1971

Generalized subroutine system for file processing in either batch or on-line code

Arthur B. Comstock
Lehigh University

ABSTRACT

GENERALIZED SUBROUTINE SYSTEM FOR FILE PROCESSING IN EITHER BATCH OR ON-LINE MODE

by

Arthur B. Comstock

This paper describes the construction and use of the File Processing System designed and implemented on the IBM System 360 with Full Operating System. The File Processing System is a collection of five subroutines written in the fortran programming language which enables a computer user to create and maintain a data file on either tape or disk storage media and access the file sequentially for tape and either sequentially, index sequentially, or randomly for disk.

The design of the File Processing System is such that it is easily adaptable to either a batch processing environment or an on-line conversational mode. It is also modular in design and routines can easily be added to extend its capabilities or improve upon its already existing features.

The primary objective in the development of the file processing technique was to enable a person with a basic fortran programming course as background, to construct and maintain a data file with a minimum of effort. The subroutines contained in this system utilize the most current computer technology available both in the hardware and software areas, and thus these advanced techniques are available to all users regardless of their prior programming experience.

GENERALIZED SUBROUTINE SYSTEM FOR FILE PROCESSING IN EITHER BATCH OR ON-LINE MODE

BY

ARTHUR "B" COMSTOCK

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Information Science

Lehigh University

1971

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 12, 1971
(date)

Andrew J Kasarda
Professor in Charge

Daniel J. Hilman
Head of the Department

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Professor A. J. Kasarda for his valuable suggestions and helpful criticisms.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This paper describes the construction and use of the File Processing System designed and implemented on the IBM System 360 with Full Operating System. The File Processing System is a collection of five subroutines written in the Fortran programming language which enable a computer user to create and maintain a data file on either tape or disk storage media and access the file sequentially for tape and either sequentially, index sequentially, or randomly for disk.

The design of the File Processing System is such that it is easily adaptable to either a batch processing environment or an on-line conversational mode. It is also modular in design and routines can easily be added to extend its capabilities or improve upon its already existing features.[1]

The primary objective in the development of the file processing technique was to enable a person with a basic Fortran programming course as background, to construct and maintain a data file with a minimum of effort. The subroutines contained in this system utilize the most current computer technology available both in the hardware and software areas, and thus these advanced techniques are available to all users regardless of their prior programming experience.

INTRODUCTION

During the past few years the development of computer technology has seen tremenduous advances both in the hardware and software areas. However, as rapid scientific technological development occurs, these advances become more and more oriented toward the highly skilled personnel in a particular field of endeavor, and such has been the case in the computer field. [2]

Computer Programming, with the advent of operating systems, mass random access devices, multi-programming, on-line terminals, etc., has been no exception, and is currently an area requiring skilled personnel. Thus a gap has developed between the computer needs of individuals who are not skilled programmers, but require, and have a serious need for computer capabilities, and the current hardware and software available.

Recent advances in the computer hardware of time sharing and on-line terminals has placed into the hands of many non-programming personnel the ready access to a powerful computer. However, the software to enable this new group of computer users to take full advantage of the computer capability available to them has been of a limited nature and consists mainly of sophisticated programs that do a specific task.

The purpose of the File Processing System developed in this study is to enable computer users to construct and

maintain a data file using the most recent com uter hard-ware and software technology available without becoming, nor needing to be, a highly skilled programmer. The main File Processing System Subroutine and its four subsidiary subroutines make use of sequential, index sequential, and random processing for either tape or disk, whichever is applicable.

DESIGN SPECIFICATIONS OF THE FILE PROCESSING SYSTEM

GENERAL DISCUSSION

Before proceeding further in the discussion it will be useful to explain a general outline of this presentation. The first section, "Design Specifications of the File Processing System", is a description of terminology utilized through out this discussion and an explanation of the overall file processing package and the applications for which it was developed. Following this section is an in-depth look at the capabilities for processing either sequentially, index sequentially, or randomly generated files and examples of how this may be accomplished using the File Processing System.

The second part, "System Design", is an in-depth look at the subroutines from the programming and systems aspect and is intended to give the necessary background information which is needed to modify or improve these routines, as well as the effort required to place this software package in the user system library.

INITIAL DESIGN

Before attempting to describe further the details of the File Processing System it will be useful to describe and define some of the initial planning required in utilizing the capabilities of the system.

In the initial planning stages terminology is of utmost importance, and therefore, a definition of terms used in

this phase and throughout the discussion will be beneficial.

Logical Record: [3] One or more data fields that represent an organized body of related data, such as, all of the basic accounting information concerning a single sales transaction.

Sequential Processing: [3] The reading or writing of each logical record in the continuous sequence in which they have been recorded on the hardware device (tape or disk).

Index Sequential Processing: [3] Method used to directly retrieve or update particular logical records on a direct access device using an index to automatically locate the record. The index consists of a key, which is a unique combination of characters associated with each logical record, and the address of that record on the direct access device.

Random Processing: [3] The reading or writing of the logical record located at an address specified on the output device (disk). For example, each logical input record has associated with it a unique record number beginning with one and numbered consecutively. Using this record number the logical record referenced is found at this location on the data file.

With these definitions in mind the first step to be performed is the development of the length of the logical data record for the file to be created. This consists of

describing the length of each data field to be maintained, based upon the number of characters in the data field, which when summed in total gives the length of the logical record. For example, a logical record containing a payroll number of four characters, an hourly wage rate of six characters, and a social security number of ten characters would be a total of twenty characters of information and be the length of the logical record.

The second decision to be made is the hardware output device to be used, and will be either tape or disk depending on the computer hardware of the system being utilized as well as the method of accessing the data file which is to be created. The choice of accessing methods can be accomplished in the following fashion for those applications which do not involve a massive amount of data.

If the number of logical records is of a significant nature, such as ten thousand or more records, and in succeeding uses of the file more than ten percent of the existing records will be referenced, the most efficient hardware device would be magnetic tape using the sequential accessing method, although the choice of a sequential disk file would be feasible. [4] The choice between tape and disk in an application such as this may be made depending upon the hardware configuration of the computer system available and the operating procedures of the computer center.

If the data file being created will have less than ten percent of the logical records utilized at each succeeding

reference to the file and each logical record has associated with it a unique data field which can be used as a search key, the hardware device should be disk, utilizing the index sequential processing method. [4]

The random method of addressing as it is utilized in these subroutines is only applicable to the disk output device and should be used under the same circumstances as the index sequential method when no readily adaptable key field exists in the logical record. Substituting for the key field is a data field associated with each logical record containing the physical number of the pertinent record in the file such as the tenth record or the twenty-fifth record.

At the completion of these two steps and having available the length of each logical record, the hardware output device desired (tape or disk), and the accessing method decided upon, the necessary information is available to make use of the File Processing System.

PARAMETER DATA REQUIRED

This part of the discussion relates to the main subroutine which is the communication link between the user of the system and the subr utines which actually do the processing. Each subroutine will be discussed in detail a little later, however, all of the information needed in the main routine is required regardless of hardware output device or access method being utilized.

The first step to be completed in using the File

Processing System is to generate a Fortran program with three statements "CALL FILEHD", "CALL EXIT", and an "END" statement. These are the only statements required to generate the linkage needed to reference the main subroutine (FILEHD) which handles all functions, either generating or maintaining, and the required Fortran statements to initiate and terminate the user program.

EXAMPLE: CALL FILEHD
CALL EXIT
END

The second event which must be performed is the completion of a parameter card to lay the foundation for the structure of the file which is desired to generate or maintain. The completed parameter card will be read into the computer memory immediately upon calling the main subroutine and consequently must be the first card of the data file. Remembering our initial design phase description preceding this section pertaining to the decision as to the logical record length and accessing method (page 6), the instructions for the completion of the parameter card are as follows.

Columns one to six contains the name of one of the four functions which is to be performed (CREATE, DELETE, ADD, or CHANGE) and must be left justified with blank fill. The definition employed in this paper for each of these four functions is as follows:

CREATE: This function is only performed one time at the initiation of the file and establishes an initial data file and generates linkages which will be used thereafter when referencing the file.

DELETE: This function is performed whenever it is required to delete one or more logical records from the previously created data file.

ADD: This function is performed whenever it is required to add one or more logical records to a previously created data file.

CHANGE: This function is performed whenever it is required to change one or more logical records already existing on a previously created data file.

With the desired function name inserted into column one to six, the next data field to be completed is column seven to ten. This is accomplished by filling in either "TAPE" or "DISK" whichever hardware output device or storage media is desired for the output to be generated on, or, in the case where prior runs have been made, the media the data file was originally written on. The next data field, columns eleven to fourteen, is a numeric field (each character must be one of the integer digits zero thru nine) indicating the number of input data cards which will be needed to generate one logical output record on either a tape or disk file. (The length of the logical output record was established in the "Initial Design Section", page 6).

For example, if the logical output record size desired is less than 72 characters, the number in columns eleven to fourteen would be '0001' because any number of characters less than 72 can be put on one input data card and thus, for each data card read in, one output record will be written on the output media. If the established logical record size for the output media is between 73 and 144 characters two input data cards must be generated for each output record written and the parameter card field (columns eleven to fourteen) would contain '0002'. This discussion pertaining to the logical output record size and the number of input data cards required to make up the information contained on the file is of critical importance, and will be referred to throughout the remainder of this discussion.

The next data field required on the parameter card is columns fifteen to eighteen and is a key field used only if the file desired is a disk file (column seven to ten of the parameter is completed with 'DISK') and a '0001' placed in this field indicates that an index sequential access method is going to be used. (The access method was decided upon in the "Initial Design Section", page 6.) Any other combination of characters or blanks indicates a direct access or random address file.

The following is a sample parameter card with the allowable information in each field.

Column 1-6	Column 7-10	Column 11-14	Column 15-18
CREATE	TAPE	0001	DISK
ADD	DISK	0002	0001
DELETE		etc.	
CHANGE			

The final event which is needed to utilize the File Processing System is the generation of input data cards. These are of two general formats.

Type One.

Type one, which is used for tape and index sequential disk files, has a control-key field (this is a data field which is unique to each logical output record) located in columns one to eight, and column one must be the alphabetic character 'A'. The control-key field will be used for matching succeeding input data cards for creating a file containing more than 72 characters of information on one logical output record, and will also be the matching field for maintaining the records on the file. The remainder of the input data card, column nine to seventy-two, may be any combination of user generated characters which will be transcribed to the output file.

Type Two.

Type two of the data card formats, which is used for direct access files, can contain any combination of user generated data from columns one to seventy-two. Columns seventy-five to eighty must be numeric and this represents

the relative address of the record on the output file, thus the exact file location of each record is designated before entry onto the output file. If the logical output record is to contain more than seventy-two characters of data the appropriate number of data cards as designated in columns eleven to fourteen is required with the same record number or relative address in columns seventy-five to eighty.

With the completion of the initial design phase and the filling out of the input parameter card the actual effort required to utilize the File Processing System is complete. As can readily be seen from the above discussion, the amount of control data required has been kept to a bare minimum in an effort to eliminate the programming education needed to use the capability within these routines. The following discussions will provide an in-depth look at each of the four subroutines or functions which can be accomplished, CREATE, ADD, DELETE, or CHANGE for all output storage media (tape or disk) and any access method.

