-CODE = 01
276                      MOVE SYS-ID1 TO W-SYS-ID
277                      MOVE SUB-ID1 TO W-SUB-ID
278                      MOVE PROC-ID1 TO W-PROC-ID
279                      MOVE ZEROS TO W-LINE-NO
280                      MOVE NAME1 TO W-DESCR
281                  ELSE
282                      IF W-FORM-CODE = 02
283                          MOVE LINE-NO2 TO W-LINE-NO
284                          MOVE DESCR2 TO W-DESCR
285                      ELSE
286                          DISPLAY "*** FORM CODE NOT RECOGNIZED  ", W-FORM-CODE
287                          GO TO CREAT-UPDATE-EXIT.
288
289
290             ******************************************
291             *  ANALYZE THE UPDATE CODE AND PERFORM THE *
292             *  APPROPRIATE PROCEDURE. IF UPDATE CODE   *
293             *  IS NOT RECOGNIZED THEN DISPLAY THE ERROR*
294             *  MESSAGE.                                *
295             ******************************************
296
297
298                  IF W-UPDATE-CODE = "A" OR "E"
299                      PERFORM ADD-PROC THRU ADD-PROC-EXIT
300                  ELSE
301                      IF W-UPDATE-CODE = "C"
302                          PERFORM CHANGE-PROC
303                      ELSE
304                          IF W-UPDATE-CODE = "D"
305                              PERFORM DELETE-PROC THRU DELETE-PROC-EXIT
306                          ELSE
307                              DISPLAY "*** UPDATE CODE IS NOT RECOGNIZED   "
308             -    , W-UPDATE-CODE.
309                  MOVE "YES" TO REC-SUCC.
310
311
312              CREAT-UPDATE-EXIT.
313                  EXIT.
314
315
316             ******************************************
317             *  FIRST CHECK IF THE PARENT RECORD EXISTS *
318             *  IN THE MASTER-FILE. IF IT EXISTS THEN   *
319             *  ADD THE INPUT RECORD IN THE MASTER-FILE *
320             ******************************************
321
322
323              ADD-PROC.
324                  IF W-LINE-NO NOT = ZERO
```

