A STUDY OF CONTROL SYSTEMS FOR HIGHLY CONFIDENTIAL DATA ACCESSIBLE FROM TERMINALS AT REMOTE POINTS

A Thesis

Presented to

The Faculty of Graduate Studies and Research

The University of Manitoba

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computing Science

---

by
Rampersad Joe Babooram
May 1969

ACKNOWLEDGEMENTS

I would like to express my deep appreciation to Professor Bernard A. Hodson, Director, Institute for Computer Studies, for his supervision in the research and preparation of this thesis.

To the following people I also extend my thanks: Mr. Donald Costin, Lecturer, Institute for Computer Studies, for his many valuable suggestions; Mr. V. Dutton, Professor, Department of Civil Engineering, for reading the thesis, and Mr. David Kuva, Graduate Student, for his assistance throughout the project.

I wish to express my gratitude to Mr. John Howorth of the Canadian Credit Men's Association, for his technical advice and the financial assistance provided by his Association.

Finally, I extend my appreciation to the following organizations that provided information for the research: Project MAC (Massachusetts Institute of Technology), Honeywell Inc., Sanders Associates Inc., and the Burroughs Corporation.

ABSTRACT

Today's rapid growth of shared computer facilities is paralleled by the requirement for the protection of data and programs stored in the computer files. These shared facilities may contain data for several types of Information Systems such as Banks, Airlines, Hospital and Credit-Reporting. In this thesis, a solution is sought for the protection of highly sensitive and critical information stored in the files of a central computer and accessed by a network of terminals located at remote points.

The initial phase of the project involved a study of some existing control systems; the protection features obtainable from IBM's Operating System/360; and the various input-output devices adaptable for use as remote terminals were investigated for any contribution they might make towards security measures. Finally, by using some features from the foregoing, a new control system providing protection at the organization level was developed and tested.

The system identifies a user of a specific Information System (Bank, Hospital, Credit-Reporting or some other type) by the hardware address of the terminal. A particular terminal operator is recognized as an authorized official by a brief computer-terminal conversation. Users are permitted to have one or more terminals and those with multiple terminals may restrict each one, or subsets of these to access

only specific files. Each user is granted full access privileges to his own files but can obtain only selected information from the entire data bank.

The method was programmed for an IBM 360/Model 65 computer operating in a batch-processing mode and under control of the Operating System. The data used in simulating the method was obtained from a Business Credit Information System. The routines are written in the Fortran IV (G-level) and Assembler programming languages. Two IBM 2260 Display Stations with keyboard were used as remote terminals.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

I. THE SECURITY PHILOSOPHY

Computer technology is changing today from the classical single-computer, single-user category to a central computer shared among several users. This computer operates either in a time-sharing or in a batch-processing mode. In the time-sharing mode users are permitted to solve problems simultaneously while in the batch-processing mode each job is processed to completion before another is begun. Sharing of the computer is done through terminals housed locally or at remote points and linked to it by data-transmission lines.

A common problem arising out of these shared facilities is that of security - the confidential handling of private customer data. Solutions to the problem become more urgent as the number of these facilities and their users increase. This increase is a rapid one as reflected in the following statistics:[i] In 1965 there were only 500 terminals and 6 companies offering time-sharing services in the United States. Today, one well-known manufacturer alone serves more than 50,000 terminals with 70 companies offering service. The latter number was expected to increase to about 100 by the end of 1968.

---

[i] "Share and Share Alike?" - BARRON'S, A New York Financial Paper, 23rd September, 1968, edition.

A shared computer facility will have a large community of users, many of whom may be competitive commercially. The facility will be used for diverse applications, and sensitive data (such as payroll records) will be stored in the computer files. Some users may want to share their data. As soon as the system is shared, even though it may be used by only one user at a time, protection is required so that the company offering the service may guarantee a high level of data security.

Aside from the foregoing category of business data banks, another area where progress necessitates the assurance of privacy is the development of National Data Banks containing general information on citizens. Here the data will consist of private information on individuals, the individuals themselves having no knowledge of what is stored in the computer files and hence having no chance to rebut. Besides denying the individual his legal right to question the validity of information which may be used against him in such applications as those for employment and loans, these data banks can be used by organized crime (i) to search files for embarrassing data on people for blackmail purposes, and (ii) to identify people who might easily be lead into compromising situations.

Protection of a customer's private data is the responsibility of the company offering the shared service. Upon failing to receive the

protection promised one then seeks legal action. The impact of the computer is a recent one, thus, unlike fields such as real estate and shipping, there is no special branch of the law dealing with computers and data-processing. However, computers and data-processing may be accommodated in the general law - a law which deals with concepts like negligence, contracts and evidence. The most one can do then is to follow the law in closely related areas and try to reason by analogy for computers. Problems arising out of the use of a computer can be very complex as usually there will be no witnesses and any evidence produced will have to be circumstancial. Such evidence will often be generated by the computer itself. Examples of such problems can be found in the ever increasing use of computers in banking and credit applications. A cheque mistakenly dishonoured for lack of funds could result in the bank facing a suit for slander of credit. A similar case can arise from making an inaccurate credit recommendation.

What guarantee does a customer receive from the computer manufacturer? Some products are sold with a built-in warranty which is like a promise or a guarantee. This may be implicitly or explicitly stated: food is sold with the implicit guarantee that it is at least non-poisonous. If such a promise is made in the case of a computer, it is the usual sales-promotion one regarding its speed, memory space, access-time,* or some other feature. If a guarantee about reliability

---

*The reader is referred to the glossary for an explanation of any term designated by this symbol.

is made, it refers to breakdowns and is of little protection to the buyer if his secret data and programs leak to competitors.

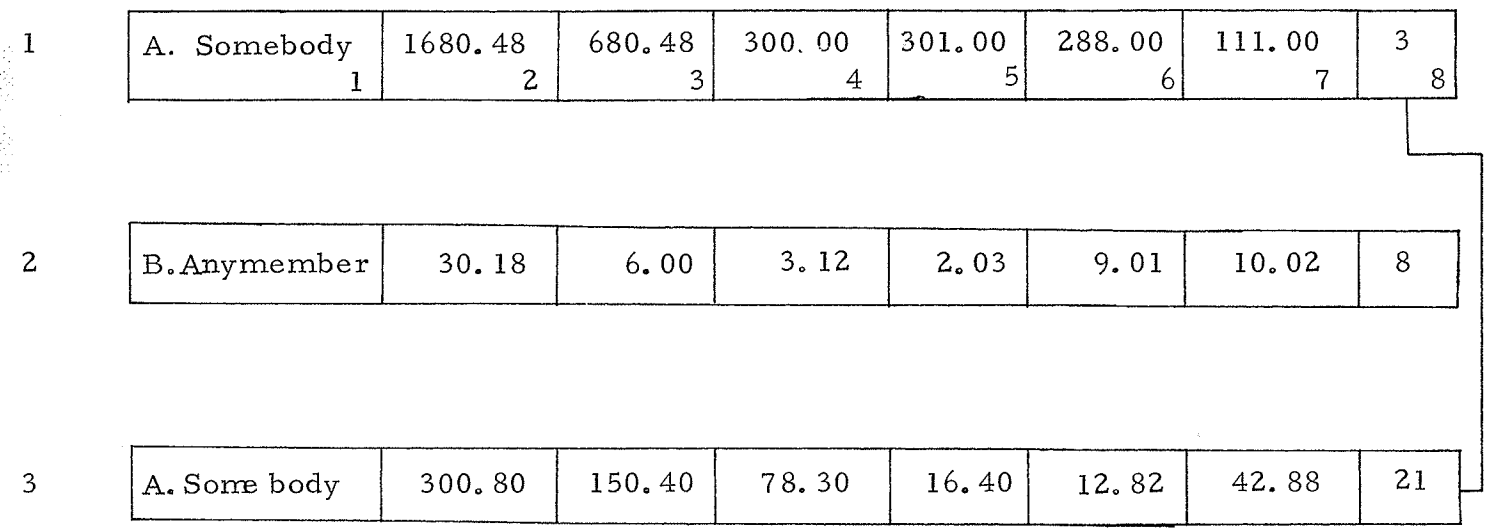
In some recent systems measures have been taken for ensuring file and program security. This is attributable partly to insight by systems designers and, to a greater extent, to pressure from public and governmental sources. The next section discusses the security philosophy as applied to credit data.

II. SECURITY OF CREDIT DATA

The data to be considered in this study was obtained from a Credit Bureau and consists of the Accounts Receivable records of several businesses (called members) throughout Canada. Generally, an Accounts Receivable record will contain the name of the debtor, the total amount he owes, the current amount* owing, and the number of days (30, 60, 90, 120) payment is overdue (fields 1 to 7 of Figure 1). These are received in different formats* and on various storage media such as magnetic tape, paper-tape, punched-cards and hard copy* documents. As the storage and updating of the data is not the subject of this study, it suffices here to state that said Accounts Receivable are stored in one massive file in the same format as they are received from the member, and that all the records of the same debtor are linked by placing the file-address* of his next record into the link field (field 8 of Figure 1) of his preceding one. This link field

FIGURE 1

LINKAGE OF TWO
RECORDS IN THE ACCOUNTS RECEIVABLE FILE



Legend

1. Name of creditor
2. Total amount owing
3. Current amount owing
4. Amount 30-60 days overdue
5. Amount 61-90 days overdue
6. Amount 91-120 days overdue
7. Amount past 120 days overdue
8. Link

is one which can be added to each record by the Bureau before storage in the file or can be automatically created by the software.* The linkage of two records in the Accounts Receivable file is shown in Figure 1 and will be discussed in the upcoming paragraph.

This file can be used by members to store their Accounts Receivable with updating taking place from their own terminals or by the Credit Bureau. It is used by the Bureau specifically for the preparation of credit reports requested by members on selected businesses. It is in this application that the link field fulfills its purposes. As soon as the program finds the first record of the subject (the business on which the report was requested) in the file, it uses the address specified in this field to locate the next record of that subject. The link field of this second record gives the location of the debtor's third record in the file. Thus this field allows the program to quickly jump through the huge Accounts Receivable file and efficiently pick out every record of the subject to form a credit report. If no information is available in the file for that subject, an alternative procedure[i] is followed. A report to a member consists of the type of business from which his client made purchases, that is hardware, grocery, lumber, drug, or some other type, the total amount he owes to each of his creditors, the

---

[i] In this procedure letters are dispatched requesting information from non-members and the credit report is constructed from their replies.

current amount owing, and finally a breakdown of his credit in terms of the number of days payment is overdue. A more complete report containing the names of the subject's creditors will be available only to the Credit Bureau. Figures 2(a) and 2(b) show a sample report as prepared for a members and for the Bureau respectively.

Since such credit information is very sensitive and since the completeness of the credit report is dependent upon the amount of data kept in the file, it is essential that such data be adequately protected so that members will have no fear of releasing it to the Credit Bureau. For example, if only a percentage (say 60 percent) of the Bureau's members were to furnish their Accounts Receivable records, then a credit report formed from the file may be incomplete since some of the 40 percent of the members with-holding their records may be creditors of that business. The report may therefore not reflect a true picture of the subject's credit standing.

The adverse effects of one member getting access to another member's data are twofold. Firstly, he can persuade the member's clients to purchase from him by offering better service and lower prices. Secondly, the records in the file can be altered so that credit reports are incorrect. Both examples illustrate unethical business practices.

Figure 3 provides a very basic representation of a central computer shared by many remotely-located terminals connected to it by communication lines.

FIGURE 2

(a) SAMPLE OF A CREDIT REPORT PREPARED FOR A MEMBER

[SUBJECT]

Business Type	Total Owing	Current Owing	Days Due			
			30-60	61-90	91-120	Past 120
A	1381.33	1260.59	20.74	100.00		
AA	1322.46	841.64	409.55	50.01	21.26	
D	8941.11	7026.34	1123.46	791.31		

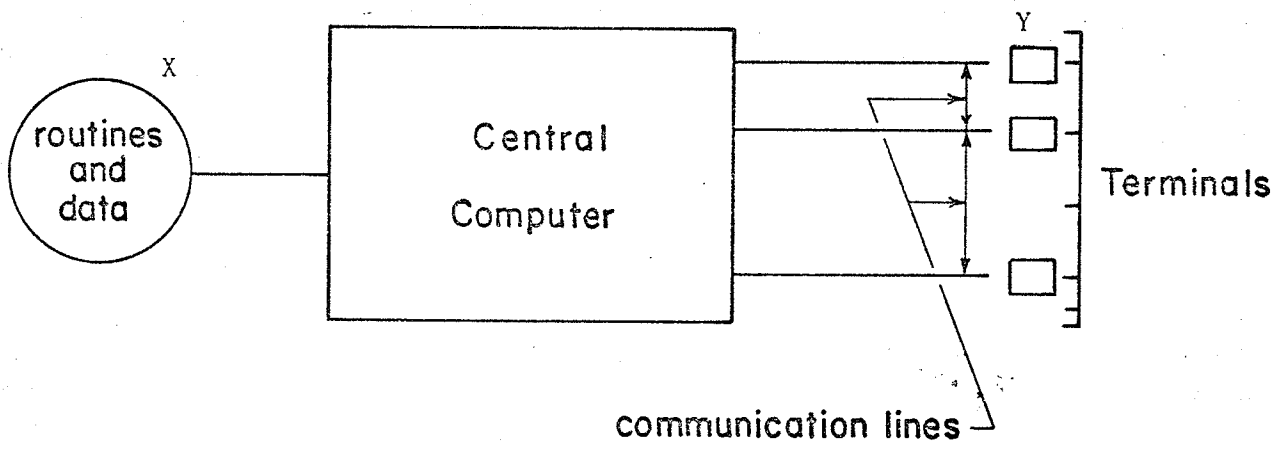
(b) SAMPLE OF A CREDIT REPORT PREPARED FOR THE BUREAU

[SUBJECT]

Creditor's Name	Business Type	Total Owing	Current Owing	Days Due			
				30-60	61-90	91-120	Past 120
A. G. Eletorn	A	1381.33	1260.59	20.74	100.00		
Emcoyee Ltd	AA	1322.46	841.64	409.55	50.01	21.26	
Weybrne Wpg	D	8941.11	7026.34	1123.46	791.31		

FIGURE 3

A SHARED COMPUTER FACILITY



Stored in the computer's files at X will be system's and user's routines, and data. The user's routines are the programs peculiar to a particular user, for example, a frequently used one to calculate square roots. The system's routines are those programs which enable the computer to do the necessary housekeeping* and to answer the requests made by the users.

The controls required at the points labelled X and Y in Figure 3 are:

(i) Protection of routines and data at X:

Both the user's and system's routines are vital for the successful operation of the facility and hence unauthorized users must be prevented from reading or writing into them. Deliberate tampering with these routines can have catastrophic results. For example, a user getting into the accounting routine of the system may alter it so as to cause incorrect charges to other users, or he may change the system and file protection routines so that they may function in a destructive fashion.

(ii) Identification of the Terminal and the Operator at Y.

Terminals may be located in a member's business office, stock-room or printing-room, and their use restricted to only certain personnel. Consequently, the central computer must identify the terminal operator as a valid one, and should make note of every

---

*See Glossary

illegal attempt to use the system. It must also keep track of every terminal in the network so that replies are sent to, and read from the correct units. Failure to do this can result in members' confidential data being displayed at terminals other than their own.

III. FORMAT OF THESIS

The initial phase of the project consisted of a study of control methods in existing systems. The results of this study are presented in Chapter II. Protection features offered by IBM's Operating System/360 were investigated and these are included in Chapter III. Chapter IV discusses the input-output mechanisms with particular attention to the device address concept along with several terminals suitable for use in a Credit Information System. Incorporating some features from IBM's Operating System/360 and from the various remote-access utilities, a new method is described in Chapter V for controlling access to confidential data. The discussion uses credit data as a working example. Creation of the necessary files and routines for the implementation of the system are discussed in Chapter VI. The thesis is concluded in Chapter VII.

CHAPTER II

CURRENT DEVELOPMENTS ON FILE PROTECTION

The main features of some existing control systems varying in their degree of sophistication, are now presented in illustrating recent developments in the area of data and program protection. These are selected from journals and from replies by companies and manufacturers contacted. They establish the necessary groundwork for later progress in the project.

I. HONEYWELL HOSPITAL COMPUTER SHARING SYSTEMS

The Honeywell Hospital Computer Sharing Systems (HCSS) consist of one or more terminal devices located in the business offices of several remote hospitals. These terminals share access to confidential patient files at a central computer facility. In its present state, hardware,* program and physical controls are applied with the emphasis on keeping each hospital within the boundaries of its own files. Attention is paid more to accidental access rather than intentional snooping or destruction. Terminals are protected physically from unauthorized users.

Access from a terminal is controlled by validity checks in a resident communications program. Each hospital uses a unique code number as a header information for transmission. A terminal is

further identified by its internal line number which is a fixed or hard-wire connection established at installation time. Hence, in HCSS access is controlled by:

(i) a terminal's functions as defined in the software and

(ii) program comparison of hospital and line number before access is permitted for the defined function.

As the system expands, Honeywell proposes to extend access controls. Badge readers may be used with priority codes available for program checks, or appended by manual data entry as a further program check key. If required, these program checks can be extended to include several layers of conversational identification. In this latter type of identification, the computer will ask the terminal operator personal questions, the answers to which will most likely not be known by an impostor. Some such questions are his father's date of birth, the amount of money he had in his savings account on a particular day and the date or time of some significant event in his life. The file protection routines will verify the terminal operator's identify by checking his replies against those acceptable answers stored in one of the computer files.

II. PROTECTION OF DATA AND PROGRAMS IN AN INFORMATION PROCESSING UTILITY

Perhaps the most elaborate protection scheme being developed today is the one for an Information Processing Utility (IPU) at the

Massachusetts Institute of Technology (MIT). In this utility there are a large number of users, many of whom may be using the system simultaneously. The system will operate in a multi-programming* mode and will have more than one Central Processing Unit. The method for protection utilizes segment addressing hardware (Page 15) and supporting software. It provides for protection of user's and system's routines, in both main and auxiliary storage, during the execution of a program and for the sharing of common data bases among all users or a selected group of users.

Before discussing the proposal for protection of data and programs in the IPU (Method 2), it is worthwile to examine a common method used in the past (Method 1) and to notice its shortcomings as the proposal will try to eradicate these.

Method I. In this method a program executes in one of two modes (master or slave) as specified by a switch, called the mode switch. In the master mode any instruction can be executed including the privileged instructions*; whereas in the slave mode, any attempt to execute a privileged instruction causes an interrupt. These privileged instructions prevent users from accessing information written on various storage media thus protecting inactive[i] information in the system.

While the mode switch will prevent access to inactive information

---

*See Glossary

[i]This is information which is not to be used by the program in execution.

on the storage device, it cannot protect information which is active and resident in working memory. This is the function of a register known as the memory bounds register. It partitions memory into two parts, one of which may not be accessed when working in slave mode.

The foregoing is an all-or-nothing type of protection. If a program has any access, it has all; that is, it can read, write or execute and if it has privileges to any data, it has privileges to the entire data bank. This is insufficient protection for an IPU since there may be users with limited access, that is, a user may be able to read some files but not write into them, and with privileges to only a fixed set of data.

Method 2. A higher degree of protection is available through segment addressing hardware. This type of hardware allows memory to be divided into many program and data segments; a segment being a contiguous block of words whose length may vary during the execution of a program. Each word in memory is addressed by an ordered pair of integers (S, W); S being the segment number, and W the word number in that segment (Figure 2 A).

Figure 2 A

A segment descriptor is formed for each segment to be used by any program. This descriptor contains the location of the beginning of

each segment, the length of the segments and the access indicator. This indicator specifies whether the segment may be addressed in slave mode, written or executed. The format of a segment descriptor is:

beginning of segment	length	access indicator

Figure 2 B

The segment descriptors for a program are placed in an area in memory called the descriptor segment. The location of the descriptor segment is held in a hardware register* referred to as the descriptor base register. Thus, when a program begins execution, the descriptor base register points to the location of the descriptor segment for that program, and the segment descriptors therein point to the segments in memory to be used by the program, and the manner in which they may be accessed.

Suppose now that more than one program requires the use of the same segment in memory. How does the sharing take place? This is implemented by placing each segment to execute in one of a set of ordered disjoint segments called rings (Figure 2 C).

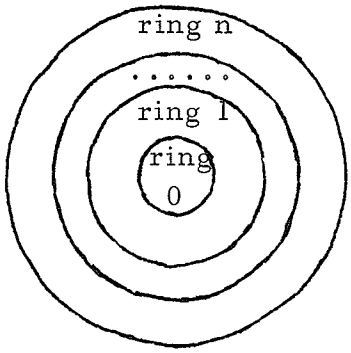


Figure 2 C

Now a program segment executing in a lower numbered ring can access a segment in a higher numbered ring, subject to a software check verifying the transfer. Transfer from a higher numbered to a lower numbered ring is never permitted. For example, a program segment executing in ring 1 can access segments in ring 3, but a program segment executing in ring 3 cannot access segments in ring 1.

The segment descriptors and word addresses are augmented to contain this ring number as shown in Figures 2 D and 2 E.

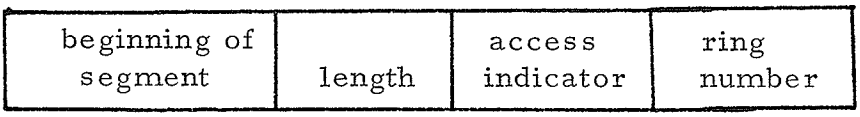


Figure 2 D

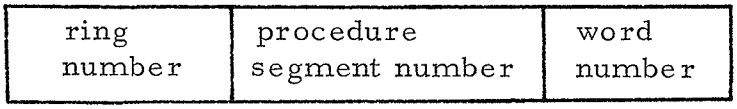


Figure 2 E

This latter method provides the degree of protection required in the IPU. It can completely isolate one program from another. Hence, a user can deny any access whatsoever to his segments. On the other hand, he can permit controlled access to any of his segments with different access privileges for different users.

III. PROJECT MAC

MAC, Machine Aided Cognition or Multiple Access Computer, was developed at MIT and became operational in November, 1963. It has since become the best known of all computer utilities, and is considered the birthplace of such a utility. MAC is a general-purpose* computer system operating in the time-sharing mode, and accommodates a host of remotely located users.

The storage protection scheme used in MAC involves additional equipment in the form of a lower boundary register and an upper boundary register. Special instructions set these registers and place the Central Processing Unit in the protection mode; a mode in which the address part of each instruction is compared to the registers before being executed. If this address is outside the limits specified by the registers, an interrupt occurs and a subroutine intervenes. Both registers are set using one instruction but the program does not enter the protection mode until an additional instruction is executed. This instruction gives the control program the opportunity to transfer control to a problem program after setting the boundary registers.

IV. UNITED PLANNING ORGANIZATION

United Planning Organization (UPO), a Washington, D.C. anti-poverty agency, is developing a data file containing information

on several thousand disadvantaged residents. The Organization's executive director claims that privacy will be assured by separating the name of the individual from his data; by not providing any data on individuals but only aggregate statistics; and by placing the management of the data files in the hands of trustees. The latter move gives the individuals the right to take legal actions if they feel that their right of privacy has been violated.

This method for protection has been severely criticized at a Senate Judiciary Subcommittee hearing in February, 1968. It was contended that anyone free to use the files can find a person's record by using the individual's age, street address, and occupation as clues; and hence the individual's privacy cannot be secure.

In closing the section, the significant difference between UPO's method and the one used in the preparation of a credit report for a member of the Bureau (Section II, Chapter I) is noted. While in the former the separation of name and data takes place in storage; in the latter it is done when the credit report is prepared; hence the security of the data file is left to the file protection system. The only information contained in the report about the creditor is his business type; and this alone does not give the member enough grounds to identify the creditor.

V. CONCLUSION

While the need for the protection of confidential data is recognized as one of the essential requirements in the establishment of data banks, those responsible for the formation of these banks tend to shy away or ignore the concern. This is true of many new systems, an example of which is found in the formation of a National Data Bank in the United States. Here, as a result of public complaints, the House Committee on Government Operations has intervened and recommended that no new work is to be done until privacy protection is guaranteed. It also charged the computer community with the requirement of guaranteeing Americans that their privacy won't be destroyed by any data system which permits a retrieval of information in a form that could relate the data to individuals.

In summarizing the state of affairs, the following words of Paul Armer, AFIPS[i] Vice-President, to the aforementioned Committee, are appropriate:[7]

> "The forces of the market place are apt to have little impact in the near future of improving the state of privacy in our society . . . the market mechanism works very imperfectly in such areas and needs prodding . . . privacy lacks an organized constituency."

---

[i] AFIPS, The American Federation of Information Processing Societies, was formed in May 1961 and holds conferences in the Spring and Fall of each year.

CHAPTER 111

DATA SET PROTECTION

FEATURES OF IBM'S OPERATING SYSTEM / 360

A data set is a named, organized collection of logically related records,* and is not restricted to a specific type, purpose or storage medium. It may be, for example, a source program or a file of data records. A data set to be stored in computer files is placed on a volume (a standard unit of auxiliary storage). Examples of volumes are a reel of magnetic tape, a disc pack and a drum. This chapter examines (i) the protection features available through the facilities provided by the Operating System for the organization, storage, identification and retrieval of these data sets, and (ii) the PASSWORD protection feature of the Operating System.

For every data set to be stored in the computer files certain characteristics (parameters) about them must be specified. These enable the Operating System to perform the required housekeeping such as allocating the data set to a volume, allocating space on the volume and updating the table of contents for the volume. The parameters are defined in a statement called the data definition statement (DD Statement), of the Job Control Language (JCL).* Having stored a data set under a set of parameters, a program can retrieve the data only when these are correctly specified. The parameters relevant

to this topic are presented as the discussion proceeds and in the end are incorporated into a complete DD statement. The upper-case* letters enclosed in brackets are the abbreviations required by the Operating System for the parameters in the DD statement.

Data Set Organization (DSORG)

Operating System/360 data sets can be organized in four ways:[10]

Sequential (PS). This is the familiar tape-like file structure, in which records are placed in physical rather than logical sequence. Thus, given one record, the location of the next record is determined by its physical position in the data set. The sequential organization is used for all magnetic tapes, and may be selected for direct-access devices. Punched-tape, punched cards, and printed outputs are considered to be sequentially organized.

Indexed Sequential (IS). Records are arranged in collating sequence according to a key that is a part of every record, on a track of a direct-access volume. In addition, a separate index or set of indices maintained by the system gives the location of certain principal records. This permits direct as well as sequential access to every record.

Direct (DA). This organization is available for data sets on direct-access volumes. The records within the data set may be organized in any manner the programmer chooses. All space allocated

to the data set is available for data records. No space is required for indices. Records are stored and retrieved directly with addressing specified by the programmer.

Partitioned (PO). This structure has characteristics of both the sequential and the indexed sequential organizations. Independent groups of sequentially organized data records, each called a member, are held in direct-access storage. Each member has a name stored in a directory that is a part of the data set, and contains the location of the member's starting point. Partitioned data sets are generally used to store programs. As a result, they are often referred to as libraries.

The data set organization is specified in the Data Control Block (DCB) parameter of the DD Statement. In addition, this parameter contains the following subparameters about the data:

Block Size (BLKSIZE). A block is a group of records. This subparameter specifies the length, in bytes, of a block.

Record Format (RECFM). Records in a data set are specified as fixed-length (F), variable-length (V), or undefined-length (U). Blocked records are specified as being fixed-blocked (FB) or variable-blocked (VB).

Record Length (LRECL). The record length specifies the length, in bytes, of each record in the data set. If the records are variable in length, the maximum record length must be specified.

The format of the DCB parameter is

DCB = (list of attributes)

where the list of attributes is replaced by the subparameters which are separated by commas.

Figure 3 A gives an example of the DCB parameter.

```
DCB = (DSORG=PS, RECFM=FB, LRECL= 120, BLKSIZE=600)
```

Figure 3 A

It indicates to the Operating System that the data set is sequentially organized and consists of fixed-blocked records, each block being 600 bytes in length. The record length is 120 bytes, and thus each block contains 5 records.

Data Set Identification (DSNAME)

The Operating System requires a unique name for each data set created on the same volume. This name is written into the table of contents of the volume, and any subsequent attempts to create a data set under that name would not be successful.

The data set name consists of a name or a name preceded by qualifiers which are separated by periods. The rules for its formation are as follows:

(i) the maximum number of characters in any component of the name is 8,

(ii) the maximum length of the name, including periods, is 44,

(iii) the first character of each component must be alphabetic, and the remaining characters alphabetic or numeric, or a combination of both.

(iv) the last character of the data set name cannot be a period.

The format of the DSNAME parameter is

DSNAME = data set name

Figure 3 B gives the entry in the DD statement for a data set with name OIL CO. ACCTIDIV. PAYROLL. STORES 1.

```
DSNAME=OIL CO. ACCTDIV. PAYROLL. STORES 1
```

Figure 3 B

Data Set Storage (VOLUME)

The Operating System requires information about the volume on which a data set is to be stored or on which a data set resides. This information is specified in the VOLUME parameter of the DD statement.

A volume can be requested in either a specific or nonspecific manner. Nonspecific references can be made only for new data sets, in which the Operating System assigns a suitable volume. Specific references are made in the following two ways:

(i) by specifying the serial number of the volume to be used. This number has a maximum length of 6 characters and may be either alphabetic, numeric or alphanumeric. The

format of this parameter is

VOLUME = SER = (ser#, - - - - -, ser#)

where ser# is the serial number of the volume.

If only one volume is involved, the parameters can be specified in the form

VOLUME = SER = ser#

The example in Figure 3 C makes a specific reference for the volume with serial number UM1406

```
VOLUME = SER =UM1406
```

Figure 3 C

(ii) by using the same volume as assigned to an earlier catalogued data set.

In this case the format of the parameter is

VOLUME = REF = DSNAME

Where DSNAME is replaced by the data set name of the previously catalogued data set. The example in Figure 3 D tells the Operating System to use the same volume on which the data set with name ONE. MONTH resides.

```
VOLUME = REF = ONE. MONTH
```

Figure 3 D

The protection of a data set by the preceding parameters depends upon these being unknown to anyone but the programmer creating it. However, it is possible for a user in attempting to create a data set, to find that he has accidentally chosen a data set name which already exists in the table of contents for the volume he specified. The user is now free to destroy or to deform that data set in whichever way he pleases. The DCB, DSNAME and VOLUME parameters, are, therefore not sufficient to ensure data security.

Expiry Date or Retention Period (LABEL)

A further measure of protection is available by specifying either the expiry date or the retention period subparameter in the LABEL parameter of the DD Statement.

The expiry date specifies the year and day after which the data set can be deleted or opened for any type of processing; and the retention period specifies the length of time, in days, that the data set is to be retained. These are used for protection against writing into data sets when not desired and are primarily for preventing the user from destroying his own data sets rather than those of others.

The format of the expiry date subparameter is

LABEL = EXPDT = yyddd

where yy is the 2-digit year number and ddd is the 3-digit day number after which the data set can be considered expired.

The format of the retention period subparameter is

LABEL = RETPD = nnnn

where nnnn is the number of days the data is to be retained.

PASSWORD Protection

PASSWORD protection is the name given to the data set security facility provided by Operating System/360 for the control of confidential data. This facility entails all of the preceding features, plus additional checks before access is permitted.

To implement this protection feature, a data set (named PASSWORD), consisting of records (Figure 4) that associate the names of protected data sets with the password designated for each data set must be created. Figure 5 gives an example of this association in which the password for the data set named UOFM.COMPSC.LAB601.SECT1 is UMSEC1. An attempt to use this data set is met with a request by the Operating System for the password. The request is directed to the console typewriter* and the operator must reply with the correct password before the data set can be used. If on the first reply the password is incorrect, the Operating System makes the request again, and thus allows the operator a second chance to produce the correct one. Failure to do so in two tries results in a termination of the job.

