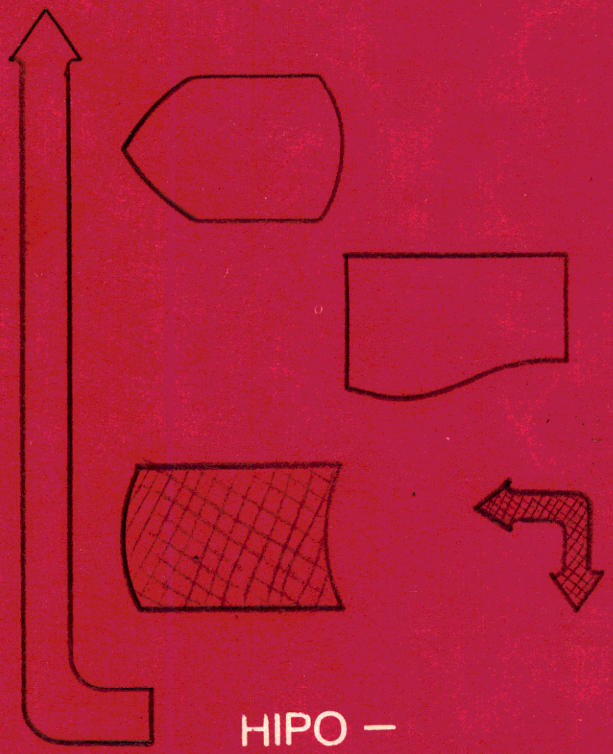
Installation Management



HIPO –
A Design Aid and
Documentation
Technique

IBM

GC20-1851-1

HIPO — A Design Aid and Documentation Technique

The purpose of this manual is to describe a design aid and documentation technique called Hierarchy plus Input-Process-Output (HIPO). Intended for systems analysts and programmers, the manual describes how to use HIPO as a design aid and documentation technique throughout the development cycle.

Preface

The purpose of this manual is to describe a design aid and documentation technique called Hierarchy plus Input-Process-Output—HIPO. HIPO describes the functions of a system from the general to the detail level, thereby providing a logical extension of a top-down development effort. It graphically shows what a system or program does and what data it uses and creates.

- As a design aid, HIPO can help define the functional structure and identify the scope of the programming assignments.
- As a documentation tool, HIPO can make a significant portion of the documentation of a system a byproduct of the development process and can reduce the time needed for locating, understanding, and altering segments of code.

This manual describes HIPO, its benefits and uses, and provides guidance in the preparation of HIPO diagrams. It is intended for systems analysts and programmers and describes how to use HIPO as a design aid and documentation technique throughout the development cycle. The manual describes the use of HIPO with the other Improved Programming Technologies and includes a case study illustrating the use of HIPO for the design and documentation of an application. Although the text provides some sample HIPO conventions that could be adapted for data processing activities, there are obviously a number of suitable conventions that can be adopted.

Formal publishing expertise is not necessary for the implementation of HIPO and it is not our intention that the exhibits and text in this manual should reflect such a requirement. In fact the intention is to demonstrate that the systems analyst or programmer can prepare correct and understandable HIPO documentation without the support of a publications specialist.

The first chapter provides a brief description of HIPO for those who will review rather than prepare HIPO packages. System analysts and programmers will also want to read the remaining chapters and appendices, which discuss HIPO in the context of a case study and provide some sample HIPO conventions.

Second Edition (May 1975)

This edition is a minor revision of, but does not obsolete, the previous edition. Various editorial changes have been made throughout.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

Address comments concerning the contents of this publication to IBM Corporation, Technical Publications/Systems, Dept. 824, 1133 Westchester Avenue, White Plains, New York 10604.

Contents

Chapter 1: What is HIPO?

One of the major concerns in the data processing industry today is the complexity of the programming systems being designed, implemented, and subsequently modified. As these systems become larger and more intricate, problems occur not only during design and development, but also during maintenance.

Good documentation is essential to the continued success of a program development effort. Unfortunately, documentation is often prepared late in the cycle and shortchanged when project schedules become tight. This can lead to problems when errors are detected in the implemented system or when changes are needed. Additionally, since many methods of documentation are not logical extensions of the design effort, it is not easy for the designer to document his efforts.

Documentation of most programming systems today consists of words, diagrams, and flowcharts. Flowcharts are helpful and consistent, but they are used to tell one side of the story: program logic. What is lacking is the function of the system, that is, what the system does rather than how it does it. Some IBM personnel believed that programming systems documentation emphasizing function could contribute to the efficiency of the program maintenance effort by speeding the location in the code of a function to be modified. They developed the HIPO technique of documenting function to meet this objective. Today HIPO is being used by some groups throughout the development cycle as a design aid and documentation technique.

Hierarchy plus Input-Process-Output (HIPO) addresses the requirements of the people who rely on documentation for many different purposes. A manager or user, for example, may want to obtain an overview of the system. An application programmer needs the documentation to determine program functions for coding purposes. Someone involved in a maintenance activity requires documentation that quickly identifies functions in which changes have to be made. HIPO meets these needs because of its graphical representation of function, its organized nesting of increasing detail, and the depiction of input and output data items at each level.

A HIPO package consists of a set of diagrams that graphically describe function from the general to the detail level. Initially, each major function is identified and then subdivided into lower-level functions; the summation of the lower-level functions equates to the higher-level functions. Programs are then developed starting with the functions described in the topmost level of diagrams. HIPO diagrams can be used from the start of the project through implementation and are useful for program maintenance by easing the identification of the code to be changed.

HIPO's Major Objectives

The major objectives of HIPO as a design and documentation technique are to:

- Provide a structure by which the functions of a system can be understood. The diagrams are organized in a hierarchy structure (see Figure 1), much like an organization chart, where each diagram at any level is a subset of the level above it. Complex systems or programs can thereby be broken into manageable pieces. For example consider a project of mapping the United States. If the project team developed page after page of prose describing the states and highways, and each city's streets and all the

connections, it would be difficult to verify or use this text even if one had the time. Even if all highways in the states and all streets of all cities were shown graphically on one diagram, it would be impossible to work with all the information at one level of detail. Therefore, a hierarchical scheme of maps, some showing states with highways, others showing cities with streets, best allows a person to view the total network as required.

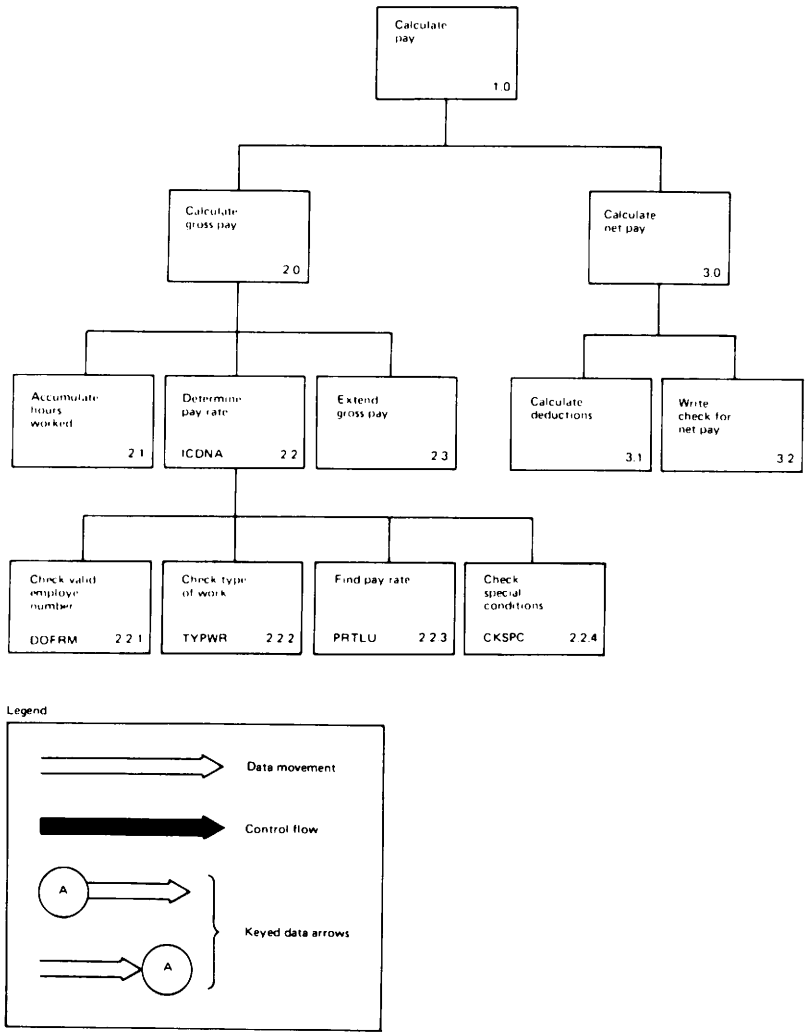


Figure 1. A hierarchy diagram

- State the functions to be accomplished by the program rather than specify the program statements to be used to perform the functions. A section in the diagrams called "Extended Description" provides additional information about the functions to reduce reliance on other documentation and to provide guidance to programmers.
- Provide a visual description of input to be used and output produced by each function for each level of diagram (see Figure 2). Typically, the most important objective in a programming system is to produce output that is technically correct and meets users' requirements. HIPO allows this transformation of input data to output data to be visible.

