April 1971

Access Control and Retrieval Optimization Functions of the Supervisor for an Extended Data Management Facility

Judith Irene Hirsch
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-71-21.

Access Control and Retrieval Optimization Functions of the Supervisor for an Extended Data Management Facility

Abstract

The purpose of the Supervisor in an Extended Data Management Facility (EDMF) is to direct the Facility's handling of a user's request for service. The Supervisor fulfills its task through the use of five main functions: Access Control, Retrieval Initialization, File Searching, Record Validating and Record Formatting. The major and most important component of the Retrieval Initialization phase is the Retrieval Optimization subfunction. This report is concerned mainly with the design and implementation of the Access Control and Retrieval Optimization functions. Macro instructions are the mechanism through which a user's program can call upon the ECMF. The Authority Item check is the EDMF's security control over file access while the Prime Keyword Search is the method used to optimize the retrieval strategy. The Authority Item check and the Prime Keyword Search are two of the major concepts of the Extended Data Management Facility.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-71-21.

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
Philadelphia, Pennsylvania

TECHNICAL REPORT

ACCESS CONTROL AND RETRIEVAL OPTIMIZATION FUNCTIONS OF THE SUPERVISOR FOR AN EXTENDED DATA MANAGEMENT FACILITY

by

Judith Irene Hirsch

April 1971

Submitted to the
Office of Naval Research
Information Systems Branch
Arlington, Virginia

under
Contract N00014-67-A-0216-0014
Research Project NR 049-153

Moore School Report No. 71-21

ACCESS CONTROL AND RETRIEVAL OPTIMIZATION
FUNCTIONS OF THE SUPERVISOR
FOR AN EXTENDED DATA MANAGEMENT FACILITY

Abstract

The purpose of the Supervisor in an Extended Data Management Facility (EDMF) is to direct the Facility's handling of a user's request for service. The Supervisor fulfills its task through the use of five main functions: Access Control, Retrieval Initialization, File Searching, Record Validating and Record Formatting. The major and most important component of the Retrieval Initialization phase is the Retrieval Optimization subfunction. This report is concerned mainly with the design and implementation of the Access Control and Retrieval Optimization functions. Macro instructions are the mechanism through which a user's program can call upon the EDMF. The Authority Item check is the EDMF's security control over file access while the Prime Keyword Search is the method used to optimize the retrieval strategy. The Authority Item check and the Prime Keyword Search are two of the major concepts of the Extended Data Management Facility.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) The Moore School of Electrical Engineering University of Pennsylvania Phila., Pa. 19104	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

ACCESS CONTROL AND RETRIEVAL OPTIMIZATION FUNCTIONS OF THE SUPERVISOR FOR AN EXTENDED DATA MANAGEMENT FACILITY

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report

5. AUTHOR(S) (First name, middle initial, last name)

Judith I. Hirsch

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1971	118	11

8a. CONTRACT OR GRANT NO. N00014-67-A-0216-0014	9a. ORIGINATOR'S REPORT NUMBER(S) Moore School Report No. 71-21
b. PROJECT NO. NR 049-153	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT
Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Office of Naval Research Information Systems Branch Arlington, Virginia

13. ABSTRACT

The purpose of the Supervisor in an Extended Data Management Facility (EDMF) is to direct the Facility's handling of a user's request for service. The Supervisor fulfills its task through the use of five main functions: Access Control, Retrieval Initialization, File Searching, Record Validating and Record Formatting. The major and most important component of the Retrieval Initialization phase is the Retrieval Optimization subfunction. This report is concerned mainly with the design and implementation of the Access Control and Retrieval Optimization functions. Macro instructions are the mechanism through which a user's program can call upon the EDMF. The Authority Item check is the EDMF's security control over file access while the Prime Keyword Search is the method used to optimize the retrieval strategy. The Authority Item check and the Prime Keyword Search are two of the major concepts of the Extended Data Management Facility

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Access control						
Control blocks						
File directory						
File search						
File status						
Generalized file						
Input parameters						
Optimization functions						
Retrieval						
Storage						
Subroutines						

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

CHAPTER 1

INTRODUCTION

Today, there is a rapid and ever increasing growth in the total volume of information. This huge volume threatens to make the information useless unless ways can be found to manage it. The purpose of the Extended Data Management Facility (EDMF) is to provide a flexible, general purpose, time-shared file management system for the orderly accumulation and dissemination of information [9].

1.1 The Extended Data Management Facility

The Extended Data Management Facility is an extension of the data management system that presently exists at the Moore School on RCA's Spectra 70/46 Time Sharing Operating System (TSOS). The EDMF makes use of the services offered under TSOS, especially the Data Management System's Indexed Sequential Access Method (ISAM), and it also incorporates its own routines into the operating system.

In order to encourage the use of the EDMF, it must be relatively simple to use. The EDMF simplifies for the user the problem of designating those records that he wishes to see. The user does not need to know the actual addresses of the desired records but he merely must express as a logical expression the characteristic contents of the records. The EDMF then takes on the responsibility of determining the actual record addresses and uses these addresses to retrieve the records. The heart of the Facility is the implementation of the generalized file structure and its general retrieval algorithm as suggested by Hsiao and Harary in [8]. For an overall description of the EDMF, the reader is referred to [9].

1.2 The Supervisor of the EDMF

The purpose of the Supervisor in the EDMF is to direct the Facility's handling of a user's request for service. In this capacity, the Supervisor assumes the roles of "doorman", "foreman", "administrator", and "dispatcher". It is at first as a "doorman" who accepts the service requests and initiates their request handling routines. Then as a "foreman", the Supervisor regulates the use of the primitive storage and retrieval routines [6] and system subroutines, and also optimizes the storage and retrieval strategy for a time-sharing environment. In its role as an "administrator", the Supervisor controls the user's access to files and validates the systems output of records to the user. It is also a "dispatcher" who returns the results of the service to the user.

In directing the handling of the user's requests, the Supervisor performs five main functions: Access Control, Retrieval Initialization, File Searching, Record Validating, and Record Formatting. The major and most important component of the Retrieval Initialization phase is the Retrieval Optimization subfunction. The five main functions in combination with each other satisfy the above roles which the Supervisor must assume.

1.3 The Scope of the Report

This report is concerned mainly with the Access Control and Retrieval Initialization Functions of the Supervisor. These functions fulfill the role of "doorman" and partially those of "foreman" and "administrator". Macro instructions are the "doorman's" entrance into the request handling routines. The Prime Keyword search is the "foreman's" method of optimizing the retrieval strategy and the check of the user's Authority Item is the "administrator's" security control over file access.

A discussion of the macro instructions and the user's Authority Item can be found in Chapter Two, the Open Function; while Chapter Three, The Retrieval Initialization Function, contains a discussion of the retrieval strategy.

CHAPTER 2

THE OPEN FUNCTION

2.1 Definitions

Before the Open Function can be discussed, the terms and concepts which are basic to the EDMF must be given precise definitions. The definitions used in this thesis are consistent with those in [7]. However, they will be found to be less formal and more descriptive.

2.1.1 Attribute-Value Pair

The most basic concept which must be defined is that of the attribute-value pair. Let there be two sets, A and V. The elements of A are those terms which are considered as "attributes", and the element of V are those terms which are considered as "values". Let a third set D be the subset of the Cartesian product A x V, whose elements are the ordered pairs of the elements of A and V. A single element of D is called an attribute-value pair, and intuitively it constitutes the basic element of information. Some examples of attributes, values, and attribute-value pairs are shown in Example 1.

2.1.2 Record

A record R is a set of attribute-value pairs which collectively convey some meaningful information. Often these attribute-value pairs are referred to as the fields of the record. An example of R, a subset of the set of all attribute-value pairs, is shown in Example 2. The attribute-value pairs in this record convey to the reader information about a book on the subject of public education.

1a: A set of attributes

A = {author, year, topic, abstract, text}

1b: A set of values

V = {Lieberman, 1960, public education, [the complete abstract of a book], [the complete text of a book]}

1c: A set D of ordered pairs which are attribute-value pairs

$A \times V \rightarrow D$ = {(author, Lieberman), (year, 1960), (topic, public education), (abstract, [the complete abstract of a paper]), (text, [the complete text of a paper])}

Example 1: Examples of attribute, values and attribute-value pairs

R = {(author, Myron Lieberman),
(title, The Future of Public Education),
(topic, public education),
(publisher, University of Chicago Press),
(year, 1960),
(abstract, [the complete abstract of the book]),
(text, [the complete text of the paper])}

Example 2: Record of a book on the subject of public education

2.1.3 Keywords

A record can be characterized by any combination of the attribute-value pairs which are in the record. Due to pragmatic considerations, it would be desirable to have those attribute-value pairs which are short and can be simply expressed, characterize the record. These short attribute-value pairs are called keywords, and will henceforth be denoted symbolically by K_i, $i = 1,2,\ldots n$. Thus we can refer to a record R by referring only to the keywords in R. The record in Example 2 can be characterized by the set of keywords shown in Example 3. In general, the set of keywords of a record R is called an index of the record R and it is usually a proper subset of R.

The index of R = {(author, Myron Lieberman),
(title, The Future of Public Education),
(topic, public education),
(publisher, University of Chicago Press),
(year, 1960)}

Example 3: The keywords characterizing the record in Example 2

At this point we would like to introduce a notational change for the attribute-value pair. Hereafter an attribute-value pair will be written in the following manner:

Attribute = Value

This is the actual notation used in the EDMF for specifying attribute-value pairs.

2.1.4 Keyword Lists

Each record is also characterized by another parameter which is not part of the actual information contained in the record. This unique number is the address of a record, which indicates the whereabouts of the record in the computer storage.

Each keyword K_i in R may have associated with it the address of another record R' which also contains the keyword K_i. Effectively this address in R "points" to R' and for this reason it is called the pointer of R with respect to K_i or the <u>K_i-pointer</u> of R. If the record R' is non-existent then the K_i pointer of R is known as the <u>null pointer</u>. It will be assumed hereafter that every keyword has a pointer associated with it. Thus we see that records containing a common keyword K_i can be linked by these pointers into a chain which is called a <u>K_i-list</u>. Putting it more precisely, a K_i-list is a chain of records, each record containing the keyword K_i, satisfying the following five conditions:

1) Each of the pointers in the K_i-list are distinct.
2) Each non-null pointer is the address of a record in the K_i-list only.
3) There is <u>one</u> record not pointed to by any other record in the K_i-list. This is the <u>beginning</u> of the K_i-list.
4) There is <u>one</u> record which has the null pointer; this is the <u>end</u> of the K_i-list.
5) For every record in the K_i-list at the address a_n $(n > 1)$, there is a sequence of K_i-pointers

$$(a_1, a_2, \ldots, a_n)$$

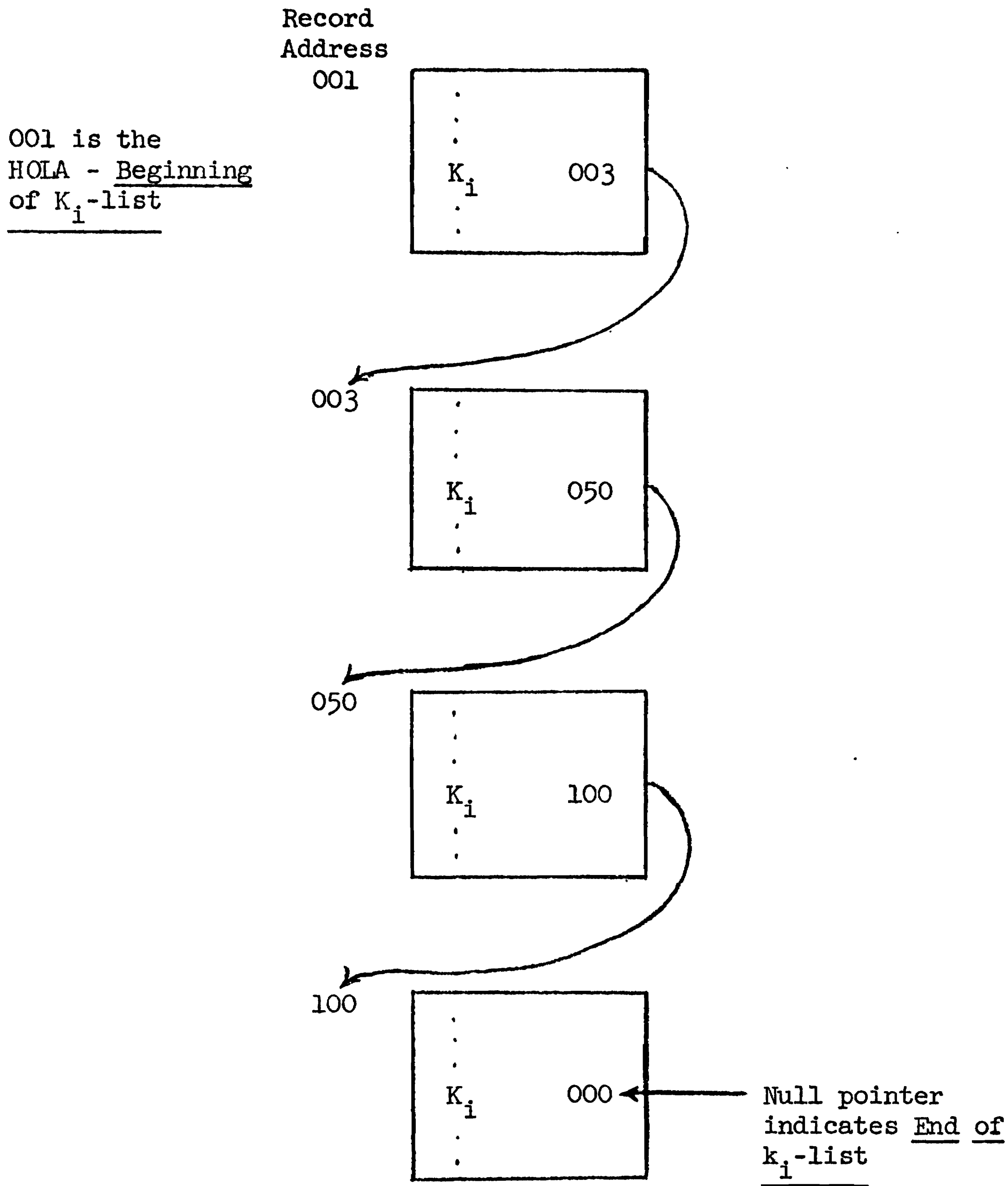


Figure 1: An illustration of a K_i-list

such that:

i) a_1 is the address of the beginning of the K_i-list.

ii) the record at the address a_j contains a K_i-pointer a_{j+1} for $j = 1, 2, \ldots, n-1$.

This means that for a given K_i, a record cannot be in more than one K_i-list. The address of the first record in a K_i-list is known as a Head-of-List Address or HOLA for short, and this term will be used hereafter when referring to the beginning address of any K_i-list. In Figure 1, a typical K_i-list is illustrated, showing the beginning and the end of the list and the pointers which chain the records together.

2.1.5 File and Directory

A file is a set of records which completely contains all the K_i-lists made up of those records. In other words, a file is a set, whose elements are records, which is the union of all the K_i-lists which contain the records. The HOLA's of all the K_i-lists in a given file must be carefully noted and kept separate from the HOLA's of the K_i-lists in another file because the same keyword, but with different meanings, can occur in both these files.

This leads us to the concept of a directory for a file. The directory associated with a file contains the HOLA's of all the K_i-lists in that file. For each keyword K_i used in the file, there is one entry in the directory, the form of the entry being shown in Example 4. More precisely, a directory for a file is a sequence of m such entries where m is the number of different keywords used in the file.

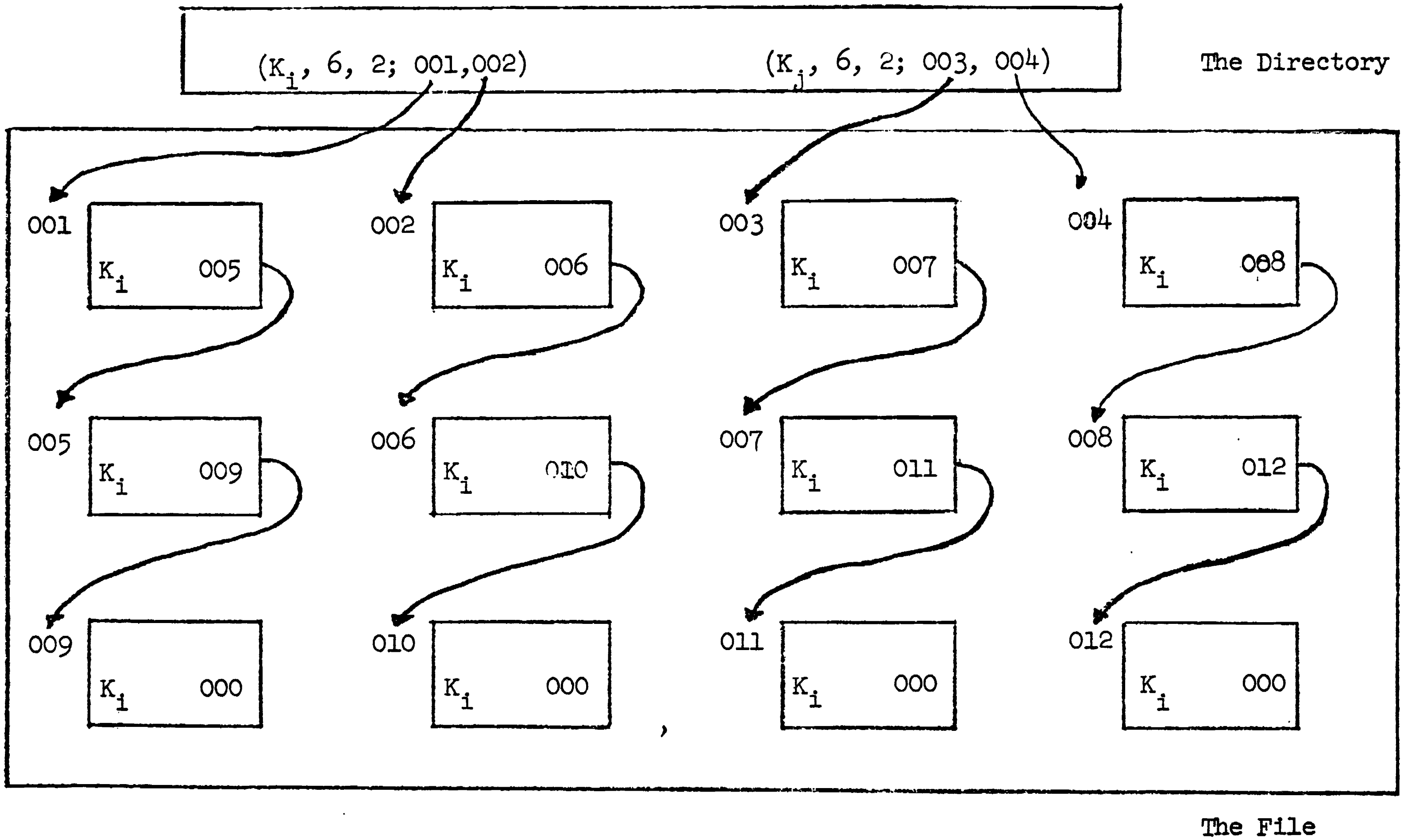


Figure 2: Example of a Generalized File Structure Showing the Logical Relationship Between the Directory Entries and the Keyword Lists

$(K_i, n_i, h_i; a_{i1}, a_{i2}, \ldots, a_{ih_i})$

K_i - the i^{th} keyword in the file F.

n_i - the number of records in F containing the keyword K_i.

h_i - the number of K_i-lists in F.

a_{ij} - the HOLA of the j^{th} K_i-list in F.

Example 4: Format of a directory entry

2.1.6 Generalized File Structure

We can now define a generalized file structure as a file with its directory. This file structure is called generalized because it can be shown that many commonly used file structures such as inverted, index-sequential, and multilist are actually special cases of the generalized file structure [8]. An example of a generalized file structure is shown in Figure 2.

As was evident in the directory format, there may be more than one list corresponding to a particular keyword K_i, but these lists are mutually exclusive because of the definition for lists presented previously. In other words, a record containing the keyword K_i, cannot be in two different K_i-lists.

However, since a record may have more than one keyword, it may be in more than one keyword list. A record containing the keywords K_i and K_j (with $i \neq j$), is a member of one K_i-list and one K_j-list simultaneously. For example, if a record contains both the keywords AUTHOR = LIEBERMAN and YEAR = 1960, then that record would be in both an AUTHOR = LIEBERMAN list and in a YEAR = 1960 list. This is illustrated in Figure 3, where the

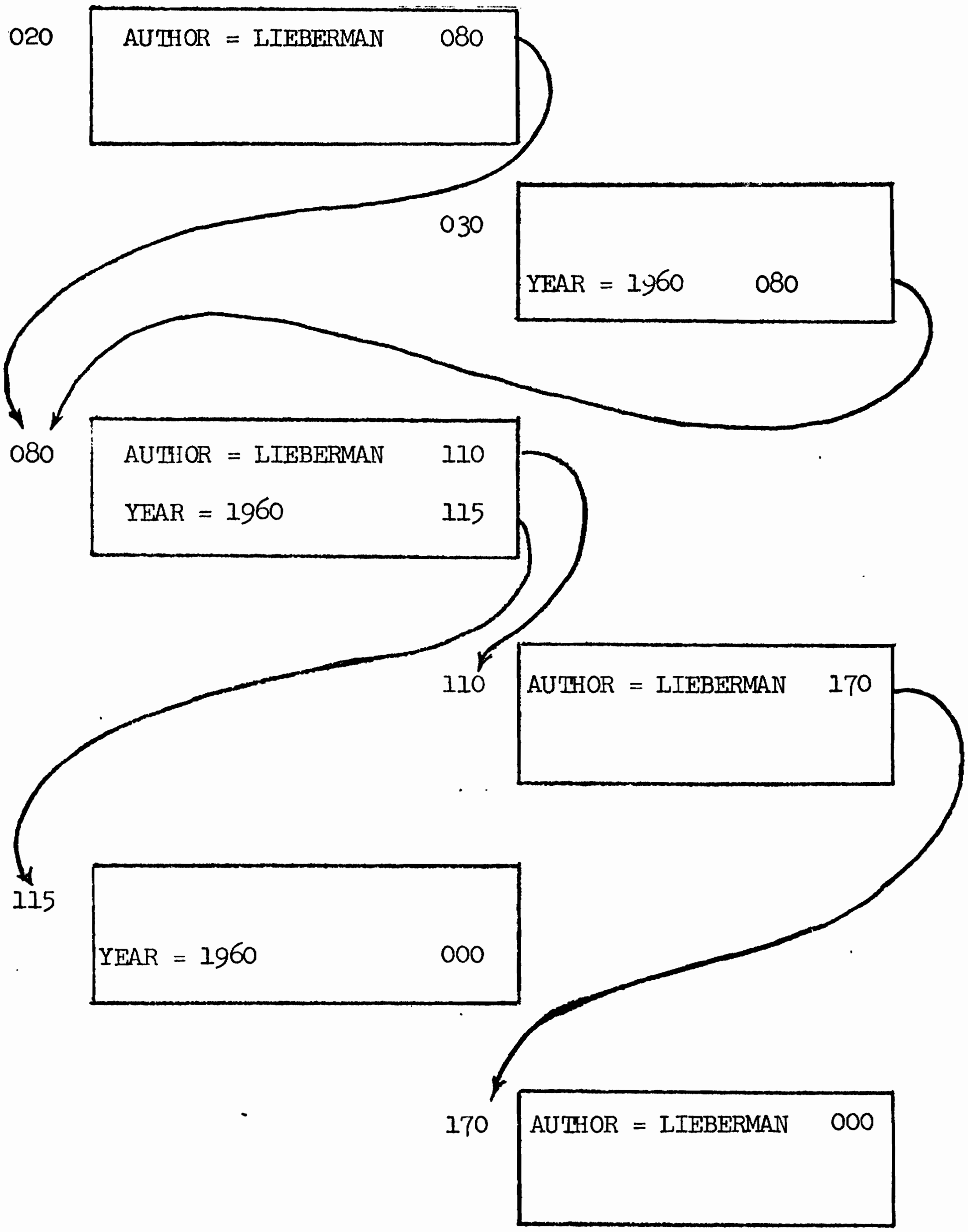


Figure 3: Example of intersecting K_i-list and K_j-list
K_i: AUTHOR = LIEBERMAN
K_j: YEAR = 1960

AUTHOR = LIEBERMAN list consists of records located at the addresses 020, 80, 110, and 170, and the YEAR = 1960 list consists of records located at the addresses 030, 80, 115.