The PASSWORD data set is a sequential one and is placed on the system residence volume.* Since protecting a newly created data set

FIGURE 4

FORMAT OF A RECORD IN THE PASSWORD DATA SET

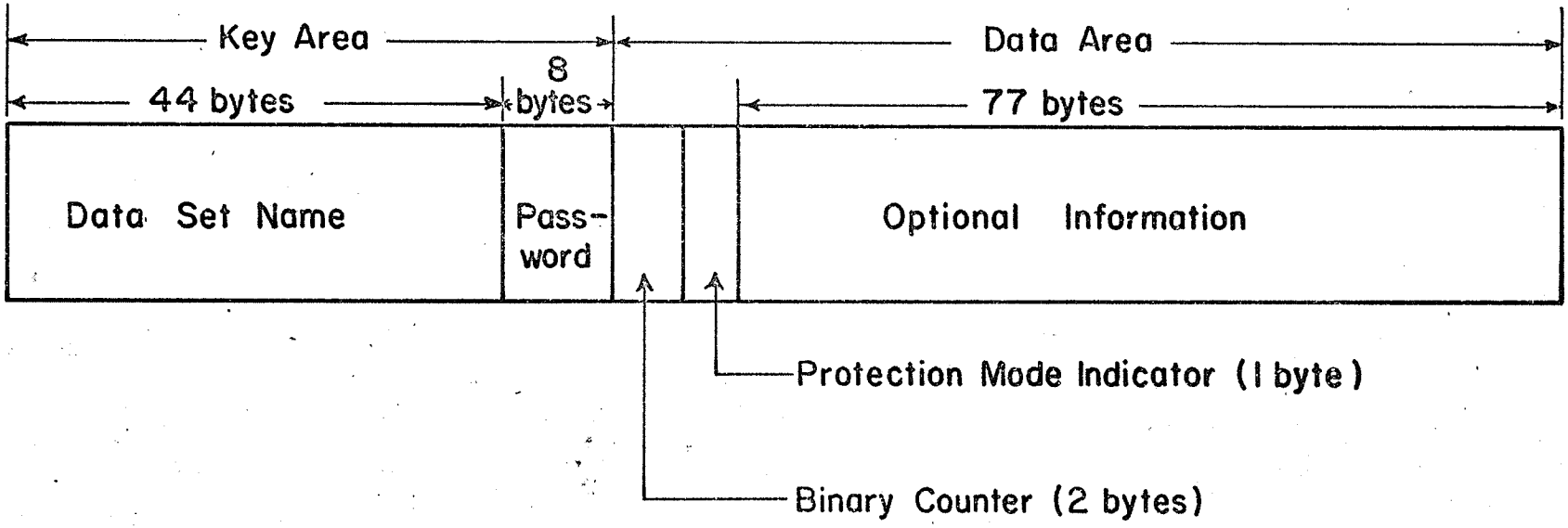
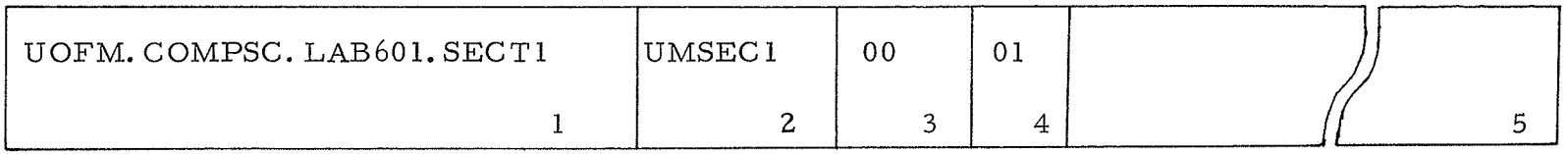


FIGURE 5

A RECORD IN THE PASSWORD DATA SET



Legend

1 Data Set Name
2 Password
3 Binary Counter
4 Protection Mode Indicator
5 Optional Information

involves an addition of a record to the PASSWORD data set, a routine for maintaining the said data set must be written and stored in the system's linkage editor library* or in the programmer's own library. A data set to be placed under protection has the following entry in the LABEL parameter of the DD statement:

LABEL = (, , PASSWORD)

The records of the PASSWORD data set are formatted into a key area and a data area. The contents of these areas are as follows:

Key Area (52 bytes). This area is divided into the following sections:

(a) The Data Set Name (44 bytes). The data set name must be left justified, and, if necessary, padded with blanks (X'40')[i] on the right. The rules for the formation of this name have been discussed previously.

(b) Password (8 bytes). This is the password for the data set under protection. It must be left justified, can be up to 8 characters long, and padded with blanks on the right if necessary.

Data Area (80 bytes). The divisions of the data area are:

(a) Binary Counter (2 bytes). This counter is incremented by 1 each time the data set is successfully accessed (except for scratching or renaming the data set, called SCRATCH and RENAME functions

---

[i] Hexadecimal notation

respectively). When the record is built, this counter must be given a starting value.

(b) Protection Mode Indicator (1 byte). This indicator must be set when the record is built, and indicates whether the data set is to be read only, or both to be read and to be written. The value is either:

zero (X'00') if the data is to be read only, or

one (X'01') if the data is to be read and to be written.

(c) Optional Information (77 bytes). This field may be used to hold any additional information the installation desires. Such information may be the date of password change, or the date of counter reset.

The PASSWORD data set itself can be protected by including a record for it in the data set. It can then be accessed only when the computer operator enters the correct password (except by the Operating System routines that use the data set).

The action of this protection feature to the termination of processing, volume switching,* data set concatenation,* SCRATCH and RENAME functions, and counter maintenance are as follows:

(i) Termination of Processing

Processing is terminated when:

(a) the operator cannot supply the correct password for the data set being opened,

(b) a password does not exist in the PASSWORD data set for the protected data set being opened,

(c) the protection mode indicator setting in the password record and the method of input-output processing specified in the open routine do not agree,

(d) there is a mis-match in the data set names for a data set involved in a volume switching operation.

(ii) Volume Switching

The Operating System end-of-volume routine does not require a password for a data set involved in a volume switch.

(iii) Data Set Concatenation

A password is requested for every protected data set that is involved in a concatenation of data sets, regardless of whether the other sets are protected.

(iv) SCRATCH and RENAME functions

The protection feature issues a request for the password whenever an attempt is made to scratch or rename a protected data set.

(v) Counter Maintenance

The Operating System increases the binary counter in the password records, but no indication of overflow is given (overflow after 67,535 accesses). A counter maintenance routine to check and, if necessary, to reset this counter must be provided.

Remarks on PASSWORD Protection

This protection feature has two main disadvantages:

1. The passwords of all the data sets has to be available to the computer operator and anyone in dire need of these may try to influence

him into collaboration. Thus, the security of the data sets depends to a great extent upon the honesty and will power of the computer operator.

2. It is unsuitable for use in a computing system in which many remotely located terminals access the protected data sets. In such an environment much operator time will be spent, or wasted, entering passwords from the console.

Examples

This chapter is concluded by including two examples enhancing the features presented in the discussion. Example 1 illustrates a sample record in the PASSWORD data set, and Example 2 incorporates the DCB, DSNAME, VOLUME and LABEL parameters into a DD statement.

Example 1

Figure 5 (Page 30) shows the record in the PASSWORD data set associated with the data set named UOFM.COMPSC.LAB601.SECT1. The password is UMSEC1. Whenever an attempt is made to access the data set, the operator must enter UMSEC1 in reply to the request by the Operating System for the password. Both the request and the reply are made through the console typewriter. The binary counter is initially set to 0, and is incremented by 1 each time the data set is successfully used. The setting of the protection mode indicator is 1 which indicates

that the data set may be read only or be read and be written. The optional information field is left blank.

Example 2

Figure 6 gives the card format of a complete DD Statement. The DCB, DSNAME, VOLUME and LABEL parameters are the ones of concern to this presentation. The subparameters specified for the first three are the same as used in previous examples (Pages 24-26). The LABEL parameter specifies that the data set is to be password protected, and that it should be retained until the 100th day of the 69th year.

Although the objective of this chapter has not been to discuss the Job Control Language, the said JCL nevertheless cannot be divorced from the discussion. Hence, the following rules for forming the DD Statement are noted:

(i) each of the first two columns of each card must be occupied by a slash (/),

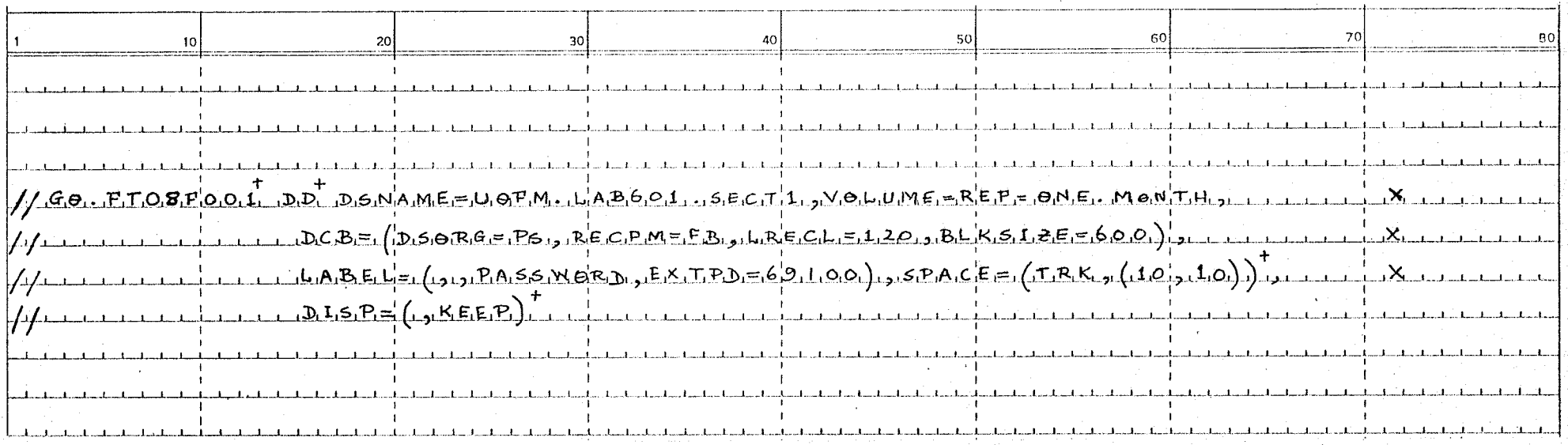
(ii) the parameters starting from the DSNAME and onwards (a) can be arranged in any order, and (b) cannot have imbedded blanks,

(iii) when a statement is to be continued on another card, a character must be punched in column 72. An 'X' is used in the example,

(iv) entries on a continuation card must begin in column 16.

FIGURE 6

A DATA DEFINITION STATEMENT



+ Irrelevant to discussion

0 Numeric zero

θ Alphabetic

CHAPTER IV

INPUT-OUTPUT CONSIDERATIONS

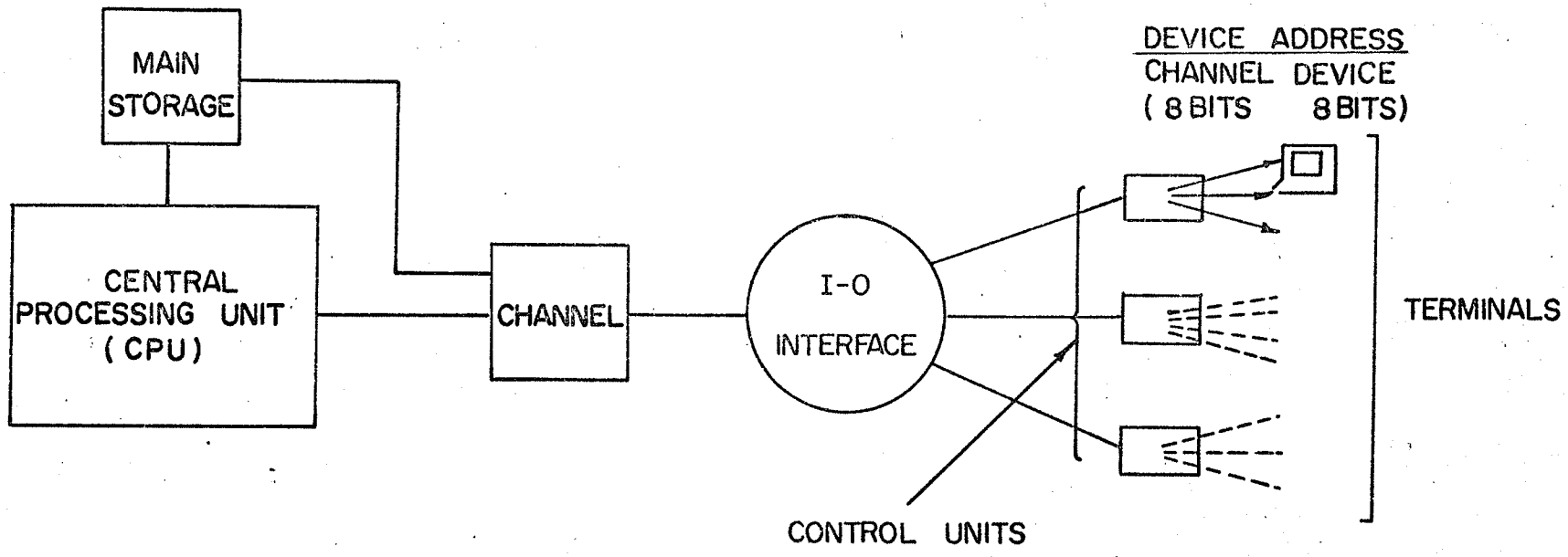
I. INPUT-OUTPUT MECHANISM

The transfer of information to and from main storage is referred to as an input-output operation. Each operation involves an input-output device. A computer facility shared by many remotely located users will have a large number of these devices connected to the Central Processing Unit by data channels, input-output interfaces, control units and telecommunication lines. This section discusses the mechanism by which the Central Processing Unit controls the numerous input-output devices with the objective of illustrating (i) the uniqueness property of a device's hardware address, and (ii) the completely operator independent manner by which the Central Processing Unit recognizes this address. Both of these properties make the address of a device an excellent factor for use in a control system.

It is necessary to establish some groundwork before the input-output mechanism can be discussed. This discussion involves (i) the functions of the units connecting the main storage, the Central Processing Unit and the terminals (Figure 7), and (ii) the hardware address associated with an input-output device (device address).

FIGURE 7

MAIN UNITS CONNECTING TERMINALS AND CPU



The functions of the units shown in Figure 7 are as follows:

(i) Main Storage

Main storage is usually the fastest storage device of a computer, and the one from which instructions are executed.

(ii) The Central Processing Unit (CPU)

The CPU is the unit of a computing system that contains the circuits that control and perform the execution of instructions.

(iii) Channel

A channel may be regarded as a small independent computer that responds to its own set of commands. Each channel has a program in main storage. The execution of these channel programs is initiated by the Operating System.

Channels direct the flow of information between main storage and the input-output devices, thus relieving the CPU of performing input-output operations. Consequently, processing and input-output operations can occur concurrently.

There are two types of channels, selector and multiplexor. The multiplexor channel may be divided into subchannels. Selector channels are used for the attachment of high speed devices while multiplexor channels are intended primarily for low speed devices such as card-readers and line printers.

(iv) Input-Output Interface

The input-output interface is the standard connection through

which the control units are attached to the channels. It consists of nine one-way lines for input and nine lines for output to accommodate one byte of data including a parity bit. It also provides the standard set of signals from the channel to which all control units respond.

(v) Control Units

The control units provide the logical capabilities necessary to operate and control the input-output devices. Their functions are:

(a) to accept and decode control signals from the channel for the input-output devices,

(b) to control the timing of data over the Input-Output Interface,

(c) to provide indications concerning the status of a device.

Device Address. Each input-output device has an address which is a 16-bit number consisting of two sectors; a channel part (8 bits) and a device part (8 bits) (Figure 4A). Though the 8-bits of the channel address field will allow for identifying up to 256 channels, only addresses in the range of 0-6 can be used since System 360 permits a maximum attachment of 7 channels (6 selector channels and a multiplexor). Any number in the range of 0-255 can be used as the device address. In the case of a multiplexor channel, the address identifies the sub-channel, control unit and device address.

The device address is assigned at installation time and normally remains fixed thereafter.

The diagram (Figure 4A) shows the channel address part and the device address part, in binary notation, of an input-output device which has a decimal value address of 201.

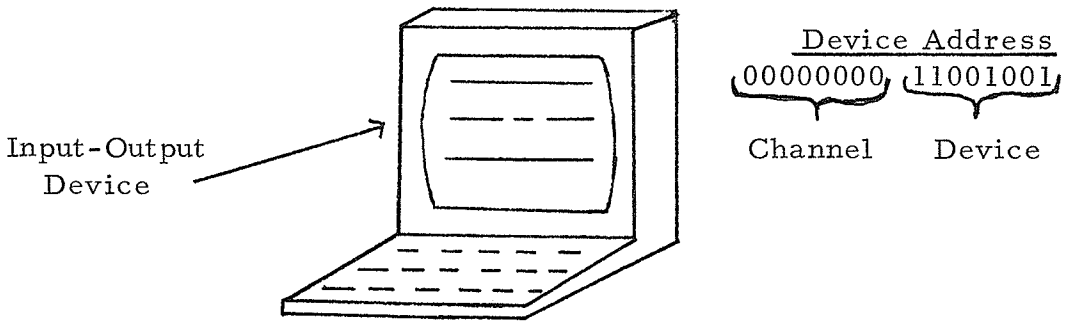


Figure 4A

The Mechanism. The operation of these units is related in the following way - the CPU drives the channel which in turn drives the control units, which operate the input-output devices.

The CPU communicates with the channel by the four instructions:

(a) START I/O - initiates an input-output operation,

(b) TEST I/O - tests the state (busy, available, etc.) of the addressed channel, subchannel and input-output device,

(c) HALT I/O - terminates a channel operation,

(d) TEST CHANNEL - indicates the state of the addressed channel.

The first three instructions address a channel and a device on that channel, whereas the fourth addresses only a channel.

When one of these instructions is passed to the channel, the latter fetches from main storage the channel program [12] required to carry out the job. The six commands available in a channel program are as follows:

(a) Read - causes data to be read from the selected input-output device and defines the area to be used in main storage,

(b) Write - causes data to be written to a selected device and specifies the area in main storage to be written,

(c) Read Backward - causes a read operation in which the characters are read backwards from the device and placed in descending main storage locations,

(d) Control - this command contains control information(called) orders) required to control the selected device. Typical orders are the rewinding of a tape, loading a tape reel and skipping a line on the printer,

(e) Sense - the sense command specifies the beginning main storage location to which sense information is transferred from the selected control unit. Sense information contains the condition of a selected device such as the stacker full condition of a card-reader,

(f) Transfer in Channel - specifies the location in main storage of the next command to be executed.

When the CPU is running a program and receives an input-output request, it initiates the operation by issuing an instruction to the appropriate channel. The channel locates the device specified in the instruction by sending the address of the device to all its attached control units. The one recognizing the address connects itself to the channel and responds by returning its address. The command is then sent over the Input-Output Interface, and this time the device itself

FIGURE 8 IBM 2260 DISPLAY STATION

responds by indicating whether it can execute the command. If an input-output operation is initiated at the device, the subchannel is set up to service the request.

II. INPUT-OUTPUT TERMINALS

This section presents the main features of several input-output devices which may be considered for use as remote terminals in a Credit Information System. Those particular advantages or disadvantages relevant to security are mentioned.

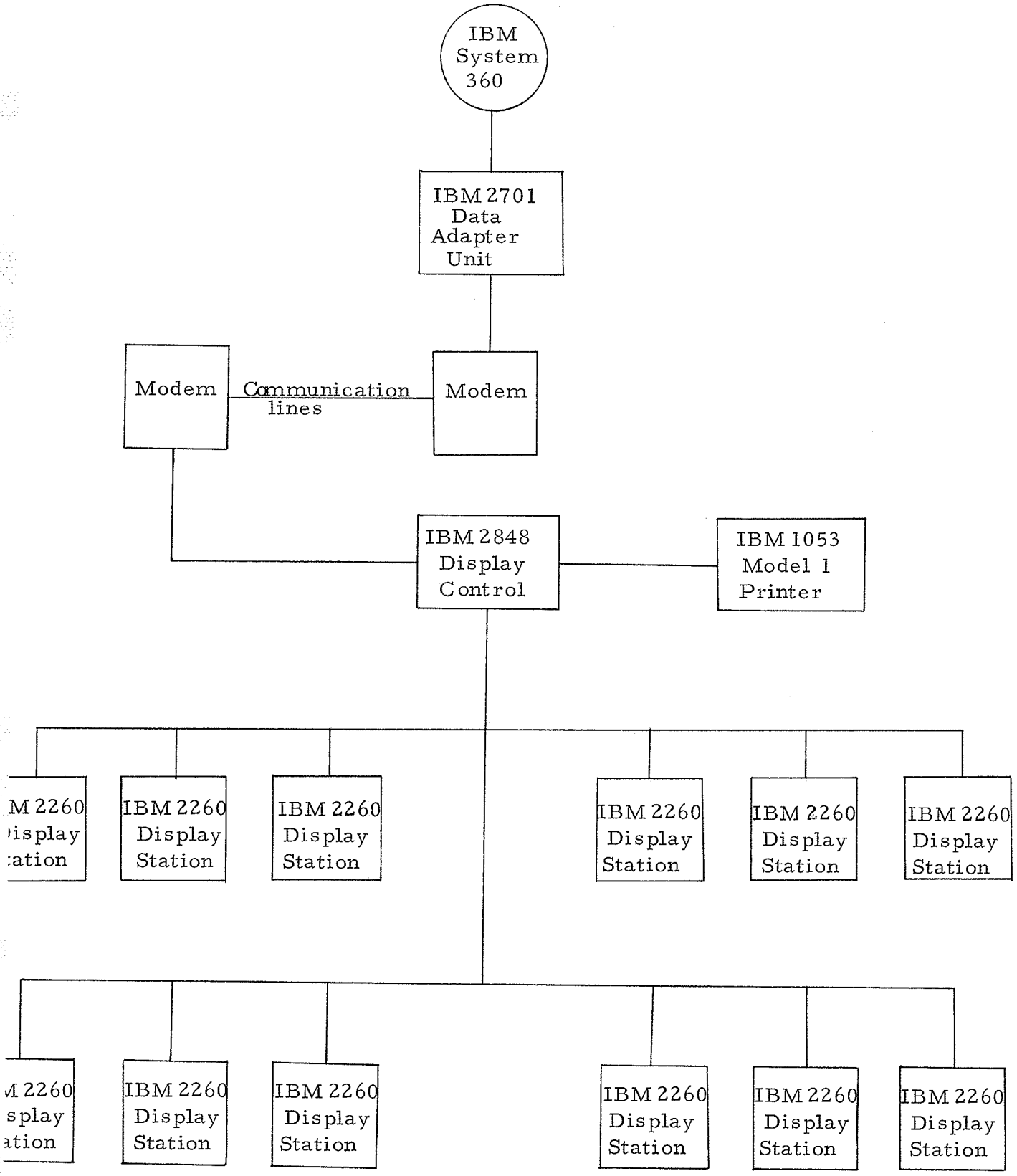
IBM 2260 Display Station (Figure 8)

The IBM 2260 Display Station, also known as a Cathode Ray Tube (CRT), has a television-like screen and is obtainable with or without a keyboard. Input to the computer is entered from the keyboard and displayed on the screen thus permitting a verification of the data before transmission. It is commendable for use in a Credit Information System, as Accounts Receivable records can be displayed and quickly updated at the terminal. Credit reports requested by the members can also be readily displayed on their screen.

The Display Stations are connected to the computer through the IBM 2848 Control Unit. In a remote application, both the Display Stations and the Control Unit are some distance away from the computer, and communicate with it via the IBM 2701 Data Adapter Unit* and modems* (Figure 9). There are three models of Control Units

FIGURE 9

IBM 2260 REMOTE DISPLAY COMPLEX CONFIGURATION



providing the following capabilities:

Model 1 - allows for a maximum attachment of 24 Display Stations. Each station is capable of displaying 240 characters (6 rows of 40 characters each),

Model 2- allows for an attachment of up to 16 Display Stations. Each station is capable of displaying 480 characters (12 rows of 40 characters each),

Model 3 - allows for an attachment of up to 8 Display Stations. Each station can display 960 characters (12 rows of 80 characters each).

An IBM 1053 Printer can be added to the 2260-2848 configuration. However, only one printer can be attached to a Control Unit and hence any hard copy documents required by users sharing a Unit will have to be prepared at the same location. It is essential then that no confidential information be printed unless tight security controls are enforced at the location of the printer. The printing of such information may be suppressed by appropriate programming. An example of this is a program feature to prohibit the printing of a user's code.

The chief disadvantage of the IBM 2260 in this application is its incapability to suppress the display of characters entered through the keyboard. It is therefore not impossible for someone to peek over the terminal operator's shoulder and to read the code entered.

Sanders CLINI-CALL System Terminal (Figure 10)

The Sanders CLINI-CALL System is a Hospital Information System developed by the Sanders Co., Massachusetts. The terminal used is

custom built and much like the IBM 2260 with modifications. It has a television screen for display, a keyboard for input and a photo-pen for selecting an item displayed on the screen. There are hardware features allowing the insertion of two identification cards. In its present usage the doctor places his card in one slot, which activates the system for him, and the patient's card in the second slot.

This type of terminal may be ideal for the situation at hand. Consideration will have to be given to the possibility of a user losing his identification card and the finder trying to use it. Tighter control can be attained by having a brief computer-terminal conversation in which the computer asks the operator personal questions such as his age and father's first name (Page 13). Alternative procedures will have to be devised to activate the system for a user who has lost, forgot, or misplaced his identification card.

Northern Electric Models 32 and 33 Teletypewriters (Figure 11)

The Northern Electric Models 32 and 33 are compact keyboard send-receive devices designed specifically for economical application while providing instant and reliable station-to-station teletypewriter services. In addition to the usual keyboard and carriage there is (i) a built-in telephone-like dial for direct calling, and (ii) a lighted push button control for initiating, accepting, controlling and ending calls. The Model 32 is equipped with a 3-row keyboard, and the Model 33 a 4-row. The keyboard permits operation at speeds up to

100 words per minute. Each model can print a maximum of 74 characters per line.

For terminal identification these models are equipped with an answer-back mechanism that will automatically transmit a pre-determined sequence of characters. There is a capacity for 20 distinct characters and these can be actuated locally by use of a special key. The characters are set when the teletypewriter is installed.

IBM 2740 Communications Terminal (Figure 12)

The IBM 2740 Communications Terminal is a keyboard send-receive device which can be used as either a standard selectric typewriter, or as a data sending and receiving device to another 2740 terminal or to a computer.

It operates at a maximum speed of 14.8 characters per second and prints 10 to 12 characters per inch, using a 13 inch width writing line on a standard carriage. It is designed for any of the following three types of operations:

(i) Point-to-Point

For this usage a dedicated communication line is required, private or leased. Operation is intended between two specific points, but others may be added.

(ii) Dial-Up

Dial-up operation allows the terminal access to a network of

terminals or to a multiplexor channel. Any 2740 connected to this network may call another.

(iii) Multiplexor

A 2740 Communications Terminal when attached to a multiplexor channel can be used with either dedicated or dial-up lines and provides the combined facilities of both types of operations.

While the 2740 may serve as a valuable piece of hardware for an office because of its dual form of operation, as a typewriter and as a terminal, it provides very little if anything at all, in the way of contributing to the privacy of a system. However, privacy could be maintained if printing on the carriage could be suppressed so that none of the private information sent to the computer for identification purposes is logged.

Optical Readers (Figure 13)

The huge quantity of input data typical of a Credit Information System may result in keyboard devices yielding to Optical Readers. Examples of such devices are the IBM 1287 Optical Reader, and the IBM 1288 Optical Page Reader. These are designed for use with the following computers: IBM 360 Models 25, 30, 40 and 50.

The IBM 1287 can:

(a) read formatted data on forms 2.25" by 3" to 5.9" by 9".

(b) read documents which are typewritten, machine printed, numerically hand printed or pencil-marked.

(c) read over 8,000 sales checks or up to 190,000 journal roll lines in one hour.

The IBM 1288 can:

(a) read formatted and unformatted documents ranging in size from 3" x 6.5" to 9" x 14",

(b) read documents which are typewritten, machine printed, numerically handprinted or pencil marked,

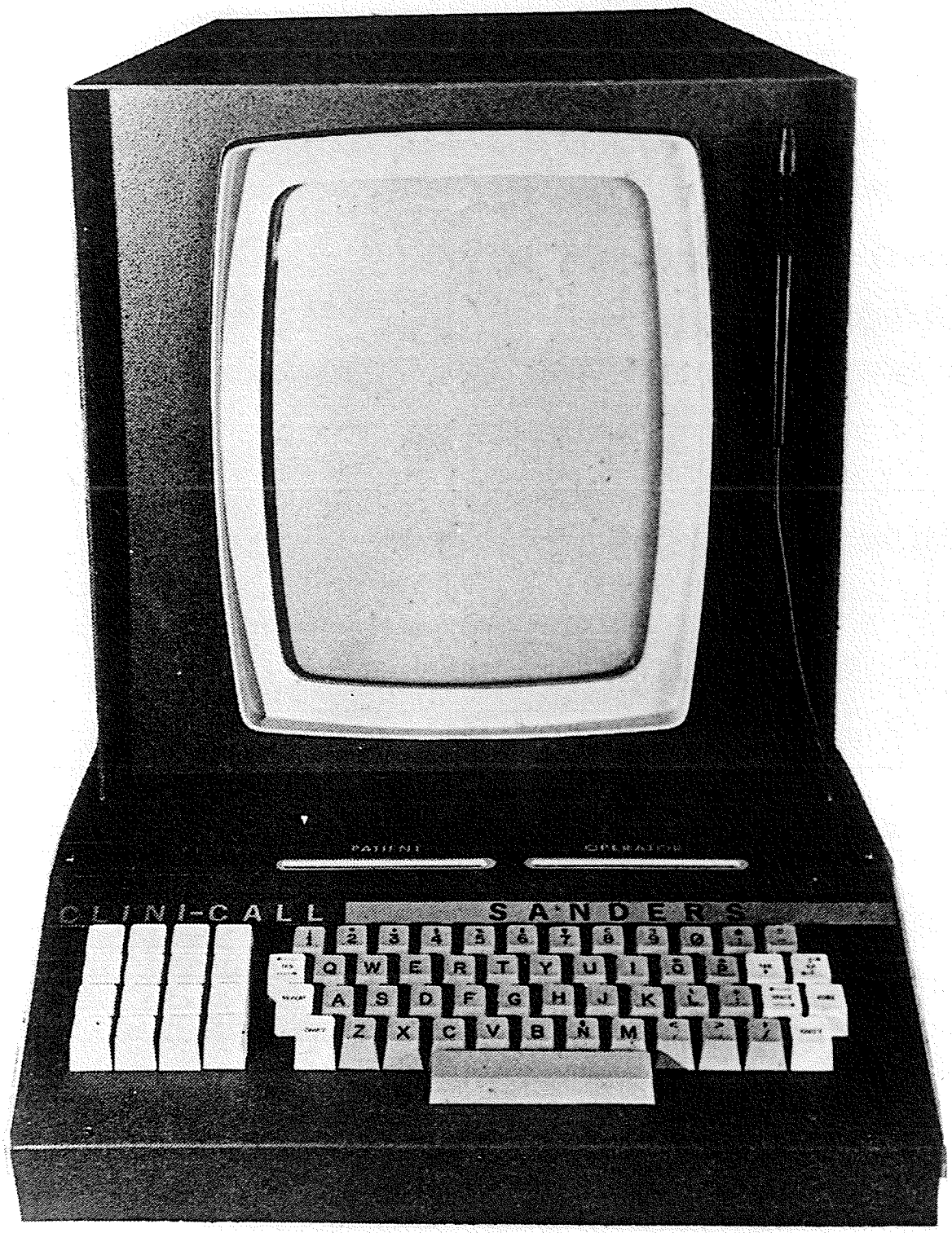
(c) translate information into System 360 language at the rate of about 840 single-spaced typewritten pages an hour.

The letters and numbers read are from the USA Standard Character Set for Optical Character Recognition. This type face is produced by certain electric typewriters including the IBM 2740.

Using an optical device will solve the problem of entering large quantities of data, but will necessitate an additional device for output from the computer. If all updating and report creations are to be done by the Central Credit Bureau, then such an optical device plus a high-speed printer will solve the input-output problem. However, it is likely that updating will be done by the members themselves, and since it may be economically unfeasible for each member to have two devices, the preceding send-receive units may be more justifiable.