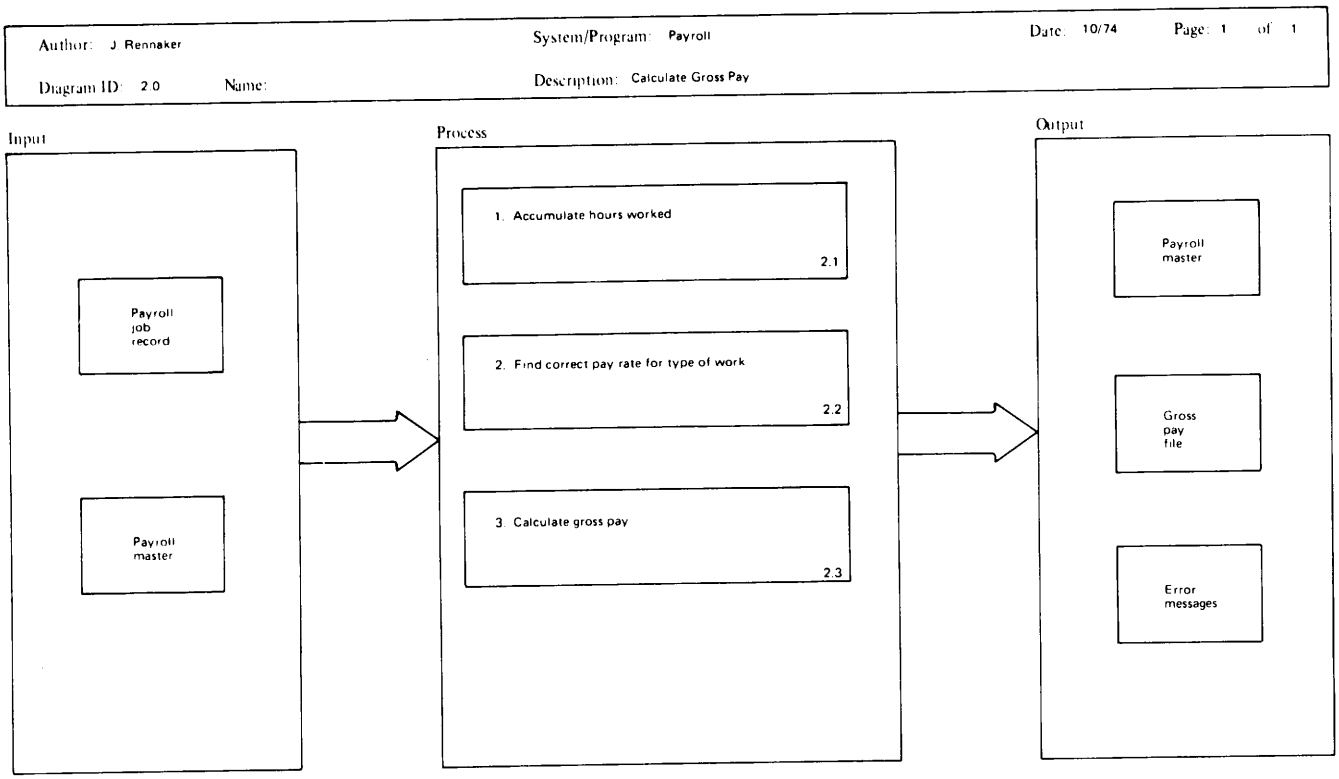


Figure 2. A HIPO diagram

Kinds of Diagrams in a HIPO Package

Before discussing the specific uses of HIPO diagrams, the basic terminology and format should be defined. A typical HIPO package contains three kinds of diagrams: a visual table of contents and overview and detail HIPO diagrams (see Figure 3).

1. Visual table of contents—This diagram contains the names and identification numbers of all the overview and detail HIPO diagrams in the package and shows the structure of the diagram package and relationship of the functions in a hierarchical fashion as depicted in Figure 1. It also contains a legend indicating how symbols in the package are to be used. With the visual table of contents, the reader can locate a particular level of information or a specific diagram without thumbing through the entire package.
2. Overview diagrams—High-level HIPO diagrams, called overview diagrams, describe the major functions and reference the detail diagrams needed to expand the functions to sufficient detail. The overview diagrams provide, in general terms, the inputs, processes, and outputs. The process section contains a series of numbered steps that describe the function being performed. The input section contains those data items used by the process steps. Arrows connect the input data items to the process steps. The output section contains those data items that are created or modified by the process steps. Arrows connect the process steps to the output data items. An extended description area can amplify the process steps and input and output data items. The extended description also refers to lower-level HIPO diagrams, non-HIPO documentation, and code. Figure 2 is an example of an overview diagram.
3. Detail diagrams—Lower-level HIPO diagrams contain the fundamental elements of the package. They describe the specific functions, show specific input and output items, and refer to other detail diagrams. The detail diagrams contain an extended description section that amplifies the process steps and references the code associated with the process steps. They also reference other HIPO diagrams as well as non-HIPO documentation such as flowcharts or decision tables of particularly complicated logic, record layouts, and so forth. The number of levels of detail diagrams is determined by the number of functional subassemblies, the complexity of the material, and the amount of information to be documented. Figure 4 is a sample detail diagram with an extended description section.

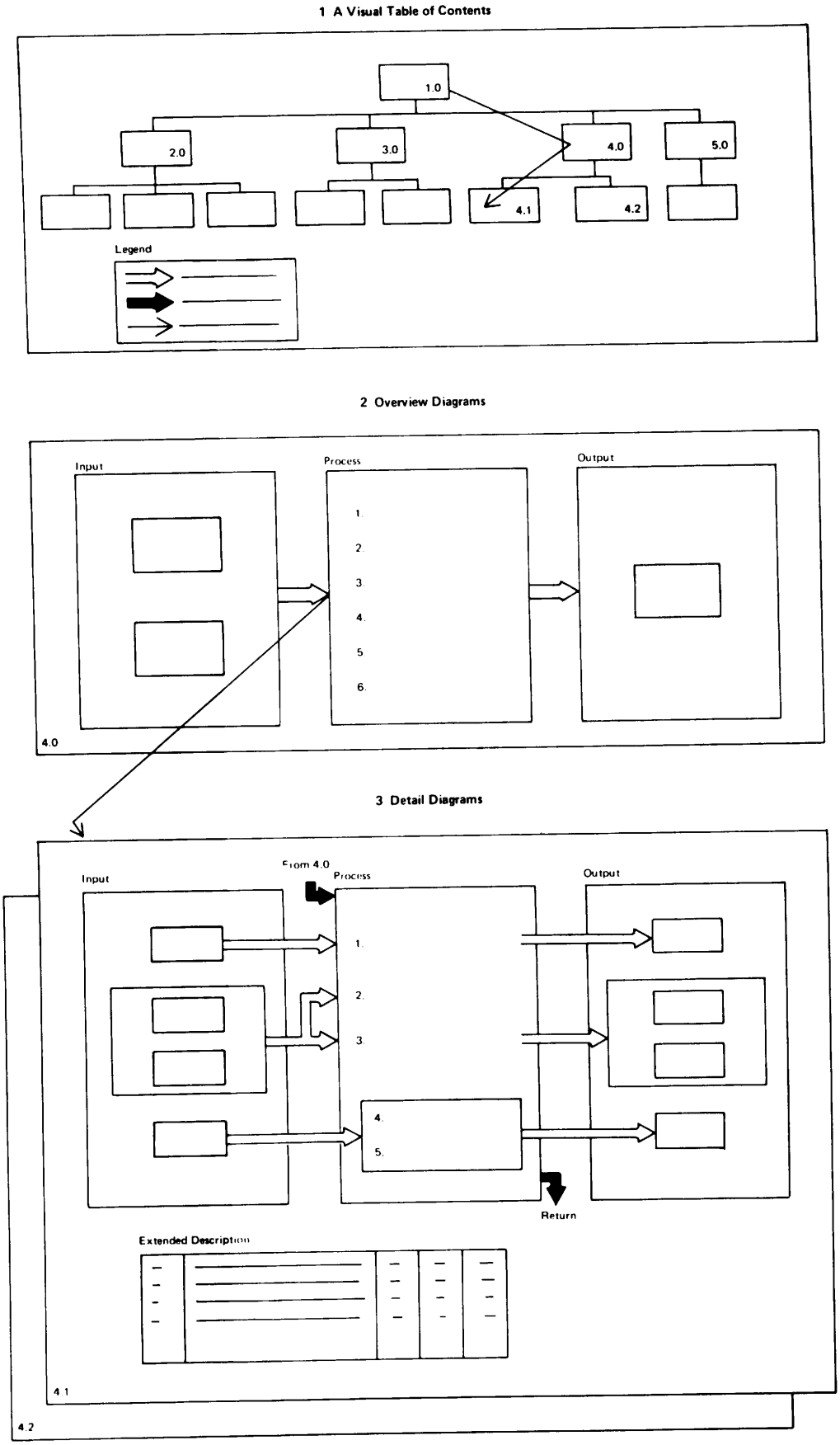
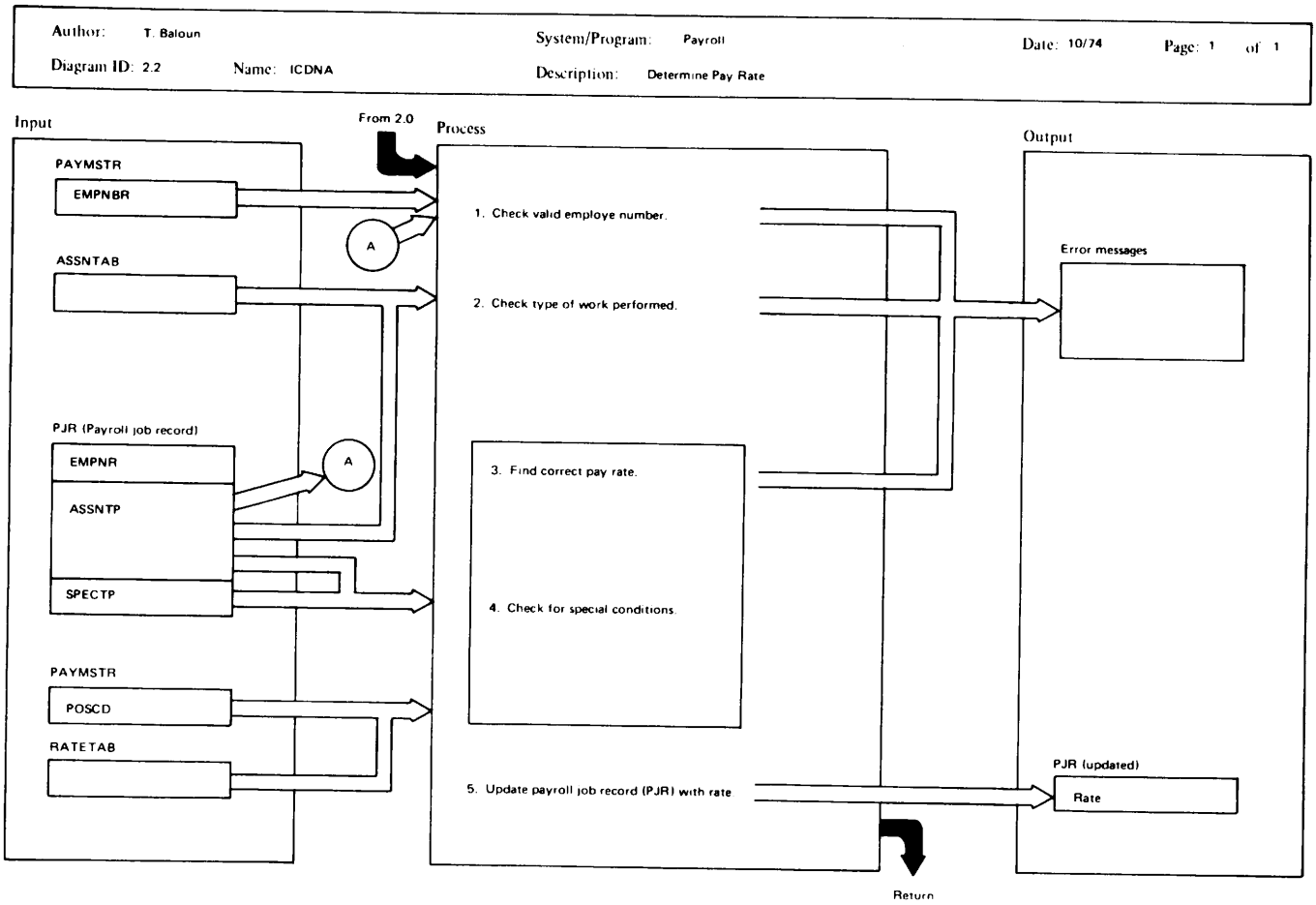