2.1.7 Request Description

When a person accesses a file, rarely does he want to see all of the records in the file. Rather, he usually wants to see only that part of the file which interests him. Such a partition can be accomplished by listing the addresses of the records which he wants. This, however, is cumbersome and requires much research on the user's part to find the addresses of the records in which he is interested. Another way to partition the file would be to describe the records of interest by listing their characterizing keywords in the form of a Boolean expression. This expression is called a user's request description. Using the propositional calculus, any Boolean expression can be uniquely written as a disjunct of conjuncts, known as the Disjunctive Normal Form (DNF). Some typical request descriptions could be

4a: AUTHOR = MYRON LIEBERMAN

4b: AUTHOR = MYRON LIEBERMAN $\wedge$ YEAR = 1960

4c: (AUTHOR = MYRON LIEBERMAN $\wedge$ YEAR = 1960) V (AUTHOR = HIRSCH)

Example 4: Typical request descriptions

All the request descriptions used in the EDMF will be in Disjunctive Normal Form.

A record satisfies a user's request description when all the keywords in at least one of the conjuncts of the request description are in the record. A record containing only the keywords K_1 and K_3 satisfies the request description containing only one conjunct $(K_1 \wedge K_3)$, but does not satisfy $(K_1 \wedge K_2 \wedge K_3)$. The problem of finding in a file, the addresses of records which satisfy a user's request description now lies with the EDMF and not the user.

2.1.8 Entering the EDMF

There are two ways to enter the EDMF - either through a terminal command or through a system macro. This thesis will discuss only the aspect of the system macro. A discussion of the command entrance can be found in [10].

It was decided that the best way for a non-conversational user to enter the EDMF would be through the use of system macros. Each macro instruction generates a group of assembly language statements. One of the statements generated is a supervisor call. The supervisor call instruction (SVC) enables the program to switch from any state to the Interrupt Control State (P_3), i.e., the SVC causes an interrupt. It is in the state P_3, through the use of the interrupt analyzer, that the supervisor decodes the SVC number and determines which routine should handle the interrupt. Statements that accompany the SVC in the macro expansion supply the necessary parameters for the processing of the user's request. Once the system knows how to respond to the interrupt, it switches to state P_2 where interrupt responses are handled. For a diagramatic flow of the above process, see Figure 4.

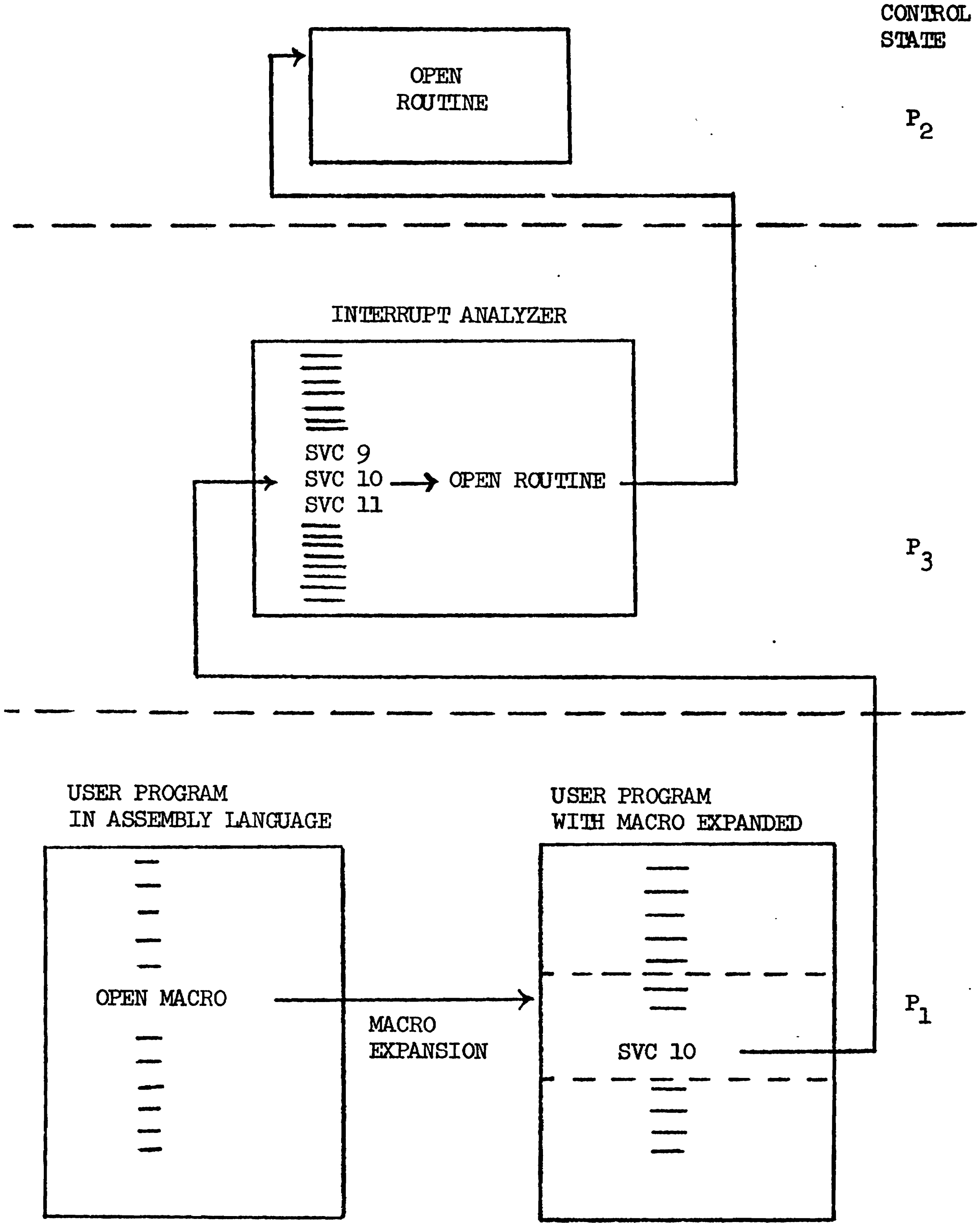


Figure 4: Diagram of Interrupt Handling Process

Macro instructions are extremely useful since they are located in a macro library accessible to all users. Each time a user writes a macro instruction, the associated statements and the SVC are generated and incorporated into his program. The only information the user needs to know in using a macro is the proper way of calling it; all the other steps, the generation of instructions and the SVC, are done by the assembler.

Necessary background material has now been discussed and the remaining part of the chapter will devote itself to the open function.

2.2 Purpose of the Open Function

The purpose of the open function is to check the user's access rights to a specified partition of a file, to set up the necessary control blocks for processing the various service requests, and then to return control to the user. Since the open function assembles the necessary system control blocks for all the available service requests, it must be the first function called upon by the user. There are two routines that implement the open function. They are called OPNPROC and FIFDIRS1. (Appendix B.1 and B.2)

2.3 Access Control

2.3.1 Introduction

In any data management facility, the security and integrity of the records are as important as the ease with which processing occurs. A good system is one in which the security precautions are reliable enough to insure file protection while simultaneously not encumbering any of the processing mechanisms. Insuring the integrity of the files encourages users to store their files in the data management facility, and to enlarge the data base. Ease of using the system will encourage frequent use of this data base, leading to an orderly and efficient utilization of information storage and dissemination.

2.3.2 File Level Check

In the Extended Data Management Facility (EDMF), the protection mechanism operates at three levels corresponding to the logical levels in any file structure. These are the file level, the record level, and the field level. This thesis will discuss only the file level check; a discussion of the other two levels of protection can be found in [4].

In general, and as it presently exists under the TSOS Data Management System (DMS), a file level check is concerned with the security of the file as a whole, and controls any access whatsoever to the file. There are two possible types of file access - either the write, or the read option. If a file has the write option, then a user can update any or all of the existing file records, create new records, and read from the entire file. If, however, the read option is in effect, changes may not be made in the existing file, i.e., the user may only see the records. The present TSOS DMS protection scheme is an "all or none" type of response; that is, either the entire file is accessible to the user, or access is completely denied and the user's request is terminated. The important point here is that access is dependent on the accessibility of the entire file.

But there certainly are cases when a user should have access to certain portions of a file and not be entirely blocked out. For example, let us suppose that we are dealing with a company's file, named PRODUCTS IN PLANNING (PIP), which is a file of records consisting of information on products currently in the planning stages. Possible products could be televisions, radios, computers, etc. Let us also suppose that a user (call him USER A) has the authority to read all the records in this file except those pertaining to computers. Under the current system, access to the file would be denied due to the "all or none" phenomenon. Since USER a is not authorized to reference any of the records pertaining to computers, he is denied access to the entire file.

There are two possible ways to circumvent this problem. One would be to set up a second file which would consist of a subset of the records in the PIP FILE and would contain all the PIP records except those pertaining to computers. Now, USER A would have a file that he could access. But, what if there exists a USER B who is allowed to work with all the records in the PIP file except those pertaining to televisions. Do you set up another file for him? This certainly would amount to a duplication of information and a large waste of storage space.

The other and more efficient way of avoiding the "all or none" restriction is by devising a method which would allow access only to those partitions of a file that a user is authorized to handle, and block him out of those that are restricted to him. It is in this way that the Extended Data Management Facility handles the problem of file protection. In order to put this method into effect, there must be a way of validating a user's authorization and secondly, a way of partitioning a file. First, we will discuss the method used to partition a file.

2.3.3 Partitioning the File

The expression used to partition a file for the open function is the same type of expression that will be used in requesting the retrieval of records. It is a logical expression in Disjunctive Normal Form (DNF) where each element of a conjunct is a keyword of the file. This partitioning method is very flexible since it can be used for any file in

RECORD 1

AUTHOR = BROWN, CHRISTY
TITLE = DOWN ALL THE DAYS
PUBLISHER = STEIN AND DAY
YEAR PUBLISHED = 1970

RECORD 2

AUTHOR = WEITZ, J
TITLE = THE VALUE OF NOTHING
PUBLISHER = STEIN AND DAY
YEAR PUBLISHED = 1970

RECORD 3

AUTHOR = TRAVERS, MILTON
TITLE = EACH OTHER'S VICTIMS
PUBLISHER = SCRIBNER
YEAR PUBLISHED = 1970

RECORD 4

AUTHOR = RAND, AYN
TITLE = WE THE LIVING
PUBLISHER = RANDOM HOUSE, INC.
YEAR PUBLISHED = 1936

RECORD 5

AUTHOR = RAND, AYN
TITLE = ATLAS SHRUGGED
PUBLISHER = RANDOM HOUSE, INC.
YEAR PUBLISHED = 1957

Figure 5: Library Catalogue File

the system. In addition, it does not require that the user know the actual addresses of those records that he is interested in.

For purposes of illustration, let us say we had a library catalogue file with only the five records that appear in Figure 5. One partition of this file would be those records which refer to books that were published by Random House, Inc. in 1936. A DNF description would appear as:

$$(\text{PUBLISHER} = \text{RANDOM HOUSE, INC.} \land \text{YEAR PUBLISHED} = 1936)$$

Only record 4 satisfies this description.

A second partition would be those books published by Stein and Day and those published by Random House, Inc.

$$(\text{PUBLISHER} = \text{STEIN AND DAY} \land \text{PUBLISHER} = \text{RANDOM HOUSE, INC.})$$

Records 1, 2, 4, and 5 satisfy this description.

A third partition might be those books published by Stein and Day in 1970 and books that were published in 1957

$$(\text{PUBLISHER} = \text{STEIN AND DAY} \land \text{YEAR PUBLISHED} = 1970) \lor (\text{YEAR PUBLISHED} = 1957)$$

The satisfactory records here are 1, 2, and 5.

2.3.4 User's Authority Item

In order to validate a user's authorization to access a file, the system must obtain information concerning the user's access rights to that particular file. This information could be stored in a record at the head of each file. This type of security system would be file-oriented.

The EDMF does not take this approach but rather a user-oriented one. The EDMF creates a system file which is known as the Authority Item file. This file consists of a set of records with one record for each user. Each record is an individual user's authority item (UAI). The UAI's contain information pertaining to the user's access rights to the files maintained by the system. Therefore, by examining a specific user's authority item the system can determine to what degree the user is allowed to utilize the existing files.

There are two advantages to this user-oriented type of protection. First of all, since all authorizing information is stored in a system file, it is better protected than if it were stored at the head of a user file. In this case, only the system is allowed to handle the information, thereby making the chance of user intervention very slight. The second advantage is that updating authority information is quite routine. The user's authority information is all stored in one place - the User's Authority Item. Since the system file's internal format is consistent with the internal format of the user files, the same retrieval and updating routines may be used. Additional processing routines for the authority items are unnecessary, consequently making the most efficient use of the EDMF retrieval and updating routines.

Upon the issuance of a call to the open function, the user's authority item is referenced. If access to the requested partition of a file is granted, processing continues with the necessary system control blocks being established; if access is denied, the system returns control to the user with an explanatory message.

2.4 Control Blocks

When access to a file is granted, the open function makes entries into two important system control blocks. One is the Service Status Block and the other is the File Status Block.

2.4.1 Service Status Block

The Service Status Block (SSB) contains status information about every file processed by a user during a TSOS session [9]. It is user-oriented, which means that each user of the system has his own SSB, containing information relevant to only those files which he is using. The SSB is created when a user logs on to TSOS, remains with the user's task throughout its existence in TSOS, and is destroyed when the user logs off.

The purpose of the SSB is to eliminate duplicate retrievals of control information about the user files. It is certainly more worthwhile to use a small amount of storage space to hold the control information, than to spend processing time to re-retrieve it. The problem can best be illustrated as follows. Suppose a user opens a file under the system and then tries to retrieve some information. Due to the structure of the TSOS system, the retrieve request, as far as the system is concerned, is a separate entity from the previous open. This means that the processing routine for the retrieval must be able to check that the requested file has been previously opened. For security reasons, this information is kept in the SSB in privileged system memory. The first file to be opened by a user results in information being stored in the SSB section created during logon. All subsequent file openings cause additional SSB sections (one per each file partition) to be chained to the initial section in a linked list. Thus, each user's SSB can grow as the number of files or their partitions referenced during a session grows. Consequently, there is one SSB section for each file partition that is requested.

One important point to note is that a file need not be opened to have an entry in the user's SSB. (See Chapter 4) What is relevant is whether or not the file's control information is already in storage. This could be the case if the file had been previously opened and then closed. If the file's control information is in storage, then addresses to this information can be found in the SSB. This procedure saves unnecessary retrievals and the waste of duplicate processing time.

2.4.2 File Status Block

The File Status Block (FSB) contains status information about every file that is currently being processed by any user during a TSOS session. It is file-oriented which means that an entry is made in the FSB each time a user opens some partition of a file. This FSB entry is established immediately after the SSB block is created. Each file referenced during a TSOS session has its own linked list whose entries include the following information: the user's Id, the type of open requested, and the partition of the file that has been opened.

The purpose of the FSB is to establish priorities relative to the use of the file. The problem can be illustrated as follows. Let us suppose that two users, USER A and USER B, want to work with FILE 1. USER A wants to read from the file while USER B wants to update it. Let us also assume that USER A issued his open request first. Then the system, by referencing the FSB, could establish that USER A has the priority and permit him to read from the file, while blocking USER B from updating it. Otherwise, USER A could possibly receive erroneous information.

Now let us look at an example where partitioning plays a part. Going back to our library catalogue example (Figure 5), suppose that USER A wants to update that partition of the file which satisfies the DNF description

PUBLISHER = RANDOM HOUSE, INC. Λ YEAR PUBLISHED = 1936

Recall that record 4 is the only member of this partition. Let us also suppose that USER B wants to read from the partition satisfying

(PUBLISHER = STEIN AND DAY) V (PUBLISHER = RANDOM HOUSE, INC.)

The satisfactory records are 1, 2, 4, and 5. Again, USER A issued his open request first and therefore had priority. But, the only requested record that USER A and USER B have in common is record 4. The system references the FSB chain for the library catalogue file to determine the position of USER A's entry. USER A's entry precedes USER B's in the chain and therefore, A has priority. USER A is allowed to update record 4 while USER B is blocked out. But, USER B is allowed to read records 1, 2, and 5.

The individual FSB entries remain in the file list until the user closes the file. It is at this time that the user no longer holds any position in the priority list and therefore his FSB entry is removed.

2.5 Return to User

After both the SSB and FSB have been constructed, the system returns control to the user. If the user entered the system via an SVC call issued from a program, then control is returned to the instruction following the SVC call. If, however, entry was from a command, then control is returned to the Terminal Command Processor which returns control to the user at the terminal. The user, now in control, is free to continue the execution of his program or call upon any other functions of the EDMF.

2.6 The EDMF's OPN Macro

One way of initiating the open function (see Sect. 2.1.8) is through the use of the EDMF macro named OPN. The OPN macro has three required parameters. One is the requested file name. A second is the type of open requested, i.e., either update or read. The third one can be either the actual partitioning description or the address of where this description can be found. For a more detailed discussion of this macro, please see Appendix A.1 .

CHAPTER 3

THE RETRIEVAL INITIALIZATION FUNCTION

3.1 Purpose

The main purpose of the Retrieval Initialization (RI) function is to optimize the retrieval processing and to obtain necessary information for the actual record retrieval. This information includes prime keywords, ISAM keys and Record Format numbers. But, before this information is obtained, the control blocks that were established by the open function must be checked.

3.2 Control Blocks

In order for the processing of the actual retrieval mechanism to start, the user must have previously issued a satisfactory open request. If this was the case, then there is an SSB entry for that partition of the file that he wishes to reference. As the first step in the processing of the RI function, it checks the SSB entries. If the required entry is found, then a TSOS DMS open macro is issued. If the SSB entry does not exist, the processing of the retrieval initialization function is terminated and an explanatory error message is returned to the user.

3.2.1 DMS Open

The TSOS DMS open must precede any call for the primitive storage and retrieval routines. Without the DMS open, the primitive routines cannot access the file. The primitive routines actually perform, through the data management facilities provided by the operating system, the input and output of records for other system components. These routines handle the actual reading and writing of the data records, the manipulation of the files' directories, and the generation and updating of the records and directories of the files.

In processing the retrieval optimization algorithm, the RI function needs to reference the file's directory. In order to use the directory routines, a DMS open must be issued. This brings us to an important point relative to the issuance of the DMS open. There are two possible times that the DMS open macro could be issued: either during the processing of the EDMF's open function or during the RI function. It was decided that the best time would be during the processing of the RI function. This decision was made for the following reason. Once a DMS open is issued, entry into the opened file is blocked to other users until a DMS close is issued. The routines that actually require a DMS open, that is, the primitive routines that handle the requested file's directories and/or records, are not needed until the RI phase of the EDMF. Therefore, the issuance of a DMS open during the EDMF's open function would block the requested file from other users for a longer period of time than necessary.

3.3 Retrieval Optimization

In an attempt to make the retrieval system as efficient as possible, an optimizing retrieval method was needed to minimize the time required to process a retrieval request. The algorithm chosen for the optimization phase was part of the General Retrieval Algorithm as suggested by D. Hsiao and F. Harary in their paper titled "A Formal System for Information Retrieval From Files" [8]. The first step of the algorithm involves the selection of prime keywords from the user's DNF description of requested records.

3.3.1 Prime Keywords

As you recall, each user's DNF request description consists of one or more conjuncts whose elements are keywords of the file. For example, a possible DNF description could be

$$(K_1 \wedge K_2 \wedge K_3) \vee K_4$$

where the K_i are keywords of the file. For the purposes of this example let us say that

K_1: AUTHOR = SMITH

K_2: YEAR = 1964

K_3: TOPIC = MATH

K_4: AUTHOR = COHEN

Our description would then appear as follows:

(AUTHOR = SMITH ∧ YEAR = 1964 ∧ TOPIC = MATH) ∨ (AUTHOR = COHEN)

Associated with each of the keywords in the file's directory is the number of records in the file in which the keyword appears. The <u>prime keyword</u> is defined as that keyword of the conjunct which appears in the least number of records in the file. Going back to our example: let N be the number of records in which a keyword appears, and let the following correspondence be established:

Keyword		N
K_1	AUTHOR = SMITH	10
K_2	YEAR = 1964	15
K_3	TOPIC = MATH	2
K_4	AUTHOR = COHEN	15

For the first conjunct $(K_1 \wedge K_2 \wedge K_3)$, K_3 would be the prime keyword since only 2 records exist in the file that contain TOPIC = MATH. The prime keyword for the second conjunct must be K_4 since it is the sole member of the conjunct.

Now, how does the designation of prime keywords relate to optimizing the retrieval? First of all, we only want to retrieve those records that satisfy each conjunct. Since a record can only satisfy a conjunct by containing every keyword in the conjunct, all satisfactory records must contain the prime keyword. Thus searching the file using the prime keyword, i.e., actually retrieving the least number of records that could possibly satisfy the expression, minimizes the costly time of actual retrieval and thereby results in an optimum retrieval scheme.

The selection of the prime keywords is accomplished in a routine called RETRIEVE. The RETRIEVE routine also picks up the ISAM keys.

3.4 ISAM Keys

In order for the primitive routines to actually retrieve records, they must know the locations of the requested records. The address of the record location depends on the type of access method used to store the records. The EDMF utilizes RCA's TSOS Data Management System Indexed Sequential Access Method (ISAM) for device level input/output. In this access method, each record of a file is assigned a key, a number from 0 to 99,999,999. This number allows one to refer to a record by a logical address (its ISAM key) instead of a physical disk address [6].

Once the prime keyword for a conjunct is established, the RETRIEVE routine must pick up the corresponding ISAM keys for the actual record retrieval. Again, the RETRIEVE routine returns to the directory. Associated with each keyword in the directory are the head of list addresses (HOLA). These head of list addresses are ISAM keys whose records contain that keyword [3]. The RETRIEVE routine then makes a list of all HOLA's that correspond to the prime keywords of the description. Once this is finished, corresponding record format numbers must be established.

3.5 Record Format Numbers

One of the major design criteria used in determining the form of the EDMF records and their control information is as follows. As much information as possible should be removed from the record and stored as file control information. This prevents duplication of information appearing in many records, thus making files smaller. In other words, general structural information is centralized into one file control block rather than decentralized in the individual records.

When records are collected into a file, the usual case is that all records have similar attributes, because they contain the same type of information. For example, all records in a file of library books are likely to contain the attribute "Author". Thus it is reasonable to expect that there are only a limited number of different attributes in a file. In order to save space in the file, the attributes are removed from the records and placed in a file control block called the Record Format Block (RFB). Associated with each attribute in the RFB is a format number. It is this format number and not the entire attribute that is stored in the record [9]. A detailed specification of the RFB can be found in Appendix C.3 .

After a record has been retrieved from disk, it is necessary for the record validating function [4] to determine if it satisfies the user's description. In order to do this, it must check to see if all the keywords of a conjunct can be found in the record. Since only the format numbers and not the actual attributes are stored in the record, it is necessary to determine the corresponding format numbers before the record validating function can operate. The program that performs this service for the RI function is called FORPROG. It checks the attributes in the user's request description against those in the RFB and then makes a list of corresponding format numbers.

3.6 Control Passed to the File Searching Function

Once the lists of prime keywords, ISAM keys, and Record Format numbers are established, the work of the Retrieval Initialization function is finished. The lists and supervisor control is then passed to the File Searching Function [4]. After the File Searching, Record Validating and Record Formatting functions [4] have completely processed the request, the system initiates a DMS close macro. The file can now be actively accessed by other users subject to the priorities established in the File Status Blocks.