CREATING A SEQUENTIAL TAPE FILE

In creating a sequential tape file the first event to be accomplished is the creation of the Fortran program to be used to initiate the appropriate subroutine. This consists of the three Fortran statements as explained in the previous section.

The next step to be accomplished is the completion of the parameter card which is used upon execution of the call statement in the Fortran program. It must immediately

precede all data cards and be filled out as follows:

Column 1-6	Column 7-10	Column 11-14
CREATE	TAPE	Numeric 0001 to 0005

with the remainder of the card blank. Column 11-14 is the only variable information needed and must be the number of input data cards needed to generate one logical output record.

Following the parameter card must be the user data cards in sequence by the control-key field which is located in column 1-8. If the parameter card generated for this particular file had a '0001' in column 11-14 to indicate the writing of a logical tape record of less than 72 characters, then the control-key in each card must be different. Assuming the parameter card were filled out using a '0001' and two cards were entered with the same control-key the second card would not be used and an error message printed 'DUP REC' and list the control-key. If however, the reverse case occurs and the parameter card is filled out with a '0002' and there is no matching control-key the first card will be used with the remainder of the output record containing fill data.

If the parameter card generated for this particular run contains '0002' in column 11-14 to indicate the writing of a logical tape record of more than 72 characters, then the control-key in the data cards needed to make up one logical tape record must be the same. Error messages are

generated similar to the preceeding case.

The logical output record which is created on the tape will be fixed in length with a maximum of 320 characters or a possible five data cards per output record. If less than five data cards are used the remainder of the record is fill. Thus, the following subroutines which maintain this file have available the tape output logical record size and can readily read this file.

The output of the 'CREATE' run which must be held for succeeding runs is the tape number of the file and the printed output at the completion of the run which gives the total number of logical output records written. This total will be helpful in ensuring the correct file is used in succeeding runs.

CHANGING THE SEQUENTIAL TAPE FILE

Prior to this run a tape must have been generated using these subroutines with the control-key information in columns 1-8 and be in sequence by the control-key field. This subroutine also requires a new tape to be used which at the completion of the run will contain the newly updated file and the old original version which should be kept for a period of time as a back-up file.

The record size and block size for the input tape will be known by the 'CHANGE' subroutine since this tape was previously generated using one of the other subroutines in the package.

The user Fortran program will be the same as described

in the previ us section, and will, as in other subr utines, expect the first card of the input data to be the parameter card which must be filled out as follows:

Column 1-6	Column 7-10	Column 11-14
CHANGE	TAPE	Numeric 0001-0005

with the remainder of the card blank. Column 11-14 again is the only variable information on the parameter card and is used to designate the number of input data cards needed to generate one logical output record on tape. (Refer to the explanation of this field in the previous section.)

Following the parameter card will be the user data cards with the control-key data field in columns 1-8 and in sequence by this field. Columns 9-72 can contain any user data required. If more than one data card is required for each tape record (as indicated in column 11-14 of the parameter card) each input data card for the same output tape record must contain the same control-key in columns 1-8.

As each group of data cards are processed which will make up the new output record the previously generated file is read and the control-key field of each is matched. If a match occurs the new record replaces the old in its entirety. There is no changing a record field by field, but the complete record is changed in each case.

If a control-key on a data card is read which does not match a record already existing on the tape file an error message is generated to the printer 'MISSING REC' with the control-key of the data card and the data card is dropped with processing continuing with the next card.

If more than one input data card is found with the same control-key to be changed on the file and the parameter card has indicated only one data card is needed for each output record, the first card will be used and the remaining one(s) dropped with the message 'DUP REC' and the control-key of the card.

If the parameter card indicates more than one input data card is needed to generate one output tape record the first data card will be used to generate the record with the remainder of the record blank.

At the completion of the 'CHANGE' run a control total will be printed giving the number of output records on the new file. This should be checked against the control totals from the previous cycle which generated the tape for input to this run. In this particular subroutine the previous tape and the new tape should be equal in the number of records written.

ADDING TO THE SEQUENTIAL TAPE FILE

Prior to this run a tape must have been generated using these subroutines with control-key information in columns 1-8 and in sequence by this field. This subroutine also

requires a new tape to be used which at the completion of the run will contain the newly updated file.

The record size and block size for the input tape will be known by the 'ADD' subroutine since this tape was previously generated using one of the package subroutines.

The user Fortran program will be the same as described in the previous section, and will, as in other subroutines, expect the first card of the input data to be the parameter card which must be filled out as follows:

Column 1-6	Column 7-10	Column 11-14
ADD	TAPE	Numeric 0001-0005

with the remainder of the card blank. Column 11-14 again is the only variable information on the parameter card and is used to designate the number of input data cards needed to generate one logical output record on tape. (Refer to the explanation of this field in the previous section).

Following the parameter card will be the user data cards with control-key in columns 1-8 and in sequence by this field. Columns 9-72 can contain any user data required. If more than one data card is required for each tape record (as indicated in column 11-14 of the parameter card) each input data card for the same tape record must contain the same control-key in columns 1-8.

As each data card or group of data cards are read which

will make up a new output record, the input file is read and the new record is placed in sequence on the new output tape file.

If a control-key on a data card is found to match a record already existing on the tape file an error message is generated to the printer 'DUP RECORD' with the control-key of the data card and the data card is dropped and processing continues.

If more than one input data card is found with the same control-key to be added to the output file and the parameter card has indicated only one data card is needed for each output record, the first one will be used and the second dropped with the message 'DUP REC' and the control-key of the card.

If the parameter card indicates more than one input data card is needed to generated one output tape record the data card will be used with the remainder of the output record blank.

At the completion of the 'ADD' run a control total will be printed giving the number of output records on the new file. This should be checked against controls from the previous cycle which generated the tape for input to the 'ADD' run by subtracting the new total from the old, which should equal the number of adds which were applied to the file from the input data cards.

DELETING FROM THE SEQUENTIAL TAPE FILE

Prior to this run a tape must have been generated using these subroutines with control-key information in columns 1-8. The delete subroutine will require a new tape to be used which will contain the newly deleted file.

The user Fortran program will be the same as described in the previous section, and will, as in other subroutines, expect the first card of the input data to be the parameter card which must be filled out as follows:

Column 1-6	Column 7-10
DELETE	TAPE

with the remainder of the card blank.

Following the parameter card will be the user data cards which must contain the control-key of the output tape record to be deleted in columns 1-8 and be in sequence by this field. As each data card is read the tape will be searched sequentially, looking for a match on the control-key field. If a data card does not match the record read from the tape that record is passed on to the new tape. If, however, the control-key on the tape and data card match, that record is not written out to the new tape and the next data card is read in.

If a data card is read in for which no match can be found on the tape the error message 'MISSING REC' and the control-key of the input data card is printed.

At the completion of the delete run the total number of records written to the new tape file is printed out and should be checked by taking the total from the tape into the delete run and subtracting the number of delete cards input.

SEQUENTIAL TAPE PROCESSING SAMPLE PROBLEM

User generated Fortran program for each of the four functions creating, deleting, adding, and changing for the IBM S/360 with full Operating System.

```
//CREATE EXEC FORTGCLG
// FORT.SYSIN DD*                          Job Control Information[5]
      CALL FILEND
      CALL EXIT                            Fortran User Program
      END
/*
//GO.FT05F001 DD*                         Job Control Information[5]
CREATETAPE0001                            Parameter Card

                                          User data cards with control-
                                          key in columns 1-[illegible]
/*
//GO.FT08F001 DD DSN-CREATE.TAP           Job Control
//GO.FT09F001 DD DSN-CREATE.TAP             Information[5]
/*
```

All of the above indicated job control cards are constant for each of the four functions. The only variable information needed for any of the runs is the parameter card which must indicate the particular function desired and column 11-14 which is initially designated for the file and will remain the same for all succeeding runs.

Figure 1 on the following page is an example of a file

INFORMATION

149	E1A	9999996902149	21	FB2 BTL DRAIN SE
1135	E1A	99999932731135		NEW
3	E1A	99999922163		MDCONNECTOR TUBE
3	E1A	99999921043	NEW	DC#35 NEBLZ CHMBI
0	E1A	99999912600		MDBATT CABLE 10F
4	E1A	99999911634		MDRELEIF VLV ASM
9	E1A	99999911079		MDRELEIF FLAP
034	E1A	9999996902034	9	FBANGLE WALL BRK
340	E1A	9999996901340	8	FBUNIVERSAL POST
5	E1A	99999939005	9 NEW	MBPEDESTAL W/WHL
	E1A	9999993609	8	DBNASAL CANL CS5I
5TD2	E1A	9999992232ST DZ	712	RASUCT CATH D
5TD2	E1A	9999992231ST DZ	712	RASUCT CATH D
	E1A	9999993575	1622	DCULTRASONIC NEB
4	E1A	99999920374	111	MDHOSE ATTCH PP
004	E1A	9999996626004		FBNOSEPIECE COMPI
102	E1A	9999996261102		
020	E1A	9999997462020		
002	E1A	9999997036002	H	FBEXTENSION ARM
101	E1A	9999995053101	12	FBBR BAG COND 1L
236	E1A	9999993173236	423	FBMAGILL T CF 36I
32	E1A	9999991595 32IN	15	RABR TUBE 32IN
LRG	E1A	9999991540 LRG	11	RAFACEMASK LARGE
4	E1A	99999923024	482	DDESCPH STETH 24I
2	E1A	99999923012	482	DDESCPH STETH 12I
TD3G	E1A	999999300 STD 3GS2		FBSTD FORTREND 3(
003	E1A	9999991405003	112	FBDENTAL ANAL API
004	E1A	9999991400004	111	FBDENTAL ANAL API
TD3G	E1A	999999100 STD 3GS2		FBSTD FORTREND 3(
404	E1A	9999996900404		
996	E1A	9999996901996		FBUNIV MOBILE CR
5TD2	E1A	99999992265ST DZ		RANSL O2 CATH D.
5TD2	E1A	99999992239ST DZ	712	D.
TD3G	E1A	999999100 STD 3GS2		FBSTD FORTREND 3(

FIGURE 1 FILE GENERATED USING CREATE SUBROUTINE (

which has been created using the above sample problem. It is a listing of the output tape records and not of the input data cards as can readily be seen by the control-key listing being at the right side of the listing.

CREATING AN INDEX SEQUENTIAL DISK FILE

For a user to build and maintain an index sequential disk file using the routines described within this manual, several key criteria must be met. First, and perhaps foremost, is that each output record which is to be generated must have a unique key associated with it and be of seven characters or less in length. This will be the only required piece of data needed for each input detail and will be used for seeking records in succeeding runs.

Second, the size of the output records must be of a fixed length and be a multiple of 64 varying from one to five. If the fixed length of the output record is greater than 64 a Fortran statement must be supplied to the subroutine being used as explained in section two of this discussion under the section "Programmers Guide to the Index Sequential Disk Routine".

Once the decision has been made as to the size of the output record and unique key associated with each detail, a small Fortran program must be completed using the three Fortran statements as explained in the previous section.

The next step to be accomplished is the completion of the parameter card which is read immediately upon execution

of the call statement in the Fortran program and consequently must be the very first data card. This parameter card is filled out as follows:

Column 1-6	Column 7-10	Column 11-14	Column 15-16
CREATE	DISK	0001-0005	0001

with the remainder of the card blank. Column 11-14 is the only variable information in the parameter card and must be the number of input details needed to generate one disk record as explained previously in the discussion.