Figure 3. A typical HIPO package



Extended Description

Notes	Module	Segment	Ref.
1. If invalid, job records are bypassed and error message is printed.	ICDNA	DOFRM	2.2.1
2. If type of work is incorrect, job records are bypassed and error message is printed.	ICDNA	TYPWR	2.2.2
3. Use pay rate table (RATETAB) to find correct rate.	ICDNA	PRTLU	2.2.3
4. Check for overtime, shift pay, or holiday pay and add to rate.	ICDNA	CKSPC	2.2.4

Figure 4. A sample detail diagram

When is HIPO Used?

HIPO diagrams are evolving documentation. The output from one development phase is input to the next. There are different documentation needs during the various phases, which, for the purposes of this manual, are referred to as requirements definition and initial design, detail design and implementation, and maintenance. The amount of information needed, the audience, and the activity to be performed affect the use of the HIPO diagrams, and the status of the evolving system affects their content.

1. Requirements definition and initial design—The emphasis in this phase is on ensuring that a system's functions are understood clearly among the users and designers so that the resultant system will meet their needs. HIPO allows the designers to get a clear picture of those needs. The designer first lists the major functions of the system in an overview diagram. Those functions requiring more detail are expanded to lower-level diagrams. A detailed review can then be performed to find any missing items or weak links. Planning begins at the top of the system, proceeding to increasingly lower levels of detail.
2. Detail design and implementation—The initial design HIPO diagrams are further subdivided into more detail about each process, showing each data interface in the input and output areas and the functions to be implemented in the process section. Functions specified in the HIPO diagrams are assigned to modules of code. The programmers assigned can then further expand the HIPO diagrams to the extent necessary to code, and then code the specified functions. Reviews can ensure that functions stated in the initial design package have been included and that the code fulfills the functions of the detail design package. Functional test cases can be developed using the HIPO diagrams to determine what output is created based on the input. These cases can be used in conjunction with other tools or procedures (for example, performance evaluation). A further benefit is that HIPO diagrams can be used to introduce new members to the project.
3. Maintenance—During maintenance of a system, HIPO diagrams can be used to educate new personnel. More important, a function to be modified can be followed through the HIPO diagram hierarchy to more quickly locate the appropriate area of code. Since the HIPO diagrams describe what the program is doing rather than how it is doing it, there is less diagram maintenance when program statements change. For example, the process steps include such information as what the loop, branch, condition testing, etc., are intended to accomplish (what the program is doing). The extended description can contain the actual field values required for detailed instructions to programmers. Frequently, when changes are made to the system, only the extended description requires modification.

Kinds of HIPO Packages

There are two major kinds of HIPO packages—the initial design and the detail design—and an optional third kind, the maintenance package. Each package contains all three kinds of diagrams—visual table of contents, overview, and detail HIPO diagrams—but each package has a distinct purpose, distinct characteristics, and a different audience (see Figure 5).

- Initial design package—This package, prepared by a design group at the start of a project, describes the overall functional design of the project and is used as a design aid. The designers communicate and validate their ideas by using HIPO diagrams. This package is then used for design reviews by management and other groups, including users.

- Detail design package—This package is prepared by a development group. Using the initial design package as a base, the analysts and programmers specify in detail, add more levels of detail HIPO diagrams, and use the resultant package for implementation and comparison with the initial design package to ensure that all requirements have been included. During the coding activities, the programmers add to the extended description by filling in program labels and other pertinent information regarding the implementation. When they are finished, they have a complete, accurate, up-to-date, and easy-to-understand document. This package is good source material for maintenance groups. It may also be used to develop instructions for system users. Frequently this package is the final HIPO documentation and is used as the maintenance package.
- Maintenance package—This package is used for corrections, changes, or additions to the system. If it exists as a separate entity, it is the detail design package with some of the lower-level diagrams deleted. The ones deleted were those created to provide implementers with detailed information from which to code. As a result, future maintenance requirements for the HIPO package are further reduced. Before removal of these diagrams, the HIPO package must be checked to ensure that proper pointers to the code are provided in the remaining diagrams.

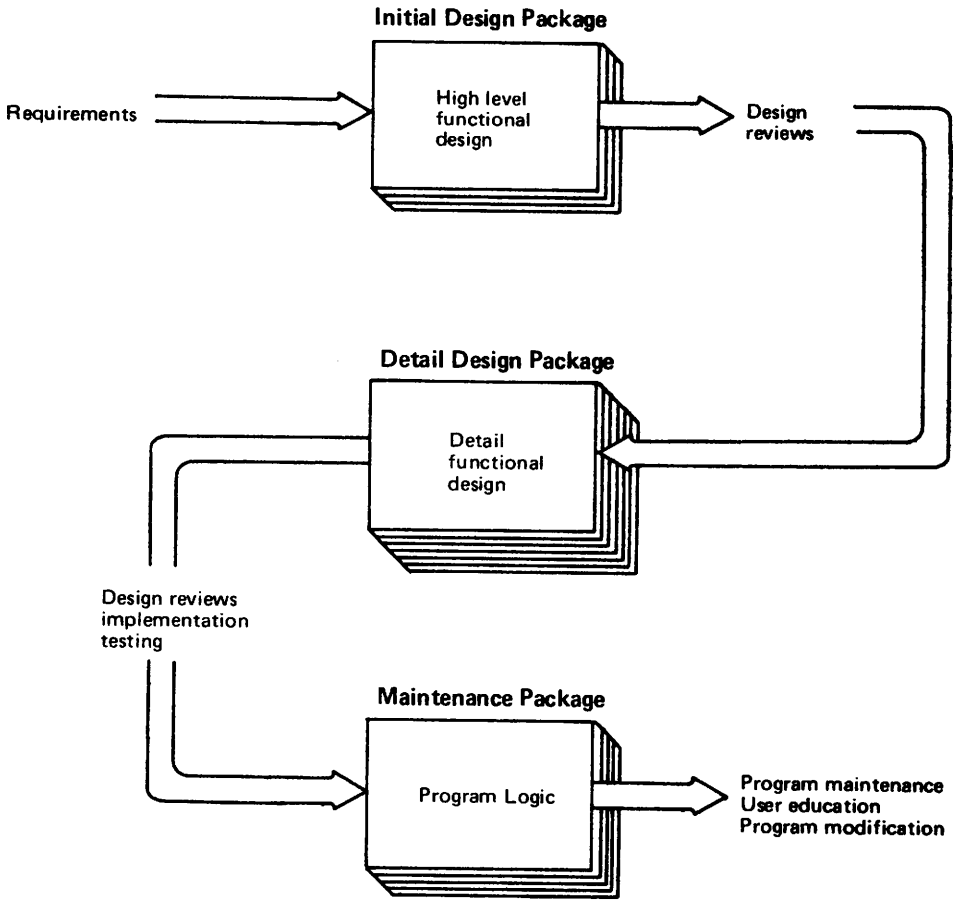


Figure 5. The three kinds of HIPO packages

Frequently in the application development cycle, the concern is for producing the code with little regard for how things really tie together. At the end of the cycle, an attempt is made to document the system by analyzing listings and reading various specification material that was generated earlier in the cycle—a difficult task at best.

The HIPO approach addresses this problem by providing a framework for establishing and documenting functional structures early in the cycle and then developing the documentation concurrently with the design and implementation of the system.

How HIPO Fits with Other Improved Programming Technologies

HIPO assumes that a system (a collection of related programs) will be organized into a hierarchical structure of functions. The scope of the topmost function encompasses all subfunctions. Those subfunctions that require further clarification are treated as major functions consisting of additional subfunctions. This process continues for as many levels as required until all functions are defined.

The hierarchical structure of HIPO is well suited to a functional design made by starting at the top and subdividing into increasingly lower levels of detail. In top-down development, the functions are implemented in the same sequence as this structure. The top module contains the highest level of control logic and decisions for each program within the system, and either passes control to lower-level modules or identifies lower-level modules for inline inclusion. The parts of each program are continually being integrated.

If structured programming is used in the implementation, the functions are considered as single entities. The code is written in segments, each with a single entry and single exit. These segments can be created from HIPO diagrams drawn with only one entry and one exit. Some recommended practices to use in writing structured code are to limit a segment of code to one page (approximately 50 lines), and to indent subfunctions or substructures. These practices enhance readability and allow easy understanding.

Combining top-down development and structured programming results in a program of extreme modularity both in function and logical structure. HIPO diagrams are a logical extension of the functions identified in top-down development and provide the necessary documentation from the start of a project through implementation.