```
COLUMN         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

```
325                     MOVE W-REC-KEY TO REC-KEY
326                     MOVE ZERO TO LINE-NO
327                     READ MASTER-FILE RECORD
328                     KEY IS REC-KEY
329                     INVALID KEY GO TO ADD-ERROR-PROC.
330                 MOVE ZERO TO LINE-NO.
331                 IF W-PROC-ID NOT = SPACE
332                     MOVE SPACES TO KEY3
333                     MOVE GR2 TO KEY2
334                     READ MASTER-FILE
335                         KEY IS REC-KEY
336                         INVALID KEY GO TO ADD-ERROR-PROC.
337                 IF W-SUB-ID NOT = SPACES
338                     MOVE SPACES TO KEY3
339                     MOVE GR1 TO KEY1
340                     READ MASTER-FILE
341                         KEY IS REC-KEY
342                         INVALID KEY GO TO ADD-ERROR-PROC.
343                 WRITE MASTER-PIC FROM SAVED-VALUE
344                     INVALID KEY PERFORM DUP-KEY.
345                     ADD 1 TO REC-ADDED.
346
347
348             ADD-PROC-EXIT.
349                 EXIT.
350
351
352             ADD-ERROR-PROC.
353                 DISPLAY "***CAN NOT ADD/EXTEND NO PARENT RECORD EXISTS FOR  "
354                 , SAVED-VALUE.
355                 GO TO ADD-PROC-EXIT.
356
357
358            ******************************************
359            *  CHANGE THE RECORD IN THE MASTER-FILE    *
360            ******************************************
361
362
363              CHANGE-PROC.
364                 REWRITE MASTER-PIC FROM SAVED-VALUE
365                     INVALID KEY PERFORM BAD-REC.
366                     ADD 1 TO REC-CHANGED.
367
368
369             DUP-KEY.
370                 SUBTRACT 1 FROM REC-ADDED.
371                 DISPLAY "*** CAN NOT ADD/EXTEND RECORD ALREADY EXISTS FOR   "
372             -   , MASTER-PIC.
373
374
375             BAD-REC.
376                 SUBTRACT 1 FROM REC-CHANGED.
377                 DISPLAY "*** CAN NOT CHANGE NO EXISTING RECORD FOR   "
378                 , MASTER-PIC.
```

```
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
379
380
381          ******************************************
382          *  DELETE RECORDS FROM THE MASTER-FILE   *
383          ******************************************
384
385
386           DELETE-PROC.
387               MOVE W-REC-KEY TO REC-KEY.
388               DELETE MASTER-FILE RECORD
389                   INVALID KEY PERFORM INV-KEY.
390               IF REC-SUCC = "YES"
391                   ADD 1 TO REC-DELETED
392               ELSE
393                   GO TO DELETE-PROC-EXIT.
394
395
396          ******************************************
397          *  CHECK THE KIND OF INPUT RECORD E.G.   *
398          *  SYSTEM,SUB-SYSTEM OR PROCEDURE RECORD *
399          *  AND PERFORM THE APPROPRIATE PROCEDURE *
400          ******************************************
401
402
403               IF W-LINE-NO NOT = ZERO
404                   GO TO DELETE-PROC-EXIT.
405               IF W-SUB-ID = SPACES
406                   PERFORM GR1-DELETE-PROC
407                       UNTIL END-DELETE = "YES"
408               ELSE
409                       IF W-PROC-ID = SPACES
410                       PERFORM GR2-DELETE-PROC
411                           UNTIL END-DELETE = "YES"
412                   ELSE
413                       IF W-LINE-NO = ZEROS
414                           PERFORM GR3-DELETE-PROC
415                               UNTIL END-DELETE = "YES".
416               MOVE "NO" TO END-DELETE.
417
418           DELETE-PROC-EXIT.
419               EXIT.
420
421
422           INV-KEY.
423               MOVE "NO" TO REC-SUCC.
424               DISPLAY "*** CAN NOT DELETE RECORD DOES NOT EXISTS FOR   "
425               , MASTER-PIC.
426
427
428          ******************************************
429          *  DELETE ALL THE DESCEDANT RECORD OF THE *
430          *  GIVEN SYSTEM                           *
431          ******************************************
432
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
433
434          GR1-DELETE-PROC.
435              READ MASTER-FILE NEXT RECORD
436                  AT END MOVE SPACES TO W-REC-KEY.
437              IF KEY1 = GR1
438                  DELETE MASTER-FILE RECORD
439                  ADD 1 TO REC-DELETED
440              ELSE
441                  MOVE "YES" TO END-DELETE.
442
443
444
445
446        ******************************************
447        *  DELETE ALL THE DESCEDANT RECORD OF THE  *
448        *  GIVEN SUB-SYSTEM                        *
449        ******************************************
450
451
452          GR2-DELETE-PROC.
453              READ MASTER-FILE NEXT RECORD
454                  AT END MOVE SPACES TO W-REC-KEY.
455              IF KEY2 = GR2
456                  DELETE MASTER-FILE RECORD
457                  ADD 1 TO REC-DELETED
458              ELSE
459                  MOVE "YES" TO END-DELETE.
460
461
462        ******************************************
463        *  DELETE ALL THE DESCEDANT RECORD OF THE  *
464        *  GIVEN PROCEDURE                         *
465        ******************************************
466
467
468          GR3-DELETE-PROC.
469              READ MASTER-FILE NEXT RECORD
470                  AT END MOVE SPACES TO W-REC-KEY.
471              IF KEY3 = GR3
472                  DELETE MASTER-FILE RECORD
473                  ADD 1 TO REC-DELETED
474              ELSE
475                  MOVE "YES" TO END-DELETE.
COLUMN 1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
 1          IDENTIFICATION DIVISION.
 2
 3
 4         *            ******************************
 5         *            *  QUERY PROCESSING PROGRAM  *
 6         *            ******************************
 7
 8
 9          PROGRAM-ID. RETREV.
10          AUTHOR. PRATIMA MATHUR.
11
12
13          ENVIRONMENT DIVISION.
14          INPUT-OUTPUT SECTION.
15          FILE-CONTROL.
16              SELECT PRINT-FILE ASSIGN TO PRNT
17                  USE "RT = Z".
18              SELECT MASTER-FILE ASSIGN TO MASTER
19                  ORGANIZATION IS INDEXED
20                  ACCESS MODE IS DYNAMIC
21                  RECORD KEY IS REC-KEY.
22
23
24          DATA DIVISION.
25          FILE SECTION.
26
27
28         ******************************************
29         *  MASTER FILE IS THE DATA BASE FILE     *
30         *  MASTER-FILE RECORDS ARE DESCRIBED BELOW *