FIGURE 10

Sanders CLINI-CALL System Terminal

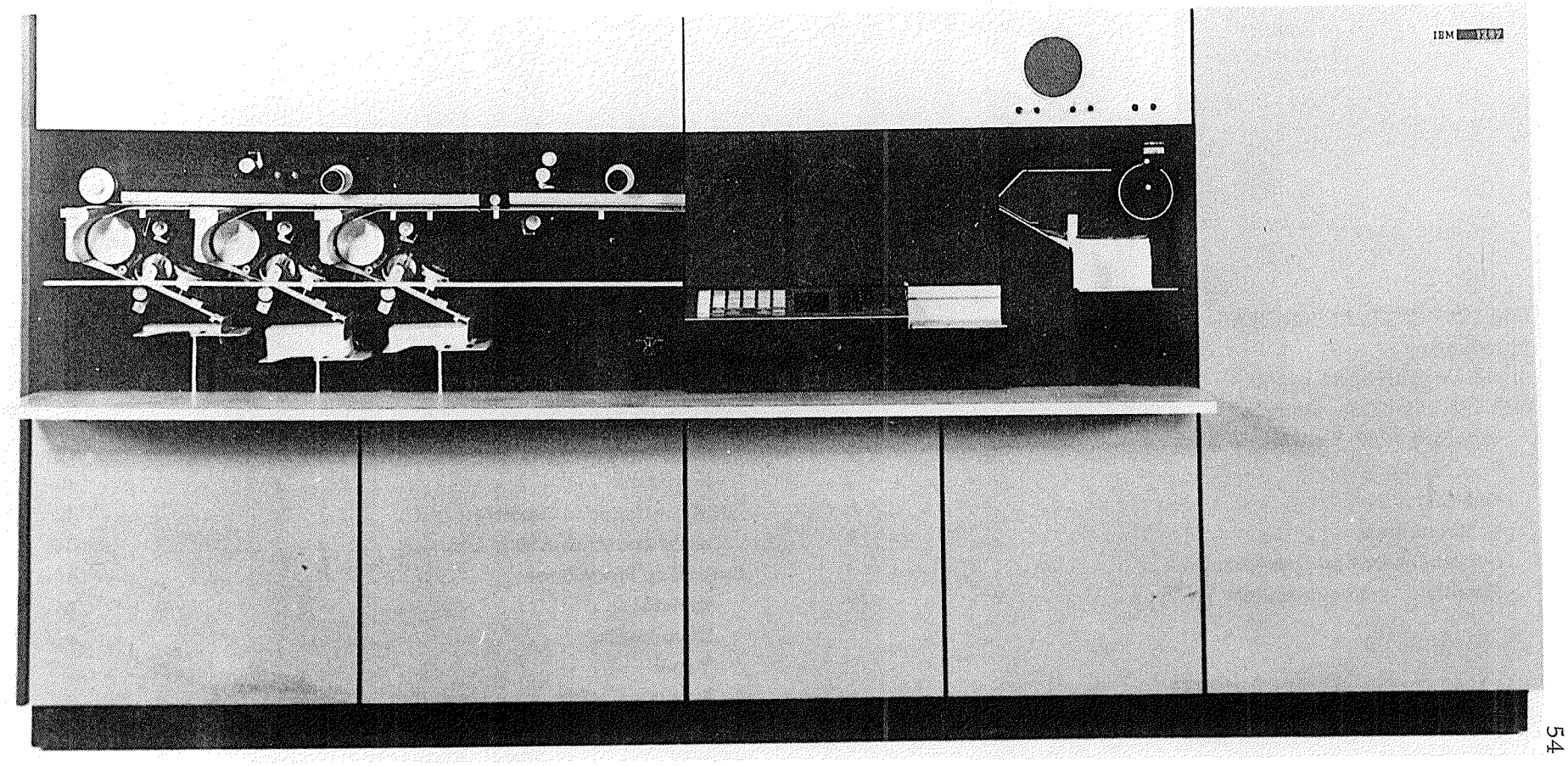


NORTHERN ELECTRIC MODEL 33 TELETYPEWRITER

53

FIGURE 13

IBM 1287 OPTICAL READER



CHAPTER V

A CONTROL SYSTEM FOR CONFIDENTIAL DATA

By utilizing some features of IBM's Operating System/360 and of various remote-access utilities, a system for controlling access to confidential data is now proposed. The data[i] is held in the auxiliary storage of a central computer and accessed by terminals located at remote points. These terminals may be any standard units such as teletypewriters, Cathode Ray Tubes (CRTs) and card-readers/punches. The control method is such that neither the routines required by the system to carry out its functions, nor the contents and structure of the files need be known by the users.

The chapter is divided into several sections. In the initial stage of the design several properties were defined which the system should fulfill, and these are outlined in Section I. Section II discusses the core of the protection scheme which involves controlling access through the medium of two files; a directory file and a subdirectory file. The threat of data leakage through computer room personnel is treated in Section III. Section IV gives the messages, called displays, by which the computer converses with the terminal operator. Having presented the basic design of the system in the preceding sections, Section V describes the manner in which it will function. The chapter is concluded in Section VI in which some general remarks about the control system are made.

---

[i]For a description of the data refer to Section II of Chapter I.

I. SYSTEM DESIGN FACTORS

The following factors were of prime concern in designing the system:

(1) The Duration of a Transaction

A transaction consists of the events that take place between the terminal operator and the computer. These events are: (i) the conversation necessary for identification purposes, (ii) the terminal operator's request for the computer to execute a set of functions, and (iii) the processing of the request by the computer and a display of the results to the terminal. The duration of the transaction is the time elapsed between the commencement of the first event and the completed display of the results.

The computing environment can be divided into the following three major components: the terminals, the communication facilities, and the computer. The duration of a transaction will depend upon the rate of data transmission between the computer and the terminal, upon the computer's processing speed, and upon the rate at which the terminal operator can make his entries. The time required by the first two factors is negligible compared to the third, and hence the duration of a transaction will be bound largely by the operator's typing speed and upon the quantity of information he has to supply. Thus, a major factor in designing the system will be to place a minimal dependence upon the

operator's typing skill.

The following example gives one case in which the quantity of input data will be reduced: Suppose that the display in Figure 5A is written to the terminal so that the operator may make a selection. The terminal operator will be required to enter only the number associated

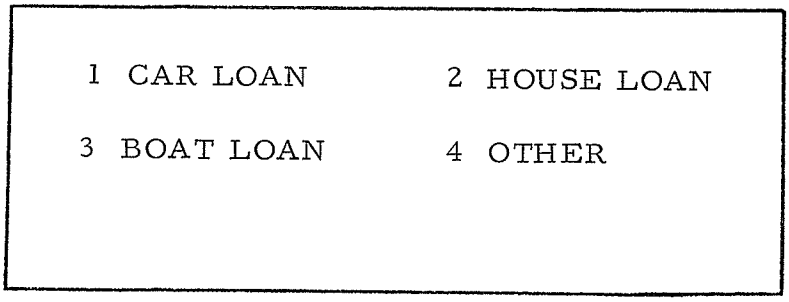


Figure 5A

with the selection rather than the lengthy item itself. Thus, if he wants to choose CAR LOAN, he only has to enter a 1.

(2) Ease of System Use

The system should be easy to use so that personnel will require very little training. The procedures for using it should be clearly defined, and terminal operators should be fully aware of what displays to expect and what action to take. The terminal itself should be simple to operate.

(3) Recovery from Errors

The system should allow the terminal operator to easily correct

typing errors. All error messages should be concise and clear and should be displayed as soon as possible so that the operator may not have to re-enter a bulk of information, the majority of which may be correct.

(4) Computer Response Time

The computer requires time, called the response time, to read a reply from the terminal, to process it and to send an answer back. The time taken to do this should be sufficiently small so that the terminal operator will not become impatient.

(5) The Displays

The displays transmitted to a terminal should be comprehensible and in an easily readable format (Section IV).

II. DIRECTORY AND SUBDIRECTORY

The set of programs to be written for implementing the control system will be referred to as the file protection routines. These routines will control access to the user's routines and data stored in the computer files by means of consulting a directory, and if necessary, a subdirectory. The directory file will be composed of one record for each user of the Information System.[i] The records may all be placed in

[i] The term "Information System" or simply "System" refers to the Credit Information System as opposed to "system" which refers to the control system.

one large file or in several smaller files classifying the users into geographic regions, alphabetic divisions or business types. A user's directory record (Figure 14) will contain his identification number, device addresses, code, trade name, a list of functions the devices contained in the device addresses fields can perform, a link and an optional information field. A subdirectory record will contain a set of device addresses, a list of the functions these devices can perform and a sublink field.

This section will account for the contents of these fields. Alternative ways will be given for assigning values to some of them, and choices will be made when the method is programmed in Chapter VI.

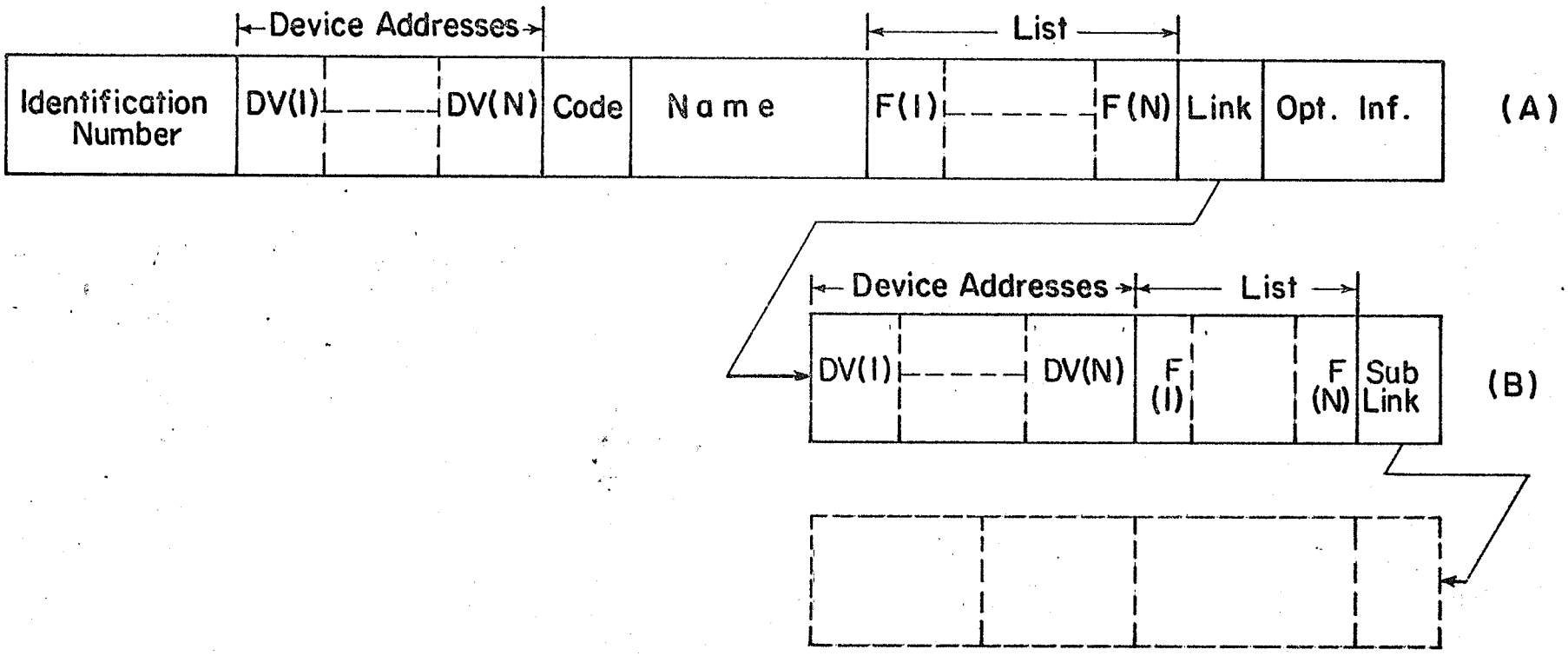
Identification Number

The Credit Bureau will assign every user of the System a unique identification number by which he will be known to all other users. If necessary, some technique can be adopted in assigning this number so that it may bear a definite meaning. Numbers can be associated with the physical address of the member's data on the storage media or with his type of business and its location. The identification number 1609118, for example, when divided into the fields 1, 60, 9, and 118 infers that the beginning address of the user's data is at Volume 1, Cylinder 9 and Track 118 of a disk storage device, and that his file contains a maximum of 60 records. Alternatively, if divided into the fields 1, 6, 09, and 118,

FIGURE 14

(A) FORMAT OF A DIRECTORY RECORD

(B) FORMAT OF A SUB-DIRECTORY RECORD



it may mean that the user is located in District 1 (say Eastern District, Head Office in Montreal), is the 118th member of that District, is in the clothing business (the code 09 stands for clothing), and that he has 6 branch stores. The District Head Offices of the Credit Bureau can be assigned special numbers which make them easily identifiable. Such a number could be an integer followed by a string of zeros. For example, the identification number of the Eastern District Head Office can be 1000000, and the Western Office 2000000, etc.

Device Addresses

A user will be permitted to have several terminals and will be restricted to using the System only through these. The device addresses fields will contain the hardware addresses of his terminals. This address, discussed in the preceding chapter, plays a leading role in the proposal. It is assigned when the terminal is installed, and normally remains fixed thereafter.

As a control factor, the device address feature has the advantage of being user independent as it is recognized by the computer without outside intervention. In fact, a user may be totally unaware that his terminal has an address.

Code

Together with an identification number, each user will be assigned a code which he will be required to furnish whenever he desires to use

the System. This code will be released only to the user and it will be his responsibility to protect it from unauthorized personnel. A choice from two types of codes is suggested. These types are:

(i) A Three-Digit Code. A three-digit code is an easily remembered and easily used number.

Applying the Principle of Permutations and Combinations[i] the number of different codes obtainable from the 10 digits is $10^3 = 1000$. However, if every business in Canada is to be accommodated in the System, there will be a maximum of 600,000 to 800,000 users, and consequently several of these will share the same code. The consequences of this factor are examined in the next chapter when the code generator (SUBROUTINE GEN) is discussed.

Three-digit codes can be assigned arbitrarily by the system's designer with the aid of a random number table. In a computer environment however, they will most likely be assigned by using a random number generator. This is a special machine routine designed to produce a random number or a series of random numbers according to specified limitations. By this latter method, codes can be quickly assigned to all present users and later codes can be assigned without the intervention of the system's designer or programmer.

---

[i] A First Year of College Mathematics by Raymond W. Brink, Second Edition, Page 453.

(ii) A Four-Character Code. Because of its simplicity and non-uniqueness, the preceding code may yield to a more complex one chosen from a larger set. This set may consist of letters, numbers and special symbols. An example of this latter type is a 4-character code formed from the elements of a 32-character table. The number of unique codes obtainable is now $32^4 = 1,048,576$ which is sufficient to allot each user a unique code.

Four-character codes can be produced by an algorithm which performs four operations on the identification number and chooses the required characters by a mapping of the result onto the 32-character table.

Probability Analysis of Codes. An unauthorized user, upon gaining access to a terminal may resort to the trial and error method of finding the code. What are his chances of being successful? In order to calculate a probability some knowledge of the code on the part of the user must be assumed. Assume that the person (a) knows the number of characters in the code and (b) is aware of all the possible characters of which a code can consist. The maximum probability is now calculated as follows:

Let p(n) be the probability of a success in n tries.

If N is the number of possible codes then

$$p(n) = n/N$$

Hence,

for a System using the 3-digit code $p(1) = 1/1000 = .001$

and

for a System using the 4-character code $p(1) = 1/1048576 = .00000095$.

The objective of the preceding calculation is to show the probability of a success in 'n' trials rather than to find the average number of trials which will be required for a success. It permits the conclusion that whichever type of code is used a large number of trials may be required to determine the correct code. To discover such an attempt, a routine is provided which writes a message on the console typewriter every 10 times the incorrect code has been produced at a terminal.

Name

The name field will contain the member's trade name.

A member's Accounts Receivable file can be updated by himself or by the Credit Bureau Head Office, and the latter may wish to update several members' accounts in the same computer run. Thus when a member's records are transmitted to the terminal for updating, the member's name will be placed in the first line of the display so that no confusion as to whose account is being updated will arise. For example, if the terminal is a CRT, the first line of each screen of data will contain the name of the member, and the last line will indicate whether the records are to be continued in the next screen.

List

The Credit Information System will provide its members with the facility of performing one or more functions which will be presented in a base display. The base display will be unique to each terminal as it would contain only those functions which that terminal can execute. The functions which will be provided are:

(1) Status Check

By requesting the status check function the user can find out how many times his code has been correctly or incorrectly specified. The latter gives an indication as to whether some unauthorized person in the office is trying to use the System.

(2) Update Accounts Receivable[i]

This function allows the user to store his Accounts Receivable file in a computer file, and update it from a terminal located in his office.

(3) Credit Report [ii]

The credit report function is used to request credit reports on businesses.

[i, ii] The programs required for these functions will be provided in another project. A simple 'Update Accounts Receivable' program will be used in this project for system demonstration purposes.

(4) Code Generator

The code generator provides the identification numbers and codes required for adding new members to the Credit Bureau.

With each of these functions there will be an associated number. For example, the status check function may be number 1 and the credit report function number 3. The list fields in a record will contain the numbers of those functions which the terminals contained in the device addresses fields can perform.

These numbers will be used to construct a base display for the terminal. Figure 31 (Page 80) shows a base display for a terminal which can execute the functions numbered 1, 2, 3 and 4.

Link

A user will be permitted to have several terminals. These may have access to different sets of data and hence they will be used for different functions. For example, there may be one in his stock-room used for inventory control and another in the accounting department to be used for payroll calculations. These two data bases are to be available only to the particular terminals for which they are intended (Figure 15, Page 68).

Because all his terminals may not be permitted to carry out all the functions available to him in the Information System, the user will have a record in the main directory and one or more records in the

subdirectory. The address of his first subdirectory record will be held in the link field of the directory record. Thus, the main directory record will contain the addresses of some of his devices and the functions which these can perform. The addresses of the remainder of his devices and the functions available to them will be listed in the subdirectory records. The example of records in the directory and subdirectory at the end of the section will help to illustrate this point(Figure 15, Page 68).

Optional Information

This field will contain any miscellaneous information the Credit Bureau wishes to keep on the member. Typical examples are the date on which the business became a member of the Bureau or the date on which the latest credit report was compiled for that member.

Subdirectory

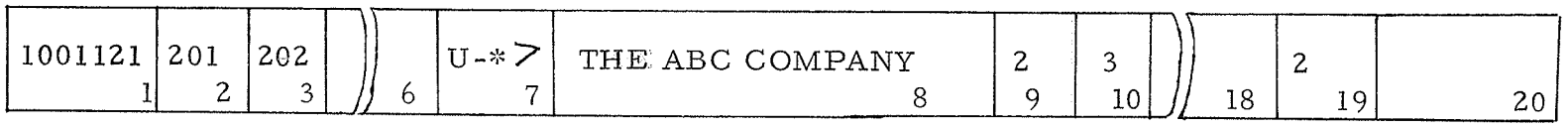
The subdirectory contains a set of device addresses and the functions which these can perform, and a sublink field. As mentioned previously ('Link' subsection) a user may have several records in the subdirectory. In this case, the sublink field of his first subdirectory record will contain the address of his second record in the said subdirectory.

Example:

Figures 15 gives sample records for the hypothetical ABC Company in the directory and subdirectory files. This company has the identification

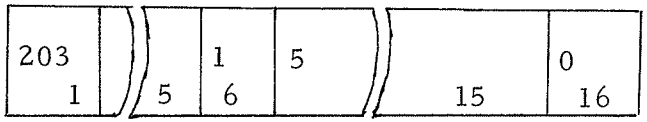
FIGURE 15

SAMPLE RECORDS IN THE DIRECTORY AND SUBDIRECTORY FILES



(A)

(B)



Legend

(A)
- 1 Identification Number
- 2-6 Device Addresses
- 7 Code
- 8 Trade Name
- 9-18 Function Numbers
- 19 Link
- 20 Optional Information

(B)
- 1-5 Device Addresses
- 6-15 Function Numbers
- 16 Sublink

number 1001121 and code YTU=. The latter, when scrambled(Section III) becomes U-*>, which is stored in the directory record. The user, however, knows only the code YTU= and this is the one he enters at the terminal. The scrambling and the comparison are done by the control system.

The ABC Company has two terminals (addresses 201 and 202), which can perform the functions whose numbers are 2 and 3. The entry in the link field indicates that this Company also has a record in the subdirectory file, record number 2. The latter record contains a terminal whose address is 203 and can execute functions 1 and 5. The 0 in the sublink field indicates that this is the last subdirectory record for the ABC Company.

III. COMPUTER ROOM PROTECTION

A threat to the security of the data and routines emerges from the possible leakage of information through collaboration between employees in the computer room and external users.

The data, and in all likelihood, the system's routines, will be stored on a direct access storage device. Several protection features of IBM's Operating System/360 will be used when these files are created. Since computer room employees will have no knowledge of the parameters under which the files will be created, a normal access by them to the data cannot be affected.

This alone is not sufficient to ensure protection since it is possible to copy the entire contents of a storage medium by using utility programs supplied by the computer manufacturer. These programs are frequently required by the computer operator for the preparation of backups; that is, for the copying of one storage medium onto another as insurance in case the original is destroyed.

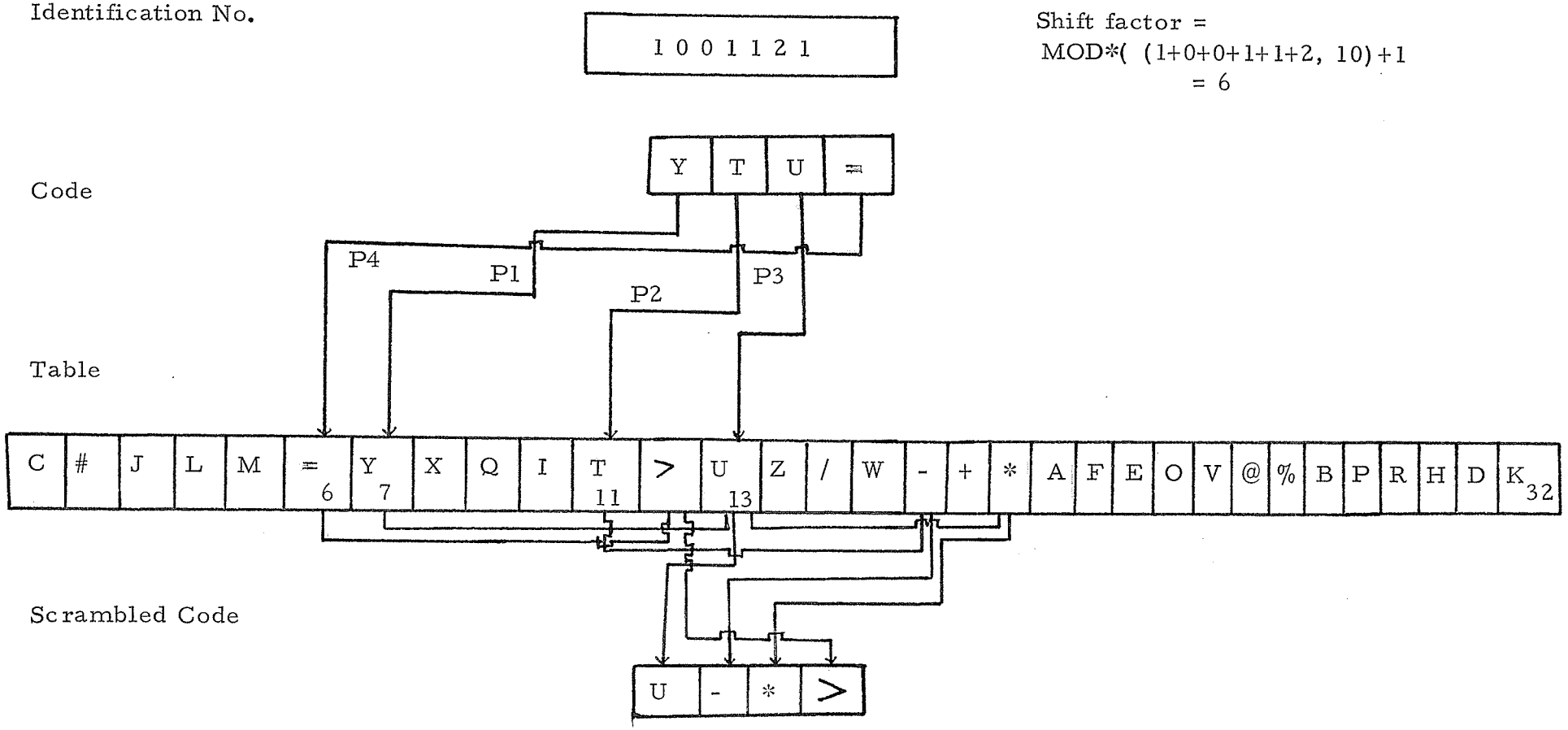
As a solution to this problem all critical data will be scrambled before storage. The term ' data scrambling' is used in this project to refer to a transformation of data at the character level. For example, the word AB can be scrambled to the new word ZQ, but someone unaware of the original word and of the scrambling process cannot decipher it.

The two most critical pieces of information which will be stored in the computer files are the code in the user's directory record and the name field in the Accounts Receivable file. Both of these data items will be stored in a scrambled form. An attempt will be made to make the transformation process sufficiently complicated so that unscrambling these fields will be difficult for anyone obtaining a copy of the data.

The scrambling concept is best explained by applying it to a 4-character code. Recall that a user's directory record will contain his identification number and his code. Figure 16 illustrates the steps followed in scrambling the code YTU=. The method applied is one which replaces each character in the code by one in the table. The table contains 32 characters arranged in random order. A character is

FIGURE 16

SCRAMBLING PROCESS



replaced in the following manner:

(1) find the position, called the index, in the table occupied by the character in the code.

(2) find a new index by adding a shift factor to the previous index. The shift factor is obtained by taking the sum of the first 6 digits of the identification number, and dividing it by 10, then adding 1 to the remainder.

(3) replace the character in the code by the one occupying the new index position in the table.

(4) if the new index value exceeds 32, then subtract 32 from it. For instance, if the new index value is 37 then this would be recalculated as 37 - 32 = 5.

For an application of these rules, follow the path P1 in the diagram to replace the Y of the code. The calculation of the shift factor is shown in the diagram. Its value is 6. The index of Y in the table is 7, and therefore the new index is 7 + 6 = 13. The 13th position in the table is occupied by U; and hence it is the new character. Repeating this process for the remaining characters (paths P2, P3, and P4), the scrambled code is obtained as U-*>.

The code which will be stored in the directory is U-*>, but when the terminal operator is requested to enter his code he must reply with YTU=. The software will then scramble this and compare it with the one stored in the directory. Thus, the users will be given a set of

codes, and the scrambled equivalent of each code will be placed in the directory file.

The scrambling algorithm is an original one developed by the author. There is no assurance that the algorithm will produce unique scrambled codes; in fact, it would be an advantage if it does not. To illustrate this consider the two hypothetical codes = IUW and X>/ +. Suppose that their shift factors are 3 and 1 respectively. After scrambling, both of these will generate the same code, namely QUW*. The advantage lies in the fact that someone trying to decipher the codes will, on first thought, believe that the original codes were also the same.

IV. THE DISPLAYS

The displays to be used in the control system are formatted for the IBM 2260 Display Station[i] (Model 2) with keyboard, but can be easily modified for other types of terminals. The screen of this terminal can display a maximum of 480 characters. These characters appear on 12 lines of 40 characters each.

The displays consist of commands, statements, error messages (diagnostics), and output transmitted to the terminals by the computer. These displays are define as follows:

(i) Commands (Pages 77-79)

A command display requires some direct action from the terminal

---

[i]The reader is referred to Appendix A of (16) for a description of the characteristics and operation of this terminal.

operator, such as an entry of his company's code. Whenever input is requested from the terminal, the start symbol* and the cursor* are positioned at that point on the screen at which the operator can begin his entry. However, the terminal operator is not obliged to begin his entry at that point, and can move the cursor to whichever position he pleases.

(ii) Statements (Page 85)

A statement is a message which informs the terminal operator of some condition, such as the busy condition pertaining to the computer facility.

(iii) Diagnostics (Pages 81-84)

The diagnostics are the messages sent to a terminal whenever its operator has supplied incorrect information or has requested a function which is unavailable to that terminal. For example, if the terminal operator entered an incorrect identification number, a diagnostic would inform him of this and would tell him what action should be taken.

(iv) Output

The output displays consist of the results of executing the functions requested by the user. Whenever the quantity of information to be displayed exceeds the number of characters that can be accommodated on the screen, a first screen of output is displayed and an indication is given that there is more information to come. After the first screen of

data is acknowledged, a second screen of output is displayed and this process is continued until all the output has been transmitted.

The displays are shown in Figures 17 to 26. For an example of an output display the reader is referred to Exhibit 29 of Appendix 3. Most of the displays are self-explanatory but the following deserves further elaboration:

(i) The Initial Display (Page 76). This is the display with which a transaction is started. The computer writes it to all the terminals and polls* for a reply. A user requiring the use of the System must respond by entering his identification number and code.

(ii) Statement-1 (Page 85). This message is written to a user who has requested the System while it was attending to someone else. The address of the user's device is placed in a queue, and Statement-1 informs him of his position in this list. When the System is ready to give him attention, the initial display will be written to his terminal.

(iii) Diagnostic-1(Page 81). Diagnostic-1 is written to the terminal when the operator has entered the incorrect code twice in a row.[i] The computer then terminates the job and checks the waiting list for the next terminal to receive attention.

---

[i] When this occurrence has been repeated for 5 times, a message is written on the console typewriter (Pages 64, 112).

FIGURE 17

THE INITIAL DISPLAY

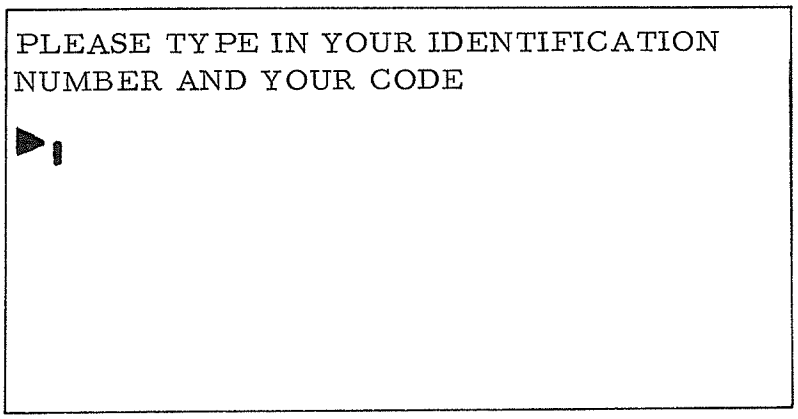


FIGURE 18

COMMAND - 1

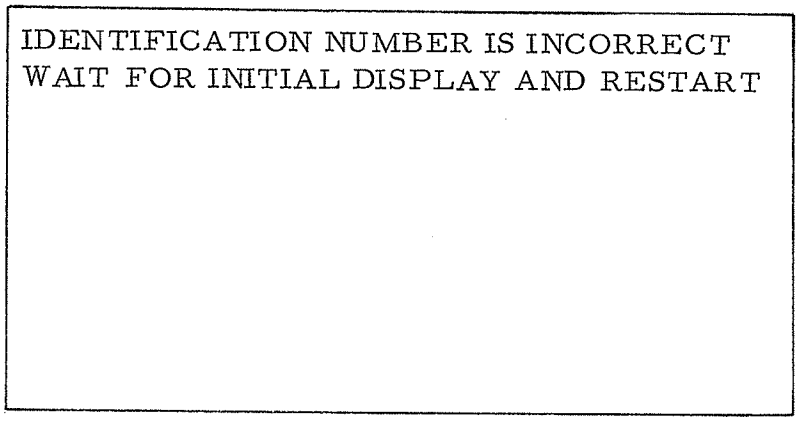


FIGURE 19

COMMAND - 2

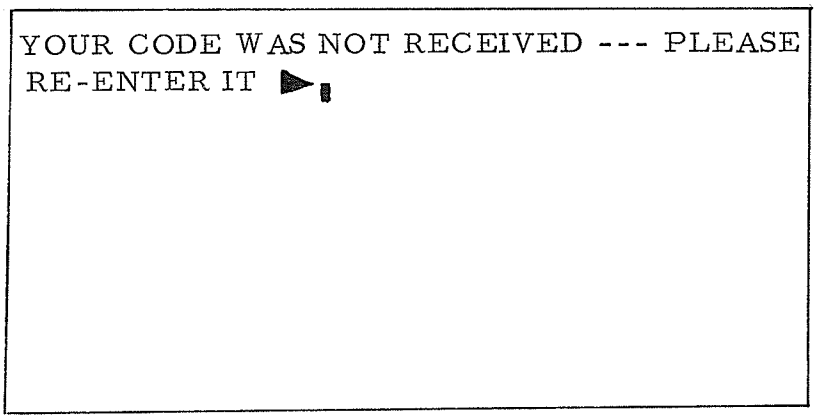


FIGURE 20

COMMAND - 3

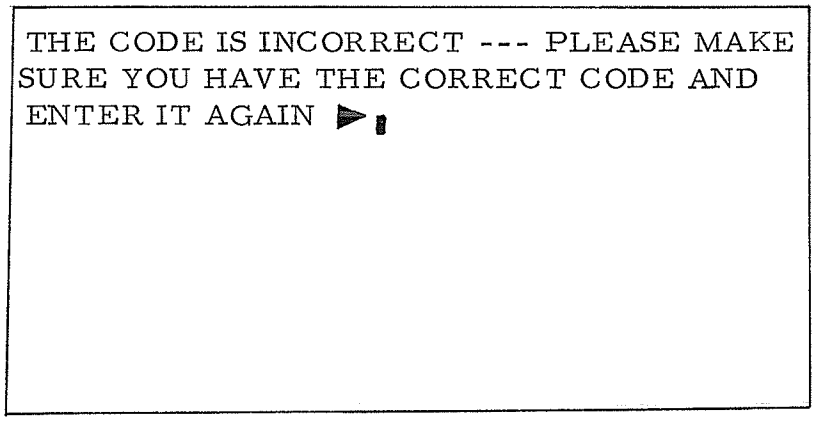


FIGURE 21

A BASE DISPLAY

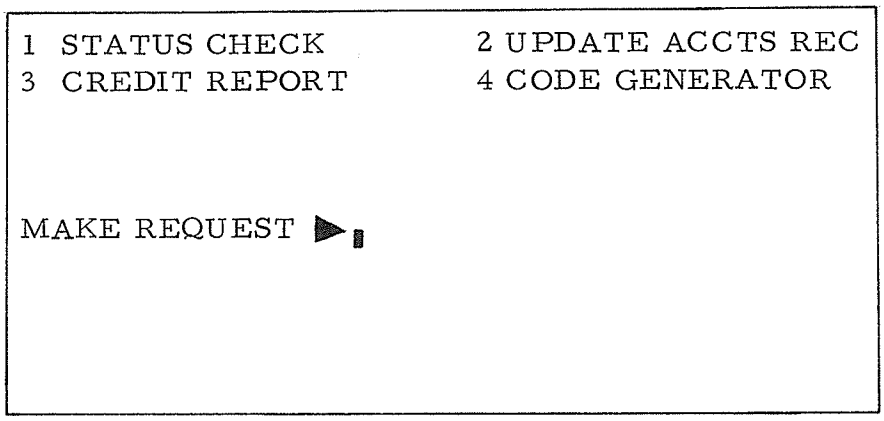


FIGURE 22

DIAGNOSTIC - 1

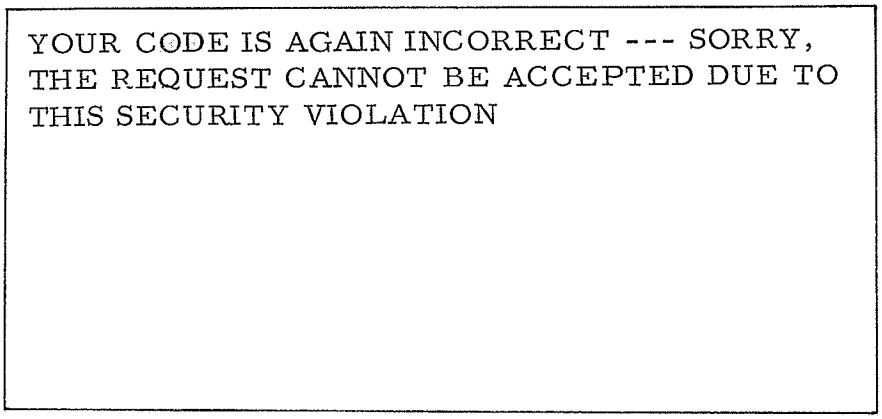


FIGURE 23

DIAGNOSTIC - 2

SORRY, THIS DEVICE IS NOT RECOGNIZED
AS ONE AVAILABLE TO YOU

FIGURE 24

DIAGNOSTIC - 3

FUNCTION NUMBER L^i IS UNAVAILABLE

[i] L will be replaced by the function number.