Another concept that is being used with top-down development and structured programming is the chief programmer team organization. Team operations represent a change in approach from a loosely structured group of programmers to a highly structured team of programming specialists. The nucleus of a team operation consists of a chief programmer, a backup programmer, and a librarian; other members are included as required. The chief programmer designs the major functions of a system and codes the topmost levels of modules. He is assisted in these tasks by the backup programmer. The librarian is responsible for entering all information into the development support library, the principal objective of which is to provide constantly up-to-date and visible descriptions of the programs, test data, and exact status of the system under development. The HIPO documentation should be included in this library and may be maintained by the librarian. When other team members are added to develop and program low-level modules, they can work independently down various paths of the hierarchy with separate HIPO diagrams.

Structured walk-throughs, a generic name given to a series of reviews, have been implemented within programming groups which are using top-down development, structured programming, and chief programmer teams. The reviewers are "walked" through the system in a step-by-step fashion to simulate the function under investigation. An attempt is made to present the material in enough detail so that major problems can be identified for subsequent correction. HIPO as a top-down design aid and documentation technique lends itself well to the structured walk-through. HIPO's graphical representation of function gives something concrete and tangible to be reviewed in a step-by-step fashion at increasing levels of detail.

The techniques described above represent a disciplined approach to application development. For more information on these improved programming technologies, refer to the bibliography in this manual.

Aids for Preparing HIPO Diagrams

There are two aids available for drawing the HIPO diagrams. One is the HIPO Worksheet, GX20-1970 (see Figure 6). One side contains a large grid and is used primarily for preliminary diagrams. The other side contains the input, process, output, and extended description areas for the final draft of the diagram.

The other aid is the HIPO Template, GX20-1971, for use in drawing HIPO diagrams (see Figure 7). It is designed to be used with the HIPO Worksheet and contains the symbols needed to produce a HIPO diagram. The template jacket describes these symbols.

Considerations for the Initial HIPO Effort

It is important to use HIPO as early as possible in a project so that the designers can document their thoughts concurrently with the design process and functions will not be lost in the code. Frequently, designers' thoughts are not preserved in any formal manner. The result can be unproductive use of time, implementing functions that are insufficiently understood, as well as a system that is unable to perform the functions the designers intended.

Preparation of HIPO diagrams is a natural byproduct of the thought process of design, rather than an additional task. At some point in the development of every system, requirements and functions (along with their inputs and outputs) must be stated. This information is often communicated informally in conversations or in the form of notes on scratch paper or blackboards. HIPO provides a way of formally recording this information.

Although HIPO is most useful when employed from the initial design phase through implementation, preparing a HIPO package for an existing application may be a convenient way to learn how to use HIPO if the preparer is very familiar with the functions of the application. If an existing application cannot be used, a system in any stage of development can be documented with HIPO. The preparer should ignore the organization of the existing application's code and concentrate on function. First he should draw the top levels of the visual table of contents along with the overview HIPO diagrams for all major functions. Then these overview diagrams can be expanded into detail diagrams and shown in the visual table of contents. For partially completed systems, the extended descriptions can reference any existing flowcharts while HIPO diagrams can be developed for the remaining functions.

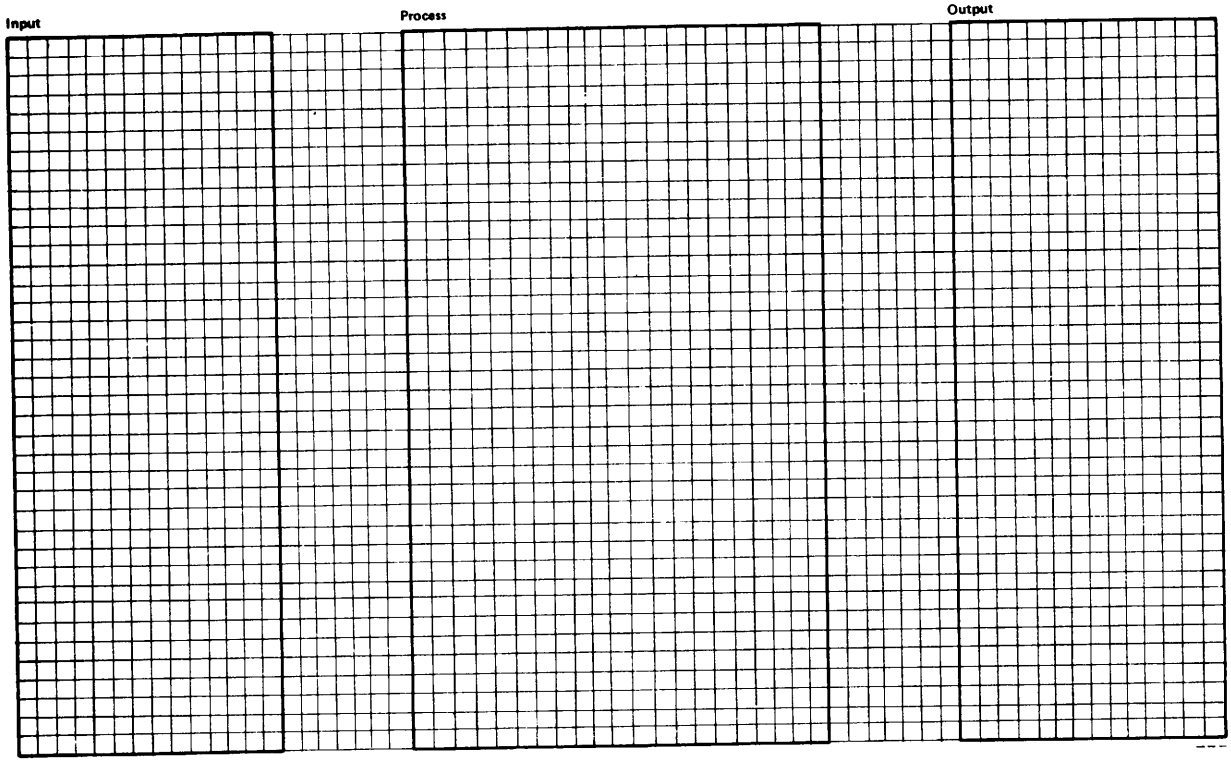
IBM® HIPO WORKSHEET

GX20-1970-0 U/M 025 *
Printed in U.S.A.

Author: _____ System/Program: _____ Date: _____ Page: _____ of _____

Diagram ID: _____ Name: _____ Description: _____

Input | Process | Output



Extended Description

Notes		Ref.

Extended Description

Notes		Ref.

* The number of sheets per pad may vary slightly

Figure 6. The HIPO Worksheet

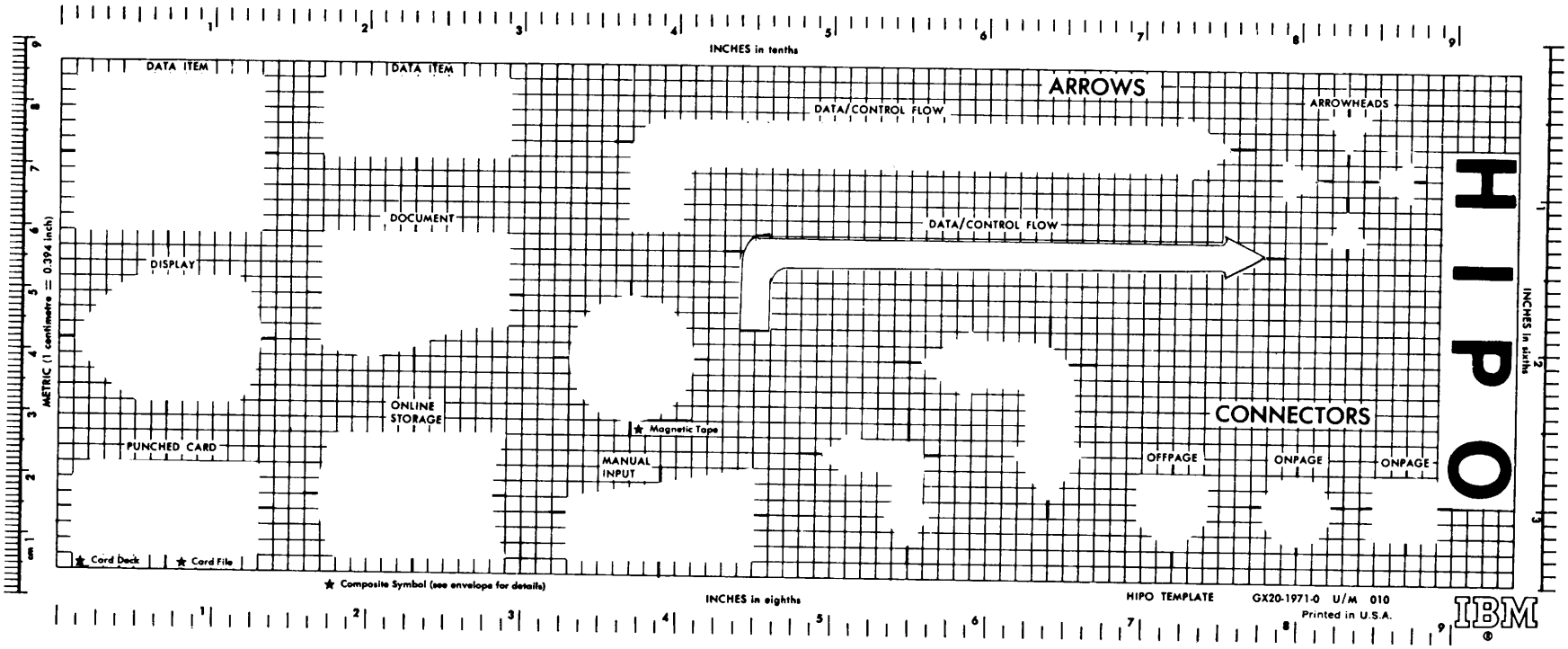


Figure 7. The HIPO Template

No drafting or other drawing experience is necessary to start a HIPO package. The HIPO worksheet and template plus a general knowledge of the characteristics of a HIPO package are the only initial requirements.

Although the first effort may be difficult, as people are unaccustomed to isolating what is to be done (function) from how to do it, experience has shown that it takes approximately 30 minutes to draw a HIPO diagram once the information to be shown is identified.

After reading this manual and analyzing the case study, the reader should be ready to try HIPO diagramming on his own.