3.7 The EDMF's RETR Macro

One entrance to the Retrieval Initialization function is through the use of the EDMF's RETR macro. This macro has six possible parameters. Of these six parameters at least three and not more than five may appear in one macro call. Two of the required parameters are the file name and the output specification. The third required parameter can be either the user's retrieval request description or the address where this description can be found. The fourth parameter, which is optional, is the maximum number of satisfactory records that the user wants retrieved. If this parameter is omitted, all the records satisfying the request description will be outputted to the user. The fifth parameter would be a label. For a more detailed discussion of the RETR macro, see Appendix A.2.

CHAPTER 4

THE CLOSE FUNCTION

4.1 Purpose

The purpose of the close function is to remove a user's priority hold over a specified partition of a file. A user initiates the EDMF's close function when he no longer desires to work with the partition of a file that he had previously opened. The close function makes necessary changes in the control blocks, the SSB and FSB, to indicate that the user has finished all processing of the specified partition of the file. Once this has been done, the user no longer has access to the partition. If he wishes to work with it again, he must re-initiate the EDMF open function. The close function is therefore the last EDMF function that a user would call upon. The routine that implements the close function is called CLSEPROC. (Appendix B.6)

4.2 Control Blocks

During the processing of the open function, a Service Status Block and a File Status Block were created (see Section 2.4). The FSB entry established for the user a position in a priority list relative to the use of the specified file partition. Now that the user has finished working with that partition, he should not maintain his position in the priority list. He no longer has the right to block out other users from accessing the records of the partition. Therefore, the system removes his FSB entry from the priority list and also indicates in the corresponding SSB entry that the EDMF close function has been referenced and that the partition is not open for his use.

4.3 Return to User

After both the FSB and SSB have been updated, the system returns control to the user. The user is now free to continue processing any other files that he had opened, initiate the EDMF open function for another file partition or terminate his session.

4.4 The EDMF's CLSE Macro

One entrance to the close function is through the use of the EDMF's CLSE macro. This macro has three possible parameters. Of these three parameters at least one, and not more than two, may appear in one macro call. The required parameter is the file name. The optional one can be either the actual partitioning description or the address of where this description can be found. If the optional parameter is omitted, the system assumes that the user wants to close out all the partitions of the specified file that he had opened. Otherwise, only the specified partition is closed. For a more detailed discussion of the CLSE macro, see Appendix A.3 .

CHAPTER 5

SUMMARY

The Extended Data Management Facility (EDMF) was implemented to provide a general purpose data management system for the orderly accumulation and dissemination of information. The EDMF utilizes a generalized file structure and an efficient retrieval algorithm for efficient data management.

It was the purpose of this thesis to discuss a portion of the Supervisor's task in the EDMF. The task is to direct the Facility's handling of a user's request and by so doing, the Supervisor assumes the oles of "doorman", "foreman", "administrator", and "dispatcher". In order for the Supervisor to fulfill its task and satisfy its roles, it performs five main functions: Access Control, Retrieval Initialization, File Searching, Record Validating, and Record Formatting. The last three functions, File Searching, Record Validating and Record Formatting, are the functions which partially fulfill the roles of "foreman", "administrator" and "dispatcher". They are discussed in detail in [4]. This thesis has discussed the Access Control and Retrieval Initialization Functions with special emphasis on the Retrieval Optimization subfunction.

These functions fulfill the role of "doorman" and partially those of "foreman" and "administrator". As you remember, macro instructions are used as the "doorman's" entrance into the request handling routines. The Prime Keyword search (Retrieval Optimization subfunction) of the user's DNF Boolean request expression is the "foreman's" method of optimizing the retrieval strategy. The "administrator's" role is fulfilled by the Access Control function. It maintains the security control over file access by checking the user's authority item before processing his request.

BIBLIOGRAPHY

1. Chen, T., et al., "An Interim Report on the Implementation of the Integrated Facility," Project Report, The Moore School of Electrical Engineering, University of Pennsylvania, April, 1970.
2. Corwin, B., et al., "An Integrated Information Storage, Retrieval and Dissemination Facility," Project Report, The Moore School of Electrical Engineering, University of Pennsylvania, June, 1969.
3. Desiato, B., "Directory Constructing and Decoding in a Generalized File Structure," M.Sc. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, work in progress.
4. Ets, A. R., "The File Searching, Record Validating and Record Formatting Functions of the Supervisor for an Extended Data Management Facility," M.Sc. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, August, 1970.
5. Gana, J., "A Command and Query Language Assembler for an Extended Data Management System," M.Sc. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, work in progress.
6. Horton, M., "Reading, Writing, Creating and Updating Records and Files in a Generalized File Structure," M.Sc. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, work in progress.
7. Hsiao, D. K., "A File System for a Problem Solving Facility," Ph.D. Dissertation, The Moore School of Electrical Engineering, University of Pennsylvania, May 1968.
8. Hsiao, D. K. and Harary, F., "A Formal System for Information Retrieval From Files," Communications of the ACM, Vol. 13, No. 2, February, 1970.

9. Manola, F., "An Extended Data Management Facility for a General Purpose Time Sharing System," M.Sc. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, work in progress.

10. McDonald, J., "A Command and Query Language Interpreter for an Extended Data Management System," M.Sc. Thesis, The Moore School of Electrical Engineering, University of Pennsylvania, August, 1970.

11. Wexelblat, R., "The Development and Mechanization of a Problem Solving Facility," Ph.D. Dissertation, The Moore School of Electrical Engineering, University of Pennsylvania, December, 1965.

APPENDIX A

MACROS

A.1 Open Macro

Name: OPN

Type: Keyword

Four possible keywords - maximum of three permissable at one time - minimum of two required.

Required

1) FILENAM - name of the file (up to 54 characters)

2)*(a) DESCRIP - the actual partitioning logical expression in DNF form (up to 127 characters, due to the system's restriction on the length of parameters). Single quotes must enclose the expression and any internal quotes or ampersands must be doubled. See the examples.

*(b) DESADDR - this parameter is mnemonic for description address and it must be used when the desired DNF partitioning expression is longer than 127 characters. This necessitates the placement of the logical expression in an area external to the macro and it is referenced by a symbolic address.

Optional

1) TYPE - the type of open requested

(a) READ - can only read from the file. Default case.

(b) UPDATE - can read and write to the file.

Examples of Macro Calls

```
1)         OPN    FILENAM=$HORTON MULTIES3,TYPE=READ,DESCRIP='AUTHOR=
                  BENNET'
2)         OPN    FILENAM=MULTIES3,TYPE=UPDATE,DESADDR=LOGEXP1
                          .
                          .
                          .
                          .
   LOGEXP1  DC    C'MONTH=MAY && YEAR=1965 ''OR'' KEY PHRASES=INFORMATION
                  STORAGE AND RETRIEVAL && PUBLISHER=THE MOORE SCHOOL
                  OF ELECTRICAL ENGINEERING OF THE UNIVERSITY OF
                  PENNSYLVANIA'
```

Note: * - Only one of these may be used in one macro call.

A.1.1 Generated Parameter List

The OPN macro generates a parameter list whose address is placed in Register 1 and which is passed on to a handling routine via an SVC call. The generated parameter list has the following format:

Bytes	Content
0 - 1	Length of file name
2 - 55	File name (left justified with spaces)
56	Code for type of open X'42' -- Read X'43' -- Update
57 - 59	Address of partitioning logical expression

60 - 63	Length of partitioning logical expression
64 - 190	Partitioning logical expression if included in macro
191	Code for presence of partitioning description X'00' -- No description X'FF' -- Description present

A.2 Retrieval Macro

Name: RETR

Type: Keyword

Six possible keywords - maximum of five permissable at one time - minimum of three required.

Required

1) FILENAM - name of the file (up to 54 characters)

2) OUTSPEC - output specification (up to 10 characters)

   (a) CORE - output is in special core format [4] in core to be used by program

   (b) COUNT - the system returns with the number of satisfactory records and not the actual records

   (c) PRINT - output is sent to the printer

   (d) TTY - output sent to teletype. Default case.

3) *(a) DESCRIP - the actual partitioning logical expression in DNF form (up to 127 characters, due to the system's restriction on the length of parameters). Single quotes must enclose the expression and any internal quotes or ampersands must be doubled. See the examples.

   *(b) DESADDR - this parameter is mnemonic for description address and it must be used when the desired DNF partitioning expression is longer than 127

characters. This necessitates the placement of the logical expression in an area external to the macro and it is referenced by a symbolic address.

Optional

1) RECNO - the number of desired records satisfying the description. If this parameter is omitted, all the records satisfying the request will be presented to the user.

2) LABEL - name associated with RETR macro will be used in a CONTINUE [9].

Examples of Macro Calls

```
1)        RETR   FILENAM=MULTIES1,RECNO=10,DESCRIP='AUTHOR=SMITH &&
                 YEAR=1964 ''OR'' TOPIC=LISP',OUTSPEC=PRINT

2)        RETR   FILENAM=MULTIES3,OUTSPEC=CORE,DESADDR=LOGEXP2,LABEL=
                 AGAIN

3)        RETR   FILENAM=MULTIES1,OUTSPEC=COUNT,DESADDR=LOGEXP2
                                  .
                                  .
                                  .
                                  .
  LOGEXP2 DC     C'AUTHOR=MANOLA && YEAR=1970 && TOPIC=INFORMATION
                 STORAGE AND RETRIEVAL && PUBLISHER=THE MOORE SCHOOL
                 OF ELECTRICAL ENGINEERING OF THE UNIVERSITY OF
                 PENNSYLVANIA ''OR'' TOPIC=MATHEMATICS'
```

Note: *- Only one of these may be used in one macro call.

A.2.1 Generated Parameter List

The RETR macro generates a parameter list whose address is placed in Register 1 and which is passed on to a handling routine via an SVC call. The generated parameter list has the following format:

Bytes	Content
0 - 1	Number of requested records to be retrieved X'0000' -- All records. Default case.
2 - 6	Output specification. CORE, COUNT, PRINT or TTY.
7 - 11	Label
12 - 13	Length of file name
14 - 67	File name (left justified with spaces)
68	Function code X'22' -- Retrieval code
69 - 71	Address of logical expression
72 - 75	Length of logical expression
76 - 202	Logical expression if included in the macro

A.3 Close Macro

Name: CLSE

Type: Keyword

Three possible keywords - maximum of two permissable at one time - one required.

Required

1) FILENAM - name of the file (up to 54 characters)

Optional

1)*(a) DESCRIP - the actual partitioning logical expression in DNF form (up to 127 characters, due to the system's restriction on the length of parameters). Single quotes must enclose the expression and any internal quotes or ampersands must be doubled. See the examples.

*(b) DESADDR - this parameter is mnemonic for description address and it must be used when the desired DNF partitioning expression is longer than 127 characters. This necessitates the placement of the logical expression in an area external to the macro and it is referenced by a symbolic address.

Examples of Macro Calls

```
1)        CLSE  FILENAM=$HORTON.MULTIES3

2)        CLSE  FILENAM=$HORTON.MULTIES3,DESCRIP='AUTHOR=BENNET'

3)        CLSE  FILENAM=MULTIES3,DESADDR=LOGEXP3
                          .
                          .
                          .
                          .
   LOGEXP3  DC  C'MONTH=MAY && YEAR=1965 ''OR'' KEY PHRASES=INFORMATION
                STORAGE AND RETRIEVAL && PUBLISHER=THE MOORE SCHOOL
                OF ELECTRICAL ENGINEERING OF THE UNIVERSITY OF
                PENNSYLVANIA'
```

Note: * - Only one of these may be used in one macro call.

A.3.1 Generated Parameter List

The CLSE macro generates a parameter list whose address is placed in Register 1 and which is passed on to a handling routine via an SVC call. The generated parameter list has the following format:

Bytes	Content
0 - 1	Length of file name
2 - 55	File name (left justified with spaces)
56	Code for type of close X'48' -- Close all partitions of the file X'49' -- Close only the specified partition
57 - 59	Address of partitioning logical expression
60 - 63	Length of partitioning logical expression
64 - 109	Partitioning logical expression if included in the macro

APPENDIX B

ROUTINES

B.1 Routine OPNPROC

The OPNPROC routine is the first of two routines that implement the Open Function of the EDMF. This routine checks the user's access rights to the specified partition of a file and sets up the SSB and FSB control blocks.

B.1.1 Entry Points

OPNPROC has three entry points. The entrance via an SVC call is at OPNPROC while the command entrance is at COMDOPN. The FIFBLOCK entrance is used when only the FCB for the File of Files (FIF) is needed.

B.1.2 Exit Points

There is only one exit point for this routine. It begins at BRETURN where control is returned to the calling program.

B.1.3 External Subroutine Calls

There are eight external subroutines that may be called upon by OPNPROC. One is AIRETR which retrieves the user's authority item. A second is AUTHCHK which checks the user's access rights to the specified partition of a file. A third is to the location ESQCAT to obtain the task number. A fourth external subroutine is FIFDIRS1. FIFDIRS1 is used to retrieve the File Information Block (FIB) for the specified file. The following three are entry points in the SSBOPTR routine [9]. SSBACQR is used to obtain the SSB chain for a specified user. SSBLOGON is used to establish the SSB chain if it has not already been done and SSBGTNU is used to obtain a new SSB block to link to the user's SSB chain.

The eighth external subroutine is FSBOPTR. This subroutine is used to establish the FSB entries. The DSECTS that are associated with the SSBOPTR and FSBOPTR routines are the following:

Name	Bytes	Content
SSB	DSECT	
SSBHDR	0 - 7	SSB Header
SSBUAI	0 - 3	Address of User's Authority Item
SSBFIF	4 - 7	Address of FCB for File Information File
SSBTXT	8 - 91	SSB text
SSBFNAM	8 - 63	2 bytes - length of file name 54 bytes - file name
SSBCL	64 - 67	Control Information
	64	Type of request
	65	Indicator - EDMF open
	66 - 67	Unused
SSBFIB	68 - 71	Address of File Information Block (FIB)
SSBFCB	72 - 75	Address of File Control Block (FCB)
SSBDTBIN	76	Open description indicator
SSBDTAB	77 - 79	Address of user description block
SSBCREC	80 - 83	Address of Core Format of the record
SSBFSB	84 - 87	Address of File Status Block
SSBCTL6	88	Control Information for pointer
SSBPTR	89 - 91	Pointer to next SSB block
FSB	DSECT	
FSBUSRID	0 - 7	User Identification
FSBCL	8 - 11	Control Information
FSBDSADR	12 - 15	Address of user's partitioning description

Name	Bytes	Content
FSBLTBLK	16 - 19	Pointer to previous FSB block in chain
FSBCTRL	20	Control Information
FSBNTBLK	21 - 23	Pointer to next FSB block

B.1.4 Input Parameter List

The address of the input parameter list (PARAMOP) must be in Register 1 and Register 13 must contain the address of the calling routine's save area.

Name	Bytes	Content
PARAMOP	DSECT	
FLNMLN	0 - 1	Length of file name
FLNAME	2 - 55	File name (left justified with spaces)
FUNCODE	56	Code for type of open requested
LOGEXPAD	57 - 59	Address of partitioning logical expression
LNLOGEXP	60 - 63	Length of partitioning logical expression
LOGEXP	64 - 190	Partitioning logical expression if included in OPN macro
DESCODE	191	Code for presence of partitioning description

B.1.5 Register Conventions

The registers in OPNPROC are assigned in the following manner:

Register	Utilization
0	Not used
1	Address of parameter list given to called subroutine. Miscellaneous use.
2	Miscellaneous use.
3	Base for OPNPROC
4	Miscellaneous use

Register	Utilization
5	Address and base of SSB
6	Counter for number of characters in User Id. Miscellaneous use.
7	Miscellaneous use
8	Address of current SSB block
9	Address and base of SSBTEXT
10	Length of requested file name
11	Address and base of OPNPROC work area
12	Address and base of input parameter list (PARAMOP)
13	Address of OPNPROC save area
14	Return address in OPNPROC
15	Subroutine call address. Error codes.

B.1.6 Internal Work Area

The internal work area (OPENRQ) used by OPNPROC also contains the parameter lists for some of the routines called by OPNPROC. The DPLISTA list is passed to AUTHCHK while PAROPEN is passed to FIFDIRS1. The work area has the following format:

Name	Bytes	Content
OPENRQ	DSECT	
DPARM	0 - 7	Parameter area for error messages
RETAREA	8 - 11	Address of area to return to after calling subroutine to check user's authority
DPLISTA	12 - 95	Parameter list passed to AUTHCHK
DADRAI	12 - 15	Address of User's Authority Item
DADREC	16 - 19	Address of record to be checked
DFNLEN	20 - 21	Length of file name

Name	Bytes	Content
DFILNAM	22 - 75	File name of file whose access is to be checked
DFDADEX	76	Code for presence of partitioning description
DFDADDR	77 - 79	Address of partitioning logical expression
DFDLEN	80 - 83	Length of partitioning logical expression
DSERREQ	84	Code for service request
	85	Code for checking level
DINCTL	86 - 87	Control information about limiting description
DNADDR	88 - 91	Address of internal form of limiting description
DNAKIB	92 - 95	Address of Key Information Buffer (KIB) for limiting description
OPRQMPAR	96 - 99 100 - 103 104 - 107 108 - 111	Parameters for $REQM
OPSAVE	112 - 183	Save area for OPNPROC
ATMODEAR	184 - 187	Address of area for TMODE macro
	188 - 189	Length of area for TMODE macro
TMODEAR	190 - 219	Area for TMODE macro
USERID	220 - 227	User Identification
FCBFIF	228 - 739	Area for File Control Block (FCBO of File of Files (FIF)
FIFKYARG	740 - 747	Parameter in FCB of FIB
PAROPEN	748 - 759	Parameter list passed to FIFDIRS1
AFCBFIF	748 - 751	Address of FCB of FIF
FILEFIB	752 - 755	Address of File Information Block for requested file

Name	Bytes	Content
FILEFCB	756 - 759	Address of FCB for requested file
STACKADR	760 - 763	Address of stack area of SVC
DNOACCES	764 - 765	For re-entrant error message
	766 - 768	For re-entrant error message
DMESAG1	769 - 859	For re-entrant error message
TSKNUM	860	Task number
CKCODE	861	Code for errors
SW1	862	Code - found matching file name
TEMPA	863 - 913	Temporary area
SW2	914	Code for macro entrance

B.1.7 Internal Codes

The various internal codes in the OPNPROC routine are listed below by hexadecimal digits.

CKCODE

Return from AUTHCHK

X'00'	Access granted
X'01'	Access denied

Return from SSBACQR

X'00'	SSB exists but has not been acquired
X'04'	SSB exists and has been acquired
X'08'	SSB does not exist

Return from SSBGTNU

X'10'	REQM error

Return from SSBLOGON

X'00'	SSB exists but has not been acquired

DESCODE (Description code)

	X'00'	Partitioning logical expression not present
	X'FF'	Partitioning logical expression present

DFDADEX

	X'FF'	Code that indicates partitioning logical expression present

FSBCTRL

	X'FF'	Code that indicates good pointer in FSB block

FUNCODE (Function code)

	X'42'	Read type open
	X'43'	Update type open

SSBCL

	X'42'	Read type open
	X'43	Update type open

SSBCL+1

	X'00'	File partition EDMF closed
	X'FF'	File partition EDMF open

SSBCTL6

	X'FF'	Code that indicates good pointer in SSB block

SSBDTBIN

	X'FF'	Code that indicates user description block present

SW1

	X'FF'	Code that indicates matching file name found on SSB

SW2

	X'FF'	Entrance from a macro

B.1.8 Return Codes

All return codes can be found in the right-most byte of Register 15 and they are listed below by hexadecimal digits.

X'00'	Everything O.K.
Otherwise	Error occurred

B.1.9 Flowchart

Figures B.1.a - B.1.d contain the flowchart for the OPNPROC routine.

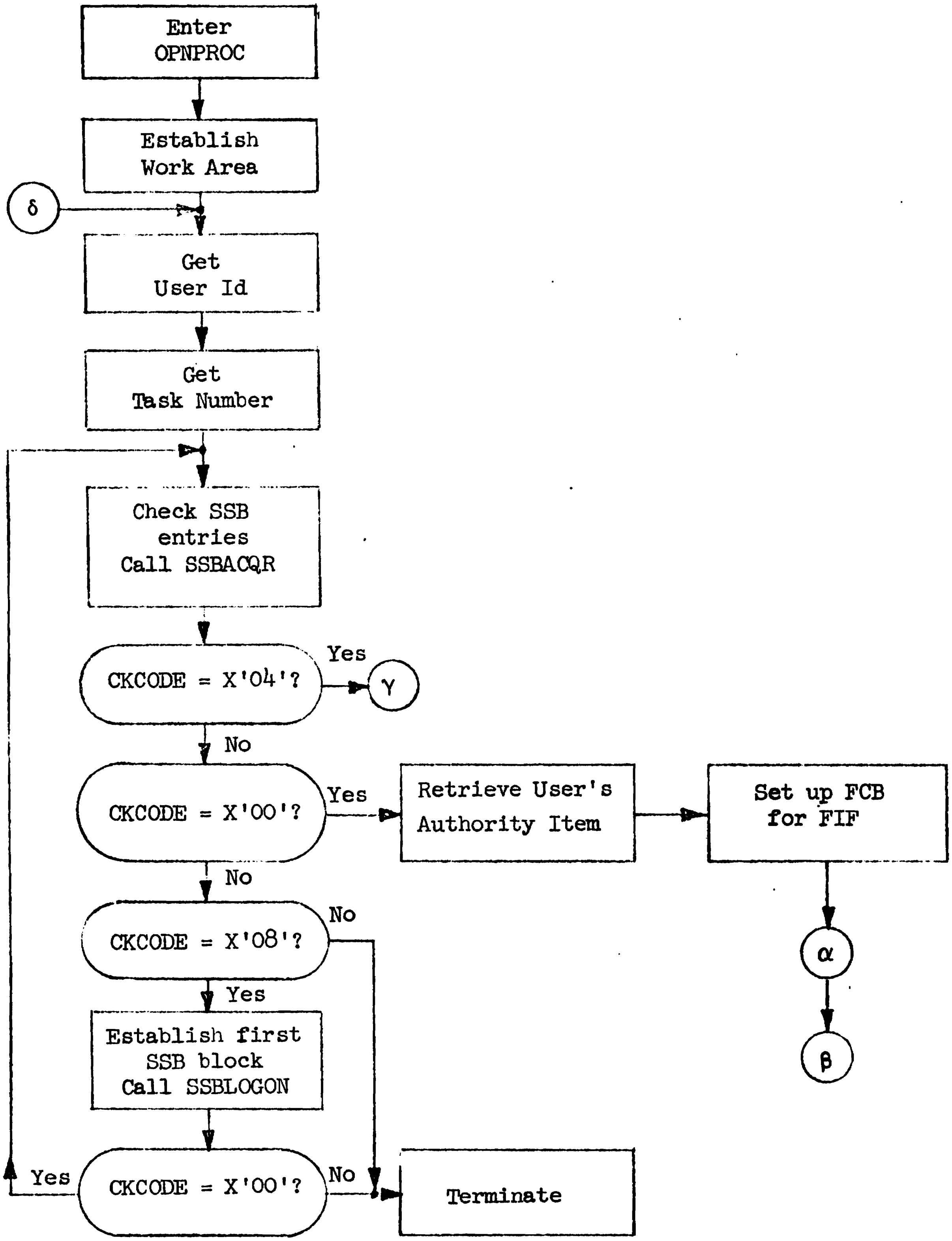


Figure B.1.a: OPNPROC Initialization.
Retrieve SSB Chain.

