COMPUTER SYSTEM MANUAL
CSM|UM|15-78
VOLUME I
1 SEPTEMBER 1978

(12)

LEVEL III

COMMAND & CONTROL TECHNICAL CENTER

NMCS INFORMATION PROCESSING SYSTEM

360 FORMATTED FILE SYSTEM (NIPS 360 FFS)

VOLUME I

INTRODUCTION TO FILE CONCEPTS

USERS MANUAL

DEFENSE COMMUNICATIONS AGENCY

COMMAND AND CONTROL TECHNICAL CENTER

Computer System Manual Number (14) CCTC-CSM-UM-15-78-VOL-1

(11) 1 Sep[illegible] 1978

(6)

NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (NIPS 360 FFS)

(12) 154 p.

Users Manual

Volume I, Introduction to File Concepts

SUBMITTED BY:

(10) CRAIG K. HILL
Captain, USA
CCTC Project Officer

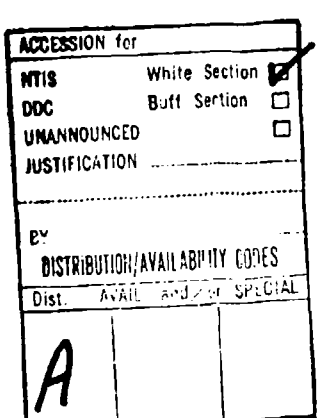
APPROVED BY:

FREDERIC A. GRAF, JR.
Captain, U.S. Navy
Deputy Director,
NMCS ADP

409 098 AB

ACKNOWLEDGMENT

This manual was prepared under the direction of the Chief for Programming with general technical support provided by the International Business Machines Corporation under contracts DCA 100-67-C-0062, DCA 100-69-C-0029, DCA 100-70-C-0031, DCA 100-70-C-0080, DCA 100-71-C-0047, and DCA 100-77-C-0065.



CONTENTS

CONTENTS

CONTENTS

ILLUSTRATIONS

ABSTRACT

This volume presents System Concepts and System Use; it shows a sample NIPS 360 FFS Data File, the Glossary of Terms, and a description of the NIPS 360 FFS Data File and File Format Table.

The NIPS 360 is the total system composed of the S/360 hardware and S/360 Operating System (OS) used to support NIPS 360 FFS software.

This document supersedes CSM UM 15-74, Volume I.

CSM UM 15-78, Volume I is part of the following additional NIPS 360 FFS documentation:

CSM UM 15-78	Vol II	- File Structuring (FS)
	Vol III	- File Maintenance (FM)
	Vol IV	- Retrieval and Sort Processor (RASP)
	Vol V	- Output Processor (OP)
	Vol VI	- Terminal Processing (TP)
	Vol VII	- Utility Support (UT)
	Vol VIII	- Job Preparation Manual
	Vol IX	- Error Codes
TR 54-78		- Installation of NIPS 360 FFS
CSM GD 15-78		- General Description

SECTION 1

INTRODUCTION

This volume is divided into five sections. Section 1 presents a general introduction of the concepts and applications of the NIPS 360 Formatted File System.

Section 2 discusses the concepts of data storage in a formatted file, the methods used for data validation/conversion, and the general language specifications employed.

Section 3 discusses the method by which the system operates and procedures used in developing the data validation/conversion routines which are defined, by the user for specific file applications.

Section 4 defines a sample data file which will be used in examples throughout the system documentation.

Section 5 contains a glossary of terms used in the documentation.

Appendix A contains a detailed explanation of the physical layout of NIPS 360 FFS data set which is the user's data file.

Appendix B contains a description and use of the transaction records output by the file analysis statistics capability.

1.1 System Components

The NIPS 360 FFS is made up of several relatively independent components, each of which performs a function in relation to data files of the system. The total complex of components, working together, provides the user with the

ability to perform the complex file processing job required in modern information management systems. Although comprehensive descriptions of each of the components are presented in the appropriate volumes of the NIPS 360 FFS Users Manual, a brief introduction to each is included in this section, since reference is made to the components in establishing the file processing and language rules covered in this document.

a. File Structuring (FS) Component - This component establishes the necessary communications media required by the balance of the system in data file processing. This communications media is called the File Format Table. Simply stated, a tabular array of the essential attributes of each of the user-described data elements is created by the component. This array is stored as part of the data file and is accessed by the other components when processing user language statements.

b. File Maintenance (FM) Component - This component generates and/or updates the user's data files. Several user languages are provided which permit the analyst to specify data validation procedures, logical data examination and/or manipulation, and summarization. Although the normal output of the process is a data file in updated form, the analyst may request additional "auxiliary" output files which are created as a by-product of the maintenance function.

c. Retrieval and Sort Processor (RASP) - The Retrieval component is an analytical tool used to extract information from one or more data files. This component has the capability to sequence the extracted information in a variety of ways determined by the requirements of the final output report to be produced.

d. Output Processor (OP) Component - This component is used for formal report production and provides a convenient method of listing, summarizing, formatting and counting data elements. Control

mechanisms are provided which permit preparation of reports of extremely complex structure. The data source used in this report production may be either a data file, or the answer set produced by the RASP component.

e. Terminal Processing (TP) - This component is actually composed of three major subsets in the current version of the system. The first is the programs required to interface with the graphic display devices. As such, the system user is relatively unaware of its existence. The second is the Quick Inquiry Processor (QUIP), which provides the user with the capability to interrogate data bases. Functions performed are similar to those performed by RASP and DP. This capability may be utilized from the batched job stream as well as from terminals. The third major subset of this component is Source Data Automation (SODA) which provides the capability of maintaining data files from terminals. Input data may be edited, corrected, and processed with prestored FM logic statements.

f. Utility Support (UT) Programs - This is a collection of general-purpose, utility programs which may be utilized by the analyst in the performance of his job. Significant among the varied capabilities provided, is the data conversion function accomplished by a set of programs of this component. This capability provides the simple and almost automatic method by which the user analyst may directly convert a 1410 NIPS data base to NIPS 360 format.

Each component mentioned above is discussed in detail in a separate volume of this manual (see listing in the Abstract).

1.2 Interrelation of Components

Because of the flexibility of the system, it is difficult to establish specific relationships between the various components. The following logical flow of information through the system should be considered a "typical" or normal example; however, it must be clearly understood that the example is in no way restrictive. Most of the system components may be used in combination with other components to build complex system functions. The various logical relations will become more apparent to the user analyst as he follows through the detailed descriptions of the various components.

FS accepts the user's description of the data elements making up the data file in punched card form. Output from the component is the File Format Table (FFT) which defines the structure of the file to be formed. Since the FFT is an actual physical part of the data base, file initiation is performed by this step.

FM accepts the FFT as a part of its input; together with transaction data the user desires to place in his file. Using the user's instructions (logic statements), it performs the actual update function which results in the updated (or new) data file. Paralleling this process, various forms of "auxiliary" output may be produced under user control.

The retriever may then be utilized to extract information from the data file. The result of this step is the creation of two data sets: one containing the records extracted from the data file, and the other consisting of the sort or sequence control fields the user specified as desired for answer sequencing. A standard sort is applied to the sort fields, and the resultant file is retained along with the data file created by the retrieval operation. Since the sort field file includes "pointers" back to the data file, a direct access technique of recovering the retrieved data is applicable.

This composite of two files is then passed to the Output Processor, which by applying user-supplied instructions

provides the desired final report. Note that the output processor may accept a data file directly, rather than first applying the retrieval process. This technique is useful when the sequence of the output in the final report is not critical or when it is the same as the original sequence of the data file.

System formatted output may be obtained with the Quick Inquiry Processor (QUIP) which can also perform a retrieval function. Using either a data file or the results of a retrieval run as a data source, output reports are quickly and simply prepared.

The TP component utilizes local 2250, 2260 and 3270 devices and remote 2260, 3270 or 1050 terminals as input/output units. Data files may be queried and reports formatted or the files may be updated. Output data may be reviewed in a conversational mode at the terminals or may be directed to printers. This processing will generally parallel the processing by other system components.

With this brief introduction, the rest of this volume addresses the general concepts applicable to the total system, and generally provides those common guides required for use of any component.

Section 2

SYSTEM CONCEPTS

NIPS 360 FFS is a generalized file-handling system. Using languages which have been specifically designed to support the requirements of the users of the various components, the analyst can define the capabilities to process a specific data file. This section presents a brief outline of the concepts of a NIPS 360 FFS data file, the method of handling data elements, and the general system language specifications.

2.1 General File Organization

A data file created by a user with NIPS 360 FFS is a collection of information pertaining to a common area. The file consists of records, each of which contain data describing the attributes of a single subject. For example, the sample file presented in section 4 is a data file containing information describing the status and disposition of all military units in the armed forces. Each record in the file contains data which completely defines a single unit. Thus the file is a collection of records with an order determined by a unit identification code.

Each record in a data file has a common format. This format is defined by the user and communicated to the system through the use of the FS component. The format of a file refers to the format of data records in a specified file. Each location in a record, where a data value is stored, is called an element of the record. When the file is being designed, the user assigns a mnemonic name to each element in the record. The collection of element names, along with their functional relationship, constitute the format of a record and hence the file itself.

The complete description of a file's format is maintained in the FFT which is generated by the FS component. During file processing, the user states his problem using the mnemonic element names to reference data locations in a record. The system translates these names through the FFT into internal code allowing access to actual record data.

Examples of usage of the various concepts covered in the following subsections are provided by Section 4, Sample NIPS 360 FFS Data File.

2.2 Data Record Organization

2.2.1 Data Record Elements

The locations in a record, where data values are stored, have been defined as elements of the record. An individual element is called a field. This is the term used to identify a portion of the data record where a single data item, such as an individual's name, may be stored. During the file definition process, this field is given a mnemonic name which is stored as an entry in the FFT. When the file is processed, the use of the field name in a language statement permits the user to operate on the data contained in a specific location of all records in the file. All the individual elements in the data record are defined by the user as fields and given unique names. This provides the system with a complete map of the data organization in a record.

Occasionally, several adjacent fields in a data record have a logical relationship, and it would be desirable to operate on them as a single item with one name. In such a case, one or more adjacent fields may be defined together as a group with a new name supplied. An example of this would be the case where two fields have been defined to contain an individual's last and first name, respectively.

These two fields could be defined together as a group for one-step data manipulation.

2.2.2 Data Record Element Hierarchy

In conventional information systems, the record is the basic unit of information containing a fixed number of element values. The NIPS 360 FFS permits the user to define a data record with a hierarchical relationship among the elements of the record. At the lower level, the record may contain a variable number of data values for each element. The term, set, is introduced to define a collection of data record elements at the same level.

2.2.2.1 Fixed Set

The fixed set corresponds to the first level in data record hierarchy. The fixed set is a collection of elements (fields) which need only one data value to satisfy requirements. An example of a fixed set element would be the field (element) of the sample file in section 4, COMDR, which contains the Commanding Officer's name. Since each record of this file contains the information on a single military unit, there will be only one Commanding Officer.

2.2.2.2 Periodic Set

In a data record there may be a collection of data elements which may assume more than one set of data values within the record itself. The collection of data elements is called a periodic set. A periodic set is a collection of data elements which are logically related and may contain multiple data entries, all with the same format.

A collection of data values whose format is defined for the periodic set is called a subset. The number of subsets for a periodic set in a data record is under the control of the user. A point of importance is that each subset is a collection of data with the same format as all other subsets of the same periodic set.

The NIPS 360 FFS allows the user to define a record format which consists of one fixed set (from 1 to 100 fields) and up to 255 different periodic sets (each of which may have from 1 to 100 fields defined). (See figure 1.)

2.2.2.3 Variable Set

The NIPS 360 FFS permits the user to define one or more variable sets for a data record format. The variable set is at the same level in the record as a periodic set. Its purpose is to allow the storage of variable length data, which cannot be formatted, in the record. Only one element is defined for the variable set and that element has the characteristics of a field with unlimited length. Data may be added to or deleted from the variable set of a data record.

2.2.2.4 Data Record Identification

Since data records identify a unique subject, a unique record identification must be provided. The user must define one or more elements of the fixed set to be used for record control. The data value(s) found in this record element(s) must be unique throughout the file. Very often the data, and the elements used for such a purpose, are known as the Record Control Group, Record ID, or Record Key.

2.2.2.5 Data Record Organization Summary

This subsection uses figure 1 as a graphic example for the points covered. Shown at the top of the figure is a block diagram representing a data file which may consist of a variable number of records. For purposes of illustration, one of the records in the file is "broken out" to show its possible configuration. The data format in this record is the same as that used by all records in the file. However, the data contents of the record, as well as the number of data entries, may differ from record to record.

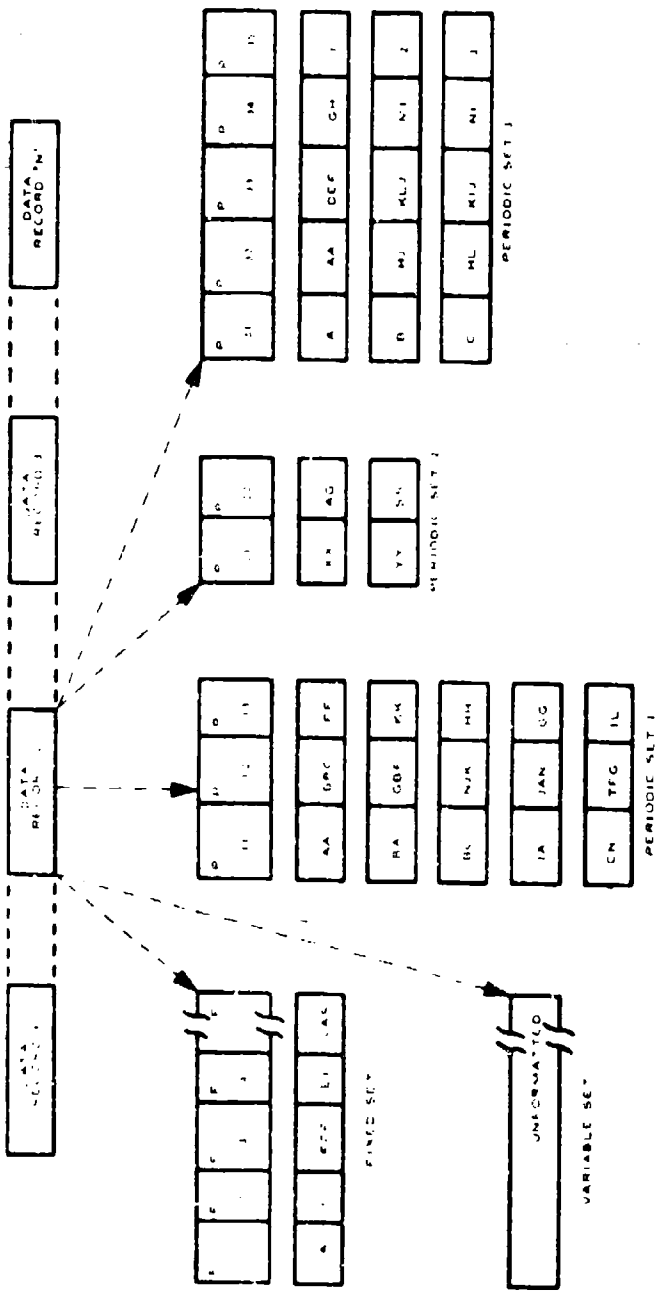


Figure 1. NIPS 360 FFS Data Record Organization

This file has four elements defined as a fixed set. These elements were defined as fields during FS with names associated with each field. For example, the names F1, F2, F3, and F4 are used. When the record is created by the FM component, the user can cause data from incoming transaction records to be placed in the fixed set of the record by using the field name as reference.

The file record shown in figure 1 has formats defined for three periodic sets. The format and data used in Periodic Set 1 will be used for illustration. When the user defines the file format (data record), three logically related elements could contain multiple groups of data values within a single record. Therefore, during the definition of the fields P11, P12, and P13, the user defined that the fields be treated functionally as Periodic Set This then established the common format which groups of values would follow as they are entered into the re. Each group of data values, conforming to the format for Periodic Set 1, is referred to as a subset. The number of subsets contained in a record's periodic set is never defined by format. For Periodic Set 1, as shown in figure 1, there exists five subsets of data. When NIPS 360 FFS is processing file records, a single subset in a periodic set is referenced at one time. Therefore, the use of the field name, P12, in a retrieval statement has sequential access to five different data values in one record.

In the variable set illustrated, no format is established for any data values. However, if a data file is to have records containing variable sets, this must be defined in the FS run to establish internal pointers in the record. Any data that is placed in the variable set for a record is maintained by internal pointers describing to the system, the actual location, and volume of information.

The sizes of data records in a NIPS 360 FFS data file may vary. If a file consists only of a fixed set, then all records in the file are of constant length. However, a data file defined with one or more periodic sets for its records will most likely have record lengths that vary considerably. This occurs since the periodic sets of some records will contain more subsets of data than others.

The maximum size of a data record is also a variable. For the Output Processor, File Maintenance, and Quick Inquiry Processor components, the system allocates space called a "processing block" to contain the part of the data record processed during the run. The core allocation size for the processing block is variable; the size allocated is determined by the specific component. The default size allocated by FM is 16,000 bytes, and the default size allocated by QUIP is 10,000 bytes. The analyst is thus assured the capability of processing complete records of a size up to 10,000 bytes in QUIP, and up to 16,000 bytes in FM. This constraint is a "worst case" condition, since the system only loads that portion of the file record that is being processed during the job, causing the record to be loaded. Loading is performed on a set basis, so that a job requiring examination of data from Periodic Sets 1 and 2 of a file requires the system to load the fixed set, Periodic Set 1, and Periodic Set 2.

Effectively then, the analyst may choose to constrain his file record size to 10,000 bytes and avoid any further considerations of processing requirements related to core size. When using FM, the analyst can determine the size of the processing block by putting

```
PARM='PBSIZE=nK'
```

(where n can range from 1 to 99) on the FM EXEC card. Similarly, when using QUIP in the batch mode, the analyst can enter PARM='PBSIZE=nK' on the QUIP EXEC card; (however, because of design constraints, the QUIP processing block cannot exceed 99K). For source direct QUIP runs against ISAM files (this includes on-line QUIP), the system will compute the size of the processing block required (up to 31K) and allocate that size. If a file design logically requires larger record sizes, the analyst may still process that file just as long as the combination of sets he desires to process in a single job can be contained within the processing block allocation of the system.

2.3 Data Value Modes

The user of NIPS 360 has the option of selecting different modes by which data will be stored in the record elements of the data file. During FS, each element in the record's format is defined to hold its data value in a specific mode. This mode selection specifies the internal method by which data is stored. It is necessary for employing and limiting certain types of operations against data during file processing.

2.3.1 Numeric Mode

Record elements (fields and groups) containing numeric values, which will be processed using mathematical operators (e.g., summations), should be defined as numeric mode. Field elements defined as numeric are limited to a maximum of 10 integers within the range of ±2,147,483,647. Although correct processing can be performed, numeric fields should generally not be defined within a group since system efficiency will be impaired. Normally, all fields defined as numeric, regardless of size, are stored in the data record as binary words. This mode permits fixed point binary arithmetic to be used by the system and allows full use of the more efficient binary set of machine instructions. When a numeric field is defined in a group, the value contained in the field is represented as zoned EBCDIC bytes. Required data conversions are made by the system without user intervention. Note that a numeric field defined within a group is initialized to EBCDIC blanks. It is the user/analyst responsibility to initialize these fields to EBCDIC zeros during FM processing. Failure to initialize these fields will result in data exception errors when using these fields with arithmetic operators and data value editing during output processing.

Any field or group defined as numeric mode will allow output editing to be defined by the user. This function permits leading zero suppression, decimal point insertion, and so forth. Subsection 2.5 discusses the use of the Edit function in NIPS 360 FFS.

The numeric mode specifies that data values are to be right-justified for a record element. This means that if a numeric value is shorter than the defined element, the value will be right-justified with zero padding on the left to fill in the rest of the allocated space. If the numeric value is longer than the defined element, truncation will take place on the left when the data is stored.

2.3.2 Alphameric Mode

Record elements (field and groups) which are defined as alphameric mode, permit all characters of the EBCDIC set to be stored as bytes. All logical operations can be performed on data stored in this mode. However, the data may not be used as values in mathematical processing (e.g., addition, subtraction, etc.), nor may the data be edited with a user-defined mask during OP.

Record elements defined as alphameric mode imply that data stored in them is left-justified. For example, if a data value is shorter than the field or group where it is to be stored, the value will appear left-justified in the location with trailing blanks. If the data value is longer than the field or group in the record, it will be stored with truncation occurring on the right.

The system assumes the alphameric mode for all variable sets in the data file.

2.3.3 Geographic Coordinate Mode

A special item mode designator, coordinate, is used for cases where geographic coordinates are to be stored in the data record for retrievals using the geographic retrieval operators, Circle Search, and Polygon Overlap. This mode may be used for both field and group definitions, depending upon the manner in which the coordinate values are stored. Each term in a coordinate pair defines a point which will be stored as a binary word in the data record. A standard system subroutine will be used automatically to translate the coordinate values to and from a binary word format when

the standard external format is followed. The user may define the coordinate point containing both latitude and longitude as a single field and the system will automatically generate two binary words to hold the values after conversion. He may also define the latitude value and longitude values as individual fields and then define them together as a coordinate group. The standard external format is shown below:

Latitude	Longitude
AAMMX (5 bytes)	BBBMMY (6 bytes)
AAMMSSX (7 bytes)	BBBMMSSY (8 bytes)

where

A = Latitude in degrees

B = Longitude in degrees

M = Minutes

S = Seconds

X&Y = Appropriate hemispheres.

If a user wished to define a coordinate value in his record with the latitude and longitude as individual fields with precision only to minutes, he would define two coordinate fields with lengths of five and six bytes, respectively. Then the two fields would be defined as a coordinate group.

If the user wished to define a single field containing a coordinate point with precision to seconds, he would define a coordinate mode field with a size of 15 bytes.

The coordinate mode may be used for a group containing several fields and/or groups of coordinate data. This

permits the use of a single name defining a line or area to be used with the polygon overlap search operator in the NIPS components. Such groups, however, are not subject to automatic input or output conversion by the system. Only field/groups whose external length is 5,6,7,8,11, or 15 will be automatically converted.