Summary

HIPO, Hierarchy plus Input-Process-Output, is a graphical technique for showing what a system does and what data it uses and creates. A typical HIPO package contains three kinds of diagrams:

- Visual table of contents
- Overview diagrams
- Detail diagrams

The information contained in overview and detail HIPO diagrams is arranged in four categories:

- Input
- Process
- Output
- Extended description

There are three kinds of HIPO packages:

- Initial design package
- Detail design package
- Maintenance package (may be the same as the detail design package)

The aids for preparing HIPO diagrams are:

- HIPO Worksheet (GX20-1970)
- HIPO Template GX20-1971)

Using the information provided in the previous section, the reader should be able to read a HIPO diagram. The following sections provide additional information on how HIPO can be used to:

- Address the requirements of a project in a systematic manner using the hierarchy structure, thus not distorting the relative importance of the functions
- Ease program error detection and modification for operational systems because of the hierarchy structure and visible data interfaces
- Provide documentation that can allow both data processing and non-data processing individuals to more easily understand the system, thereby reducing the likelihood of an erroneous or incomplete implementation
- Include the documentation as an integral part of the project from its inception through maintenance, thereby increasing the completeness of the documentation
- Achieve smoother information transfers and less duplication of effort throughout the development cycle because the same form of documentation can be used throughout the project
- Simplify coding because the data interfaces are visible
- Conduct more meaningful reviews because the system is broken down into manageable pieces that can be reviewed at the desired level of detail
- Facilitate functional testing because test cases can be created directly from the HIPO diagrams

Chapter 2: Requirements Definition and Initial Design Phase

Chapters 2, 3, and 4 use a case study to illustrate how HIPO diagrams can be prepared and used throughout the three programming development phases of a project. Selected functions of the Alpha Search Inquiry System, an IBM Program Product (program number 5736-N14), are used as the basis for the case study. Initial design and detail design HIPO packages describing the selected functions appear in Appendices A and B. These HIPO packages were developed from the requirements statement shown in this chapter rather than from the actual code. The Alpha Search Inquiry System was developed before the widespread use of the HIPO technique.

At the beginning of a project, the designers, in conjunction with the users, produce from user input:

1. A complete statement of the requirements
2. An initial design HIPO package

The First Step—A Statement of Requirements

The initial activity in developing a data processing system is to make a complete and detailed statement of the user's requirements. This initial planning may be done by a small group of users and analysts who will:

- Determine availability of information needed to perform the function
- Point out the implications regarding the amount of effort required to do the job
- Fully define the functions to be performed by the system and their interrelationships

The following is an initial statement of the requirements for this case study:

A terminal-oriented system is needed for the retrieval of name records through a phonetic technique. The phonetic technique allows most sound-alike names, such as Klein and Cline, to be retrieved when one of them is provided.

Specifically required are:

1. A means of creating, maintaining, and reorganizing a direct access file containing customer names and contract numbers
2. Retrieval and display of records on a 3270 Information Display System using CICS even though the exact spelling of the name is not known, thus reducing the effect of transcription errors, partially illegible signatures on correspondence, and sound-alike names

Once this statement of requirements is completed, the first task is to organize it, sort out any extraneous comments, and add any necessary details not already included. An evaluation of the case study statement of requirements by users might net the following additions:

1. Superfluous information in personal names (such as Mr., Mrs., and MD) and common business words in commercial names (such as Agency and Corp.) should be discarded when building the name record.
2. If a name is changed on the file, the old name should be retained for 90 days and should refer to the new name.
3. Separate phonetic encoding algorithms for personal names and commercial names (names of business organizations) should be included.
4. Since the phonetic encoding routine will not handle all possible sound-alike names, an alternate spelling facility should be provided to handle such names as Lisle and Lyle or Creighton and Crayton. This facility should

consist of a table of common sound-alike names to be searched when requested by the terminal operator.

5. Birthdate and partial address for personal names, and partial address only for commercial names will be used as additional data (secondary identifiers) to reduce the number of names displayed.
6. The operator should be able to select the degree of likeness, that is, how close the name in the file must be to the entered name before it is considered to be a match.
7. If a displayed name is selected, the operator should have the option of passing all associated contract numbers or selected contract numbers to another routine.

Once the requirements statement is complete, the next step is to concentrate on identifying the "top" of the problem and the major subsections beneath it. Once a starting point is selected, the visual table of contents and the HIPO diagrams can be drawn.

Creating the Visual Table of Contents

This step in the design is to create a visual table of contents diagram showing the major functions to be performed by the system and their relationships to each other (see Figure 8). Actually, the visual table of contents and the HIPO diagrams are usually prepared jointly until the major functions and subfunctions have been identified. Initially, functions are specified without regard for how they might become packaged in code.

The top box in the visual table of contents states the overall function of the system (retrieve alpha search records). The next level breaks that function down into logical subfunctions: creating the records, building the file, updating the file, reorganizing the file, and inquiring against the file. The lower levels break down each of those functions into subfunctions as needed. (Usually four to five levels in this top-down structure are all that is necessary for the initial design phase, but certain systems may require more.) A general rule is to continue to lower levels until both the designer and user completely understand the functions being described.

Each box in the visual table of contents refers to a HIPO diagram whose description and identification number are shown in the box. For example, Diagram 2.0, entitled "Create Encoded Name Records", further describes the create function. While the initial purpose in drawing the visual table of contents is to functionally define and break down the system, the end product serves as a table of contents that shows where each diagram can be found. A legend should be added to the visual table of contents to aid the reader; it should describe how the diagrams are to be read, that is, what the various symbols mean.

The first visual table of contents drawn is not typically the final version, because many functional structures may meet the design criteria; as each level is broken down into further subfunctions, a better structure may be found. If this occurs, the topmost block and the several blocks under it should be reorganized. One value of using HIPO is that HIPO graphically follows a deliberate process of stating functions, thus aiding the designers in selecting from a more precise range of possibilities. If a low-level function is needed that is not implied in a higher-level function, the higher-level function was not chosen correctly.

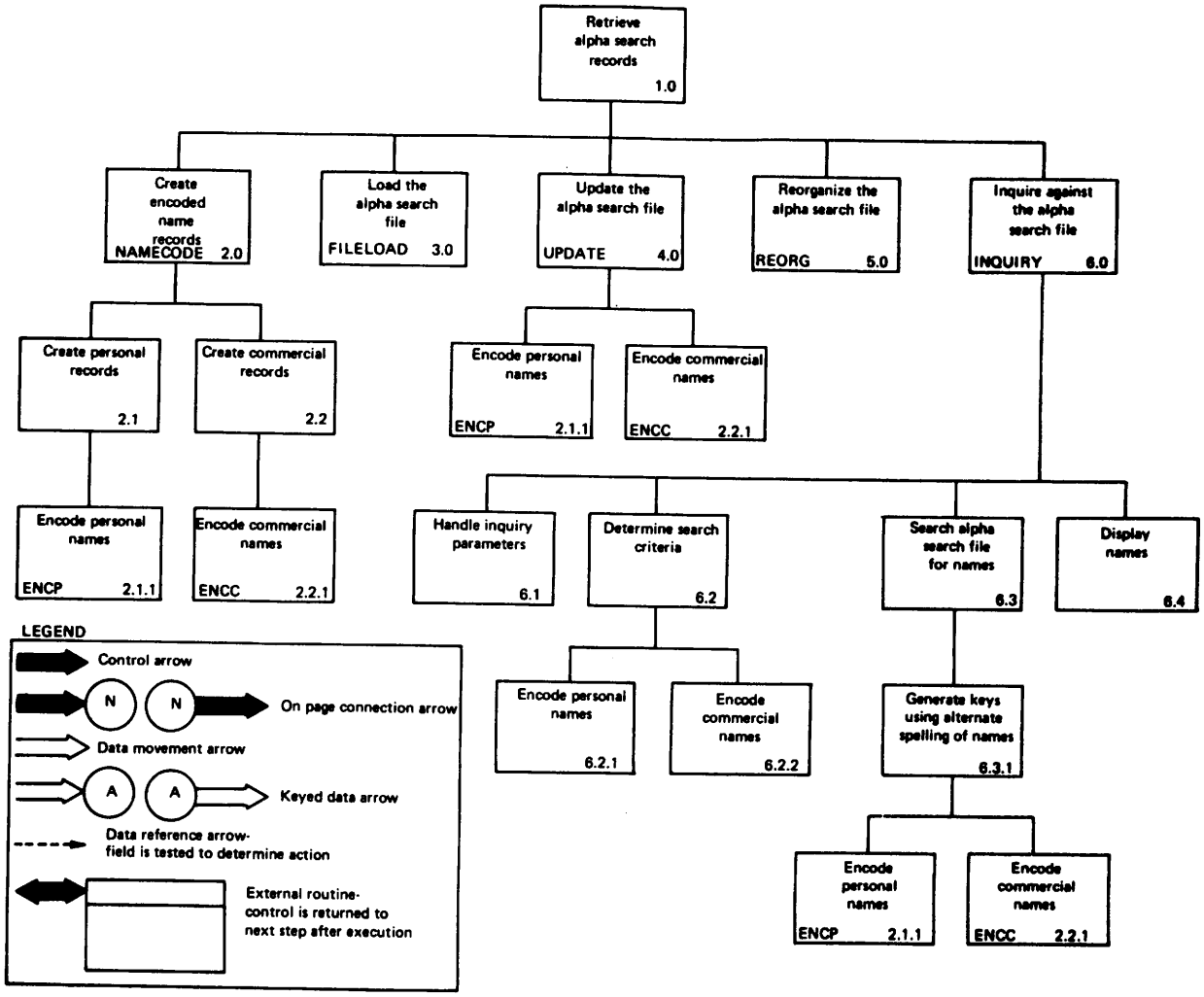


Figure 8. Initial design level visual table of contents for the Alpha Search Inquiry System

Toward the end of this phase, the programs in the system will be identified. For the Alpha Search Inquiry System, it was decided that the five functions shown at the second level of the visual table of contents will be five separate programs.

Note that the Alpha Search Inquiry System, like many data processing systems today, has been developed to fit into and operate within the framework of a larger system. While the visual table of contents fully describes the newly developed system itself and specifies its five component programs, it does not indicate where it fits into other data processing systems. A general system chart may be used to show this (see Figure 9) and can be included in the HIPO package as additional documentation. Note that the Alpha Search Inquiry System is defined inside the dotted lines. No function outside these lines is included in the Alpha Search Inquiry System.