FIGURE 25

DIAGNOSTIC - 4

THE FOLLOWING IDENTIFICATION NUMBERS ARE INVALID:

I[i] J[ii] XXXXXXX[iii] - - - - - - - - - - - - - - -

REQUEST NEXT SCREEN AND RE-ENTER THE FUNCTION AND IDENTIFICATION NUMBERS

---

[i] I will be replaced by the function number.

[ii] J will be replaced by the position of the identification number in the sequence of identification numbers entered.

[iii] XXXXXXX will be replaced by the incorrect identification number entered.

FIGURE 26

STATEMENT - 1

SORRY - THE SYSTEM IS IN USE - - - YOU HAVE NOW BEEN ADDED TO THE QUEUE AS NUMBER: J[i]

[i] J will be replaced by the user's position in the queue.

V. CONTROL SYSTEM OPERATION

This section examines the manner in which the file protection routines will interact with the routines and data of the Credit Information System in handling a transaction. The said Information System will provide the user with the facilities for updating his Accounts Receivable file and for obtaining Credit Reports on businesses. To these will be added the Status Check function and the Code Generator provided by the control system.

A member of the Credit Bureau will be restricted to checking only his own status and updating his own Accounts Receivable file, but may receive credit reports on several businesses. To perform the first two functions he has to enter the numbers associated with them when the base display is presented. For the Credit Report function he must enter the number of the function, and the identification numbers of the businesses on which he desires the reports. A member is never allowed to use the code generator. The Credit Bureau on the other hand, can perform all of these functions for several members. To accomplish this, it selects the numbers associated with the functions desired, and provides the identification numbers of the members. The Credit Bureau can also check its own status.

When the central computer starts attending to the Credit Information System, either after overnight shutdown or after servicing other applications,

it will write the initial display to all the terminals and poll for a reply (Figure 27-1). A user wishing to use the System will reply by entering his identification number and code. The file protection routines will now begin their validity checks. The identification number will be checked (Figure 27-2), and if incorrect, Command-1 (Page 79) will be transmitted to the terminal (Figure 27-3).

Upon receiving a correct identification number the control system will use it to calculate the record number of the user's directory record by the formula:

Record Number = Identification Number - Starting Number + 1

where

the Starting Number is the identification number contained in the first record of the directory file.

For example, if the identification number obtained from the terminal is 100119, and the one contained in the first directory record is 100118, then the

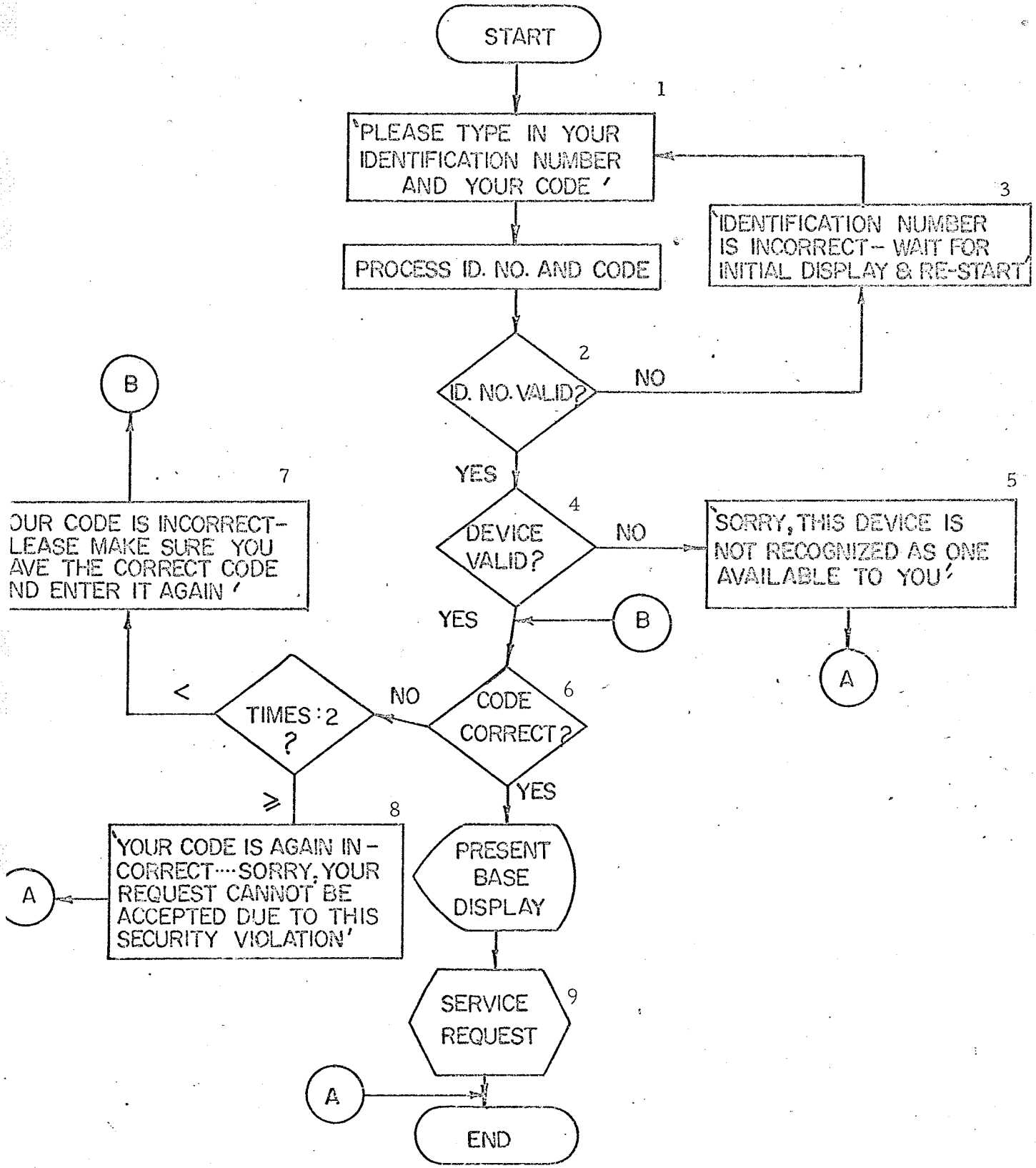
Record Number = 100119 - 100118 + 1 = 2.

This calculation is based upon a numbering system in which the identification numbers are assigned in an ascending sequence, increasing by unity.

The user's directory record will now be read. At this time the computer will already be cognizant of the hardware address of the terminal and will compare it with the ones in this record (Figure 27-4).

FIGURE 27

COMPUTER-TERMINAL CONVERSATION
PRIOR TO ACCEPTING A REQUEST



If the address cannot be found in the record, a check will be made to see whether that user has a record in the subdirectory, and if so whether that device address is present there. If the address cannot be found in the directory or subdirectory, Diagnostic-2 (Page 82) will be transmitted to the terminal (Figure 27-5).

If the device address is found, then the codes will be compared (Figure 27-6) and if they do not match, Command-3 (Page 79) will be displayed at the terminal giving the operator a second chance to produce the correct code (Figure 27-7). If an incorrect code is entered again, Diagnostic-1 (Page 81) will be sent to the terminal and the job will be terminated (Figure 27-8).

When the correct code is received, the file protection routines will construct the base display by using the numbers contained in the list of fields of the record. These numbers give the numbers of the functions the terminal using the system can execute.

The terminal operator now makes his selections and processing begins (Figure 27-9). Although only those functions which can be performed by that terminal will be included in the base display, the functions selected will nevertheless be compared to the contents of the list field to ensure that the user is not attempting to perform other functions. All identification numbers will be verified, and any incorrect ones will be removed from the request list and placed in a stack for later display in Diagnostic-4 (Page 84) to the terminal.

The System will now be ready to process the valid requests and will transmit the necessary output to the terminal. After completing this, the list of incorrect identification numbers will be checked and any such numbers will be displayed at the terminal. The operator will be asked to re-enter the functions and identification numbers, and that request will be processed. Finally, a list of the functions requested, but which are not available to that terminal will be displayed. This completes the transaction for that user and the initial display will be rewritten on his terminal.

During the time in which the System will be attending to a user, others may try to use it. Thus, a queue will be built, into which these members will be placed in the order in which their interrupts are received. On completing a transaction, the System will check this list and will write the initial display to the terminal highest in the queue. This terminal will be removed and the list will be pushed up.

Though the control system has been designed to handle only one terminal at a time, it may be extended to allow several terminals to operate simultaneously. This can be done by storing the requests made by the terminals, and processing these requests when the files they require are read. For example, if 5 terminals are using the System simultaneously and 3 of these require the Status Check file (Page 98), the file will be read once and the request of the 3 terminals will be processed. By permitting several terminals to operate simultaneously,

the System will require less time to process their set of requests than if it took each one individually to completion.

VI. REMARKS

The successful operation of the control system will demand each user to protect his terminal from outsiders and to provide his code only to trusted personnel. The code is mainly a protection against unauthorized personnel in an office. It is of no use to a competitor unless he can get access to the terminal associated with that code.

The directory can be used for other applications in the Information System. It can be expanded:

(i) to store the formats of the members' Accounts Receivable records since these will be required for updating purposes,

(ii) to keep the addresses of the user's data on the storage medium,

(iii) to link all the businesses of the same owner trading under different names, thus allowing a complete credit report on the owner.

In addition, it may be used to create a periodical list of all the users in sequence according to names or identification numbers.

The directory is expandable to include new members, additional terminals for existing members, and more user functions as they are added to the System.

CHAPTER VI

CONTROL SYSTEM IMPLEMENTATION

This chapter presents the programming and testing of the control system proposed in Chapter V. The programs are written in the Fortran IV (G-level) and Assembler languages. An IBM System 360/Model 65 computer operating in a batch-processing mode and under control of the Operating System was used. The peripheral equipment utilized consisted of two terminals (IBM 2260s with keyboard), a high-speed printer, disc drive, and a card reader (Figure 28). Files were created using test data similar to actual data received from the Bureau for handling 8 members and the Central Credit Bureau. The programs may be easily expanded to include more members and Bureau Head Offices; accommodation is limited only by the amount of auxiliary storage available.

The programming and testing of the method took place in several steps and is discussed under the sections of:

(1) File Creation - Directory, Accounts Receivable, Status Check and Subdirectory files.

(2) System Routines

(3) System Example

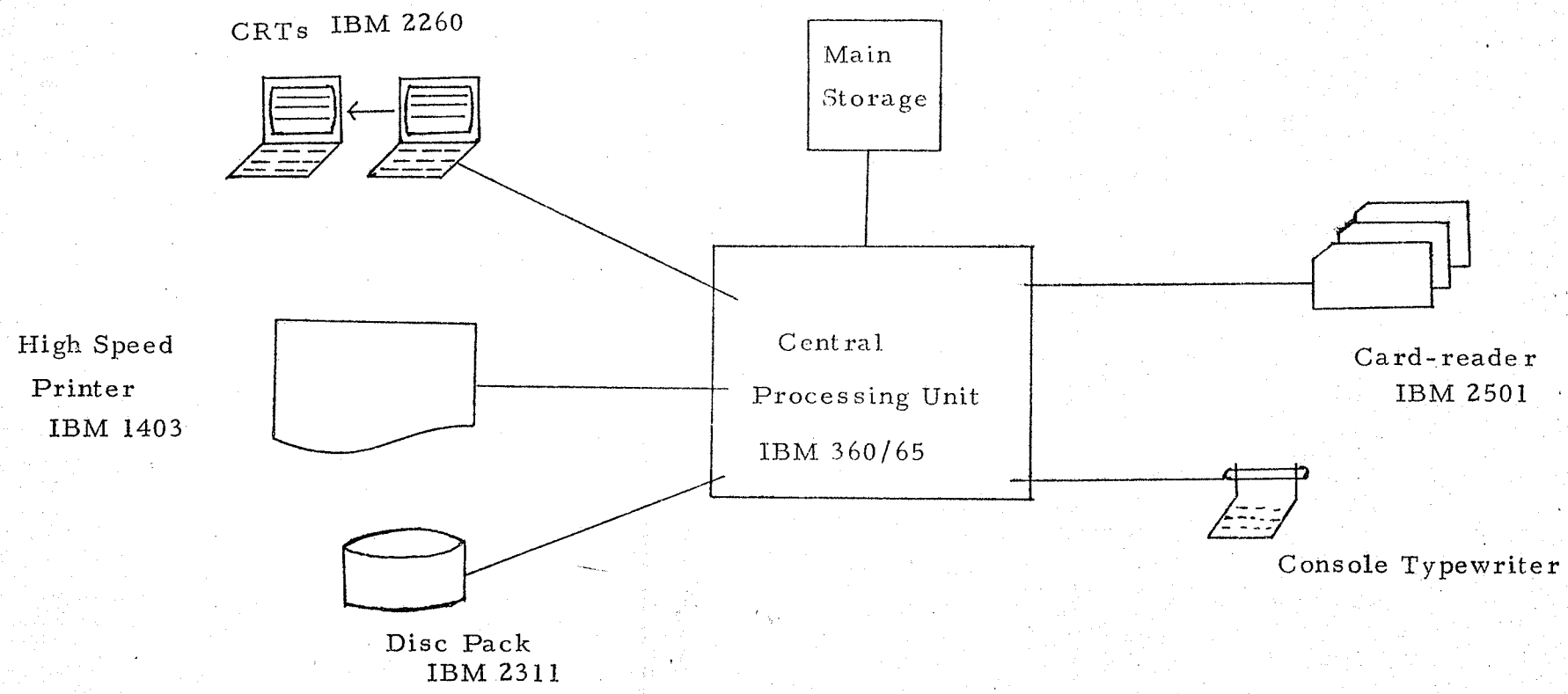
I. FILE CREATION

Data Storage Features

Data Set Organization. Most of the files required for developing

FIGURE 28

HARDWARE CONFIGURATION

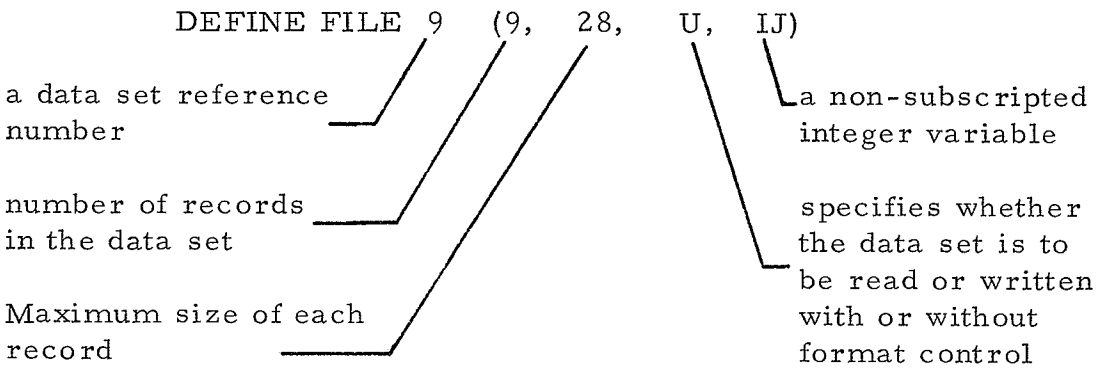


the control system contain a large number of records; the directory and status files have one record for each user and the Accounts Receivable file contains several records per member. Generally though, only a small percentage of these will be required to process a transaction. For example, if the System is in use by a member, only one record in the directory file will be read, and one in the status file will be updated. The need for processing an entire file will scarcely ever become necessary, the only possible necessity being for a periodic changing of the codes in the directory file. For this kind of file processing, the Direct Method of Organization is preferred as this method minimizes the time required to fetch a single record. Hence, all the files are organized in this manner.

Storage Optimization. An attempt is made to optimize the use of storage space. The binary representation of a device address never requires more than 16 bits (page 40), and the highest value of a function number can be accommodated in 8 bits. Thus, the device addresses and list fields of the directory and subdirectory files are stored in 2-byte words rather than the usual 4 bytes. Since these files are created to handle 5 terminals and 10 functions in each record, this results in a saving of 30 bytes per record. If storage space becomes a critical factor in the System, the Optional Information field (40 bytes) in the directory records can be omitted.

Data Set Characteristics. Various Data Set Protection features of Operating System/360 are used in file storage. Each file is assigned a Data Set Name, a Volume, and a set of Data Control Block parameters are taken implicitly from the DEFINE FILE[14] statement used in creating Direct Organization files.

An example of a Define File Statement is



The subparameters formed from this statement are:

Record length = 28 storage units
= 28 x 4 bytes
= 112 bytes
Block length = Record length = 112 bytes
Method of Organization = DA (Page 22)

The Files Created

(i) Directory (110 bytes per record)

The directory is the initial file consulted in executing a transaction and is used for verifying a user's identification number, code, device address and list of functions selected.

The fields in the directory records are those discussed in Chapter V. Each record is appended with the beginning and ending address of the member's Accounts Receivable records (Page 90). The identification number and code, however, require further discussion. Once read, a record is kept in main storage throughout the execution of a transaction.

Each record in the directory allows for 5 terminals, 10 functions, a name field of 20 characters, and 40 characters of optional information.

Identification Number. The identification number received from a terminal is used as the key for calculating the record number of a user's record in the directory (for an example, see Page 87). It is thus an essential factor in the control system as an incorrectly specified one will result in an attempt to locate a non-existent record in the file. For such an attempt, the Operating System terminates the job without allowing the control system the opportunity of trapping the error and informing the terminal operator. The latter is thus deprived of recovering from such an easy-to-make error.

One solution to this problem requires a file of all identification numbers to be kept and those received from the terminals compared with the ones in this file. This method is unsatisfactory when there is a large quantity of numbers, even if an efficient searching technique such as a binary search* is made. A second solution requires each identification number to be augmented with an additional digit. This digit, called a check digit, is produced by performing several operations on the

6-digit number. This is the solution employed here.

Thus when an identification number is read from a terminal, its check digit is verified thus determining whether the number is a valid one. This method eliminates the storage of an additional file and the time consuming comparisons of identification numbers as would be required in the first solution.

Hence, the identification number used in this System is a 7-digit one, including the check digit. This will allow the Credit Bureau to allot every business in Canada a unique identification number.

Code. Due to the small probability of a user's code being found by the trial and error method, and because of the complexity in its method of assignment, the 4-character code discussed in the preceding chapter is chosen. As a measure for computer room protection, it is scrambled prior to storage in the directory. The manner of scrambling takes one of ten possible paths depending upon the user's identification number. In each path, every character of the code is replaced by one chosen from a 32-character table (Page 71). Thus, anyone obtaining a copy of the directory is unaware that he has a set of incorrect codes until he tries to use them.

(ii) Accounts Receivable (80 bytes per record)

All the Accounts Receivable records for the members are held in one file with the beginning and ending address of each member's accounts

kept in his directory record. The name field of each record is scrambled before storage and unscrambled when presented for updating. The principle of the scrambling process used is the same as discussed previously (Page 70).

The system allows a member to have full access privileges only to his set of records, but he may obtain selected information, in the form of a credit report, from the entire file.

(iii) Status File (36 bytes per record)

The Status file keeps a count of the number of times each user has successfully and unsuccessfully used the System (this is the equivalent of saying the number of times the correct and incorrect codes have been given at a terminal). It provides management with the facility for checking at any time (a) whether an unauthorized person in the office is attempting to use the terminal, as this will be reflected in the number of times the code produced has been incorrect, and (b) how many times the company have actually used the System.

Whenever any user has accumulated 10 unsuccessful tries, a routine intervenes and writes a message on the console typewriter giving the identification number of the user together with a message to this effect (Page 112). The computer operator is provided with a procedure for informing the owner of the terminal about such an occurrence. He is given a list containing the names, identification numbers and telephone numbers of all the users and uses this to contact management of the

company whose terminal is suspected of being abused.

(iv) Subdirectory (34 bytes per record)

The fields of the subdirectory are the same as those discussed in Chapter V. Each record allows for 5 terminals and 10 functions.

II. SYSTEM ROUTINES

The routines written for the realization of the proposal are:

(1) THE MAIN PROGRAM (2) EVAL

(3) CHEK (4) CHEKUP

(5) UPDATE (6) GEN

(7) STACK (8) SHCODE

(9) ATTEND (10) QUEUE

(11) OPEN, POLLIN and POLOUT

This section provides a summary of the main functions of these routines, and detailed descriptions of the programming techniques applied can be found in their internal documentation (Appendix 3). The linkage of the subroutines is shown in the flow diagram of Figure 29 (Page 118). The numbers enclosed in parentheses in the discussion to follow are references to the said flow diagram.

(1) THE MAIN PROGRAM

The subroutines used are placed under control of a single program, called the main program. This program is the first one used in processing

a transaction and calls on the subroutines as they are required. After a subroutine is executed, control returns to the main program.

The main program also does the following:

(i) reads the messages for display to the terminals from the card-reader (1),

(ii) calls subroutine OPEN to open the terminals for communications (2),

(iii) initializes the counters and flags[i] for later use by itself and by the subroutines (3),

(iv) uses subroutine POLOUT to write the initial display to the terminals and polls for a reply (6). It uses subroutine POLLIN to read the reply (7) and then calls subroutine EVAL to evaluate the response (8),

(v) calls subroutine GEN to verify the identification number received (9) and then reads the user's directory record (11). It now checks the device address (13) and code (15) and uses subroutine POLOUT to write the appropriate displays to the terminal when a violation is encountered (10, 14, 29),

(vi) accepts the selections from subroutine EVAL and compares them with the list in the user's directory record (31). All functions unallowable to that terminal

---

[i] Flags are indicators used frequently to tell some later part of a program that some condition occurred earlier.

are placed in an array for later attention (3),

(vii) answers the valid requests by using the necessary subroutines and files,

(viii) displays a list of the rejected requests (63),

(ix) displays a list of the incorrect identification numbers found in the selections (65),

(x) checks the waiting list (66) and takes the following actions:

(a) if the list is empty the initial display is written to all the terminals and polling takes place again (6),

(b) if the list is not empty, attention is given to the terminal which has been waiting the longest (5). The list is then pushed up (67) and interrupts are added to the bottom of the list as they are received. Processing the list in this manner ensures that requests are attended to in the sequence of their arrival. Figures 30(a) and 30(b) (Page 124) use 4 terminals to show the queue at the end of a request and after a terminal has been chosen for attention.

(2) EVAL

Subroutine EVAL is used by the main program for the following purposes:

(i) to read messages from (18) and write messages (17) to the

terminals. Each message is stored in an array whose maximum size is 480 bytes and hence 480 characters.

(ii) to separate the reply into its components: code, function, identification numbers (20), and, whenever necessary, to place these parts into their respective stacks (23). For example, a member is given a base display from which he selects the functions he wishes to perform by entering the number associated with each function. The subroutine reads the response, searches it for these numbers, and where stacking is required, it calls subroutine STACK to create the stacks to be used by the main program.

When first called, a flag is set to 1 which indicates that the subroutine is to search the input message for an identification number and a code. When these values are found, the flag is reset to 0 and control returns to the main program.

After the main program makes the validity checks, subroutine EVAL is called again. This time it writes the base display and scans the response for the functions requested and accompanying identification numbers. Subroutine STACK is called to stack the numbers.

In a reply to the initial display, a user may enter his identification number and forget to enter his code. Subroutine EVAL responds to this by:

(a) displaying a message indicating the error and requesting a re-entry of the code,

(b) resetting all the relevant counters and flags,

(c) reading the response,

(d) processing continues.

When both the identification number and code have been omitted, the control system returns to the initial display.

(3) CHEK

In a system containing sensitive information it is desirable to provide management with the tools for checking at any time whether attempts are being made to access their data. This facility is provided by the status check function, a function which gives each user a display informing him of the total number of times his terminal has successfully and unsuccessfully produced its code.

Subroutine CHEK uses the identification number to calculate the record required in the status file. It retrieves the record (45) and places the information into an output array (46) to be displayed later by the main program (47).

The Credit Bureau can check the status of several members and itself, whereas a member can check only his own status.

(4) CHEKUP

Whenever the System is used, subroutine CHECKUP is called to

update the status file (24). It finds the record to be updated and then increments the count of successful or unsuccessful attempts as indicated by a flag which is set to 1 or 2 by the main program. If the flag is equal to 1, then the number of unsuccessful times is incremented. If it is equal to 2, the number of successful times is increased.

When the field containing the number of unsuccessful accesses is increased, a check is made to determine whether its total is a multiple of 10 (25). If so, subroutine ATTEND is called (26) to write a message on the console typewriter indicating this condition. Each total on attaining a value of 9999 is reset to 0.

(5) UPDATE

Subroutine UPDATE is called by the main program to update a member's Accounts Receivable record in the Accounts Receivable file. The member's name and the beginning and ending address of his data are obtained from his directory record by the main program (52) and are passed to the subroutine.

UPDATE prepares the output for the terminal in the following manner:

(i) it places the member's name in the first line of the output buffer,

(ii) it retrieves the records (53), calls on subroutine SHCODE (54) to unscramble the data and then fills in lines 2 to 11

of the output buffer (55),

(iii) it indicates in line 12 whether the data is to be continued on the next screen,

(iv) it writes the full screen (12 lines) of information to the terminal (56),

(v) it reads the updated records (57).

The Credit Bureau can update the accounts of several members whereas a member can update only his own records. Any attempts by members to update data other than their own will result in a retrieval of their own data.

(6) GEN

Since all the files are stored in the Direct Method of Organization and the identification number is used as the key in calculating record numbers, it is essential that all such numbers read from the terminal be verified before usage in calculations. Subroutine GEN verifies these numbers in response to a call by subroutine STACK (9, 21). It is given a 6-digit number and returns a check digit to be compared with the one received in the 7-digit identification number.

Subroutine GEN is also used in response to the selection of the Code Generator (function 4) for assigning identification numbers and codes to new members of the Bureau. In the numbering system used (Page 87) the Bureau assigns to a new member the next higher 6-digit number

available. This is passed to subroutine GEN which appends it with a check digit to give the 7-digit identification number. Subroutine GEN also generates the 4-character code to be assigned to the member, and the scrambled code for storage in the member's directory record (37, 38).

The following example helps to illustrate how subroutine GEN, given a 6-digit number, will generate the identification number and the 4-character code.

6 Digit No. = 1 0 0 1 1 2

A	B	C	D	E	F
1	0	0	1	1	2
1	2	3	4	5	6

(i) Identification Number

Check Digit = MOD ($3(\sum^{i}$ even numbered digits) + ($\sum$ odd numbered digits), 10)

= MOD (3(0+1+2) + (1+0+1), 10)

= 1

Therefore Identification Number = 1 0 0 1 1 2 1

(ii) 4-character code

The table used in generating this code is the same as shown in Figure 16 on Page 71.

---

^{i} ' $\sum$ ' denotes the sum of

1st character MOD ((A+E+F), 32) + 3

= MOD ((1+1+2), 32) + 3 = 7

the 7th character in the table is 'Y' which becomes the first character of the code.

2nd character MOD (2 (A+B+C+D+E+F) , 32) + 1

= MOD (2(1+0+0+1+1+2), 32) + 1

= 11

the 11th character in the table is 'T' which becomes the 2nd character in the code.

3rd character MOD (MOD(6 digit No, 100) , 32) + 1

= MOD (MOD (100112, 100) , 32) + 1

= MOD (12, 32) + 1

= 13

the 13th character in the table is 'U' which becomes the 3rd character in the code.

4th character MOD (A+B+C+D+E+F, 32) +1

= MOD (1+0+0+1+1+2, 32) + 1

= 6

the 6th character in the table is '=' and so this is the 4th character in the code.

Putting the characters together the 4-character code is YTU=.

Thus the identification number allotted to the member is 1001121 and his code is YTU=.

Subroutine SHCODE is then called by subroutine GEN to scramble the 4-character code for storage in the directory (38). Finally the identification number and codes are placed in an output buffer (39) for display to the terminal.

Code Generator's Performance. While it is not essential for each user to be assigned a unique code, repetition should be sufficiently sparse so as to eliminate the following situation: Consider an employee whose duty involves using the System and who leaves his present employer and goes to work with another company where he is not permitted to use the terminal. Suppose that somehow he gets to the terminal and is trying to access his new employer's data. After trying several possible codes, he decides to try the one of his ex-employer. What is the probability that his new employer has the same code as the previous one?

Intuitively, one can deduce that the employee's chance of being successful is somehow related to the number of users sharing the same code. In order to calculate this probability, the code generating performance of subroutine GEN must be analyzed. Several codes were generated between specific limits of 6-digit numbers and a sum was formed for the code which was repeated most frequently (Table 1).

This sum gives the number of times that code was repeated. A success will be most likely when the percentage of users sharing a code is greatest and hence a probability calculated by using cases 2, 3 or 4 will give the maximum probability of a success. Case 4 is used in the calculation; the System consists of 15,000 members,[i] 5 of whom have the same code. Assuming that the first employer is one of these 5, there will be 14,999 members left, 4 of whom have that code. The probability that the second employer is one of these is 4/14,999 = .00027. Hence, the probability that the employee will be able to access his present employer's data by using the code of his previous employer is .00027.

This theory can be extended to calculate the probability of any set of users, chosen at random, having the same code. In general, if n is the number of users sharing a code, N the total number of users, then the probability of an q users having the same code is

$$\prod_{i=1}^{q} \frac{n-(q-i)}{N-(q-i)}$$

where $\prod_{i=1}^{q}$ is the product of the expression from i=1 to i=q

For example, what is the probability that any 2 users will have the same code in a System comprising of 5000 users? By reference to Table 1, it is seen that this is Case 2. Hence, n = 2.

---

[i] It is assumed here that each member has only one terminal. The probability will be smaller otherwise.

TABLE 1

CODE GENERATOR'S PERFORMANCE

Case	Limits of 6-digit Numbers[i]	No. of codes generated	Maximum times a code was repeated	% users having same code
1	100000 - 100999	1,000	0	.000
2	200000 - 204999	5,000	2	.040
3	350000 - 359999	10,000	4	.040
4	480000 - 495999	15,000	6	.040
5	610000 - 629999	20,000	7	.035

[i] The full identification number (7 digits) is not used here, since the 7th digit is not considered in deriving the code.

$N = 5000 \qquad n = 2 \qquad q = 2$

$$\prod_{i=1}^{2} \frac{n-(q-i)}{N-(q-i)}$$

$$= \frac{2-(2-1)}{5000-(2-1)} \times \frac{2-(2-2)}{5000-(2-2)}$$

$$= \frac{1}{4999} \times \frac{2}{5000}$$

$$= .00000008$$

(7) STACK

A member's selections from the base display are passed to subroutine STACK for appropriate stacking (23). Functions with accompanying identification numbers, for example the credit report function, are placed into a 2-dimensional array; the first and second indices containing the function and identification numbers respectively.

All identification numbers are verified by a call to subroutine GEN (21), and invalid ones are stored in an array to be displayed later by the main program (22).

(8) SHCODE

The control proposed for computer room protection requires a transformation of critical data before storage. Subroutine SHCODE scrambles the input data by making a character by character comparison to the elements of a 32-character table, replacing each character of

the input by one in the table (Page 71). The table contains alphabetic characters and some special symbols such as +, -, and /.