2.4 Data Value Conversion

The user has the capability of defining routines which may be used to perform data value conversion as data is placed into or taken from a record. Data may also be validated either as a transaction item or as it resides in a record using this technique.

The conversion routines may be developed in two ways. In one method, the user actually writes a subroutine using one of the OS/360 programming languages to perform the desired conversion process. The subroutine is written to accept, through a calling sequence, the data item to be converted. It returns the converted data value to the calling sequence when finished. The other method available to the user is to develop a table consisting of a collection of argument-function pairs of data. The argument, being the data to be converted and the function being the converted data value, is supplied as a group on each source card. Both methods have an appropriate cataloged procedure which is used to develop the actual executable load modules using source statements as input. These resulting load modules are placed on a predesignated library for use by all components of the system. When the subroutines/tables are developed by the user for an application, they are defined for use in either input conversion or output conversion. In addition, they are defined to accept data and supply data with specific lengths and modes. An input conversion subroutine/table is used to accept data input from either a system work area, a transaction record during update processing, or a query statement and produce a result compatible for direct placement in a data record field or group. An output conversion subroutine/table is used during output processing to accept a data value from either a data

record element or system work area to supply the converted result for output.

The use of conversion subroutines/tables may be either automatic or under control of the user through the language statements defining the particular run. The following comments describe the methods by which the conversion routines are called into action.

During FS each field and group in the record may be flagged with the name of an input and/or output subroutine/table. This definition, at FS time, will cause the automatic use of the conversion routines whenever the field or group names so flagged are mentioned in the language statements of the RASP, OP, and QUIP components. The user may negate their automatic use in a run by associating a special term with the field or group name when mentioned in a statement. Conversely, the user may override the specified conversion subroutine/table and substitute another one by providing the new subroutine/table name with the field/group name in a statement.

All components of the system which perform file processing allow the user the capability to dynamically state in his language statements the use of a conversion subroutine/table for a particular field or group. Thus conversion may be effected for special applications with a data file. This technique also permits subroutines/tables to be used for data validation or direct conversion in a data record using the FM component.

2.5 Data Value Editing

Numeric mode elements in a data record may be edited during output processing. This option permits the user to suppress leading zeros, insert decimal points, and perform other editing functions. To define the editing function performed on a record element, the user constructs an edit mask containing control characters. Special characters in the mask indicate to the system the nature of the editing operation.

The user may define the editing function to the system in two different ways. The first method is to define an edit mask for a record element when the file is structured using the FS component. The FFT entry for this element will then always carry the edit mask for use by the OP components. If the edit mask is defined in this manner, QUIP and OP will automatically use it whenever the record element is referenced for output display.

The second way the user may define an edit operation is to actually include an edit mask as a literal associated with an element in the language statement for a particular application. The procedure used to write an edit mask is defined in subsection 3.4 of this manual.

So far in the discussion of edit masks, it has been assumed that the data value to be edited has come from a numeric mode record element. However the user may employ a different approach to data value editing as follows. Data from a record element may be processed by an output conversion subroutine/table and the result edited by a user defined mask. Care must be taken to ensure that the output from the conversion subroutine is numeric so that it is acceptable to the edit process.

2.6 General Language Specifications

Each system component has its own language which is used by the analyst to define the file processing functions for a computer run. Despite the number of different languages, they may be easily learned by the analyst since they are basically similar and differ only in their application to a problem. This section of the manual is concerned only with introducing the terminology of the languages. Each volume of this manual will define the characteristics and use of the associated language for that component.

2.6.1 Definitions

The following list defines some elementary language terms.

Word	-	A contiguous string of characters, generally considered to be composed of the alphameric set and explicitly restricted to exclude the special characters blank, comma, period, single quote, "at" symbol, and ampersand.
Term	-	Generally used synonymously with word.
Clause	-	A string of words separated by commas and/or blanks. The period is explicitly excluded from the body of a clause.
Statement	-	May contain one or more clauses and is always terminated by a period.
Operator	-	A system reserved word explicitly directing an action. For example, LIST, EQUALS, GREATER, THAN, SUM, etc., are all considered operators.
Connectors	-	Generally restricted to the Boolean connectors AND and OR.
Condition Statement	-	A special case of the general category of Statement, this form implies that the user is requesting the system to test for a specified condition. Implies the existence of an action directive statement, either explicitly or implicitly stated.
Action Statement	-	A special case of the general category of Statement, this form is a user request for a specific system action, and may or may not be preceded by a Condition Statement.

2.6.2 Language Format

Several formats are commonly used in systems work. They are often identified by names such as free-format, comma-

format, and fixed-format. The preferred form generally used in this system is known as free-format. This format by definition offers the following characteristics:

a. Words may be separated by either commas or blanks or both in any combination, and in any number.

b. Statements and/or clauses may run serially from card to card, or more generally, from input record to input record. Words may not be split between records or cards.

c. Statements may be initiated in any character position of the input record, and may terminate in any position.

d. Other than cases in which the sequence of the input statements are related to the sequence of functions required by the system, no sequencing requirements are arbitrarily imposed.

Card columns 1-72 generally contain language statements. Some components offer the capability of providing a card sequence check if the user provided a sequence number of all cards in his source deck in locations 73 through 80.

Some of the components require a parameter string with optional values in the string. Since interrogation of the string is based on a positional relation and identification of the field information is not feasible without this relation, omitted fields must be clearly indicated to the system. When this condition occurs, the basic punctuation rule is changed:

Note: Words may be separated by one or more blanks, or not more than one comma, with or without multiple blanks. The notation "double comma" indicates to the system that a field has been omitted.

The FM component uses a language which deviates somewhat from the conventions outlined above. Because of the power

and flexibility offered by the component, the language resembles that of a computer's assembly language.

2.6.3 NIPS 360 FFS Language Contents

The words or terms used by the analyst to describe a file processing function must conform to the language specification for the appropriate system component. However, all component languages may have an analogy relating them to our own spoken language. For example, in writing a statement to direct a processing function, the words used are similar to the subject, verb, object, and conjunctions in an English sentence. In all of the system component languages, there are two basic types of words. First are the system reserved words which are recognized as indicating specific operations. The combination of these words in a statement define the logic to be used by the system component. In an analogy to the English sentence, these words would be considered the verb indicating the action to be performed and/or the conjunctives indicating the logical relationship of words.

The second major type of words in the NIPS 360 FFS languages are those supplied by the user. These words could be considered analogous to the subject and/or object of an English sentence indicating what is involved in the processing function and the result obtained. The words supplied by the user are of several classes and are discussed below:

a. Names -- Names are used by the analyst to reference a file, subfile, record element or field conversion subroutines, conversion tables, and edit masks. All names are formed under the following rule:

   - A name may be from one to seven characters with no embedded blanks or special characters. The DSNAME parameter of the Data Definition (DD) card which corresponds to the 7-character file name may be a qualified data set name up to 44 characters in length. The file name must be the last segment of the qualified name. The first character must be alphabetic.

All remaining characters may be alphabetic or numeric.

- Names for data files, libraries and index data sets must not end with the characters X, L, or S. Various NIPS components append these characters to the end of the data set name for identification. Their use as the last user supplies character can produce unpredictable results.

- Names for RASP titles, subroutines and tables must not end with the character zero. Zeroes are appended when organizing their object modules for storage on the file library. Their use as the last user supplied character will result in errors and/or unpredictable results.

b. Self-defining Terms and Literals - The user quite often must supply a data value to a systems component directly through the language statement. Two different options are available for this approach, and such words are called self-defining terms and literals.

- A self-defining term is a word made up of a string of characters with no embedded blanks which is interpreted by the system as a data value. The word is recognized as a self-defining term due to its syntactical position with respect to other words in a statement. The following self-defining terms are treated as data values by the system:

  454
  Tank

- A literal is similar in concept to a self-defining term except that it is enclosed within delineator characters to define its width. The delineator used is the single quote sign (although some components allow the

alternate use of an "at" sign). The purpose of the delineator is to allow the definition of data values containing blank and/or special characters. Examples of literals are:

'Heavy Tank'
'F-105'

c. System Work Areas -- Most components of NIPS 360 FFS have intermediate work areas which are used by the analyst to store data values. These work areas are defined in several ways according to the component concerned. Although they are reserved words capable of recognition by the component being used, they are used like names. This is because they function as the subject or object of a sentence; i.e., they do not connote any action to be taken, but merely are used to represent where data may be found or stored.

d. Figurative Constants -- Some components of NIPS 360 FFS permit the user of figurative constants to represent data values. These are reserved words which stand for specific data values and may be used in place of literals or self-defining terms if appropriate. Figurative constant words may be such as:

ZERO
BLANK

As an example of a NIPS 360 FFS language, the following FASP component language statement is illustrated. This is a conditional statement causing search of a data file for qualifying records to be retrieved. The retrieval criterion is indicated by user-supplied data values in the statement itself.

IF AREA EQUAL 'SOUTH VIETNAM' AND SERVICE EQUAL ARMY.

The underlined words are reserved words recognized by the system to cause specific actions to occur. The remaining words are user supplied and defined words indicating the

specific qualification for action. Due to the syntax of the language, the system will interpret the words AREA and SERVICE as data record element names. The word SOUTH VIETNAM is a literal used to introduce a data value to the system through the source language. Likewise, the word ARMY is a self-defining term used to supply a data value.

The special characters such as comma, blank, and period are used by the different component languages for special usage and have special significance to the system. The mathematical operators, plus, minus, and equal symbols, portray their normal math function in some uses. Multiplication will be represented by the asterisk and division by a slash. Parentheses are used to logically group clauses. In addition to these direct and straightforward rules, the following special characters are used for the indicated purposes.

Character	Use
# (pound or number symbol) (8-3) punch)	Used to delineate subroutine names in the input source language (other than PS). Used in double form, negates an FPT specification for a subroutine.
/ (Slash) (0-1 punch)	Used to separate numeric digits when indicating partial field notation.
$ (Dollar Sign) (11-3-8 punch)	Used as an "universal" match character in comparison literals.
' (Single Quote) (5-8 punch)	Used to delineate literals. Used in double form, negates an FPT specification for a edit mask.
& (Ampersand) (12 punch)	Used to identify a field name used as an operand of a conditional expression in place of a literal or self-defining term.

#D# (descending sort flag)	Used to identify a field to be sorted in a descending manner in either QUIP or RASP.

Optional use of selected special characters which permit compatibility with 1410 FPS source statements is discussed where applicable in each component volume of this manual.

2.6.4 NIPS 360 PFS Reserved Words

This subsection contains a list of reserved words which are interpreted by the system. They may not be used as names in any language statements.

RESERVED WORDS

A	CLASS	FINAL
ABSENT	CLASSIF	FIND
ADD	COMPUTE	FOR
AFTER	CONTAINS	FROM
ALL	COORD	FURTHER
ALPHA	*COUNT(N)	GE
AND	CREATE	GO
ANY	D	GOTO
ARE	DECIMAL	GREATER
AT	DELETE	GROUP
AVERAGE	DDISK	GROUPID
BEFORE	DISPLAY	GT
BETWEEN	DIV	GTE
BINARY	EARLIER	*HEADER(N)
BLANK	EDIT	HTOTAL
BLANKS	EJECT	IF
BT	EQ	IN
CARD(X)	EQUAL	INCLUDES
CH	EQUALS	INITIAL
CHANGES	EXECUTE	IS
CIR	EXTRACT	*LABEL(X)
CIRCLE	FIELD	LATER
	FIELDS	LE

LIMIT	OR	SUB
*LINE(X)	OVERLAP	SUBFILE
LIST	OVP	SUBRT
LOAD	PAGEND	*SUM(N)
LT	PARAM	SYSDATE
LTE	PER	TAB
MARK	PERCENT	TABLE
MAXIMUM	PRINT	*TAB(X)
MINIMUM	PSCT	TEST
MUL	PUNCH	THAN
NE	QUERY	THAT
NEQ	*RECORD(X)	THE
NLE	RECORDS	THEN
NLINES	REPLACE	TITLE
NLT	SELECT	TO
NLTE	SET	TRAILER
NO	SIGNOFF	VSCTL
NOGO	SIGNON	WITHIN
NOT	SPACE	*WORK(M)
NOTE	START	ZERO
NUMBR	STOP	ZEROS
OP		
OPDATE		

*Note

1. (N) stands for either a blank or the numbers zero through nine.

2. (M) stands for either a blank or the numbers one through nine.

3. (X) represents the digits 1-999 and also digits followed by a letter, e.g., LINE10A

4. The following name prefixes are not allowed: PSSQ, VSET, VSZ.

The name D should not be given to a subroutine or table because this is used to specify descending sort in QUIP and RASP.

SECTION 3

SYSTEM USE

3.1 Cataloged Procedures

When the analyst prepares a job using one of the system components, two basic types of information are supplied to the system to define its function. The first set of information consists of job control statements written using the OS/360 Job Control Language (JCL). These statements are interpreted by the S/360 to define the characteristics of the job such as input/output devices required and the name(s) of the program(s) to be run. Refer to the IBM SRL publication, IBM System/360 Operating System-Job Control Language (Form C28-6539), for a description of JCL. The second set of information supplied consists of source statements written in the language of the required NIPS 360 FFS component which define the specific file processing techniques.

To ease the requirement on the user that he supply all the necessary job control statements whenever a system component is used, cataloged procedures have been prepared. These procedures are sets of previously written job control statements which have been stored in a System Library. Each procedure is given a name which is used by the analyst for a particular job. The use of such a name in a JCL Execute statement causes the system to automatically retrieve the information necessary to define a job to the computer. In the simplest case, a job using the cataloged procedures for the FS component would appear as follows:

```
1.  //JOBXYZ JCB  (Standard Parameters)
2.  //        EXEC    XFS,ISAM=TESTER,LIB=TESTER
3.  //FS.SYSIN DD    *
4.     (FS language source statements)
5.  /*
```

Card 1 -- Is required for each job submitted and must be first in the input deck. It is known as the JOB statement and is used to give the job a name such as JOBXYZ.

Card 2 -- Defines the cataloged procedure used for the job. The name XFS defines a set of job control statements in the library necessary to support the execution of the File Structuring Component. The remaining parameters identify the name and type of file to be structured and the name of the File Library.

Card 3 -- Defines the location where the source input language statements may be found. In this case, the asterisk is a parameter which indicates to the system that the source input immediately follows.

Card 4 -- Is the source language statement(s) written by the user to define the specific functions desired from the component.

Card 5 -- Is a special JCL statement indicating the end of the source statement deck.

The parameters entered on the execute statement (Card 2) are known as symbolic parameters. Their function is to dynamically alter the prestored procedures at execution time. The values entered in this manner replace those that were defined when the cataloged procedure was placed in the Procedure Library.

3.2 Development of Conversion Tables

When the user has the occasion which warrants the conversion of data values from one form to another and the problem lends itself to tabular conversion, the cataloged procedure XTABGEN may be used to easily generate such a table. The input to the procedure XTABGEN consists of cards each of which contain an argument-function pair of data values. The argument is the data value which is to be converted and the function is the data value resulting from conversion. The procedure will accept these source cards supplied by the user and build the table into an executable load module capable of linkage with any NIPS 360 PFS

component. The load module table may be stored in a library along with other tables, subroutines, retrievals, and RITs (Report Instruction Table used by the OP component to direct output processing). The name supplied by the user for the conversion table must conform to system standards and be unique in the library in which it is stored. The table may be called by name for use with any file when it is appropriate.

Information and examples on the manner in which the procedure XTABGEN is used may be found in the Utility Support Programs volume of the NIPS 360 PFS User's Manual.

3.3 Development of Conversion Subroutines

When conversion for record element data is desired, but does not lend itself to a tabular approach, the user may wish to write a subroutine to perform the conversion. The subroutine may be written using any of the OS/360-supported problem processing languages. The subroutine is compiled, link edited, and tested by the user before inclusion in the system. A cataloged procedure XSUBLDR is available to the user for loading the subroutine (in load module form) into a library with NIPS 360 PFS compatible linkage established. Use of this cataloged procedure requires the user to have the tested subroutine as an independent load module on any library. Its location is defined to the cataloged procedure XSUBLDR through a JCL statement. Description on the use of XSUBLDR is found in the Utility Support Programs volume of the NIPS 360 PFS Users Manual.

When writing the conversion subroutines, certain conventions must be followed. The remainder of this section describes such conventions.

The user-written subroutine should be written as a single root segment that is reusable, and the calling sequence for the subroutine from a system component should follow standard OS/360 linkage conventions. Three parameters are provided to the user routine. Parameter one is the entry point to the system subroutine loader.

Parameter two points to the area P2 described below and Parameter three is a cell for return code storage.

```
P2    DC    H 'N'        N = number of argument bytes including
                             trailing blanks or leading zeros

      DC    CLN '....'   argument bytes

      DS    CLM          M = function length
```

The argument and function may be either alphameric, binary full word, coordinate data or EBCDIC decimal (a particular subroutine is designed for a specific type of argument and function combination). No boundary alignment of argument and function areas can be assumed. The output function area should be filled with leading zeros for decimal data and trailing blanks for alphameric data. Decimal data will have 'F' and 'D' sign zone bits.

The function output area immediately follows the argument bytes. The high-order position of this area is P2 + N + 2. Conversion routines must be written to accept variable length alpha, decimal or coordinate arguments. The output function size is fixed for a given routine and should always be completely filled. The combined lengths of the argument and function may not exceed 256 bytes.

Upon return from the user routine, either register 15 can contain one of the following return characters or the cell designated by parameter three can be filled accordingly:

S = Successful

M = No Match, unsuccessful

The subroutine loader entry point is provided to user routines so that they may request loading or linking to other routines. No input/output functions should be performed by the user routine.

When the subroutine is placed on a Work Library, the entry point name and the load module name (PDS member name) must be the same. The names must be identical due to the requirements established for use of the SUBLDR utility program.

3.3.1 Assembly Language Routines

The routine should use the following macro as its first instruction.

```
SUBNAME FFSBEGIN BASEREG
```

This macro will generate the proper CSECT and SAVE linkage. Register 13 will point to a generated SAVE area and should not be used by the conversion routine. Register BASEREG will have been initialized as the routine base register along with the appropriate USING statement. Register 1 will point to the parameter address constant list. When returning control, register 15 may contain the return code as discussed previously and the following macro used to return control.

```
FFSRETRN RC=(15)
```

Otherwise, the byte indicated by parameter three must be filled with 'S' or 'M'.

The following is an example of an assembly language subroutine:

```
//ASMSUB     EXEC  ASMFCL,PARM.LKED='MAP,LIST,LET,DC'
//ASM.SYSLIB DD
//            DD  DSN=FFS.MACLIB,DISP=SHR
//ASM.SYSIN DD *
DTGDS     START
*A DATE CONVERSION ROUTINE
* CHANGES FILE DATE FROM YYMMDDTTTT TO OUTPUT AS DDMMMYY/TTTT
*LOAD BASE REGISTER, SAVE CALLING PROGRAM REGISTERS, LINK CALLING PGM
*
DTGDS     FFSBEGIN 7
          L      8,4(1)                LOAD ADDRESS OF DUMMY SECTION
```

```
         USING PARMLIST,8          INIT REG 8
*
         SR    6,6                 ZERO OUT 6
         LA    6,12(6)             ADD 12 TO 6  PUT IN REG 6
*
*MOVE INPUT DATE TO WORK AREA, REFORMAT DD AND YY
*CONVERT TWO DIGIT MONTH TO SYMBOLIC THREE CHARACTERS
*RETURN AN 'S' SUCCESSFUL OR 'M' UNSUCCESSFUL IN REG 15
*
         MVC   DIGMNTH(2),PARM1POS+2 MOVE MONTH WORK AREA FOR COMPARE
         LA    5,TABLE               LOAD ADDRESS OF TABLE INTO REG 5
LOOP     CLC   DIGMNTH(2),0(5)       COMPARE TWO DIGIT MONTH TO TABLE
         BE    FOUND                 IF EQUAL GO MOVE SYMBOLIC MONTH
         LA    5,5(5)                ADD 5 TO REG 5 and PUT IT IN REG 5
         BCT   6,LOOP                EXIT IF R6 GETS TO ZERO
TOOF     IC    15,=C'M'              UNSUCCESSFUL CONVERT
         MVC   MONTH(3),=C'XXX'      TEMPORARY FIXER ********
         B     DATEOEXT              GO TO EXIT ROUTINE
FOUND    IC    15,=C'S'              SUCCESSFUL CONVERT
         MVC   MONTH(3),2(5)         MOVE SYMBOLIC MONTH TO WORK AREA
DATEOEXT MVC   DAY(2),PARM1POS+4
         MVC   YEAR(2),PARM1POS      REFORMAT YEAR
         MVC   TIME(4),PARM1POS+6    SAVE TIME
         MVC   PARMLNH+12(12),WORK1  MOVE REFORMATTED DATE TO LIST
         PFSRETRN RC=(15)
*CONSTANT SECTION
*
         DS    0F
WORK1    DS    CL12                  WORK
DAY      DC    CL2'  '                   AREA
MONTH    DC    CL3'   '                        REFORMAT
YEAR     DC    CL2'  '                                 DATE
         DC    CL1'/'                AND
TIME     DC    CL4'    '             TIME
*
DIGMNTH  DC    CL2'  '               TWO-DIGIT MONTH WORK AREA
*
SAVE     DS    18F                   AREA TO STORE REGISTERS
TABLE    DC    C'01JAN'
         DC    C'02FEB'
         DC    C'03MAR'
         DC    C'04APR'
         DC    C'05MAY'
```

```
         DC      C'06JUN'
         DC      C'07JUL'
         DC      C'08AUG'
         DC      C'09SEP'
         DC      C'10OCT'
         DC      C'11NOV'
         DC      C'12DEC'
*
*DUMMY SECTION
*
PARMLIST DSECT
PARMLNH  DC      H'10'              ARGUMENT LENGTH
PARM1POS DS      CL10'              '  ARGUMENT
         DS      CL12               FUNCTION MAX SIZE
         DC      CL1' '             RETURN CODE
*
         END
/*
//LKED.SYSLMOD DSN=TESTERL(DTGOS),DISP=OLD
/*
```

3.3.2 COBOL User Subroutines

The subroutine is called as follows:

```
CALL 'SUBNAME' using DUMMY P2 P3.
```

The first linkage parameter is provided for use by assembly language routines only but must be accounted for by COBOL subroutines.

```
LINKAGE SECTION.

     01   DUMMY.
          02 NOTHING PICTURE X.
     01   P2.
          02 ARGLEN PICTURE S(99) usage computational.
          02 ARGFNC PICTURE etc.
     01   P3.
          02 RETURN-CODE PICTURE X.
```

CODE must be filled with 'S' or 'U' to indicate successful or unsuccessful conversion, respectively. ARGLEN contains the number of bytes in the ARGFNC area containing the argument data. Function data should be inserted in ARGFNC immediately following the last argument byte (ARGFNC+N where N=number of bytes in the argument).