Overview HIPO Diagrams

The purpose of the overview diagram is to give the reader general knowledge of a particular function. The overview diagram acts as an introduction to the function and a director to the lower-level diagrams. All major input and output items of any lower-level diagrams are included in the overview diagram. Although specific input and output devices need not be identified at this time, they may, if known, be noted with appropriate symbols.

Note that the overview diagram of a system consists of functions performed by multiple programs related to each other via input and output. The overview diagram for the case study (see Figure 10) also shows the five programs of the system. A strict sequence of execution is not necessarily intended in such an overview diagram (a system overview diagram). In Figure 10, for example, steps 1, 2, and 3 must be performed to create the alpha search file. Step 6 may be executed many times before steps 4 and 5. Sequence is implied in the overview diagrams for a program.

Note that all but one step in Diagram 1.0 are enclosed in boxes, with a number in the lower right-hand corner of each box. This is the identification number of the next lower diagram that further explains the subfunction. For example, step 6 is a general statement of the inquiry function. The major inputs are the alpha search file and requests for names from the terminal operator; the major outputs are the names, displayed on a terminal, that are identical or similar to that requested. This input, output, and function is expanded in Diagram 6.0, the overview diagram of the inquiry function (see Figure 11). Though more detailed than step 6 in Diagram 1.0, the subfunctions, inputs, and outputs of the inquiry function are still stated in somewhat general terms. Abbreviations should be avoided.

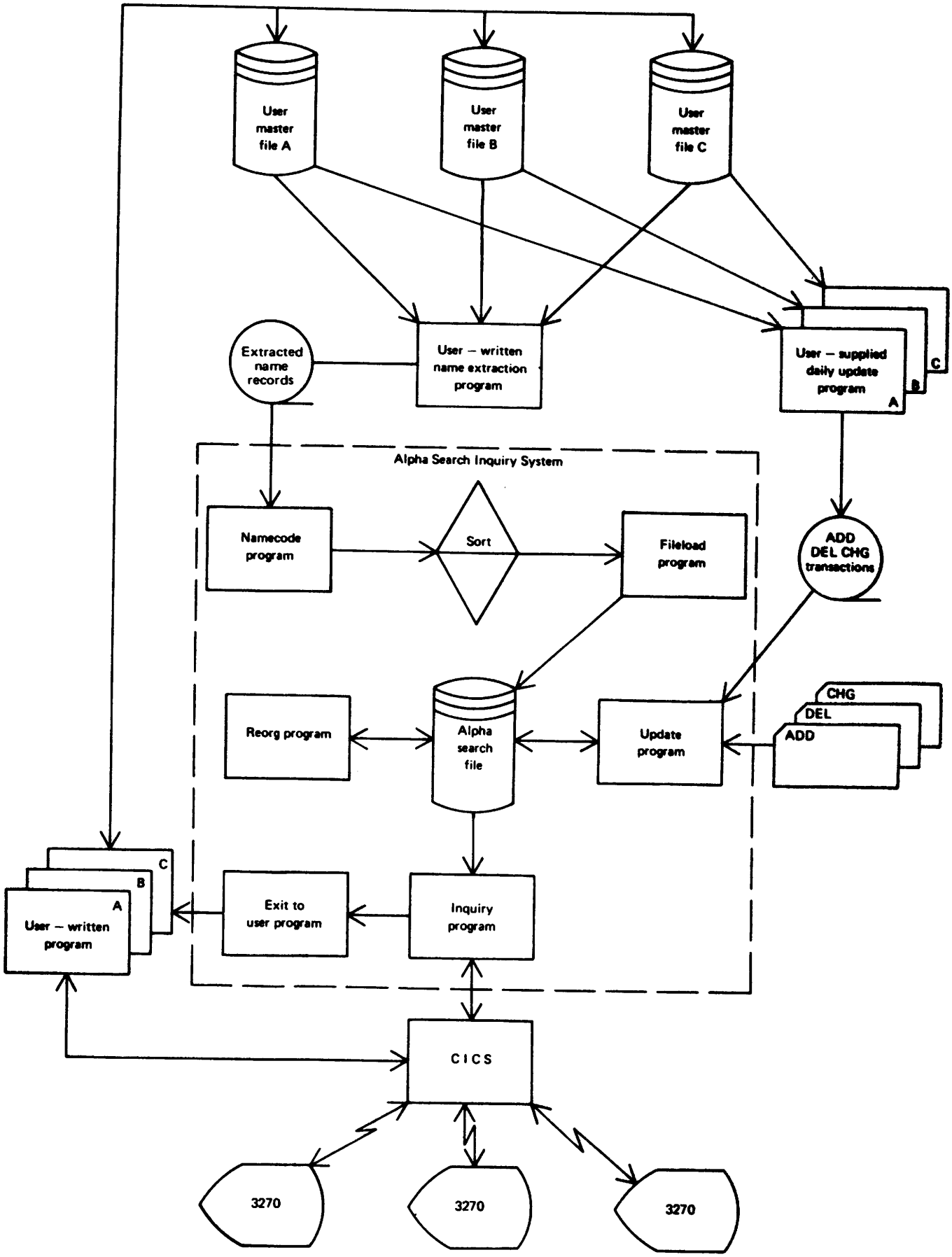


Figure 9. General system chart

Author: T.Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 1.0 Name: Description: Retrieve Alpha Search Records System Overview

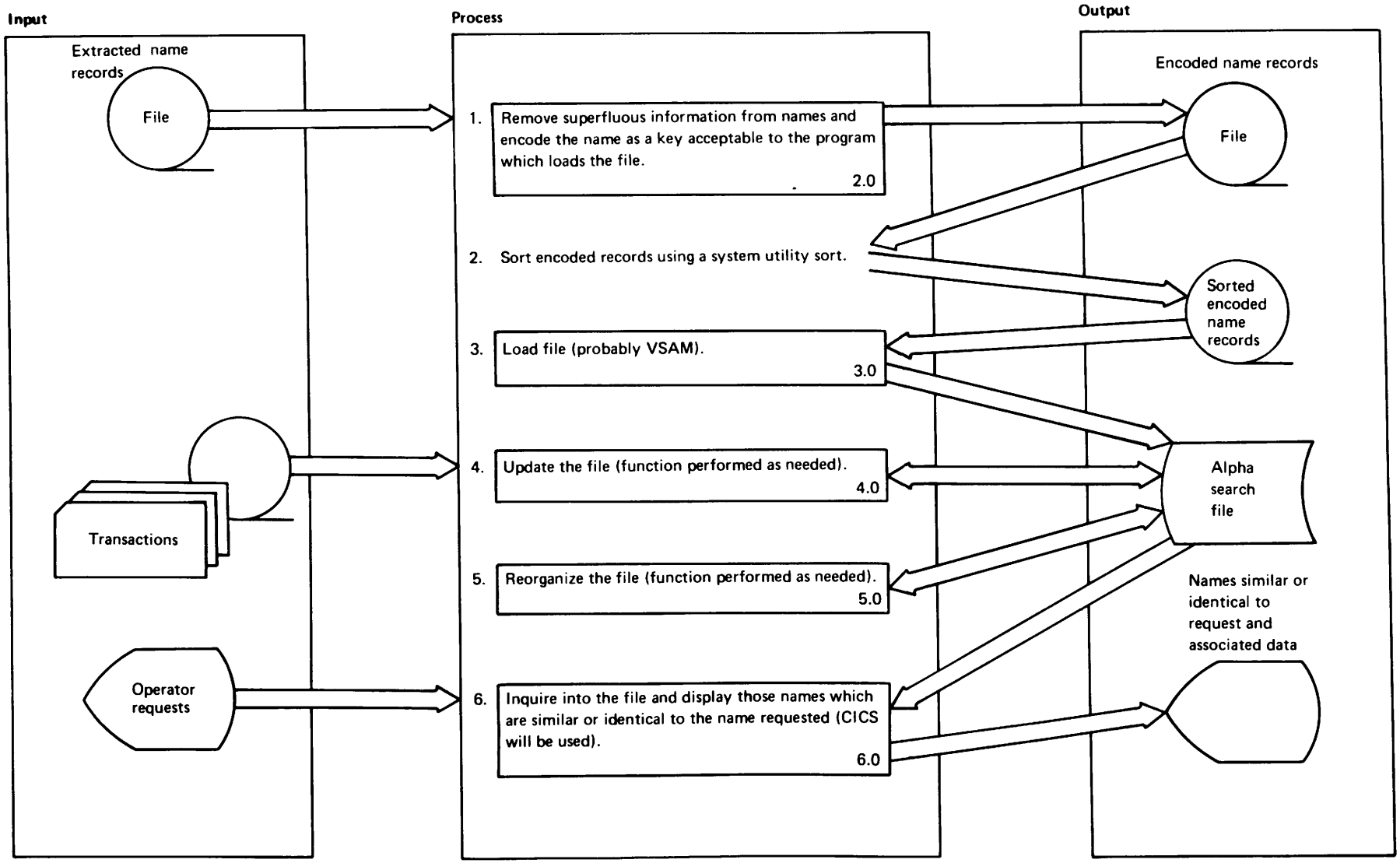


Figure 10. Alpha Search Inquiry System overview diagram (1.0)