SHCODE is passed a code by subroutine GEN and scrambles it for storage in the directory (38). It is called by (i) the main program to scramble a code sent by a terminal for comparison with the one in the directory record (12) and (ii) subroutine UPDATE to unscramble a member's Accounts Receivable records for output to the terminal (54).

(9) ATTEND

Subroutine ATTEND is called by subroutine CHECKUP to write a message on the console typewriter (26) whenever a terminal has failed to produce the correct code 10 times (25). The format of this message is

XXXXXXX HAS FAILED 10 TIMES * * *

where

XXXXXXX is replaced by the user's identification number.

On receipt of this message, management at the remote terminal installation would be informed (Page 67) as the frequencey of its occurrence may be an indication that attempts are being made to access their data by an unauthorized person.

(10) QUEUE

Subroutine QUEUE provides the facility for attending to interrupts

received while the system is busy. It stacks these interrupts into a waiting list. The subroutine checks whether the device causing the interrupt is already in the waiting list, and if not inserts its address at the bottom of the list (Figure 30). In either case a message is displayed to the terminal informing the operator that the System is busy and giving his position on the queue. On completing a transaction this list is consulted and the member waiting longest (the member at the top of the list) is given attention.

(II) OPEN, POLLIN and POLOUT

Subroutine OPEN, POLLIN and POLOUT are standard routines provided by the Institute for Computer Studies, University of Manitoba.[15,16]

Subroutine OPEN prepares the terminals for communication (2). POLLIN and POLOUT are used by the system's routines for reading messages from and writing messages to the terminals.

III. SYSTEM EXAMPLE

The system performance is described with reference to the exhibits included in Appendix C. These are the displays written to, and the responses read from the terminals (Units 201 and 202) in testing the features of the control system. Though the start symbol and the cursor appear in the displays on the CRT screen, they are not present in the exhibits because:

(i) these symbols are not in the character set of the printer, and

(ii) when a message is read from the CRT, the softward reads only the data between these two symbols, and does not include the symbols themselves. [11]

The characteristics of the user environment are as follows:

	User 1	User 2
Name	Bureau Office	ABC Company
Device Address	201	202
Id. No.	1000001	1001121
Code	LJC#	YTU=

The Exhibits

Exhibit 1 - The computer facility has now begun attending to the Credit Information System and hence the initial display is written to all the terminals.

Exhibit 2 - The response received from the terminal whose address is 201 (Unit 201).

Exhibit 3 - The identification number is not a valid one and the terminal operator is asked to wait for the initial display before trying again.

Exhibit 4 - The initial display written to Unit 201.

Exhibit 5 - The response read from Unit 201.

Exhibit 6 - The code was not entered in a response to the initial display. The operator is given the benefit of the doubt and is asked to re-enter it.

Exhibit 7 - The code received from Unit 201.

Exhibit 8 - The code given is incorrect. The operator is given a second opportunity to produce the correct one.

Exhibit 9 - The code produced by Unit 201.

Exhibit 10 - The code has been incorrect for the second time. This message is written to the terminal and the job is terminated.

Exhibit 11 - The initial display is written back to Unit 201.

Exhibit 12 - A response read from Unit 202.

Exhibit 13 - The identification number 'checks' but device 202 is not found in that user's list of devices in his directory record. This message is written to the terminal and the job is terminated.

Exhibit 14 - The initial display is written back to Unit 202.

Exhibit 15 - A reply from Unit 202.

Exhibit 16 - The security checks have been met and the base display is written to the terminal so that the operator may make his selections.

Exhibit 17 - The selections read from the terminal. Note that function 4 was not in the base display yet it was

requested (Exhibit 22).

Exhibit 18 - The output for function 1.

Exhibit 19 - While the System was attending to Unit 202, Unit 201 interrupted for attention. Unit 201 is placed in a queue and this message informs the operator of his position in this list.

Exhibit 20 - The output for selection 2, The Accounts Receivable records for the ABC Company.

Exhibit 21 - The Accounts Receivable read back for storage.

Exhibit 22 - The user requested function 4 but Unit 202 is not allowed to carry out this function.

Exhibit 23 - The System is now finished attending to terminal 202 and writes the initial display back to the terminal.

Exhibit 24 - Unit 201 is found in the queue and the initial display is written to that terminal.

Exhibit 25 - The response read from Unit 201.

Exhibit 26 - The base display for Unit 201.

Exhibit 27 - The user wants to perform function 1 for the 3 members listed, function 2 for member 1001134 and function 4 for 2 prospective members.

Exhibit 28 - The output for selection 1. Note that though a status check was requested on 3 members only 1 member appears in the display (Exhibit 33).

Exhibit 29 - A part of the Accounts Receivable records for member 1001134 is presented for updating.

Exhibit 30 - The update records are read back.

Exhibit 31 - The remainder of the Accounts Receivable records are written to terminal 201.

Exhibit 32 - The updated records are read back.

Exhibit 33 - The output for selection 4.

Exhibit 34 - In selecting function 1, the operator entered 2 incorrect identification numbers. The message informs the terminal operator that the 2nd or 3rd identification numbers associated with function 1 are invalid.

Exhibit 35 - This display is presented so that the operator may re-enter the identification number.

Exhibit 36 - The response to Exhibit 35.

Exhibit 37 - The output for the selections made in Exhibit 36.

Exhibit 38 - The transaction is now complete for that user and the initial display is written on the terminal.

FIGURE 29

LINKAGE OF SUBROUTINES

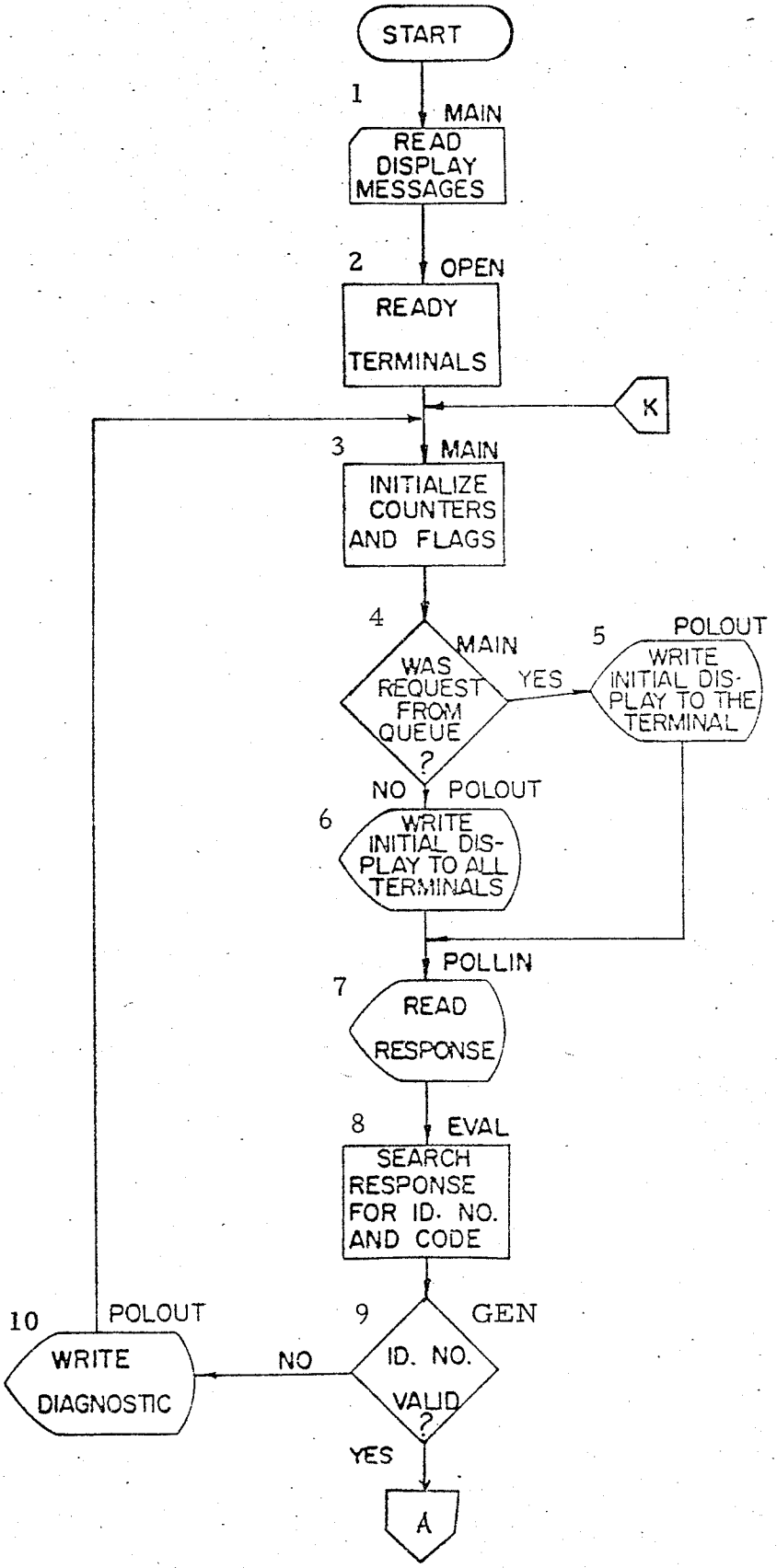


FIGURE 29 (Continued)

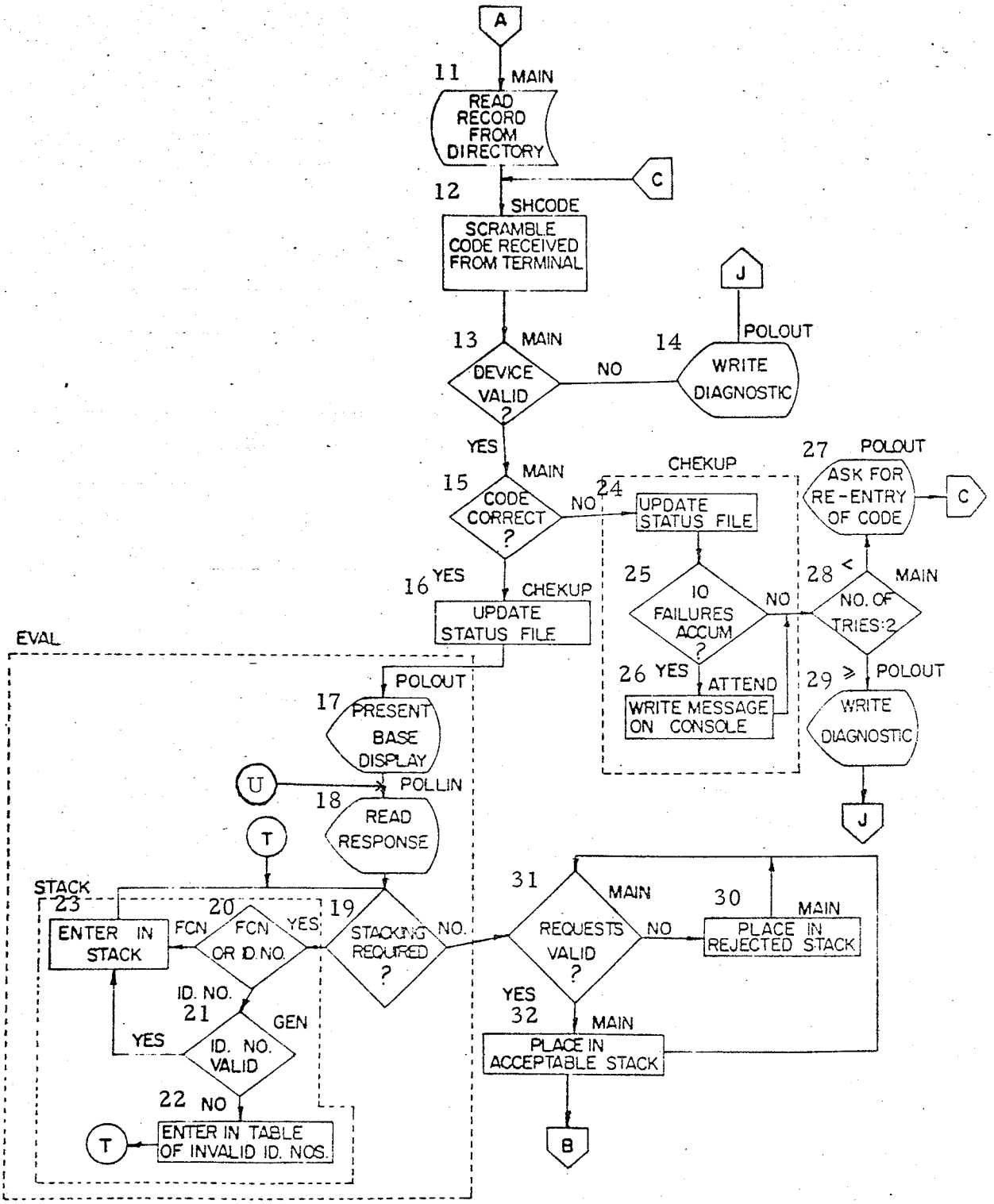


FIGURE 29 (Continued)

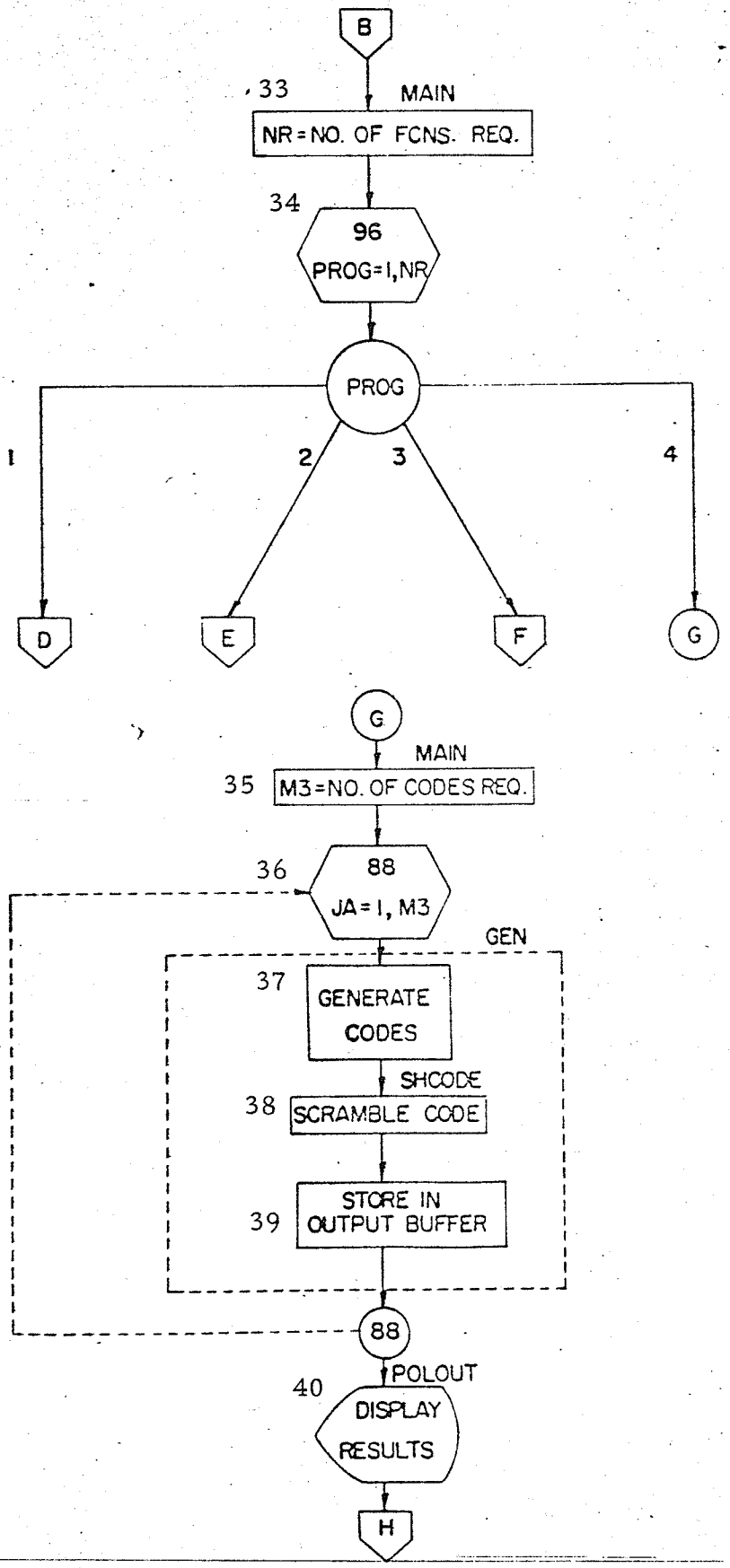


FIGURE 29 (Continued)

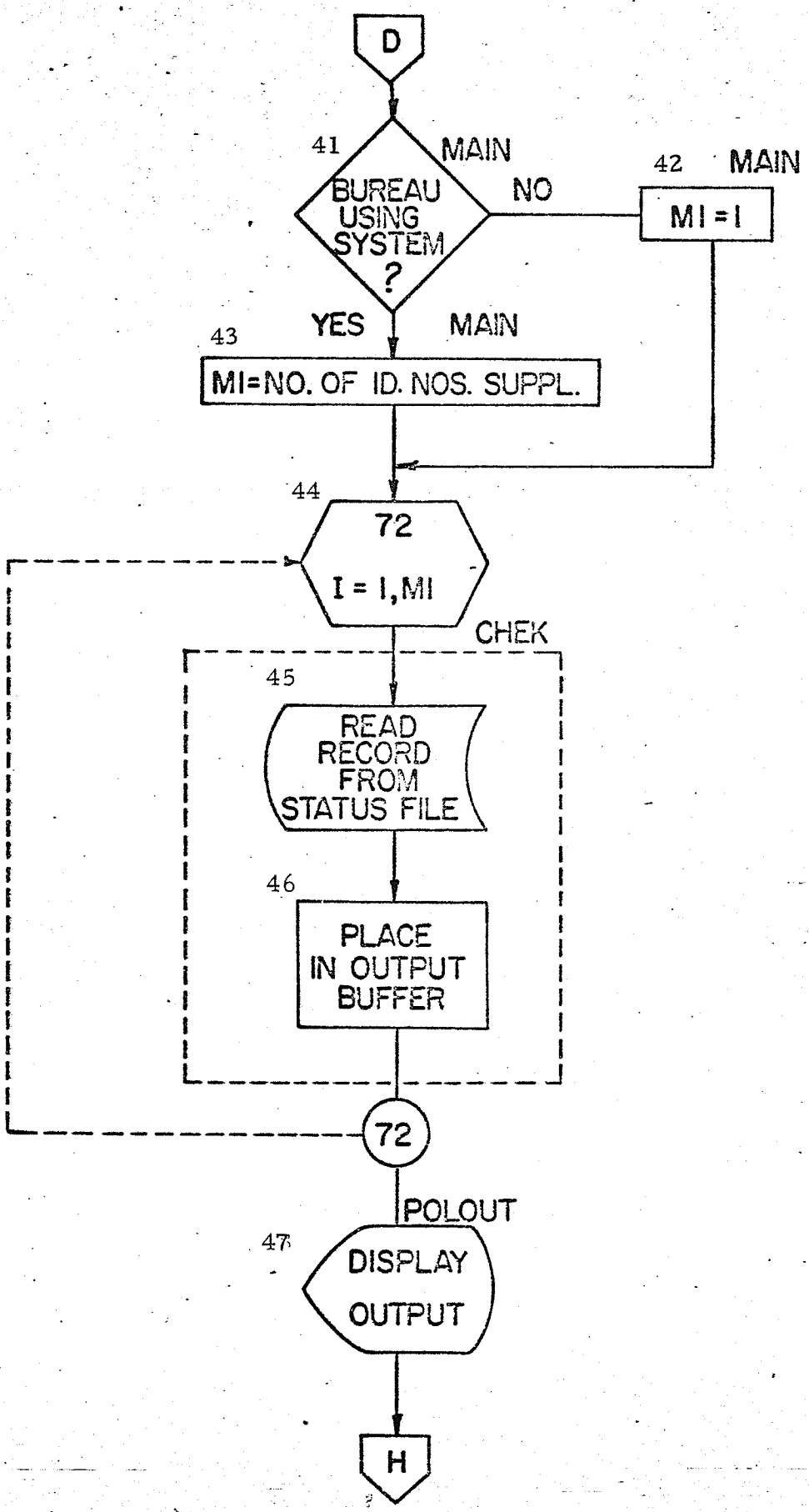


FIGURE 29 (Continued)

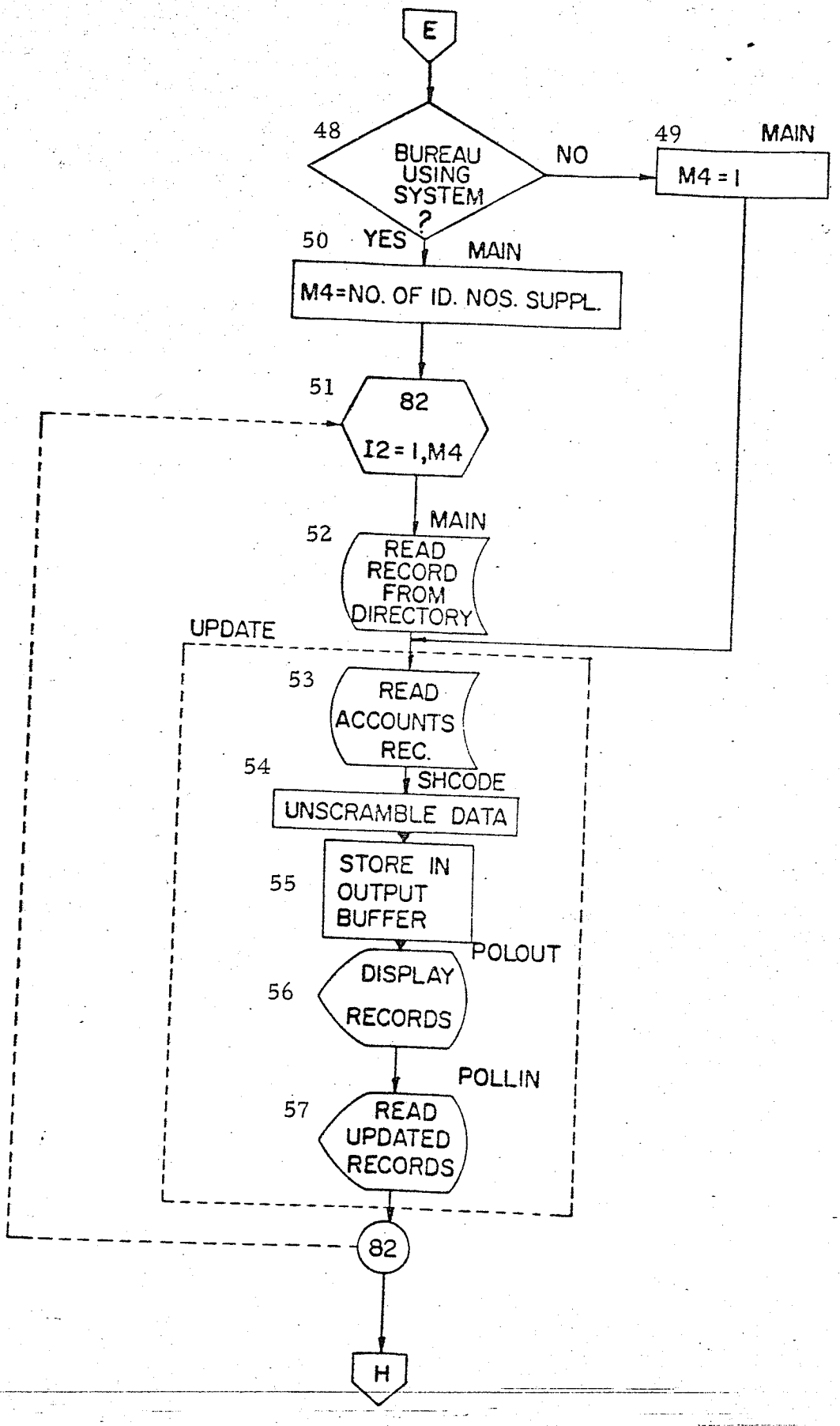
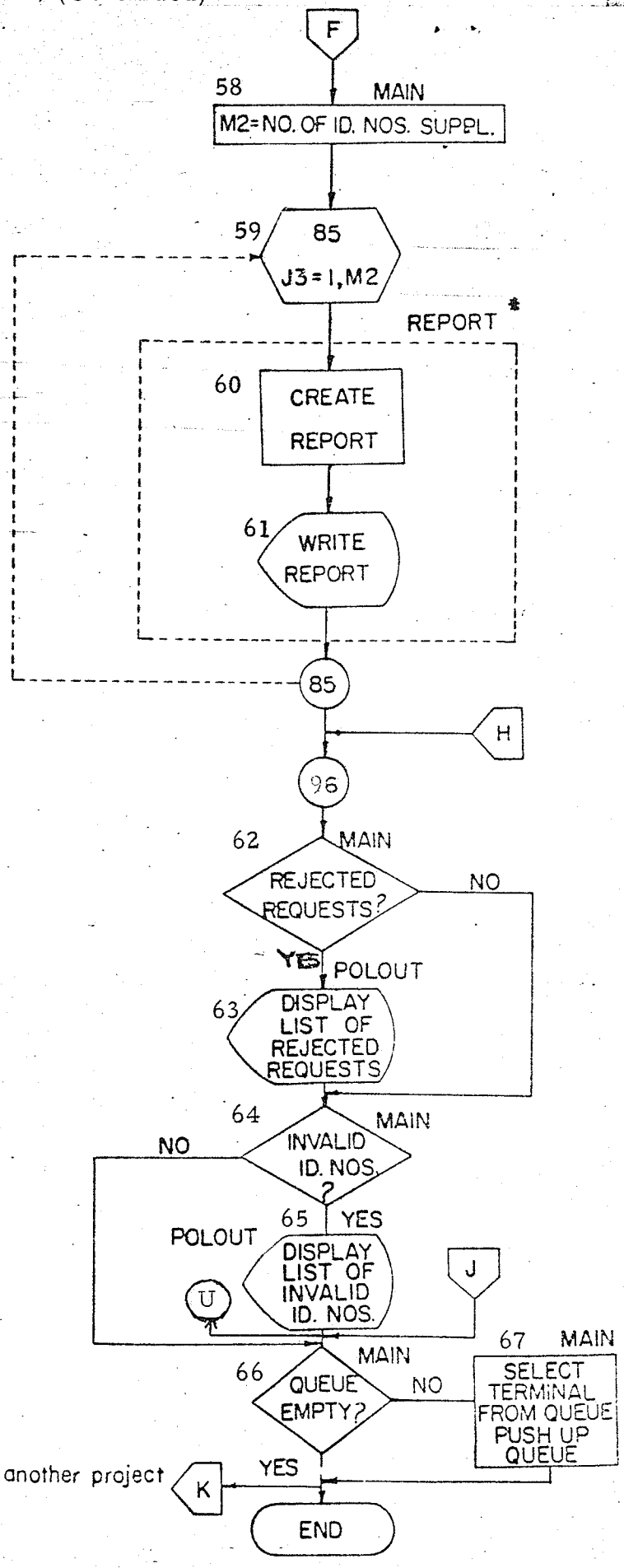


FIGURE 29 (Continued)

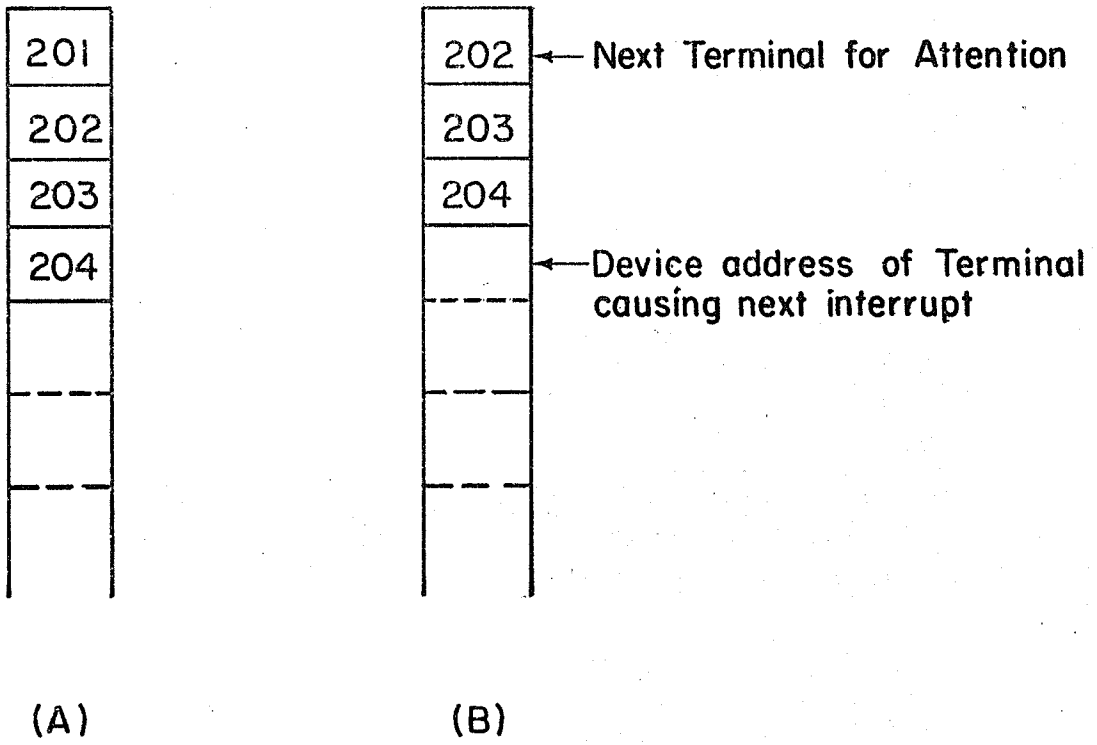


*This routine is to be provided by another project

FIGURE 30

(A) QUEUE AT THE END OF A REQUEST

(B) QUEUE AFTER A TERMINAL HAS BEEN SELECTED



CHAPTER VII

CONCLUSIONS

The control system developed provides protection at the organization level for the security of confidential data and programs. The system allows companies to have full access privileges to their own files but may obtain only selected information from the rest of the data bank. Companies can allot chosen personnel to carry out the functions provided by the System or can permit all employees to execute the functions.

Several levels of protection are provided. Firstly, the identity of the terminal operator is checked. Secondly, the hardware address of the terminal is verified and the control system ensures that both the user and the terminal remain within their domain of functions. Lastly, all critical data is stored in a scrambled form so that anyone obtaining a copy of the data sets will have to decode them.

Although the system was implemented to provide protection at the organization level, it can be extended to provide similar protection at the individual level. This can be done by reconstructing the directory so that it will contain a record for each individual, consequently treating each individual as a company. The criterion for constructing the base display will now be the individual's identification number, rather than the device address as in the case of a company.

The advantage of using a code system for identifying a terminal operator should be noted. In many control systems, the terminals chosen are those which require a medium such as a card or a key to activate the System and to identify the operator. In these systems, the loss of that medium will not only 'lock out' the individual from using the System, but will also leave the System open to the finder of the card or key. In addition, alternative procedures have to be developed so that personnel may still be able to use the System if they forgot or misplaced the medium. This situation cannot arise in a code system. The most one can do is forget his company's code, and this can be readily obtained by consulting management.

It may be argued that using a code system will suffer the disadvantage of requiring both main and auxiliary storage space, while a card or key system does not. However, the space occupied by the code in a directory record is very small, being only two fifty-fifths of the entire record. This consideration would not be a major one, unless auxiliary storage becomes a limiting factor in the System. Regarding main storage, the code occupies only four bytes, even if more than one directory record is required to execute a transaction. In such a case, each code is read, as it becomes needed, into the same storage area occupied by the previous one.

The control system can be improved by the following major developments:

1. The method of identifying a terminal operator by a computer-terminal conversation, requires operator, CPU and data transmissi on time. This identification can be taken over by a well designed terminal. Hence, the development of a terminal which can recognize either voice, facial features or finger prints would be a definite asset to security controls.

2. It is estimated that in the Credit Information System a company can have up to, or more than, eighteen hundred records in the Accounts Receivable file. The scrambling and unscrambling of this data by software will make the updating of the Accounts Receivable file a slow process. Also, terminal operators may become impatient waiting for displays. Hence the possibility of utilizing hardware scrambling equipment should be investigated. On the other hand, data scrambling may be avoided altogether if the PASSWORD protection feature of IBM's Operating System/360 can be altered so that the terminal operator will have to furnish the password, instead of the computer operator. This being so, computer operators could not know the passwords and hence could not access the files.