Following the parameter card must be the user data cards in sequence by the control-key field which is located in column 2-8, and the card code field which is column 1 and must be the alphabetic 'A'. The remaining card columns 9-64 can be any user generated information.

If the parameter card generated for this particular run had a '0001' in column 11-14 to indicate the writing of a logical disk record of less than 64 characters, then the control-key in each card must be different. More than one data card appearing in the data stream with the same control-key will cause the second and following cards to be dropped with an error message printed saying 'DUP REC' and list the control-key of those cards eliminated.

If we alter the case to be the one in which the parameter card is filled out with a '0002' in column 11-14 the program

will expect two cards in a row with the same control-key in column 2-8. The case where only one appears will produce output containing the first card information with the remainder of the output record blank. If more than two cards are input with the same control-key all those after the first two will be dropped with the error message 'DUP REC' and the control-key listed on the printer.

The create subroutine to generate an index sequential file will produce two output disk files, one of which is the user data file, the other being a file of control-keys and addresses. This second file is only used internally and consequently is not important to the user, however, it is of importance to someone desiring to alter this routine and is therefore described more thoroughly in the programmers guide, section two of this manual.

The output of the 'CREATE' run will then be a disk file plus a printed total at the completion of the run showing the number of logical output records on the file.

ADDING TO THE INDEX SEQUENTIAL DISK FILE

Prior to this run a disk file must have been generated using one of the other subroutines in this package with control-key information in columns 1-8.

The user Fortran program will be the same as described previously and will expect the first card of the input data to be the parameter card which must be filled out as follows:

Column 1-6	Column 7-10	Column 11-14	Column 15-16
ADD	DISK	0001-0005	0001

with the remainder of the card blank. Column 11-14 is the only variable information on the parameter card and must be the number of input details needed to generate one output disk record.

Following the parameter card must be the user data cards with the control-key information in column 1-8 and column 1 the alphabetic 'A'. The remaining card columns 9-64 can be any user information.

If the parameter card generated for this particular run had a '0001' in column 11-14 to indicate the writing of a logical disk record of less than 64 characters, then the control-key in each card must be different. More than one data card appearing in the data stream with the same control-key will cause the second and following cards to be dropped with an error message 'DUP RECORD' and list the control-key of those data cards dropped.

If we assume the case to be the one in which the parameter card is filled out with a '0002' the program will expect two cards in a row in the data stream with the same control-key. The case where only one appears will produce an output record containing the first card of information with the remainder of the output record blank. If more than two cards are input with the same control-key, all

those after the first two will be dropped with the error message 'DUP RECORD' and the control-key listed on the printer.

When all validation tests are completed the output record is written at the very end of the user data file, however, the address disk file is updated placing the new control-key in its proper sequence in the file. Therefore, when later doing a seek for this record two reads will be done, one to read the address file and match on control-key, the second to read the disk file at the address given from the first read. Thus, the sequence on the data file is immaterial because the sequence will be preserved by the address file. The output of the 'ADD' run is the updated data file and address file, plus a control total on the printer giving the number of new logical output records created on the disk file.

CHANGING RECORDS ON THE INDEX SEQUENTIAL DISK FILE

Prior to this run a disk file must have been generated using one of the other subroutines in this package with control-key information in columns 1-8.

The user Fortran program will be the same as described previously, and will expect the first card of the input data to be the parameter card which must be filled out as follows:

Column 1-6	Column 7-10	Column 11-14	Column 15-18
CHANGE	DISK	0001-0005	0001

with the remainder of the card blank. Columns 11-14 are the only variable information on the parameter card and must be the number of input details needed to generate one output disk record.

Following the parameter card must be the user data cards with the control-key information in columns 2-9 and column 1 the alphabetic 'A'. The remaining card columns 2-64 can be any user information.

If the parameter card generated for this particular run had a '0001' in column 11-14 to indicate the writing of the logical disk record of less than 64 characters, then the control-key in each card must be different. More than one data card appearing in the data stream with the same control-key will cause the second and following cards to be dropped with the error message 'DUP REC' and list the control-key of those data cards dropped.

If we assume the case to be the one in which the parameter card is filled out with a '0002' the program will expect two cards in a row in the data stream with the same control-key. The case where only one appears will produce an output record containing the first card of information with the remainder of the output record blank. If more than two cards are input with the same control-key, all those after the first two will be dropped with the error message 'DUP REC' and the control-key listed on the printer.

As each data card or group of data cards are read which

will make up a new output record, procedures are performed on the disk file using the control-key from the card. If a match occurs the new record replaces the old in its entirety. There is no changing a record field by field, but the complete record is changed in each case.

If a control-key in a data card is read which does not match a record already existing on the disk file an error message is generated to the printer 'MISSING REC' with the control-key of the data card, and the data card is dropped with the processing continuing with the next card.

If more than one input data card is found with the same control-key to be changed on the file and the parameter card has indicated only one data card is needed for each output record, the first card will be used and the remaining ones dropped with the message 'DUP REC' and the control-key of the cards. If the parameter card indicates more than one input data card is needed to generate one disk output record and on reading the input data only one data card is found containing the same control-key the first data card will be used to generate the record with the remainder of the disk record blank.

The output of the 'CHANGE' run is the updated data file and address file, plus a control total on the printer giving the number of changes made to the disk file.

DELETING FROM THE INDEX SEQUENTIAL DISK FILE

Prior to this run a disk file must have been generated

using one of the other subroutines in this package.

The user Fortran program will be the same as described previously, and will expect the first card of the input data to be the parameter card which must be filled out as follows:

Column 1-6	Column 7-10	Column 11-14	Column 15-18
DELETE	DISK	0001	0001

with the remainder of the card blank. All information on the parameter card is fixed as noted above.

Following the parameter card must be the user data cards with the control-key information in column 2-8 and column 1 the alphabetic 'A'.

The input data cards need not be in sequence and as each detail card is read the control-key is matched against the smaller address file. If a match is found the addresses on the file are altered appropriately and the record dropped from the address file. No match or reading of any type is done on the actual user data file. If a match is not found in the address file the input record is dropped from processing with the error message 'MISSING REC' and the control-key of the input data card.

At the completion of the delete run a total of the number of records deleted from the file is printed out.

CREATING A RELATIVE ADDRESS DISK FILE

To generate a relative address direct access disk file the first event to be completed is the creation of the Fortran program to be used to initiate the appropriate subroutine. This consists of the three Fortran statements as explained previously.

The next step to be accomplished is the completion of the parameter card which is used upon execution of the call statement in the Fortran program. It must immediately precede all data cards and be filled out as follows:

Column 1-6	Column 7-10	Column 11-14
CREATE	DISK	0001-0005

with the remainder of the card blank. Column 11-14 is the only variable information needed and must be the number of input data cards needed to generate one logical output record.

Following the parameter card must be the user data cards with user information in columns 1-72 while column 77-80 must be the relative address of the output disk record. The relative address field must be a numeric field with value 0001-9999 and will become the direct address of the output record for retrieving or matching purposes. For example, if the first user data card has '0005' in the relative address field this will become the fifth record on

the file and can be referenced at any time by requesting the record with '0005' in the key field. The input data cards for the creation of any other subroutine relating to the relative address file need not be in sequence by the relative address field.

If the parameter card had a '0001' in column 11-14 indicating that one input data card is needed to generate one output record on the disk, each data card must contain a different relative address number in column 77-80. If the case occurs where two cards with the relative address are input in the same run the last data card read will be the one which updates the file.

If the parameter card generated for this particular run contains '0002' in column 11-14 to indicate the writing of a logical disk record of more than 72 characters, then the relative address field in the data cards needed to make up the output record must be the same. If the case occurs where only one input card is read with the same relative address then the remainder of the record will be left blank. If the reverse case happens and more than two cards are read with the same relative address the last two cards will be read and first dropped.

The logical output record which is created on the disk will be fixed in length and agree with the 'DLFINE FILE' Fortran statement which is explained in section two of this discussion. This card must be changed for each run or the

default option of 72 characters per output record is had. In either of the above instances however, the succeeding subroutines will have passed to it the va ue of the rec rd size.

The output of the 'CREATE' run which sh uld be kept is the disk file name used for later reference (see disk subroutine job control information) and the printed output at the completion of the run which gives the t tal number of logical records written to the output disk. This t tal will be helpful in maintaining control of the file in succeeding runs.

ADDING OR CHANGING A RELATIVE ADDRESS DISK FILE

Prior to the generation of a run to dd or change the relative disk file, a file must have be n previously generated using one of the other relative address disk file subroutines.

The user Fortran program will be the same as described previously, and will expect the first card of the input data to be the parameter card filled out as follows:

Column 1-6	Column 7-10	Column 11-14
ADD	DISK	0001-0005
CHANGE	DISK	0001-0005

with the remainder of the card blank. Column 11-14 again is the only variable information on the parameter card and is used to designate the number of input data cards needed

to generate one logical output record on tape.

Following the parameter card will be the user data cards in random sequence with a numeric address in columns 77- 0 and any combination of user information in columns 1-7 . If more than one input data card is needed for each output disk record (as indicated in column 11-14 of the parameter card) each input data card for the same output disk record must have the same relative address in columns 77-80.

The input data cards are read until a change occurs in the relative address field at which time a disk record is written and processing continues by reading in the next data card or group of data cards.

If the parameter card had a '0001' in columns 11-14 and more than one data card is read with the same relative address in columns 77-80 the first card will be processed and the succeeding ones dropped with a printer message 'DUP REC' and the relative address number.

If the parameter card has a number greater than '0001' in columns 11-14 and only one data card is read with the same relative address number then the first card will be used with the remainder of the disk output record blank.

Both the add and change subroutine function at the output record level, i.e., a complete record must be changed or added and no field by field manipulation is allowed.

At the completion of the 'ADD' or 'CHANGE' run a control

total will be listed on the printer giving the number of input transactions which were processed against the file.

As in other subroutines using the relative address disk file in a random processing method the total number of records on the file is not calculated within the program. This eliminates the need for processing each record on the file and referencing only those being altered for each run. However, the total number of adds or changes is given at the completion of the run.

DELETING A RELATIVE ADDRESS DISK FILE

Prior to the initiation of a run to delete from the relative address disk file, a file must have been generated using one of the other three relative address disk file routines.

The user Fortran program will be the same as described previously, and will expect the first card of the input data to be the parameter card filled out as follows:

Column 1-6	Column 7-10
DELETE	DISK

with the remainder of the card blank.

Following the parameter card, and in random sequence, will be the user prepared data cards. The only information required on the card is the relative address of the record

to be deleted in columns 77-80 and it must be an all numeric field.

As each data card is read the relative address in column 77-80 is used as an address for a disk seek. When the record is found zero's are moved into the relative address field which will be used in later subroutines as a trigger to denote a record on the file which is no longer needed.

If a relative address is found on an input data card for which no matching disk record is located and an error message is generated giving the relative address key.

At the completion of the delete subroutine the control totals are listed on the printer giving the number of records deleted from the file in this run.