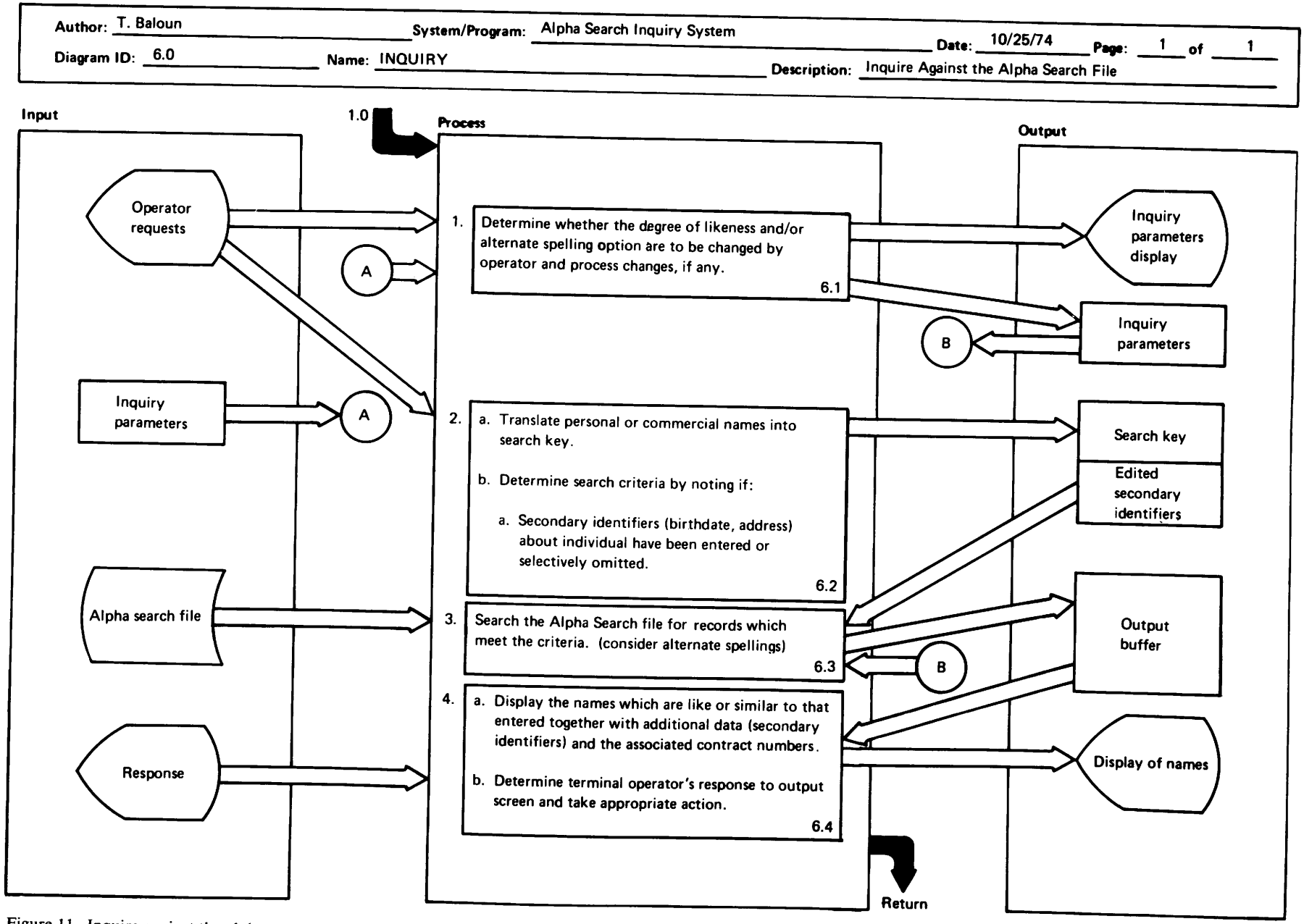


Figure 11. Inquire against the alpha search file overview diagram (6.0)

THIS PAGE INTENTIONALLY LEFT BLANK

Detail HIPO Diagrams

The overview and detail diagrams are graphically similar, but the purpose of the detail diagram is to provide all the information necessary to understand a function specified in the next-higher-level diagram.

Figure 12 is one of the detail diagrams (6.3) of the inquiry function; it describes the search alpha search file subfunction. As in the overview diagrams, the detail diagram consists of three boxes: input, process, and output. However, the information given is changed. In Diagram 6.0 (the overview diagram of the inquiry function) the inputs and outputs are files and terminal displays. Note that in Diagram 6.3 the inputs and outputs are a subset of files and terminal displays, giving a more detailed view of each. The discussion of the process steps of Diagram 6.3 below shows how the overview information from Diagram 6.0 has been expanded.

Step 1: The function performed is to search for all records containing the search key. The search key (created in Diagram 6.2 as can be seen in Diagram 6.0) and the alpha search file are input. An alpha search record is output.

Step 2: This function, to provide a means for limiting the duration of a file search and to provide the operator option to continue or terminate processing, is simply stated at this level of design. No specific formats of input or output have been designed at this time.

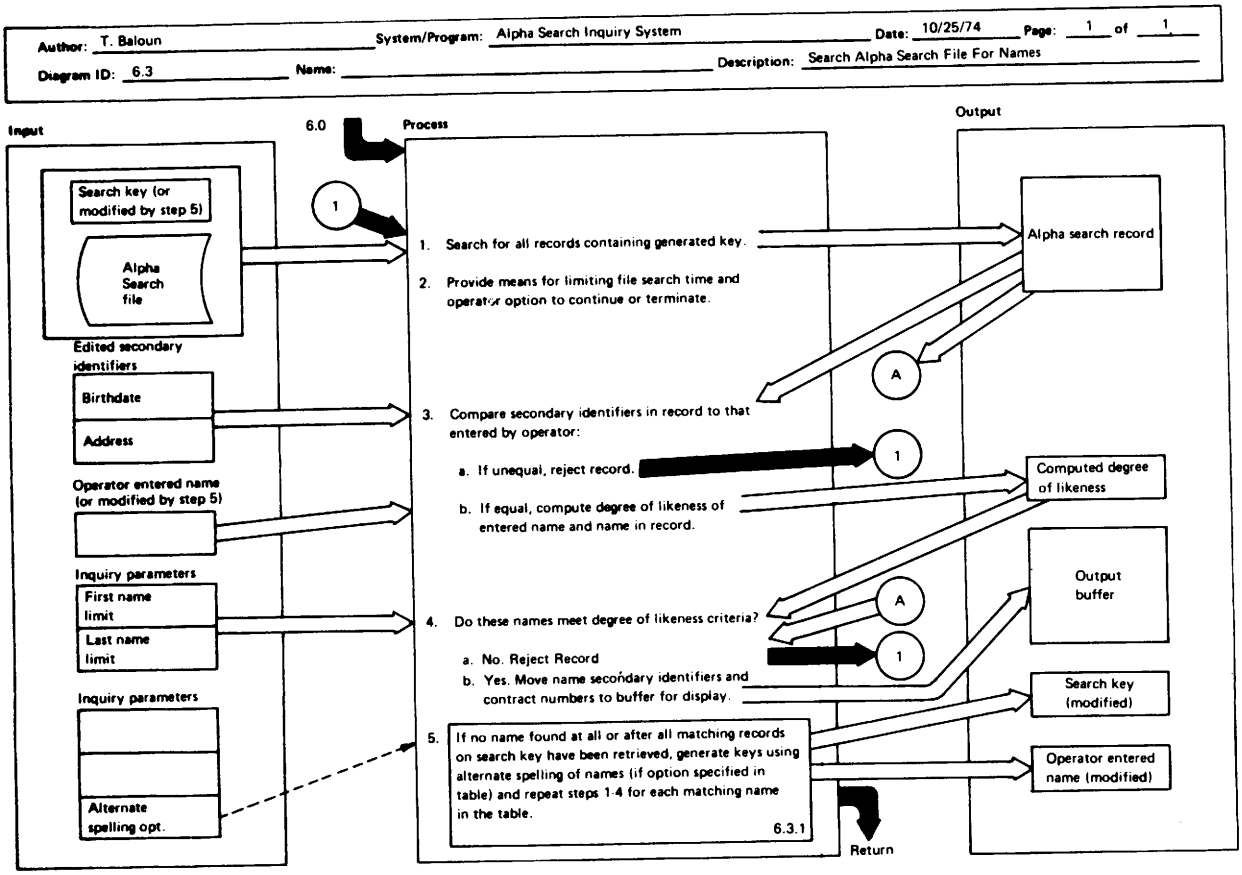
Step 3: The function performed is to compare secondary identifiers (operator-entered vs. those on the alpha search record). The operator-entered secondary identifiers were previously edited in Diagram 6.2. The first two fields of the inquiry parameter table are the only ones used and are therefore the only ones noted in the box. The degree of likeness between the entered name and the name on the file is computed.

Step 4: If the degree of likeness meets the criteria stated in the inquiry parameter table, information from the alpha search record (name, secondary identifiers, and contract numbers) is moved to a buffer for display.

Step 5: After the file is searched, by use of the search key, for any matching records, another search is performed using alternate spellings of the entered name, provided this option is specified in the inquiry parameter table. The alternate spelling option is the only one referenced in this step and is therefore the only one shown in the box. The dotted line arrow indicates that the value is tested to determine the action to be taken. This function is further explained in Diagram 6.3.1.

After all records (or a page of records) containing the the generated keys have been retrieved, control is returned to the next step in Diagram 6.0, to display the records. This function is expanded in Diagram 6.4.

In addition to the input, process, and output sections on a HIPO diagram, there is the optional extended description section (see Figure 12), which is used to give the reader more detailed information than can be put in a process step. For the initial design HIPO package, this consists primarily of notes on possible implementation.



Extended Description

Notes	Routine	Label	Ref.
2. The CICS time option should be used for limiting the file search time.			
3b. The name entered by the operator is compared to the name on the Alpha Search file letter by letter. After that, the degree of likeness is computed by dividing the number of letters in the longer of the two names into the number of letters which were equal.			
4. If the computed number is less than the number in the inquiry parameters the names are accepted.			
5. Before implementation of this step a table will be created containing alternate spellings.			

Extended Description

Notes	Routine	Label	Ref.

Figure 12. Search alpha search file for names detail diagram (6.3) extended description

Steps in Creating HIPO Diagrams

The following paragraphs suggest an approach to drawing both overview and detail diagrams.

1. Use the reverse side of the HIPO Worksheet to draw the three boxes: input, process, and output.
2. List any known output in the output box. It is a good practice to list output prior to filling in the input and process boxes. Many of these can be set down immediately from the system requirements. In the overview diagrams, the output is stated in general terms, that is, files. Actual devices need not be shown at this time.
3. Develop the contents of the input and process boxes concurrently and fill in any intermediate output not previously specified.
4. Illustrate only the bare necessities on the input side. If something does not apply, do not write it down. If in doubt, include it; anything that is not necessary can be deleted when the diagrams are refined. Again, for the overview diagram, the input is stated in general terms.
5. State what has to be done in the process box. This is simply a functional listing of the processes to be accomplished. State each function in as few words as possible.
6. Connect the corresponding input items and the process steps with arrows; connect the corresponding process steps and the output items with arrows. Since the result is only a preliminary diagram (see Figure 13), do not be discouraged if it seems confusing.
7. The next step is to turn the preliminary diagram into a completed HIPO diagram, as shown in the HIPO diagram in Figure 11. This is a process of consolidating and combining related data items into logical groupings with boxes. Boxes within boxes are certainly allowed, but for graphical and practical reasons use only what is necessary. The simpler the chart, the more effective it is. The last step is to draw the arrows. The shaded arrows show entry to and exit from the diagram.