APPENDIX A

```
      PROGRAM TO CREATE DIRECTORY ON DISK
      - RECORDS ARE STORED IN DIRECT ACCESS

      INTEGER * 2 DEVADR(5), LIST(10)
      INTEGER CODENM,NAME(5), OPTINF(10), UPBEG,UPEND
      DEFINE FILE 9(9,28,U,IJ)
      IJ = 1

         READ DIRECTORY FROM CARDS ONTO DISK

    1 READ (5,51,END=4) IDNO,DEVADR,CODENM,NAME,LIST,UPBEG,UPEND,
     1LINK,OPTINF
      WRITE (9'IJ)       IDNO,DEVADR,CODENM,NAME,LIST,UPBEG,UPEND,
     1LINK,OPTINF
      GO TO 1
    4 IM = IJ - 1
      IJ = 1
      WRITE (6,50)
      WRITE (6,54)

         MAKE A LISTING OF THE DIRECTORY

      DO 8 I = 1,IM
      READ (9'IJ)        IDNO,DEVADR,CODENM,NAME,LIST,UPBEG,UPEND,
     1LINK,OPTINF
      WRITE (6,52)       IDNO,DEVADR,CODENM,NAME,LIST,UPBEG,UPEND,
     1LINK,OPTINF
    8 CONTINUE
   50 FORMAT ('1'50X,'DIRECTORY'//' '31X,' SHIFTED')
   51 FORMAT (I7,5I4,6A4/13I4/10A4)
   52 FORMAT (' 'I7,1X,5(I4,1X),A4,1X,5A4,1X,10(I2,1X),2(I4,1X),
     13X,I4,5X,2A4/8A4/)
   54 FORMAT ('  ID. #     DEVICE ADDRESS        CODE        NAME
     1                 LIST              A.R. ADDR    LINK     OPT INF'/
     2)
      CALL EXIT
      END
```

#	DEVICE ADDRESS				SHIFTED CODE	NAME	LIST										A.R. ADDR		LINK	OPT INF
)01	201	0	0	0	0	=MJL BUREAU OFFICE	1	2	3	4	0	0	0	0	0	0	0	0	0	
121	202	0	0	0	0	U-*> THE ABC COMPANY	1	2	3	0	0	0	0	0	0	0	1	4	0	
134	203	0	0	0	0	/AFZ ADLER ASSOC.	1	2	3	0	0	0	0	0	0	0	5	10	0	
147	204	0	0	0	0	-OOW CANADIAN IND. LTD.	2	3	0	0	0	0	0	0	0	0	11	15	1	
150	210	0	0	0	0	*%@+ HAVELL LTD.	1	2	3	0	0	0	0	0	0	0	16	18	0	
163	206	0	0	0	0	FRBA RAND-SELLERS LTD.	1	2	3	0	0	0	0	0	0	0	19	23	0	
176	207	0	0	0	0	U+*> HANDEZ BROS	1	2	3	0	0	0	0	0	0	0	24	25	0	
189	208	0	0	0	0	//FZ MUTT COMPANY	1	2	3	0	0	0	0	0	0	0	0	0	0	
192	209	0	0	0	0	-+OW A.G. LETOURNE	1	2	3	0	0	0	0	0	0	0	0	0	0	

```
      PROGRAM TO CREATE ACCOUNTS RECEIVABLE FILE FOR ALL MEMBERS
                                                                    131.
      INTEGER  OUTDEV,INFO(20)
      INTEGER ALFA(32)/'C','#','J','L','M','=','Y','X','Q','I','T','>',
     1'U','Z','/','W','-','+','*','A','F','E','O','V','@','%','B','P',
     2'R','H','D','K'/
      DEFINE FILE 8(30,80,L,IREC)
      IREC=1
      INDEV=5
      OUTDEV=6
      NC = 0
      ISHIFT = 3
      WRITE(OUTDEV,50)
      WRITE (OUTDEV,54)

         READ FILE FROM CARDS ONTO DISK

    2 READ(INDEV,51,END=4)INFO
      WRITE (OUTDEV,52) INFO
      NC = NC + 1

         SCRAMBLE  DATA

      DO 3 I = 2,6
      INF = INFO(I)
      CALL SHCODE (INF,ALFA,ISHIFT,NEWCDE)
      INFO(I) =NEWCDE
    3 CONTINUE
      WRITE(8'IREC)INFO
      GO TO 2
    4 END FILE 8
      WRITE (OUTDEV,53)
      IREC = 1

         MAKE A LISTING OF THE FILE

      DO 6 I = 1, NC
      READ (8'IREC) INFO
      WRITE (OUTDEV,52) INFO
    6 CONTINUE
   50 FORMAT('1',30X,'ACCOUNTS  RECEIVABLE'//)
   51 FORMAT ( 20A4)
   52 FORMAT (' '20A4)
   53 FORMAT (' '/// ' '20X,' ACCOUNTS RECEIVABLE AFTER SCRAMBLING AND S
     1TORAGE'//)
   54 FORMAT (' '8X,'NAME',32X,'AMOUNT',3X,'CURRENT',5X,'30-60',4X,
     *    '61-90',3X,'91-120',3X,'OVER 120',/46X,'DUE',7X,'DUE',
     $7X,'DAYS',5X,'DAYS',4X,'DAYS',5X,'DAYS'/)
      CALL EXIT
      END
```

```
      SHCODE SCRAMBLES DATA FOR THE ACCOUNTS RECEIVABLE FILE

      SUBROUTINE SHCODE (CODE,ALFA,SHIFT,NEWCDE)
      INTEGER CODE,SHIFT,ALFA(32),INTCDE(4),BLANK/'  '/
      LOGICAL * 1 LGCDE(4),FLWD(16)
      EQUIVALENCE (ICODE,LGCDE(1)), (FLWD(1),INTCDE(1))
      ICODE = CODE

         PLACE BLANKS INTO WORK  AREA

      DO 2 I = 1,4
      INTCDE(I) = BLANK
    2 CONTINUE

         PLACE NAME INTO NEW AREA - ONE CHARACTER PER BYTE

      DO 3 I = 1,4
      J = 4 * I - 3
      FLWD(J) = LGCDE(I)
    3 CONTINUE

         PERFORM SCRAMBLING

      DO 12 J = 1,4
      DO 4 I = 1,32
      INDX = I
      IF ( INTCDE(J) .EQ. ALFA(I)) GO TO 8
    4 CONTINUE
      GO TO 12
    8 NINDX = INDX + SHIFT
      IF ( NINDX .GT. 32) NINDX = NINDX - 32
      IF (NINDX .LT. 1 ) NINDX = NINDX + 32
      INTCDE(J) = ALFA(NINDX)
   12 CONTINUE

         READY SCRAMBLED DATA FOR OUTPUT FILE

      DO 16 M = 1, 4
      J = 4 * M - 3
      LGCDE(M) = FLWD(J)
   16 CONTINUE

C
C        PLACE SCRAMBLED DATA INTO OUTPUT AREA
C
      NEWCDE= ICODE
      RETURN
      END
```

ACCOUNTS RECEIVABLE

NAME	AMOUNT DUE	CURRENT DUE	30-60 DAYS	61-90 DAYS	91-120 DAYS	OVER 120 DAYS
LAMBERTE HARDWARE	668.44	668.44				
A.G. LETOURNE	1381.33	1360.59	20.74			
MUTT COMPANY	6135.03	6118.03	17.00			
TEREAU RACINE LTD.	3685.04	1354.20	2299.37			
WESTOURN	18197.54	17806.10	37.03			
CUCHRAN DONLUP	100.66	83.36	13.50	3.80		
J.E. ALLEN	8634.24	8634.24				
EMCOE LTD	1322.46	841.64	409.55	71.27		
MUTT COMPANY	100.66	83.36	13.55	3.80		
A.G. LETOURNE	1381.00	1360.26	20.74			
MCOE LIMITED	2303.16	2285.38	17.78			
HASTEE PLUMBING	6812.21	6728.59	83.62			
HODSUN PLUMBING	2447.94	2447.94				
MUTT COMPANY	2942.86	2267.97	674.89			
A.G. LETOURNE	21764.85	14585.82	7173.05	13.88		
OSHAWA PLBG R HTG	118.77	118.77				
QUALITY UTILITY	8506.87	8506.87				
TURRET SUPPLY	2108.80	1098.04	990.70	20.06		
WESTBURNE WPG.	8941.11	7026.34	1123.46	791.31		
WESTON SUPPLIES	2411.16	1858.38	552.78			
J.H. ASHDUNE	5043.31	4853.50	189.81			
A.G. LETOURNE	1448.34	1448.34				
MUTT COMPANY	10557.62	10338.59	219.03			
P.E. JUNSON	12095.91	6314.18	5766.74			
A.G. LETOURNE	140.72	140.72				

ACCOUNTS RECEIVABLE AFTER SCRAMBLING AND STORAGE

YOXH@KZ@ COK#*OK@	668.44	668.44				
O.G. Y@Z%WKN@	1381.33	1360.59	20.74			
XWZZ L%XDONI	6135.03	6118.03	17.00			
Z@K@OW KOLUN@ YZ#.	3685.04	1354.20	2299.37			
*@SZ%WKN	18197.54	17806.10	37.03			
LWLCKON #%NYWD	100.66	83.36	13.50	3.80		
=.@. OYY@N	8634.24	8634.24				
@XL%@ YZ#	1322.46	841.64	409.55	71.27		
XWZZ L%XDONI	100.66	83.36	13.55	3.80		
O.G. Y@Z%WKN@	1381.00	1360.26	20.74			
XL%@ YUXUZ@#	2303.16	2285.38	17.78			
COSZ@@ DYWXHUNG	6812.21	6728.59	83.62			
C%#SWN DYWXHUNG	2447.94	2447.94				
XWZZ L%XDONI	2942.86	2267.97	674.89			
O.G. Y@Z%WKN@	21764.85	14585.82	7173.05	13.88		
%SCO*O DYHG K CZG	118.77	118.77				
>WOYUZI WZUYUZI	8506.87	8506.87				
ZWKK@Z SWDDYI	2108.80	1098.04	990.70	20.06		
*@SZHWKN@ *DG.	8941.11	7026.34	1123.46	791.31		
*@SZ%N SWDDYU@S	2411.16	1858.38	552.78			
=.C. OSC#WN@	5043.31	4853.50	189.81			
O.G. Y@Z%WKN@	1448.34	1448.34				
XWZZ L%XDONI	10557.62	10338.59	219.03			
D.@. =WNS%N	12095.91	6314.18	5766.74			
O.G. Y@Z%WKN@	140.72	140.72				

```
      PROGRAM TO CREATE THE INITIAL STATUS FILE
      INTEGER CHEC(9)
      DEFINE FILE 7 (9,9,U,IR)
      IR = 1

         READ FILE FROM CARDS ONTO DISK

    1 READ (5,50,END=4) CHEC
      WRITE (7'IR) CHEC
      GO TO 1
    4 IZ = IR - 1
      WRITE (6,52)
      WRITE (6,53)

         MAKE A LISTING OF THE FILE

      IR = 1
      DO 5 I = 1,IZ
      READ (7'IR) CHEC
      WRITE (6,51) CHEC
    5 CONTINUE
   50 FORMAT (9A4)
   51 FORMAT (' '2A4,1X,5A4,1X,A4,6X,A4/)
   52 FORMAT ('1'9X,'INITIAL  STATUS  FILE'//)
   53 FORMAT ( ' ID. NO.       NAME                FAIL    SUCCESS'/)
      CALL EXIT
      END
```

INITIAL STATUS FILE

NO.	NAME	FAIL	SUCCESS
00001	BUREAU OFFICE	0	0
01121	THE ABC COMPANY	0	0
01134	ADLER ASSOC	0	0
01147	CANADIAN INC. LTD	0	0
01150	HAVELL LTD	0	0
01163	RAND-SELLERS LTD.	0	0
01176	HANDEZ BROS	0	0
01189	MUTT COMPANY	0	0
01192	A. G. LETOURNE	0	0

```
      PROGRAM TO CREATE SUB-DIRECTORY

      INTEGER * 2 DEVADR(5),LIST(10)                                    136
      INTEGER SUBLNK
      DEFINE FILE 10(1,9,U,IQ)
      IQ = 1

         READ SUB-DIRECTORY FROM CARDS ONTO DISK

    1 READ (5,50,END=2) DEVADR,LIST,SUBLNK
      WRITE (10'IQ) DEVADR,LIST,SUBLNK
      GO TO 1
    2 IQ = 1
      WRITE (6,52)
      WRITE (6,53)
         MAKE LISTING OF SUB-DIRECTORY


      READ (10'IQ) DEVADR, LIST,SUBLNK
      WRITE (6,51) DEVADR, LIST,SUBLNK
   50 FORMAT (16I4)
   51 FORMAT (' '5I4,4X,10I4,4X,I4)
   52 FORMAT ('1'27X 'SUB-DIRECTORY'//)
   53 FORMAT ('    DEVICE ADDRESS                    LIST'    29X,'SUB-LIN
     1K'/)
      CALL EXIT
      END
```