RELATIVE ADDRESS DISK SAMPLE PROGRAM

User generated Fortran program for each of the four functions creating, deleting, adding, and changing for the IBM S/360 with full operating system.

```
//CREATE EXEC FORTGCLG                    Job Control Information[5]
//FORT.SYSIN DD*
      CALL FILEHD
      CALL EXIT                           Fortran Coding
      END
/*
//GO.FT05FT001 DD*                        Job Control Information[5]
CREATEDISK0001                            User Parameter Card

                                          User data cards with re-
                                          lative address in
                                          columns 76-80
/*
//GO.FT07FT001 DD DSN-CREATE.DK,UNIT-SYSDA,SPACE-(TRK,(5))
//DISP-(OLD)
/*
```

All of the job control cards are constant for each of the four functions. The only variable information needed for any of the runs is the parameter card which must indicate the particular function desired and column 1-10 which is initially designed for the file and will remain constant for all succeeding runs.

Figure 2 on the following page is an example of a file which has been created using the above sample problem. It is a listing of the output disk records and in this instance coincides with a listing of the input data cards.

INFORMATION

M12	1400004	E1A	9999991400004	111	FBDE
M63	6902149	E1A	9999996902149	21	FB2
M12	1405003	E1A	9999991405003	112	FBDE
M12	3005TD3G	E1A	999999300 STD 3GS2		FBST
M25	32731135	E1A	9999992731135		NEW
M25	22163	E1A	999999221 63		MDCOI
M25	21043	E1A	99999921043	NEW	DC#3'
M25	12600	E1A	9999991 2600		MDBA
M25	11634	E1A	999999116 34		MDRE
M25	11079	E1A	999999110 79		MDRE
M24	6902034	E1A	9999996902034	9	FBAN
M24	6901340	E1A	9999996901340	B	FBUN
M24	39005	E1A	999999 39005	9 NEW	MBPE
M24	3609	E1A	999999 3609	8	DBNA
M13	23012	E1A	9999992 3012	482	DDES
M13	23024	E1A	9999992 3024	482	DDES
M24	22325TD2	E1A	9999992232ST DZ	712	RASU
M24	22315TD2	E1A	9999992231ST DZ	712	RASU
M23	3575	E1A	999999 3575	1622	DCUL
M21	20374	E1A	99999920374	111	MDHJ
M15	6626004	E1A	9999996626004		FBNU
M15	6261102	E1A	9999996261102		
M14	7462020	E1A	9999997462020		
M14	7036002	E1A	9999997036002	H	FBEX
M14	5053101	E1A	9999995053101	12	FBBR
M14	3173236	E1A	9999993173236	423	FBMA
M14	1540LRG	E1A	9999991540 LRG	11	RAFA
M14	159532	E1A	9999991595 32IN	15	RABR
M14	1540LRG	E1A	9999991540 LRG	11	RAFA
M13	23024	E1A	9999992 3024	482	DDES
M13	23012	E1A	9999992 3012	482	DDES
M12	3005TD3G	E1A	999999300 STD 3GS2		FBST
M12	1405003	E1A	9999991405003	112	FBDE
M12	1400004	E1A	9999991400004	111	FBDE

FIGURE 2 FILE GENERATED USING CREATE S.

PROGRAMMING GUIDE TO THE SISTEM 3 FINAL

GENERAL DISCUSSION

This part of the manual is a user guide to the actual Fortran programming of the subroutines and is only useful to a user with basic Fortran programming knowledge.

The manual is divided into three sections, sequential tape routine, relative address disk subroutine, and the indexed sequential disk subroutine. Each section begins with a general description of the particular routine in point, then a flow chart of the routine and finally the actual Fortran coding.

Throughout the Fortran coding listing you will notice hand-written comments which are primarily for the benefit of a user who needs to alter some of the assumptions made in the program, for example, increase the number of user data cards needed to generate one logical output record above the current value of five.

Also included in the Fortran coding listing are numerous comment cards which should help to trace thru the programs and their routines. Before reading this section you should have a thorough knowledge of section one of the discussion because those sections assume the reader to be familiar with the ideas and methods used.

PROGRAMMERS GUIDE TO THE INDEXED SEQUENTIAL DISK SUBROUTINE

This section of the user guide is to enable a user who

is familiar with Fortran coding to be able to understand and maintain the index sequential routines. Appendix I contains a listing of the actual Fortran coding for each of the subroutines required in the index sequential routine.

The disk routine is initialized by a user Fortran program issuing a 'CALL FILEHD' statement as described in section one of the manual. The 'FILEHD' subroutine reads the first data card of the user deck which is the appropriate parameter card and stores this information to be passed on to later routines. It then tests column 7-10 of the parameter card to determine if the user wants a tape or disk routine.

If the parameter card indicates that the disk routine is needed the 'FILEHD' subroutine calls the 'DISCFL' subroutine which does the actual manipulation of the files and remainder of the processing. The parameter card information is then passed by way of common to the disk subroutines.

The first part of the 'DISCFL' subroutine checks to see if this is to be a relative address file or an index sequential file by checking the parameter card information which came from card columns 15-18. If this is '0001' the decision is an index sequential file and the processing continues in the appropriate section of the subroutine.

The index sequential routine requires two disk storage areas, one of which will be the user data file as in other

routines, and secondly, an area which will contain the control-key of each record and its address in the data file plus the address of the next record in sequence. This address file is currently limited to 1000 records, however, by altering the arrays which are printed out in the Fortran listing by comment cards, this can be readily expanded.

In the data description area at the beginning of the routine are two 'DEFINE FILE' statements which are of key interest. The first is the data file descriptor and must be altered depending upon the file that is needed to be generated. The first integer is the unit number which will be constant thru out all cases. The first number inside the parenthesis indicates the total number of records that this file will contain and is used to check space allocation. The current listing is set up to handle a maximum of 35 records. The second number is the record size and must be a multiple of sixty-four which is indicated in the parameter card by columns 11-14. The third and fourth entries inside the parenthesis are both constant and remain 'I' and 'I9' in each case.

The second 'DEFINE FILE' statement is the descriptor for the address or finder file sometimes referred to as the cylinder index. Again the first integer is the unit number and remains constant. The first number inside the parenthesis is the total number of records which is always two for

this particular file. The second number is the record size which will vary depending upon the total number of records which will be in this file. It is currently set up to be 1600, however, any number exceeding this will require a change to this card. The remaining 'L' and 'I8' are both constant for all cases.

Due to the increased complexity of this routine, the following write-up will be done in the sequence of the Fortran listing in Appendix I with the numbers indicating the actual Fortran statement numbers used in the routines.

STATEMENT NUMBER	EXPLANATION OF STATEMENT
500.	This is a do loop which initialises the array in which the address file will be generated.
34.	The next 'READ' statement reads the first data card and stores the information.
66.	This do loop moves the data from the input area to a save area for the next read.
65.	Generates counters which point to the end of the first data card in storage. If this is a multiple card input run (column 11-14 is greater than '0001') the next record will be tagged on immediately behind the first.
200.	The next four 'IF' statements determine which of the four functions is to be performed from

from columns 1-6 on the parameter card and branches to the appropriate section. If the function does not match one of the four the routine halts.

201. This is the create function. The next card is read and if the control-key matches the previous card read control branches back to 66 and this card is tagged on the end of the first card in storage. If the control-key does not match the

299. previous detail the routine writes out the detail

206. which has been held in memory and initialises the counters for multiple card input back to one. At this point the setting up of the address file begins. 'L' which is the array address is incremented by one. Using this as a subscript the control-key of the record just written to the data file is placed in the array, the address of this record is placed in the array (L) and the address of the next record is put in (M). The routine then checks for end of file and if not branches to 66. If it is end, 'ISW' is set

94. and control branches to complete the record held in memory and then goes to 298.

298. This is only done at the completion of the run and is the write which puts the address file out on disk for succeeding runs.

202. This is where the routine branches if the function is 'DELETE'. The 'IF' statement checks to see if the disk address file has been read into core and if not does so.

220. This reads the first data card into storage and checks to see if the control-key is equal to the previous record for multiple card input. If it is equal control branches to 66 as in other functions.

222. Once a change occurs in the detail control-key the array counters for the address file are initialized.

224. The detail control-key is matched against the first control-key in the array, if a match occurs control goes to 221. If the array is less than the detail control goes to 223.

54. If the detail is greater than the address file
55. it is an error and the appropriate message printed.

223. The array was found less than the detail so the subscript 'L' is set equal to the address which is contained in the address file for the next record. It's current place in the array is saved in 'I', control branches to 224.

221. The array and the detail are equal on control-key so it has a record which is to be deleted. (The record will only be deleted from the address file

and not the actual data file.) The address of the previous record in the array which pointed to the record just read which matched on control-key for deletion must be changed to point to the next record after the one to be deleted. This is done by placing the next address of the deleted record back in the previous record read. The control-key of the deleted record is set to zero, a check is made for end of file and the program branches back to 66.

296. If end of file is found the switch is set the last record which is in core processed and the branch goes to 295.

295. The newly updated address file is written out and control goes to 99.

203. This is where the routine branches if the function is 'ADD'. A check is made to see if the address file has been read into memory and if not this is done.

300. The next detail data card is read and stored in memory. If the control-key of this card is the same as the one previously read it indicates multiple card input and control branches to 66.

392. A break is had on the control-key of input and an input record is ready to be processed. The address file array counters are initialized.

304. The input detail is matched against the address file on the control-key field. If the file is less than the input detail control branches to 301, if greater to 302.

303. If the detail to be added equals one already on the file, the appropriate error message is printed and control passes to 66.

301. The input detail is less than the address file so the address of the next record to be used on the address file is put from the current record into the array counter. The address of the previous record on the address file is saved, and control branches to 304.

302. The detail key is greater than the address file which means this record should be placed just prior to the one read. If the control-key of

312. this record is zero, which means it had been deleted in a previous run this area will be used and control passes to 310. If it does not equal zero each control-key is checked until one is found which equals zero, then to 310.

311. If the array of addresses is exceeded an error message 'ARRAY FULL' is printed. The array must be expanded as well as the counter in 311.

310. Upon finding a zero record the address of the record on the data file is set by its location in

the array (K), the address of the next record is taken from the previous detail read (K), and the address of the previous detail's next record is changed to the address of the new add. The new record is then written out to the data file, the multiple input detail counters reset and control goes to 66.

390. End of file routine control comes back to
391. 391 where the address file is written out.
204. This is where the routine branches for the function 'CHANGE'. A check is made to see if the address file has been read into memory and if not this is done.
400. The next detail card is read and the control-key compared with the previous. If they are equal it branches to 66 for multiple card input.
492. The address file array counters are initialised.
404. If the key of the address file is less than the detail the branch is to 401, if it is equal to 402. If it is greater the appropriate error
403. message is printed and control goes to 66.
401. If the array is less than the detail the array counter is set to the next record on the address file and control goes back to 404.
402. If the array is equal to the detail this is the

record to be changed and the address is taken from the file and the new output record written. The counters are then reset for multiple card input, end of file check and return to 66.

490. End of file routine.

491. The address array is written back out to disk.

PROGRAMMER'S GUIDE TO THE TAPE SUBROUTINES

This part of the user guide is to enable someone who is familiar with Fortran coding to understand the workings of the subroutines involved in the sequential tape file handling package. Appendix II contains the actual Fortran coding for the tape subroutines.