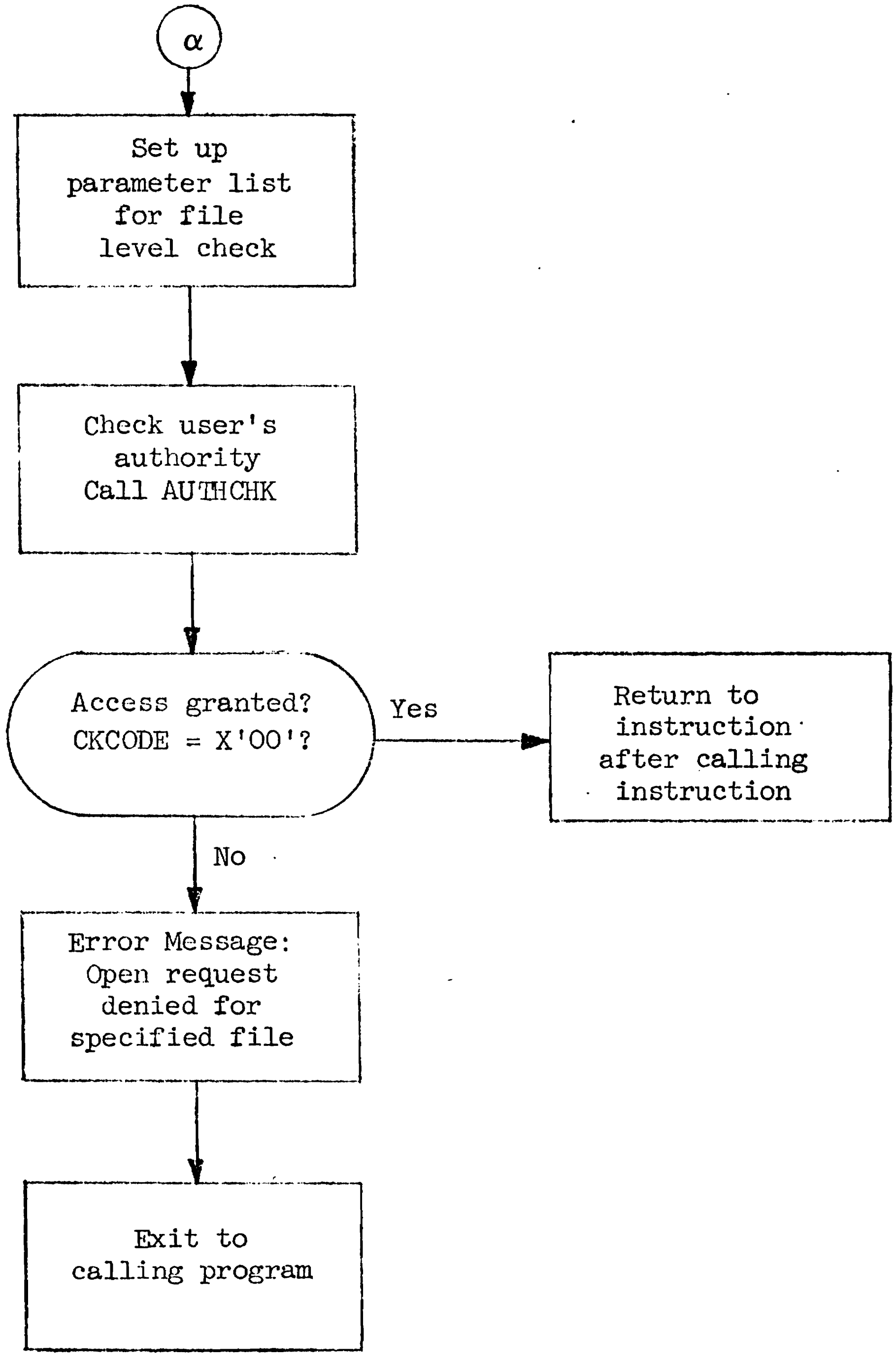


Figure B.1.b: Authority Item Check

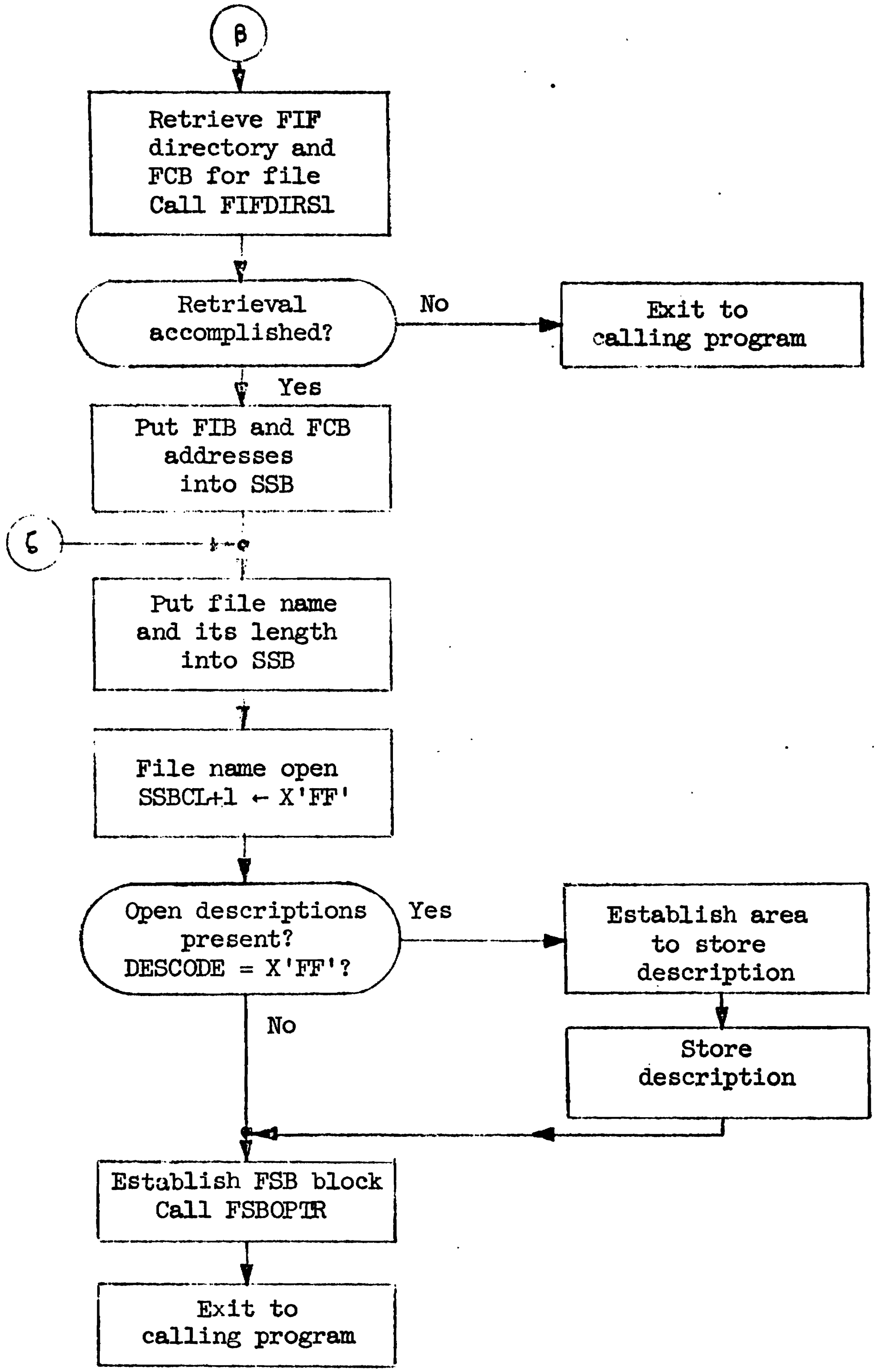


Figure B.1.c: Set up SSB.
Establish FSB.

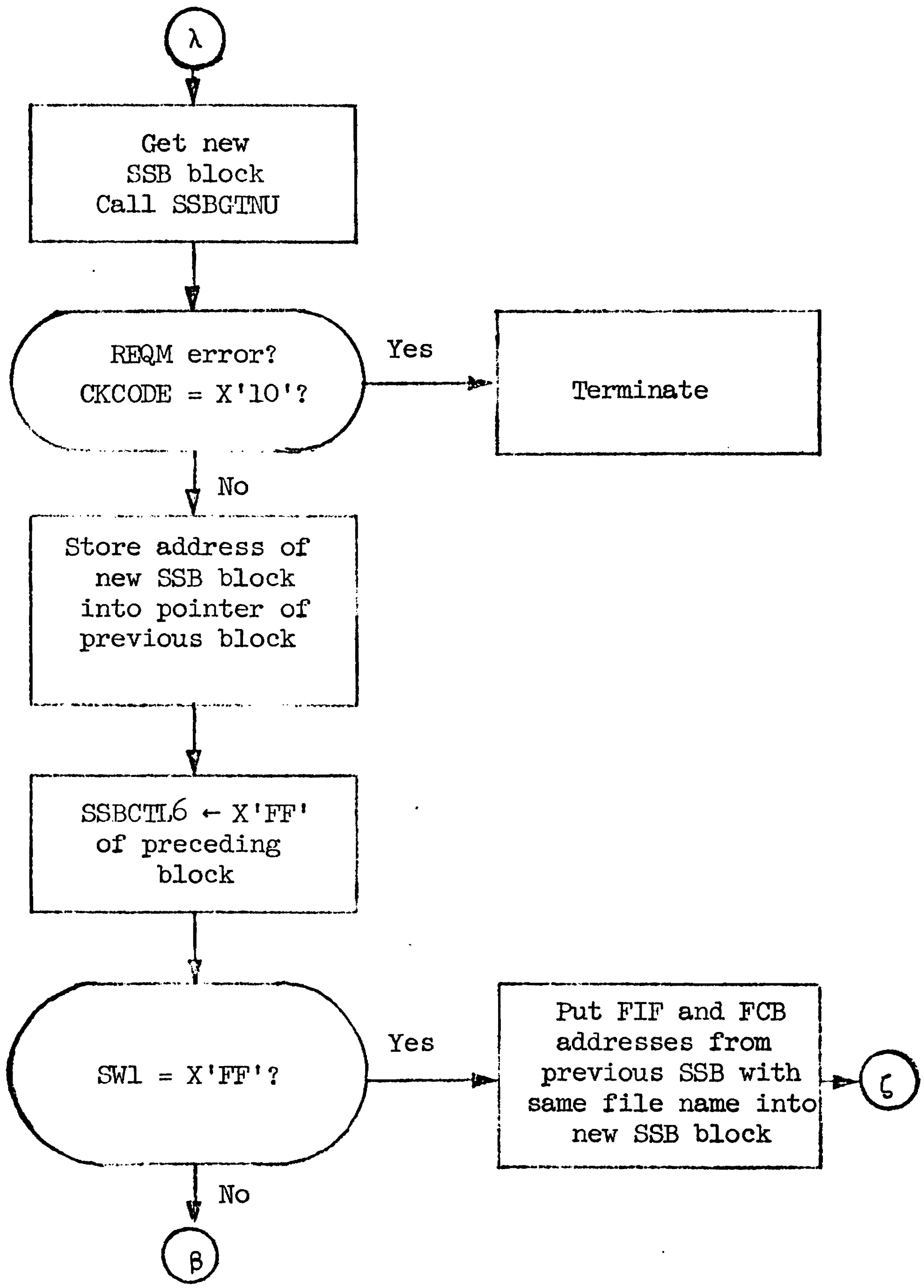


Figure B.1.d: Get New SSB

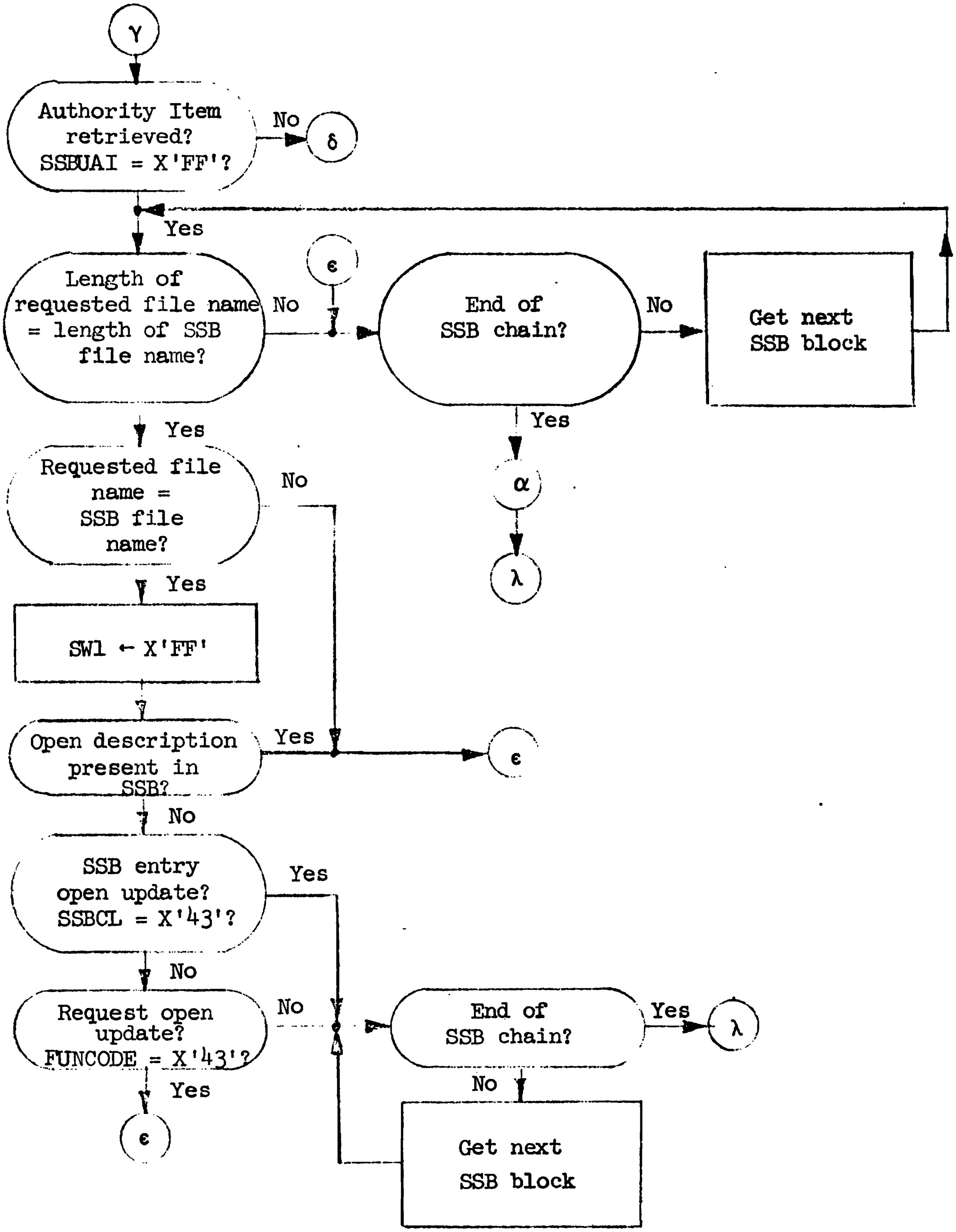


Figure B.1.e: SSB Check

B.2 Routine FIFDIRS1

The FIFDIRS1 routine is the second of two routines that implement the EDMF's Open Function. This routine establishes the File Control Block (FCB) for the File of Files (FIF), searches the FIF directory and retrieves the File Information Block (FIB) for the requested file.

B.2.1 Entry Points

FIFDIRS1 is the only entry point in this routine.

B.2.2 Exit Points

FIFDIRS1 has two exit points. The normal exit begins at OUT2 and the error exit begins at OUT1. In both cases, program control is returned to the calling program.

B.2.3 External Subroutine Calls

RETRREC [6] is the only external subroutine called by FIFDIRS. The first time RETRREC is called it retrieves the FIF directory; the second time, it retrieves the FIB for the requested files. The DSECT's that are associated with the FIF directory and the FIB are the following:

Name	Bytes	Content
DIRFIF	DSECT	
HEADERD	0 - 14	Header
LENGTHD	0 - 2	Length of FIF directory
COUNTD	3 - 4	Count of FIF directory
LKEYD	5 - 9	Lowest key in directory
HKEYD	10 - 14	Highest key in directory
ENTRIES	---	Individual entries
FIB	DSECT	
	0 - 92	Beginning of FIB

Name	Bytes	Content
FCB	93 - 252	File Control Block
RFB	253 -	Record Control Block

B.2.4 Input Parameter Lists

There are two necessary input parameter lists for the FIFDIRS1 routine. The address of the PAROPEN input list must be in Register 1 while the address of the PARAMOP input list must be in Register 12. Register 13 must contain the address of the calling routine's save area.

Name	Bytes	Content
PAROPEN	DSECT	
AFCBFIF	0 - 3	Address of FCB of FIF
FILEFIB	4 - 7	Address of FIB of the requested file
FILEFCB	8 - 11	Address of FCB of the requested file
PARAMOP	DSECT	
FLNMLN	0 - 1	Length of file name
FLNAME	2 - 55	File name (left justified with spaces)
FUNCODE	56	Code for type of open requested
LOGEXPAD	57 - 59	Address of partitioning logical expression
LNLOGEXP	60 - 63	Length of partitioning logical expression
LOGEXP	64 - 190	Partitioning logical expression if included in OPN macro
DESCODE	191	Code for presence of partitioning description

B.2.5 Register Conventions

The registers in FIFDIRS1 are assigned in the following manner:

Register	Utilization
0	Not used
1	Address of parameter list given to called subroutine
2	Length of FIF directory
3	Length of requested file name
4	Address and base of DIRFIF. Address and base of FIB.
5	Base for FIFDIRS1
6	Length-1 of file name in FIF directory. Miscellaneous use.
7	Pointer to entry in FIF directory
8	=H'7'
9	Address of last byte in FIF directory
10	Address and base of input parameter list (PAROPEN)
11	Address and base of FIFDIRS1 work area
12	Address and base of input parameter list (PARAMOP)
13	Address of FIFDIRS1 save area
14	Return address in FIFDIRS1
15	Subroutine call address. Error codes.

B.2.6 Internal Work Area

The internal work area (SUP1) used by the FIFDIRS1 routine also contains the parameter list (PLIST) to be passed to RETRREC [6]. The work area has the following format:

Name	Bytes	Content
SUP1	DSECT	
SAVE1	0 - 71	Save area for FIFDIRS1
WKAREA	72 - 75	Temporary work area
OPPARAM	76 - 83	Parameter area for DMS open
CLPARAM	84 - 91	Parameter area for DMS close
WFCB	92 - 603	File Control Block
KEYARG	604 - 611	Parameter in FCB
PLIST	612 - 627	Parameter area passed to RETRREC
PFCBADDR	612 - 615	Address of FCB
PRECADDR	616 - 619	Address of area to place retrieved record
PISAM	620 - 624	ISAM key for requested record
PLREC	625 - 627	Length of area to place retrieved record

B.2.7 Internal Codes

The various internal codes in the FIFDIRS1 routine are listed below by hexadecimal digits.

DESCODE (Description code)

X'00'	Partitioning logical expression not present
X'FF'	Partitioning logical expression present

FUNCODE (Function code)

X'42'	Read type open
X'43'	Update type open

B.2.8 Return Codes

All return codes can be found in the right-most byte of Register 15 and they are listed below by hexadecimal digits.

X'00'	Everything O.K.
X'04'	Unable to open FIF

X'08'	Unable to retrieve FIF or FIB of requested file
X'0C'	Requested file does not exist in the system
X'0F'	REQM error

B.2.9 Flowchart

Figures B.2.a - B.2.b contain the flowchart for the FIFDIRS1 routine.

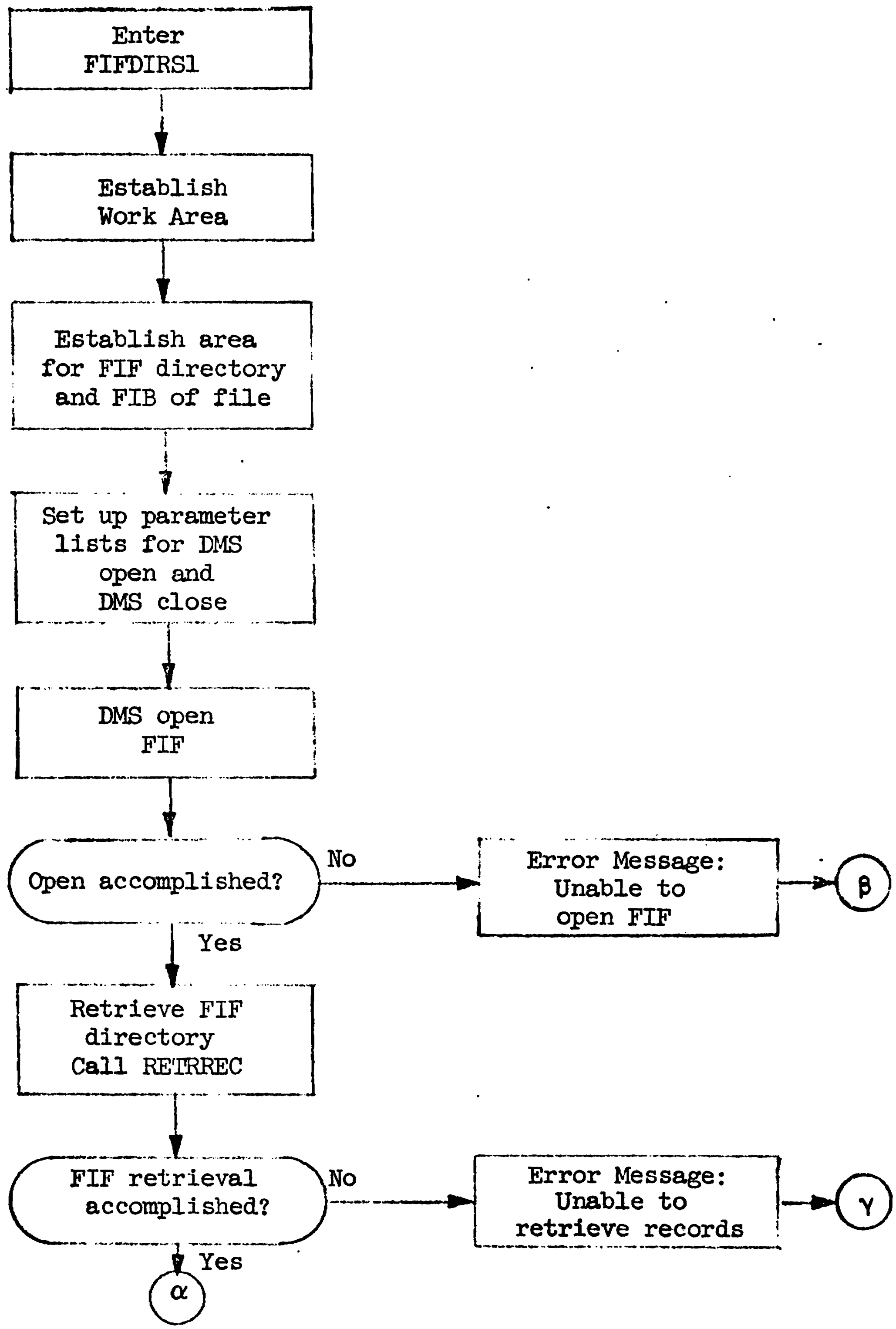


Figure B.2.a: FIFDIRS1 Initialization and FIF Directory Retrieval

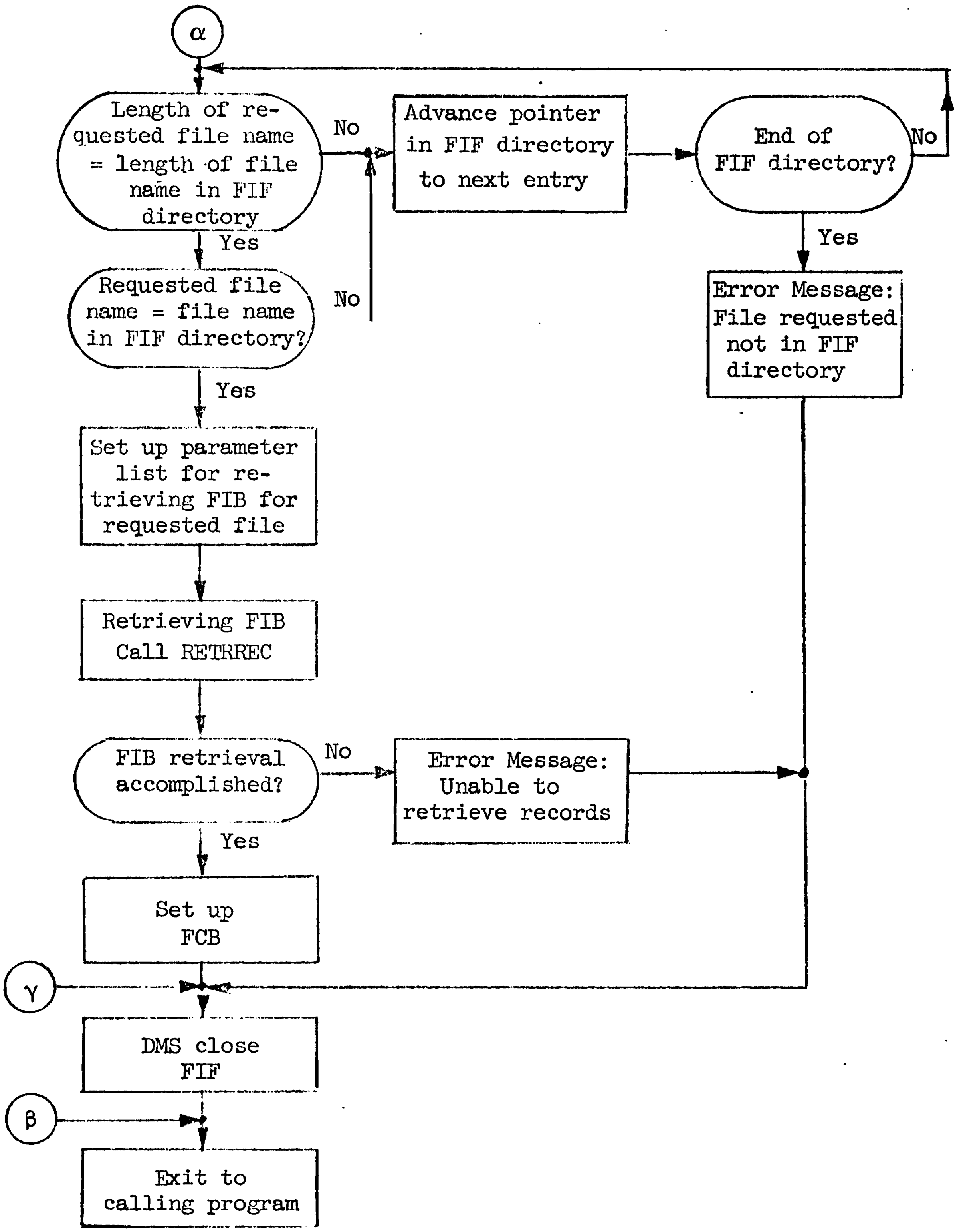


Figure B.2.b: Retrieval of File FIB and FCB

B.3 Routine MACPROC

The MACPROC routine obtains necessary information before the retrieval optimization phase is entered. The main function of this routine is to check if an EDMF open has been issued and if so, issue a DMS open. Also, if entry is non-conversational in nature, the routine obtains the internal form of the user's request description. If entry is conversational, the internal form has already been obtained.