31         ******************************************
32
33
34          FD  MASTER-FILE
35              LABEL RECORDS ARE OMITTED.
36
37
38          01  MASTER-PIC.
39              02  REC-KEY.
40                  03  KEY3.
41                      04  KEY2.
42                          05  KEY1.
43                              06  CURR-STATUS PIC X.
44                              06  SYS-ID      PIC XX.
45                          05  SUB-ID          PIC XX.
46                      04  PROC-ID             PIC XX.
47                  03  LINE-NO                 PIC 99.
48                  03  FILLER                  PIC X.
49              02  NAME-OR-DESCR.
50                  03  DESCR.
51                      04  NAME                PIC X(22).
52                      04  FILLER              PIC X(28).
53
54
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
55
56
57          ******************************************
58          *  PRINT-FILE IS USED TO PRINT THE OUTPUT  *
59          *  RECORDS                                 *
60          ******************************************
61
62
63           FD  PRINT-FILE
64               LABEL RECORDS ARE OMITTED.
65           01  PRINT-REC.
66               02  FILLER                          PIC X(8).
67               02  P-CURR-STATUS                   PIC X.
68               02  FILLER                          PIC X(11).
69               02  P-SYS-ID                        PIC XX.
70               02  FILLER                          PIC X(8).
71               02  P-SUB-ID                        PIC XX.
72               02  FILLER                          PIC X(9).
73               02  P-PROC-ID                       PIC XX.
74               02  FILLER                          PIC X(6).
75               02  P-NAME                          PIC X(22).
76           01  PRINT-DESCR-REC.
77               02  FILLER                          PIC X(8).
78               02  P-DESCR                         PIC X(50).
79               02  FILLER                          PIC X(8).
80          WORKING-STORAGE SECTION.
81           01  HEADINGS.
82               02  SHORT-HEAD-LINE.
83                   03  FILLER          PIC X        VALUE SPACE.
84                   03  CURR            PIC X(16)    VALUE "CURRENCY STATUS ".
85                   03  SYS             PIC X(8)     VALUE " SYSTEM ".
86                   03  SUB             PIC X(12)    VALUE " SUB SYSTEM ".
87                   03  PROC            PIC X(11)    VALUE " PROCEDURE ".
88                   03  FILLER          PIC X(5)     VALUE SPACES.
89                   03  NAM             PIC X(4)     VALUE "NAME".
90                   03  FILLER          PIC X(9)     VALUE SPACES.
91               02  DESCR-HEAD-LINE.
92                   03  DESC            PIC X(12)    VALUE " DESCRIPTION".
93                   03  FILLER          PIC X(58)    VALUE SPACES.
94
95
96          ******************************************
97          * FOLLOWING FLAGS ARE USED IN THE PROGRAM  *
98          *  END-FLAG:  REACHED END OF MASTER-FILE   *
99          *             WHILE READING IT SEQUENTIALLY*
100         *  END-SEC:   END OF CURRENCY STATUS       *
101         *  PRINT-LINE:ONE LINE OF INFORMATION IS   *
102         *             PRINTED                      *
103         *  CONT: WHETHER USER WANTS TO SEE THE     *
104         *        OUTPUT INFORMATION AFTER 22 LINES *
105         *        OF INFORMATION IS DISPLAYED ON THE*
106         *        SCREEN                            *
107         *  MSG: WHETHER THE PROGRAM IS RUNNING INT-*
108         *        ERECTIVELY OR NOT                 *
COLUMN      1         2         3         4         5         6         7         8
            12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
109          ******************************************
110
111
112           01  FLAGS.
113               02  END-FILE              PIC XXX VALUE "NO".
114               02  END-SEC               PIC XXX VALUE "NO".
115               02  PRINT-LINE            PIC XXX VALUE "NO".
116               02  CONT                  PIC XXX VALUE "YES".
117               02  MSG                   PIC X   VALUE "N".
118
119
120          ******************************************
121          *  FOLLOWING COUNTERS ARE USED IN THE    *
122          *  PROGRAM                               *
123          *  REC-CNT: TO DISPLAY ON SCREEN THE NUMBER*
124          *  OF RECORDS PRINTED ON THE HARD COPY   *
125          *  DISPL-CNT: NUMBER OF RECORDS DISPLAYED *
126          *  ON THE SCREEN                         *
127          *  PRINT-CNT: NUMBER OF RECORDS PRINTED ON *
128          *  THE HARD COPY                         *
129          ******************************************
130
131
132           01  COUNTERS.
133               02  REC-CNT               PIC 999  VALUE ZERO.
134               02  DISPL-CNT             PIC 99   VALUE   LOW-VALUES.
135               02  PRINT-CNT             PIC 99   VALUE   LOW-VALUES.
136
137
138          ******************************************
139          *  FOLLOWING SEPERATOR VARIABLES ARE USED *
140          *  WHICH HOLDS THE SEPERATORS WHILE      *
141          *  UNSTRING THE QUERY                    *
142          ******************************************
143
144
145           01  SEPERATORS.
146               02  CURR-SEP              PIC X.
147               02  SYS-SEP               PIC X.
148               02  SUB-SEP               PIC X.
149               02  PROC-SEP              PIC X.
150               02  LEVEL-SEP             PIC X.
151
152
153          ******************************************
154          *  PRINT-OUT: IT IS USED TO HOLD THE HARD *
155          *  COPY OPTION                           *
156          *  W-LEVEL: IT IS USED TO HOLD THE LEVEL *
157          *  NUMBER OF THE QUERY                   *
158          ******************************************
159
160
161           01  OPTIONS.
162               02  PRINT-OUT             PIC X.
```

```
COLUMN    1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
163                 02 W-LEVEL                PIC X.
164
165
166            ******************************************
167            *  SAVED-KEY-VALUE IS USED TO HOLD THE     *
168            *  RECORD WHICH HAS TO BE PRINTED          *
169            ******************************************
170
171
172             01  SAVED-KEY-VALUE.
173                 02  W-REC-KEY.
174                     03  GR3.