The tape function is initialized by a user Fortran program issuing a 'CALL FILEHD' statement as described in earlier sections of the discussion. The 'FILEHD' subroutine reads the first data card of the user deck which is the appropriate parameter card and stores this information to be passed on to later routines. It then tests column 7-10 of the parameter card to determine if the user wants a tape or disk routine.

If it decides that the tape routine is needed the 'FILEHD' subroutine calls the 'TAPFIL' subroutine which does the actual manipulation of the files and remainder of the processing. The parameter card information is then passed by way of common to the tape subroutine.

The tape subroutine reads the second user prepared data card which should be the first of the data cards assuming the parameter card was the initial card and stores this information in core. (It will be assumed throughout this routine that the user data cards are in sequence by the control-key field in columns 1-8.)

The next data card is read and the control-key is compared to the first which has been saved in core. If they match, and the parameter card key indicates more than one input card needed for each output record (column 11-14 of the parameter card) the second card is placed in storage at the end of the first record and processing continues thru this cycle until a different control-key is found. At this point a check is made of the parameter data to see which of the four functions is to be performed at this time and the branch is made to the appropriate area in the program.

CREATE: Write the output tape record from the data saved in storage. The record is fixed length with the control-key in the first eight positions.

ADD: Read the previously generated tape writing out all records in which the control-key of the tape record is less than the input detail. If a match occurs an error message is generated as explained in section one. As soon as the detail control-key differs from that in the file the new record is added in its proper sequential

location on the file and processing continues.

CHANGE: Read the previously generated tape writing out all records in which the control-key of the tape record is less than the input detail. When a match occurs between the detail and the master the new detail is written out to the tape and the current master dropped. (This is a complete record change not field by field.) If in reading the master file the control-key becomes greater than the detail file without having had a match, the input detail is listed on the printer with the appropriate error message.

When the end of processing is reached in any of the four functions control totals are printed as explained in section one.

PROGRAMMERS GUIDE TO THE RELATIVE DISK ROUTINE

This section of the user guide is to enable someone who is familiar with Fortran coding to be able to understand and maintain the relative address routine. Appendix III contains a listing of the Fortran coding in the subroutines required for the relative disk routine.

The disk routine is initialized by a user Fortran program issuing a 'CALL FILEHD' statement as described in section one of the discussion. The 'FILEHD' routine reads the parameter card and stores this information to be used in later routines. It then tests colum 7-10 for disk

or tape and being disk calls the 'DISCFL' subroutine. The parameter card information is passed thru common for this routine. The 'DISCFL' routine then checks column 15-18 and if it is anything other than '0001' it assumes a relative address file and goes to the appropriate section of the routine.

The input data cards are then read and stored until a break on relative address in column 75-80 is found.

With the group of detail cards read in memory which will generate the output record, the parameter card input is again checked to see which function is being performed by this run and proceeds as follows.

CREATE: Using the relative address number (which is in column 75-80 of the input detail) in the variable address field of the write statement (IO), the record is written out. Processing then returns to the next detail record.

DELETE: The relative address of the input detail is put in the variable address field of the read statement and the record to be deleted brought into memory. The address is now moved to the write statement and blanks are moved into this record area. A blank record is then written on the disk with zeros in the relative address field.

ADD or CHANGE: The relative address of the input detail

is put in the variable address field of the write statement and the new information placed in the disk data file at this location.

At the completion of each run the total number of records processed against the file is printed on the printer.

DISCUSSION AND CRITICISM

The File Processing Subroutine System as described in this paper has the capability to utilize the most recent hardware and software available with a minimum of programming effort. A limiting feature to the subroutines as they are now being used, in the case of the tape routines, is that each record written is 320 characters in length regardless of the input data size. This could be a drawback for its use in applications where 50,000 records or more might be processed.

Although the attempt has been made to minimize programming effort, all current systems have some operating information required to handle the hardware processing such as tape numbers, disk areas available, disk file names, etc., and with each operating system being slightly different it is impossible to write subroutines general enough to work on each individual manufacturers machine. However, utilizing skilled programming initially to handle these operating type problems, these subroutines can be quickly updated for each individual case and consequently, utilized to the extent of their capabilities.

Another limiting feature is the length of the control-key information required for matching existing records in the file with new input data. The current routines handle seven digits in the first seven positions of the data card.

This could easily be extended t more charact rs at the beginning of the data card, however, to make it more general and have the control-key appearing anywhere on the c rd would be a much more complex job.