The following statements should be inserted in the PROCEDURE DIVISION --

```
    ENTER LINKAGE.

    ENTRY 'SUBNAME' USING DUMMY P2 P3.

    ENTER COBOL.
```

The following is an example of a COBOL subroutine which serves the same function as the ALC conversion subroutine in the previous paragraph.

```
//COBSUBS1  EXEC COBFCL,PARM.COB='MAP,BUF=12282,NOSEQ,LINECNT=50'
//COB.SYSIN   DD *
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. 'PGMNAM'.
000030 ENVIRONMENT DIVISION.
000040 CONFIGURATION SECTION.
000050 SOURCE-COMPUTER.  IBM-360-50.
000060 OBJECT-COMPUTER.  IBM-360-50.
000070 DATA DIVISION.
000080 LINKAGE SECTION.
000090 01  DUMMY.
000100     02 NOTHING PICTURE XXXX.
000110 01  P2.
000120     02  ARGLEN PICTURE XX.
000130     02  IN-YEAR PICTURE XX.
000140     02  IN-MONTH PICTURE XX.
000150     02  IN-DAY PICTURE XX.
000160     02  IN-TIME PICTURE XXXX.
000170     02  OUT-DAY PICTURE XX.
000180     02  OUT-MONTH PICTURE XXX.
000181     02  OUT-YEAR PICTURE XX.
000190     02  SLASH PICTURE X.
000200     02  OUT-TIME PICTURE XXXX.
```

```
001010 01   P3.
001020      02  RODE PICTURE X.
001030 PROCEDURE DIVISION.
001040      ENTER LINKAGE.
001050      ENTRY 'COBSUB'USING DUMMY P2 P3.
001060      ENTER COBOL.
001070 INITIALIZE.
001080      MOVE 'S' TO RODE.
001090      MOVE '/' TO SLASH.
001100      MOVE ZEROS TO OUT-DAY.
001110      MOVE 'XXX' TO OUT-MONTH.
001120      MOVE ZEROS TO OUT-YEAR.
001130      MOVE ZEROS TO OUT-TIME.
001140 CHECK-YEAR.
001150      IF IN-YEAR IS GREATER THAN '99',
001160          OR IN-YEAR IS LESS THAN '00',
001170          MOVE 'E' TO RODE, GO TO CHECK-MONTH.
001180      MOVE IN-YEAR TO OUT-YEAR.
001190 CHECK-MONTH
001200      IF IN-MONTH IS EQUAL TO '01', MOVE 'JAN' TO OUT-MONTH,
002010          GO TO CHECK-DAY31.
002020      IF IN-MONTH IS EQUAL TO '02', MOVE 'FEB' TO OUT-MONTH,
002030          GO TO CHECK-DAY28.
002040      IF IN-MONTH IS EQUAL TO '03', MOVE 'MAR' TO OUT-MONTH,
002050          GO TO CHECK-DAY31.
002060      IF IN-MONTH IS EQUAL TO '04', MOVE 'APR' TO OUT-MONTH,
002070          GO TO CHECK-DAY30.
002080      IF IN-MONTH IS EQUAL TO '05', MOVE 'MAY' TO OUT-MONTH,
002090          GO TO CHECK-DAY31.
002100      IF IN-MONTH IS EQUAL TO '06', MOVE 'JUN' TO OUT-MONTH,
002110          GO TO CHECK-DAY30.
002120      IF IN-MONTH IS EQUAL TO '07', MOVE 'JUL' TO OUT-MONTH,
002130          GO TO CHECK-DAY31.
002140      IF IN-MONTH IS EQUAL TO '08', MOVE 'AUG' TO OUT-MONTH,
002150          GO TO CHECK-DAY31.
002160      IF IN-MONTH IS EQUAL TO '09', MOVE 'SEP' TO OUT-MONTH,
002170          GO TO CHECK-DAY30.
002180      IF IN-MONTH IS EQUAL TO '10', MOVE 'OCT' TO OUT-MONTH,
002190          GO TO CHECK-DAY31.
002200      IF IN-MONTH IS EQUAL TO '11', MOVE 'NOV' TO OUT-MONTH,
003010          GO TO CHECK-DAY30.
003020      IF IN-MONTH IS EQUAL TO '12', MOVE 'DEC' TO OUT-MONTH,
003030          GO TO CHECK-DAY31.
```

```
003040      MOVE 'M' TO RODE.
003050 CHECK-DAY31.
003060      IF IN-DAY IS GREATER THAN '00',
003070          AND IN-DAY IS LESS THAN '32', MOVE IN-DAY TO OUT-DAY,
003080          GO TO CHECK-TIME.
003090      MOVE 'M' TO RODE, GO TO CHECK-TIME.
003100 CHECK-DAY30.
003110      IF IN-DAY IS GREATER THAN '00',
003120          AND IN-DAY IS LESS THAN '31', MOVE IN-DAY TO OUT-DAY,
003130          GO TO CHECK-TIME.
003140      MOVE 'M' to RODE, GO TO CHECK-TIME.
003150 CHECK-DAY28.
003160      IF IN-DAY IS GREATER THAN '00',
003170          AND IN=DAY IS LESS THAN '29', MOVE IN-DAY TO OUT-DAY,
003180          GO TO CHECK-TIME.
003190      MOVE 'M' TO RODE.
003200 CHECK-TIME.
004010      IF IN-TIME IS GREATER THAN '00',
004020          AND IN-TIME IS LESS THAN '2401',
004030          MOVE IN-TIME TO OUT-TIME, GO TO DEPART.
004040      MOVE 'M' TO RODE.
004050 DEPART.
004060     IF RODE IS NOT EQUAL TO 'M', MOVE 'S' TO RODE.
004070     ENTER LINKAGE.
004080     GOBACK.
004090     ENTER COBOL.
/*
//LKED.SYSLMOD  DD  DSN=TESTERL(COBSUB),DISP=OLD,UNIT=2314
//LKED.SYSIN    DD  *
     ENTRY COBSUB
/*
```

Note: The linkage editor control card, ENTRY COBSUB, is necessary for a COBOL subroutine (this name must correspond with the name of the subroutine as defined on the ENTRY statement in the PROCEDURE DIVISION and on the LKED.SYSLMOD DD statement).

3.3.3 PL1 User Subroutine

PL1 user subroutines require an assembly language interface in order to be called by a NIPS application

program. The assembly language routine sets up the dope vectors required to pass parameters to the PL1 subroutine. A complete discussion of assembly language/PL1 interface conventions can be found in the PL1 Programming Guide.

An example of JCL, assembly language interface, and sample PL1 subroutine is shown in the following section of code.

```
//STEP1  EXEC ASMFCP,PARM.ASM='LOAD NODECK',REGION=114K
//ASM.SYSLIB DD DSN=PPS.JOBMACRO,DISP=SHR
//ASM.SYSIN DD * ASEM TYPEIN
//STEP2  EXEC PLIXCL,REGION=120K,PARM.PLI='MAP',PARM.LKED='LET'
PLI.SYSIN DD *  PLISUB TYPEIN
//LKED.SYSLMOD DD DSN=&TEMP(PREDIX),UNIT=SYSDA,SPACE=(TRK,(10,1,1)),
//       DISP=(NEW,PASS)
//LKED.SYSIN DD *
  NAME PREDIX(R)
  ENTRY PREDIX
//STEP3  EXEC XSUBLDR,VLIB='SEP=xxxxxx',LIB=TSTLIB,ULIB=2314,
//       LIBDISP=OLE,BLK=6144
//SYSIN DD *
SUBRT PREDIX 18  100  B A A PREDIX LEWISER
/*
//STEP4  EXEC XSUBCHK,LIB=TSTLIB,ULIB=2314
//  VLIB='SER=XXXXXX'
//SUBCHK.XYZ DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1500)
//SUBCHK.ABC DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1500)
//SUBCHK.PLIDUMP DD SYSOUT=A
//SUBCHK.SYSUDUMP DD SYSOUT=A
//SUBCHK.SYSIN DD *
PREDIX      018
?E5896210764666257
/*


*THIS ASSEMBLY LENGUAGE ROUTINE WAS WRITTEN AS AN INTERFACE BETWEEN
*NIPS AND PLI.
*THE NAME OF THE ROUTINE IS PREDIX. ANY PERSON NEEDING A NIPS/PLI
*INTERFACE CAN USE THIS CODE.
```

```
*
*
PREDIX   PPSBEGIN 12
*        REG3 PONTS TO THE NIPS AREA
         L     3,4(1)
         MVC   INPUT,2(3)
         LA    1,PARMS
         L     15,=V(PLICALLB)
         BALR  14,15
         MVC   20(100,3),OUTPUT
         SR    15,15
         IC    15,RC
         PPSRETRN RC=(15)
PARMS    DC    A(ADLIST)
         DC    A(0)
         DC    A(0)
         DC    A(0)
         DC    A(0)
         DC    X'80'
         DC    AL3(0)
ADLIST   DC    A(IN1)
         DC    A(OUT1)
         DC    X'80'
         DC    AL3(RC1)
IN1      DC    A(INPUT)
         DC    2AL2(L'INPUT)
INPUT    DC    CL18' '
OUT1     DC    A(OUTPUT)
         DC    2AL2(L'OUTPUT)
OUTPUT   DC    AL100' '
RC1      DC    A(RC)
         DC    2AL2(L'RC)
RC       DC    C' '
         END   PREDIX
```

```
PRED :   PROCEDURE(INPUT,OUT,RC)    OPTIONS(MAIN) REORDER;
           DCL XYZ FILE PRINT OUTPUT STREAM;
            DCL SYSPRINT   FILE PRINT OUTPUT STREAM;
              DCL INPUT CHAR(18);
           DCL OUT CHAR(100);
       DCL RC CHAR(2);
```

```
        DCL IC CHAR(2) INITIAL('S ');
        OPEN FILE (SYSPRINT) TITLE ('ABC')
      PUT FILE(XYZ) SKIP EDIT ('  WE HAVE ENTERED THE PLI ROUTINE')
        (A);
      PUT FILE(XYZ) EDIT (INPUT) (A);
       RC = IC ;
      RETURN;
END;
```

3.3.4 User Scan Subroutine

If the System Scan subroutine for use with the Keyword Indexing capability does not meet a user's needs, he can write a Scan subroutine of his own. See the Keyword Indexing section for a description of the System Scan Subroutine. The user must specify the name of his routine in each keyword indexed field entry in the FFT that is to be scanned by it (see Volume II, File Structuring Index Definitions for Keyword Fields). When he codes the subroutine he must conform to general NIPS subroutine interface requirements and he must meet the following Scan Processor interface requirements:

a. The subroutine must distinguish a call to scan a new field from a call to resume scanning a current field. The return code byte in the parameter list may be used for this purpose.

b. The subroutine must accept the following parameter list from the Scan Processor (the controlling routine):

   P - address of the field to be scanned.

   P - scan limit; address of the field plus its length minus 1.

   P - address of the caller's word hold area.

   CL1 - hyphen option byte from the FFT for the field being scanned.

For information about defining this byte in the FFT and for the system interpretation of it, refer to the Index Definition for Keyword Fields section of Volume II, File Structuring - Index Definitions for Keyword Fields, and to the Keyword Indexing section (3.7) of this volume.

The user's scan may ignore this byte or it may interpret it any way it pleases. The byte contains one of the following binary values:

1 - DROP

2 - RETAIN

3 - SEPARATE

4 - TEXT(default)

CL1 - return code byte; the scan subroutine must place a return code in this byte before it returns to the scan processor. The scan processor does not modify the byte unless it loads a new scan subroutine, at which time it sets an end-of-scan indicator.

c. It must return one word in the caller's word hold area in the following format:

CL1 - length of the word.

CL30 - recovered word, left-justified.

It must set bit 0 of the length byte to 1 if the recovered word is a literal. This flag bit is used to bypass dictionary processing of literal words.

d. It must return one of the following codes in the parameter return code byte each time it returns to the scan processor

0 - a word was recovered; there is more data to scan.

4 - a word was recovered; end-of scan.

8 - no word recovered; end-of-scan.

The scan processor will recall the scan subroutine to continue processing the same field if the scan subroutine returns a zero. Otherwise, it will assume that all words have been recovered from the current field. If there are no more fields to process, it will not recall the scan subroutine (i.e., there is an end-of-file).

3.4 Definition of Edit Masks

The user writes an edit mask in a language statement as a literal. That is, single quote signs are used for delineation. The edit capability of NIPS 360 PFS permits the user the following features when applied to a numeric data value:

a. Zero suppression

b. Sign control left or right

c. Leading and trailing significant characters

d. Character insertion.

The remainder of this section discusses the techniques of writing an edit mask.

Any character which can be printed may be used in the edit mask except a quote mark. However, certain characters, namely ampersands, blanks, and zeros, will not appear as such in the output. Furthermore, minus or credit (CR) symbols have special meanings. One character position in the output is represented by one character in the edit mask. Nonspecial characters in the mask will be printed in the same relative position in the output field. A mask may be 132 characters long; however, certain NIPS components have shorter limits. As in most cases, since no more than 10

replaceable characters (blanks or zeros) can be filled by source data, edit masks should tend to be less than 70 characters long.

The actions taken for each special character in the edit mask are given below.

Blank -- Each blank in the edit mask will be replaced by a digit from the source field.

Zero -- each zero in the edit mask will be replaced by a digit from the source field, and the leftmost zero will be the right most limit of zero suppression.

Ampersand -- Each ampersand in the edit mask will be replaced by a blank in the output field.

Minus sign -- If the minus sign is to the left of the first replaceable character or to the right of the last, it is considered a sign control character. If the sign field is negative, the minus sign and any other nonreplaceable characters occurring with it are printed. If the sign is positive, neither the minus sign nor the accompanying characters are printed.

CR -- If the character C is immediately followed by the character R on the left of the first replaceable character or on the right of the last replaceable character, they are considered as sign control characters, and are treated just like a minus sign.

The following examples should clarify the use of these special characters.

Edit Mask	Source	Result
'bbØbb'	12345	12345
	00001	bbbØ1
'XXCR&ØbbXX'	123	bbbbb123XX
	-123	XXCRb123XX
	001	bbbbbb01XX
	-001	XXCRbb01XX
'$.bb-'	12	$.12b
	-12	$.12-
	01	$.01b
	-01	$.01-
'Øb/bb/bb'	010168	b1/01/68

In output, if the size of the source field is known when the edit mask is first processed, a test is made to see whether that many replaceable characters exist. If the source is too long, the edit mask is rejected. If the source is too short, the system will start at the left and replace the blanks and zeros with ampersands until the desired number of replaceable characters remain. This occurs before the test for CR and -, but after the test for zero. Thus, a mask of 0-bbb for which a 3-character source field is specified will cause a 001 field to be printed as bb001.

If the size of the source field is not known when the edit mask is first processed, the system will count the number of replaceable characters and return this number to the calling program.

In File Structuring(FS), the replaceable characters in a defined edit mask and the field which is defined to use it must be of equal length.

3.5 Maintenance of Source Language Programs

In a large installation, keeping track of source decks becomes a real problem. Source decks for NIPS components may be maintained on direct access libraries to facilitate housekeeping and source program maintenance. Source can be stored on a library by the NIPS batch components and the

UTSOURC utility program. From a terminal, the EDIT component of the Terminal Processing System can be used to maintain source libraries.

3.6 Secondary Indexing Capability

Secondary Indexing provides the user with the capability to index any ISAM, SAM or VSAM file by the contents of a field or fields other than the Record ID. The primary purpose of the capability is to provide a faster response time for qualifying data records during retrievals.

Secondary Indexing encompasses three basic areas: Index Specification, which allows the user to specify fields and/or groups which are to be added (or deleted) as indexes; Index Maintenance, which ensures current and accurate indexing information; and Index Processing which uses this information in the retrieval process.

The fields and groups which are to be indexed are designated during File Structuring (FS) or Index Specifier (UTNDXSPC) runs. From that point on, all Secondary Indexing functions occur automatically, transparent to the user, and cannot be overridden, except in the case of retrieval.

The user's prime concern in the use of Secondary Indexing is the judicious choice of index fields. These fields should be chosen on the basis of expected frequency of use in retrievals. Choosing too few indexes would limit the effectiveness of the capability while choosing too many would load down the capability with needless overhead costs.

Since the user interfaces directly with Secondary Indexing only during Index Specification, the rest of this section will discuss the techniques involved in Index Specification.

Regardless of whether an index is specified as part of an FS run or by using the stand-alone Index Specifier (UTNDXSPC), an Index Specification statement is required and includes the items listed below:

- Statement Identifier - INDEX
- Name of field/group to be indexed
- Action to be taken (ADD or DELETE)
- Conversion or analyzer routine, or table (optional)

If a conversion or analyzer routine or table is specified on the Index Specification, the routine or table must have been previously defined on a SUB/TAB statement. This statement contains the following information.

- Statement Identifier - SUB, SUBROUTINE, TAB, or TABLE
- Name of subroutine or table
- Function (CONVERT or ANALYZE)-for subroutines only
- Input and output lengths of data
- Mode of input and output data - (ALPHA, BINARY, COORD, or DECIMAL) applicable only for CONVERT subroutines

As mentioned above, once the indexes for a file are specified, no further user actions are required to use the Secondary Indexing capability. File updating by FM or SODA causes routines to generate corresponding transactions to update the indexes. RASP or QUIP will use the information provided, wherever possible, in order to improve retrieval efficiency.

More detailed information regarding the Index Specification and associated SUB/TAB statements is contained in Volume II, File Structuring. Information regarding usage of indexes will be found in Retrieval and Sort Processor, Volume IV, and Terminal Processing, Volume VI.

3.7 Keyword Indexing Capability

Keyword Indexing is a text-retrieval capability that provides a method by which the NIPS user can access and retrieve records based upon the contents of variable-length or text data fields. Just as Secondary Indexing enables a user to query a file and access only those data records known to contain the fixed-length fields of interest to him, Keyword Indexing enables the user to retrieve records based on the presence of 'keywords' within a field. To provide a capability tailored to his needs, the user has options to direct the selection of 'keywords' from the indexed field. This selection is based on the presence or absence of a stop word table and a dictionary. These tables are user provided and are designated at the field level, although one table may be applied against more than one field.

A stop word table consists of a list of irrelevant (noise) words. Any words found in this table are eliminated from further consideration as keywords. The primary advantage of the stop word table is the reduction of the number of words which must be sorted (and matched against a dictionary if one is specified). A system stop word table named ICKSTOP is available to the user. It contains the following words:

A	BY	IS	OR	USE
AFTER	CAN	IT	BE	USED
ALLOW	COULD	MODE	SINCE	WAS
AM	EVEN	MORE	SO	WE
AN	EVER	NO	THAT	WHEN
AND	EVERY	NOT	THE	WHERE
ANY	FOR	NOW	THEN	WHEREAS
BE	HAS	OF	THIS	WHICH
BEEN	IF	ON	TO	WILL
BUT	IN	ONLY	UNDER	WITH

A dictionary consists of words which are potential index data set entries or synonyms for potential entries. For example, the following values are to qualify as keywords - Aircraft, Bomber, Plane and Fighter. For retrieval purposes these values are synonomous and are entered into the dictionary as such. The index data set will carry the ID of

each record where any of these values occurred in the designated field. Then, when a retrieval is made against this field with a keyword argument of either Aircraft, Bomber, Fighter or Plane, all records containing any of these values in the indexed field will qualify. The maximum value length of a keyword is 30 characters.

When a data word is matched against a stop word table it must exactly match the table word to be eliminated from further processing. When a data word is matched against a dictionary, however, the entire data word need not exactly match a dictionary word because of two functions - suffix processing and suffix specification.

Suffix Processing - Suffix Processing is an automatic programming function that is conditionally applied to all argument words matched to a dictionary. Suffix specification is a control statement notation which permits a user to indicate keyword ending changes when a keyword is suffixed. If a keyword exactly matches the most significant (left-most) characters of a longer argument word, the unmatched argument characters are analyzed to determine if they are a suffix to the matched keyword. The unmatched characters are looked up in a table of acceptable suffixes. If they exactly match a table entry, the ending characters of the keyword are examined to determine if the matched suffix can be validly attached to them. If they pass the test, the keyword is substituted for the argument word. Note that a double substitution could be required; the keyword could be a convert keyword (synonym), in which case the synonym base keyword is substituted for the argument.

Suffix Specification - If a user specifies to the Dictionary Maintenance utility that a keyword's ending changes when the word is suffixed, the changed form of the word is stored in the dictionary as an independent keyword with the restriction that if an argument word exactly matches the changed form of the word, that argument is not a keyword. In other words, the changed form of the keyword must match the significant characters of a longer argument, and the unmatched

argument characters must be a valid suffix (see Suffix Processing). If these conditions are met, the unchanged form of the word (or its associated base keyword if the word is a convert keyword) is substituted for the argument.

Table of All Valid Suffixes

ABLE	ERS	ITION
ABLES	EST	ITIONS
AGE	FUL	ITY
AGED	FULLY	IVE
AGES	IBLE	IVES
AL	IBLES	LIES
ALLY	IC	LY
ANCE	ICATION	MENT
ANCES	ICATIONS	NESS
ATION	ILIES	NESSES
ATIONS	ILITIES	OR
ATIVE	ILITY	R (when preceded by E)
ATIVES	ILY	RS(when preceded by E)
D(when preceded by E)	INAL	S
DS(when preceded by E)	ING	SION
ED	INGS	SIONS
EDS	ION	TIES
ENCE	IONAL	TION
ER	ITIES	TY
		Y(when preceded by L or T)

Literals in the data file are never matched against a stop word table or a dictionary. Each literal automatically becomes a index data set entry. A literal is defined as all characters, including blanks, which appear between two single quotation marks in the same subset record. The user should analyze the content of his file so that he may select those keywords which are best suited for retrieval logic.