175                         04  GR2.
176                             05  GR1.
177                                 06  W-CURR-STATUS PIC X.
178                                 06  W-SYS-ID      PIC XX.
179                             05  W-SUB-ID          PIC XX.
180                         04  W-PROC-ID             PIC XX.
181                     03  W-LINE-NO                 PIC 99.
182                     03  FILLER                    PIC X.
183                 02  W-NAME                        PIC X(22).
184
185
186            ******************************************
187            *  QUERY-REC IS USED TO HOLD THE QUERY     *
188            ******************************************
189
190
191             01  QUERY-REC.
192                 02  W-QUERY              PIC X(16)   VALUE " THE QUERY IS   ".
193                 02  QUERY                    PIC X(14).
194             01  W-STATUS                 PIC XX.
195
196
197             PROCEDURE DIVISION.
198             OPEN-FILES.
199                 OPEN I-O MASTER-FILE.
200                 OPEN OUTPUT PRINT-FILE.
201
202
203             DISPLAY-MSGS.
204                 DISPLAY "ARE YOU ON THE TERMINAL? (TYPE ""Y"" OR ""YES"")".
205                 ACCEPT MSG.
206
207
208             READ-QUERY.
209                 IF MSG = "Y"
210                     DISPLAY "TYPE NEXT QUERY ".
211                     ACCEPT QUERY.
212                 IF QUERY = "END"
213                     GO TO FINISH.
214                     PERFORM INITIALIZE-PROC
215                     PERFORM UNSTRING-PROC THRU UNSTRING-PROC-EXIT
216                     GO TO READ-QUERY.
```

```
COLUMN    1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
217
218
219             FINISH.
220                 CLOSE MASTER-FILE, PRINT-FILE.
221                 STOP RUN.
222
223
224           ******************************************
225           *  INITIALIZE THE COUNTERS AND FLAGS     *
226           ******************************************
227
228
229             INITIALIZE-PROC.
230                 MOVE ZERO TO DISPL-CNT, PRINT-CNT, REC-CNT.
231                 MOVE "NO" TO END-FILE, END-SEC, PRINT-LINE.
232                 MOVE "YES" TO CONT.
233                 MOVE SPACES TO PRINT-REC, W-STATUS, SAVED-KEY-VALUE,
234                 OPTIONS.
235
236
237           ******************************************
238           *  BREAK THE QUERY IN SEPARATE DATA ITEMS *
239           ******************************************
240
241
242             UNSTRING-PROC.
243                 UNSTRING QUERY
244                     DELIMITED BY ALL "/" OR ALL "." OR ALL ","
245                     INTO W-STATUS, DELIMITER IN CURR-SEP
246                     W-SYS-ID, DELIMITER IN SYS-SEP
247                     W-SUB-ID, DELIMITER IN SUB-SEP
248                     W-PROC-ID, DELIMITER IN PROC-SEP
249                     W-LEVEL, DELIMITER IN LEVEL-SEP
250                     PRINT-OUT.
251
252
253           ******************************************
254           *  MOVE THE VALUES IN CORRECT VARIABLES BY *
255           *  EXAMINING THE DELIMITERS               *
256           ******************************************
257
258
259                 IF CURR-SEP = "."
260                     MOVE W-SUB-ID TO PRINT-OUT
261                     MOVE W-SYS-ID TO W-LEVEL
262                     MOVE SPACES TO W-SUB-ID, W-SYS-ID
263                 ELSE
264                     IF SYS-SEP = "."
265                         MOVE W-PROC-ID TO PRINT-OUT
266                         MOVE W-SUB-ID TO W-LEVEL
267                         MOVE SPACES TO W-PROC-ID, W-SUB-ID
268                     ELSE
269                         IF SUB-SEP = "."
270                             MOVE W-LEVEL TO PRINT-OUT
```

```
COLUMN  1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
                                MOVE W-PROC-ID TO W-LEVEL
                                MOVE SPACES TO W-PROC-ID.
                IF CURR-SEP NOT = "/"
                    MOVE W-SUB-ID TO W-PROC-ID
                    MOVE W-SYS-ID TO W-SUB-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 GR1 TO KEY1
                IF W-SYS-ID NOT = SPACE
                    START MASTER-FILE
                        KEY = KEY1
                        INVALID KEY GO TO RECORD-ERROR-PROC.
                IF W-SYS-ID = SPACE
                    START MASTER-FILE
                        KEY > KEY1
                        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        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789
```

```
325
326          ******************************************
327          *  PROCESS THE DETAILED DESCRIPTION OUTPUT *
328          ******************************************
329
330
331           LEVEL0-PROC.
332               IF W-SYS-ID = SPACES
333                   GO TO LEVEL0-EXIT.
334               MOVE GR3 TO KEY3.
335               MOVE ZERO TO LINE-NO.
336
337
338          ******************************************
339          *  POSITION THE MASTER-FILE IN SHORT       *
340          *  DESCRIPTIVE RECORD OF THE GIVEN KEY AND *
341          *  PERFORM THE APPROPRIATE PROCEDURE       *
342          ******************************************
343
344
345               READ MASTER-FILE RECORD
346                   KEY IS REC-KEY
347                   INVALID KEY GO TO LEVEL0-EXIT.
348               PERFORM LEVEL0-CHECK-PROC
349                   UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".
350
351
352           LEVEL0-EXIT.
353               EXIT.
354
355
356          ******************************************
357          *  OUTPUT SHORT DESCRIPTIVE RECORD AND THE *
358          *  DETAILED DESCRIPTION OF THE GIVEN KEY   *
359          ******************************************
360
361
362           LEVEL0-CHECK-PROC.
363               IF KEY3 = GR3
364                   IF LINE-NO = ZERO
365                       MOVE NAME TO W-NAME
366                       PERFORM PRINT-SHORT-LINE-PROC
367                       PERFORM PRINT-DESCR-HEAD-PROC
368                   ELSE
369                       PERFORM PRINT-DESCR-PROC
370               ELSE
371                   MOVE "YES" TO END-SEC.
372                   READ MASTER-FILE NEXT RECORD
373                       AT END MOVE "YES" TO END-FILE.
374
375
376          ******************************************
377          *  PRINT ALL THE RECORDS OF A GIVEN AT     *
378          *  SYSTEM"S LEVEL                          *
```