Without a doubt the most important f all the routines is the index sequential disk package. Operationally it is very efficient, maintaining its addresses or master index in core for its matching procedures. This of course limits the number of records which can be placed on the file and the current number is 1600. Although this can be expanded depending on the size of the computer memory available, each additional record requires twelve memory locations and can soon expand beyond the range f memory available. However, the altering of the system to handle more than 1600 records is a relatively easy task if that is desired.

```
FORTRAN IV G LEVEL  19                MAIN                DATE = 71097        15/33/53        PAGE 0001

0001                      CALL FILEHD
                    C***  THIS IS THE MAIN PROGRAM WHICH MUST BE COMPLETED BY EACH USER.
                    C     THE THREE STATEMENTS ARE REQUIRED BY ALL SUBROUTINES.
0002                      CALL EXIT
0003                      END
```

```
FORTRAN IV G LEVEL  19                FILEHD          DATE = 71097          13/35/14          PAGE 0001

0001                  SUBROUTINE FILEHD
                C***  THIS IS CALLED FROM THE MAIN USER GENERATED FORTRAN PROGRAM.
                C     ITS FUNCTION IS TO READ THE PARAMETER CARD, DECIDE UPON THE
                C     DISK OR TAPE FUNCTION AND PASS THE PARAMETER INFO TO THAT
                C     ROUTINE.
0002                  INTEGER RECSZE,KEY
0003                  REAL*8 TYPE,UNIT,UNT1
0004                  DATA UNT1/'DISK'/
0005                1 READ(5,2) TYPE,UNIT,RECSZE,KEY
0006                2 FORMAT (A6,A4,I4,I4)
0007                  I1=RECSZE*16
0008                  IF (UNIT.EQ.UNT1) GO TO 3
0009                  CALL TAPFIL (I1,TYPE)
0010                  GO TO 100
0011                3 CALL DISCFL (I1,TYPE,KEY)
0012              100 RETURN
0013                  END
```

```
FORTRAN IV G LEVEL  18                  DISCFL              DATE = 70342          15/48/00          PAGE 0001

0001                    SUBROUTINE DISCFL (IA,TYPE5,KEY1)
                  C***  THIS IS THE SUBROUTINE USED FOR THE HANDLING OF DISK FILES
0002                    DIMENSION REC3(16),REC4(16), REC5(16)
                  C     ARRAYS INTO WHICH THE FINDER FILE OR ADDRESS FILE IS READ
                  C     FOR INDEX SEQUENTIAL PROCESSING
0003                    DOUBLE PRECISION C(100)
0004                    DIMENSION M(100),N(100)
0005                    INTEGER IA,KEY1
0006                    REAL*8 TYPE5,TYP,TYP1,TYP2,TYP3,CTLKEY,CTLKY1,CTLKY2
0007                    DATA TYP/'CREATE'/,TYP1/'DELETE'/,TYP2/'ADD'/,TYP3/'CHANGE'/
                  C     DISK FILE PARAMETER CARD
                  C     (ESTIMATE OF TOTAL NO. RECORDS,RECORD SIZE, FORMAT, VARIABLE
                  C      RECORD NO.)
0008                    DEFINE FILE 7(38,72,L,I9)
0009                    DEFINE FILE 8(2,1600,L,I8)
                  C     INITIATE COUNTERS
0010                    I8=1
0011                    L=0
                  C     SWITCH TO DESIGNATE EOF
0012                    ISW=0
0013                    ICT=1
0014                    ICT2=16
0015                    I9=1
0016                    DO 500 I=1,100
0017                    M(I)=0
0018                    C(I)=0
0019                500 N(I)=0
                  C     READS INPUT DATA CARD
0020                    READ(5,34,END=99) CTLKEY,REC3
0021                 34 FORMAT (A8,16A4)
                  C     MOVES CURRENT DETAIL DATA TO STORAGE
0022                 66 K=1
0023                    DO 65 I=ICT,ICT2
0024                    REC4(I)=REC3(K)
0025                 65 K=K+1
0026                    CTLKY1=CTLKEY
0027                    ICT=ICT+16
0028                    ICT2=ICT2+16
                  C     RELATIVE ADDRESS ROUTINES GO TO STATEMENT 14
0029                    IF (TYP5.EQ.1) GO TO 14
0030                200 IF (TYP5.EQ.TYP) GO TO 201
0031                    IF (TYP5.EQ.TYP1) GO TO 202
0032                    IF (TYP5.EQ.TYP2) GO TO 203
0033                    IF (TYP5.EQ.TYP3) GO TO 204
0034                    GO TO 99
                  C     CREATE FUNCTION
0035                201 READ (5,34,END=94) CTLKEY,REC3
                  C     MULTIPLE CARD INPUT
```

```
0036                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
                    C     WRITE DISK FILE
0037                299 WRITE (7'I9,206) REC4,CTLKY1
0038                    ICT=1
0039                    ICT2=16
0040                206 FORMAT (16A4,A8)
                    C     CREATE ADDRESSES FOR FINDER FILE
0041                    L=L+1
                    C     CONTROL-KEY FOR FINDER FILE
0042                    C(L)=CTLKY1
                    C     ADDRESS OF CURRENT RECORD ON DATA FILE
0043                    M(L)=I9-1
                    C     ADDRESS OF NEXT RECORD ON DATA FILE
0044                    N(L)=I9
0045                    IF (ISW.EQ.1) GO TO 298
0046                    GO TO 66
0047                 94 ISW=1
0048                    GO TO 299
                    C     END ROUTINE
0049                298 WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0050                297 FORMAT (100(A8,A4,A4))
0051                    GO TO 99
                    C     DELETE FUNCTION
0052                202 IF (I8.NE.1) GO TO 220
                    C     READ FINDER FILE
0053                    READ (8'I8,297) (C(I),M(I),N(I),I=1,100)
                    C     READS INPUT DATA CARD
0054                220 READ(5,34,END=296) CTLKEY,REC3
0055                    IF (CTLKEY.EQ.CTLKY1) GO TO 66
0056                    ICT=1
0057                    ICT2=16
0058                222 L=1
0059                    I=1
                    C     COMPARE CONTROL-KEY OF FINDER FILE AND INPUT DETL
0060                224 IF (C(L).EQ.CTLKY1) GO TO 221
0061                    IF (C(L).LT.CTLKY1) GO TO 223
0062                 54 WRITE (6,55) CTLKY1
0063                    WRITE (6,55) C(L)
0064                 55 FORMAT (A8,'RECORD MISS')
0065                    IF (ISW.EQ.1) GO TO 295
0066                    GO TO 66
                    C     MOVE ARRAY COUNTER UP TO CHECK NEXT RECORD
0067                223 I=L
0068                    L=N(L)
0069                    GO TO 224
                    C     SET NEXT ADDRESS EQUAL TO CURRENT
                    C     CONTROL-KEY FOR FINDER FILE
0070                221 N(I)=N(L)
```

```
                C       ZERO CONTROL-KEY
0071                    C(L)=0
0072                    IF (ISW.EQ.1) GO TO 295
0073                    GO TO 66
0074              296   ISW=1
0075                    GO TO 222
0076              295   I8=1
0077                    WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0078                    GO TO 99
                C       ADD FUNCTION
0079              203   IF (I8.NE.1) GO TO 300
                C       READ FINDER FILE
0080                    READ (8'I8,297) (C(I),M(I),N(I),I=1,100)
                C       READS INPUT DATA CARD
0081              300   READ (5,34,END=390) CTLKEY,REC3
                C       MULTIPLE CARD INPUT
0082                    IF (CTLKEY.EQ.CTLKY1) GO TO 66
0083                    ICT=1
0084                    ICT2=16
0085              392   L=1
0086                    I=1
                C       COMPARE CONTROL-KEY OF FINDER FILE AND INPUT DETL
0087              304   IF (C(L).LT.CTLKY1) GO TO 301
0088                    IF (C(L).GT.CTLKY1) GO TO 302
0089                    WRITE (6,303) CTLKY1
0090              303   FORMAT (A8,'DUP RECORD')
0091                    IF (ISW.EQ.1) GO TO 391
0092                    GO TO 66
                C       UPDATE ADDRESS FOR FINDER FILE
0093              301   I=L
0094                    L=N(L)
0095                    GO TO 304
                C       NEW RECORD TO BE ADDED TO FILE
0096              302   K=1
                C       IS THIS EMPTY FROM A PREVIOUS DELETE
0097              312   IF (C(K).EQ.0) GO TO 310
0098                    K=K+1
0099                    IF (K.LT.101) GO TO 312
0100                    WRITE (6,311) CTLKY1
0101              311   FORMAT (A8,'ARRAY FULL')
0102                    GO TO 391
                C       CREATE ADDRESSES FOR FINDER FILE
0103              310   M(K)=K
0104                    N(K)=N(I)
0105                    N(I)=K
0106                    C(K)=CTLKY1
0107                    I9=K
                C       WRITE DISK FILE
```

```
0108                  WRITE (7'I9,206) REC4,CTLKY1
0109                  IF (ISW.EQ.1) GO TO 391
0110                  GO TO 66
                    C     END ROUTINE
0111                390 ISW=1
0112                  GO TO 392
0113                391 I8=1
0114                  WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0115                  GO TO 99
                    C     CHANGE FUNCTION
0116                204 IF (I8.NE.1) GO TO 400
0117                  READ (8'I8,297) (C(I),M(I),N(I),I=1,100)
                    C     READS INPUT DATA CARD
0118                400 READ (5,34,END=490) CTLKEY,REC3
                    C     MULTIPLE CARD INPUT
0119                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0120                  ICT=1
0121                  ICT2=16
0122                492 L=1
0123                  I=1
                    C     COMPARE CONTROL-KEY OF FINDER FILE AND INPUT DETL
0124                404 IF (C(L).LT.CTLKY1) GO TO 401
0125                  IF (C(L).EQ.CTLKY1) GO TO 402
0126                  WRITE (6,403) CTLKY1
0127                403 FORMAT (A8,'NO MATCH')
0128                  IF (ISW.EQ.1) GO TO 491
0129                  GO TO 66
                    C     GET NEXT FINDER RECORD
0130                401 I=L
0131                  L=N(L)
0132                  GO TO 404
                    C     USE FINDER FILE ADDRESS AND WRITE NEW RECORD
0133                402 I9=M(L)
0134                  WRITE (7'I9,206) REC4,CTLKY1
0135                  IF (ISW.EQ.1) GO TO 491
0136                  GO TO 66
                    C     END ROUTINE
0137                490 ISW=1
0138                  GO TO 492
0139                491 I8=1
0140                  WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0141                  GO TO 99
                    C     RELATIVE ADDRESS DISK ROUTINE
0142                 14 IF (TYPES.EQ.TYP) GO TO 13
0143                  IF (TYPES.EQ.TYP1) GO TO 15
0144                  IF (TYPES.EQ.TYP2) GO TO 16
0145                  IF (TYPES.EQ.TYP3) GO TO 16
                    C     CREATE FUNCTION
```

```
0146            13 READ (5,34,END=95) REC3,CTLKEY
                C     MULTIPLE CARD INPUT
0147                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0148                  I9=CTLKY1
                C     WRITE DISK FILE
0149                  WRITE (7'I9,42) REC4,CTLKY1
0150            42 FORMAT (16A4,2X,I6)
0151                  ICT=1
0152                  ICT2=16
0153                  GO TO 66
                C     END ROUTINE
0154            95 I9=CTLKY1
0155                  WRITE (7'I9,42) REC4,CTLKY1
0156                  GO TO 99
                C     DELETE FUNCTION
0157            15 READ (5,34,END=96) REC3,CTLKEY
0158                  I9=CTLKY1
                C     READ MASTER FILE
0159                  READ(7'I9,42) REC5,CTLKY2
0160                  I9=CTLKY2
0161                  CTLKY2=0
                C     WRITE DELETED MASTER
0162                  WRITE (7'I9,42) REC5,CTLKY2
0163                  GO TO 66
                C     ADD FUNCTION
                C     CHANGE FUNCTION
0164            16 READ (5,34,END=90) REC3,CTLKEY
                C     MULTIPLE CARD INPUT
0165                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0166                  I9=CTLKY1
                C     WRITE NEW RECORD
0167                  WRITE (7'I9,42) REC4,CTLKY1
0168                  GO TO 66
                C     END ROUTINE
0169            96 I9=CTLKY1
0170                  READ(7'I9,42) REC5,CTLKY2
0171                  I9=CTLKY2
0172                  CTLKY2=0
0173                  WRITE (7'I9,42) REC5,CTLKY2
0174                  GO TO 99
0175            90 I9=CTLKY1
0176                  WRITE (7'I9,42) REC4,CTLKY1
0177                  GO TO 99
0178            99 RETURN
0179                  END
```

```
FORTRAN IV G LEVEL  19                MAIN                DATE = 71097          13/35/14          PAGE 0001

0001                    CALL FILEHD
                C***    THIS IS THE MAIN PROGRAM WHICH MUST BE COMPLETED BY EACH USER.
                C       THE THREE STATEMENTS ARE REQUIRED BY ALL SUBROUTINES.
0002                    CALL EXIT
0003                    END
```

```
0001                  SUBROUTINE FILEHD
                C***  THIS IS CALLED FROM THE MAIN USER GENERATED FORTRAN PROGRAM.
                C     ITS FUNCTION IS TO READ THE PARAMETER CARD, DECIDE UPON THE
                C     DISK OR TAPE FUNCTION AND PASS THE PARAMETER INFO TO THAT
                C     ROUTINE.
0002                  INTEGER RECSZE,KEY
0003                  REAL*8 TYPE,UNIT,UNT1
0004                  DATA UNT1/'DISK'/
0005                1 READ(5,2) TYPE,UNIT,RECSZE,KEY
0006                2 FORMAT (A6,A4,I4,I4)
0007                  I1=RECSZE*16
0008                  IF (UNIT.EQ.UNT1) GO TO 3
0009                  CALL TAPFIL (I1,TYPE)
0010                  GO TO 100
0011                3 CALL DISCFL (I1,TYPE,KEY)
0012              100 RETURN
0013                  END
```

```
FORTRAN IV G LEVEL  19                TAPFIL              DATE = 71097          15/33/53                PAGE 0001

 0001                  SUBROUTINE TAPFIL (L,TYPE1)
                C***   THIS IS THE SUBROUTINE USED FOR HANDLING TAPE FILES
                C      STORAGE AREAS FOR SAVING INPUT DATA CARDS
 0002                  DIMENSION REC(16), REC1(16), REC2(16)
 0003                  INTEGER L
 0004                  REAL*8 TYPE1,TYP,TYP1,TYP2,TYP3,CTLKEY,CTLKY1,CTLKY2
 0005                  DATA TYP/'CREATE'/,TYP1/'DELETE'/,TYP2/'ADD'/,TYP3/'CHANGE'/
 0006                  ICT=1
 0007                  ICT2=16
 0008                  ISTRT=1
 0009                  ISW=0
                C      READS  INPUT DATA CARD
 0010                  READ(5,34,END=99) CTLKEY,REC
 0011              34  FORMAT (A8,16A4)
                C      MOVES CURRENT DETAIL DATA TO STORAGE
 0012              66  K=1
 0013                  DO 65 I=ICT,ICT2
 0014                  REC1(I)=REC(K)
 0015              65  K=K+1
 0016                  CTLKY1=CTLKEY
 0017                  ICT=ICT+16
 0018                  ICT2=ICT2+16
 0019                  IF (TYPE1.EQ.TYP) GO TO 4
 0020                  IF (TYPE1.EQ.TYP1) GO TO 100
 0021                  IF (TYPE1.EQ.TYP2) GO TO 200
 0022                  IF (TYPE1.EQ.TYP3) GO TO 300
                C      CREATE FUNCTION
 0023              4   READ(5,34,END=94) CTLKEY,REC
                C      MULTIPLE CARD INPUT
 0024                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
                C      WRITE NEW TAPE RECORD
 0025                  WRITE (3,43) CTLKY1,REC1
 0026              43  FORMAT (A8,(16A4))
 0027                  ICT=1
 0028                  ICT2=16
 0029                  GO TO 66
                C      END ROUTINE
 0030              94  WRITE (3,43) CTLKY1,REC1
 0031                  GO TO 99
                C      DELETE FUNCTION
 0032             100  READ (5,34,END=199) CTLKEY,REC
 0033                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
 0034                  ICT=1
 0035                  ICT2=ICT2+16
 0036             101  READ (3,43,END=197) REC2,CTLKY2
                C      MATCH DETAIL AND MASTER
 0037             104  IF (CTLKY1.EQ.CTLKY2) GO TO 101
 0038                  IF (CTLKY2.LT.CTLKY1) GO TO 102
```

```
0039                IF (ISW.EQ.1) GO TO 102
0040                GO TO 66
                C   END ROUTINE
0041              105 READ (8,43,END=99)  REC2,CTLKY2
0042              102 WRITE (9,43) REC2,CTLKY2
0043                  IF (ISW.EQ.1) GO TO 105
0044                  GO TO 101
0045              199 ISW=1
0046                  GO TO 104
0047              197 WRITE (6,55) CTLKEY
0048               55 FORMAT (A8,'RECORD MISS')
0049                  GO TO 99
                C   ADD FUNCTION
0050              200 READ (5,34,END=299) CTLKEY,REC
                C   MULTIPLE CARD INPUT
0051                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0052                  ICT=ICT+16
0053                  ICT2=ICT2+16
0054                  IF (ISTRT.EQ.2) GO TO 298
0055                  IF (ISTRT.NE.1) GO TO 205
0056                  ISTRT=0
                C   READ MASTER FILE
0057              201 READ (8,43,END=298) REC2,CTLKY2
                C   MATCH DETL AND MASTER
0058              205 IF (CTLKY2.LT.CTLKY1) GO TO 202
0059                  IF (CTLKY2.EQ.CTLKY1) GO TO 203
0060                  WRITE (9,43) REC1,CTLKY1
0061                  GO TO 204
                C   WRITE NEW TAPE RECORD
0062              202 WRITE (9,43) REC2,CTLKY2
0063                  GO TO 201
                C   DETAIL ALREADY ON MASTER
0064              203 WRITE (6,56) CTLKY1
0065               56 FORMAT (A8,'DUPLICATE')
0066                  IF (ISTRT.EQ.3) GO TO 206
0067                  GO TO 66
                C   END ROUTINE
0068              299 ISTRT=3
0069                  GO TO 205
0070              206 IF (ISTRT.EQ.2) GO TO 99
0071                  WRITE (9,43) REC2,CTLKY2
0072                  READ (8,43,END=99) REC2,CTLKY2
0073                  GO TO 206
0074              204 IF (ISTRT.EQ.3) GO TO 206
0075                  GO TO 66
0076              298 WRITE (9,43) REC1,CTLKY1
0077                  ISTRT=2
0078                  GO TO 66
```

```
                C       CHANGE FUNCTION
0079              300 READ (5,34,END=399) CTLKEY,REC
                C       MULTIPLE CARD INPUT
0080                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0081                  ICT=1
0082                  ICT2=16
0083                  IF (ISTRT.NE.1) GO TO 304
0084                  ISTRT=0
                C       READ MASTER FILE
0085              301 READ (8,43,END=398) REC2,CTLKY2
                C       MATCH DETL AND MASTER
0086              304 IF (CTLKY2.LT.CTLKY1) GO TO 302
0087                  IF (CTLKY2.EQ.CTLKY1) GO TO 303
0088                  WRITE (6,55) CTLKY1
0089                  IF (ISTRT.EQ.3) GO TO 302
0090                  GO TO 66
0091              302 WRITE (9,43) REC2,CTLKY2
0092                  IF (ISTRT.NE.3) GO TO 301
0093                  READ (8,43,END=99) REC2,CTLKY2
0094                  GO TO 302
                C       WRITE NEW TAPE RECORD
0095              303 WRITE (9,43) REC1,CTLKY1
0096                  READ (8,43,END=99) REC2,CTLKY2
0097                  IF (ISTRT.NE.3) GO TO 66
0098                  GO TO 302
                C       END ROUTINE
0099              399 ISTRT=3
0100                  GO TO 304
0101              398 IF (ISTRT.EQ.3) GO TO 99
0102                  WRITE (6,55) CTLKY1
0103                  GO TO 99
0104               99 RETURN
0105                  END
```

```
FORTRAN IV G LEVEL  18                MAIN                DATE = 70342          15/48/00          PAGE 0001

0001                      CALL FILEHD
                    C***  THIS IS THE MAIN PROGRAM WHICH MUST BE COMPLETED BY EACH USER.
                    C     THE THREE STATEMENTS ARE REQUIRED BY ALL SUBROUTINES.
0002                      CALL EXIT
0003                      END
```

```
FORTRAN IV G LEVEL  18                FILEHD          DATE = 70342          15/48/00          PAGE 0001

0001                  SUBROUTINE FILEHD
                C***  THIS IS CALLED FROM THE MAIN USER GENERATED FORTRAN PROGRAM.
                C     ITS FUNCTION IS TO READ THE PARAMETER CARD, DECIDE UPON THE
                C     DISK OR TAPE FUNCTION AND PASS THE PARAMETER INFO TO THAT
                C     ROUTINE.
0002                  INTEGER RECSZE,KEY
0003                  REAL*8 TYPE,UNIT,UNT1
0004                  DATA UNT1/'DISK'/
0005                1 READ(5,2) TYPE,UNIT,RECSZE,KEY
0006                2 FORMAT (A6,A4,I4,I4)
0007                  I1=RECSZE*16
0008                  IF (UNIT.EQ.UNT1) GO TO 3
0009                  CALL TAPEIL (I1,TYPE)
0010                  GO TO 100
0011                3 CALL DISCEL (I1,TYPE,KEY)
0012              100 RETURN
0013                  END
```

```
0001                    SUBROUTINE DISCFL (IA,TYPE5,KEY1)
                C***    THIS IS THE SUBROUTINE USED FOR THE HANDLING OF DISK FILES
0002                    DIMENSION REC3(16),REC4(16), REC5(16)
                C       ARRAYS INTO WHICH THE FINDER FILE OR ADDRESS FILE IS READ
                C       FOR INDEX SEQUENTIAL PROCESSING
0003                    DOUBLE PRECISION C(100)
0004                    DIMENSION M(100),N(100)
0005                    INTEGER IA,KEY1
0006                    REAL*8 TYPE5,TYP,TYP1,TYP2,TYP3,CTLKEY,CTLKY1,CTLKY2
0007                    DATA TYP/'CREATE'/,TYP1/'DELETE'/,TYP2/'ADD'/,TYP3/'CHANGE'/
                C       DISK FILE PARAMETER CARD
                C       (ESTIMATE OF TOTAL NO. RECORDS,RECORD SIZE, FORMAT, VARIABLE
                C        RECORD NO.)
0008                    DEFINE FILE 7(38,72,L,I9)
0009                    DEFINE FILE 8(2,1600,L,I8)
                C       INITIATE COUNTERS
0010                    I8=1
0011                    L=0
                C       SWITCH TO DESIGNATE EOF
0012                    ISW=0
0013                    ICT=1
0014                    ICT2=16
0015                    I9=1
0016                    DO 500 I=1,100
0017                    M(I)=0
0018                    C(I)=0
0019                500 N(I)=0
                C       READS INPUT DATA CARD
0020                    READ(5,34,END=99) CTLKEY,REC3
0021                 34 FORMAT (A8,16A4)
                C       MOVES CURRENT DETAIL DATA TO STORAGE
0022                 66 K=1
0023                    DO 65 I=ICT,ICT2
0024                    REC4(I)=REC3(K)
0025                 65 K=K+1
0026                    CTLKY1=CTLKEY
0027                    ICT=ICT+16
0028                    ICT2=ICT2+16
                C       RELATIVE ADDRESS ROUTINES GO TO STATEMENT 14
0029                    IF (KEY1.NE.1) GO TO 14
0030                200 IF (TYPE5.EQ.TYP) GO TO 201
0031                    IF (TYPE5.EQ.TYP1) GO TO 202
0032                    IF (TYPE5.EQ.TYP2) GO TO 203
0033                    IF (TYPE5.EQ.TYP3) GO TO 204
0034                    GO TO 99
                C       CREATE FUNCTION
0035                201 READ (5,34,END=94) CTLKEY,REC3
                C       MULTIPLE CARD INPUT
```

```
0036                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
                    C     WRITE DISK FILE
0037                299 WRITE (7'I9,206) REC4,CTLKY1
0038                    ICT=1
0039                    ICT2=16
0040                206 FORMAT (16A4,A8)
                    C     CREATE ADDRESSES FOR FINDER FILE
0041                    L=L+1
                    C     CONTROL-KEY FOR FINDER FILE
0042                    C(L)=CTLKY1
                    C     ADDRESS OF CURRENT RECORD ON DATA FILE
0043                    M(L)=I9-1
                    C     ADDRESS OF NEXT RECORD ON DATA FILE
0044                    N(L)=I9
0045                    IF (ISW.EQ.1) GO TO 298
0046                    GO TO 66
0047                 94 ISW=1
0048                    GO TO 299
                    C     END ROUTINE
0049                298 WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0050                297 FORMAT (100(A8,A4,A4))
0051                    GO TO 99
                    C     DELETE FUNCTION
0052                202 IF (I8.NE.1) GO TO 220
                    C     READ FINDER FILE
0053                    READ (8'I8,297) (C(I),M(I),N(I),I=1,100)
                    C     READS INPUT DATA CARD
0054                220 READ(5,34,END=296) CTLKEY,REC3
0055                    IF (CTLKEY.EQ.CTLKY1) GO TO 66
0056                    ICT=1
0057                    ICT2=16
0058                222 L=1
0059                    I=1
                    C     COMPARE CONTROL-KEY OF FINDER FILE AND INPUT DETL
0060                224 IF (C(L).EQ.CTLKY1) GO TO 221
0061                    IF (C(I).LT.CTLKY1) GO TO 223
0062                 54 WRITE (6,55) CTLKY1
0063                    WRITE (6,55) C(I)
0064                 55 FORMAT (A8,'RECORD MISS')
0065                    IF (ISW.EQ.1) GO TO 295
0066                    GO TO 66
                    C     MOVE ARRAY COUNTER UP TO CHECK NEXT RECORD
0067                223 I=I
0068                    L=N(I)
0069                    GO TO 224
                    C     SET NEXT ADDRESS EQUAL TO CURRENT
                    C     CONTROL-KEY FOR FINDER FILE
0070                221 N(I)=N(I)
```