B.3.1 Entry Points

There are three entry points. MACPROC is used when entry is from a user program (non-conversational); COMENTER is the point a conversational user enters. After the EDMF has processed a retrieval, it is necessary to DMS close the specified file. This is accomplished at the CLFILE entry point.

B.3.2 Exit Points

MACPROC has two exit points. One is the normal exit point and the other is used when an error occurs. The normal exit is to the RETRIEVE routine. The error exit is at CHKEXIT.

B.3.3 External Subroutine Calls

Two external subroutines are called by MACPROC. The first is to the location ESQCAT to obtain the task number. The second is to the entry point SSBACQR of the SSBOPTR routine [9]. This is used to obtain the SSB chain for a specific user. The DSECT that is associated with the SSBOPTR routine is the following:

Name	Bytes	Content
SSB	DSECT	
SSBHDR	0 - 7	SSB Header
SSBUAI	0 - 3	Address of User's Authority Item

Name	Bytes	Content
SSBFIF	4 - 7	Address of FCB for File Information File
SSBTXT	8 - 91	SSB text
SSBFNAM	8 - 63	2 bytes - length of file name 54 bytes - file name
SSBCL	64 - 67	Control Information
	64	Type of request
	65	Indicator - EDMF open
	66 - 67	Unused
SSBFIB	68 - 71	Address of File Information Block (FIB)
SSBFCB	72 - 75	Address of File Control Block (FCB)
SSBDTBIN	76	Open description indicator
SSBDTAB	77 - 79	Address of user description block
SSBCREC	80 - 83	Address of Core Format of the record
SSBFSB	84 - 87	Address of File Status Block
SSBCTL6	88	Control Information for pointer
SSBPTR	89 - 91	Pointer to next SSB block

B.3.4 Input Parameter List

There are two possible input parameter lists for the MACPROC routine. MACDS is the input list used when entrance is non-conversational. RPARA is the conversational parameter list and it is also the list that is passed to RETRIEVE. The address of the input parameter list, either MACDS or RPARA, must be in Register 1 and Register 13 must contain the address of the calling routine's save area.

Name	Bytes	Content
RPARA	DSECT	
AFCB	0 - 3	Address of File Control Block (FCB)
RFBA	4 - 7	Address of Record Format Block (RFB)
USRID	8 - 15	User Identification
RECNO	16 - 17	Number of requested records to be retrieved.
OUTSPEC	18 - 27	Output specification
FLNMLN	28 - 29	Length of file name
FLNAME	30 - 83	File name (left justified with spaces)
FUNCODE	84	Code for function requested
CONTROL	85	Part of internal form of user's description
LILEP	86 - 87	Length of DCB and KIB
LDCB	88 - 89	Length of DCB
DCB	--	Actual Description Control Block (DCB)
KIB	--	Actual Key Information Buffer (KIB)

Name	Bytes	Content
MACDS	DSECT	
MRECNO	0 - 1	Number of requested records to be retrieved
MOUTSPEC	2 - 11	Output specification
MFLNMLEN	12 - 13	Length of file name
MFLENAME	14 - 67	File name (left justified with spaces)
MFUNCODE	68	Code for function requested
MADLOGEP	69 - 71	Address of logical expression
MLENLEXP	72 - 75	Length of logical expression
MILOGEXP	76 - 202	Logical expression if included in the macro.

B.3.5 Register Conventions

The registers in MACPROC are assigned in the following manner:

Register	Utilization
0	Not used
1	Address of parameter list given to called subroutine. Miscellaneous use.
2	Miscellaneous use
3	Base for MACPROC
4	Miscellaneous use
5	Not used
6	Counter for number of characters in User Id. Miscellaneous use.
7	Not used
8	Length of requested file name. Miscellaneous use.
9	Address and base of SSBTEXT.
10	Address and base of MACPROC work area.
11	Address and base of input parameter list (MACDS)
12	Address and base of input parameter list (RPARA)
13	Address of MACPROC save area
14	Return address in MACPROC
15	Subroutine call address. Error codes.

B.3.6 Internal Work Area

WORK is the name of the internal work area used by MACPROC and it has the following format:

Name	Bytes	Content
WORK	DSECT	
SAVE2	0 - 71	Save area for MACPROC
DPRM	72 - 79	Parameter area for error messages
OPPARM	80 - 87	Parameter area for DMS open
CLPARAM	88 - 95	Parameter area for DMS close
RATMODE	96 - 99	Address of area for TMODE macro
	100 - 101	Length of area for TMODE macro
RTMODE	102 - 131	Area for TMODE macro
ADSTACK	132 - 135	Address of stack of SVC
TEMPF	136 - 139	Temporary area
TEMPH	140 - 141	Temporary area
CHKCODE	140	Code for errors in SSB routine
TASKNUM	141	Task number
DNTMOPN	142 - 143 144 - 146	For re-entrant error message
DM1	147 - 242	For re-entrant error message
DNOPDMS	243 - 244 245 - 247	For re-entrant error message
DM2	248 - 317	For re-entrant error message
TEMP	318 - 368	Temporary area
SW1	369	Code for macro entrance

B.3.7 Internal Codes

The various internal codes in the MACPROC routine are listed below by hexadecimal digits.

CHKCODE

X'04'	SSB exists and has been acquired

FUNCODE (Function code)

X'22'	Code for retrieval

MFUNCODE (Function code)

X'22'	Code for retrieval

SSBCL+1

X'00'	File EDMF closed
X'FF'	File EDMF open

SSBCTL6

X'FF'	Code that indicates good pointer in SSB block

SSBDTBIN

X'FF'	Code that indicates user description block present

SW1

X'FF'	Entrance from macro

B.3.8 Flowchart

Figures B.3.a - B.3.c contain the flowchart for the MACPROC routine.

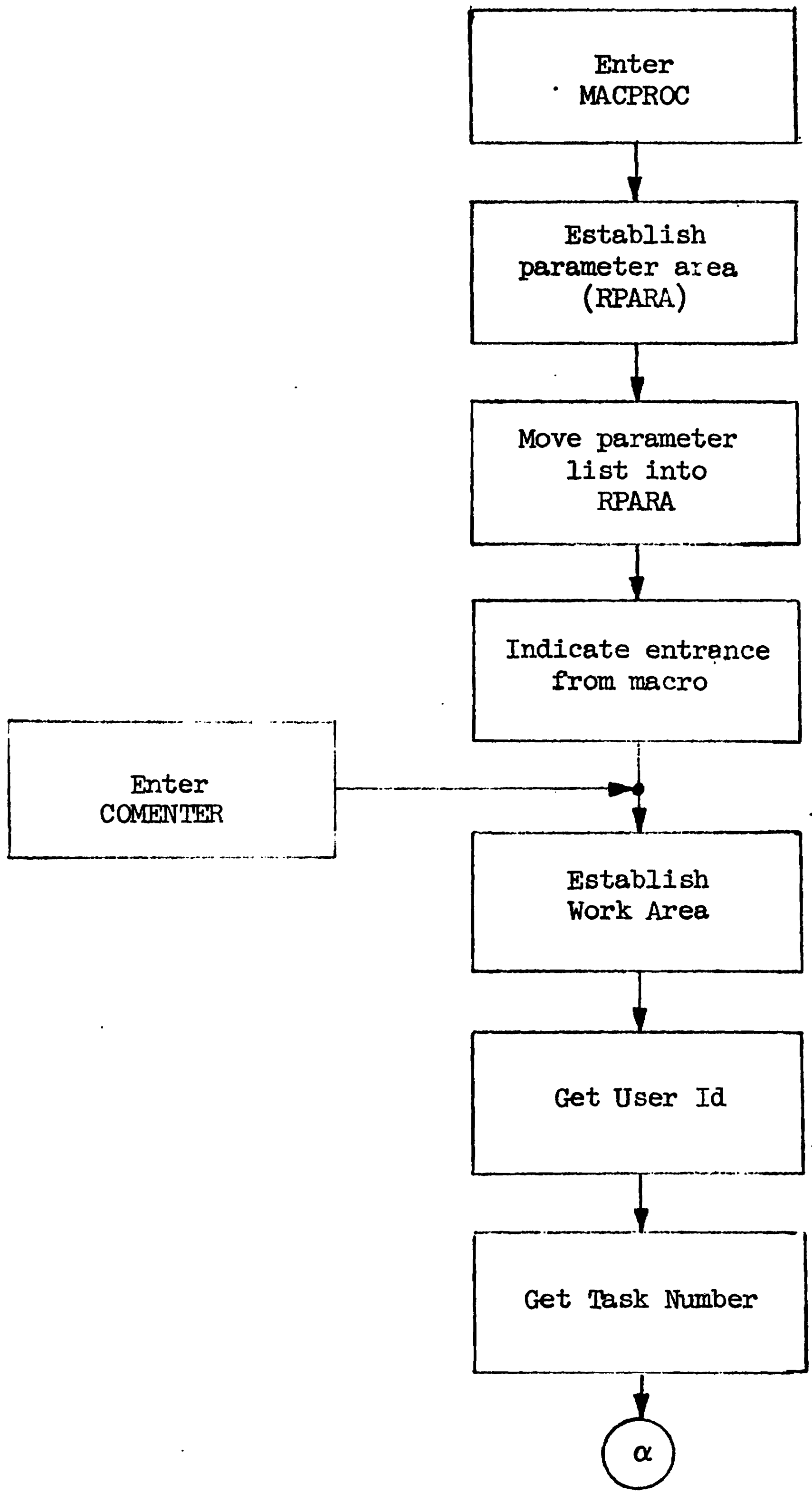


Figure B.3.a: MACPROC Initialization

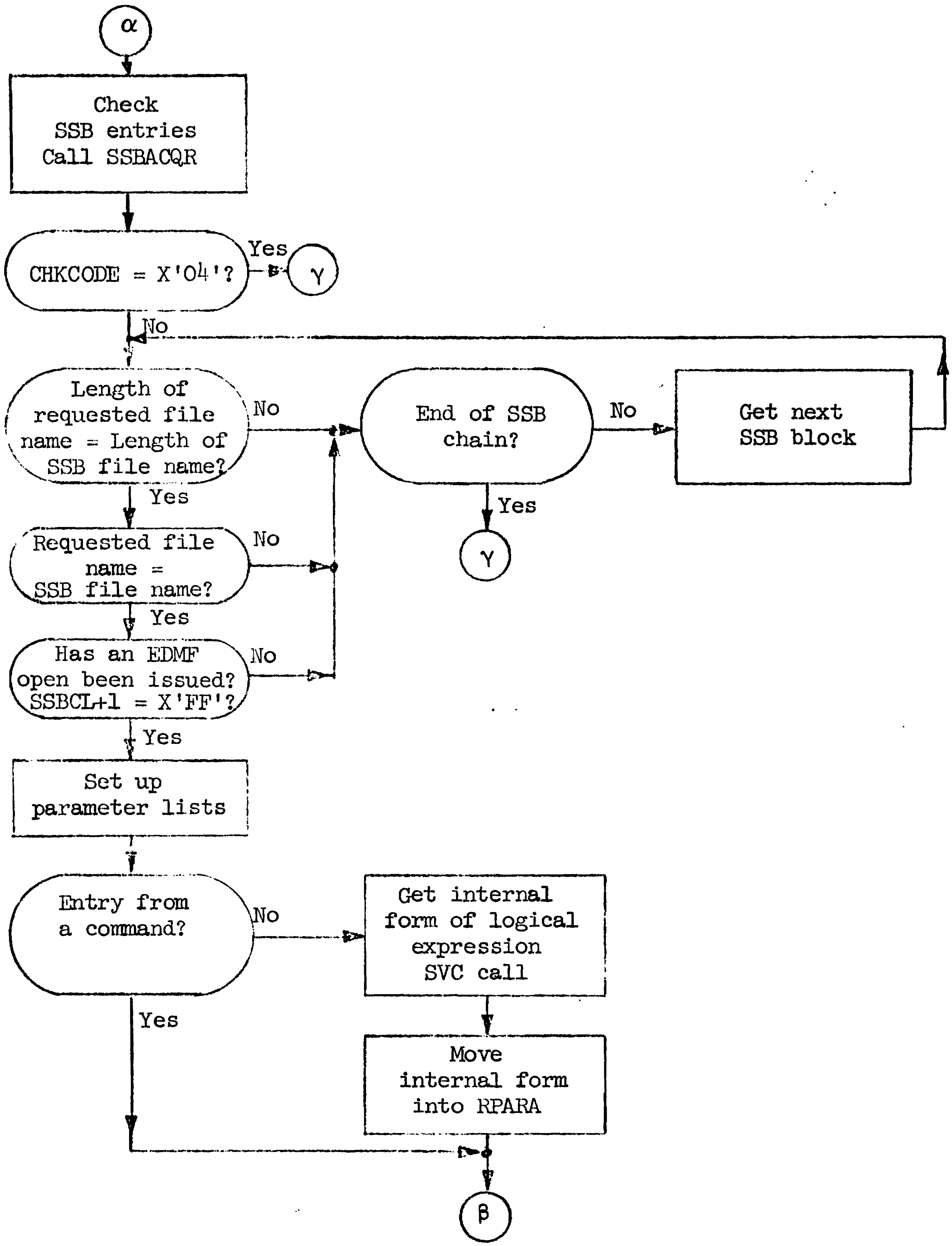


Figure B.3.b: SSB Check and Translation of Logical Expression

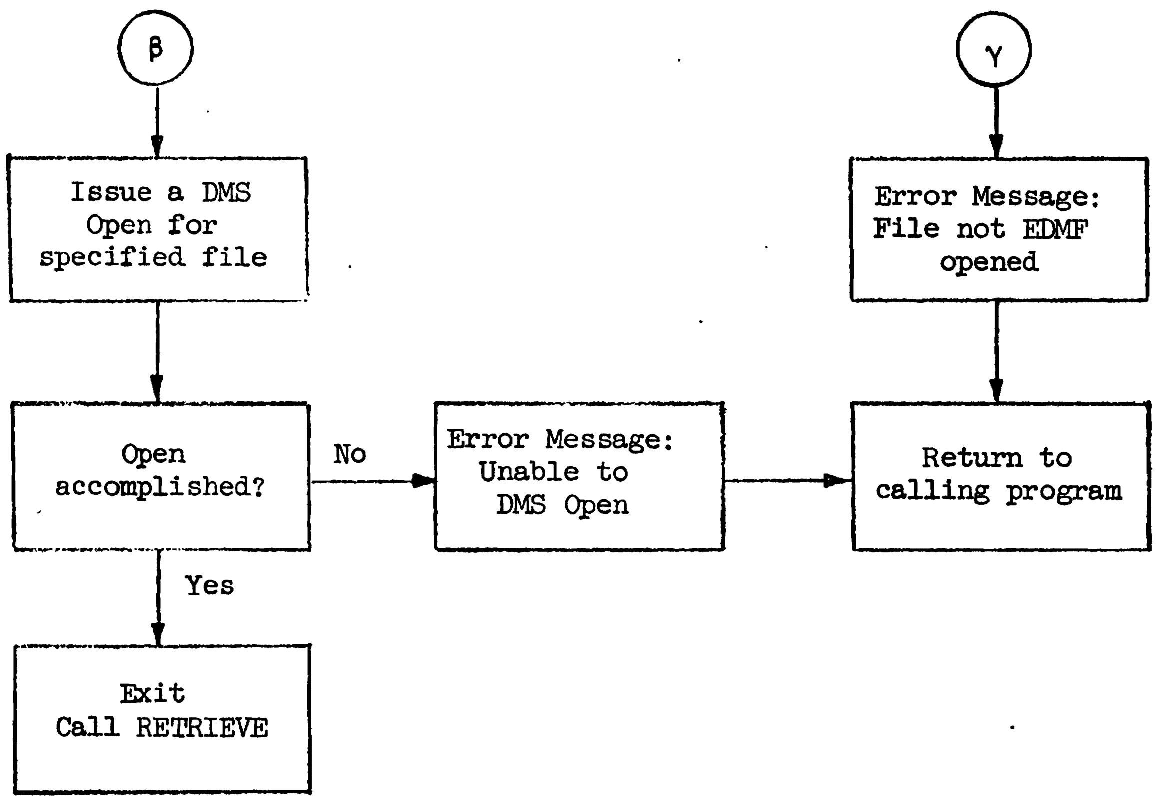


Figure B.3.c: DMS Open

B.4 Routine RETRIEVE

The routine RETRIEVE is the part of the Supervisor that implements the Retrieval Optimization function by selecting the prime keywords and also obtaining the ISAM keys that are Head of List Addresses.

B.4.1 Entry Points

There are two entry points. The normal entrance is at RETRIEVE. The second entrance is at SPCEN01; this is an error message entrance for other routines that cannot request memory.

B.4.2 Exit Points

RETRIEVE has three exit points. One is the normal exit point and the other two are used when an error occurs. The normal exit point begins at MARK and a call for the routine FORPROG is issued. The error exits are at ROUT1 and ROUT2.

B.4.3 External Subroutine Calls

Two external subroutines are called by RETRIEVE. The first is RETRDIR which retrieves the requested file's highest level directory. The second subroutine called is DECODE [3]. DECODE is used to decode the directory to determine the prime keywords and it also passes the corresponding HOLA's to RETRIEVE.

B.4.4 Input Parameter List

The address of the input parameter list (RPARA) must be in Register 1 and Register 13 must contain the address of the calling routine's save area.

Name	Bytes	Content
RPARA	DSECT	
AFCB	0 - 3	Address of File Control Block (FCB)
RFBA	4 - 7	Address of Record Format Block (RFB)
USRID	8 - 15	User Identification
RECNO	16 - 17	Number of requested records to be retrieved
OUTSPEC	18 - 27	Output specification
FLNMLN	28 - 29	Length of file name
FLNAME	30 - 83	File name (left justified with spaces)
FUNCODE	84	Code for function requested
CONTROL	85	Part of internal form of user's description
LILEP	86 - 87	Length of DCB and KIB
LDCB	88 - 89	Length of DCB
DCB	--	Actual Description Control Block (DCB)
KIB	--	Actual Key Information Buffer (KIB)

B.4.5 Register Conventions

The registers in RETRIEVE are assigned in the following manner:

Register	Utilization
0	Not used
1	Address of parameter list given to called subroutine. Miscellaneous use.
2	Miscellaneous use
3	Length of entire DCB
4	Pointer to DCB

Register	Utilization
5	Base for RETRIEVE
6	Length of DCB segments (18)
7	Not used
8	Pointer to PRIMEKEY stack
9	Address and base of input parameter list (RPARA)
10	Pointer to ADDRESS. Miscellaneous use.
11	Address and base of RETRIEVE work area
12	Pointer to RQADD. Miscellaneous use.
13	Address of RETRIEVE save area
14	Return address in RETRIEVE
15	Subroutine call address. Error codes.

B.4.6 Internal Work Area

The internal work area (SUP) used by RETRIEVE also contains the parameter lists for some of the routines called by RETRIEVE. LISTP is passed on to the RETRDIR and the DECODE routine while PLFOR is passed to DECODE and FORPROG. The work area has the following format:

Name	Bytes	Content
SUP	DSECT	
DNOAT	0 - 1 2 - 4	For re-entrant error message (DNOAT through DMSG2)
DMSG1	5 - 59	
DNOVAL	60 - 61 62 - 64	
DMSG2	65 - 132	
	133 - 135	Not used
DPARAM	136 - 143	Parameter area for error messages

Name	Bytes	Content
NLENG	144 - 151	
N	144 - 147	Smallest n* in conjunct
LENGTH	148 - 151	Length of associated HOLA's
TAREA	152 - 159	
AREAN	152 - 155	N of current keyword
TLENGTH	156 - 159	Length of associated HOLA's
ENDPRKY	160 - 163	Address of PRIMEKEY - 4
RQADD	164 - 327	Pointers to ISAM keys, length of keys
PRIMEKEY	328 - 491	Pointers to beginning of conjuncts and prime keywords
SAVE	492 - 635	Save area for RETRIEVE
REQMPAR	636 - 639 640 - 643 643 - 647 647 - 651	Parameters for $REQM
LISTP	652 - 671	Parameter list passed to DECODE and RETRDIR
ERRCODE	652	Error code
PDCB	652 - 655	Pointer to current location in DCG
PTAREA	656 - 659	Address of TAREA
ISAMIND	660	Code for DECODE
PADDRSS	660 - 663	Address of ADDRESS
PISAM	664 - 667	Pointer to area - where to put ISAM keys
PHDIR	668 - 671	Address of highest level directory
PLFOR	672 - 696	Parameter list passed to DECODE and FORPROC
PFCB	672 - 675	Address of FCB
PKIB	676 - 679	Pointer to KIB
PPRMKY	680 - 683	Pointer to PRIMEKEY Stack

Name	Bytes	Content
RFBADD	684 - 687	Address of RFB
ARQADD	688 - 691	Pointer to location in RQADD
PRECNO	692 - 693	Number of requested records
FNCODE	694	Code for function requested
ATCKLEV	695	Code for level of Authority Item Check
PRCODE	696	Code for output

*n = number of records in file containing a specified keyword

B.4.7 Internal Codes

The various internal codes in the RETRIEVE routine are listed below by hexadecimal digits.