For a detailed description of how to generate and maintain these tables, see Volume VII, Utility Support,

Dictionary Maintenance section. Tables are deleted by using the OS Utility IEHPROGM

Use and content of tables should be based on the nature of the data. If no tables are specified, each value found within the indexed field becomes an entry in the index data set. The user should analyze the data content of his file so that he may select those keywords which are best suited for retrieval logic. For efficient operation, extraneous words not used as a basis for retrieval should not be carried as keyword entries in the index data set. If an indexed field holds a large amount of text data, it would most likely be beneficial to have both a stop word table and a dictionary. If a field contains 'keywords' interspersed with noise words, a stop word table would be applied against it. But if a field consists mainly of retrieval based words with synonym or suffix (e.g., plural ending) functions, only a dictionary need be applied.

Keyword indexing adheres to the same basic rules, considerations and functions as does secondary indexing and applies to the three areas of index specification, maintenance and retrieval.

Indexed fields are defined through File Structure, File Revision or the Index Specification utility (UTNDXSPC). Any variable field, variable set or fixed-length alpha field defined for the file may be designated as a keyword indexed field. There is no limit to the number of indexed fields. A fixed-length field may not be defined as both a secondary and keyword index. An Index Specification statement consisting of the following items is required for each indexed field:

Statement Identifier - INDEX

Name of field/set to be indexed

Action (ADD or DELETE)

Text Retrieval - KEYWORD

Stop Work Table (Optional)

Dictionary (Optional)

User-Scan Routine (Optional)

Hyphen Option (Optional)

For each stop word table, dictionary or user-scan routine specified, a SUB/TAB statement must have been included to define the named function. This statement consists of the following information:

Statement Identifier - SUB, SUBROUTINE. TAB or TABLE

Use Function - STOP, DICT, DICTIONARY or SCAN

This action creates Index Descriptor Records in the FFT. The presence of these records causes Index Maintenance to be invoked either through File Maintenance or the Index Specification utility.

The generalized system scan routine in Index Maintenance processes the keyword indexed field(s) of the transaction records looking for textual words and literals. It determines the limits of words by identifying literals and text words.

For literals, the system scan subroutine recognizes a single quote as a literal word delimiter. All characters between the single quotes except double quotes are recovered to form one literal word. Any number of double quotes may appear between the single quotes; all are deleted during recovery. If an odd number of single quote characters appear in a field, no word (literal or textual) will be recovered between the last single quote and the end of the field.

For nonliterals (text words), the system scan subroutine recognizes three categories of characters: textual, superfluous, and word separator.

Textual - Those data word characters that are recovered to form the work passed to the subroutine caller.

Superfluous - Those data word characters which are deleted from the word passed to the system scan subroutine caller. They are imbedded in one word; they separate segments of one word.

Word Separator - Those characters in the data field that are not part of a word. They are word delimiters.

Letters and numbers are always textual. The period and hyphen may be textual, superfluous, or word separators. The comma may be superfluous or a word separator. A single quote is a word separator as well as a literal word delimiter. A double quote is ignored; it is a word separator if another word separator precedes or follows it; otherwise it is superfluous. All other characters are word separators.

Period Rules - A period is recovered as a textual character when it appears in a word composed entirely of numbers. One or more periods are superfluous if they are used to punctuate a symbol (i.e., U.S.A.). The word in which they appear may be composed of letters or letters mixed with numbers. For all other occurrences, the period is a word separator.

Comma Rules - One or more commas are superfluous if they are used to punctuate a number; that is, when they appear in a word composed entirely of numbers or of numbers and a period. For all other occurrences, the comma is a word separator.

Single Quote Rules - Single quote is always a word separator as well as a literal word delimiter.

Double Quote Rules - All occurrences of a double quote are ignored. If it is imbedded in a word, it is superfluous, otherwise it is a word separator.

Hyphen Rules - Treatment of hyphens is dependent upon the hyphen option specified by the user in the FFI. Refer to the Index Definition for Keyword Fields section of Volume II, File Structuring, for these options.

If the nature of the data is such that it does not lend itself to the system's scan routine, the user may write and designated his own. For a detailed description on the requirements for a user's scan routine, see the User Scan Subroutine (section 3.3.4).

If a Stop Word Table had been specified, each word is passed against it searching for a match. When a match is found, the word is eliminated from any further processing. The remaining words (or all words from the field when no Stop Word Table is specified) become keyword candidates.

After all transaction records have been processed, the index data set is updated. When a dictionary is not specified for an indexed field, all keyword candidates become entries in the index data set. When a dictionary is specified, each non-literal word is passed against it to determine if the value qualifies as a keyword. An entry is made in the index data set for each qualifying value; the remainder are dropped from any further processing. Literal words always qualify without being matched to the dictionary.

To perform record qualification based on keywords, the user must include a KEYWORD statement in his query. Structually and functionally, the statement is similar to an IF statement. The same rules of logic apply. The statement is on the same level of hierarchy as the LIMIT statement in QUIP and the secondary LIMIT statement in FASP. Only one KEYWORD statement per query is allowed. The format is as follows:

KEYWORD fieldname INCLUDES keyword argument(s) AND...clause.

clause OR

Each field name must have been defined as a keyword indexed field or the retrieval will be terminated. The

keyword argument is passed against the dictionary (if one were specified) and against the keyword entries for that field in the index data set to determine which records qualify.

This list of record IDs is passed to the retrieval routine to indicate what records should be accessed. The keyword fields will not be scanned again as they have already qualified. Retrieval is on the record level only. When any of the queried fields have been defined as secondary indexed fields, such that a second qualifying list is present, the two lists will be merged into one final list containing only those records that qualify for both the keyword and secondary indexed fields.

Keyword indexing is designed to provide retrieval based on the occurrence of 'keywords' within a field. To achieve optimum utilization, the user responsible for defining the indexed fields must have a good knowledge of the text data in the file and the terms that will be used as a basis for retrieval. This is necessary in order to define the best method of selecting keywords from a designated field (via stop word table, dictionary) and to avoid overloading the index data set with fields and entries that are unlikely candidates for retrieval.

The Keyword Analysis Utility (see Volume VII, Utility Support) is provided to assist the user in making this selection for a file already in existance. The contents of any fields in the file may be listed to determine the likely candidates for indexing and what values are likely to be keywords or entries in a stop word table and dictionary. This will also assist in determining whether a stop word table and/or a dictionary should be applied against the field.

3.8 Interfile Output (IFO)

Interfile Output (IFO) is the capability to combine data from several related files during the output processing. It is available in the batch environment in the OP and QUIP components and in the online environment in QUIP.

INTRODUCTION TO FILE CONCEPTS

A master or primary file contains data elements which are maintained by the user and which identify records in other referenced or secondary files. These data elements are called pointers and consist of the record key (or minimally, the high-order portion thereof) for the secondary files and may also include a secondary file identifier. In some cases, the record key of the secondary file may be identical to the record key of the primary file so as to allow use of the primary file key as the pointer element that can be used as a pointer to a secondary file. Should the interfile relationships be complex, the pointer may additionally require an indicator to identify the secondary file to be referenced. This is accomplished by structuring the pointer as a group. The first field contains the name of the secondary file or a code which is converted to the secondary file name by a subroutine or table specified in the FFT; and the second field contains the record keys, or high-order portion, of the referenced records. To allow the components to recognize this group as the special type of pointer containing the file identifier and record key, it has a standard system name consisting of the six characters IFO3PP with a suffix which is a unique (for the file) letter or digit. Thus, a maximum of 36 such pointers is possible for any file. Finally, defined literals and sortkey values (when the primary file is a RASP answer set or retrieval-type IF/SORT statements are used in QUIP query) may also be used as pointers to secondary files.

IFD causes processing of the primary file record to be interrupted and a retrieval made from a secondary file of one or more data records. Those records are then processed against the logical specifications coded for the secondary file. After processing has been completed for those retrieved secondary file records, primary file processing is resumed.

The selection of any secondary file records depends on the pointer and its contents at the time the primary file processing is interrupted. If the length of the field used as the pointer is equal to the length of the record key of the secondary file, at most one record is retrieved from the secondary file with its record key matching the pointer. If the field used as the pointer has length greater than the

length of the record key of the secondary file, the field is truncated before retrieval and at most one record is retrieved with its record key matching the truncated field. If the length of the pointer field is less than the length of the secondary file record key, the retrieval of secondary file records consists of all these records where the high-order portion of the record key matches the pointer, the length of the high-order portion being determined by the length of the pointer. When more than one record is retrieved in this manner, processing of the secondary file specifications is repeated for each record.

3.9 File Analysis and Run Optimization Statistics

The File Analysis and Run Optimization Statistics capability provides the user with statistical data concerning the activity of fields in the data file and statistical data concerning the allocation of core resources for NIPS 360 PFS production runs.

3.9.1 File Analysis Statistics

The File Analysis Statistics involve both a stand-alone utility to count data file field references of all of a users component source statements and component execution statistics showing the number of times a source statement was executed. Using this statistical data a user may more efficiently determine the activity of the data file fields.

Statistics for file analysis are gathered by the user from two sources:

- A NIPS utility, UTFLDSCN, which determines which file fields are referenced in each source RIT, retrieval, and logic statement
- A use count of each RIT, retrieval, and logic statement provided by each NIPS component.

These two types of statistics may be combined in a NIPS file for analysis.

The stand-alone utility, UTFLDSCN, will accept as input all of the component source statements for a single data file. The format of the input may be cards and/or a or member of a partitioned data set in card image record format. Primary input, which consists of control cards will be a card. Each set of source statements must be preceded by a control card to identify the component, module name, and member name. The format of the card will be as follows:

```
./   SOURCE     COMP=XXXX,NAME=SNAME,MEMBER=MNAME
```

where:

./ must be in columns 1 and 2 followed by one or more blanks.

SOURCE must appear as shown.

COMP=XXXX where XXXX may be FM, QUIP, RASP or OP.

NAME=SNAME where SNAME is the name of the source module.

MEMBER=MNAME where MNAME indicates that the source is a member of a partitioned data set and that this is the member name. This operand must be omitted if the source is in card format. Mixed formats will be allowed.

The output from the utility will consist of a listing of the source statement, followed by a listing of the file fields with a count of references. After all source statements have been processed, a summary listing will be provided showing the count of data file field references by component. A sequential data set will also be provided suitable for transaction input to an FM run.

The utility will process fields in the source statement from a single file in one execution. Multi-file RASP queries and multi-file RITs will be accepted as input. However, only the fields from the input data file will be processed. To process all fields from all files referenced by multi-file queries and multi-file RITs, it would be necessary to include the source statement in a utility run for each file.

The batch component file analysis statistics will be initiated dynamically by the presence of an entry in the catalog for a transaction data set. The data set name must be the file name concatenated with a 'T' suffix and must be cataloged to the same volume as the data file or a DD statement named TRANST included as an additional DD statement. If the statistics are initiated through the catalog entry and the data set does not exist, the space for the data set will be allocated on the same volume as the data file. If the additional DD is included, it must fully describe the data set according to JCL requirements. The disposition for the transaction data set will be MOD so that new transactions will be added and will not destroy any transactions gathered previously.

A parameter PARM=NOSTAT, may be entered on the EXEC card to override the gathering of the statistics.

Samples of the transactions output by the file analysis statistics, with a sample FFT, logic statements, query, and RIT utilizing the transaction are included in appendix B.

3.9.2 Run Optimization Statistics

This capability produces, on user's request, statistical data reflecting the core allocation during maximum core utilization for the execution of the component. The breakdown of the statistics detail the amount of core used for user subroutines and tables, the source module (RIT, query, logic statement), processing block or stash area, I/O buffers, and OS access method subroutines. The information allows the user on subsequent runs to specify options to make more efficient use of his core allocation by specifying through the PARM field on the EXEC card the core required for elements such as process block, stash area, and the BLDL list. The amount of core allocated to subroutines and tables is indirectly specified by adjusting the region size. The core allocation for I/O access method routines remains constant for a given run and may not be overridden.

This capability is designed primarily for use with production runs where the job setup remains constant.

The batch components FM, RASP, OP, and QUIP recognize the Run Optimization parameters in the PARM field of the execute card. The presence or absence of parameters are interpreted by the initialization routines to determine processing required by the execution phases of each component. At completion of the execution of the component, the reporting phase outputs all gathered statistical data as a printed listing.

Each component prints a statistics page showing core allocation and dynamic resource loading. The core allocation portion indicates the processing block or stash area allocated and used, the amount of core used for the source modules and user subroutines and tables, and the amount of core used for I/O buffers and access methods. The dynamic resource portion of the statistics includes the number of times each subroutine, table, or logic statement was executed and how many times it was rolled out of core. The reason for rolling the module, either insufficient core or insufficient BLDL list will also be indicated.

The user may alter the scheme of core allocation through use of the PARM field on the EXEC card. For example, if all of the stash area in RASP is not used but subroutines or tables referenced on the SORT statement are rolled due to insufficient core, the stash area may be decreased. The excess will automatically be assigned to the dynamic subroutine area. The BLDL list is a list maintained by SUBSUP, the NIPS Subroutine Supervisor, containing the name, size, and disk address of each subroutine, table, or logic statement used. Each entry in the list requires about 70 bytes of core. Ideally, there should be an entry in the list for each subroutine, table, table page, and logic statement executed in the run, and there should be sufficient core so that no modules have to be rolled out. This is rarely possible in batch production jobs. The user can alter the length of the list to use core most efficiently.

The Run Optimization capability will be initiated through the use of parameters in the PARM field on the EXEC card for each component. The parameters and their associated functions are as follows:

ROS - This parameter specifies that run optimization information is to be gathered and output.

NOROS - This parameter omits optimization processing. If no other parameters are used, this parameter need not be coded, as it is the default.

The parameters to be used to supply parameters to tailor core allocation are as follows:

TCP=NK - Indicates the number of bytes requested for processing block or stash area in 1000 (K) bytes.

TCB=N - This parameter indicates the number of entries to be used in the BLDL list for SUBSUP.

TCS - This parameter indicates that the statistics record on the ISAM data file is to be used to determine the process block size. This parameter should not be used with the TCP parameter.

When core allocation parameters are coded, ROS will be performed unless the NOROS is coded in the PARM field on the execute card.

The definitions for the statistics output by the Run Optimization are as follows:

Core Usage

Region size - The size in decimal of the area requested on the JOB or EXEC card.

Free core - The amount of core not used. This figure will not include fragments of core of less than one K.

Access methods - Includes the area allocated to the access methods.

Buffers - Amount of core used for buffers on open data sets.

Component - Amount of core taken up by the batch component and any programs loaded by the component, excluding logic statements, PASP queries, and RITS.

Note: Core sizes are given in bytes (1024 bytes = 1K).

Process_Block/Stash_Area_Statistics

Process block size - This is the amount of core allocated to the process block or stash area.

PM - Amount for process block

PASP - Amount allocated to stash area.

OP - Minimum amount to contain one data record and associated sets.

QUIP - Amount allocated to process block.

Amount used - Amount of process block or stash area used for storage of data records.

Subroutine/Table_Execution_Statistics

The number of entries for BLDL lists allocated is output with the number of entries actually used. If subroutines were rolled because of insufficient BLDL entries, the number necessary to prevent rolling will also be output. The user may then on the next run specify the number required to prevent rolling for this reason.

The amount of core used for subroutines and tables will be listed whether or not they were added. If subroutines or tables were rolled for lack of core, the amount of core listed is that amount required to prevent rolling. The user may then reduce processing block or stash area size to provide more core for subroutines and tables, if feasible,

or increase his region size to prevent rolling for this reason.

Roll information will detail the modules that were rolled by SUBSUP, the number of times rolled, and the reasons for rolling.

If the run is an FM component execution, the subroutine and table statistics will also include logic statements.

3.10 Improved NIPS File Processing

As NIPS is used in an ever-widening range of applications, data file usage is becoming more varied and data file size is becoming larger. The following three capabilities provide enhanced file processing techniques.

3.10.1 Qualified Data Set Names

Qualified data set names are allowed for NIPS data files, libraries and index data sets. The basic name must conform to the NIPS naming conventions (section 2.6.3). To this basic name, additional qualifiers may be prefixed up to a total of 44 characters. Symbolic parameters and data definition (DD) statement overrides must specify the fully qualified data set name. In the batch mode, only the basic name (last segment of the qualified data set name) is specified in the NIPS component control cards. When comparison is required, the component addresses only the last segment of the qualified data set name on the appropriate DD statement. In online applications (QUIP, FM, SODA, EDIT, VIEW and FORMATTER), NIPS components utilize the catalogue to locate data sets, and the fully qualified data set name must be specified.

3.10.1.1 Specifying a User Library

A NIPS user can specify a library by use of selected data definition statements in the NIPS cataloged procedures

which include the SLIB, TLIB and DLIB DDNAMES. The library name may be a qualified or an unqualified data set name.

In those cases where the DSNAME of the library is determined by suffixing the data file name with an "L", the fully qualified name will be used. If this name cannot be located in the catalog, the last segment of the qualified data set name will be used with an "L" suffix. If a qualified name is not used, then the file name will be the only segment.

3.10.1.2 Specifying an Index Data Set

A NIPS user can specify an Index Data Set by use of the XINDEX DD card. The DSNAME of the Index Data Set, less the suffix "X", must be the same as the fully qualified data file name. In those cases where the DSNAME of the Index Data Set is determined by suffixing the ISAM or SAM data base name with an "X", the file name used will be the fully qualified data set name. If a qualified name is not used, then the file name will be the only segment.

3.10.1.3 Data Generation Group

The DSNAME parameter of the DD cards in the JCL stream for the user and system libaries (SLIB), sequential data files (SAMFILE), and Index Data Set (XINDEX) may be a generation data set. A generation data set is one of a collection of successive, historically related, cataloged data sets known as a generation data group. To create or retrieve a generation data set, you identify the generation data group name in the DSNAME parameter and follow the group name with a relative or absolute generation number.

A generation data group can consist of cataloged sequential, partitioned, indexed sequential (if the data set is defined on one DD statement), and direct data sets residing on tape volumes or direct access volumes. A NIPS ISAM data file would normally not be part of a generation data group since the data set is defined using three DD statements.

3.10.2 Cataloged Procedures and File Block Size Considerations

All NIPS 360/FPS cataloged procedures, components and utility programs are designed to process files with a block size of 1,004 bytes or greater. The standard file block size is 1,004, but the block size of any file can be set by the user to a size of 1,004 or greater as long as the specified size does not conflict with S/360/OS Data Management rules or storage device limitations.

If a user elects to change the block size of a file, he can do so with any cataloged procedure that generates a file or produces a new copy. These procedures are easily recognized because they include the symbolic parameter BSZNEWP, which allows the user to specify the block size of the new file.

Once a file's block size is changed, the file will retain that block size until changed again by the user.

Block size specifications or the use of symbolic parameters, BSZFILE or BSZNEWP are always required for the following conditions:

1. BSZNEWP must be used to change a file's block size.
2. BSZNEWP must be used to generate a file with a block size greater than 1,004.
3. BSZFILE must be used when processing a non-standard block size file residing on unlabeled tape.

The effects of increasing a file's block size are:

1. Less storage space required because of fewer inter-record gaps
2. Less I/O time required during processing
3. More core required during processing.

The BSZNEWF parameter cannot be used to modify the block size of an output VSAM file. The block size of the VSAM file must be set when the file is defined by the VSAM service routine IDCAMS as set forth in Volume VIII - Job Preparation. When going from VSAM to SAM the block size of the SAM file will be 1004 unless overridden using the BSZNEWF parameter.

3.10.3 Compression and Compaction of Data Records

Compression and compaction provide a means for the reduction of intermediate storage requirements for data without altering the integrity of the data. This data reduction scheme is particularly suited to data files that contain strings of identical characters or a large quantity of alphabetic data.

A string of identical characters is compressed by translating it to two bytes. The first byte is a control byte which indicates that compression has been applied and gives a count of the number of identical consecutive bytes that were in the original string. The second byte is identical to those in the original string.

A string of alphabetic characters is compacted by translating it to a control byte followed by a string of coded characters. The control byte indicates that compaction has been applied and gives a count of the coded characters. Each coded character represents a combination of two adjacent alphabetic characters.

Compression or compaction can be applied to data files by specifying COMPRESS or COMPACT respectively for the PARM on the EXEC card of the SAM to ISAM/VSAM and ISAM/VSAM to SAM utilities. A combination of both can be specified by including both keywords in the PARM list. If both are specified, compression is applied to a record first and data within the record that cannot be compressed is operated upon by the compaction routine.

The compression and/or compaction process can be reversed by specifying EXPAND for the PARM on the EXEC card

for either utility. All components process compressed and/or compacted files transparently to the user.

3.11 Subfile Capability

The subfile capability allows the user to define increasingly discrete queries so that each new query processes a decreasing number of data records of the file. The capability is available in the QUIP component in the online environment.

The subfile capability is accomplished by allowing the user to create a subfile consisting of selected entries from the file, the entries being selected based on the conditional expressions in the query. Subsequent queries are automatically directed against the subfile until the user creates a lower level subfile or explicity specifies a different source of input.

Subfile processing is initiated by the user when he includes the SUBFILE operator in a QUIP query in the online environment. The first use of the operator causes the naming and allocation of a partitioned data set on a direct access system work volume. Each subfile request results in creation of a new member of the partitioned data set.

The name of the subfile partitioned data set is internally generated by forming a qualified data set name of two levels: the first is the terminal identification as defined at NIPS TP Monitor system generation and the second is the internal computer time of the query at the time the data set is allocated. This name is displayed at the terminal when it is allocated and may be used later for identification to access the subfiles should a system failure occur which causes a checkpoint restart of NIPS TP to occur.

The subfile partitioned data set is deleted when the user signs off from QUIP. However, the existing subfiles for a partition file are made unavailable when a query is directed against another NIPS PFS data file.

A subfile consists of the record keys of the qualifying records for the query, rather than the entire record, thereby reducing I/O time and space requirements for processing subfiles. When a query is directed against a subfile, the record keys in that subfile are examined to determine which records in the master file are to be accessed. If a query directed against a subfile contains a retrieval which operates in the candidate-access mode, the candidate list built by Index Processing is used to further restrict the access of master file records to those subfile entries which are also candidates from Index Processing.