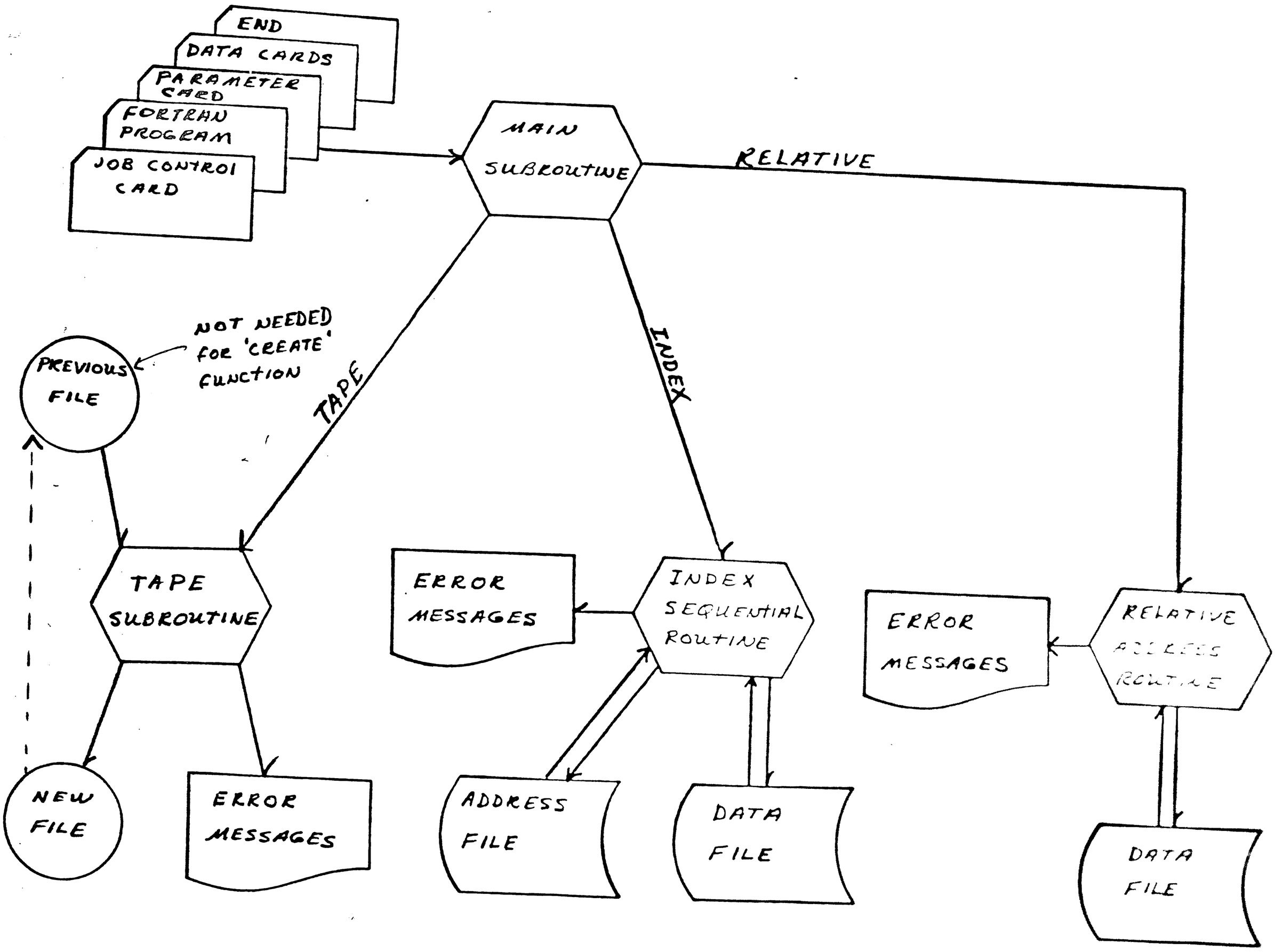
```
                C       ZERO CONTROL-KEY
0071                    C(L)=0
0072                    IF (ISW.EQ.1) GO TO 295
0073                    GO TO 66
0074              296   ISW=1
0075                    GO TO 222
0076              295   I8=1
0077                    WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0078                    GO TO 99
                C       ADD FUNCTION
0079              203   IF (I8.NE.1) GO TO 300
                C       READ FINDER FILE
0080                    READ (8'I8,297) (C(I),M(I),N(I),I=1,100)
                C       READS INPUT DATA CARD
0081              300   READ (5,34,END=390) CTLKEY,REC3
                C       MULTIPLE CARD INPUT
0082                    IF (CTLKEY.EQ.CTLKY1) GO TO 66
0083                    ICT=1
0084                    ICT2=16
0085              392   L=1
0086                    I=1
                C       COMPARE CONTROL-KEY OF FINDER FILE AND INPUT DETL
0087              304   IF (C(L).LT.CTLKY1) GO TO 301
0088                    IF (C(L).GT.CTLKY1) GO TO 302
0089                    WRITE (6,303) CTLKY1
0090              303   FORMAT (A8,'DUP RECORD')
0091                    IF (ISW.EQ.1) GO TO 391
0092                    GO TO 66
                C       UPDATE ADDRESS FOR FINDER FILE
0093              301   I=L
0094                    L=N(L)
0095                    GO TO 304
                C       NEW RECORD TO BE ADDED TO FILE
0096              302   K=1
                C       IS THIS EMPTY FROM A PREVIOUS DELETE
0097              312   IF (C(K).EQ.0) GO TO 310
0098                    K=K+1
0099                    IF (K.LT.101) GO TO 312
0100                    WRITE (6,311) CTLKY1
0101              311   FORMAT (A8,'ARRAY FULL')
0102                    GO TO 391
                C       CREATE ADDRESSES FOR FINDER FILE
0103              310   M(K) =
0104                    N(K) =(I)
0105                    N(I) =
0106                    C(K) =TLKY1
0107                    I=K
                C       WRITE DISK FILE
```

```
0108                  WRITE (7'I9,206) REC4,CTLKY1
0109                  IF (ISW.EQ.1) GO TO 391
0110                  GO TO 66
                    C     END ROUTINE
0111                390 ISW=1
0112                  GO TO 392
0113                391 I8=1
0114                  WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0115                  GO TO 99
                    C     CHANGE FUNCTION
0116                204 IF (I8.NE.1) GO TO 400
0117                  READ (8'I8,297) (C(I),M(I),N(I),I=1,100)
                    C     READS INPUT DATA CARD
0118                400 READ (5,34,END=490) CTLKEY,REC3
                    C     MULTIPLE CARD INPUT
0119                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0120                  ICT=1
0121                  ICT2=16
0122                492 L=1
0123                  I=1
                    C     COMPARE CONTROL-KEY OF FINDER FILE AND INPUT DETL
0124                404 IF (C(L).LT.CTLKY1) GO TO 401
0125                  IF (C(L).EQ.CTLKY1) GO TO 402
0126                  WRITE (6,403) CTLKY1
0127                403 FORMAT (A8,'NO MATCH')
0128                  IF (ISW.EQ.1) GO TO 491
0129                  GO TO 66
                    C     GET NEXT FINDER RECORD
0130                401 I=L
0131                  L=N(L)
0132                  GO TO 404
                    C     USE FINDER FILE ADDRESS AND WRITE NEW RECORD
0133                402 I9=M(L)
0134                  WRITE (7'I9,206) REC4,CTLKY1
0135                  IF (ISW.EQ.1) GO TO 491
0136                  GO TO 66
                    C     END ROUTINE
0137                490 ISW=1
0138                  GO TO 492
0139                491 I8=1
0140                  WRITE (8'I8,297) (C(I),M(I),N(I),I=1,100)
0141                  GO TO 99
                    C     RELATIVE ADDRESS DISK ROUTINE
0142                 14 IF (TYPE5.EQ.TYP) GO TO 13
0143                  IF (TYPE5.EQ.TYP1) GO TO 15
0144                  IF (TYPE5.EQ.TYP2) GO TO 16
0145                  IF (TYPE5.EQ.TYP3) GO TO 16
                    C     CREATE FUNCTION
```

```
0146            13 READ (5,34,END=95) REC3,CTLKEY
                C     MULTIPLE CARD INPUT
0147                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0148                  I9=CTLKY1
                C     WRITE DISK FILE
0149                  WRITE (7'I9,42) REC4,CTLKY1
0150            42 FORMAT (16A4,2X,I6)
0151                  ICT=1
0152                  ICT2=16
0153                  GO TO 66
                C     END ROUTINE
0154            95 I9=CTLKY1
0155                  WRITE (7'I9,42) REC4,CTLKY1
0156                  GO TO 99
                C     DELETE FUNCTION
0157            15 READ (5,34,END=96) REC3,CTLKEY
0158                  I9=CTLKY1
                C     READ MASTER FILE
0159                  READ(7'I9,42) REC5,CTLKY2
0160                  I9=CTLKY2
0161                  CTLKY2=0
                C     WRITE DELETED MASTER
0162                  WRITE (7'I9,42) REC5,CTLKY2
0163                  GO TO 66
                C     ADD FUNCTION
                C     CHANGE FUNCTION
0164            16 READ (5,34,END=90) REC3,CTLKEY
                C     MULTIPLE CARD INPUT
0165                  IF (CTLKEY.EQ.CTLKY1) GO TO 66
0166                  I9=CTLKY1
                C     WRITE NEW RECORD
0167                  WRITE (7'I9,42) REC4,CTLKY1
0168                  GO TO 66
                C     END ROUTINE
0169            96 I9=CTLKY1
0170                  READ(7'I9,42) REC5,CTLKY2
0171                  I9=CTLKY2
0172                  CTLKY2=0
0173                  WRITE (7'I9,42) REC5,CTLKY2
0174                  GO TO 99
0175            90 I9=CTLKY1
0176                  WRITE (7'I9,42) REC4,CTLKY1
0177                  GO TO 99
0178            99 RETURN
0179                  END
```