DEVICE ADDRESS	LIST	SUB-LINK
)5 0 0 0 0	1 0 0 0 0 0 0 0 0 0	0

APPENDIX B

: STATEMENT F01JAN68 2/12/69

```
'.....................................................................

JBROUTINE  ATTEND

JRPOSE

  SUBROUTINE ATTEND WRITES A MESSAGE TO THE COMPUTER OPERATOR
EVERY  10  TIMES A USER HAS FAILED TO PRODUCE THE CORRECT
CODE

ETHOD OF CALLING

  CALL ATTEND (IDH)

ESCRIPTION OF PARAMETER

  IDH  -  THE IDENTIFICATION NUMBER OF THE USER

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

  CSECT
  SAVE  (14,12)
  DS    0H
  STM   14,12,12(13) SAVE REGISTERS
  LR    12,15                    ESTABLISH  ADDRESSABILITY
  USING ATTEND,12
  B     START
                                 CONSTANTS AND STORAGE AREAS
  DS    CL8
  DC    X'4020202020202020'

  DC    1D'0'
LIST FORM OF THE WRITE TO OPERATOR
  WTO   '          HAS  FAILED  10  TIMES *****',MF=L
  DS    0F
  DC    AL2(IHB0002-*) MESSAGE LENGTH
  DC    AL2(0)
  DC    C'          HAS  FAILED  10  TIMES *****' MESSAGE
  EQU   *
  L     4,0(1)                   LOAD ADDRESS OF PARAMETER
                                 IN REG 4
  L     4,0(4)                   LOAD VALUE OF PARAMETER IN REG 4
  CVD   4,DECIMAL                CONVERT TO DECIMAL
  MVC   EDIT,EDITMSK
  ED    EDIT(8),DECIMAL+4        REMOVE LEADING ZEROES
  MVC   MSG+4(8),EDIT            MOVE NUMBER TO OUTPUT MESSAGE
  LA    1,MSG                    LOAD ADDRESS OF OUTMESSAGE IN
                                 REG 1
  WTO   ,MF=(E,(1))              WRITE MESSAGE TO OPERATOR
  SVC   35 ISSUE SVC
  RETURN (14,12)
  LM    14,12,12(13) RESTORE THE REGISTERS
  BR    14 RETURN
  END
```

```
.....................................................................  MAIN
                                                                       MAIN
MAIN  PROGRAM                                                          MAIN
                                                                       MAIN
   THE MAIN PROGRAM OPENS THE REMOTE TERMINALS, WRITES                 MAIN
THE INITIAL MESSAGE AND POLLS UNTIL A RESPONSE IS RECEIVED.            MAIN
IT ENSURES THAT THE USER IS A VALID ONE BY COMPARING HIS               MAIN
IDENTIFICATION NUMBER, CODE AND DEVICE ADDRESS AGAINST THOSE           MAIN
IN HIS RECORD IN THE DIRECTORY.  IT CHECKS THAT ALL ROUTINES           MAIN
AND FILES REQUESTED ARE AVAILABLE TO THE USER AND SENDS APPR-          MAIN
OPRIATE OUTPUT TO THE TERMINALS.                                       MAIN
                                                                       MAIN
   NEW ROUTINES CAN BE ADDED TO THE PROGRAM BY CHANGING                MAIN
ONE STATEMENT AS INDICATED IN THE DOCUMENTATION.  NEW MEMBERS          MAIN
ARE ACCOMODATED BY EXPANDING THE DIRECTORY.                            MAIN
                                                                       MAIN
                                                                       MAIN
DESCRIPTION OF ARRAYS                                                  MAIN
                                                                       MAIN
MESS1, MESS2, MESS3, MESS4, MESS5, MESS6, MESS7, MESS8, MESS9,         MAIN
MESS10                                                                 MAIN
          -  CONTAIN DISPLAY MESSAGES TO THE TERMINALS                 MAIN
PROGNO    -  A LIST OF THE ROUTINES TO BE USED IN ANSWERING            MAIN
             THE REQUEST                                               MAIN
IDNTNO    -  A TEMPORARY STORAGE AREA FOR NUMBERS AND CODES READ       MAIN
             FROM THE TERMINALS                                        MAIN
DEVADR    -  A LIST OF THE ADDRESSES OF THE USER'S TERMINALS           MAIN
OPTINF    -  OPTIONAL INFORMATION                                      MAIN
REPSTK    -  A STACK INTO WHICH ALL REQUESTS ARE PLACED                MAIN
REJCT     -  A LIST OF ROUTINES WHICH A USER HAS ATTEMPTED TO          MAIN
             USE BUT WHICH ARE UNAVAILABLE TO HIM                      MAIN
CDEDSP, X-   ARRAYS CONTAINING OUTPUT INFORMATION FOR THE TERMINAL    MAIN
MESS      -  AN AREA INTO WHICH ALL RESPONSES FROM THE TERMINALS       MAIN
             ARE PLACED                                                MAIN
                                                                       MAIN
DESCRIPTION OF COUNTERS AND INDICES                                    MAIN
                                                                       MAIN
   M1, M2, M3, M4 ARE INDICES USED IN STACKING REQUESTS                MAIN
   FLAG   -  INDICATES WHETHER THE NEXT WORD TO BE EVALUATED           MAIN
             IS A CODE                                                 MAIN
   ITEST  -  INDICATES WHETHER THE REQUESTS ARE TO BE PLACED IN        MAIN
             A STACK                                                   MAIN
   TIMES  -  A COUNT OF THE NUMBER OF TRIES A USER CAN MAKE            MAIN
             TO USE THE SYSTEM.  WHEN IT BECOMES GREATER THAN 2        MAIN
             A DIAGNOSTIC IS WRITTEN TO THE USER AND THE JOB IS        MAIN
             TERMINATED.                                               MAIN
   IFRAME -  A SWITCH SET IN THE PROGRAM TO INDICATE WHICH             MAIN
             DISPLAY IS TO BE PRESENTED                                MAIN
   ILL&L11-  COUNTERS FOR INCREMENTING THE NUMBER OF REJECTED          MAIN
             AND PERMISSIBLE ROUTINES                                  MAIN
                                                                       MAIN
                                                                       MAIN
SUBROUTINES CALLED                                                     MAIN
                                                                       MAIN
   OPEN                                                                MAIN
   POLOUT                                                              MAIN
   POLLIN                                                              MAIN
   EVAL                                                                MAIN
```

```
   CHEKUP                                                              MAIN
   CHEK                                                                MAIN
   UPDATE                                                              MAIN
   GEN                                                                 MAIN
   SHCODE                                                              MAIN
   QUEUE                                                               MAIN
                                                                       MAIN
                                                                       MAIN
REMARKS                                                                MAIN
                                                                       MAIN
    SUBROUTINES OPEN, POLOUT AND POLLIN ARE HELD IN THE USER'S         MAIN
 LIBRARY.  THEIR PURPOSE AND USAGE ARE AS FOLLOWS:                     MAIN
                                                                       MAIN
   SUBROUTINE OPEN                                                     MAIN
                                                                       MAIN
PURPOSE                                                                MAIN
                                                                       MAIN
   SUBROUTINE OPEN OPENS THE REMOTE TERMINALS                          MAIN
                                                                       MAIN
USAGE                                                                  MAIN
                                                                       MAIN
SUBROUTINE OPEN (R)                                                    MAIN
                                                                       MAIN
DESCRIPTION OF PARAMETER                                               MAIN
                                                                       MAIN
   R      -  HAS A VALUE OF 1 IF OPEN IS OK AND                        MAIN
                            2 IF OPEN FAILS                            MAIN
                                                                       MAIN
                                                                       MAIN
SUBROUTINE POLOUT                                                      MAIN
                                                                       MAIN
PURPOSE                                                                MAIN
                                                                       MAIN
   SUBROUTINE POLOUT WRITES OUTPUT MESSAGES TO THE TERMINALS           MAIN
                                                                       MAIN
USAGE                                                                  MAIN
                                                                       MAIN
   SUBROUTINE POLOUT ( MESS,L,U,R)                                     MAIN
                                                                       MAIN
DESCRIPTION OF PARAMETERS                                              MAIN
                                                                       MAIN
   MESS   -  MESSAGE TO BE WRITTEN TO THE TERMINAL                     MAIN
   L      -  LENGTH OF THE MESSAGE IN BYTES                            MAIN
   U      -  THE UNIT (DEVICE ADDRESS) TO WHICH THE MESSAGE IS TO      MAIN
             BE WRITTEN                                                MAIN
   R      -  A RESULTANT CODE                                          MAIN
             R = 0  NO SUCCESSFUL STARTS                               MAIN
             R = 1  READ STARTED BUT NOT COMPLETED SUCCESSFULLY        MAIN
             R = 2  READ COMPLETED SUCCESSFULLY                        MAIN
                                                                       MAIN
                                                                       MAIN
SUBROUTINE POLLIN                                                      MAIN
                                                                       MAIN
PURPOSE                                                                MAIN
                                                                       MAIN
   READS INPUT MESSAGES FROM THE TERMINALS                             MAIN
                                                                       MAIN
USAGE                                                                  MAIN
```

```
                                                                        MAIN
      SUBROUTINE POLLIN ( MESS,L,U,R)                                   MAIN
                                                                        MAIN
      DESCRIPTION OF PARAMETERS                                         MAIN
                                                                        MAIN
         MESS    -  THE INPUT MESSAGE READ FROM THE TERMINAL            MAIN
         L,U,R   -  SAME AS IN POLOUT                                   MAIN
                                                                        MAIN
     ..................................................................  MAIN
                                                                        MAIN
                                                                        MAIN
      INTEGER * 2 DEVADR(5), LIST(10)                                   MAIN
      INTEGER MESS(120),MESS1(120),MESS2(120),MESS3(120),MESS4(120),    MAIN
     1MESS6(120),MESS7(120),MESS8(120),MESS9(120),NAME(5),NWPRGS(20),   MAIN
     2PROGNO(20),IDNTNO(20),OPTINF(10),REPSTK(6,20),REJCT(20),WTSTK(10),MAIN
     3MESS10(120),XX(120),BASE(10,5),MESS11(120)                        MAIN
      INTEGER NUM(10)/'0', '1','2', '3','4', '5','6','7','8','9'/      MAIN
      INTEGER MESS5(120)/'FUNC','TION',' NUM','BER','  ','IS','UNAV',   MAIN
     1'AILA','BLE','   ',110*' '/                                       MAIN
      INTEGER ALFA(32)/'C','#','J','L','M','=','Y','X','Q','I','T','>',MAIN
     1'U','Z','/','W','-','+','*','A','F','E','O','V','@','%','B','P', MAIN
     2'R','H','D','K'/                                                  MAIN
      INTEGER X(120)/'   ',2*' ','FAIL',2*' ','SUCC','ESS',112*' '/     MAIN
      INTEGER CDEDSP(120)/' ID.','NO.',2*' ','OR C','ODE',2*' ','SCR',  MAIN
     1'CODE',110*' '/                                                   MAIN
      INTEGER OTDV,TIMES,CODENM,PROG,FLAG,U,R,SUBLNK,CHKDGT             MAIN
      INTEGER BLANK/'  '/                                               MAIN
      COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,    MAIN
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10, NTERM, NDEX, WTSTK,MD,MESS11   MAIN
      DEFINE FILE 9(9,28,U,IJ)                                          MAIN
      DEFINE FILE 10(1,9,U,IQ)                                          MAIN
      INDV = 5                                                          MAIN
      OTDV = 6                                                          MAIN
                                                                        MAIN
         READ MESSAGES                                                  MAIN
                                                                        MAIN
      READ (INDV,50) MESS1, MESS2, MESS3, MESS4, MESS6, MESS7, MESS8,   MAIN
     1MESS9,MESS10,MESS11,((BASE(I,J),J=1,5),I=1,10)                    MAIN
                                                                        MAIN
         OPEN REMOTE TERMINALS                                          MAIN
                                                                        MAIN
      R = 0                                                             MAIN
      CALL OPEN (R)                                                     MAIN
                                                                        MAIN
         INITIALIZE  COUNTERS                                           MAIN
                                                                        MAIN
      NEW = 1                                                           MAIN
      NDEX = 0                                                          MAIN
      IWAIT = 0                                                         MAIN
    2 FLAG = 1                                                          MAIN
      ITEST = 0                                                         MAIN
      L1 = 0                                                            MAIN
      M1 = 0                                                            MAIN
      M2 = 0                                                            MAIN
      M3 = 0                                                            MAIN
      M4 = 0                                                            MAIN
      ILL = 0                                                           MAIN
      L11 = 0                                                           MAIN
```

```
      IQQ = 0                                                          MAIN
      TIMES = 0                                                        MAIN
      L = 480                                                          MAIN
      MD = 0                                                           MAIN
      IFRAME = 0                                                       MAIN
    4 TIMES = TIMES + 1                                                MAIN
      GO TO (6,16), TIMES                                              MAIN
    6 INIT = 1                                                         MAIN
                                                                       MAIN
         IS THE WAITING LIST EMPTY ?                                   MAIN
                                                                       MAIN
      IF (NDEX .NE. 0 ) GO TO 10                                       MAIN
      IF ( IWAIT .NE. 0) GO TO 9                                       MAIN
      IF (NEW .NE. 1) GO TO 10                                         MAIN
                                                                       MAIN
         POLL ALL TERMINALS                                            MAIN
                                                                       MAIN
      DO  8 U = 201,202                                                MAIN
      CALL POLOUT (MESS1, 81,U,R)                                      MAIN
    8 CONTINUE                                                         MAIN
      WRITE (OTDV,55) MESS1                                            MAIN
      NEW = 0                                                          MAIN
      GO TO 12                                                         MAIN
    9 IWAIT = 0                                                        MAIN
   10 CALL POLOUT (MESS1,81,U,R)                                       MAIN
      WRITE (OTDV,56) U, MESS1                                         MAIN
   12 DO 14 LL = 1,120                                                 MAIN
      MESS(LL) = BLANK                                                 MAIN
   14 CONTINUE                                                         MAIN
                                                                       MAIN
         READ MESSAGE FROM RESPONDING TERMINAL                         MAIN
                                                                       MAIN
      CALL POLLIN (MESS,L,U,R)                                         MAIN
      NTERM = U                                                        MAIN
      WRITE (OTDV,57) U, MESS                                          MAIN
                                                                       MAIN
         EVALUATE MESSAGE                                              MAIN
                                                                       MAIN
      CALL EVAL (MESS,NUM,BLANK,IDNTNO,FLAG)                           MAIN
      INIT = 0                                                         MAIN
      GO TO 18                                                         MAIN
   16 MD = 1                                                           MAIN
      L = 98                                                           MAIN
      CALL EVAL (MESS2, NUM, BLANK,IDNTNO,FLAG)                        MAIN
      GO TO 25                                                         MAIN
   18 IDH = IDNTNO(1)                                                  MAIN
                                                                       MAIN
         IS THE ID. NO. READ A VALID ONE ?                             MAIN
                                                                       MAIN
      NEWIDH = IDH/10                                                  MAIN
      NCHK = MOD(IDH,10)                                               MAIN
      MTEST = 1                                                        MAIN
      CALL GEN (NEWIDH,CDEDSP, IST,CHKDGT,MTEST, ISHIFT)               MAIN
      MTEST = 0                                                        MAIN
      IF (NCHK .EQ. CHKDGT) GO TO 20                                   MAIN
                                                                       MAIN
         ID. NO. IS INVALID - WRITE DIAGNOSTIC TO TERMINAL             MAIN
        AND RETURN TO INITIAL DISPLAY                                  MAIN
```

```
C                                                                       MAIN
   19 CALL POLOUT (MESS8, 480,U,R)                                      MAIN
      WRITE (OTDV,56) U, MESS8                                          MAIN
      GO TO 114                                                         MAIN
C                                                                       MAIN
C        IF THE ID. NO. IS 1000001 THEN THE USER IS THE BUREAU AND      MAIN
C       SET IFRAME TO 2                                                 MAIN
C                                                                       MAIN
   20 IF (IDH .NE. 1000001) GO TO 22                                    MAIN
      IFRAME = 2                                                        MAIN
      IJ = 1                                                            MAIN
      GO TO 24                                                          MAIN
   22 IFRAME = 1                                                        MAIN
      IJ = NEWIDH - 100110                                              MAIN
C                                                                       MAIN
C        READ RECORD FROM DIRECTORY                                     MAIN
C                                                                       MAIN
   24 READ (9'IJ)        IDNO,DEVADR,CODENM,NAME,LIST,UPBEG,UPEND,      MAIN
     1LINK,OPTINF                                                       MAIN
      DO 27 I = 1,60                                                    MAIN
      MESS6(I) = BLANK                                                  MAIN
   27 CONTINUE                                                          MAIN
C                                                                       MAIN
C        CONSTRUCT THE BASE DISPLAY                                     MAIN
C                                                                       MAIN
      IDT = 0                                                           MAIN
      DO 29 J = 1,10                                                    MAIN
      IF (LIST(J) .EQ. 0) GO TO 29                                      MAIN
      DO 29 IO = 1,5                                                    MAIN
      IDT = IDT + 1                                                     MAIN
      MESS6(IDT) = BASE(LIST(J),IO)                                     MAIN
   29 CONTINUE                                                          MAIN
   25 CALL SHCODE (IDNTNO(2), ALFA,ISHIFT, NEWCDE)                      MAIN
      IF ( TIMES .EQ. 2) GO TO 37                                       MAIN
      IF ( IDH .NE. IDNO) GO TO 19                                      MAIN
C                                                                       MAIN
C        IS THE DEVICE IN THE USER'S LIST OF DEVICES ?                  MAIN
C                                                                       MAIN
   28 DO 30 II = 1,5                                                    MAIN
      IF (U .EQ. DEVADR(II)) GO TO 37                                   MAIN
   30 CONTINUE                                                          MAIN
C                                                                       MAIN
C        IF THE DEVICE ADDRESS IS NOT FOUND IN THE RECORD THEN          MAIN
C       TEST LINK.  IF THE USER HAS OTHER DEVICES FOR WHICH THERE WILL  MAIN
C       BE A RECORD IN THE SUB-DIRECTORY THE ADDRESS OF THIS RECORD     MAIN
C       IS GIVEN IN LINK                                                MAIN
C                                                                       MAIN
   32 IF ( LINK .EQ. 0 ) GO TO 36                                       MAIN
      IQ = LINK                                                         MAIN
      READ (10'IQ) DEVADR,LIST,SUBLNK                                   MAIN
      DO 34 I2 = 1,5                                                    MAIN
      IF (U .EQ. DEVADR(I2)) GO TO 37                                   MAIN
   34 CONTINUE                                                          MAIN
      LINK = SUBLNK                                                     MAIN
      GO TO 32                                                          MAIN
C                                                                       MAIN
C        IF THE DEVICE BEING USED IS NOT IN THE USER'S LIST OF DEVICES, MAIN
C       WRITE DIAGNOSTIC MESSAGE AND TERMINATE JOB                      MAIN
```

```
 36 CALL POLOUT (MESS7,480,U,R)
    WRITE (OTDV,56) U, MESS7
    GO TO 114

       IS THE CODE CORRECT ?

 37 IF (NEWCDE .EQ. CODENM) GO TO 38

       UPDATE STATUS CHECK FILE AND ALLOW A SECOND CHANCE
      TO USE THE SYSTEM

 26 ISW = 1
    CALL CHEKUP (ISW)
    IF (TIMES .LT. 2) GO TO 4
    CALL POLOUT (MESS3,480,U,R)
    WRITE (OTDV,56) U, MESS3
    GO TO 114

       UPDATE STATUS CHECK FILE

 38 ISW = 2
    CALL CHEKUP (ISW)
 40 ITEST = 1

       PRESENT BASE DISPLAY ASKING USER TO MAKE REQUEST

 42 L = 256
    CALL EVAL (MESS6,NUM, BLANK,IDNTNO,FLAG)
 46 IF (L1 .EQ. 0) GO TO 114
    IF (IFRAME .EQ. 2 ) GO TO 64
    DO 62 IT = 1,L1

       MATCH THE LIST OF PROGRAMS REQUESTED AGAINST THE LIST
      IN THE USER'S RECORD

    DO 48 JT = 1,10
    IF (PROGNO(IT) .EQ. LIST(JT)) GO TO 60
 48 CONTINUE
    ILL = ILL + 1

       PLACE UNUSABLE PROGRAMS INTO AN ARRAY TO BE DISPLAYED
      AFTER ALL VALID REQUESTS HAVE BEEN SERVICED

    REJCT(ILL) = PROGNO(IT)
    GO TO 62
 60 L11 = L11 + 1
    NWPRGS(L11) = PROGNO(IT)
 62 CONTINUE
    M1 = 1
    M2 = 1
    REPSTK(1,1) = IDH
    REPSTK(2,1) = IDH
    GO TO 68
 64 DO 66 JP = 1,L1
    NWPRGS(JP) = PROGNO(JP)
 66 CONTINUE
    L11 = L1
```

```
C                                                                       MAIN
C        SERVICE REQUEST BY USING THE APPROPRIATE SUBROUTINES AND FILES MAIN
C                                                                       MAIN
   68 DO 96 IZ = 1,L11                                                  MAIN
      PROG = NWPRGS(IZ)                                                 MAIN
C                                                                       MAIN
C        NEW ROUTINES CAN BE ADDED BY EXPANDING THIS GO TO STATEMENT    MAIN
C                                                                       MAIN
      GO TO (70,80,114,86),PROG                                         MAIN
   70 IS = 1                                                            MAIN
      IF (M1 .EQ. 0 ) GO TO 96                                          MAIN
      DO 72 I1 = 1,M1                                                   MAIN
      CALL CHEK (REPSTK(1,I1), IS,X)                                    MAIN
   72 CONTINUE                                                          MAIN
C                                                                       MAIN
C        DISPLAY STATUS ARRAY                                           MAIN
C                                                                       MAIN
      CALL POLOUT (X,480,U,R)                                           MAIN
      WRITE (OTDV,56) U,X                                               MAIN
   74 CALL POLLIN (XX,L,U,R)                                            MAIN
      IF ( U .EQ. NTERM) GO TO 76                                       MAIN
      CALL QUEUE                                                        MAIN
      U = NTERM                                                         MAIN
      GO TO 74                                                          MAIN
   76 DO 78 IN = 9,120                                                  MAIN
   78 X(IN) = BLANK                                                     MAIN
      GO TO 96                                                          MAIN
   80 IF (IFRAME .EQ. 1) GO TO 84                                       MAIN
C                                                                       MAIN
C        IF THE USER IS THE BUREAU THEN THE ACCOUNTS RECEIVABLE OF      MAIN
C      SEVERAL MEMBERS MAY BE UPDATED                                   MAIN
C                                                                       MAIN
      IF (M4 .EQ. 0) GO TO 96                                           MAIN
      DO 82 I2 =1,M4                                                    MAIN
      IJ = REPSTK(2,I2)/10 -100110                                      MAIN
      READ (9'IJ)       IDNO,DEVADR,CODENM,NAME,LIST,UPBEG,UPEND,       MAIN
     1LINK,OPTINF                                                       MAIN
      CALL UPDATE (UPBEG,UPEND, NAME)                                   MAIN
   82 CONTINUE                                                          MAIN
      GO TO 96                                                          MAIN
C                                                                       MAIN
C        PRESENT MEMBER'S ACCOUNTS RECEIVABLE                           MAIN
C                                                                       MAIN
   84 CALL UPDATE (UPBEG,UPEND,NAME)                                    MAIN
      GO TO 96                                                          MAIN
C                                                                       MAIN
C        GENERATE IDENTIFICATION NUMBERS AND CODES                      MAIN
C                                                                       MAIN
   86 IST  = 1                                                          MAIN
      MTEST = 0                                                         MAIN
      DO 88 JA =1,M3                                                    MAIN
      CALL GEN (REPSTK(4,JA),CDEDSP,IST,CHKDGT,MTEST,ISHIFT)            MAIN
   88 CONTINUE                                                          MAIN
      WRITE (OTDV,56) U, CDEDSP                                         MAIN
      CALL POLOUT (CDEDSP,480,U,R)                                      MAIN
   90 CALL POLLIN (MESS4,L,U,R)                                         MAIN
      IF ( U .EQ. NTERM) GO TO 92                                       MAIN
      CALL QUEUE                                                        MAIN
```

```
      U = NTERM                                                          MAIN
      GO TO 90                                                           MAIN
   92 DO 94 IN = 11,120                                                  MAIN
   94 CDEDSP(IN) = BLANK                                                 MAIN
   96 CONTINUE                                                           MAIN
      IF ( ILL .EQ. 0 ) GO TO 106                                        MAIN
C                                                                        MAIN
C        DISPLAY LIST OF REJECTED REQUESTS                               MAIN
C                                                                        MAIN
      DO 104 MN = 1,ILL                                                  MAIN
      MESS5(5) = NUM(REJCT(MN) + 1)                                      MAIN
      WRITE (OTDV,56) U, MESS5                                           MAIN
      CALL POLOUT (MESS5,480,U,R)                                        MAIN
   98 CALL POLLIN (MESS4,L,U,R)                                          MAIN
      IF ( U .EQ. NTERM) GO TO 100                                       MAIN
      CALL QUEUE                                                         MAIN
      U = NTERM                                                          MAIN
      GO TO 98                                                           MAIN
  100 DO 102 JL=11,120                                                   MAIN
      MESS5(JL) = BLANK                                                  MAIN
  102 CONTINUE                                                           MAIN
  104 CONTINUE                                                           MAIN
C                                                                        MAIN
C        ARE THERE INVALID IDENTIFICATION NUMBERS WAITING FOR OUTPUT     MAIN
C                                                                        MAIN
  106 IF ( IQQ. EQ. 0) GO TO 114                                         MAIN
C                                                                        MAIN
C        DISPLAY INVALID IDENTIFICATION NUMBERS                          MAIN
C                                                                        MAIN
      WRITE (OTDV,56) U,MESS9                                            MAIN
      CALL POLOUT (MESS9,480,U,R)                                        MAIN
  108 CALL POLLIN (MESS4,L,U,R)                                          MAIN
      IF ( U .EQ. NTERM) GO TO 110                                       MAIN
      CALL QUEUE                                                         MAIN
      U = NTERM                                                          MAIN
      GO TO 108                                                          MAIN
  110 DO 112 JL =21,100                                                  MAIN
  112 MESS9(JL) = BLANK                                                  MAIN
C                                                                        MAIN
C        RESET COUNTERS                                                  MAIN
C                                                                        MAIN
      ILL = 0                                                            MAIN
      L1 = 0                                                             MAIN
      L11 = 0                                                            MAIN
      L = 49                                                             MAIN
      M1 = 0                                                             MAIN
      M2 = 0                                                             MAIN
      M3 = 0                                                             MAIN
      M4 = 0                                                             MAIN
      IQQ = 0                                                            MAIN
      CALL EVAL (MESS11,NUM,BLANK,IDNTNO,FLAG)                           MAIN
      GO TO 46                                                           MAIN
C                                                                        MAIN
C        TAKE A REQUEST FROM THE WAITING LIST AND THEN PUSH UP           MAIN
C       THE STACK                                                        MAIN
C                                                                        MAIN
  114 IF (NDEX .EQ. 0) GO TO 2                                           MAIN
      WRITE (OTDV,56) U,MESS1                                            MAIN
```

```
      CALL POLOUT (MESS1,480,U,R)                                       MAIN
      IF (NDEX .GT. 1) GO TO 116                                        MAIN
      U = WTSTK(1)                                                      MAIN
      NDEX = NDEX - 1                                                   MAIN
      IWAIT = 1                                                         MAIN
      GO TO 2                                                           MAIN
  116 U = WTSTK(1)                                                      MAIN
      DO 118 J1 = 2,NDEX                                                MAIN
      WTSTK(J1-1) = WTSTK(J1)                                           MAIN
  118 CONTINUE                                                          MAIN
      NDEX = NDEX - 1                                                   MAIN
      GO TO 2                                                           MAIN
   50 FORMAT (20A4)                                                     MAIN
   51 FORMAT (I6,I4,5A4,8I4)                                            MAIN
   52 FORMAT (' 'I6,4X,5(I4,1X),A4,4X,5A4,10(I4,1X)/5I4,10A4/)          MAIN
   53 FORMAT (I6,5I4,6A4/14I4/10A4)                                     MAIN
   55 FORMAT('1',/////,17X,' MESSAGE WRITTEN TO UNITS 201,202',///,     MAIN
     1(' '10X,10A4))                                                    MAIN
   56 FORMAT ('1',/////,17X,'MESSAGE WRITTEN TO UNIT',I4///,            MAIN
     1(' '10X,10A4))                                                    MAIN
   57 FORMAT ('1',/////,17X,'MESSAGE READ FROM UNIT',I4///,             MAIN
     1(' '10X,10A4))                                                    MAIN
  120 CALL EXIT                                                         MAIN
      END                                                               MAIN
```

```
      ..................................................................... EVAL
                                                                            EVAL
      SUBROUTINE EVAL                                                       EVAL
                                                                            EVAL
      PURPOSE                                                               EVAL
                                                                            EVAL
          SUBROUTINE EVAL WRITES OUTPUT MESSAGES TO THE REMOTE              EVAL
       TERMINALS AND EVALUATES THE RESPONSES.                               EVAL
                                                                            EVAL
      USAGE                                                                 EVAL
                                                                            EVAL
          SUBROUTINE EVAL (AMESS, NUM, IBLANK, IDNTNO, FLAG)                EVAL
                                                                            EVAL
      DESCRIPTION OF PARAMETERS                                             EVAL
                                                                            EVAL
          AMESS  - AN ARRAY CONTAINING THE OUTPUT  MESSAGE                  EVAL
                   PASSED FROM THE CALLING  PROGRAM                         EVAL
          NUM    - AN ARRAY OF THE INTEGERS  0  TO 9                        EVAL
          IBLANK - A WORD WHOSE VALUE IS A BLANK                            EVAL
          IDNTNO - AN ARRAY IN WHICH ALL NUMBERS AND CODES READ             EVAL
                   FROM  THE CRTS  IS  TEMPORARILY HELD                     EVAL
          FLAG   - HAS A VALUE OF 1 IF THE SUBROUTINE EXPECTS               EVAL
                   A CODE AS THE  NEXT WORD  TO BE  EVALUATED ,             EVAL
                   0 OTHERWISE                                              EVAL
                                                                            EVAL
      SUBROUTINES CALLED                                                    EVAL
                                                                            EVAL
          POLOUT                                                            EVAL
          POLLIN                                                            EVAL
          STACK                                                             EVAL
          QUEUE                                                             EVAL
                                                                            EVAL
      .....................................................................EVAL
                                                                            EVAL
                                                                            EVAL
      SUBROUTINE EVAL (AMESS,NUM, IBLANK,IDNTNO,FLAG)                       EVAL
       INTEGER AMESS(120),NUM(1),FLAG,REPSTK(6,20),PROGNO(20),WTSTK(10),    EVAL
     1MESS10(120),MESS11(120)                                               EVAL
       INTEGER WORK,MESS(120),IDNTNO(1),AMP/'&'/,U,R,MESS9(120)             EVAL
       INTEGER ALPHA(28)/'A', 'B','C', 'D','E','F','G', 'H','I','J','L',    EVAL
     1'M','N','O','P','Q','R','S', 'T','U','V', 'W','X','Y', 'Z','*',      EVAL
     2'-','K'/                                                              EVAL
       INTEGER BMESS(120)/'YOUR',' COD','E WA','S NO','T RE','CEIV',        EVAL
     1'ED ','---',' PLE','ASE','RE-E','NTER',' IT','   ',106*' '/          EVAL
       LOGICAL * 1 WORD(480), BYTE (4)                                      EVAL
       COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,       EVAL
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10, NTERM, NDEX, WTSTK,MD,MESS11       EVAL
       EQUIVALENCE (WORD(1),MESS(1)), (BYTE(1),WORK)                        EVAL
       K = 0                                                                EVAL
C                                                                           EVAL
C        'I' IS A POINTER WHICH  IS ALWAYS SET TO THE LETTER OR DIGIT       EVAL
C       BEING EXAMINED                                                      EVAL
C                                                                           EVAL
       I = 0                                                                EVAL
       NS = 16                                                              EVAL
C                                                                           EVAL
C          IF INIT = 1  THEN  THE INITIAL  MESSAGE HAS BEEN WRITTEN         EVAL
C       AND THE RESPONSE READ BACK  IN THE CALLING PROGRAM  - HENCE         EVAL
```

```
      PROCEED TO EVALUATE MESSAGE                                       EVAL
                                                                        EVAL
      IF (INIT .EQ. 1) GO TO 3                                          EVAL
      WRITE (6,56) U, AMESS                                             EVAL
      CALL POLOUT (AMESS,L,U,R)                                         EVAL
      DO 26 LL = 1,120                                                  EVAL
      MESS(LL) = IBLANK                                                 EVAL
   26 CONTINUE                                                          EVAL
   23 CALL POLLIN (MESS,L,U,R)                                          EVAL
      WRITE (6,57) U, MESS                                              EVAL
      IF ( U .EQ. NTERM) GO TO 3                                        EVAL
      CALL QUEUE                                                        EVAL
      U = NTERM                                                         EVAL
      GO TO 23                                                          EVAL
                                                                        EVAL
         PLACE CHARACTER READ INTO A WORK  AREA                         EVAL
                                                                        EVAL
    3 WORK =IBLANK                                                      EVAL
                                                                        EVAL
      IF ( MD .EQ. 1 ) GO TO 18                                         EVAL
         IS STACKING REQUIRED ?                                         EVAL
                                                                        EVAL
    1 IF (K .GT. 0 .AND. ITEST .EQ.1) CALL STACK(IDNTNO,NS)             EVAL
    2 I = I + 1                                                         EVAL
      IF (I .GT. 480) GO TO 22                                          EVAL
      BYTE (1) = WORD(I)                                                EVAL
                                                                        EVAL
         SEARCH INPUT MESSAGE FOR NUMBER ENTERED                        EVAL
                                                                        EVAL
      IF (WORK .EQ.IBLANK) GO TO 2                                      EVAL
      DO 5 J = 1,28                                                     EVAL
      IF (WORK .EQ. ALPHA(J)) GO TO 2                                   EVAL
    5 CONTINUE                                                          EVAL
                                                                        EVAL
         EVALUATE NUMBER                                                EVAL
                                                                        EVAL
      K = K + 1                                                         EVAL
      IDNTNO(K) = 0                                                     EVAL
    7 DO 8 IJ = 1,10                                                    EVAL
      IQ = IJ - 1                                                       EVAL
      IF (WORK .EQ. NUM(IJ)) GO TO 10                                   EVAL
    8 CONTINUE                                                          EVAL
   10 IDNTNO(K) = IDNTNO(K) * 10 +IQ                                    EVAL
      I = I + 1                                                         EVAL
      IF (I .GT. 480) GO TO 22                                          EVAL
      BYTE (1) = WORD(I)                                                EVAL
C                                                                       EVAL
C        IF FLAG = 1 THEN THE NEXT WORD IS A CODE, OTHERWISE CONTINUE   EVAL
C      SCANNING  FOR NUMBERS                                            EVAL
C                                                                       EVAL
      IF (WORK .EQ. IBLANK .AND. FLAG .EQ. 1) GO TO 18                  EVAL
      IF (WORK .EQ.IBLANK) GO TO 1                                      EVAL
      DO 12 IZ = 1,28                                                   EVAL
      IF (WORK .EQ. ALPHA(IZ)) GO TO 1                                  EVAL
   12 CONTINUE                                                          EVAL
      GO TO 7                                                           EVAL
C                                                                       EVAL
C        EVALUATE  CODE                                                 EVAL
```

```
   18 I = I + 1                                                                 EVAL
      IF (I .GT. 480) GO TO 21                                                  EVAL
      BYTE(4) = WORD(I)                                                         EVAL
      IF (WORK .EQ. IBLANK ) GO TO 18                                           EVAL
      MA = I + 1                                                                EVAL
      MB = I + 3                                                                EVAL
      DO 20 LA = MA,MB                                                          EVAL
      WORK = WORK * 2**8                                                        EVAL
      BYTE(4) = WORD(LA)                                                        EVAL
   20 CONTINUE                                                                  EVAL
C        RESET FLAG TO 0                                                       EVAL
   14 FLAG = 0                                                                  EVAL
      MD = 0                                                                    EVAL
      IDNTNO(2) = WORK                                                          EVAL
   15 RETURN                                                                    EVAL
C                                                                               EVAL
C        IF I > 480 THE CODE WAS NOT SUPPLIED.  WRITE A MESSAGE ASKING          EVAL
C       FOR A RE-ENTRY OF THE CODE                                              EVAL
C                                                                               EVAL
   21 CALL POLOUT (BMESS, 56,U,R)                                               EVAL
      WRITE (6,56) U, BMESS                                                     EVAL
      DO 28 IM = 1,120                                                          EVAL
   28 MESS(IM) = IBLANK                                                         EVAL
   24 CALL POLLIN ( MESS,L,U,R)                                                 EVAL
      WRITE (6,57) U, MESS                                                      EVAL
      IF ( U .EQ. NTERM ) GO TO 25                                              EVAL
      CALL QUEUE                                                                EVAL
      U = NTERM                                                                 EVAL
      GO TO 24                                                                  EVAL
C                                                                               EVAL
C        RESET COUNTERS                                                         EVAL
C                                                                               EVAL
   25 I = 0                                                                     EVAL
      K = 0                                                                     EVAL
      L1 = 0                                                                    EVAL
      M1 = 0                                                                    EVAL
      M2 = 0                                                                    EVAL
      M3 = 0                                                                    EVAL
      M4 = 0                                                                    EVAL
      MD = 1                                                                    EVAL
      GO TO 3                                                                   EVAL
   22 RETURN                                                                    EVAL
   56 FORMAT ('1',/////,17X,'MESSAGE WRITTEN TO UNIT',I4///,                    EVAL
     1(' '10X,10A4))                                                            EVAL
   57 FORMAT ('1',/////,17X,'MESSAGE READ FROM UNIT',I4///,                     EVAL
     1(' '10X,10A4))                                                            EVAL
      END                                                                       EVAL
```

```
C     ..................................................................  STACK
C                                                                         STACK
C      SUBROUTINE  STACK                                                  STACK
C                                                                         STACK
C      PURPOSE                                                            STACK
C                                                                         STACK
C          SUBROUTINE  STACK PLACES ALL ROUTINES TO BE USED IN AN         STACK
C       ARRAY CALLED  PROGNO , AND ALL REQUESTS TO BE ANSWERED IN         STACK
C      REPSTK                                                             STACK
C                                                                         STACK
C      USAGE                                                              STACK
C                                                                         STACK
C          SUBROUTINE STACK ( IDNTNO, NS)                                 STACK
C                                                                         STACK
C      DESCRIPTION OF PARAMETERS                                          STACK
C                                                                         STACK
C          IDNTNO - AN ARRAY CONTAINING THE NUMBER  TO BE PLACED          STACK
C                   IN ONE OF THE STACKS                                  STACK
C          NS     - A COUNTER USED IN FILLING THE ARRAY OF  INVALID       STACK
C                   IDENTIFICATION NUMBERS                                STACK
C                                                                         STACK
C      SUBROUTINES CALLED                                                 STACK
C                                                                         STACK
C          GEN                                                            STACK
C                                                                         STACK
C      REMARKS                                                            STACK
C                                                                         STACK
C          REPSTK IS A  TWO-DIMENSIONAL  ARRAY                            STACK
C             THE FIRST INDEX  GIVES THE ROUTINE TO BE USED AND           STACK
C             THE SECOND INDEX GIVES THE IDENTIFICATION  NUMBER           STACK
C              OF THE PERTINENT MEMBER                                    STACK
C                                                                         STACK
C     ..................................................................  STACK
C                                                                         STACK
C                                                                         STACK
      SUBROUTINE STACK (IDNTNO,NS)                                        STACK
      INTEGER IDNTNO(1), PROGNO(20), REPSTK(6,20),MESS(120),U,R,CHKDGT,   STACK
     1MESS9(120),MESS10(120),WTSTK(10),MESS11(120)
      COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,      STACK
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10, NTERM, NDEX, WTSTK,MD,MESS11    STACK
C                                                                         STACK
C        IS THE NUMBER THAT OF A ROUTINE OR OF A MEMBER                   STACK
C                                                                         STACK
      IF ( IDNTNO(K) .GE. 100000) GO TO 2                                 STACK
      L1 = L1 + 1                                                         STACK
      KJJ = 0                                                             STACK
C        PLACE IN STACK OF PROGRAM NUMBERS                                STACK
      PROGNO(L1) = IDNTNO(K)                                              STACK
      RETURN                                                              STACK
    2 IPR= PROGNO(L1)                                                     STACK
      KJJ = KJJ + 1                                                       STACK
C                                                                         STACK
C        IF IFRAME =1 THEN THE USER IS A MEMBER AND CAN FORM A            STACK
C       STACK  ONLY  WHEN  ROUTINE 3 (THE REPORT PROGRAM) IS REQUIRED     STACK
C                                                                         STACK
      IF ( IPR .EQ. 4) GO TO 3                                            STACK
C                                                                         STACK
C        IS THE ID.  NO.  VALID ?                                         STACK
```

```
      NCHK = MOD(IDNTNO(K),10)                                          STACK
      IDNT = IDNTNO(K)/10                                               STACK
      MTEST = 1                                                         STACK
      CALL GEN (IDNT,CDEDSP,IST, CHKDGT,MTEST, ISHIFT)                  STACK
      MTEST = 0                                                         STACK
      IF (NCHK .EQ. CHKDGT) GO TO 3                                     STACK
                                                                        STACK
         PLACE INVALID IDENTIFICATION NUMBER  IN AN OUTPUT BUFFER       STACK
                                                                        STACK
      IQQ = 1                                                           STACK
      REWIND 11                                                         STACK
      WRITE (11,50) IPR,KJJ,IDNTNO(K)                                   STACK
      REWIND 11                                                         STACK
      READ (11,51) IPROK,IOUT1,IOUT2                                    STACK
      NS = NS + 5                                                       STACK
      MESS9(NS) = IPROK                                                 STACK
      MESS9(NS+1) = IOUT1                                               STACK
      MESS9(NS+2) = IOUT2                                               STACK
      K = K - 1                                                         STACK
      RETURN                                                            STACK
    3 IF (IPR .NE. 3 .AND. IFRAME .EQ. 1 )RETURN                        STACK
      GO TO (4,5,6,7), IPR                                              STACK
                                                                        STACK
         PLACE IN STACK OF REPORTS TO BE ANSWERED                       STACK
                                                                        STACK
    4 M1 = M1 + 1                                                       STACK
      REPSTK(IPR,M1) = IDNTNO(K)                                        STACK
      RETURN                                                            STACK
    5 M4 = M4 + 1                                                       STACK
      REPSTK(IPR,M4) = IDNTNO(K)                                        STACK
      RETURN                                                            STACK
    6 M2 = M2 + 1                                                       STACK
      REPSTK(IPR,M2) = IDNTNO(K)                                        STACK
      RETURN                                                            STACK
    7 M3 = M3 + 1                                                       STACK
      REPSTK(IPR,M3) = IDNTNO(K)                                        STACK
   50 FORMAT(2I2,I8)                                                    STACK
   51 FORMAT (3A4)                                                      STACK
      RETURN                                                            STACK
      END                                                               STACK
```

```
C     ..................................................................... UPDATE
C                                                                           UPDATE
C      SUBROUTINE UPDATE                                                    UPDATE
C                                                                           UPDATE
C      PURPOSE                                                              UPDATE
C                                                                           UPDATE
C         SUBROUTINE UPDATE RETRIEVES A MEMBER'S ACCOUNTS RECEIVABLE        UPDATE
C        FROM THE ACCOUNTS RECEIVABLE FILE  AND DISPLAYS IT ON HIS CRT      UPDATE
C                                                                           UPDATE
C      USAGE                                                                UPDATE
C                                                                           UPDATE
C     SUBROUTINE UPDATE ( BEGIN, END, NAME)                                 UPDATE
C                                                                           UPDATE
C      DESCRIPTION OF PARAMETERS                                            UPDATE
C                                                                           UPDATE
C          BEGIN - GIVES THE STARTING ADDRESS OF THE MEMBER'S               UPDATE
C                  ACCOUNTS RECEIVABLE IN THE FILE                          UPDATE
C          NAME  - GIVES THE NAME OF THE MEMBER                             UPDATE
C          END   - GIVES THE ENDING ADDRESS OF THE MEMBER'S                 UPDATE
C                                                                           UPDATE
C      SUBROUTINES CALLED                                                   UPDATE
C                                                                           UPDATE
C          POLLIN                                                           UPDATE
C          POLOUT                                                           UPDATE
C          SHCODE                                                           UPDATE
C          QUEUE                                                            UPDATE
C                                                                           UPDATE
C      REMARKS                                                              UPDATE
C                                                                           UPDATE
C          THIS ROUTINE DOES NOT ALLOW UPDATING AS THIS IS THE              UPDATE
C        SUBJECT OF ANOTHER PROJECT.  IT IS INCLUDED ONLY FOR               UPDATE
C        DEMONSTRATION PURPOSES.                                            UPDATE
C                                                                           UPDATE
C     ..................................................................... UPDATE
C                                                                           UPDATE
C                                                                           UPDATE
      SUBROUTINE UPDATE( BEGIN,END,NAME)                                    UPDATE
      INTEGER BEGIN,END,ODV,NAME(1),U,R,WTSTK(10),MESS10(120),              UPDATE
     1PROGNO(20),REPSTK(6,20),MESS(120),MESS9(120),MESS11(120)              UPDATE
      INTEGER BLANK/'  '/                                                   UPDATE
      INTEGER INFO(20),DISPL(120)/'  ','ACCT','S RE','C FO','R',115*' '/UPDATE
      INTEGER CTUE(10)/5*'  ','CONT','INUE','D ','==>','  '/,ENDOF(10)/ UPDATE
     1'END','OF A','CTS','REC','FOR',5*' '/                                 UPDATE
      INTEGER ALFA(32)/'C','#','J','L','M','=','Y','X','Q','I','T','>', UPDATE
     1'U','Z','/','W','-','+','*','A','F','E','O','V','@','%','B','P',  UPDATE
     2'R','H','D','K'/                                                      UPDATE
      COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,    UPDATE
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10, NTERM, NDEX, WTSTK,MD,MESS11   UPDATE
      DEFINE FILE 8(30,80,L,IREC)                                           UPDATE
      ODV=6                                                                 UPDATE
      IREC=BEGIN                                                            UPDATE
      ITESS=BEGIN                                                           UPDATE
C                                                                           UPDATE
CC         PUT NAME  INTO  FIRST  LINE OF DISPLAY                           UPDATE
C                                                                           UPDATE
      DO 1 J = 1,5                                                          UPDATE
      DISPL(J+5) = NAME(J)                                                  UPDATE
    1 CONTINUE                                                              UPDATE
```

```
      ISHIFT = -3                                                       UPDAT
                                                                        UPDAT
          BLANK OUT  NEXT 10 LINES OF DISPLAY                           UPDAT
                                                                        UPDAT
    3 DO 2 I = 11,110                                                   UPDAT
    2 DISPL(I) = BLANK                                                  UPDAT
      IC = 10                                                           UPDAT
    4 IF(ITESS .GT. END) GO TO 8                                        UPDAT
      ITESS=ITESS+1                                                     UPDAT
                                                                        UPDAT
          PLACE  ACCOUNTS RECEIVABLE INTO  OUTPUT BUFFER                UPDAT
                                                                        UPDAT
      READ(8'IREC)INFO                                                  UPDAT
                                                                        UPDAT
         UNSCRAMBLE DATA                                                UPDAT
                                                                        UPDAT
      DO 5 J = 2,6                                                      UPDAT
      INF = INFO(J)                                                     UPDAT
      CALL SHCODE (INF,ALFA,ISHIFT,NEWCDE)                              UPDAT
      INFO(J) = NEWCDE                                                  UPDAT
    5 CONTINUE                                                          UPDAT
      DO 6 J=1,20                                                       UPDAT
      IC=IC+1                                                           UPDAT
      DISPL(IC)=INFO(J)                                                 UPDAT
      IF (IC .EQ. 110 .AND. ITESS .GT. END) GO TO 8                     UPDAT
      IF(IC .EQ. 110) GO TO 7                                           UPDAT
    6 CONTINUE                                                          UPDAT
      GO TO 4                                                           UPDAT
C                                                                       UPDAT
C         FILL IN LAST LINE OF DISPLAY  WITH  'CONTINUED ==>'           UPDAT
C                                                                       UPDAT
    7 DO 9 J1 = 1,10                                                    UPDAT
    9 DISPL(J1+110) = CTUE(J1)                                          UPDAT
      CALL POLOUT (DISPL,480,U,R)                                       UPDAT
      WRITE (ODV,56) U, DISPL                                           UPDAT
   12 CALL POLLIN (DISPL,L,U,R)                                         UPDAT
      WRITE (ODV,57) U, DISPL                                           UPDAT
      IF ( U .EQ. NTERM) GO TO 3                                        UPDAT
      CALL QUEUE                                                        UPDAT
      U = NTERM                                                         UPDAT
      GO TO 12                                                          UPDAT
    8 DO 10 J2 = 1,5                                                    UPDAT
   10 ENDOF(J2+5) = NAME(J2)                                            UPDAT
C                                                                       UPDAT
C         FILL IN LAST LINE OF DISPLAY  WITH  ARRAY ENDOF               UPDAT
C                                                                       UPDAT
      DO 11 J3 = 1,10                                                   UPDAT
   11 DISPL(J3+110) = ENDOF(J3)                                         UPDAT
      CALL POLOUT (DISPL,480,U,R)                                       UPDAT
      WRITE (ODV,56) U, DISPL                                           UPDAT
   13 CALL POLLIN (DISPL,L,U,R)                                         UPDAT
      WRITE (ODV,57) U, DISPL                                           UPDAT
      IF ( U .EQ. NTERM) GO TO 14                                       UPDAT
      CALL QUEUE                                                        UPDAT
      U = NTERM                                                         UPDAT
      GO TO 13                                                          UPDAT
   56 FORMAT ('1',/////,17X,'MESSAGE WRITTEN TO UNIT',I4///,            UPDAT
     1(' '10X,10A4))                                                    UPDATE
```

```
   57 FORMAT ('1',/////,17X,'MESSAGE READ FROM UNIT',I4///,                UPDATE
     1(' '10X,10A4))                                                      UPDATE
   14 RETURN                                                              UPDATE
      END                                                                 UPDATE
```

```
C.....CHEK
C                                                                      CHEK
C     SUBROUTINE  CHEK                                                 CHEK
C                                                                      CHEK
C     PURPOSE                                                          CHEK
C                                                                      CHEK
C        SUBROUTINE CHEK READS A FILE CONTAINING THE NUMBER            CHEK
C       OF TIMES A MEMBER OR A BUREAU OFFICE HAS USED THE              CHEK
C       SYSTEM.  IT PLACES THIS INFORMATION INTO AN OUTPUT BUFFER X,    CHEK
C       UNDER THE HEADINGS 'FAIL' AND 'SUCCESS'.                       CHEK
C                                                                      CHEK
C     USAGE                                                            CHEK
C                                                                      CHEK
C       SUBROUTINE CHEK (IDN,IS,X)                                     CHEK
C                                                                      CHEK
C     DESCRIPTION OF PARAMETERS                                        CHEK
C                                                                      CHEK
C         IDN  - IS THE IDENTIFICATION NUMBER WHICH IS USED            CHEK
C                TO FIND THE RECORD IN THE FILE                        CHEK
C         IS   - A COUNT USED IN FILLING THE OUTPUT BUFFER             CHEK
C         X    - THE OUTPUT BUFFER                                     CHEK
C                                                                      CHEK
C     SUBROUTINES CALLED                                               CHEK
C                                                                      CHEK
C         NONE                                                         CHEK
C                                                                      CHEK
C.....CHEK
C                                                                      CHEK
C                                                                      CHEK
      SUBROUTINE CHEK (IDN,IS,X)                                       CHEK
      INTEGER PROGNO(20), REPSTK(6,20),CHEC(9),X(1),MESS(120),U,R,     CHEK
     1MESS9(120),MESS10(120),WTSTK(10),MESS11(120)                     CHEK
      COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,   CHEK
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10,NTERM,NDEX,WTSTK,MD,MESS11     CHEK
      DEFINE FILE 7 (9,9,U,IR)                                         CHEK
C                                                                      CHEK
C        IF THE USER IS THE BUREAU THEN THE RECORD REQUIRED            CHEK
C       IS THE FIRST IN THE FILE                                       CHEK
C                                                                      CHEK
      IF ( IDN .NE. 1000001) GO TO 3                                   CHEK
      IR = 1                                                           CHEK
      GO TO 4                                                          CHEK
C                                                                      CHEK
C        CALCULATE RECORD NUMBER FROM NEWIDH                           CHEK
C                                                                      CHEK
    3 IR = IDN/10-100110                                               CHEK
    4 READ (7'IR) CHEC                                                 CHEK
C                                                                      CHEK
C        ADVANCE  COUNTER                                              CHEK
C                                                                      CHEK
      IS = IS + 10                                                     CHEK
C        FILL OUTPUT BUFFER                                            CHEK
C                                                                      CHEK
      X(IS) = CHEC(1)                                                  CHEK
      X(IS+1) = CHEC(2)                                                CHEK
      X(IS+3) = CHEC(8)                                                CHEK
      X(IS+6) = CHEC(9)                                                CHEK
      RETURN                                                           CHEK
```

```
EVEL 1, MOD 2          CHEK          DATE = 69043          00/03/20

      END                                                  CHEK
```

```
      ..................................................................  CHEKUP
                                                                          CHEKUP
      SUBROUTINE  CHEKUP                                                  CHEKUP
                                                                          CHEKUP
      PURPOSE                                                             CHEKUP
                                                                          CHEKUP
         SUBROUTINE CHEKUP UPDATES THE FILE CONTAINING THE                CHEKUP
       NUMBER OF SUCCESSFUL AND UNSUCCESSFUL TIMES THE SYSTEM             CHEKUP
        HAS BEEN USED BY EACH MEMBER AND THE BUREAU                       CHEKUP
                                                                          CHEKUP
      USAGE                                                               CHEKUP
                                                                          CHEKUP
        SUBROUTINE CHEKUP (ISW)                                           CHEKUP
                                                                          CHEKUP
      DESCRIPTION OF PARAMETER                                            CHEKUP
                                                                          CHEKUP
               ISW  -  A SWITCH WHICH HAS A VALUE OF 1 IF                 CHEKUP
                       THE USER HAS NOT BEEN GIVEN ACCESS TO              CHEKUP
                       THE SYSTEM AND 2 IF THE USER IS                    CHEKUP
                       IDENTIFIED AS VALID AND CAN USE THE                CHEKUP
                       SYSTEM.                                            CHEKUP
                                                                          CHEKUP
      SUBROUTINES CALLED                                                  CHEKUP
                                                                          CHEKUP
          ATTEND                                                          CHEKUP
                                                                          CHEKUP
      ..................................................................  CHEKUP
                                                                          CHEKUP
      SUBROUTINE CHEKUP (ISW)                                             CHEKUP
      INTEGER CHEC(9),TEMP(4),PROGNO(20),REPSTK(6,20), VAL,MESS(120),     CHEKUP
     1U,R,MESS9(120),WTSTK(10), MESS10(120),MESS11(120)                   CHEKUP
      LOGICAL LGTEMP(4), MASK(4)/Z0F000000,ZF0000,ZF00,ZF/                CHEKUP
      COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,      CHEKUP
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10, NTERM, NDEX, WTSTK,MD,MESS11     CHEKUP
      EQUIVALENCE (TEMP(1),LGTEMP(1))                                     CHEKUP
      DEFINE FILE 7 (9,9,U,IR)                                            CHEKUP
C                                                                         CHEKUP
C        IF IFRAME = 2 THEN THE USER IS THE BUREAU AND THE RECORD         CHEKUP
C       TO BE UPDATED IS THE FIRST IN THE FILE                            CHEKUP
C                                                                         CHEKUP
      IF (IFRAME .NE. 2) GO TO 3                                          CHEKUP
      IR = 1                                                              CHEKUP
      GO TO 10                                                            CHEKUP
C                                                                         CHEKUP
C        CALCULATE RECORD NUMBER IN FILE                                  CHEKUP
C                                                                         CHEKUP
    3 IR = NEWIDH - 100110                                                CHEKUP
   10 READ (7'IR) (CHEC(I),I=1,9)                                         CHEKUP
      IR = IR- 1                                                          CHEKUP
      VAL = 0                                                             CHEKUP
      GO TO (1,6),ISW                                                     CHEKUP
C                                                                         CHEKUP
C        ADD 1 TO THE NUMBER OF UNSUCCESSFUL TIMES                        CHEKUP
C                                                                         CHEKUP
    1 INDX = 8                                                            CHEKUP
C                                                                         CHEKUP
C        CONVERT FROM ZONED TO INTEGER FORMAT                             CHEKUP
C                                                                         CHEKUP
```

```
  2 DO 4 I = 1,4                                                    CHEKUP
  4 TEMP(I) = CHEC(INDX)                                            CHEKUP
    DO 5 I = 1,4                                                    CHEKUP
    LGTEMP(I) = LGTEMP(I) .AND. MASK(I)                             CHEKUP
    IF (I .EQ. 4) GO TO 5                                           CHEKUP
    TEMP(I) = TEMP(I)/2**(8*(4-I))                                  CHEKUP
  5 VAL = VAL * 10 + TEMP(I)                                        CHEKUP
    VAL = VAL + 1                                                   CHEKUP
    IF (VAL .GT. 9999) VAL = 0                                      CHEKUP
    GO TO 7                                                         CHEKUP
                                                                    CHEKUP
      ADD 1 TO THE NUMBER OF SUCCESSFUL TIMES                       CHEKUP
                                                                    CHEKUP
  6 INDX = 9                                                        CHEKUP
    GO TO 2                                                         CHEKUP
  7 IF (INDX .EQ. 9) GO TO 8                                        CHEKUP
    REWIND 11                                                       CHEKUP
    WRITE (11,53) (CHEC(I),I=1,7), VAL,CHEC(9)                      CHEKUP
    NOTICE =MOD(VAL,10)                                             CHEKUP
                                                                    CHEKUP
       IF THE NUMBER OF TIMES THE USER HAS FAILED IN                CHEKUP
      USING THE SYSTEM IS A MULTIPLE OF 10 THEN WRITE THE           CHEKUP
      FOLLOWING MESSAGE TO THE COMPUTER OPERATOR:                   CHEKUP
            * XXXXXXX  HAS FAILED 10 TIMES *                        CHEKUP
      WHERE XXXXXXX IS THE USER'S IDENTIFICATION NUMBER             CHEKUP
                                                                    CHEKUP
    IF (NOTICE .EQ. 0) CALL ATTEND(IDH)                             CHEKUP
    GO TO 9                                                         CHEKUP
  8 REWIND 11                                                       CHEKUP
    WRITE (11,52) (CHEC(I),I=1,8), VAL                              CHEKUP
  9 REWIND 11                                                       CHEKUP
                                                                    CHEKUP
      WRITE UPDATED RECORD INTO FILE                                CHEKUP
                                                                    CHEKUP
    READ (11,54) (CHEC(I),I=1,9)                                    CHEKUP
    WRITE (7'IR) (CHEC(I),I=1,9)                                    CHEKUP
 52 FORMAT (8A4,I4)                                                 CHEKUP
 53 FORMAT (7A4,I4,A4)                                              CHEKUP
 54 FORMAT (9A4)                                                    CHEKUP
    RETURN                                                          CHEKUP
    END                                                             CHEKUP
```

```
C*****************************************************************GEN
C                                                                 GEN
C     SUBROUTINE GEN                                              GEN
C                                                                 GEN
C     PURPOSE                                                     GEN
C                                                                 GEN
C        SUBROUTINE GEN USES A 6 DIGIT NUMBER AS INPUT TO PRODUCE :GEN
C              (1)  A 3 DIGIT CODE                                GEN
C              (2)  A 4 CHARACTER CODE                            GEN
C              (3)  A NEW ID. NO WITH A CHECK DIGIT               GEN
C              (4)  A SCRAMBLED 4 CHARACTER CODE FOR USE IN THE   GEN
C                   DIRECTORY                                     GEN
C                                                                 GEN
C     USAGE                                                       GEN
C                                                                 GEN
C     SUBROUTINE GEN (NUM,CDEDSP,IST,CHKDGT,MTEST,ISHIFT)         GEN
C                                                                 GEN
C     DESCRIPTION OF PARAMETERS                                   GEN
C                                                                 GEN
C        NUM     - THE 6 DIGIT NUMBER USED TO PRODUCE THE CODES   GEN
C        CDEDSP  - AN OUTPUT ARRAY OF THE RESULT                  GEN
C        IST     - A COUNTER FOR INCREMENTING THE ELEMENTS OF THE ARRAY GEN
C        CHKDGT  - A CHECK DIGIT  ATTACHED TO THE  6 DIGIT  ID. NO. GEN
C        MTEST   - A FLAG  INDICATING WHETHER  THE SUBROUTINE IS   GEN
C                  CALLED TO  CHECK THE  VALIDITY OF AN ID. NO OR  GEN
C                  TO PRODUCE A NEW ID. NO                         GEN
C        ISHIFT  - THE VALUE  TO BE USED IN SCRAMBLING  THE CODE   GEN
C                  PRODUCED AT A TERMINAL FOR  COMPARISON IN       GEN
C                  THE DIRECTORY                                   GEN
C                                                                 GEN
C     SUBROUTINES CALLED                                          GEN
C                                                                 GEN
C        SHCODE                                                   GEN
C                                                                 GEN
C                                                                 GEN
C*****************************************************************GEN
C                                                                 GEN
C                                                                 GEN
      SUBROUTINE GEN (NUM,CDEDSP,IST, CHKDGT,MTEST,ISHIFT)        GEN
      INTEGER A,B,C,D,E,F,UNITS,TENS,HUNDS,CODLET,COD1,COD2,G,H,CHKDGT GEN
      INTEGER CDEDSP(1)                                           GEN
      INTEGER ALFA(32)/'C','#','J','L','M','=','Y','X','Q','I','T','>', GEN
     1'U','Z','/','W','-','+','*','A','F','E','O','V','@','%','B','P', GEN
     2'R','H','D','K'/                                            GEN
      LOGICAL * 1 LGALFA(128), CODE(4)                            GEN
      EQUIVALENCE (CODLET,CODE(1)),(LGALFA(1),ALFA(1))            GEN
C                                                                 GEN
C        FIND THE SUM OF  THE  ODD  NUMBERED  DIGITS              GEN
C                                                                 GEN
      A = NUM/ 100000                                             GEN
      B = MOD(NUM,10000)/1000                                     GEN
      C = MOD(NUM,100)/10                                         GEN
      UNITS = A + B + C                                           GEN
      G = UNITS                                                   GEN
      UNITS = MOD(UNITS,10)                                       GEN
C                                                                 GEN
C        FIND THE SUM OF THE  EVEN NUMBERED DIGITS                GEN
C                                                                 GEN
```

```
      D = MOD(NUM,100000)/10000                                          GEN
      E =  MOD (NUM,1000)/100                                            GEN
      F =  MOD ( NUM,10)                                                 GEN
      TENS = D + E + F                                                   GEN
      H = TENS                                                           GEN
                                                                         GEN
         CALCULATE  THE CHECK  DIGIT                                     GEN
                                                                         GEN
      CHKDGT = 3 *H + G                                                  GEN
      CHKDGT = MOD(CHKDGT,10)                                            GEN
      TENS = MOD (TENS,10)                                               GEN
      HUNDS = G + H                                                      GEN
      LAST = HUNDS                                                       GEN
      ISHIFT = MOD(LAST,10) + 1                                          GEN
                                                                         GEN
         IF MTEST = 1 , THE ROUTINE IS TO BE USED  FOR  VERYING AN       GEN
            ID. NO.                                                      GEN
                                                                         GEN
      IF (MTEST .EQ. 1) RETURN                                           GEN
      HUNDS = MOD(HUNDS,10)                                              GEN
      IF (HUNDS .EQ. 0) HUNDS=6                                          GEN
      KL = HUNDS                                                         GEN
                                                                         GEN
         CALCULATE THE 3 DIGIT CODE AND PLACE IN NUMBER                  GEN
                                                                         GEN
      NUMBER = HUNDS*100 + TENS *10 + UNITS                              GEN
                                                                         GEN
         PRODUCE THE REAL CODE  TO BE GIVEN TO THE  MEMBER               GEN
                                                                         GEN
      CODLET = A + E + F                                                 GEN
      CODLET = MOD(CODLET,32)  +3                                        GEN
      INDEX = CODLET*4-3                                                 GEN
      CODE(1) = LGALFA(INDEX)                                            GEN
      COD1 = G + H + KL                                                  GEN
      COD1 = MOD(COD1,32) + 1                                            GEN
      INDEX = COD1*4 - 3                                                 GEN
      CODE(2) = LGALFA(INDEX)                                            GEN
      COD2 = MOD(NUM,100)                                                GEN
      COD2 = MOD(COD2,32) + 1                                            GEN
      INDEX = COD2 * 4 - 3                                               GEN
      CODE(3) = LGALFA(INDEX)                                            GEN
      LAST = MOD(LAST,32) + 1                                            GEN
      INDEX = LAST*4-3                                                   GEN
      CODE(4) = LGALFA(INDEX)                                            GEN
      NUM = NUM * 10 + CHKDGT                                            GEN
C                                                                        GEN
C        CALL ROUTINE TO SCRAMBLE CODE                                   GEN
C                                                                        GEN
      CALL SHCODE (CODLET,ALFA,ISHIFT,NEWCDE)                            GEN
      REWIND 11                                                          GEN
      WRITE (11,50) NUM,CODLET,NEWCDE                                    GEN
      IST = IST + 10                                                     GEN
      REWIND 11                                                          GEN
C                                                                        GEN
C        PLACE RESULT INTO OUTPUT ARRAY                                  GEN
C                                                                        GEN
      READ (11,51) CDEDSP(IST),CDEDSP(IST+1),CDEDSP(IST+4),              GEN
     1CDEDSP(IST+8)                                                      GEN
```

```
  50 FORMAT (I8,2A4)                                              GEN
  51 FORMAT (4A4)                                                 GEN
     RETURN                                                       GEN
     END                                                          GEN
```