AREAN

X'00000000'	No records within range of GT, GE, LT, LE or FROM-TO relations
X'FDFFFFFF'	Attribute of specified keyword does not exist in the file
X'FEFFFFFF'	Error in range of FROM-TO relation
X'FFFFFFFF'	Value of specified keyword does not exist in the file
Other	Number of records in file containing specified keyword

ERRCODE

X'04'	Part of directory unretrievable
X'0C'	Hardware error

ISAMIND

X'00'	Code for DECODE to return only n*
X'FF'	Code for DECODE to return n* and HOLA's

FNCODE (Function code)

X'22'	Code for retrieval

PRCODE (Print code indicating method of output)

X'00'	Output on Low Speed Terminal (LST)
X'02'	Output on high speed terminal
X'04'	Output to program in core format
X'80'	No output of actual records. Only output number of satisfactory records.

B.4.8 Flowchart and Supplementary Diagrams

Figures B.4.a - B.4.g contain the flowchart for the RETRIEVE routine. Figures B.4.h - B.4.j contain supplementary diagrams.

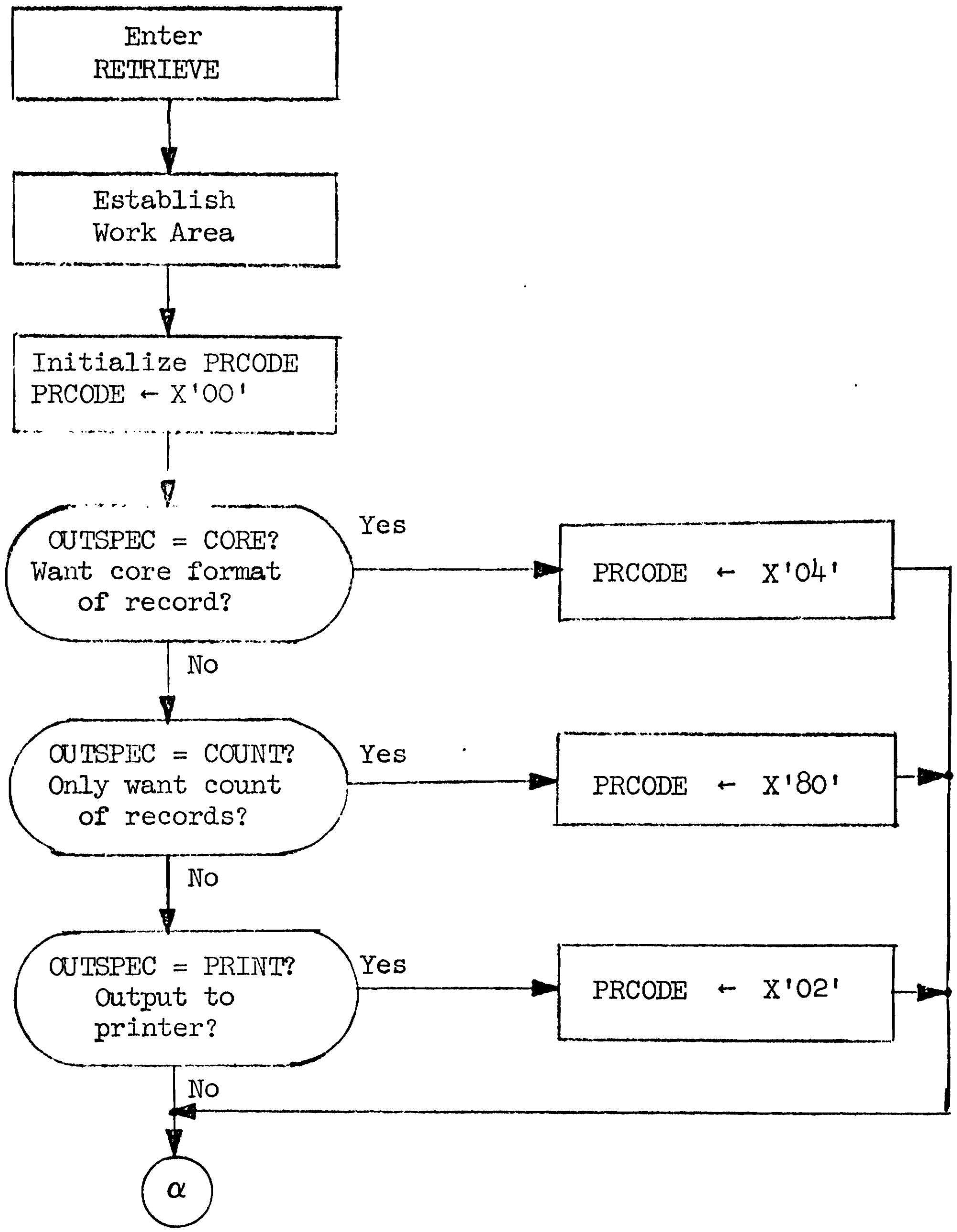


Figure B.4.a: RETRIEVE Initialization

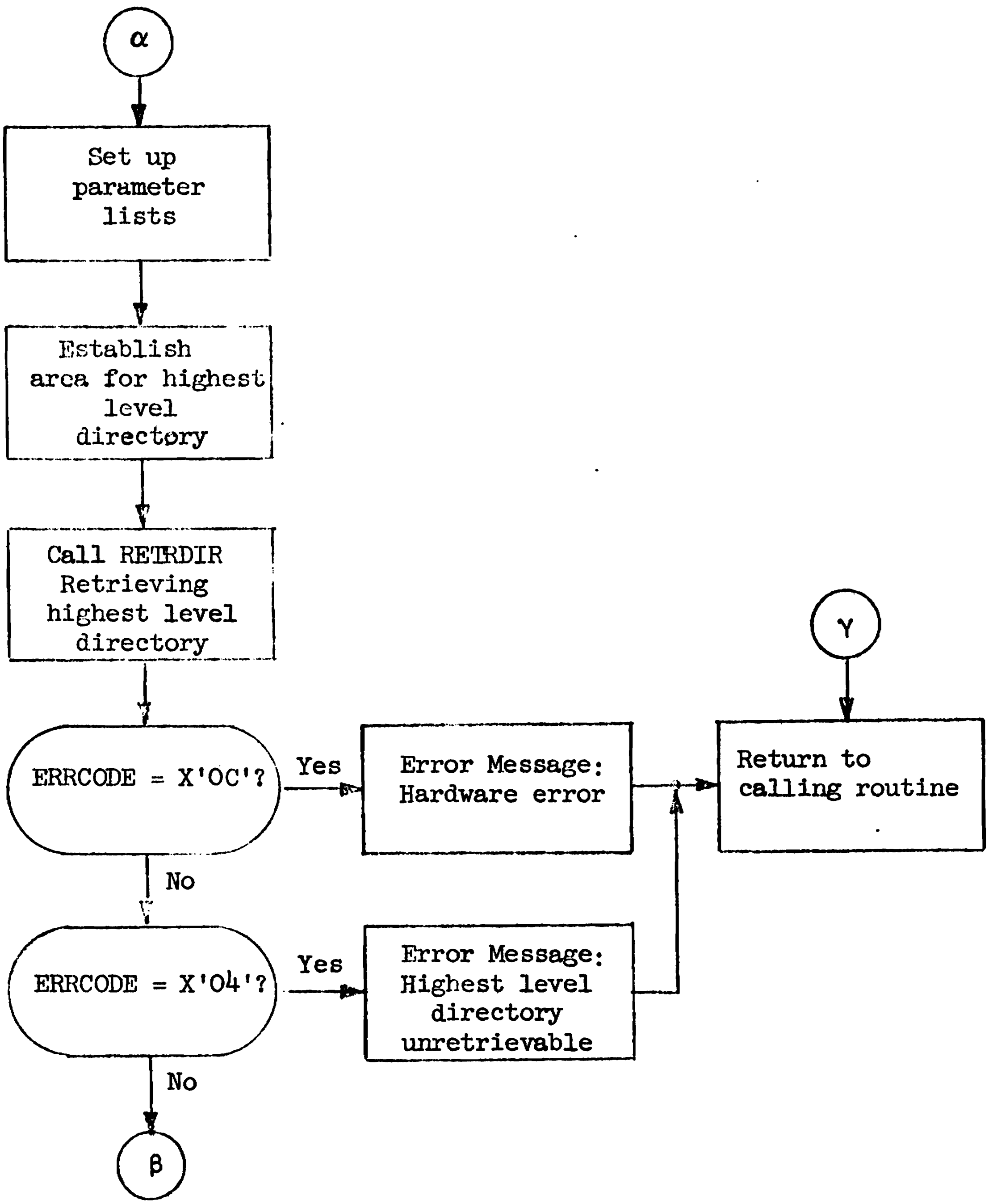


Figure B.4.b: Retrieve Highest Level Directory

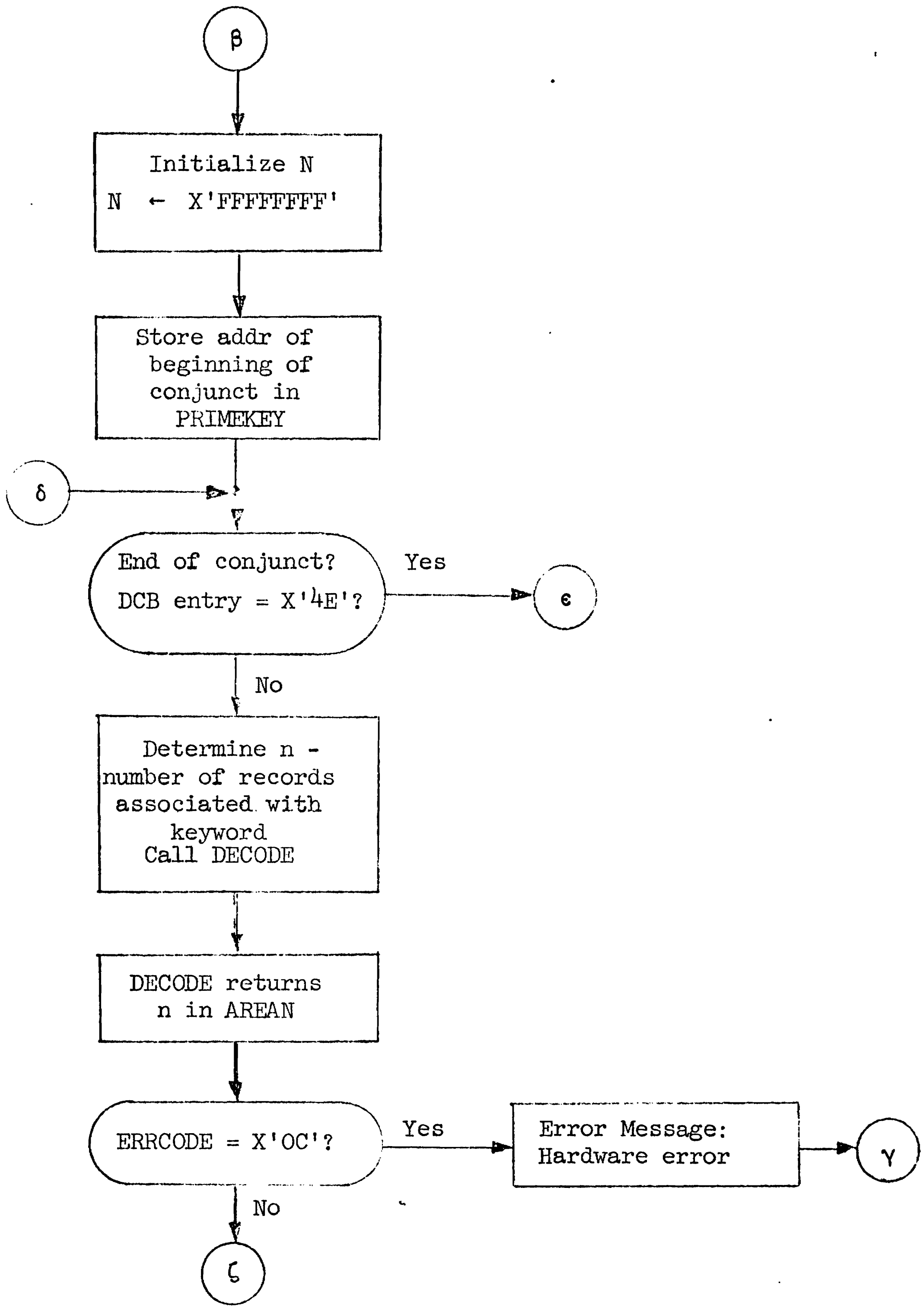


Figure B.4.c: Prime Keyword Selection

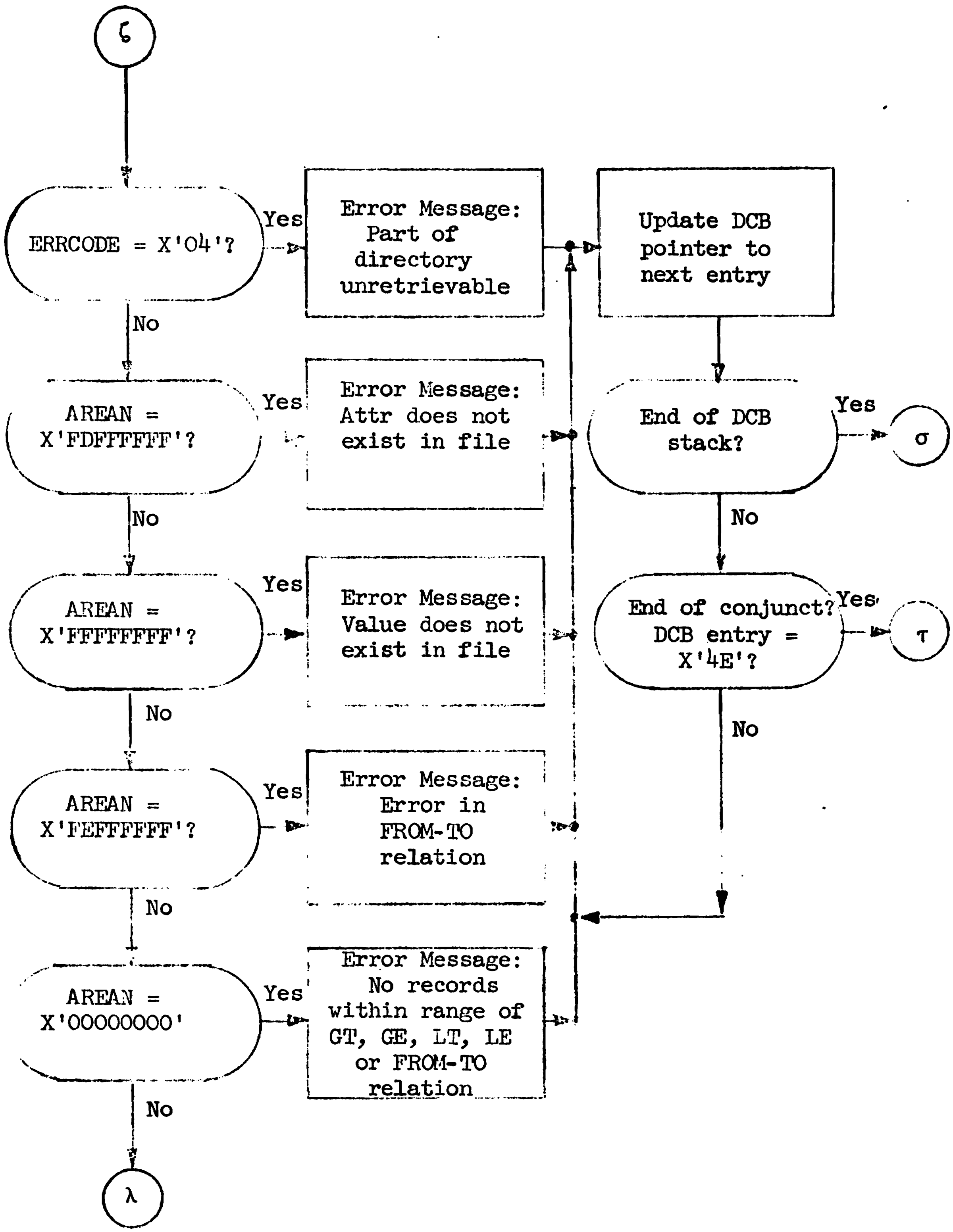


Figure B.4.d: Prime Keyword Selection

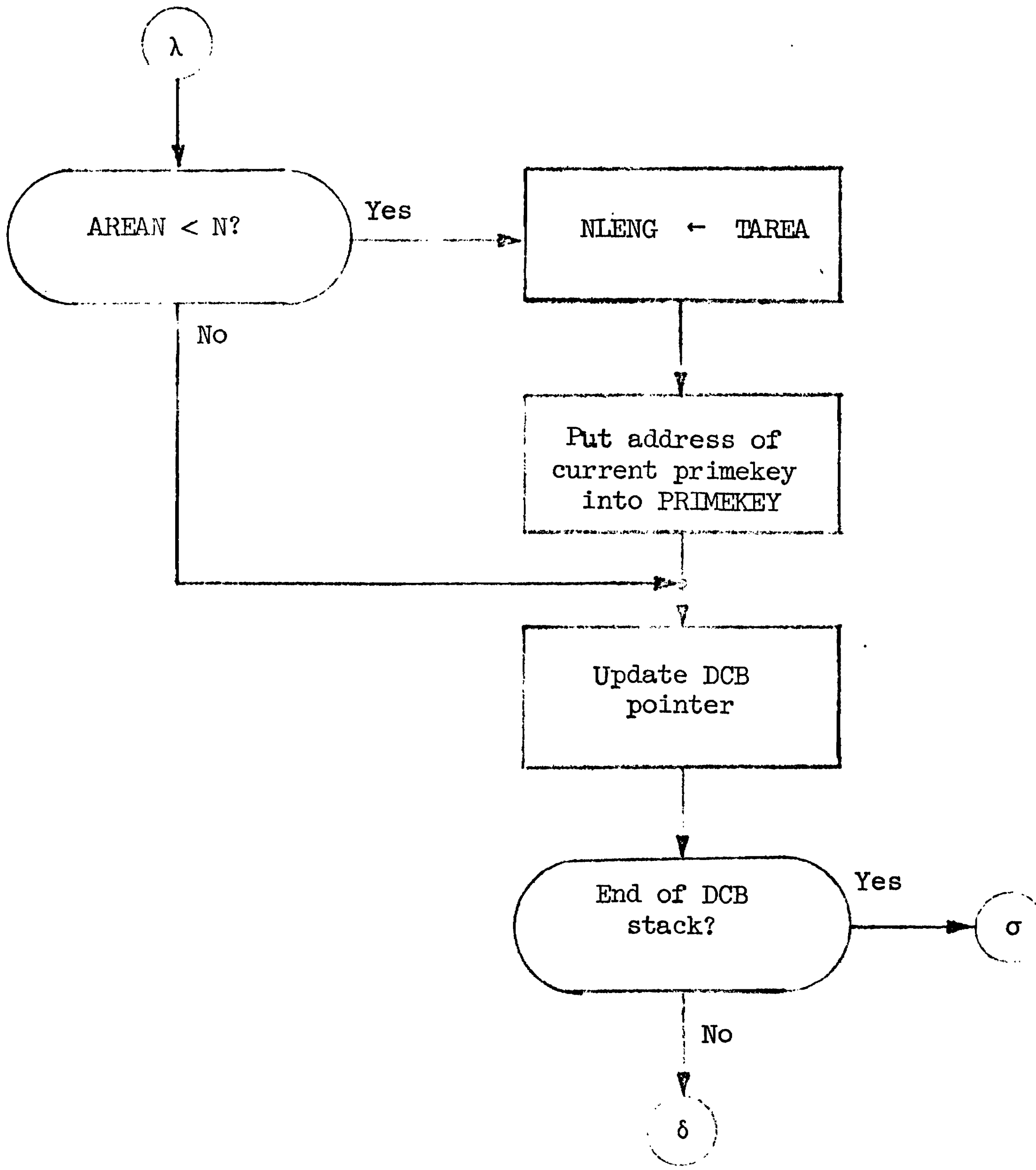


Figure B.4.e: Prime Keyword Selection

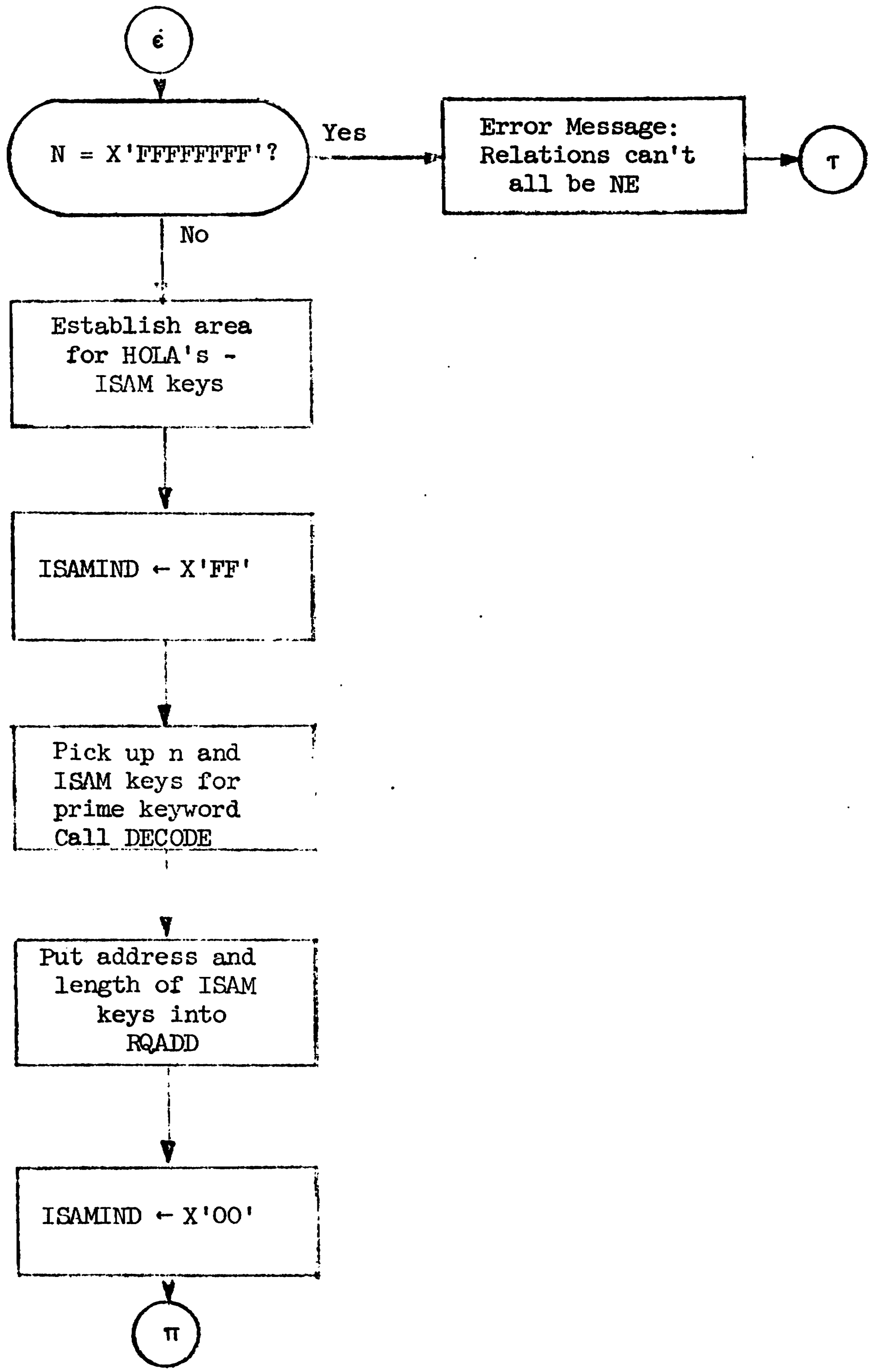


Figure B.4.f: Obtaining ISAM Keys

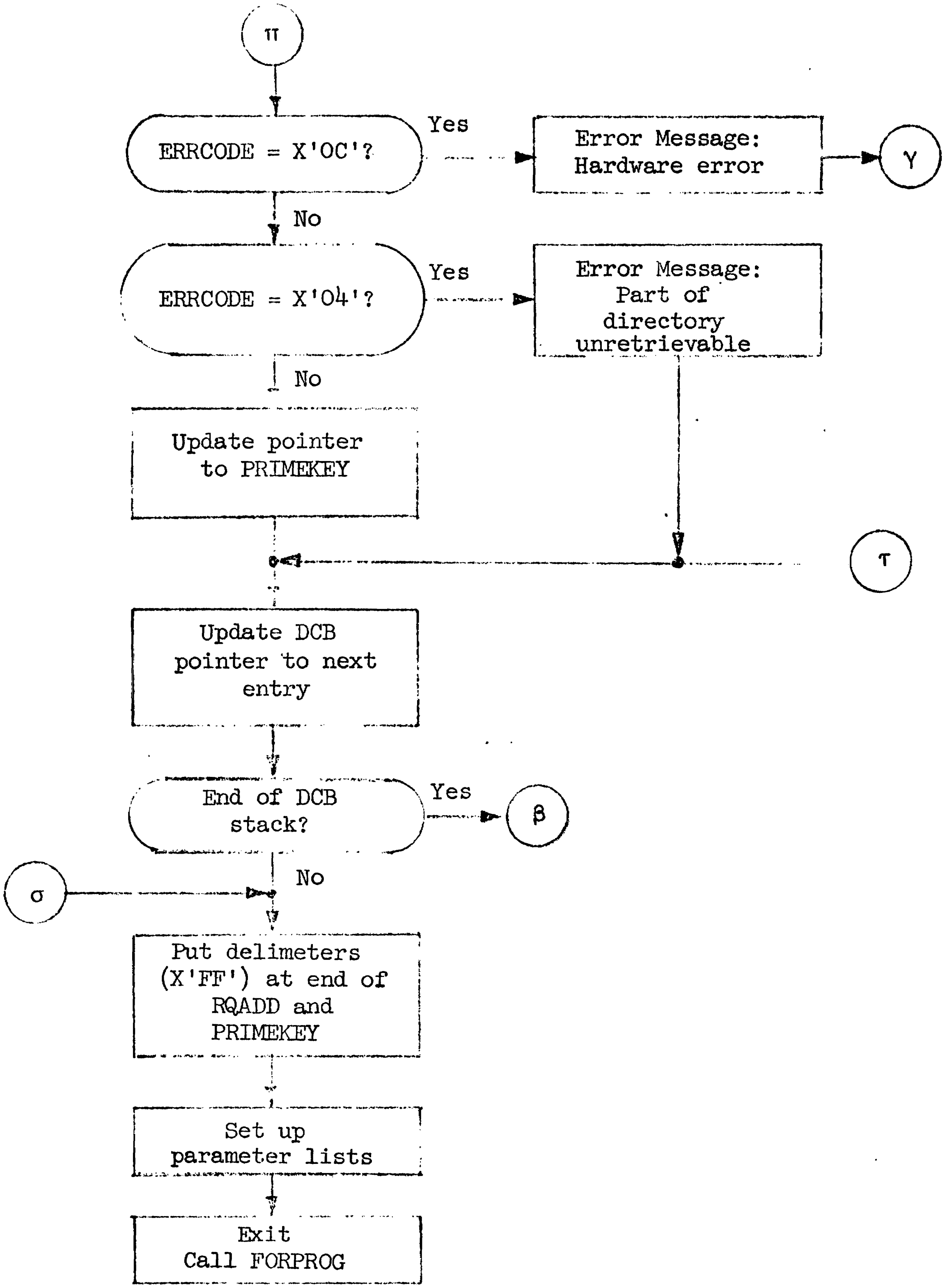


Figure B.4.g: Return to Process Next Conjunct
and the Exit

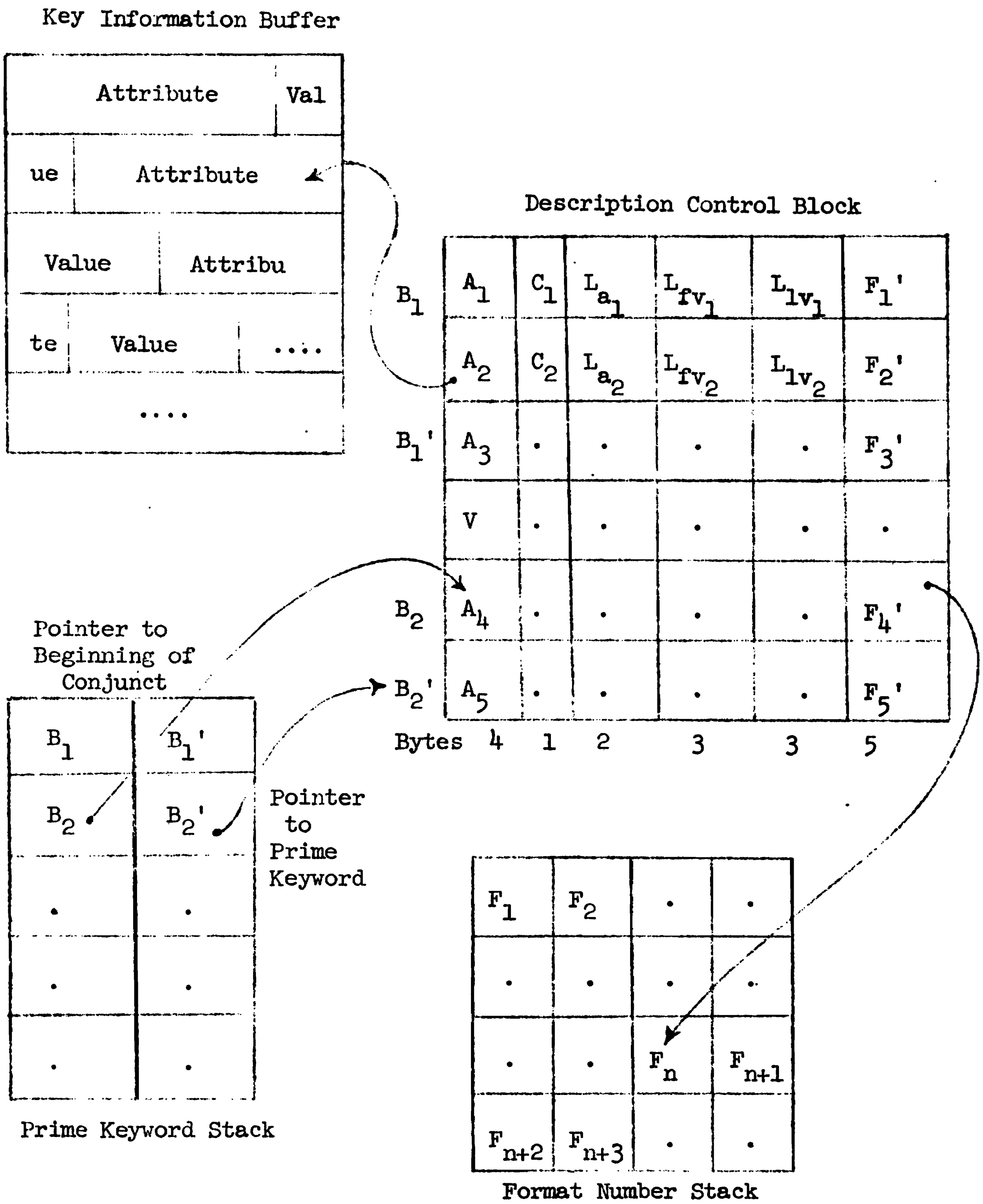


Figure B.4.h: Important Areas Used in the RETRIEVE and FORPROG Routines

Description of Control Block Areas for Figure B.4.h .

A_i is a pointer to the beginning of the i^{th} keyword that is stored in the Key Information Buffer. The attribute and value(s) are stored in their entirety, i.e. exactly the way the user specified them.

C_i is the control code that indicates the relation between the attribute and the value.

L_{a_i} is the length of the i^{th} attribute.

L_{fv_i} is the length of the first value of the i^{th} keyword.

L_{lv_i} is the length of the last value of the i^{th} keyword.

F_i' is the pointer to the beginning of a list of format numbers associated with the attribute.

Prime Keyword Stack Areas:

B_i is the pointer to the beginning of the i^{th} conjunct in the Description Control Block.

B_i' is the pointer to the prime keyword in the i^{th} conjunct in the Description Control Block.

F_i' will appear as follows:

Address of beginning of list	# of elements in list
4 bytes	1 byte

PRIMEKEY

	B_1	B_1'
	B_2	B_2'
	. . .	. . .
	B_i	B_i'
	. . .	. . .
	FF	
Bytes	4	4

where B_i : Pointer to beginning of conjunct in DCB

B_i': Pointer to prime keyword of conjunct beginning with B_i

Note: X'FF' on a B_i boundary indicates the end of the stack.

Figure B.4.i: Prime Keyword Stack

RQADD

	A_1	L_1
	A_2	L_2
	⋮	⋮
	A_i	L_i
	⋮	⋮
	FF	
Bytes	4	4

where A_i: Address of HOLA's (ISAM keys) that correspond to i^{th} prime keyword

L_i: Total length of ISAM keys

Note: X'FF' on an A_i boundary indicates the end of the stack.

Figure B.4.j: RQADD Area

B.5 Routine FORPROG