COLUMN 1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

```
379             ******************************************
380
381
382              LEVEL1-PROC.
383                  PERFORM LEVEL1-CHECK-PROC
384                      UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".
385
386
387              LEVEL1-CHECK-PROC.
388                  IF W-CURR-STATUS = CURR-STATUS
389                          IF SUB-ID = SPACE AND PROC-ID = SPACE AND LINE-NO =
390                            ZERO
391                          IF W-SYS-ID = SPACES
392                              PERFORM PRINT-SHORT-LINE-PROC
393                          ELSE
394                              IF W-SYS-ID = SYS-ID
395                                  MOVE "YES" TO END-SEC
396                                  PERFORM PRINT-SHORT-LINE-PROC
397                              ELSE
398                                  MOVE "YES" TO END-SEC
399                      ELSE
400                          NEXT SENTENCE
401                  ELSE
402                      MOVE "YES" TO END-SEC.
403                      READ MASTER-FILE NEXT RECORD
404                          AT END MOVE "YES" TO END-FILE.
405
406
407             ******************************************
408             *  PRINT ALL THE PARENT RECORD AND THE     *
409             *  RECORD OF A GIVEN KEY AT SUB-SYSTEM"S   *
410             *  LEVEL                                   *
411             ******************************************
412
413
414              LEVEL2-PROC.
415                  PERFORM LEVEL2-PROC1
416                      UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".
417
418
419              LEVEL2-PROC1.
420                  PERFORM LEVEL1-CHECK-PROC
421                          UNTIL PRINT-LINE = "YES" OR END-SEC = "YES" OR
422                              END-FILE = "YES" OR CONT = "NO".
423                  IF PRINT-LINE = "YES" AND CONT = "YES"
424                      MOVE "NO" TO PRINT-LINE, END-SEC
425                      PERFORM LEVEL2-CHECK-PROC
426                          UNTIL END-SEC = "YES" OR END-FILE = "YES" OR CONT =
427                            "NO"
428                  MOVE "NO" TO END-SEC, PRINT-LINE.
429
430
431              LEVEL2-CHECK-PROC.
432                  IF (W-CURR-STATUS = CURR-STATUS AND W-SYS-ID = SPACE) OR

COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
433                 GR1 = KEY1
434                     IF PROC-ID = SPACES AND LINE-NO = ZERO
435                         IF W-SUB-ID = SPACE
436                             PERFORM PRINT-SHORT-LINE-PROC
437                         ELSE
438                             IF W-SUB-ID = SUB-ID
439                                 MOVE "YES" TO END-SEC
440                                 PERFORM PRINT-SHORT-LINE-PROC
441                             ELSE
442                                 NEXT SENTENCE
443                     ELSE
444                         NEXT SENTENCE
445                 ELSE
446                     MOVE "YES" TO END-SEC.
447                     READ MASTER-FILE NEXT RECORD
448                         AT END MOVE "YES" TO END-FILE.
449
450
451             ******************************************
452             *  PRINT ALL THE PARENT RECORD AND THE   *
453             *  RECORD OF A GIVEN KEY AT PROCEDURE LEVEL*
454             ******************************************
455
456
457              LEVEL3-PROC.
458                  PERFORM LEVEL3-PROC1
459                      UNTIL END-FILE = "YES" OR END-SEC = "YES" OR CONT = "NO".
460
461
462              LEVEL3-PROC1.
463                  PERFORM LEVEL1-CHECK-PROC
464                          UNTIL PRINT-LINE = "YES" OR END-SEC = "YES" OR
465                                END-FILE = "YES" OR CONT = "NO".
466                  IF PRINT-LINE = "YES" AND CONT = "YES"
467                      MOVE "NO" TO PRINT-LINE, END-SEC
468                      PERFORM LEVEL3-PROC2
469                          UNTIL END-SEC = "YES" OR END-FILE = "YES" OR
470                            CONT = "NO"
471                      MOVE "NO" TO END-SEC, PRINT-LINE.
472
473
474              LEVEL3-PROC2.
475                  PERFORM LEVEL2-CHECK-PROC
476                          UNTIL PRINT-LINE = "YES" OR END-SEC = "YES" OR
477                                END-FILE = "YES" OR CONT = "NO".
478                  IF PRINT-LINE = "YES" AND CONT = "YES"
479                      MOVE "NO" TO PRINT-LINE, END-SEC
480                      PERFORM LEVEL3-CHECK-PROC
481                              UNTIL END-SEC = "YES" OR END-FILE = "YES" OR
482                                CONT = "NO"
483                      MOVE "NO" TO END-SEC, PRINT-LINE.
484
485
486              LEVEL3-CHECK-PROC.

         COLUMN    1         2         3         4         5         6         7         8
         12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
487                 IF (W-CURR-STATUS = CURR-STATUS AND (W-SYS-ID = SPACE OR
488                 W-SUR-ID = SPACE )) OR GR2 = KEY2
489                     IF LINE-NO = ZERO
490                         IF W-PROC-ID = SPACE
491                             PERFORM PRINT-SHORT-LINE-PROC
492                         ELSE
493                             IF W-PROC-ID = PROC-ID
494                                 MOVE "YES" TO END-SEC
495                                 PERFORM PRINT-SHORT-LINE-PROC
496                             ELSE
497                             NEXT SENTENCE
498                     ELSE
499                         NEXT SENTENCE
500                 ELSE
501                     MOVE "YES" TO END-SEC.
502                     READ MASTER-FILE NEXT RECORD
503                         AT END MOVE "YES" TO END-FILE.
504
505
506             ******************************************
507             *  PRINT THE HEADING ON THE HARD COPY    *
508             ******************************************
509
510
511              PRINT-HEAD-PROC.
512                 WRITE PRINT-REC FROM QUERY-REC
513                     AFTER ADVANCING PAGE.
514                 WRITE PRINT-REC FROM SHORT-HEAD-LINE
515                     AFTER ADVANCING 2 LINE.
516                 MOVE SPACES TO PRINT-REC.
517                 WRITE PRINT-REC.
518                 MOVE 2 TO PRINT-CNT.
519
520
521             ******************************************
522             *  DISPLAY THE HEADING ON THE SCREEN     *
523             ******************************************
524
525
526              DISPL-HEAD-PROC.
527                 DISPLAY SHORT-HEAD-LINE.
528                 MOVE SPACES TO PRINT-REC.
529                 DISPLAY PRINT-REC.
530                 MOVE 2 TO DISPL-CNT.
531
532             ******************************************
533             *  PRINT THE DESCRIPTION HEADING ON THE  *
534             *  HARD COPY OR DISPLAY IT ON THE SCREEN *
535             ******************************************
536
537
538              PRINT-DESCR-HEAD-PROC.
539                 IF PRINT-OUT NOT = "P" AND DISPL-CNT = 3