$

```
C.....SHCODE
C                                                                SHCODE
C     SUBROUTINE  SHCODE                                         SHCODE
C                                                                SHCODE
C     PURPOSE                                                    SHCODE
C                                                                SHCODE
C        (1)  WHEN CALLED BY SUBROUTINE GEN SCRAMBLES A USER'S CODE SHCODE
C               FOR  PLACING IN THE DIRECTORY AND                SHCODE
C        (2)  WHEN CALLED BY THE MAIN PROGRAM SCRAMBLES THE CODE SHCODE
C               PRODUCED  AT A TERMINAL  FOR COMPARISON WITH ONE  SHCODE
C               IN THE DIRECTORY                                 SHCODE
C                                                                SHCODE
C     USAGE                                                      SHCODE
C                                                                SHCODE
C     SUBROUTINE  SHCODE (CODE, ALFA,ISHIFT,NEWCDE)              SHCODE
C                                                                SHCODE
C     DESCRIPTION OF PARAMETERS                                  SHCODE
C                                                                SHCODE
C        CODE   - THE INPUT PARAMETER FROM THE CALLING  PROGRAM  SHCODE
C        ALFA   - AN  ARRAY OF CHARACTERS USED IN SCRAMBLING     SHCODE
C        ISHIFT - THE CONSTANT BY WHICH  SHIFTING IS TO TAKE PLACE SHCODE
C        NEWCDE - THE NEW CODE                                   SHCODE
C                                                                SHCODE
C     SUBROUTINES CALLED                                         SHCODE
C                                                                SHCODE
C        NONE                                                    SHCODE
C                                                                SHCODE
C                                                                SHCODE
C.....SHCODE
C                                                                SHCODE
C                                                                SHCODE
      SUBROUTINE SHCODE (CODE,ALFA,SHIFT,NEWCDE)                 SHCODE
      INTEGER CODE,SHIFT,ALFA(32),INTCDE(4),BLANK/'  '/          SHCODE
      LOGICAL * 1 LGCDE(4),FLWD(16)                              SHCODE
      EQUIVALENCE (ICODE,LGCDE(1)), (FLWD(1),INTCDE(1))          SHCODE
      ICODE = CODE                                               SHCODE
C                                                                SHCODE
C        PLACE  BLANKS  INTO WORK AREA                           SHCODE
C                                                                SHCODE
      DO 2 I = 1,4                                               SHCODE
      INTCDE(I) = BLANK                                          SHCODE
    2 CONTINUE                                                   SHCODE
C                                                                SHCODE
C        PLACE  CODE INTO NEW AREA - 1 CHARACTER PER FULLWORD    SHCODE
C                                                                SHCODE
      DO 3 I = 1,4                                               SHCODE
      J = 4 * I - 3                                              SHCODE
      FLWD(J) = LGCDE(I)                                         SHCODE
    3 CONTINUE                                                   SHCODE
C                                                                SHCODE
C        PERFORM  SCRAMBLING                                     SHCODE
C                                                                SHCODE
      DO 12 J = 1,4                                              SHCODE
      DO 4 I = 1,32                                              SHCODE
      INDX = I                                                   SHCODE
      IF ( INTCDE(J) .EQ. ALFA(I)) GO TO 8                       SHCODE
    4 CONTINUE                                                   SHCODE
      GO TO 12                                                   SHCODE
```

```
  8 NINDX = INDX + SHIFT                                                SHCODE
    IF ( NINDX .GT. 32) NINDX = NINDX - 32                              SHCODE
    IF (NINDX .LT. 1 ) NINDX = NINDX + 32                               SHCODE
                                                                        SHCODE
       READY  SCRAMBLED CODE FOR  OUTPUT AREA                           SHCODE
                                                                        SHCODE
    INTCDE(J) = ALFA(NINDX)                                             SHCODE
 12 CONTINUE                                                            SHCODE
    DO 16 M = 1, 4                                                      SHCODE
    J = 4 * M - 3                                                       SHCODE
    LGCDE(M) = FLWD(J)                                                  SHCODE
 16 CONTINUE                                                            SHCODE
                                                                        SHCODE
       PLACE SCRAMBLED  CODE INTO OUTPUT AREA                           SHCODE
                                                                        SHCODE
    NEWCDE= ICODE                                                       SHCODE
    RETURN                                                              SHCODE
    END                                                                 SHCODE
```

```
C     ..................................................................  QUEUE
C                                                                         QUEUE
C     SUBROUTINE  QUEUE                                                   QUEUE
C                                                                         QUEUE
C     PURPOSE                                                             QUEUE
C                                                                         QUEUE
C        SUBROUTINE QUEUE PLACES REQUESTS FOR THE SYSTEM INTO A           QUEUE
C     WAITING LIST, WTSTK.  A MESSAGE IS SENT TO THE TERMINAL INFORMING   QUEUE
C     THE USER OF HIS POSITION IN THE STACK                               QUEUE
C                                                                         QUEUE
C     USAGE                                                               QUEUE
C                                                                         QUEUE
C     SUBROUTINE QUEUE                                                    QUEUE
C                                                                         QUEUE
C     SUBROUTINES CALLED                                                  QUEUE
C                                                                         QUEUE
C        NONE                                                             QUEUE
C                                                                         QUEUE
C                                                                         QUEUE
C     ..................................................................  QUEUE
C                                                                         QUEUE
C                                                                         QUEUE
      SUBROUTINE QUEUE                                                    QUEUE
      INTEGER PROGNO(20), REPSTK(6,20),MESS(120),MESS9(120),MESS10(120),QUEUE
     1WTSTK(10),U,R,MESS11(120)                                           QUEUE
      COMMON  PROGNO,REPSTK,ITEST,IFRAME,IDH,K,L1,M1,M2,M3,M4,L,U,R,      QUEUE
     1MESS,INIT,NEWIDH,IQQ,MESS9,MESS10, NTERM, NDEX, WTSTK,MD,MESS11     QUEUE
C                                                                         QUEUE
C        IS THE WAITING LIST EMPTY ?                                      QUEUE
C                                                                         QUEUE
      IF (NDEX .EQ. 0) GO TO 4                                            QUEUE
C                                                                         QUEUE
C        IS THE USER ALREADY IN THE LIST                                  QUEUE
C                                                                         QUEUE
      DO 3 J = 1,NDEX                                                     QUEUE
      JJ = J                                                              QUEUE
      IF ( U .EQ. WTSTK(J)) GO TO 7                                       QUEUE
    3 CONTINUE                                                            QUEUE
C                                                                         QUEUE
C        PLACE THE USER IN THE WAITING LIST                               QUEUE
C                                                                         QUEUE
    4 NDEX = NDEX + 1                                                     QUEUE
      WTSTK(NDEX) = U                                                     QUEUE
      JNDEX = NDEX                                                        QUEUE
      GO TO 8                                                             QUEUE
    7 JNDEX = JJ                                                          QUEUE
C                                                                         QUEUE
C        WRITE A MESSAGE GIVING POSITION IN THE LIST                      QUEUE
C                                                                         QUEUE
    8 REWIND 11                                                           QUEUE
      WRITE (11,50) JNDEX                                                 QUEUE
      REWIND 11                                                           QUEUE
      READ (11,51) MESS10(26)                                             QUEUE
      CALL POLOUT (MESS10,104,U,R)                                        QUEUE
      WRITE (6,52) U, MESS10                                              QUEUE
   50 FORMAT (I4)                                                         QUEUE
   51 FORMAT (A4)                                                         QUEUE
   52 FORMAT ('1',/////,17X,'MESSAGE WRITTEN TO UNIT',I4///,              QUEUE
```

```
      1(' '10X,10A4))                                          QUEUE
       RETURN                                                  QUEUE
       END                                                     QUEUE
```

APPENDIX C

MESSAGE WRITTEN TO UNITS 201,202

* PLEASE TYPE IN YOUR IDENTIFICATION
NUMBER AND YOUR CODE *

EXHIBIT I

MESSAGE READ FROM UNIT 201

1000012 LJC#

EXHIBIT 2

MESSAGE WRITTEN TO UNIT 201

* IDENTIFICATION NUMBER IS INCORRECT
WAIT FOR INITIAL DISPLAY AND RESTART *

EXHIBIT 3

MESSAGE WRITTEN TO UNIT 201

* PLEASE TYPE IN YOUR IDENTIFICATION
NUMBER AND YOUR CODE *

EXHIBIT 4

MESSAGE READ FROM UNIT 201

1000001

EXHIBIT 5

MESSAGE WRITTEN TO UNIT 201

YOUR CODE WAS NOT RECEIVED --- PLEASE
RE-ENTER IT

EXHIBIT 6

MESSAGE READ FROM UNIT 201

LJCC

EXHIBIT 7

MESSAGE WRITTEN TO UNIT 201

* THE CODE IS INCORRECT --- PLEASE MAKE
SURE YOU HAVE THE CORRECT CODE AND
ENTER IT AGAIN *

EXHIBIT 8

MESSAGE READ FROM UNIT 201

LJCU

EXHIBIT 9

MESSAGE WRITTEN TO UNIT 201

* YOUR CODE IS AGAIN INCORRECT -- SORRY,
THE REQUEST CANNOT BE ACCEPTED DUE TO
THIS SECURITY VIOLATION *

EXHIBIT 10

MESSAGE WRITTEN TO UNIT 201

* PLEASE TYPE IN YOUR IDENTIFICATION NUMBER AND YOUR CODE *

EXHIBIT 11

MESSAGE READ FROM UNIT 202

1000001 LJC#

EXHIBIT 12

MESSAGE WRITTEN TO UNIT 202

* SORRY, THIS DEVICE IS NOT RECOGNIZED
AS ONE AVAILABLE TO YOU *

EXHIBIT 13

MESSAGE WRITTEN TO UNIT 202

* PLEASE TYPE IN YOUR IDENTIFICATION
NUMBER AND YOUR CODE *

EXHIBIT 14

MESSAGE READ FROM UNIT 202

1001121 YTU=

EXHIBIT 15

MESSAGE WRITTEN TO UNIT 202

1 STATUS CHECK 2 UPDATE ACCTS REC
3 CREDIT REPORT

MAKE REQUEST

EXHIBIT 16

MESSAGE READ FROM UNIT 202

1 2 4

EXHIBIT 17

MESSAGE WRITTEN TO UNIT 202

	FAIL	SUCCESS
1001121	0	1

EXHIBIT 18

```
MESSAGE WRITTEN TO UNIT 201

 SORRY - THE SYSTEM IS IN USE ---
YOU HAVE NOW BEEN ADDED TO THE
 QUEUE AS NUMBER  :   1
```

EXHIBIT 19

MESSAGE WRITTEN TO UNIT 202

ACCTS REC FOR THE ABC COMPANY

LAMBERTE HARDWARE		
668.44	668.44	
A.G. LETOURNE		
1381.33	1360.59	20.74
MUTT COMPANY		
6135.03	6118.03	17.00
TEREAU RACINE LTD.		
3685.04	1354.20	2299.37

END OF ACTS REC FOR THE ABC COMPANY

EXHIBIT 20

MESSAGE READ FROM UNIT 202

```
 ACCTS REC FOR   THE ABC COMPANY
  LAMBERTE HARDWARE
 668.44     668.44
  A.G. LETOURNE
1381.33    1360.59      20.74
  MUTT COMPANY
6135.03    6118.03      17.00
  TEREAU RACINE LTD.
3685.04    1354.20    2299.37


END OF ACTS REC FOR THE ABC COMPANY
```

EXHIBIT 21

MESSAGE WRITTEN TO UNIT 202

FUNCTION NUMBER 4 IS UNAVAILABLE

EXHIBIT 22

MESSAGE WRITTEN TO UNIT 202

* PLEASE TYPE IN YOUR IDENTIFICATION
NUMBER AND YOUR CODE *

EXHIBIT 23

MESSAGE WRITTEN TO UNIT 201

* PLEASE TYPE IN YOUR IDENTIFICATION
NUMBER AND YOUR CODE *

EXHIBIT 24

MESSAGE READ FROM UNIT 201

1000001 LJC#

EXHIBIT 25

```
      MESSAGE WRITTEN TO UNIT 201

1 STATUS CHECK          2 UPDATE ACCTS REC
3 CREDIT REPORT         4 CODE GENERATOR


MAKE  REQUEST
```

EXHIBIT 26

MESSAGE READ FROM UNIT 201

```
 1 1000001 1001135 1001148 2 1001134 4 1
00181 100182
```

EXHIBIT 27

MESSAGE WRITTEN TO UNIT 201

	FAIL	SUCCESS
1000001	2	1

EXHIBIT 28

MESSAGE WRITTEN TO UNIT 201

ACCTS REC FOR ADLER ASSOC.

WESTOURN			
18197.54	17806.10	37.03	
CUCHRAN DONLUP			
100.66	83.36	13.50	3.80
J.E. ALLEN			
8634.24	8634.24		
EMCOE LTD			
1322.46	841.64	409.55	71.27
MUTT COMPANY			
100.66	83.36	13.55	3.80

CONTINUED ==>

EXHIBIT 29

MESSAGE READ FROM UNIT 201

ACCTS REC FOR ADLER ASSOC.

WESTOURN			
18197.54	17806.10	37.03	
CUCHRAN DONLUP			
100.66	83.36	13.50	3.80
J.E. ALLEN			
8634.24	8634.24		
EMCOE LTD			
1322.46	841.64	409.55	71.27
MUTT COMPANY			
100.66	83.36	13.55	3.80

CONTINUED ==>

EXHIBIT 30

MESSAGE WRITTEN TO UNIT 201

ACCTS REC FOR ADLER ASSOC.
A.G. LETOURNE
1381.00 1360.26 20.74

END OF ACTS REC FOR ADLER ASSOC.

EXHIBIT 31

MESSAGE READ FROM UNIT 201

ACCTS REC FOR ADLER ASSOC.
A.G. LETOURNE
1381.00 1360.26 20.74

END OF ACTS REC FOR ADLER ASSOC.

EXHIBIT 32

MESSAGE WRITTEN TO UNIT 201

ID.NO.	OR CODE	SCR CODE
1001815	=U+>	X/AZ
1001828	Y/*U	I+EW

EXHIBIT 33

MESSAGE WRITTEN TO UNIT 201

THE FOLLOWING IDENTIFICATION NUMBERS
ARE INVALID :
1 2 1001135 1 3 1001148

REQUEST NEXT SCREEN AND RE-ENTER THE
FUNCTION AND IDENTIFICATION NUMBERS

EXHIBIT 34

MESSAGE WRITTEN TO UNIT 201

RE-ENTER FUNCTION AND IDENTIFICATION
NUMBERS

EXHIBIT 35

MESSAGE READ FROM UNIT 201

1 1001134 1001147

EXHIBIT 36

MESSAGE WRITTEN TO UNIT 201

	FAIL	SUCCESS
1001134	0	0
1001147	0	0

EXHIBIT 37

MESSAGE WRITTEN TO UNIT 201

* PLEASE TYPE IN YOUR IDENTIFICATION
NUMBER AND YOUR CODE *

EXHIBIT 38

GLOSSARY OF TERMS

Access	The process of obtaining data from or placing data in storage.
Access-Time	The time taken to read data from or write data into storage.
Algorithm	A precisely described procedure; a set of steps for arriving at a special goal.
Alphanumeric	A character or set of characters that is either alphabetic, numeric or a combination of both.
Binary Search	A search in which a set of items is divided into two parts; where one part is rejected, and the process is repeated on the accepted part until the item with the desired property is found.
Bit	An abbreviation of 'binary digit'.
Byte	A term used to indicate a measurable portion of consecutive bits. For example, an 8-bit byte.
Central Computer	Computer shared by remotely located input-output terminals.
Central Processing Unit (CPU)	The unit of a computing system that contains the circuits that control and perform the execution of instructions.
Compiler	A program that translates source language statements into machine language.

Console Typewriter	The typewriter through which the CPU and the computer operator communicate.
Current Amount Owing	The amount which a debtor owes but payment for which is not yet overdue.
Cursor	A symbol which denotes the position on the CRT screen which the next character entered will occupy.
Data Adapter Unit	A hardware device which converts the 8-bit code of System/360 to a 7-bit code for the IBM 2848 Control Unit, [11] and conversely, it converts the 7-bit code of the 2848 to an 8-bit code for the System/360.
Data Set Concatenation	A technique whereby several data sets residing on different volumes can be read as if they were a single data set. This technique makes it possible for a program to obtain its input from several different types of devices.
Field	A set of one or more characters which is treated as a whole unit.
File	A collection of related records treated as a unit.
File-Address of a Record	The record number of a record in a file.

Format	The size, number, and significance of fields in a record.
General-purpose Computer	A computer designed to solve a wide variety of problems.
Hard Copy	A printed copy of machine output in a visually readable form. Examples are printed reports and computer listings.
Hardware	The mechanical, magnetic, electrical and electronic devices or components of a computer. Examples are disk drives and registers.
Housekeeping	The administrative or overhead operations or functions which are necessary in order to maintain control of a situation.
Instruction	A set of characters, together with one or more addresses (or no address), that defines an operation and which, as a unit, causes the computer to operate accordingly on the indicated quantities.
Interrupt	A special control signal that diverts the attention of the computer from the program in progress because of a particular event or set of circumstances. Control is transferred to a specific

address which is directly related to the type of interrupt that has occurred.

Job Control Language (JCL) — Statements provided by a program and used by the Operating System to initiate and control the processing of jobs.

Linkage Editor — A service program used to prepare loadable programs from the output of compilers. A loadable program is one which has passed the compilation stage and is to be prepared for execution.

MOD (A, B) — The remainder upon dividing the quantity preceding the comma, by the quantity following it.

Modem — A device that converts data from a form which is compatible with data-processing equipment to a form that is compatible with transmission facilities, and vice-versa.

Operating System — An integrated collection of service routines for supervising the sequencing of programs by a computer; it may perform input-output, accounting, and debugging.

Poll — A request by the CPU to each terminal for a message or for readiness to receive a reply.

Privileged Instruction	The input and output instructions plus the instructions for changing the mode switch and memory bounds register (Page 14).
Program	A set of steps that tells the computer exactly how to handle a complete problem.
Record	A collection of fields related to a common identifier.
Register	A device for the temporary storage of one or more words to facilitate arithmetic, logical or transferral operations.
Secondary(Auxiliary) Storage	Storage that is not an integral part of the computer, but is directly linked to and controlled by it. Examples are magnetic tape, disk and drum.
Software	The internal programs or routines professionally prepared to simplify programming and computer operations.
Start Symbol	A symbol which indicates the beginning of data that is to be transferred from a CRT to the data channel.
System Residence Volume	A volume that houses frequently used programs such as compilers.
Terminal	A subset of the input-output devices such as teletypewriters and CRTs.

Upper-case Letters	Capital letters.
Volume	A standard unit of auxiliary storage, such as a reel of tape or a disc pack.
Volume Switching	A change of control from one volume to another. This may happen when a data set resides on more than one volume.
Volume Table of Contents(VTOC)	A table containing the name, location, organization and other control information for a data set stored on the volume.

BIBLIOGRAPHY

1. Banzhaff III, John F., When Your Computer Needs a Lawyer, CACM, Volume 11, Number 8, August 1968

2. Graham, Robert M., Protection in an Information Processing Utility, A Paper from Massachusetts Institute of Technology

3. Parkhill, Douglas F., The Challenge of the Computer Utility, Addison-Wesley Publishing Co., 1966

4. _____, House Group Alarmed by Data Bank Threat, Computerworld, Volume 2, Number 35, August 28, 1968

5. _____, Improving the Security of Files, Computer-world, Volume 2, Number 37, September 11, 1968

6. _____, Optical Page Reader, Datamation, Volume 14, Number 9, September 1968

7. _____, Privacy Hearing Witnesses Offer Conflicting Views, Datamation, Volume 14, Number 3, March 1968

8. Daley, R. C., and Neumann, P. G., A General-Purpose File System for Secondary Storage, Paper on the MULTICS System presented at the Fall Joint Computer Conference, Las Vegas, Nevada, November 30, 1965

9. IBM System/360 Operating System
   System Programmer's Guide File No. S360-20
   Form C28-6550-2

10. IBM System/360 Operating System
    Supervisor and Data Management Services File No. S360/36
    Form C28-6648-0

11. IBM System/360 Operating System
    Graphic Programming Services for IBM 2260 Display Station
    (Local Attachment) File No. S360-30
    Form C27-6912-3

12. IBM System/360 Principles of Operation
    File No. S360-01
    Form A22-6821-3

13. IBM System/360 Operating System
Job Control Language File No. S360-48
Form C28-6539-4

14. IBM System/360
Fortran IV Language File No. S360-25
Form C28-6515-4

15. Hodson, B. A., and Costin, D., A Generalized Program of Modular Design for Remote On-Line Information Systems, Proceedings of a National Symposium on Modular Programming, Published by Information and Systems Institute Inc., 1968.

16. Costin, D., The Design and Development of a Man-Computer Communication System, University of Manitoba, April 1968.