The subfile capability allows the user to create a subfile and an output display simulataneously so that the data for entries in the subfile may be examined while the subfile is being created.

There are no restrictions on the number of subfile levels the user is able to create. Any existing subfile for the current master file may be referenced as an input source so that interrogations may be performed from that level.

Subfiles generated at one terminal are available for interrogation at other terminals in the system. The user is required to refer to the subfile name in his query for access to the subfile.

The subfile capacility includes a trace function which permits a user to display part or all of the contents of the subfile partitioned data set. This function allows the user to review all subfiles created from the same (current) data file.

3.12 Non-NIPS Query Capability

NIPS can also be used to retrieve and output data from data files that were created either by other data management systems or by special purpose programs. This capability is available through QUIP in both the batch and online environments.

To perform non-NIPS processing, the user must first create a pseudo-FPT which describes the data file. Creation of the pseudo-FPT is by FS through the standard FS language plus one non-NIPS control statement and several non-NIPS operands. After the pseudo-FPT has been structured and stored as an ISAM data set, QUIP can be used to query the non-NIPS data file. All of the standard QUIP processing techniques can be applied to a non-NIPS file with the exception of index processing. Care must also be taken when attempting to process numeric or coordinate data fields as there is no NIPS check for valid contents of these fields.

Files which are to be processed via the non-NIPS query capability must conform to the following general restrictions:

1. The file can be either a SAM or an ISAM data set. Records in the file can be either fixed length (up to 996 bytes) or variable length (up to 1000 bytes), and either blocked or unblocked.

2. Each record must contain record identification data which is contained in one or more user designated fields. These fields need not be contiguous; but must be present in each record. The location of these fields may differ for differing record formats but the size of the fields and their relative position within the total record ID must remain constant. The total length of the ID must not exceed 256 bytes.

3. All records with a common record ID must be physically together in the file and the file must be in ascending sort sequence as specified by the record ID.

4. Each record must contain a single data element of up to 10 bytes in length which can serve as a record type field to uniquely identify each record format. The location and length of the record type field must remain the same for all records. A maximum of 256 different record types may be identified.

5. One record type must be identified as a non-repeating format to serve as a fixed set. This format must occur once (and only once) for each group of records having a common record ID. All other record types are considered to be repeating, and their presence is optional.

6. Numeric data for which arithmetic computations are to be performed must be defined in either binary or zoned decimal (EBCDIC) format. All other numeric data fields (non-fullword binary, packed decimal, and floating point) must be treated as alphabetic data and a user-written subroutine must be supplied for output conversion. Binary format requires that the data be contained in a fullword on a fullword boundary.

7. Coordinate data must be stored in a fullword for each longitude and latitude and must conform to the standard NIPS binary representation.

8. Variable length data fields are permitted provided only one variable field is defined for a record type and it is the last field in the format. Also, the format must contain a fullword binary field which will contain the length of the variable field.

SECTION 4

SAMPLE NIPS 360 FFS DATA FILE

This section introduces a sample data file which is typical for the files handled by the system. It is presented here since the Users Manuals for all components will use examples pertaining to this file.

4.1 General File Organization

The name of the sample file is TEST360. Its structure is defined to contain information concerning the status, organization, location, and equipment of combat units of the armed forces. Each data record in the file defines a single unit in the armed forces. Hence, the key to each record will be the unit's identification code. Data in each record has been formatted into a fixed set, six periodic sets, and a variable set. Data conversion subroutines and tables have been defined to process some of the record's data.

The logical breakdown of data in a record is discussed below.

FIXED SET - The fixed set contains data defining the attributes of the unit which need only one data value for satisfaction. Examples of this are the unit's location, status, activity, and commander's name.

Periodic Sets - The six periodic sets are used to contain information defining the unit whose record elements may have more than one data value. For a periodic set, each collection of data having the same format is called a subset of the periodic set.

PERIODIC SET 1 - Each subset contains data describing a piece of major equipment or a weapon type possessed by the unit.

PERIODIC SET 2 - Each subset contains data describing a piece of secondary equipment or non-essential material not required for the unit's operation.

PERIODIC SET 3 - Each subset contains data describing an operation plan which the unit must follow.

PERIODIC SET 4 - Each subset contains the name of a treaty to which the unit is responsible.

PERIODIC SET 5 - Each subset contains information on a senior or staff officer of the unit.

PERIODIC SET 6 - Each subset lists a subordinate unit reporting to the unit.

VARIABLE SET - The variable set in each record contains commentary information about the unit.

4.2 Record Element Description

This section describes each element in the file's record format. The source language statements used to define the format of this file appear in the File Structuring volume of the NIPS 360 FFS Users Manual.

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
SERV	Record Control Field	Fixed	1	ALPHA	RCMDS	OCMDS	Service Branch Code
UUIN	Record Control Field	Fixed	5	ALPHA			Unit Identifier (Service)
UIC	Record Control Group	Fixed	6	ALPHA			Unit Identification Code (Fields - SERV, UUIN)
UNTYY	Field	Fixed	4	ALPHA			Military Unit Type Code
UNTYZ	Field	Fixed	1	ALPHA			Major Unit Indicator
UNLVL	Field	Fixed	3	ALPHA		UNLVS	Unit Organization Level
UNTLY	Group	Fixed	8	ALPHA			Unit Type and level (Fields-UNTYY, UNTYZ, UNLVL)
HOME	Field	Fixed	1	ALPHA	RCMDS	OCMDS	Current Home Command
UNFLG	Field	Fixed	1	ALPHA			Unit Flag - Reserved for Special Use
MJFOR	Field	Fixed	1	ALPHA			Major Force Indicator
PREV	Field	Fixed	1	ALPHA	RCMDS	OCMDS	Previous Home Command

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
ATACH	Field	Fixed	1	ALPHA	RCMDS	OCMDS	Attached Command Reporting Units Status
FUTU	Field	Fixed	1	ALPHA	RCMDS	OCMDS	Future Home Command
TRDTG	Field	Fixed	10	ALPHA	DTGIS	DTGOS	Transfer Date to New Command
UNRDY	Field	Fixed	2	ALPHA			Readiness Status
REASN	Field	Fixed	1	ALPHA			Readiness Downgrade Reason
RATTN	Field	Fixed	2	ALPHA			Readiness Expected to Attain
RECDE	Group	Fixed	5	ALPHA			Unit Readiness Status (Fields-UNRDY, REASN, RATTN)
RADTG	Field	Fixed	10	ALPHA	DTGIS	DTGOS	Attainable Readiness Status Date and Time
UNIT	Field	Fixed	12	ALPHA			Short Unit Name
UNAME	Field	Fixed	27	ALPHA			Full Unit Name
OPCON	Field	Fixed	6	ALPHA			UIC of Higher Unit Having Operational Control
CONOR	Field	Fixed	20	ALPHA			C.O. Name and Rank
LOC	Field	Fixed	18	ALPHA			Location or Hull Number of Unit
POINT	Field	Fixed	11	COORD			Geographic Location (Lat-Long) of Units Headquarters

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
DAPT1	Field	Fixed	11	COORD			Geographic Points (Lat-Long) Defined in Counterclockwise Order Which Defines the Unit's Area of Deployment or responsibility
DAPT2	Field	Fixed	11	COORD			
DAPT3	Field	Fixed	11	COORD			
DAPT4	Field	Fixed	11	COORD			
AREA	Group	Fixed	44	COORD			Coordinate Area (Fields-DAPT1, DAPT2, DAPT3, DAPT4)
CNTRY	Field	Fixed	2	ALPHA		CTRYS	Country Code Where Unit is Located
CNAM	Field	Fixed	15	ALPHA			Country Name Where Unit is Located
GEPOL	Field	Fixed	2	ALPHA		CTRYS	Geopolitical Code Where Unit is Located
PERS	Field	Fixed	6	NUMER			Authorized Personnel Strength
ACTIV	Field	Fixed	2	ALPHA		ACTVS	Current Activity Code
LAUD	Field	Fixed	10	ALPHA			Date-Time of Last Record Update

Element Name	Element Type	Set Nz	Length	Mode	Input Conv.	Output Conv.	Remarks
LYB	Field	Fixed	1	ALPHA			Location Status Whether Known, Unknown, or Embarked
RPERS	Field	Fixed	1	NUMER			Personnel Readiness Code
REQPT	Field	Fixed	1	NUMER			Equipment Readiness Code
RTRNG	Field	Fixed	1	NUMER			Training Readiness Code
RMGRP	Group	Fixed	4	NUMER			Readiness Group (Fields-RPERS, RSPLY, REQPT, RTRNG)
READAVG	Field	Fixed	3	NUMER			Readiness Average to Hundredths
DITNM	Field	Fixed	3	NUMER			[illegible] of Maximum D[illegible]nce from Command Ship - to Tenth. Naut. Miles
UNTYP	Field	Fixed	5	ALPHA			Unit Type Code
TPNAM	Field	Fixed	42	ALPHA			Unit Type Name
UNTOE	Field	Fixed	17	ALPHA			T/O and E Reference
HIER	Field	Fixed	11	ALPHA			Unit Hierarchy Code

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
COMMENT	Field	Fixed	Variable				Variable Length Field to Hold Comments
MECL	Field	1	3	ALPHA			Major Equipment Class
MEQPT	Field	1	10	ALPHA			Major Equipment ID
MECLQ	Subset Control Group	1	13	ALPHA			Major Equipment Class and Type (Fields - MECL, MEQPT)
MEMOD	Field	1	10	ALPHA			Major Equipment Model Number
MENAM	Field	1	18	ALPHA			Major Equipment Name
MECAP	Field	1	1	ALPHA			Weapon Delivery Capability Code
MEPSD	Field	1	3	NUMBER			Number of Equipments Possessed
MEADA	Field	1	3	NUMBER			Number of Equipments on Alert
MEDRC	Field	1	3	NUMBER			Number of Equipments Ready for Conventional Weapon Delivery
MEDRN	Field	1	3	NUMBER			Number of Equipments Ready for Nuclear Weapon Delivery

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
MESQP	Field	1	3	NUMER			Number of Equipments on Special Alert
MESWP	Field	1	3	NUMER			Number of Equipments on Special Alert with Nuclear Capability
MESIA	Group	1	6	NUMER			Special Alert Group (Fields - MESQP, MESWP)
MESIC	Field	1	3	NUMER			Number of Equipments Committed for Special Alerts
MEREC	Field	1	10	ALPHA			Equipment Reconnaissance Capability
MEDEP	Field	1	1	ALPHA			Code indicating if Equipment is at Home Location or TDY
MEDDT	Field	1	5	NUMER			Date Equipment went on TDY Status (Julian Date)
MEDUR	Field	1	1	ALPHA			TDY Duration Code
MELYR	Field	1	1	ALPHA			TDY Deployment Status

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
MELOC	Field	1	18	ALPHA			TDY Equipment Location
MEPNT	Field	1	11	COORD			Geographic Location (Lat-Long) of TDY Equipment
METRY	Field	1	2	ALPHA		CTRYS	Country Code where TDY Equipment is Located
MEPOL	Field	1	2	ALPHA		CTRYS	Geopolitical Area Code where TDY Equipment is Located
MECNA	Field	1	15	ALPHA			Country Name for TDY Location
SECLASS	Field	2	3	ALPHA			Secondary Equipment Classification
SEMODEL	Field	2	10	ALPHA			Secondary Equipment Model Number
SENAME	Field	2	18	ALPHA			Secondary Equipment Popular Name
SEPOSSD	Field	2	4	NUMER			Number of Equipments Possessed
SEAUTH	Field	2	4	NUMER			Number of Equipments Authorized
PLAN	Field	3	4	NUMER			Plan Identification Number

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
PLBAC	Field	3	1	ALPHA			Plan Status Code for Unit
PLDTG	Field	3	10	ALPHA	DRGIS	DTGOS	Date-Time Unit Adhered to Plan
PLFST	Field	3	1	ALPHA			Expected Plan Status Code
PLFDG	Field	3	10	ALPHA	DTGIS	DTGOS	Expect Date-Time Unit will be Committed to Plan
PLRT	Field	3	6	ALPHA			Plan Response Time
PLTRT	Field	3	6	ALPHA			Transportation Staging Time
TRTY	Field	4	6	ALPHA			Treaty Code of Unit Affiliation
NAME	Field	5	18	ALPHA			Senior Officer/PO Name
RANK	Field	5	4	ALPHA			Senior Officer/PO Rank
SERNUMB	Field	5	6	ALPHA			Serial Number
SERVICE	Field	5	1	ALPHA			Service Branch Code
ASSGN	Field	5	20	ALPHA			Unit Assignment
SPCODE	Field	5	5	ALPHA			Specialty Code
SBUIC	Field	6	6	ALPHA			Subordinate Unit UIC

Element Name	Element Type	Set No.	Length	Mode	Input Conv.	Output Conv.	Remarks
SBFLG	Field	6	6	ALPHA			Reason for Subordinate UIC
REMER	Variable Set						Unit Remarks/ Comments in Unformatted Form

4.3 Subroutine/Table Description

This subsection describes the conversion subroutines and tables used by the sample file.

4.3.1 Table - RCMDS

The table RCMDS is used for input data conversion. It will accept up to a 6-character argument and produce a single character code as a function. The table is used for converting names of unified/specified commands to single-character codes. A sample of the table contents follows:

ARGUMENT	FUNCTION
USCG	E
USAG	J
USMC	M
JCS	U
.	.
.	.
.	.
NORAD	3
SAC	8

4.3.2 Table - OCMDS

The table OCMDS is used for output conversion. It accepts a single-character code representing a unified/specified command and expands it to a name of up to six

characters. The table is used with the input conversion table, RCMDS. A sample of the table contents follows:

ARGUMENT	FUNCTION
M	MARINE
N	NAVY
R	RCAF
.	.
.	.
.	.
4	ANGAC
2	LANT
4	EUCOM
7	STRIKE

4.3.3 Table - CTRYS

The table CTRYS is used for output conversion. It accepts as an argument a 2-character code and expands to a country or geopolitical area name which may be up to 15 characters in length. A sample of the table contents follows:

ARGUMENT	FUNCTION
AC	ATLANTIC OCEAN
AL	ALBANIA
AT	AUSTRALIA
BD	BERMUDA ISLANDS
CB	CAMBODIA
EG	EGYPT
GU	GUAM
.	.
.	.
.	.
TH	THAILAND
19	LOUISIANA
37	OKLAHOMA
47	VIRGINIA
65	PACIFIC ISLANDS

4.3.4 Table - ACTVS

The table ACTVS is used for output conversion. It accepts a 2-character code and expands it to state a certain military activity of up to 15 characters. A sample of the table contents follows:

ARGUMENT	FUNCTION
AC	ACTIVATING
CD	CIVIL DISTURB
CO	COMBAT
DE	DEACTIVATING
EX	EXER/MANEUVER
MA	MAINTENANCE
.	.
.	.
.	.
SP	SHOW OF FORCE
SR	SEARCH/RESCUE
TR	TRAINING

4.3.5 Table - UNLVS

The table UNLVS is used for output conversion. It accepts up to a 3-character code and expands it to state a unit's level using up to 15 characters. A sample of the table contents follows:

ARGUMENT	FUNCTION
ACD	ACADEMY
ANX	ANNEX
CO	COMPANY
DAY	DIV ARTILLERY
FLT	NUMBERED FLEET
HQ	HEADQUARTERS
HSP	HOSPITAL
MER	MERCHANT SHIP
PLT	PLATOON
RCT	RGT COMBAT TEAM
.	.

.	.
.	.
SYD	SHIP YARD
TF	TASK FORCE
USS	US SHIP

4.3.6 Subroutine - DTGIS

The subroutine DTGIS is used for input data conversion. It accepts a 12-character data item which is a Date-Time group and converts it to a 10-character form suitable for sorting dates in sequence.

The input format to the subroutine is:

DDTTTTSMMMYY

where

DD	=	Day of Month
TTTT	=	2400 Hour Time
S	=	Flag Indicating Greenwich Time
MMM	=	Month (Jan, Feb, --- Dec)
YY	=	Year (65, 66 ---).

The output format from the subroutine is:

YYMMDDTTTT

where

YY	=	Year (65, 66 ---)
MM	=	Month Code (Jan=01, Feb=02)
DD	=	Day of Month
TTTT	=	Greenwich Time.

4.3.7 Subroutine - DTGOS

The subroutine DTGOS is used for output conversion. It accepts as input the 10-character Date-Time Group produced by DTGIS and converts it to the 12-character source format.

4.3.8 Table - KEYSTOP

The table KEYSTOP is used as a STOPWORD table in connection with KEYWORD processing of the index file for TEST360 file. STOPWORD tables are generated and updated by the UTNDXKMD utility and contain a list of words that are to be eliminated from keyword processing. A sample of the table contents follows:

```
A
AND
AS
BUT
FOR
THE
THERE
THIS
.
.
.
```

4.3.9 Table - ACCEPT

The table ACCEPT is used as a DICTIONARY table in connection with Keyword processing of the index file for TEST360. DICTIONARY tables are generated and updated by the UTNDYKMD utility and contain a list of all allowable keywords that may appear in the index file. A sample of the table contents follows:

```
BATTALION
DIVISION
MANUEVERS
STRATEGIC
TRAINING
TRANSPORT
WEAPONS
.
.
.
```

Section 5

GLOSSARY

This section contains a list of terms commonly used with the NIPS 360 FFS. A brief description is supplied. Most of the terms the user may come across which are related to S/360 hardware and standard software are not repeated here since they are adequately discussed in the IBM SRL publications.

Block	1. A physical record (separated from other records by inter-record gaps) which contains multiple, logical data records. Refer to blocking of records. 2. A group of computer words considered as a unit by virtue of their being stored in successive storage locations.
Block Count (Field)	A field which is the first four characters of each block of file records, containing the number of characters in the block. Do not confuse with record character count.
Blocking of Records	The combining of multiple logical records into one block of information on tape to decrease the time wasted due to acceleration and deceleration of tape and to conserve space on tape.
Circle Search	A special geographic retrieval operator which permits selection of file records by determining if a point carried in the file record falls within a circle specified as the search criteria.
Component	A major functional unit within NIPS 360 FFS.

Control Field	Refer to record control field.
Control Group	Refer to record control group or record ID.
Data Base	The collection of data files (data sets) used under the system.
Data File	Also called FFS data file or formatted file or file. A collection of data records, called file records, which can be logically grouped on the basis of subject matter. Since the organization of the data is formatted, the file is called a formatted file.
Data Set	NIPS 360 term essentially implying a data file. Used to describe a collection of data records, stored in common, and accessed as an entity.
Data Record	As a general term, means a group of related fields of data treated as a unit. Often used to mean FFS file record (refer to file record).
Dictionary	A user-defined table that consists of all words (including any synonyms) that are to qualify as keywords from the associated indexed field.
FFS	Formatted File System.
FFT	File Format Table.
Field	The smallest defined logical unit of data in a record handled by the FFS consisting of one or more adjacent characters.
Field Name	The synonym or mnemonic assigned to represent a discrete area (field or group) in the data record.

File	Generally a nonspecific term meaning an organized collection of information directed toward some purpose. However, in this documentation, file means FFS data file, unless otherwise qualified. (Refer to data file.)
File Format Table	A collection of records which completely describes the format of the FFS data file. They are generated by the File Structuring Component. There is one FFT for each data file.
File ID	Name of the FFS data file.
File Mnemonic	Same as file ID.
File Record	(Also called data record.) A group of related fields of data. The file record is formatted - that is, each element of the file record has been defined, identified, and assigned a relative position. Each file record has a fixed set which contains the record ID. The file record may also contain a number of periodic sets and/or variable sets.
FIT	File Information Table.
Fixed Field	A field defined in the fixed set of a file record and which must appear once and only once in the file record.
Fixed Group	Refer to group.
Fixed Set	That portion of a file record consisting of all the fixed fields/groups of the file record.
FM	System component -- File Maintenance.
Format	A predetermined arrangement of characters, fields, or other data. A format does not

	describe the data, but describes its organization.
Formatted File	Refer to data file.
FR	System component -- File Revision.
FS	System component -- File Structuring.
Group	A collection of one or more adjacent fields of the same type which are related. A group is capable of being processed or otherwise manipulated as a unit. The system may treat a group the same as a field. The fields within a group in no way lose their individual identities and may be treated as if they were not grouped. If fixed fields are grouped, the group is a fixed group. A periodic group is a grouping of periodic fields.
High-Order Position	The leftmost (most significant) position of a field.
HOP	High-Order Position.
Index	A file field or group that has been specified as part of the File Indexing capability. The file is cross-indexed, by Record ID, on the contents of an index field. Fixed fields are defined as secondary indexes, variable fields, variable sets and fixed-length alpha fields containing textual data are defined as Keyword Indexes.
Index Data Set	The repository of all information required to support the File Indexing capability. It contains all secondary and keyword indexed fields, all index fields values (or keywords) and their associated Record IDs.

Input Descriptor	A deck of cards which describes the external format of input data for the FM component.
Input File	A card or tape file which contains all or a portion of the data needed by FM to update a NIPS data file (also known as a transaction file).
Input Group	All of those input records containing information to be extracted for the purposes of creating or updating a single (the same) file record.
Input Group Control Field	An artificial control field or an actual data field (or fields) by which the input file is sorted or manually arranged prior to input to the system. This is done so that all input records belonging to the same input group (i.e., pertaining to the same file record) will be grouped together.
Input Record	A single card (or tape record) in an input file.
Input Record Type Code	The code used to distinguish one input record type from another.
Input Table Subroutine	A user-supplied data conversion/validation table or subroutine utilized to convert data from its external form to an internal form required by the user.
Keyword	A value from a keyword indexed field that has been defined as a qualifying value for a specified field in the Index Data set.
Keyword Indexing	The capability to provide file-indexing on values (keywords) included in textual data fields.
Library	A partitioned data set used to store programs, subroutines, tables, RITs, retrievals, and

	queries. The term library may also be used to describe the partitioned data set which is allocated for the subfile capability.
Logic Statement	An executable load module generated by FM from user logic specifications to perform the file update function for one transaction type.
Logical Record	A collection of data elements which is distinct and complete as interpreted by the system. One physical record (block) may contain many logical records.
LOP	Low-Order Position.
Low-Order Position	The rightmost (least significant) position of a field.
Mnemonic	Generally refers to a symbol or name which stands for an equivalent machine-oriented value.
Mode	Refers to the method by which data is stored in a data record (i.e., alphameric, numeric, or coordinate). Also may identify the status of an executing component, e.g., signon mode, update mode etc.
Module	A term used to refer to any mix of components, sections, phases, routines, or subroutines.
Multireel File	A file so large as to require more than one physical reel of tape for storage.
Multivolume File	Same as multireel file except it may pertain to either tape reels or disk packs.
NIPS	NMCS Information Processing System.
OP	System component -- Output Processor.
OS/360	System/360 Operating System.