APPENDIX IV

ERROR	ACTION
Column 11-14 of a parameter card contains '0001' and two cards are entered with same control-key	The first card entered updates the file, all others are dropped 'DUP REC'
An input data card is read with a control-key that does not match any existing records on the file to be updated	The input data card is dropped from processing 'MISSING REC'
An input data card is read in, in an 'ADD' run in which the control-key matches a record already on the file	The input data card is dropped from processing 'DUP REC RD'
Column 11-14 of parameter card contains '0002' and only one input data card is entered with the same control-key	The first input data card is used with the remainder of the record blank 'MISSING REC RD'
Column 1 of data card for index sequential file does not contain the letter 'A'	The input data card is dropped from processing 'COL 1 NOT A'
Column 77-80 of input data card does not have numeric information in relative disk address field	The input data card is dropped from processing



OPERATIONAL FLOW CHART

BIBLIOGRAPHY

1. MAYNARD, J., "Objectives of Program Design", Software Age, August and September 1970.
2. WILKES, M. V., "Computers Then and Now", Journal of the Association for Computing Machinery, Volume 15, Number 1, January 1968
3. IBM S/360 Job Control Language Reference. Form number GC28-6704-0. IBM Corporation, Programming Publications, Dept. D58, Poughkeepsie, N. Y. , 1970
4. IBM S/360 Introduction to Direct Access Organization Methods, Form number C20-1649-2. IBM Corporation, Technical Publications Dept., 112 East Post Road, White Plains, N. Y. 10601
5. IBM S/360 Job Control Language, Form number GC28-6704-0. IBM Corporation, Programming Publications, Dept. D58, Poughkeepsie, N. Y., 1970
6. IBM S/360 FORTRAN IV Language, Form number GC28-6515-7. IBM Corporation, Programming Publications, New York, N. Y., 1968.

VITA

Arthur B. Comstock was born on December 3, 1937, in the town of Nanticoke, Pennsylvania. His parents are Mr. and Mrs. Arley B. Comstock. He attended Bloomsburg State College and received a B. S. in education in August 1961. He taught Physics and Chemistry at Neshaminy Senior High School in Langhorne, Pennsylvania for one year and since that time has been employed as a Programmer and Systems Analyst at Air Products and Chemicals, Inc. at Trexlertown, Pennsylvania.

GENERALIZED SUBROUTINE SYSTEM FOR FILE PROCESSING IN EITHER BATCH OR ON-LINE MODE

by
Arthur B. Comstock

ABSTRACT

This paper describes the construction and use of the File Processing System designed and implemented on the IBM System 360 with Full Operating System. The File Processing System is a collection of five subroutines written in the Fortran programming language which enable a computer user to create and maintain a data file on either tape or disk storage media and access the file sequentially for tape and either sequentially, index sequentially, or randomly for disk.

The design of the File Processing System is such that it is easily adaptable to either a batch processing enviroment or and on-line conversational mode. It is also modular in design and routines can easily be added to extend its capabilities or improve upon its already existing features. [1]

The primary objective in the development of the file processing technique was to enable a person with a basic Fortran programming course as background, to construct and maintain a data file with a minimum of effort. The subroutines contained in this system utilize the most current computer technology available both in the hardware and software areas, and thus these advanced techniques are available to all users regardless of their prior programming experience.

GENERALIZED SUBROUTINE SYSTEM FOR FILE PROCESSING IN EITHER BATCH OR ON-LINE MODE

by
Arthur B. Comstock

ABSTRACT

This paper describes the construction and use of the File Processing System designed and implemented on the IBM System 360 with Full Operating System. The File Processing System is a collection of five subroutines written in the Fortran programming language which enable a computer user to create and maintain a data file on either tape or disk storage media and access the file sequentially for tape and either sequentially, index sequentially, or randomly for disk.

The design of the File Processing System is such that it is easily adaptable to either a batch processing enviroment or and on-line conversational mode. It is also modular in design and routines can easily be added to extend its capabilities or improve upon its already existing features. [1]

The primary objective in the development of the file processing technique was to enable a person with a basic Fortran programming course as background, to construct and maintain a data file with a minimum of effort. The subroutines contained in this system utilize the most current computer technology available both in the hardware and software areas, and thus these advanced techniques are available to all users regardless of their prior programming experience.