The FORPROG routine determines and lists the record format numbers for each attribute in the user's request description. The address of each list is placed in the 14th - 17th bytes of the DCB entry for the corresponding attribute. The number of associated format numbers is placed in the last byte of the DCB entry (see Figure B.4.h).

B.5.1 Entry Points

FORPROG is the only entry point in the routine.

B.5.2 Exit Points

FORPROG has three exit points. One is the normal exit point and the other two are used when an error occurs. The normal exit point begins at DONE where a call for the ESTAB entry of RETALG is issued [4]. The error exits are at FSPCEN1 and FSPCEN2.

B.5.3 Input Parameter List

The address of the input parameter list (PLFOR) must be in Register 1 and Register 13 must contain the address of the calling routine's save area.

Name	Bytes	Content
PLFOR	DSECT	
PFCB	0 - 3	Address of FCB
PKIB	4 - 7	Pointer to KIB
INDAI	8	Code for Authority Item Checking routine
PDPRKY	8 - 11	Pointer to PRIMEKEY Stack for Authority Item Checking routine
PPRMKY	8 - 11	Pointer to PRIMEKEY Stack
RFBADD	12 - 15	Address of RFB
ARQADD	16 - 19	Pointer to RQADD
PRECNO	20 - 21	Number of requested records

Name	Bytes	Content
FNCODE	22	Code for function requested
ATCKLEV	23	Code for level of Authority Item Check
PRCODE	24	Code for output

B.5.4 Register Conventions

The registers in FORPROG are assigned in the following manner:

Register	Utilization
0	Number of possible format entries (125)
1	Address of parameter list given to called subroutine.
2	Address of KIB
3	Base for FORPROG
4	Pointer to RFB
5	Counter for RFB
6	Length of attribute
7	Pointer to KIB
8	Pointer to PRIMEKEY
9	Pointer to FORMATNO
10	Pointer to DCB
11	Address and base of input parameter list (PLFOR)
12	Address and base of FORPROG work area (SUP2)
13	Address of FORPROG save area. Number of format numbers associated with a specific attribute
14	Return address in FORPROG. Miscellaneous use
15	Subroutine call address

B.5.5 Internal Work Area

The internal work area used by FORPROG is called SUP2.

Name	Bytes	Content
SUP2	DSECT	
ALFULL	0 - 3	Temporary storage
PLFRST	4 - 7	Current address in FORMATNO
SAVE	8 - 79	Save area for FORPROG
ALHALF	80 - 81	Temporary storage
TFORNUM	82 - 83	Format number that is being checked
FORMATNO	84 - 331	List of satisfactory format numbers

B.5.6 Internal Codes

The various internal codes in the FORPROG routine are listed below by hexadecimal digits.

INDAI

	X'FF'	Indicates entrance is from the Authority Item Checking routine

FNCODE (Function code)

	X'22'	Code for retrieval

MARKER

	X'FFFF'	Placed in the 6th and 7th bytes of the DCB entry to indicate a no attribute case

B.5.7 Flowchart

Figures B.5.a - B.5.d contain the flowchart for the FORPROG routine.

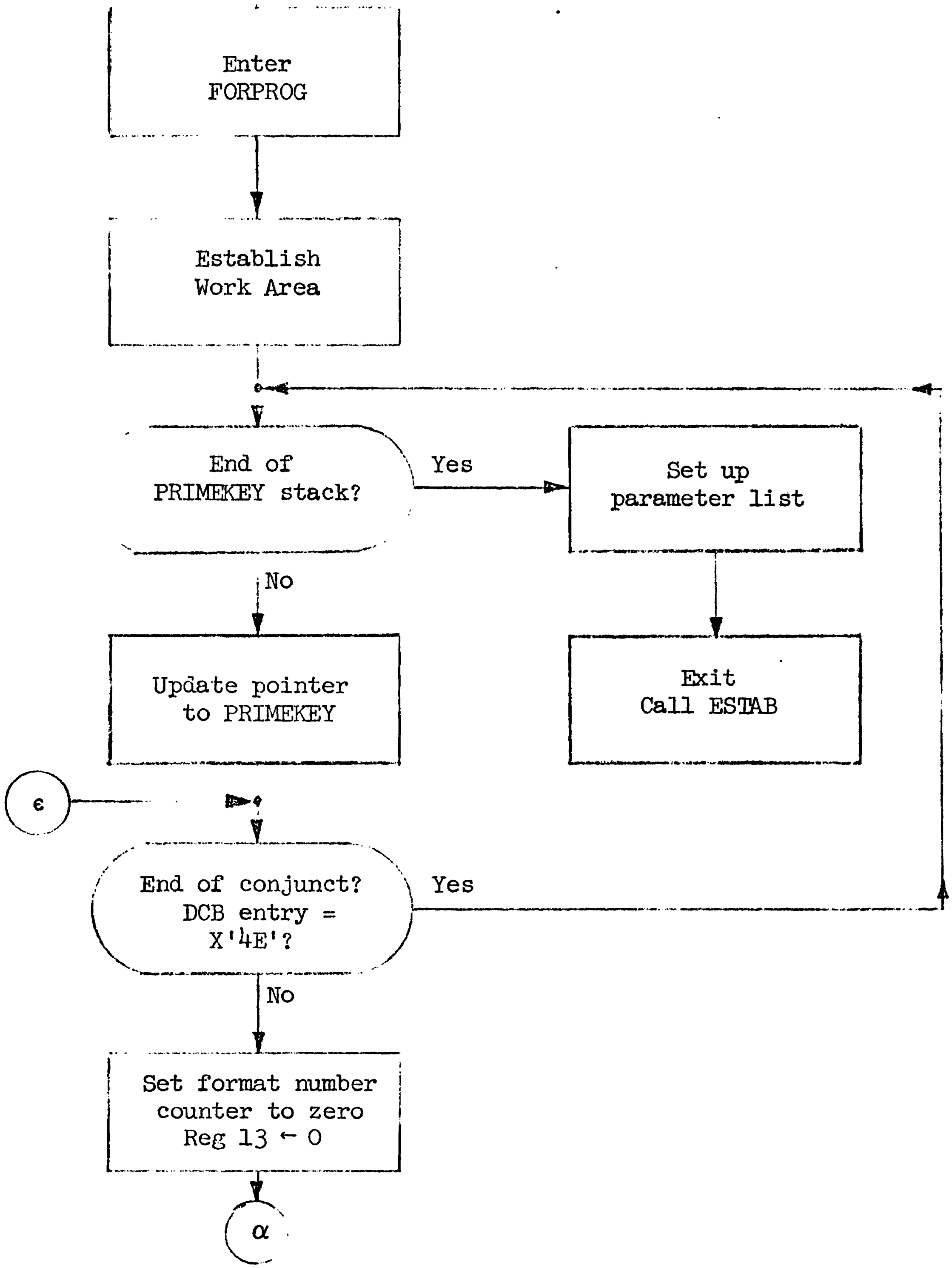


Figure B.5.a: FORPROG Initialization

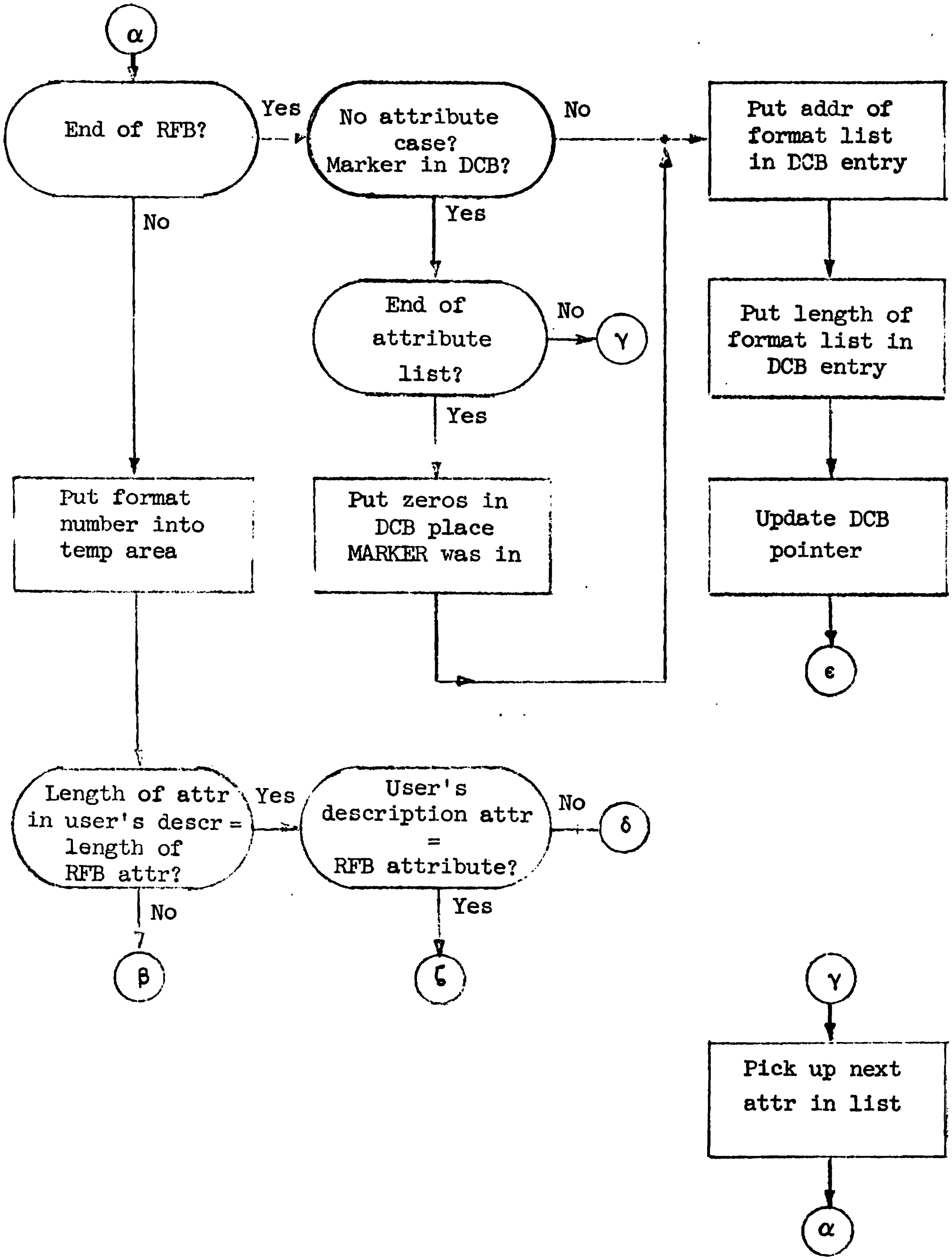


Figure B.5.b: Obtaining Format Numbers

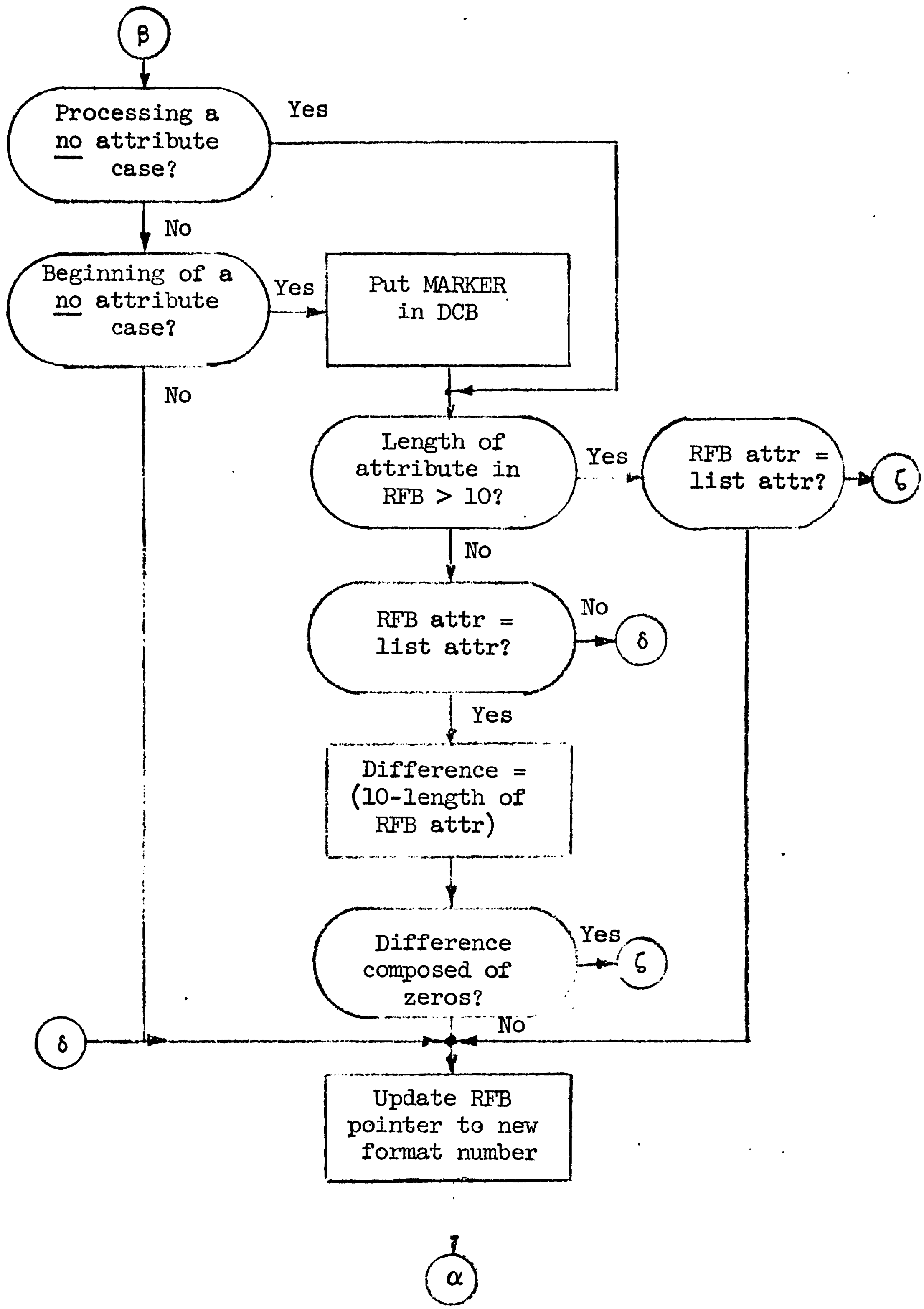


Figure B.5.c: Processing No Attribute Case

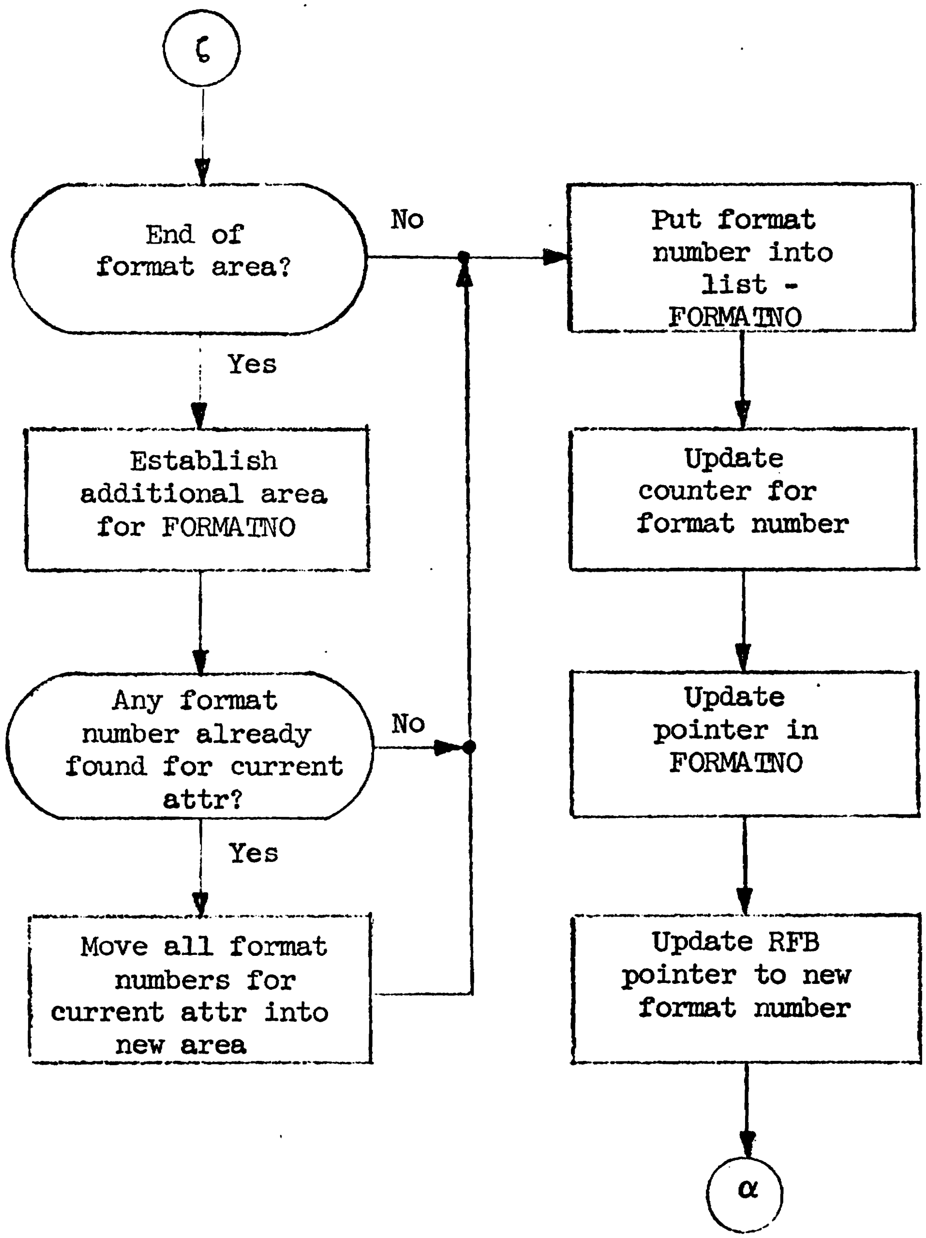


Figure B.5.d: Establishing Additional Area for Format Numbers

B.6 Routine CLSEPROC

The CLSEPROC routine is the one that implements the Close Function of the EDMF. It indicates in the SSB chain that the specified partition(s) have been closed and also removes the corresponding FSB entries.