Output Table/ Subroutine	A user-supplied data conversion table/ subroutine which is used to convert data from an internal system form to an external form required by the user.
Periodic Field	A field defined in a periodic set of a file record, and which may appear more than once in a file record.
Periodic Group	Refer to group. One or more contiguous fields of the same periodic subset, handled as one logical entity.
Periodic Set	A collection of periodic subsets having the same format.
Phase	A collection of routines and/or subroutines which are treated together as a module loaded in core together (also may be referred to as an overlay).
Polygon Overlap	A special geographic retrieval operator which permits selection of file records on such criteria as a point falling within an area, two areas overlapping, a line intersecting another line, etc. See RASP Users Manual.
PSSQ	Periodic subset sequence number.
QDF	Qualifying Data File - An output of RASP; this data set, together with the QRT performs the function of providing an Answer" file. See RASP Users Manual.
QUIP	System component -- Quick Inquiry Processor.
QRT	Qualifying Record Table - See QDF.
RASP	System component -- Retrieval and Sort Processor.

Record Character Count (Field)	A field which is the first two characters of every logical record. It contains the count of characters in the logical record.
Record Control	Refer to Record ID.
Record ID (also called Record Control Group or Record Key)	The initial data field(s) of the fixed set which make each file record in a file unique, and are used to identify the file record. The file records in a file are sequenced according to the contents of their record control group or record ID.
RIT	Report Instruction Table generated by OP to direct output format.
Routine	A logical collection of subroutines and instructions, and is a logical portion of a phase.
Secondary Indexing	The capability to provide file-indexing on values contained in fields other than Record ID's used to speed up the retrieval process.
Section	A named phase(s) for a component.
Section/Phase	When there are no phases within a section, the section, a single operation, is termed a section/phase.
Set	A collection of fields and groups of the same type.
SODA	System component -- Source Data Automation.
Stop Work Table	A user-defined table for the Keyword Indexing capability that consists of values that are to be eliminated from consideration as keywords.

Subfile	A file which is a member of the subfile partitioned data set and consists of selected entries from a data file.
Subroutine	A collection of machine instructions performing a simple, single logical function, and is a logical portion of a routine.
Subset	A periodic subset. A segment of recurring information, composed of periodic fields.
Table	A collection of argument-function pairs organized for efficient searching.
TP	System component -- Terminal Processing.
Transaction	An input record to the PM or SODA components which contains data file update information.
Variable Field	Each set in a record format may have one variable field defined. When defined it carries no size specification and may be used to store unformatted data of variable lengths.
VSCTL	Variable set control field.
VSET	Variable set - A segment of variable length recurring information.

Appendix A

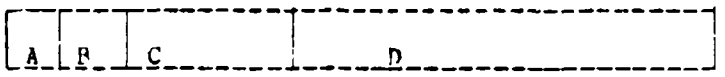
PHYSICAL DESCRIPTION OF THE NIPS 360 PFS DATA FILE AND FILE FORMAT TABLE

The material contained in this appendix is quite technical and should not generally be needed by the average user of the NIPS 360 PFS. However, it is presented here for those users who are interested in the actual manner in which data is referenced and stored in a file. In addition it will aid users who, having dumped the file in image form, desire to locate specific items of information.

The NIPS 360 PFS data file and its associated File Format Table are stored as a DATA SET. The term data set is the OS/360 terminology used to refer to a logical collection of data which is accessible to the system through a unique name.

A.1 Data Set Organization

The NIPS 360 PFS data set is built and maintained using the OS/360 Indexed Sequential Access Method or the Sequential Access Method. Logical records in the data set are variable length and may be up to 1,000 bytes in length. These logical records are blocked into physical records which have a standard or default maximum size of 1,004 bytes or a larger user specified size. When the data set is indexed, each logical record has a key field used to uniquely identify the record. The generalized format of a logical record in the data set is as shown:



A - Four bytes used for OS control; contains length of record.

B - One byte used as a flag to contain a delete code when the record is to be removed from the data indexed set.

C - This field is the record key containing data to uniquely identify the logical record in the data set.

D - This portion of the logical record contains the actual data.

The data set contains several categories of information in its logical records. The primary purpose of the data set is to contain the user's data file which requires the bulk of the space used. Also contained in the data set is supporting information consisting of the FFT and the FM logic statements used during file maintenance. Discussion in this appendix is limited to describing the format and organization of the FFT and data file.

The first character in the record key of each logical record in the data set is used as a code indicating the type of information carried. Being first in the key, it is also used to cause the data set to be sequenced in ascending order based on record types. The general order of record types is as follows:

a. File Format Table records

b. FM Logic Statement records

c. The Statistics Record for ISAM data files

d. Segment Records for Segmented SAM data files

e. User's Data File Records

The character codes used are as follows:

B - Classification Record FFT

C	- Data File Control Record	FFT
D	- Data File Index Descriptor Record	FFT
E	- Non-NIPS Format/ID Record	FFT
F	- Element Format Records	FFT
L&M	- FM Logic Statement Records	
N	- Statistics Record	
P	- Segment Records	
R	- User's Data File Records	

A.2 Data File Records

The format and organization of records making up the data file are discussed in this section.

Each user data record will consist of one or more logical records in the OS/360 data set. There will be a logical record for each fixed set and each subset in a periodic set of the user data record. The major key field for all logical records related as a single user data record will be the same and will contain the record control group. However, the minor key fields will differ based on set type and subset number. Within the data base records, the storage of information will be in two types of notation. For alphameric fields, the information will be stored as EBCDIC characters (i.e., one byte for each character). The numeric fields will be stored as binary words (i.e., four bytes used in binary notation). During FS, the location of binary fields within the logical data record will be controlled so as to conform to boundary alignment requirements when the data record is brought into internal memory.

When the FS component is executed, the format for the logical records is created. All user-defined record elements for the fixed set will define a format for a

logical record used to contain the fixed set. All user-defined record elements for a periodic set will define a format to be used with each logical record which contains a subset of data and so forth. In addition to user-defined elements of a logical record, some elements are automatically generated by the FS component and given special names. They are used for system control. Each distinct element in a logical record (user and system defined) has a corresponding logical record in the FFT which contains information completely describing the attributes of the element. The element name is used in the key of such records.

The maximum size of the user-defined fields in a set can be calculated by knowing the size of the record key and system overhead fields. An FFS set (logical record) consists of a maximum of 1000 characters. These 1000 characters are split among the record key, system overhead fields, and user-defined fields.

The record key contains a minimum of 15 bytes and consists of the following:

- One byte for record type field
- One byte for set ID field
- User defined major control field
- User or system defined set control field.

The size of the record control field will be larger if the major Record ID field is greater than seven bytes or the periodic sets have user-defined control fields greater than four bytes.

The system overhead field consists of a maximum of 15 bytes:

- Four bytes for record size field
- One byte for deletion code field

- Maximum of three bytes for full word alignment prior to the binary block and at the end of the record
- Four bytes for size of variable field.

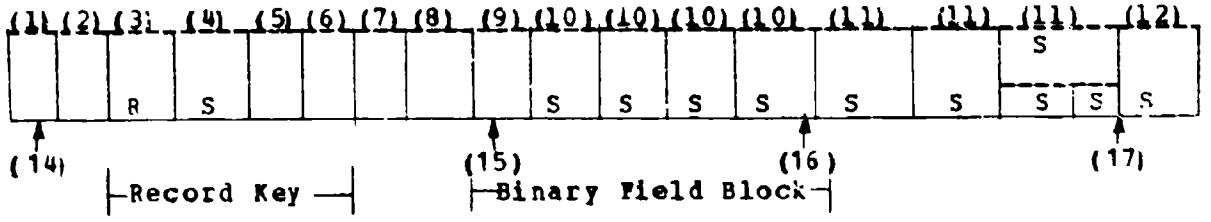
The maximum number of user-defined characters in user-defined fields can be determined by summing bytes in the record control field and system overhead field and subtracting this total from 1000. Whenever the sum of the three categories - record control, system overhead, user-defined - exceeds 1000, FS will issue an error message.

The remainder of this appendix illustrates the typical format for data file records when they reside in the data set. All elements which would be generated are shown.

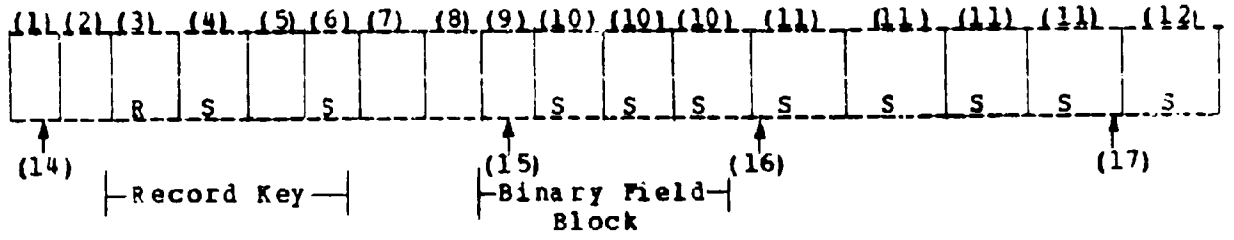
Elements which were directly defined by the user with source statements using the FS component are flagged with the character "S" (see format which follows) to represent the generalized case. Some of the system generated elements have names which start with the character "*". This is used to represent a byte containing all zero bits. When the format for a user's defined set is translated into the format for a logical record, all numeric fields (binary words) are blocked together. This is to ease the requirements for binary field boundary alignment when the logical record is resident in core. That is, data can be worked using machine instructions directly. To accomplish this, whenever the logical record is read into core memory, the record is started on a fullword boundary address. Then, if it is necessary, slack bytes are generated by FS between the key and the block of binary words in the logical record to force the binary block to begin on a fullword boundary in core.

When FS defines the format for a logical record, any needed slack bytes are accounted for in the record description.

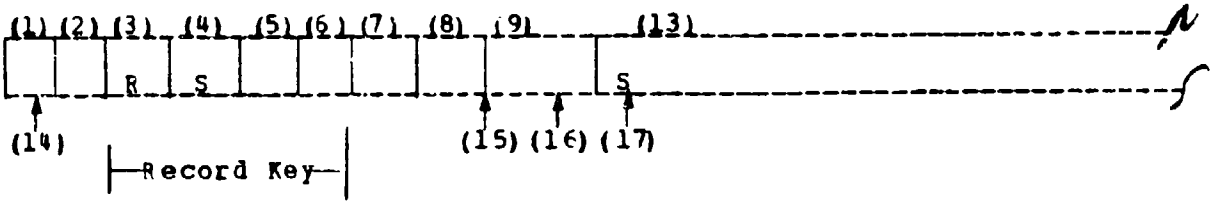
Fixed Set Logical Record Format



Periodic Set (Subset) Logical Record Format



Variable Set Logical Record Format



(1) Record Size Field
Length - Four bytes
Contents - First two bytes are used as a binary halfword to indicate logical record length. The last two bytes are reserved for OS use.

(2) Deletion Code Field
Length - One byte
Contents - Field is set to all binary 1s by the system if the record is to be deleted from the data set under the control of the I/O Supervisor. Otherwise contents are immaterial. Not accessible by user.

The following items (3) through (6) are treated together as the key to the logical record and contents are unique in the data set.

(3) Record Type Field
Length - One byte
Contents - The character "R" to distinguish data records within the data set. System generated name for this field is +FIL.

(4) Record Control Field
Length - Variable
Contents - Contains the data record control group which logically ties all logical records together in the data set which are related to each other (i.e., the fixed set with all its associated periodic subsets). This field size is specified by the user for a particular data set. If the contents for a particular data record are shorter than the field itself, the contents are left-justified. The system generated name for this field is +RCN.

(5) Set ID Field
Length - One byte

Contents - Uses binary notation to identify whether the logical record is fixed or periodic in use. If periodic, it will identify which set it belongs to. The scheme used for identification is -

00000000 - Fixed Set
00000001 - 1st Periodic Set
.
.
.
11111111 - 255th Periodic Set

The system generated name for this field is +PCN.

(6) Subset Control Field

Length - Minimum of four bytes

Contents - When a periodic set does not have a secondary ID specified, these four bytes are used as a number (unsigned zoned EBCDIC) for assigning sequence numbers to the subsets.

When a periodic set has a field(s) specified as a subset control group, the field(s) will appear in the access key and the key field length will be adjusted to accommodate it. When a periodic set has a control field defined which is greater than four bytes, then the length of this key field is enlarged to accept the control data, and this new size will appear for all periodic sets. Periodic sets which have no control field will have their sequence numbers left-justified in the field. Fixed sets will have binary zeros in this field. If necessary, any padding to the right of the decimal sequence number will be with binary zeros.

The system-generated names for this field are PSSQ(n) and +SC(b) when no subset control group is defined for the periodic set. If a subset control group is defined, the only system-generated name is +SC(b).

(Note (b) stands for a byte using binary notation to express the set number.)

(7) Length of Binary Data Block
Length - One byte
Contents - Number of full words making up the binary data block in the data record (field 9 and 10) expressed in binary. System-generated name for this field is +BSZ.

(8) Logical Record Padding
Length - Variable number of bytes.
Contents - Binary zeros for the number of bytes necessary for field nine to begin on a fullword boundary in core memory.

(9) Size of Variable Field
Length - Four bytes (binary fullword).
Contents - Size of variable field if existing. Otherwise all binary zeros. The system generated name for this field is VSZ(n). The system name VSCTL may also reference this field. It is the first variable set created.

(10) User-Defined Numeric Fields
Length - Each is four bytes (binary fullword)
Contents - User-supplied numeric data.

(11) User-Defined Alphameric Fields and Groups
Length - Variable length using EBCDIC characters.
Contents - User-supplied alphameric data.

(12) Variable Fields (fixed or periodic set)
Length - Variable length using EBCDIC characters.
Contents - User-supplied alphameric data.

(13) Variable Field (Defined Variable Set)
Length - Variable as specified on the VSET source language statement in PS.
Contents - User-supplied alphameric data.

(14) The first byte of the data record will be on fullword boundary alignment.

(15) The first byte of the binary word block of a data record is adjusted by the padding of field (8) so as to be on fullword boundary alignment.

(16) The low-order byte of the rightmost binary fullword is addressed by entry number (16) in the control record for a fixed set and by entry number (19) in the control record for a periodic set.

(17) The first byte of a variable field is referenced by the appropriate user-assigned name as found in the element format record.

The following discussion defines in greater detail the operation of the system-generated fields PSSQ(n) and +SC(b).

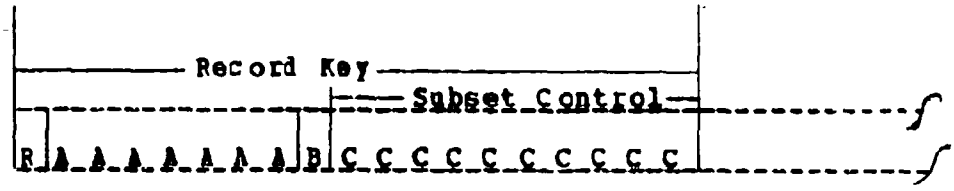
The minor sort field of the key for a logical record is defined as the Subset Control Field. For data files defined with periodic sets in which no subset control groups were required (data dependent), this subset control field will be four bytes in length. Two system-generated field names (+SC(b) and PSSQn) will reference this field. Its contents will be decimal numbers used for subset sequencing.

For a data file having mixed periodic sets (i.e., periodic sets without control groups and some with control groups), the following conventions apply. A PSSQ(n) field name will be generated only for those sets which have no control group and reference is made to the first four high-order bytes of the subset control field. A +SC(b) field

name will be generated for all periodic sets and will reference only the significant data contained in the subset control field.

An example for discussion above, consider the case when a data file has three periodic sets defined. Two of these periodic sets have subset control groups which differ in length. In the following format, each character represents a byte.

FIXED SET

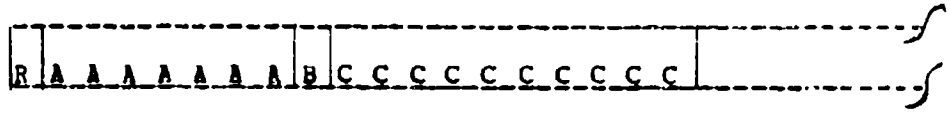


A - Record ID Value

B - Eight binary zeros indicating fixed set

C - All 10 bytes have binary zeros

PERIODIC SET 1 (10-characters periodic control group)

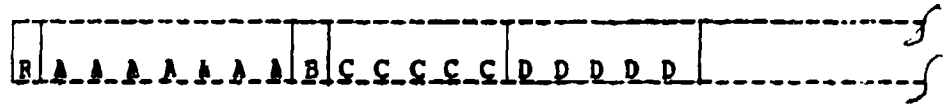


A - Record ID value

B - Binary content of byte is 00000001 indicating 1st periodic set.

C - Contains periodic control value. The system generates the field name +SC(b) for this 10-byte field. "b" has the binary value 00000001.

PERIODIC SET 2 (5-character periodic control group)



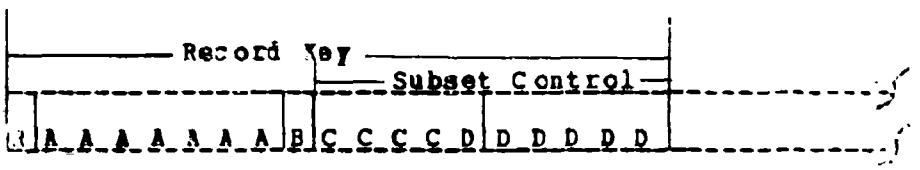
A - Record ID value

B - Binary content of byte is 00000010 indicating second periodic set.

C - Contains periodic control value. The system generates the field name +SC(b) for this 5-byte field. "b" has the value 00000010.

D - Remaining five bytes are padded with binary zeros.

PERIODIC
SET 3
(No periodic
control group)



A - Record ID value

B - Binary content of byte is 00000011 indicating third periodic set.

C - Contains the subset sequence number. The system generates the field name +SC(b) and PSSQ 3 for this 4-byte field. 'b' has the value 00000011.

D - Remaining six bytes are padded with binary zeros. Note that the length of the subset control field in the access key for the entire data file is dependent upon the largest periodic control group defined. All other sets have their values left justified. Also the names +RCN and +SC(b) are generated by the system even though the user-supplied names for the same fields.

The following conventions concerning group definitions during FS are used:

- An alphameric group containing all alphameric fields will have all fields in EBCDIC character notation (mode code "A").
- An alphameric group containing one or more numeric fields will have these numeric fields generated in zoned EBCDIC decimal notation (mode code "D").

- A numeric group containing all numeric fields will have all fields generated in zoned EBCDIC decimal notation (mode code "D").

- A numeric group containing both alphameric and numeric fields will not be allowed.

- Numeric fields or groups may not be used as record control or subset control groups. Only EBCDIC characters may be used in the access key.

- A coordinate group contains fields in the binary block of the logical records. Each field is a binary word capable of containing either a latitude or longitude value.

A.3 File Format Table Records

This subsection discusses in sequence the types records found in the FFT portion of the data set.

A.3.1 Classification Record

There is one classification record in the OS/360 data set. It appears first, and its purpose is to carry the user-supplied classification label defined when File Structuring was run. The format for the classification record is:

(A)	(B)	(C)	(D)	(E)	(F)
				XXXX...XX	

(A) Record size field - contains X'104' (for files structured under 360 NIPS), or X'108' (for files converted from 1410 to 360 NIPS)

(B) Reserved for OS

(C) Delete code field - contains X'00'

(D) Record type field - contains C'B'

(E) Classification - contains classification literal left-justified in a 32-byte field. Any padding to right will be with blanks.

(F) Date/Time of last update - contains date/time of last FM or SODA update of the file.
Format: YYDDDHHMMSS (year, day, hour, minute, second)

(G) Number of fields indexed - two bytes binary

(H) Number of SUB/TAB entries in all Index Descriptor Records

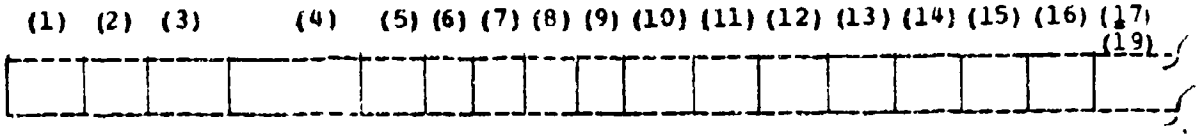
(I) Slack bytes to bring record to a size greater than a maximum key.

A.3.2 Data File Control Record

The data file Control Record(s) appear sequentially following the Classification Record. Its purpose is to supply information to the using FFS component on the organization and format of the element format records. In a sense, it provides the bootstrap information needed for a component to interpret correctly the element format records. In addition it supplies basic information on the organization of the resident data file.

The format of the data file Control Record and description of its contents follow:

Group Repeats for each periodic set



(1) Record Size Field
Length - Four bytes
Contents - First two bytes are a binary halfword

used to specify record length. The last two are reserved for OS use.

(2) Deletion Code Field
Length - One byte
Contents - All binary 1s set by system if record is to deleted from the data set. Otherwise contents are immaterial. Not accessible by user.

(3) Record Type Field
Length - One byte EBCDIC
Contents - The character 'C'.

(4) Control Record Key Padding
Length - 254 bytes
Contents - Binary zeros throughout all bytes. Used to force the fixed information carried in the control record beyond the largest access key that may be defined. Will contain a 'C' in high-order byte in continuation records, followed by hexadecimal blanks throughout the remainder of the field.