540                     DISPLAY PRINT-REC
```

```
COLUMN  1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
                    DISPLAY DESCR-HEAD-LINE
                    ADD 2 TO DISPL-CNT.
                IF PRINT-OUT = "P" AND PRINT-CNT = 3
                    WRITE PRINT-REC
                    WRITE PRINT-REC FROM DESCR-HEAD-LINE
                MOVE SPACES TO PRINT-REC.
                    ADD 2 TO PRINT-CNT.


            ******************************************
            *  PRINT ON HARD COPY OR DISPLAY ON SCREEN *
            *  THE SHORT DESCRIPTIVE LINE              *
            ******************************************


             PRINT-SHORT-LINE-PROC.
                IF PRINT-OUT NOT = "P" AND DISPL-CNT = ZERO
                    PERFORM DISPL-HEAD-PROC.
                IF PRINT-OUT = "P" AND PRINT-CNT = ZERO
                    PERFORM PRINT-HEAD-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 SUB-ID TO P-SUB-ID.
                MOVE PROC-ID TO P-PROC-ID.
                MOVE NAME TO P-NAME.
                IF PRINT-OUT NOT = "P"
                    DISPLAY PRINT-REC
                    ADD 1 TO DISPL-CNT
                    IF DISPL-CNT = 22
                        MOVE ZERO TO DISPL-CNT
                        PERFORM DISPL-CONT-PROC.
                IF PRINT-OUT = "P"
                    WRITE PRINT-REC
                    ADD 1 TO REC-CNT
                    ADD 1 TO PRINT-CNT
                    IF PRINT-CNT = 55
                        MOVE ZERO TO PRINT-CNT.
                MOVE SPACES TO PRINT-REC.


            ******************************************
            *  PRINT ON HARD COPY OR DISPLAY ON SCREEN *
            *  THE DETAILED DESCRIPTION                *
            ******************************************


        COLUMN  1         2         3         4         5         6         7         8
        12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
595              PRINT-DESCR-PROC.
596                  IF PRINT-CNT = ZERO AND PRINT-OUT = "P"
597                      PERFORM PRINT-HEAD-PROC.
598                  IF DISPL-CNT = ZERO AND PRINT-OUT NOT = "P"
599                      PERFORM DISPL-HEAD-PROC.
600                  IF PRINT-CNT = 2 OR DISPL-CNT = 2
601
602
603              ******************************************
604              *  ASSEMBLE THE INFORMATION IN PRINT-REC  *
605              ******************************************
606
607
608                      MOVE CURR-STATUS TO P-CURR-STATUS
609                      MOVE SYS-ID TO P-SYS-ID
610                      MOVE SUB-ID TO P-SUB-ID
611                      MOVE PROC-ID TO P-PROC-ID
612                      MOVE W-NAME TO P-NAME.
613                  IF DISPL-CNT = 2 AND PRINT-OUT NOT = "P"
614                      DISPLAY PRINT-REC
615                      ADD 1 TO DISPL-CNT.
616                  IF PRINT-CNT = 2 AND PRINT-OUT = "P"
617                      WRITE PRINT-REC
618                      ADD 1 TO PRINT-CNT.
619                  IF PRINT-CNT = 3 OR DISPL-CNT = 3
620                      MOVE SPACES TO PRINT-REC
621                      PERFORM PRINT-DESCR-HEAD-PROC.
622                  MOVE DESCR TO P-DESCR.
623                  IF PRINT-OUT NOT = "P"
624                      DISPLAY PRINT-REC
625                      ADD 1 TO DISPL-CNT
626                      IF DISPL-CNT = 22
627                          MOVE ZERO TO DISPL-CNT
628                          PERFORM DISPL-CONT-PROC.
629                  IF PRINT-OUT = "P"
630                      WRITE PRINT-REC
631                      ADD 1 TO REC-CNT
632                      ADD 1 TO PRINT-CNT
633                      IF PRINT-CNT = 55
634                          MOVE ZERO TO PRINT-CNT.
635                  MOVE SPACES TO PRINT-REC.
636
637
638              ******************************************
639              *  CHECK IF THE USER WANTS TO SEE MORE    *
640              *  AFTER DISPLAYING 22 LINES OF INFORMATION*
641              *  ON THE SCREEN                          *
642              ******************************************
643
644
645              DISPL-CONT-PROC.
646                  DISPLAY "DO YOU WANT TO SEE MORE ? (TYPE ""YES"" OR ""NO"")".
647                  ACCEPT CONT.
```

```
COLUMN      1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
*****************************
* PROGRAM TO CONSTRUCT FORMS  *
*****************************

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$(27);"J";
130 FOR A=1 TO J
135 FOR L=V(A,3) TO V(A,4) STEP V(A,5)
140 CALL V1(V$(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 H1(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 DROW HORIZENTAL LINE
510 SUB H1(F$,S,E,L)
520 IF F$="TKL" THEN X$=";"
530 IF F$="TNL" THEN X$=","
535 FOR I=S TO E
540 PRINT MOVCA(L,I);CHR$(27);")B";CHR$(14);X$;CHR$(15);
545 NEXT I
550 SUBEND
610 SUB V1(F$,S,E,L)
620 IF F$="TKC" THEN X$=":"
630 IF F$="TNC" THEN X$="."
640 FOR I=S TO E
650 PRINT MOVCA(I,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 CORNNER CHARACTERS
1010 IF S>C OR E<C OR P>L OR Q<L THEN 1610
1020 A=9
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 S=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 A GOSUB 1250,1290,1190,1380,1420,1330,1460,1510,1560
1160 PRINT MOVCA(L,C);X$;
1170 GOTO 1610
1190 ON B GOTO 1210,1220,1230,1220
1195 REM LEFT MIDDLE CORNNER
1210 X$="1"\ GOTO 1240
1220 X$="5"\ GOTO 1240
1230 X$="!"\ GOTO 1240
1240 RETURN
1250 ON B GOTO 1260,1260,1260,1270
1255 REM UPPER LEFTMOST CORNNER
1260 X$="Q"\ GOTO 1280
1270 X$="R"\ GOTO 1280
1280 RETURN
1290 ON B GOTO 1300,1300,1300,1310
1295 REM LOWER LEFTMOST CORNNER
1300 X$="A"\ GOTO 1320
1310 X$="F"
1320 RETURN
```

```
1330 ON B GOTO 1340,1350,1360,1350
1335 REM RIGHT MIDDLE CORNNER
1340 X$="2"\ GOTO 1370
1350 X$="6"\ GOTO 1370
1360 X$=CHR$(34)
1370 RETURN
1375 REM UPPER RIGHTMOST CORNNER
1380 ON B GOTO 1390,1390,1390,1400
1385 REM UPPER RIGHTMOST CORNNER
1390 X$="W"\ GOTO 1410
1400 X$="T"
1410 RETURN
1420 ON B GOTO 1430,1430,1430,1440
1425 REM LOWER RIGHTMOST CORNNER
1430 X$="S"\ GOTO 1450
1440 X$="G"
1450 RETURN
1460 ON B GOTO 1470,1480,1490,1490
1465 REM UPPER MIDDLE CORNNER
1470 X$="3"\ GOTO 1500
1480 X$="#"\ GOTO 1500
1490 X$="7"
1500 RETURN
1510 ON B GOTO 1520,1530,1540,1540
1515 REM LOWER MIDDLE CORNNER
1520 X$="4"\ GOTO 1550
1530 X$="$"\ GOTO 1550
1540 X$="8"
1550 RETURN
1560 ON B GOTO 1570,1580,1590,1600
1565 REM MIDDLE INTERSECTION
1570 X$="0"\ GOTO 1605
1580 X$="+"\ GOTO 1605
1590 X$="*"\ GOTO 1605
1600 X$="/"
1605 RETURN
1610 SUBEND
1700 SUB Field(F$,S,E,L)
1710 REM DISPLAY FIELD
1720 IF F$="UPF" THEN X$=CHR$(27)&"["\Y$=CHR$(27)&"]"\ GOTO 1770
1730 IF F$="IVF" THEN X$=CHR$(27)&"&dB"\Y$=CHR$(27)&"&d@"\ GOTO 1770
1740 IF F$="HBF" THEN X$=CHR$(27)&"&dH"\Y$=CHR$(27)&"&d@"\ GOTO 1770
1750 IF F$="IHF" THEN X$=CHR$(27)&"&dJ"\Y$=CHR$(27)&"&d@"\ GOTO 1770
1760 IF F$="TOF" THEN X$=CHR$(27)&"("\Y$=CHR$(27)&")"
1770 PRINT MOVCA(L,S);X$;RPT$("#",E-S+1);Y$;
1780 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
PAYROLL
```