B.6.1 Entry Points

CLSEPROC has two entry points. CLSEPROC is the SVC entrance while the command entrance is at COMDCLSE.

B.6.2 Exit Points

There is only one exit point for this routine. It begins at CKEXIT where control is returned to the calling program.

B.6.3 External Subroutine Calls

Three external subroutines are called by CLSEPROC. The first is to the location ESQCAT to obtain the task number. The second is to the entry point SSBACQR of the SSBOPTR routine [9]. This is used to obtain the SSB chain for the specified user. The third external subroutine that is called upon is FSBOPTR [9]. It is through the use of the FSBOPTR routine that FSB entries are removed. The DSECT's that are associated with the SSBOPTR and the FSBOPTR routines are the following:

Name	Bytes	Content
SSB	DSECT	
SSBHDR	0 - 7	SSB Header
SSBUAI	0 - 3	Address of User's Authority Item
SSBFIF	4 - 7	Address of FCB for File Information File
SSBTXT	8 - 91	SSB text
SSBFNAM	8 - 63	2 bytes - length of file name 54 bytes - file name
SSBCL	64 - 67	Control Information
	64	Type of request

Name	Bytes	Content
	65	Indicator - EDMF open
	66 - 67	Unused
SSBFIB	68 - 71	Address of File Information Block (FIB)
SSBFCB	72 - 75	Address of File Control Block (FCB)
SSBDTBIN	76	Open description indicator
SSBDTAB	77 - 79	Address of user description block
SSBCREC	80 - 83	Address of Core Format of the record
SSBFSB	84 - 87	Address of File Status Block
SSBCTL6	88	Control Information for pointer
SSBPTR	89 - 91	Pointer to next SSB block
FSB	DSECT	
FSBUSRID	0 - 7	User Identification
FSBCL	8 - 11	Control Information
FSBDSADR	12 - 15	Address of user's partitioning description
FSBLTBLK	16 - 19	Pointer to previous FSB block in chain
FSBCTRL	20	Control Information
FSBNTBLK	21 - 23	Pointer to next FSB block

B.6.4 Input Parameter List

The address of the input parameter list (CLSEPARM) must be in Register 1 and Register 13 must contain the address of the calling routine's save area.

Name	Bytes	Content
CLSEPARM	DSECT	
FLNMLN	0 - 1	Length of file name
FLNAME	2 - 55	File name (left justified with spaces)

Name	Bytes	Content
FUNCODE	56	Code for type of close requested
LOGEXPAD	57 - 59	Address of partitioning logical expression
LNLOGEXP	60 - 63	Length of partitioning logical expression
LOGEXP	64 - 190	Partitioning logical expression

B.6.5 Register Conventions

The registers in CLSEPROC are assigned in the following manner:

Register	Utilization
0	Not used
1	Address of parameter list given to called subroutine. Miscellaneous use.
2	Miscellaneous use
3	Base for CLSEPROC
4	Miscellaneous use
5	Address of partitioning description in SSB
6	Counter for number of characters in User Id
7	Length of description in SSB
8	Length of requested file name
9	Address and base of SSBTEXT
10	Pointer to FSBLIST
11	Address and base of CLSEPROC work area
12	Address and base of input parameter list (CLSEPARM)
13	Address of CLSEPROC save area
14	Return address in CLSEPROC
15	Subroutine call address

B.6.6 Internal Work Area

CLSEWORK is the internal work area used by the CLSEPROC routine. It contains the parameter list (FSBLIST) that is passed to the FSBOPTR routine. The work area has the following format:

Name	Bytes	Content
CLSEWORK	DSECT	
DPRM2	0 - 7	Parameter area for error messages
CLSAVE	8 - 79	Save area for CLSEPROC
ATMODEAR	80 - 83	Address of area for TMODE macro
	84 - 85	Length of area for TMODE macro
TMODEAR	86 - 115	Area for TMODE macro
USERID	116 - 123	User Identification
FSBLIST	124 - 207	List of addresses of FSB blocks to be removed
ADRSTACK	208 - 211	Address of stack area of SVC
TEMPA	212 - 262	Temporary area
TSKNUM	263	Task number
SW1	264	Code - found appropriate SSB block
CHKCODE	265	Code for errors
DNTMOPN2	266 - 267	For re-entrant error message
	268 - 270	For re-entrant error message
DMESS1	271 - 366	For re-entrant error message
SW2	367	Code for macro entrance

B.6.7 Internal Codes

The various internal codes in the CLSEPROC routine are listed below by hexadecimal digits.

CHKCODE

X'04'	SSB exists and has been acquired

FSBCTRL

X'FF'	Code that indicates good pointer in FSB block

FUNCODE (Function code)

X'48'	Close all partitions
X'49'	Close specified partition

SSBCL+1

X'00'	File partition EDMF closed
X'FF'	File partition EDMF open

SSBCTL6

X'FF'	Code that indicates good pointer in SSB block

SSBDTBIN

X'FF'	Code that indicates user description block present

SW1

X'FF'	Code that indicates found appropriate SSB block

SW2

X'FF'	Entrance from macro

B.6.8 Return Codes

All return codes can be found in the right-most byte of Register 15 and they are listed below by hexadecimal digits.

X'00'	Everything O.K.
X'04'	Appropriate SSB block does not exist

B.6.9 Flowchart and Supplementary Diagram

Figures B.6.a - B.6.c contain the flowchart for the CLSEPROC routine while Figure B.6.d contains a supplementary diagram.

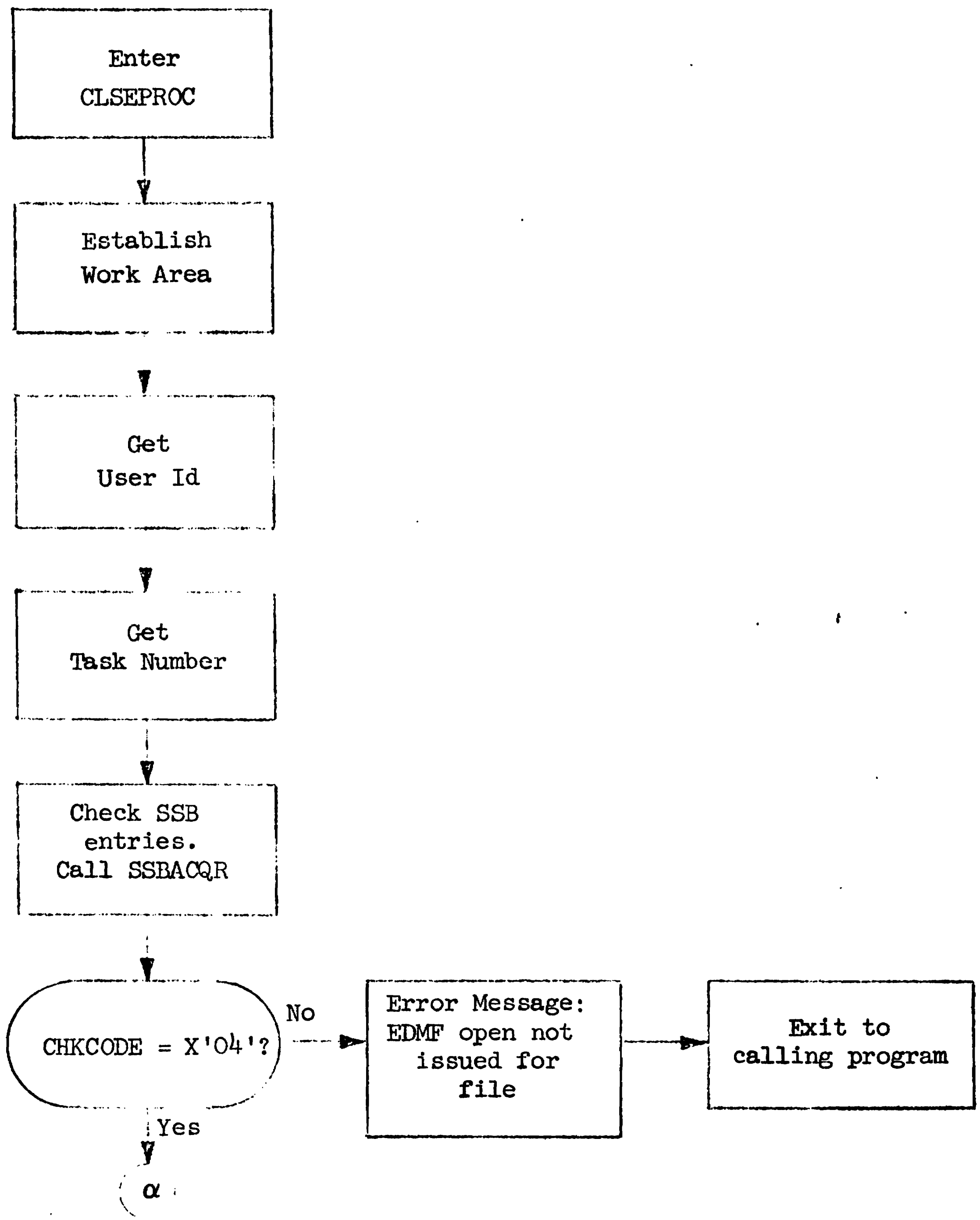


Figure B.6.a: CLSEPROC Initialization

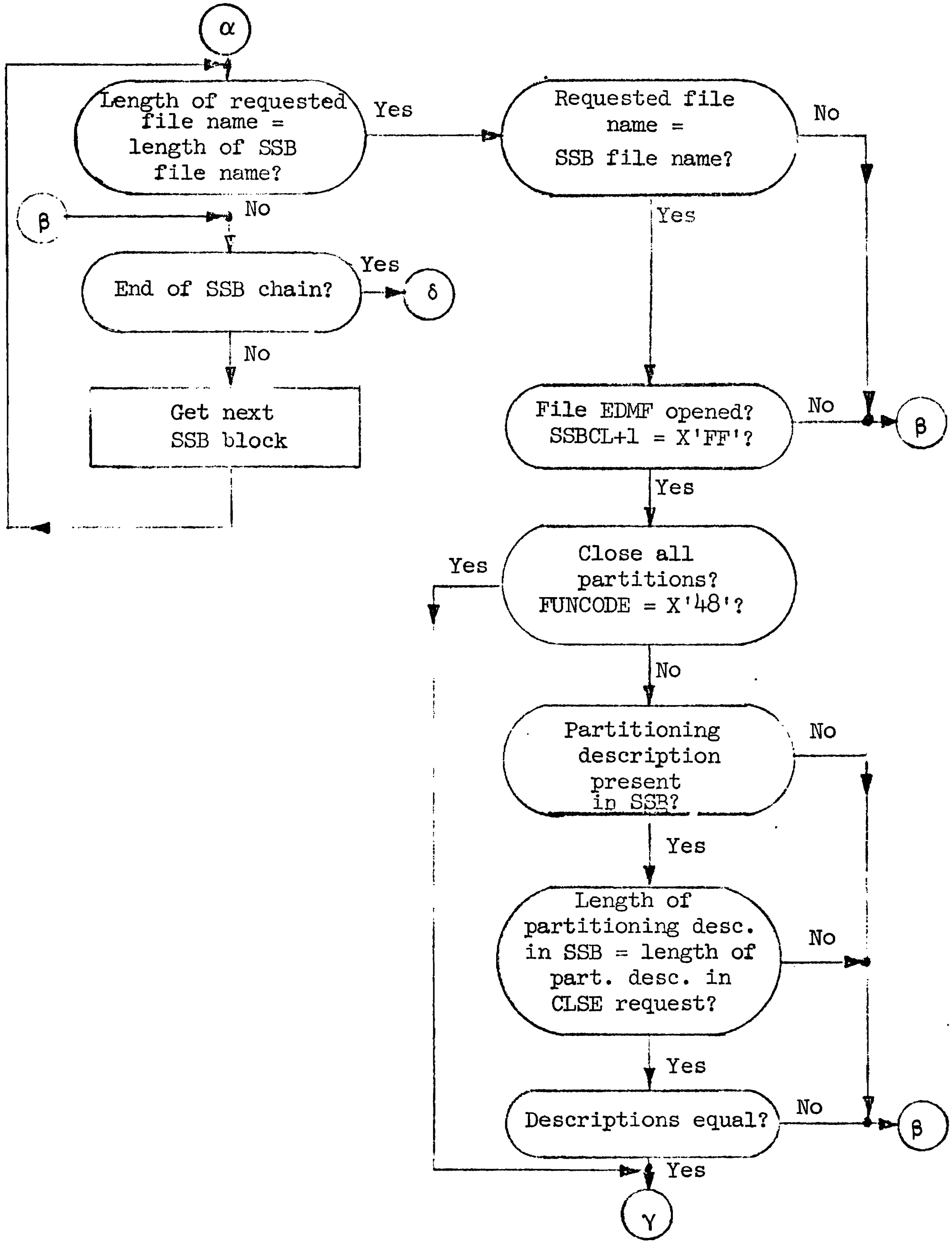


Figure B.6.b: SSB Check

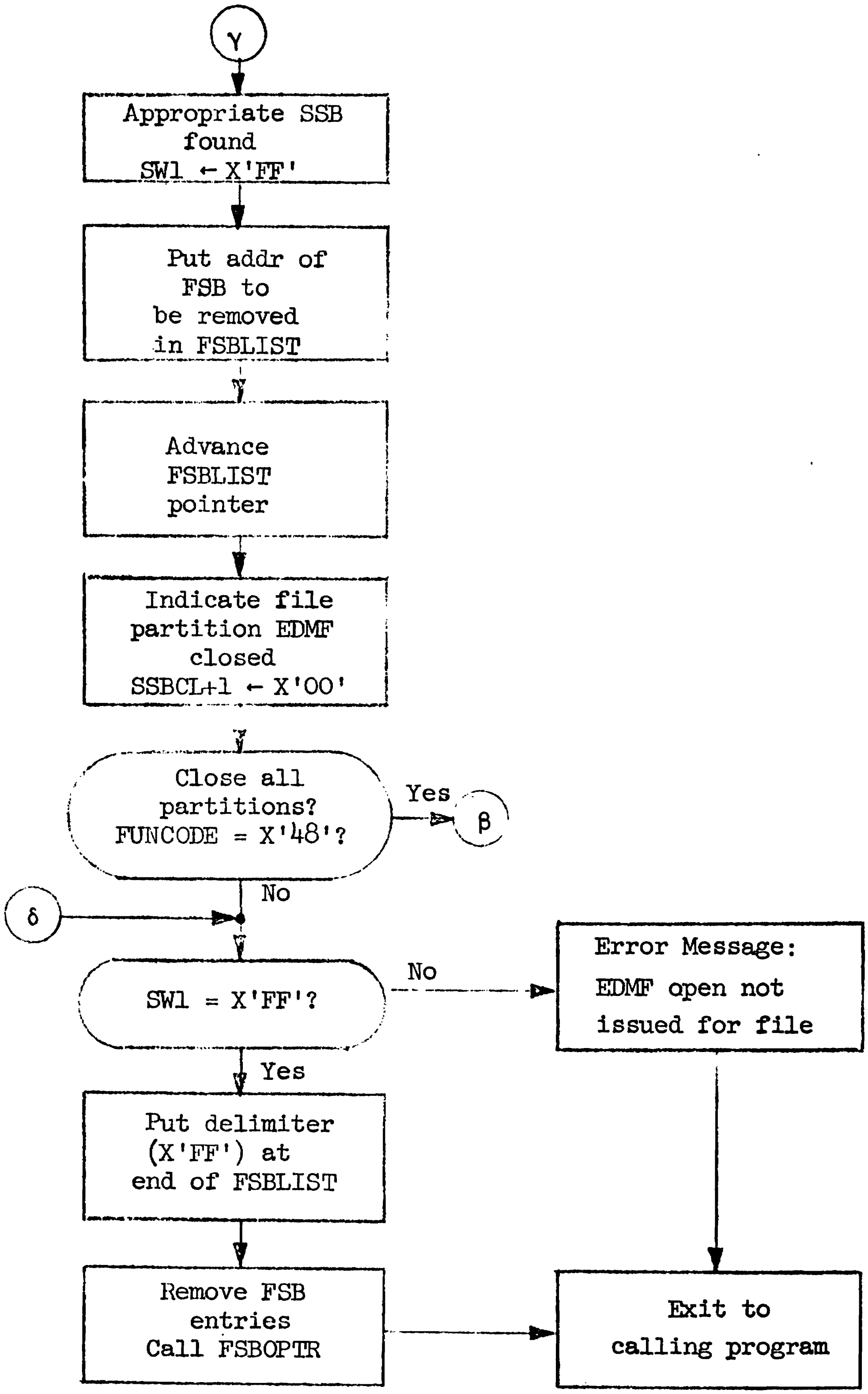
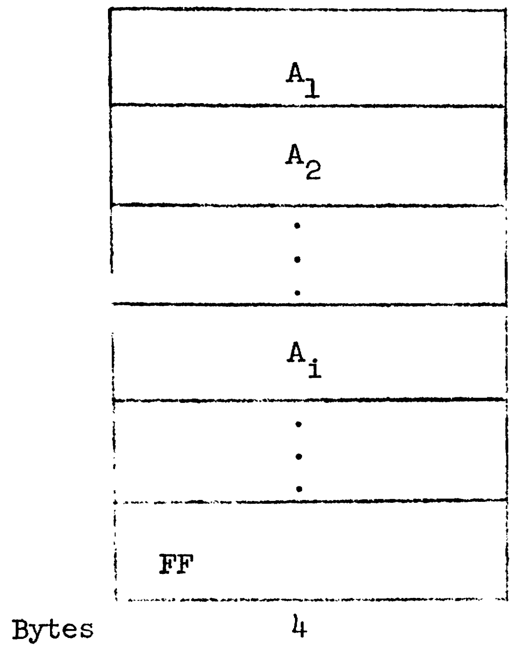


Figure B.6.c: Closing SSB, Setting Up FSBLIST and Exit

FSBLIST



where A_i: Pointer to a File Status Block

Note: X'FF' on a boundary indicates the end of the stack.

Figure B.6.d

APPENDIX C

CONTROL BLOCKS

C.1 File Status Block

8 bytes	User Identification
4 bytes	Control Information
4 bytes	Control--Address of user description block
4 bytes	Control--Pointer to previous FSB block
4 bytes	Control--Pointer to next FSB block
	00--null pointer FF--good pointer

Notes on the File Status Block

1. Unless stated explicitly, all control information is 1 byte, all addresses are 3 bytes.

C.2 Service Status Block (SSB)

Size	Contents	Part
4 bytes	Control--Address of User's Authority Item	HEADER
4 bytes	Control--Address of FCB for FIF	HEADER
2 bytes	Length of Filename	TEXT
54 bytes	Filename	TEXT
4 bytes	Control Information	TEXT
4 bytes	Control--Address of FIB for filename	TEXT
4 bytes	Control--Address of FCB for filename	TEXT
4 bytes	Control--Address of user description block	TEXT
4 bytes	Control--Address of core format record	TEXT
4 bytes	Control--Address of corresponding FSB block	TEXT
4 bytes	Control--Pointer to next SSB entry 00--null pointer FF--good pointer	TEXT

Notes on the Service Status Block

1. Unless stated explicitly, all control information is 1 byte, all addresses are 3 bytes.

2. The header appears on the first SSB block only--all subsequent SSB entries contain only the text.

   1st SSB block = 8 + 84 bytes = 92 bytes

   all subsequent SSB blocks = 84 bytes

C.2.1 User Description Block

4 bytes	Length of partitioning description
n bytes	Partitioning description

C.3 Record Format Block (RFB)

Size	Field	Section
4 bytes	Control Information	HEADER
2 bytes	Pointer to first format relative to first byte of RFB	
2 bytes	Last format number assigned	
2 bytes	Format number	TABLE OF CONTENTS
2 bytes	Control information	
2 bytes	Relative address of first format	
2 bytes	Format number	
2 bytes	Control information	
2 bytes	Relative address of second format	
.	.	
2 bytes	Format number	FORMAT ENTRY
4 bytes	Type of format	
2 bytes	Level number	
2 bytes	Repetition number	
3 bytes	Size of value	
1 byte	Control information	
2 bytes	blank	
4 bytes	Field protection data	
2 bytes	Length of attribute	
n bytes	Full attribute name	
. . .		

Notes on the Record Format Block

1. All relative addresses in the Table of Contents are relative to the first byte in the first format, hence a pointer to the first format is placed in the header. This arrangement obviates the need for changing relative addresses in the Table of Contents if new formats are added to the block.

2. Format numbers appear in the Table of Contents in order of their appearance in file records.

3. The Type of Format field may be used to indicate a program which processes the format.

4. Like the size of value entry, the repetition number will not appear in the format if the format may repeat a variable number of times. Variable repetition is indicated by a bit in the control information.

5. Control information in the format entry is one byte long with the following specification:

   abcd   ee00

   a: 0 Repetition number is variable
      1 Repetition number is fixed

   b: 0 Value size is variable
      1 Value size is fixed

   c: 0 Attribute is not in the directory
      1 Attribute is in the directory

   d: 0 Attribute optionally appears in a record
      1 Attribute appears in every record

   ee: 00 Value is packed decimal
       10 Value is alphabetic
       01 Unassigned
       11 Unassigned