Note: The access key for the control record is made up of field (3) and all or part of field (4) depending on the length required for the data file.

(5) High-Order Position of Record Control Group in the Record Key of user data records (logical).
Length - Binary halfword
Contents - Location is relative to the high-order byte of the record size field which is based at zero. Data content of this field in continuation records is not significant.

(6) Length of Record Control Group
Length - Binary halfword
Contents - Size of record control group. Data content of this field in continuation record is not significant.

(7) High-Order Position of Set ID Field in the Record of User Data Records (Logical) Key
Length - Binary halfword
Contents - Location specification same as (5). Data content of this field in continuation record is not significant.

(8) Length of Set ID Field
Length - Binary halfword
Contents - Size of field (1 byte). Data content of this field in continuation record is not significant.

(9) High-Order Position of Subset Control Group in the Record Key of User Data Record (Logical)
Length - Binary halfword
Contents - Location specification same as (5). Data content of this field in continuation record is not significant.

(10) Length of Subset Control Group
Length - Binary halfword
Contents - If no periodic set control group for the data file has been defined, the size will be four bytes, otherwise the size of the largest periodic set control group specified will be used. Data content of this field in continuation record is not significant.

(11) Number of Periodic Sets
Length - Binary halfword
Contents - The number of periodic sets defined for the data file up to 179. If more than 179 periodic sets were defined, the number in excess of 179 will appear in this field in the continuation record. If no periodic sets were defined, this field will contain binary zeros.

(12) High-Order Position of Significant Data in the Element Format Records

Length - One byte using binary notation
Contents - Provides the relative high-order position of data contained in the element format records. The first byte of the element format record is considered at a zero location. This field is used because there may be byte padding between the last character of the access key and the first byte of data contained in the element format record. This will allow half boundary alignment for the binary entries in those records.

(13) Dummy Entry

Length - Three bytes
Contents - High-order byte contains a 'C' if a continuation record follows. Data content of this field in continuation record is not siginificant.

(14) Length of Fixed Set Logical Record

Length - One byte using binary notation
Contents - Size in fullwords includes record size field, deletion code field, access key, and all defined fixed length fields. In addition, it may include some padding (binary zeros) at end of set so that the entire logical record will conform to full word boundary alignment. Data content of this field in continuation record is not significant.

(15) Number of Binary Words in the Fixed Set Logical Record

Length - One byte using binary notation.
Contents - Number of fullwords in the block of binary words which are contained in the fixed set. Data content of this field in continuation record is not significant.

(16) Low-Order Position of Binary Block in Fixed Set (low-order byte) Logical Record
Length - Binary halfword
Contents - Location relative to the first byte of the record size field which is based at zero. Data content of this field in continuation record is not significant.

Note: The following fields in the control record are optional.

(17) Length of First Periodic Set Logical Record
Length - One byte using binary notation
Contents - Size in fullwords as was specified for field (14) above.

(13) Number of Binary Words in First Periodic Set Logical Record
Length - One byte
Contents - Number of fullwords in the block of binary words which are contained in the first periodic set. Binary notation is used in this field.

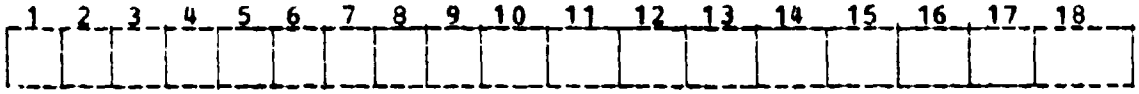
(19) Low-Order Position of Binary Block in the First Periodic Set Logical Record
Length - Binary halfword
Contents - Position specified same way as for field (16) above

Note: The fields (17), (18), and (19) will be repeated as a group to define each periodic set, up to a maximum of 179 sets. A continuation record will be generated to 255. Contents of fields (17), (18), and (19) in the continuation record is as stated for the control record. See paragraph A.3.5, Continuation Techniques, in this appendix. While reading this appendix, it may be best to study the data record format for logical records contained in paragraph A.2 of this appendix.

A.3.3 Index Descriptor Record

Each field or group in the user's data file which has been specified as an index has a special record in the data set signifying this fact as well as containing certain field location and attribute information. These records are known as Index Descriptor Records. Each is a logical record containing in its key field the field or group name which has been indexed. The records are generated along with other FFT records during an FS run, or they are inserted into an existing FFT during an Index Specification run. In addition to information supplied by the user on Index Specification statements, elements from the corresponding "F" (Element Format) records for that field or group are present in the record.

The remainder of this section illustrates the format and contents of the Index Descriptor Record.



(1) Record Size Field
- Length - Four bytes
- Contents - First two bytes make up a binary halfword providing the size of the logical record. The last two bytes are reserved for OS use.

(2) Deletion Code Field
- Length - One byte
- Contents - All binary 1's set by system if the record is to be deleted from the data set by the I/O Supervisor. Otherwise, contents are immaterial.

(3) Record Type Field
- Length - One byte EBCDIC.
- Contents - The character 'D'.

(4) Field or Group Name
- Length - Variable number of EBCDIC characters.

Contents - Data record element name.

(5) Padding
Length - Variable number of bytes required to ensure that fields 4 and 5 fill space equal to that of the record key for the file.
Contents - Binary zeros throughout all bytes.

(6) Boundary Alignment Byte
Length - One byte, if necessary.
Contents - This is a slack byte which will appear if necessary to force all following fields to observe halfword boundary alignment.

(7) Set Identification
Length - One byte in binary notation
Contents - 00000000 - Fixed Set
00000001 - Periodic Set 1
00000010 - Periodic Set 2
Etc.

(8) Type Identification
Length - One byte using binary notation.
Contents - The content and format of this byte are identical to the Element Type Identification described as element #9 under Section A.3.4, Element Format Records.

(9) High-Order Location of Element in Record
Length - Four EBCDIC bytes.
Contents - The high-order location of the specified element relative to beginning of record (based at zero).

(10) Length of Element in Logical Record
Length - Three EBCDIC bytes.
Contents - The content and format of this element are identical to the Length of Element in Logical Record described as element #11 under Section A.3.4, Element Format Records. The low-order bit (bit 7) will

be set to 1 for a keyword indexed field.

NOTE: No field longer than 30 bytes may be specified as an index. There is no maximum length for a keyword indexed field, however, there is a maximum length of 30 for any designated keyword value within the field.

(11) Element Mode in the Logic Record
Length - One EBCDIC byte.
Contents - A=Alphameric, N=Numeric, C=Coordinate or D=Decimal are the acceptable modes.

(12) Input Subroutine Conversion or Stop Word Table Name
Length - Eight bytes EBCDIC.
Contents - Subroutine name left justified. Blank if no conversion on input. Asterisk (*) left justified if element is coordinate mode and has external length of 5, 6, 7, 8, 11, or 15. This automatically invokes a standard system conversion subroutine. This routine is specified during File Structuring not Index Specification. For a keyword index, this field contains the name of the stop word table, if one were designated, otherwise, the field is blank.

(13) File Form to Index Form Subroutine Name
Length - Eight bytes EBCDIC
Contents - Subroutine name left justified. For a secondary index, the name specifies a conversion subroutine. For a keyword index, the name indicates the dictionary. The field is blank if a subroutine or table is not designated. The function is defined on a SUB/TAB statement during Index Specification. Refer to the RASP manual (Volume IV) for a discussion of this function.

(14) Analyzer/Scan Subroutine Name
Length - Eight bytes EBCDIC
Contents - Subroutine name left justified. For a secondary index, it designates the name of an analysis subroutine. For a keyword index, it designates the name of the user scan subroutine. The field is blank when a subroutine is not specified.

(15) Length of Element as Carried in the Index Data Set/Hyphen Option
Length - One byte binary
Contents - Length minus 1 (e.g., a true length of 1 becomes 0 under this concept) of the data as it is actually carried in the Index Data Set. The element length has no meaning for variable length keyword data in the index data set. Therefore, for keyword indexes, this byte will specify the option indicating how a hyphen is to be treated when found within the data field of a transaction record.

(16) Reserved Byte
Length - One byte binary.
Contents - Zero.

(17) Mode of Element in Index Data Set Format
Length - One byte EBCDIC.
Contents - Same as for (11) above, but this entry refers to the data format of the Index Data Set.

A.3.4 Element Format Records

Every element in a user's data record has a special record in the data set defining its location and attributes. These records are known as Element Format Records. Each is a logical record containing in its key field the name of the element that it describes. The records are generated along

with the classification and control records by the FS component. In addition to user-defined record elements (from File Structuring source statements) additional elements appear in the logical record format as illustrated in section A.2. These elements are generated automatically during structuring for internal control purposes. They have special names and their own corresponding Element Format Records. The system-generated elements and their purposes are listed below:

a. ♦FIL - This element contains the first character in the logical record key which contains "R." This character is common to all data records and is used to batch all data records as a block within the OS/360 data set.

b. ♦RCN - This element contains the total record control group as found in the logical record key.

c. ♦PCN - This element contains the set ID field in the key of the logical record.

d. ♦SC(b) - This element redefines the subset control group in the key for a specific subset logical record. The fourth byte in the name(b) will use binary notation to reference a specific set; for example:

0000000 - Fixed Set
0000001 - 1st Periodic Set
0000010 - 2nd Periodic Set

e. ♦BSZ - This element will occur immediately after the key in a logical record (1 byte in length) and will specify, via binary notation, the number of binary fullwords within the logical record's binary data block.

f. PSSQ(n) - This element definition is generated only for those periodic sets which have not been defined by the user to have a subset control field (based on a data value). It identifies a 4-byte field in the key of a logical record used

for subset sequencing within a periodic set. The term (n) represents a one-to-three EBCDIC character suffix used for periodic set identification; for example:

PSSQ25 will reference the subset sequence field for a logical record of Periodic Set 25.

g. VSZ(n) - This element is the first binary word in the binary data block of a logical record (fixed set or periodic subset). This binary word will indicate the number of characters currently contained in the logical record's variable field. The characters indicated by (n) will refer to the periodic set involved and are stated using EBCDIC numbers. For example:

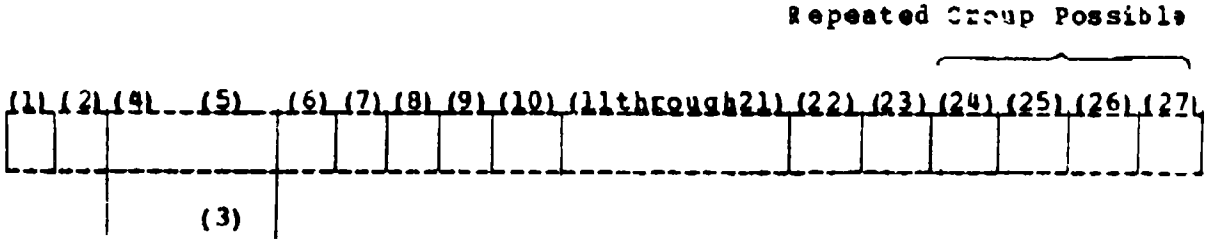
VSZ - Fixed Set
VSZ15 - 15th Periodic Set

If there is no variable field for a logical record, this field (four bytes) will contain binary zeros.

h. VSCTL - This element is a redefinition of VSZ(n) element for the logical record containing the first defined variable set.

Note: The system-generated fields (a) through (e) may only be used internally by the FFS component. No analyst/user may communicate to component using these names. In contrast, the field names PSSQ(n), VSZ(n) and VSCTL may be used by the analyst as a method of controlling this particular run. The use of the character (*) in the above names means a byte consisting of binary zeros. For a complete understanding of the use of the generated field names, it may be best to refer to the description of the data record found in section 2.

The remainder of this section illustrates the format and contents of the Element Format Record.



(1) Record Size Field
- Length - Four bytes
- Contents - First two bytes make up a binary halfword providing the size of the logical record. The last two bytes are reserved for OS use.

(2) Deletion Code Field
- Length - One byte
- Contents - All binary 1's set by system if the record is to be deleted from the data set by the I/O Supervisor. Otherwise contents are immaterial.

(3) Record Key
- Length - Variable EBCDIC characters. Length is standard for entire data set and is dependent on the user specifications concerning the size of the record control group and the periodic control group (if defined).
- Contents - See (4) and (5).

(4) Record Type Field
- Length - One-byte EBCDIC
- Contents - The character "F." This code defines the logical record and an Element Format Record.

(5) Element Name
- Length - Variable length EBCDIC characters.

Contents - Data record element name left justified within this portion of the access key. If the element name is less than seven characters, it is padded to the right with blanks until a total size of seven is reached. After that, any remaining key padding is done with zero bits. See Continuation Record Techniques at end of section for modifications on continuation records.

(6) Boundary Alignment Byte
Length - One byte if necessary
Contents - This is a slack byte which may appear in the Element Format Record. This is used as padding to force all following fields in the record to observe halfword boundary alignment. Entry 12 of the control record is used to point to the location immediately following this byte indicating the start of record data. (High-order address of entry 7.)

(7) Dummy Parameter
Length - Four bytes
Contents - Null characters normally. Contains 'C' in high-order byte in continuation records.

(8) Element Set Identification
Length - One byte in binary notation
Contents - ********0 - Fixed Set
********1 - Periodic Set 1
*******10 - Periodic Set 2
Etc.
Not used in continuation records.

(9) Element Type Identification
Length - One byte using binary notation
Contents - The element definition is accomplished by the presence of bits in certain

locations of the byte. A bit turned on will contain a "1." A bit turned off will contain a "0." The format of the byte is as follows:

Bit No.	Ø 1 2 3 4 5 6 7
Ø	ON - Field OFF - Group
1	ON - Field or group is used for record or subset control. OFF - Non control use
2	ON - System generated field/group OFF - User-defined field/group
3	ON - Field/group may not be used by the analyst. OFF - Field/group is unrestricted
4	ON - Fixed Length Field OFF - Not Fixed Length
5	ON - Variable Length Field OFF - Non variable length
6	ON - Variable set field OFF - Non variable set field
7	Always Ø.

Ø 1 2 3 4 5 5 7
Ø 1 Ø Ø 1 Ø Ø Ø

The file format record describes the user-defined record control group. The field is not used in continuation records.

The hex values of this byte for all element types are summarized below.

A. System Generated Elements:

+FIL +RCN +PCN +SC(B)	X'F8'
VSCTL VSZ(n)	X'A8'
PSSQ(n) -	X'E8'
+BSZ -	X'B8'

B. User-Defined Elements:

Non-control field	- X'88'
Non-control group	- X'08'
Variable set name	- X'82'
Variable field	- X'84'
Control field	- X'C8'
Control group	- X'48'

(10) High-Order Location of Element in Logical Record
Length - Four bytes using EBCDIC notation
Contents - Location is relative to the high-order byte of the record size field which is based at zero. Data content of this field in a continuation record is not significant.

(11) Length of Element in Logical Record
Length - Three bytes using EBCDIC notation
Contents - (A) Length is specified for the number of alphameric characters represented. For alphameric mode elements (A), this will be the actual number of bytes appearing in the data record. For numeric mode elements (B), a

binary word (4 bytes) will appear in the logical record regardless of the length specified. For decimal mode elements (D), this value will be the actual number of bytes in the logical record. See paragraph (12) below for a discussion on element modes.

(B) If this is a variable field, the entry will contain the number of characters per line to be printed during output.

(C) If this is a variable set field, the length is as specified in the VSET FS statement.

(D) Coordinate mode elements are handled in a special manner. The size appearing in (11) depends on certain circumstances. The Element Format Records generated to define coordinate fields/groups are similar to other user-defined fields/groups with the following exceptions noted:

ALL FIELDS defined for coordinate use and single coordinate groups (one latitude field and one longitude field) will carry the external decimal length value (i.e., length as defined by user in the FS field statement) in the element format as parameter (11)). All groups defining more than one coordinate point will carry the actual internal length in bytes of the binary representation of the coordinates (internally, latitudes and longitudes are each represented by a full binary word). For

example, if POINT is defined as a field of length of 11, representing both latitude and longitude, the length carried in entry 11 of the Element Format Record will be 11. If POINT is defined as a group of two fields of length 5 and 6 characters, the length of the group will be specified in the Element Format Record as 11 (representing the sum of the two fields). If LINE is defined as a group of two coordinate fields, each characters long externally, the length of the group will be specified in the Element Format Record as 16 bytes for the four full binary words representing the coordinates internally.

Three cases and their handling during FS:

Case 1 - A user defines a single coordinate field intending to store both latitude and longitude values in it. The field will be either 11 or 15 characters in length depending on the precision desired.

The FS component will cause a single element format to be built with the name supplied by the user. However, this record will define two adjacent binary words in the block portion of the logical record, and will address the high-order byte of the leftmost word. The length of the coordinate field will be specified as either 11 or 15 characters as defined by the analyst in parameter 11 of the Element Format Record.

Case 2 - A user defines two fields of length 5(7) and 6(8) characters intending to identify latitude

and longitude separately. In addition, a group is defined as containing these two fields.

The FS component will cause two adjacent binary fields to be generated, with an Element Format Record for each. The contents of the Element Format Record describing each field will be as in case one, except that the field length entry (11) will describe only the user-specified length for that field. The group format record will contain the sum of the user-specified length of each field defined in the group.

Case 3 - A user has defined several sets of coordinates by the method of case one or case two, as discussed previously. In addition, he defines this collection as a group.

In addition to the Element Format Records generated as in cases one or two, the FS component will generate a group format record describing this collection of fields. Parameter 11 in the group format record will state in bytes the space needed for binary words. This field is filled with null characters in continuation records.

(12) Element Mode Specification
Length - One byte using EBCDIC notation
Contents - Alphameric mode element (A).
Numeric mode element (B).
Coordinate mode element (C).
Decimal mode element (D).
(Decimal mode is implicit in File Structuring and is assigned to numeric fields included in a GROUP statement.) The data content of this field in a continuation record is not significant.

(13) Input Subroutine Conversion Name
Length - Eight bytes EBCDIC
Contents - Subroutine name left justified.
Zero bits if no conversion on input.

Asterisk (*) left-justified if element is coordinate mode and has external length of 5, 6, 7, 8, 11, or 15. This invokes automatically a standard system conversion subroutine. Data content of this field in a continuation record is not significant.

(14) Output Subroutine Conversion Name
Length - Eight bytes EBCDIC
Contents - Subroutine name left justified. Zero bits if no conversion on output. Asterisk left justified, same as (13). Data content of this field in a continuation record is not significant.

(15) High-Order Location of Element Label in this Format Record.
Length - Binary halfword
Contents - Location specification same as (10) if label present. All zero bits for no label. Null characters in continuation records.

(16) Length of Element Label in this Element Format Record
Length - Binary halfword
Contents - Size if label exists. All zero bits if no label. Data content of this field in a continuation record is not significant.

(17) High-Order Location of Edit Mask this Element Format Record
Length - Binary halfword
Contents - Location specification same as (8) if pattern assigned to element during File Structuring. All zero bits if no pattern. Data content of this field in a continuation record is not significant.

(13) Length of Edit Mask in this Element Format Record
Length - Binary halfword
Contents - Size if pattern assigned to element during File Structuring.
All zero bits if no editing is used.
Data content of this field in a continuation record is not significant.

Note: When edit masks appear in element Format Records, they are in FPS edit pattern form.

(19) Size of Element on Output
Length - Binary halfword
Contents - This field contains the size (in bytes) for output.
If output conversion is used, the size of the subroutine output is provided.
Data content of this field in a continuation record is not significant.

(20) High-Order Location of the String of Field Names in the Record Making up the Group
Length - Binary halfword
Contents - Location specification same as (10) if required.
All zero bits are used if entry is not a group.
Data content of this field in a continuation record is not significant.

(21) Number of Fields Making up the Group
Length - Binary halfword
Contents - Size if requirement exists.
All zero bits if not required.

All the following entries are optional and are used if required.

(22) Field Label Used for Output
Length - Variable (EBCDIC Character)
Contents - User-assigned label name.
Not used in continuation records.

(23) Edit Mask Pattern
Length - Variable (EBCDIC Characters)
Contents - Edit pattern.
Not used in continuation records.

(24) Field Name within Group
Length - Eight bytes EBCDIC
Contents - Field name left justified.

(25) High-Order Location of Field in Logical Record
Length - Four bytes in EBCDIC notation
Contents - Location specification same as (10).

(26) Length of Field in Logical Record
Length - Three bytes in EBCDIC notation
Contents - Length specification same as (11).

(27) Character Set Specification
Length - One byte EBCDIC
Contents - A - alphameric field (EBCDIC)
D - decimal field (EBCDIC).

Note: Fields 24-25-26-27 may appear as multiple entries specifying from left to right the fields making up the group for which the current record is identifying. All system-generated fields will use entries (1) through (10) with the exception of +RCN and +SC(B) which will list all user-defined fields making up the control group with entries (20), (21), and (24) through (27).

A.3.5 File Statistics Record

NIPS files that have been generated in ISAM and VSAM organization will contain a statistical (N) record. If the file has been generated in sequential organization a statistical record will not exist, however, a statistical record will be generated when the sequential file is copied to ISAM or VSAM organization using the NIPS UTBLDISM utility. The statistics record is maintained by a NIPS utility when the file is generated or updated and certain information concerning lengths of NIPS records. The following information is recorded in the 'N' record:

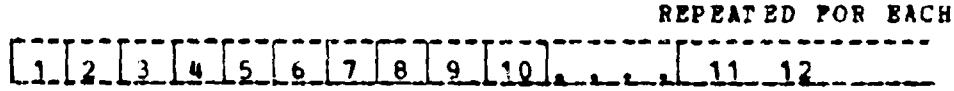
a. Number of sets described by the 'N' record

b. Number of sets in the file

c. Length of the longest subset for each set

d. Maximum number of subsets for each set.

The remainder of this section illustrates the format and contents of the File Statistics Record.



(1) Record size field
Length - Four bytes
Contents - First two bytes make up binary half-word providing the size of the logical record. the next two bytes are not used.

(2) Deletion Code Field
Length - One byte
Contents - All binary 1's set by system if the record is to be deleted from the data set by the I/O supervisor. Otherwise, contents are immaterial.

(3) Record type field

Length - One byte
Contents - The character 'N'

(4) Continuation Number
Length - One byte
Contents - First 'N' record blank. continuation record contains a number indicating order of sequence starting with 1.