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

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    01                              STUDENT INFO SYSTEM
 TYPE NEXT QUERY
? 01.03,2

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    01                              STUDENT INFO SYSTEM
                    01        03                    ADMISSIONS
 TYPE NEXT QUERY
? 01.03.05,3

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    01                              STUDENT INFO SYSTEM
                    01        03                    ADMISSIONS
                    01        03          05        PART-TIME ADMISSION
 TYPE NEXT QUERY
? 01,2

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    01                              STUDENT INFO SYSTEM
                    01        03                    ADMISSIONS
                    01        05                    REGISTRATION
                    01        07                    EXAMINATIONS
                    01        09                    GRADUATION
                    01        11                    TRANSCRIPT
 TYPE NEXT QUERY
? 01,3

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    01                              STUDENT INFO SYSTEM
                    01        03                    ADMISSIONS
                    01        03          01        OAC INPUT
                    01        03          03        GRADUATE ADMISSION
                    01        03          05        PART-TIME ADMISSION
                    01        05                    REGISTRATION
                    01        05          01        UNDER-GRADUATE REGISTR
                    01        05          03        LATE REGISTRATION
                    01        05          05        GRADUATE REGISTRATION
                    01        07                    EXAMINATIONS
                    01        07          01        WINTER
                    01        07          03        SUMMER
                    01        07          05        SPRING
                    01        09                    GRADUATION
                    01        09          01        SPRING
                    01        09          03        FALL
                    01        11                    TRANSCRIPT
```

```
 TYPE NEXT QUERY
? 02,3

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    02                              PAYROLL
                    02        01                    SALARIED PAYROLL
                    02        01         01         EMPLOYEE HIRE
                    02        01         03         CHANGE PROCEDURE
                    02        01         05         PAY
                    02        01         07         T4 YEAR-END
                    02        03                    HOURLY PAYROLL
                    02        03         01         TIME REPORTIN
                    02        03         03         CHANGE PROCEDURE
                    02        03         05         PAY
                    02        03         07         T4 YEAR-END
                    02        05                    TEMPORARY PAYROLL
                    02        05         01         PAY
                    02        05         03         T4 YEAR-END
 TYPE NEXT QUERY
? 02,0

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    02                              PAYROLL

  DESCRIPTION
        THE SCOPE OF THIS SYSTEM INCLUDES THE PRODUCTION
        OF PAY CHEQUES AND THE PAYROLL JOURNAL FOR ALL
        SALARIED AND HOURLY-PAID EMPLOYEES, THE
        WITHOLDING OF INCOME TAX, PENSION, AND OTHER
        DEDUCTIONS FROM PAY, THE COSEQUENT PROCESSING
        OF SUCH DEDUCTIONS ON A QUATERLY AND ANUAL BASIS.
        THIS ENTAILS THE MAINTANANCE 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 NESESSARY
        PERSONAL INFORMATION.
 TYPE NEXT QUERY
? 02.01,0

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

                    02        01                    SALARIED PAYROLL

  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 INOVKED.
 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
         GOVERMENT.
```

```
 TYPE NEXT QUERY
? X/,3

  CURRENCY STATUS  SYSTEM  SUB SYSTEM  PROCEDURE      NAME

         X           01                             STUDENT INFO SYSTEM
         X           01        01                   PURCHASE
         X           01        01         01        REQUESTION
         X           01        01         03        PURCHASE ORDER
         X           01        03                   RECEIVING
         X           01        03         01        RECEIVING ADVICE
         X           01        03         03        BACK ORDER
         X           01        05                   ACCOUNT PAYMENT
         X           01        05         01        CHEQUES
 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/,P

 QUERY NOT RECOGNIZED X/,P
 TYPE NEXT QUERY
? X/,3,P

 NUMBER OF RECORDS PRINTED =     9
 TYPE NEXT QUERY
? END
 JOB REPRIEVED.
```

THE QUERY IS ,1,P

CURRENCY STATUS	SYSTEM	SUB SYSTEM	PROCEDURE	NAME
	01			STUDENT INFO SYSTEM
	02			PAYROLL

THE QUERY IS ,2,P

CURRENCY STATUS	SYSTEM	SUB SYSTEM	PROCEDURE	NAME
	01			STUDENT INFO SYSTEM
	01	03		ADMISSIONS
	01	05		REGISTRATION
	01	07		EXAMINATIONS
	01	09		GRADUATION
	01	11		TRANSCRIPT
	02			PAYROLL
	02	01		SALARIED PAYROLL
	02	03		HOURLY PAYROLL
	02	05		TEMPORARY PAYROLL

THE QUERY IS X/,3,P

CURRENCY STATUS	SYSTEM	SUB SYSTEM	PROCEDURE	NAME
X	01			STUDENT INFO SYSTEM
X	01	01		PURCHASE
X	01	01	01	REQUESTION
X	01	01	03	PURCHASE ORDER
X	01	03		RECEIVING
X	01	03	01	RECEIVING ADVICE
X	01	03	03	BACK ORDER
X	01	05		ACCOUNT PAYMENT
X	01	05	01	CHEQUES

THE QUERY IS ,3,P

CURRENCY STATUS	SYSTEM	SUB SYSTEM	PROCEDURE	NAME
	01			STUDENT INFO SYSTEM
	01	03		ADMISSIONS
	01	03	01	OAC INPUT
	01	03	03	GRADUATE ADMISSION
	01	03	05	PART-TIME ADMISSION
	01	05		REGISTRATION
	01	05	01	UNDER-GRADUATE REGISTR
	01	05	03	LATE REGISTRATION
	01	05	05	GRADUATE REGISTRATION
	01	07		EXAMINATIONS
	01	07	01	WINTER
	01	07	03	SUMMER
	01	07	05	SPRING
	01	09		GRADUATION
	01	09	01	SPRING
	01	09	03	FALL
	01	11		TRANSCRIPT
	02			PAYROLL
	02	01		SALARIED PAYROLL
	02	01	01	EMPLOYEE HIRE
	02	01	03	CHANGE PROCEDURE
	02	01	05	PAY
	02	01	07	T4 YEAR-END
	02	03		HOURLY PAYROLL
	02	03	01	TIME REPORTIN
	02	03	03	CHANGE PROCEDURE
	02	03	05	PAY
	02	03	07	T4 YEAR-END
	02	05		TEMPORARY PAYROLL
	02	05	01	PAY
	02	05	03	T4 YEAR-END

SYSTEM DIRECTORY - SHORT FORM: 01 UPDATE CODE: A CURRENCY STATUS:

	SYSTEM	SUB-SYSTEM	PROCEDURE	NAME
1	>>> 01			STUDENT INFO SYSTEM
2	01	>>> 03		ADMISSIONS
3	01	03	>>> 01	OAC INPUT
4	01	>>> 05		REGISTRATION
5	01	05	>>> 01	UNDER GRADUATE REGISTR
6	01	05	>>> 03	LATE REGISTRATION
7	01	05	>>> 05	GRADUATE REGISTRATION
8	##	##	##	##########################
	##	##	##	##########################
	##	##	##	##########################

LEGEND:

UPDATE CODE:
A - ADD
C - CHANGE
D - DELETE
P - PURGE

CURRENCY STATUS
b - CURRENT
X - OBSOLETE
A - PROTOTYPE A
B - PROTOTYPE B
C - PROTOTYPE C
.....ETC.

```
SYSTEM DIRECTORY - LONG FORM: 02      .UPDATE CODE: A          CURRENCY STATUS:

                          >>>|SYSTEM.....|ID: 02|NAME: PAYROLL                 |
                             |SUB-SYSTEM.|       |                              |
                             |PROCEDURE..|       |                              |
                        |#: |DESCRIPTION:                                      |
                        |05 |THE SCOPE OF THIS SYSTEM INCLUDES THE PRODUCTION  |
                        |10 |OF PAY CHEQUES AND THE PAYROLL JOURNAL FOR ALL    |
                        |15 |SALARIED AND HOURLY-PAID EMPLOYEES,THE            |
                        |20 |WITHOLDING OF INCOME TAX, PENSION, AND OTHER      |
LEGEND:                 |25 |DEDUCTIONS FROM PAY, THE CONCEQUENT PROCESSING OF |
                        |30 |SUCH DEDUCTIONS ON A QUATERLY AND ANUAL BASIS.    |
UPDATE CODE:            |35 |THIS ENTAILS THE MANTENANCE OF APPROPRIATE        |
  A - ADD               |40 |EMPLOYEE RECORDS FROM THE POINT OF HIRING TO THE  |
  C - CHANGE            |45 |END OF THE YEAR IN WHICH THE EMPLOYEE TERMINATES, |
  E - EXTEND            |50 |IN WHICH ARE CARRIED THE CURRENT AND YEAR-TO-DATE |
CURRENCY STATUS:        |55 |PAY AND DEDUCTIONS DATA, AND ALL NECESSARY        |
  b - CURRENT           |60 |PERSONAL INFORMATION.                             |
  X - OBSOLETE          |65 |##################################################|
  A - PROTOTYPE A       |## |##################################################|
  B - PROTOTYPE B       |## |##################################################|
  C - PROTOTYPE C       |## |##################################################|
      .....ETC.
```

B I B L I O G R A P H Y

Anderson, R. J., The External Audit 1, Pitman Publishing, Toronto, 1977.

Cobol Version 5 Reference Manual, CDC Operating Systems, NOS1, NOS/BE1, Pub. No. 60497100.

Cobol Version 5 User's Guide, CDC Operating Systems: NOS1, NOS/BE1, Pub. No. 60497200.

Cyber Record Manager Version 1, 700 Record Manager Reference Manual, Pub. No. 60495700.

Intelligent Graphics 2647A Reference Manual, Hewlett Packard Company Part - No. 02647 - 90005, 1978.

Intelligent Terminal Basic Reference Manual, Hewlett Packard Company Part - No. 02647 - 90005, 1978.

Maginnis, James B., Fundamental ANSI Cobol Programming, Prentice-Hall, Inc., Englewood Cliff, New Jersey, 1975.

Martin, James, Computer Database Organization, 2nd. edition, Prentice-Hall, Inc., Englewood Cliff, New Jersey, 1977.

Masterson, J. S., Analytical Audit Charting by Computer Graphics, A Gross Feasibility Study, McMaster University, June 12, 1978.

McCracken, Daniel, D., A Simplified Guide to Structural Cobol Programming, John Wiley & Sons, Inc., 1976.

Skinner, R. M., Anderson, R. J., Analytical Auditing, Pitman Publishing Corporation (United States), Sir Isaac Pitman (Canada) Limited, Toronto, May 1966.