(5) Number of Sets Described by 'N' Record
Length - One byte

Contents - Number of sets, including the fixed set described by the 'N' record.

(6) Highest 'N' Record
Length - Two bytes
Contents - First byte contains an 'N' in EBCDIC. Second byte contains highest continuation number for 'N' records in this file.

(7) High Order Position of Significant Data
Length - One byte
Contents - High order position in the 'N' record, of the description of the first set (fixed) in the file, (see field 11).

(9) Number of sets in the file
Length - One byte
Contents - Number of sets in the file.

(10) Reserved by future use
Length - Three bytes
Contents - Binary zeroes.

(11) Length of Longest Set N in the File
Length - Four bytes
Contents - Contains the total length of all subsets for the longest set N for a single NIPS record in the file.

(12) Maximum Number of Subsets
Length - Two bytes
Contents - Contains the maximum number of subsets for the longest set n for a single NIPS record in the file.

Note: Fields 11 and 12 will repeat once for each set in the file.

A.3.6 File Segment Record

Sequential files that have been segmented contain segment records (P) that specify the range of record control values permitted for a segment and the volume serial number

of the segment. A file segment may be composed of more than one range of records. A segment record will be generated for each range of records in one data set.

The remainder of this section illustrates the format and contents of the File Segment Record.

(1) Record Size Field
Length - Four bytes
Contents - First two bytes make up a binary halfword providing the size of the record. The last two bytes are reserved for OS use.

(2) Deletion Code Field
Length - One byte
Contents - All binary 1's set by OS if the record is to be deleted.

(3) Record Type Field
Length - One byte
Contents - The character 'P'

(4) File Record Control Value (Low Range)
Length - Variable; based on major record ID length.
Contents - Record control value that is assigned as the low range limit for data file records on this segment.

(5) File Record Control Value (High Range)
Length - Variable; based on major record ID length.
Contents - Record control value that is assigned as the high limit for data file records on this segment.

(6) Volume Serial Number
Length - Six bytes
Contents - Volume serial number for data file where range of records are indicated in the 'P' record.

A.3.7 Continuation Record Techniques

There are occasions when the data contents of the control record and group format records may exceed the 1,000 byte logical record size allowed in the OS/360 data set. This section describes the manner in which the File Structuring component handles such cases.

A.3.7.1 Continuation Records for the FFT Control Record

Because of the logical record length limitations, the control record is only able to supply information on a maximum of 179 periodic sets. Since the system has been designed to handle, theoretically, up to 255 periodic sets for each named data set, it becomes necessary to provide a continuation record when the number of periodic sets defined by the user exceeds 179. When such a case occurs, a second control record will be created to continue the information on periodic sets (entries 17-18-19).

The primary (first) control record specifies the total number of periodic sets that it defines in entry 11. The high-order byte of entry 13 contains the character "C" indicating that a continuation record follows. The secondary control record will have the same format as the primary. However, it will have the character "C" in its key immediately following the record type field (entry 3). The entries 5-10 and 12-16 are not maintained, but their length is the same as in the primary. Entry 11 contains the number of periodic sets defined by the secondary record. Entries 17, 18, and 19 are used and repeated until all periodic sets have been accounted for.

A.3.7.2 Continuation Records for Group Format Records

Similar to the problem faced by the control record, the Element Format Record for a group may experience overflow cases. This overflow of data results from the series of entries which lists each field (group) contained within the defined group. The following table illustrates the number of fields that a group format record may define using a single logical record.

	OS Fields	five bytes
FFT	Record key	from 8 to 255 bytes
RECORD ENTRY LENGTHS	Fixed entries	40 bytes
	Field/group label	from 0 to 132 bytes
	Edit pattern	from 0 to 132 bytes
	Field length specs in group format record	16 bytes per field

a. Worst case assuming max key, label, and edit length will allow 27 fields (groups) to be defined as a single group within a 1,000-byte record.

b. Best case assuming min key, and no label or edit pattern, will allow 59 fields (groups).

c. Typical case with key length of 15 bytes, label length of 8 bytes, and edit length of 8 bytes will allow 57 fields (groups).

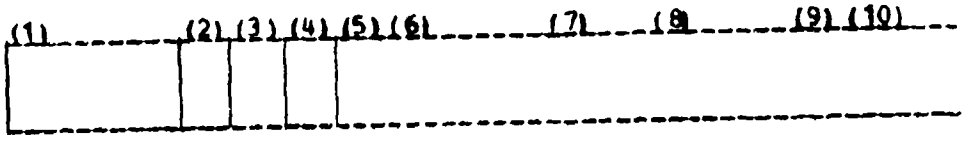
When a continuation record is generated, entry 21 in the primary record will state only the number of fields that it lists. The high-order byte of entry 7 will contain the character "C" to indicate that continuation record(s) follow. The continuation records will have the same format as the primary. However entries 8 through 20 will not cont[illegible] valid data and entries 22 and 23 will not appear. The secondary record key will contain the group name, as

usual, but will be suffixed by an eighth byte using binary notation to indicate the number of the continuation record. The first continuation record would contain "1" in binary and so forth. Entry 21 in the continuation record will contain a number indicating how many fields are contained in the list of entries 24, 25, 26, and 27.

A.3.8 Non-NIPS Format/ID Record

When a pseudo-FFT is structured, one or more records are created which contain the Format/ID table. This table describes the location of each portion of the user designated record ID within each record format. The table is ordered as the different formats were defined in FS.

The format of the non-NIPS Format/ID record and description of its contents follows:



(1) Record Size Field
Length - Four bytes.
Contents - First two bytes make up a binary halfword providng the size of the logical record. The last two bytes are reserved for OS use.

(2) Deletion Code Field
Length - One byte.
Contents - All binary 1's set by system if the record is to be deleted from the data set by the I/O supervisor. Otherwise contents are immaterial.

(3) Record Type Field
Length - One byte.
Contents - The character "E". This code identifies the record as a non-NIPS Format/ID record.

(4) Continuation Indicator Field
Length - One byte.
Contents - Binary zeros for the first "E" record or the character "C" if the record is a continuation of a previous "E" record.

(5) Continuation Sequence Field
Length - One byte.
Contents - A binary sequence number starting with a binary 1 for the first continuation record.

(6) Control Record Key Padding
Length - Six bytes.
Contents - Binary zeros to insure proper sorting of the FFT.

Fields 7 through 10 are repeated for each record format defined in the FFT.

(7) Next Segment Offset
Length - Binary halfword.
Contents - The offset to the next segment of the "E" record. The offset is relative to the beginning of the record. When the offset is all binary 1's, the next segment is in a continuation "E" record. When the offset is all binary zeros, this is the final segment.

(8) Record Type Code
Length - Variable, maximum of 10 bytes.
Contents - The record type code used to uniquely identify the current record format.

(9) Alignment Byte
Length - One byte, when used.
Contents - Immaterial, used to force halfword alignment for field 10.

(10) Record I Segment Descriptor
Length - Four bytes.
Contents - This field consists of two halfword binary fields. The first field contains the relative high-order position of the record ID segment. The second field contains the length, in binary, of the record ID segment. These two fields are repeated as necessary to describe each segment of the record ID in the current record format.

Appendix B

DESCRIPTION AND USE OF THE TRANSACTION RECORDS OUTPUT BY THE FILE ANALYSIS STATISTICS CAPABILITY

The material in this appendix illustrates the use of the transactions that are output by the File Analysis Statistics capability. A sample of the two transaction formats are included. Sample FFT, logic statements, queries, and RITs are shown illustrating a possible use of the transaction records.

B.1 Sample Transactions from File Analysis Statistics

Transaction Record - UTFLDSCN 50 bytes

```
          1         2         3         4         5
STEST360FM  TESTO  A     SERV    00000003031771
```

Column 1	-	CHARACTER 'S'
2-8	-	FILE NAME
9-12	-	COMPONENT NAME
13-25	-	SOURCE MODULE NAME
26-33	-	FIELD NAME
34-36	-	SET NUMBER, DECIMAL
37-42	-	COUNT OF REFERENCES, DECIMAL
43-48	-	DATE - MMDDYY
49-50	-	UNUSED

Transaction Record - Component Execution 50 bytes

```
          1         2         3         4         5
CTEST360FM  TESTO  A     000026   031771
```

Column 1	-	CHARACTER 'C'
2-8	-	FILE NAME
9-12	-	COMPONENT NAME
13-25	-	SOURCE MODULE NAME
26-31	-	COUNT OF EXECUTIONS, DECIMAL
35-40	-	DATE EXECUTED, MMDDYY
41-50	-	UNUSED

B.2 File Structure Deck

```
    STRUCTURE TESTERF.

    CLASSIFICATION 'UNCLASSIFIED'.

NOTE:  This data file is the statistics data file for the
       TEST360 file.  The file will utilize the transactions
       output by the File Analysis Statistics capability.

NOTE:  The following elements are used for record control.

EDIT     DATE      'ØM/ØØ/ØØ'.
EDIT     COUNT     'ØØØØØ0'.
FIELD    FILEN     7 C  ALPHA       'FILE NAME'.
FIELD    COMP      4 C  ALPHA       'COMPONENT'.
FIELD    SOURCE   13 C  ALPHA       'COMPONENT SOURCE NAME'.
GROUP    RECID     FILEN,COMP,SOURCE  ALPHA   'RECORD CONTROL'.
FIELD    CNTEX     6 X  NUMER COUNT 'COUNT OF EXECUTIONS'.
FIELD    IMON      2 X  NUMER.
FIELD    IDAY      2 X  NUMER.
FIELD    IYEAR     2 X  NUMER.
GROUP    IDATE     IMON,IDAY,IYEAR  DATE  'INITIAL DATE'.
FIELD    LMON      2 X  NUMER.
FIELD    LDAY      2 X  NUMER.
FIELD    LYEAR     2 X  NUMER.
GROUP    LDATE     LMON,LDAY,LYEAR DATE  'LATEST DATE'.

NOTE:  The following field is used for subset control.

FIELD    FLDNAM    8 CL  ALPHA      'FIELD NAME'.
FIELD    SETN      3  1  ALPHA      'SET NUMBER'.
FIELD    CNTREF    6  1  NUMER COUNT 'COUNT OF REFERENCES'.
FIELD    UMON      2  1  NUMER.
FIELD    UDAY      2  1  NUMER.
FIELD    UYEAR     2  1  NUMER.
GROUP    UDATE     UMON,UDAY,UYEAR  DATE  'DATE UTILITY EXEC'.
END.
```

B.3 FM Logic Statements

The following logic statements may be used to either generate new records or to update existing records on the statistics file.

B.3.1 C Transaction Record - Component Execution

The 'C' transaction record generated by the component modifications will be used by this logic statement to build or update the fixed set of the statistics file. The logic statement name is in column one. A report 'STAT' exists in the file.

```
$ASP,STAT,C,50
$RECID,2,25,C1,A
$COUNT,26,31
$DATE,35,40
        POOL
        BNR     NEWREC
        COA     IDATE,'b'
        BEQ     RESET
        ADD     $COUNT,CNTEX,CNTEX
        MAL     $DATE,LDATE
        PRT     'LDATE AND COUNT FIELD UPDATED'
        HLT
NEWREC  MAL     'NEW RECORD GENERATED ',W1/23
        MAL     $RECID,W24/47
        PRT     W1/47
UPDATE  MAL     $DATE,LDATE
        MAL     $DATE,IDATE
        MNU     $COUNT,CNTEX
        HLT
RESET   PRT     'LDATE AND IDATE BLANK,UPDATED'
        BRA     UPDATE
        END
```

If the transaction causes a new record to be generated, the initial date (IDATE) and the most recent date (LDATE) will be set to the date in the transaction $DATE. The count of executions (CNTEX) will be set and the message that a new record has been generated with the record ID will be printed on the auxiliary output printer.

If a new record was not generated, the count of executions field in the file (CNTEX) will be incremented by the count in the transaction record. The latest date will be set to the date in the transaction and a message will be printed.

B.3.2 'S' Transaction Record - UTFLDSCN Utility

The 'S' transaction record generated by the UTFLDSCN utility will be used to build or update periodic set one of the statistics file.

```
         $ASP,STAT,S,50
         $RECID,2,25,C1,A
         $FIELDN,26,33,C2,A
         $SETNO,34,36
         $CNTREP,37,42
         $DATE,43,48
               POOL
               BNR       NEWREC
               POV       $FIELDN,FLDNAM,NEWSS
               COA       $DATE,UDATE
               BEQ       DUPE
               CLR       IDATE   CLEAR FIXED SET FIELDS
               CLR       LDATE
               MNU       0,CNTEX
               PRT       'FIXED SET FIELDS CLEARED'
               PRT       'PERIODIC SET ONE UPDATED'
               PRT       T2/36
               BRA       MOVE
DUPE           PRT       'DUPLICATE TRANSACTION,NO ACTION'
               PRT       T2/36
               HLT
NEWREC         PRT       'NEW FIXED SET GENERATED BY S TRANS'
NEWSS          BSS       FLDNAM
               MCS       $FIELDN,FLDNAM
               PRT       'NEW SUBSET GENERATED'
               PRT       T2/36
MOVE           MAL       $SETNO,SETN
               MNU       $CNTREP,CNTREP
               MAL       $DATE,UDATE
               END
```

If this logic statement causes a new record to be generated, a message will be printed, the subset will be built and the field name will be moved to the subset control field. The fields in the subset will be filled from the transaction record.

If it is not a new record, a check is made for the existence of a subset containing the same field name. If the subset does not exist, a subset will be built. If it does exist, a comparison is made between the date in the subset and in the transaction. If it is the same, a duplicate message is printed and no further action is taken. If the dates are not the same, the fields in the fixed set are cleared and the fields in the periodic set will be updated with the fields in the transaction record.

B.4 RASP Query

The following sample query will retrieve records from the statistics file.

```
TITLE STAT/01.
FILE TESTERF.
IF COMP EQ FM AND CNTEX NE 0.
SORT SOURCE.
```

This query will retrieve all FM logic statements that have had both 'C' and 'S' transactions entered. If no executions have been set in the record, the record will be omitted.

B.5 OP RIT

The following RIT will be used with the previous query to format the output.

```
CREATE RIT ID=STATRIT STORE PERM=OLD
FILE TESTERF
FORMAT PRINT
HEADER1   92 CLASSIF
SPACE 2
HEADER2   95 'TESTERF STATISTICS FILE - FILE ANALYSIS
             STATISTICS CAPABILITY'
```

```
HEADER2  132 QPDATE
SPACE 3
OVERFLOW1   8 FILEN
OVERFLOW1  19 COMP
OVERFLOW1  39 SOURCE
OVERFLOW1  51 '(CONTINUED)'
SPACE 2
OVERFLOW2  54 'FIELD NAME'
OVERFLOW2  64 'SET NO.'
OVERFLOW2  77 'REF. COUNT'
LABEL1      9 'FILE NAME'
LABEL1     21 'COMPONENT'
LABEL1     40 'SOURCE STATEMENT'
LABEL1     54 'EXEC. COUNT'
LABEL1     69 'INITIAL DATE'
LABEL1     85 'LATEST DATE'
LABEL1    101 'DATE EXEC.'
LINE1     FIRST 1
LINE1       8 FILEN
LINE1      19 COMP
LINE1      39 SOURCE
LINE1      52 CNTEX
LINE1      67 IDATE
LINE1      84 LDATE
LINE1     100 UDATE
SPACE 2
LABEL2     54 'FIELD NAME'
LABEL2     64 'SET NO.'
LABEL2     77 'REF. COUNT'
LINE2      53 FLDNAM
LINE2      62 SETN
LINE2      75 CNTREF
EJECT BETWEEN RECORDS IF SOURCE COMPLETE
TRAILER1   92 CLASSIF
TRAILER1  126 'PAGE'
TRAILER1  132 PAGENO
END
SOURCE RETRIEVAL
PUBLISH RITID=STATRIT ANSID=0001 CLASS=UNCLASSIFIED
```

DISTRIBUTION

CCTC CODES	COPIES
C124 (Reference)	2
C124 (Record Copy)	1
C240 (COR for CSC)	20
C300	1
C310	1
C311	1
C312	1
C313	1
C314	1
C315	1
C316	1
C317	1
C320 (Training)	5
C321	1
C322	1
C323	1
C324	1
C325	1
C340	5
C341 (Maintenance Contractor)	8
C341 (Stock)	50
C700	1
C702	1
C703	1
C705	1
C710 (Tech Ser Support)	2
C720	1
C730	2

DCA CODES	
C100	1
C110	1
C205	1
C530	1
C600	1

EXTERNAL	COPIES
Director of Administrative Services, Office of the Joint Chiefs of Staff Attn: Chief, Personnel Division, Room 2A944, The Pentagon Washington, D.C. 20301	1
Director for Personnel, J-1, Office of the Joint Chiefs of Staff, Attn: Chief, Data Service Office, Room 1B738C, The Pentagon, Washington, D.C. 20301	1
Director for Operations, J-3, Office of the Joint Chiefs Staff, Attn: Chief, Data Processing Division, Room 2C869, The Pentagon, Washington, D.C. 20301	1
Director for Operations, J-3, Office of the Joint Chiefs of Staff, Attn: P & AD, Room 2B870, The Pentagon, Washington, D.C. 20301	1
Director for Operations, J-3, Office of the Joint Chiefs of Staff, Attn: Deputy Director for Operations (Reconnaisance and Electronic Warfare) Room 2D921, The Pentagon, Washington, D.C. 20301	1
Director for Logistics, J-4, Office of the Joint Chiefs of Staff, Room 2E828, The Pentagon, Washington, D.C. 20301	1
Chief, Studies Analysis and Gaming Agency, Attn: Chief, Force Analysis Branch, Room 1D928A, The Pentagon, Washington, D.C. 20301	1
Automatic Data Processing, Liaison Office, National Military Command Center, Room 2D901A, The Pentagon, Washington, D.C. 20301	1
Automatic Data Processing Division Supreme Headquarters Allied Powers, Europe Attn: SA & P Branch, APO New York 09055	1
Defense Civil Preparedness Agency, Computer Systems Division, Room 3D317, The Pentagon, Washington, D.C. 20301	1
Director, Defense Communications Agency, Office of NLECN System Engineering, Attn: Code 960T, Washington, D.C. 20301	1
Director, Defense Communications Engineering Center, Hybrid Simulation Facility, 1860 Wiehle Avenue, Reston, VA 22070	1

EXTERNAL	COPIES
Commander, Joint Technical Support Activity Attn: Chief, Software Operations Division 1860 Wiehle Avenue, Reston, VA 22070	1
Director, Defense Intelligence Agency Attn: DS - 5C2 Washington, DC 20301	5
Commander-in-Chief, Pacific, Attn: J6331, FPO San Francisco, 96610	1
Commander-in-Chief, Europe, Attn: Chief, EUCOM ADP Systems Office (ECJDP) APO New York 09128	1
Commander-in-Chief, US Army Europe and Seventh Army Attn: ODCS, OPS APO New York 09403	1
Commanding General, US Army Forces Command, Attn: Data Support Division, Building 206, Fort McPherson, GA 30330	1
Commander, Fleet Intelligence Center, Europe, Box 18, Naval Air Station, Jacksonville, FL 32212	1
Commanding Officer, Naval Air Engineering Center, Ground Support Equipment Department, SE 314, Building 76-1, Philadelphia, PA 19112	1
Commanding Officer, Naval Security Group Command, 3801 Nebraska Avenue, NW, Attn: GF22, Washington, DC 20390	1
Commanding Officer, Navy Ships Parts Control Center Attn: Code 712, Mechanicsburg, PA 17055	1
Headquarters, US Marine Corps, Attn: System Design and Programming Section (HC-JSMD-7) Washington, DC 20380	1
Commanding Officer, US Army Forces Command Intelligence Center, Attn: AFIC-ID, Fort Bragg, NC 28307	1
Commander, US Army Foreign Science and Technology Center Attn: AMXST-CS, 220 Seventh Street, NE, Charlottesville, VA 22712	1

EXTERNAL	COPIES
Commanding Officer, US Army Security Agency, Command Data Systems Activity (CDSA) Arlington Hall Station Arlington, VA 22212	1
Commanding Officer, US Army Security Agency Field Station - Augsburg, Attn: IAEADP, APO New York 09458	1
Commander, Fleet Intelligence Center, Atlantic, Attn: DPS, Norfolk, VA 23511	1
Commander, Fleet Intelligence Center, Pacific, Box 1275 FPO San Francisco, 96610	1
Air Force Operations Center, Attn: Systems Division (XOOCSC) Washington, DC 20301	1
Commander, Armed Forces Air Intelligence Training Center, Attn: TINI-MIT, Lowry AFB, CO 80230	1
Commander, Air Force Data Services Center, Attn: Director of System Support, Washington, DC 20330	1
Commander-in-Chief, US Air Forces in Europe, ATTN: ACDI APO New York 09332	1
Commander, USAF Tactical Air Command, Langley AFB, VA 23665	1
Commander, Space and Missile Test Center, Attn: (ROCA) Building 7000, Vandenberg, AFB, CA 93437	1
Naval Air Systems Command, Naval Air Station, Code 13000, Jacksonville, FL 32212	1
Commanding General, US Army Computer Systems Command, Attn: Support Operations Directorate, Fort Belvoir, VA	1
Defense Documentation Center, Cameron Station, Alexandria, VA 22314	12
TOTAL	171

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CSM UM 15-78	2. GOVT ACCESSION NO
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): NMCS Information Processing System, 360 Formatted File System (NIPS 360 FFS) - Users Manual Vol I - Introduction to File Concepts	5. TYPE OF REPORT & PERIOD COVERED
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s): DCA 100-77-C-0065
9. PERFORMING ORGANIZATION NAME AND ADDRESS: International Business Machines, Corp. Rosslyn, Virginia	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: National Military Command System Support Center The Pentagon, Washington, D.C. 20301	12. REPORT DATE: 1 September 1978
	13. NUMBER OF PAGES: 154
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.
This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This volume presents System Concepts and System Use; it shows a sample NIPS 360 FFS Data File, the Glossary of Terms, and a description of the NIPS 360 FFS Data File and File Format Table. The NIPS 360 is the total system composed of the S/360 hardware and S/360 Operating System (OS) used to support NIPS 360 FFS software.

This document supersedes CSM UM 15-74, Volume I.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)