Some of the problems that may be encountered in early efforts are:

- Putting ideas down concisely
- Deciding what functions can be broken down into subfunctions
- Trying to put too much into one diagram rather than creating lower-level diagrams
- Trying to do low-level diagrams without first doing higher-level diagrams
- Thinking in terms of program logic rather than function

A technique used in this package in the solution of the second and third items was to limit a diagram to one page; otherwise it was subdivided. One page (including no more than 7-10 steps) is easier to comprehend than multiple pages. It is not always easy to follow this, but it pays both in reducing review time and in assisting in the subsequent development phase.

Some visually important points in developing HIPO diagrams are:

- Arrows generally go from left to right adjacent to the referenced or connected items, and from items higher on the page to those lower on the page.
- Where the tail of the arrow starts and where the tip of the arrow ends must be clear.
- Arrow types must be be consistent with the legend.
- Boxes should contain only functionally related items.
- Abbreviations should be avoided in overview diagrams.
- All terms and labels should be defined in the extended description when first used.
- Data in overview diagrams should be generally described to minimize arrows.

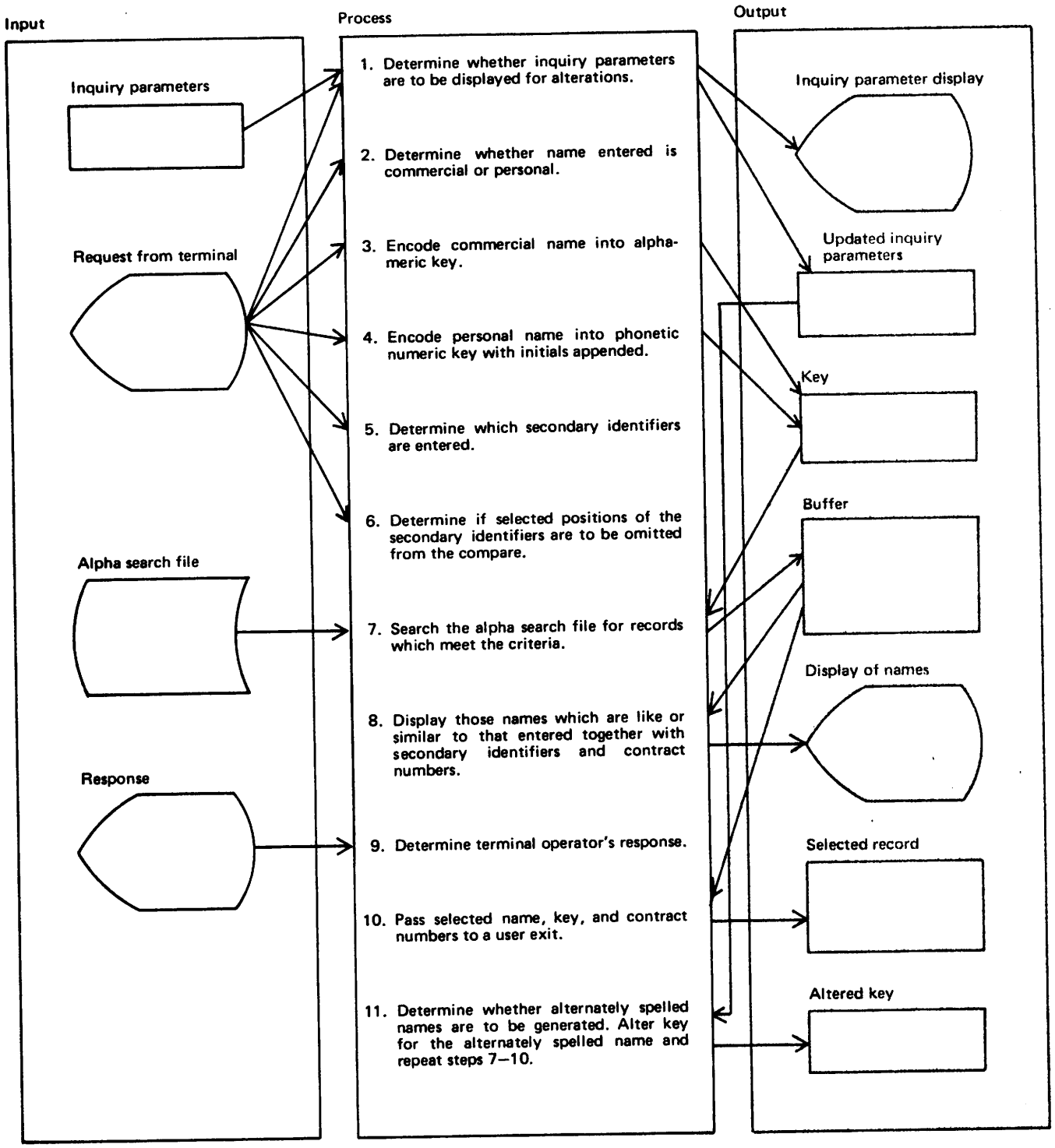


Figure 13. A preliminary diagram of the inquiry function

Reviews of the Initial Design HIPO Package

With the HIPO technique, a system can be reviewed much earlier in the design phase than with flowcharts or text, because the system is being specified and documented level by level from general to increasing amounts of detail.

The reviews of the HIPO package can be both informal and formal and can involve people whose review purposes are different. For most of these reviews the diagrams need not be in final form; they need only be legible. Line arrows can be used at this time.

The first review will probably have as an audience an analyst or programmer. He should look for:

- Content
  Is the visual table of contents in a hierarchy structure?
  Is the diagram at one level described by the collection of diagrams beneath it?
  Are the major input and output items at each level a subset of the input and output items of the level above?
- Consistency
  Are the symbols, arrows, and graphic techniques consistent? (Some possible conventions are discussed in the section entitled "Sample HIPO Conventions".)
  Is the terminology consistent and understandable? Data items should be given the same name throughout. Acronyms and terms should be defined. The analyst or programmer may not be the best judge of this, since data processing jargon completely understandable to him but not to everyone may have been used.
- Technical accuracy—The analyst or programmer may not be as familiar with the user requirements as others but he can determine:
  Is it clear what specific output is produced by a given step?
  Is the function described in specific terms or is it vague?
  Are input or output items missing?

The second review may be for those knowledgeable in the functions of the system being designed. The reviewers, who are probably not members of the data processing department, are interested primarily in:

- Functional accuracy
  Is the function correctly stated?
  Has anything been added or omitted?
  Are the input and output items correct?

While functional accuracy as related to the user's requirements is the reviewer's primary concern, problems with transition, consistency, or terminology may arise. How easily these reviewers can understand and comment on the package determines whether problems exist in these last three areas.

In all reviews, the author should let the documentation stand on its own by allowing the reviewers to study the visual table of contents and HIPO diagrams unassisted. The HIPO package probably needs correction if the reviewers need repeated explanations.

Additional reviews of particular sections may be required by experts in those functions. The reviewers are assisted by the visual table of contents and overview diagrams, which may be used to gain a general understanding of the system.

Uses of the Initial Design HIPO Package

The initial design HIPO package has many uses both for the data processing department and for people who interface with that department. It can be used by:

- A manager within or outside data processing, to obtain an overview of the system
- Design reviewers (both data processing and user departments), to determine whether the design is feasible and to ensure that it does what is intended
- Data processing management, to define the scope of the project
- Programmers or analysts, to do the detail design for programming purposes
- Data processing management, to educate new systems analysts or programmers as they are added to the project

Summary

This section has shown how the HIPO technique can be used throughout the requirements and initial design phase of system development. Using a case study as an example, a statement of requirements was developed into a visual table of contents and overview and detail HIPO diagrams. How the HIPO package can be reviewed and used has also been discussed. (See Appendix A for the complete case study initial design HIPO package.)

This initial design package serves as input to the next step, the detail design and implementation phase, where the existing diagrams are expanded and more detail diagrams are added. At each stage of development, the previously developed HIPO package is used as input and expanded upon.

Chapter 3: Detail Design and Implementation Phase

The objectives of the detail design and implementation phase are to:

- Expand the requirements and initial design documentation into a detailed document from which code can be written
- Assign functions to modules of code
- Develop the code
- Design and execute test cases
- Create operators' instructions

The input to this phase is the initial design HIPO package. These diagrams are expanded and more detail diagrams are drawn until they are sufficiently detailed for coding. As additional personnel are added to the programming effort, they must be taught the proposed system. Data processing managers must also make programming assignments and track the development of the program. The HIPO package in this phase can serve both as an educational tool and as a management aid to define the scope of the projects. When the detail design and implementation phase is complete, the HIPO package should provide clear, accurate, detailed documentation for any future maintenance required on the system.

In the following sections, the initial design HIPO package for the case study (Alpha Search Inquiry System) is expanded to the detail design level for certain subfunctions of the inquiry function. The functions of create encoded name record, build the alpha search file, update the alpha search file, and reorganize the alpha search file have not been expanded beyond the initial design level.

Expanding the Visual Table of Contents

The function of the visual table of contents remains the same as in the requirements and initial design phase: to show the hierarchical structure of the system and to provide a directory of the documentation for that system. The expansion of the visual table of contents takes place as more detail diagrams are added.

If it is impossible to contain the entire hierarchy of the system on one page, the first visual table of contents can refer to other visual tables of contents and so on until the entire system is included. This is illustrated in Figure 14, which contains two visual tables of contents for the detail design phase of the case study. When an entire system is implemented, there will be a hierarchy of visual tables of contents, each describing an individual program.

During the requirements definition and initial design phase, the application functions are stated. However, many of the functions specifically related to the programming effort are not included. During detail design both the application and the program functions are specified and shown in the visual table of contents and HIPO diagrams.

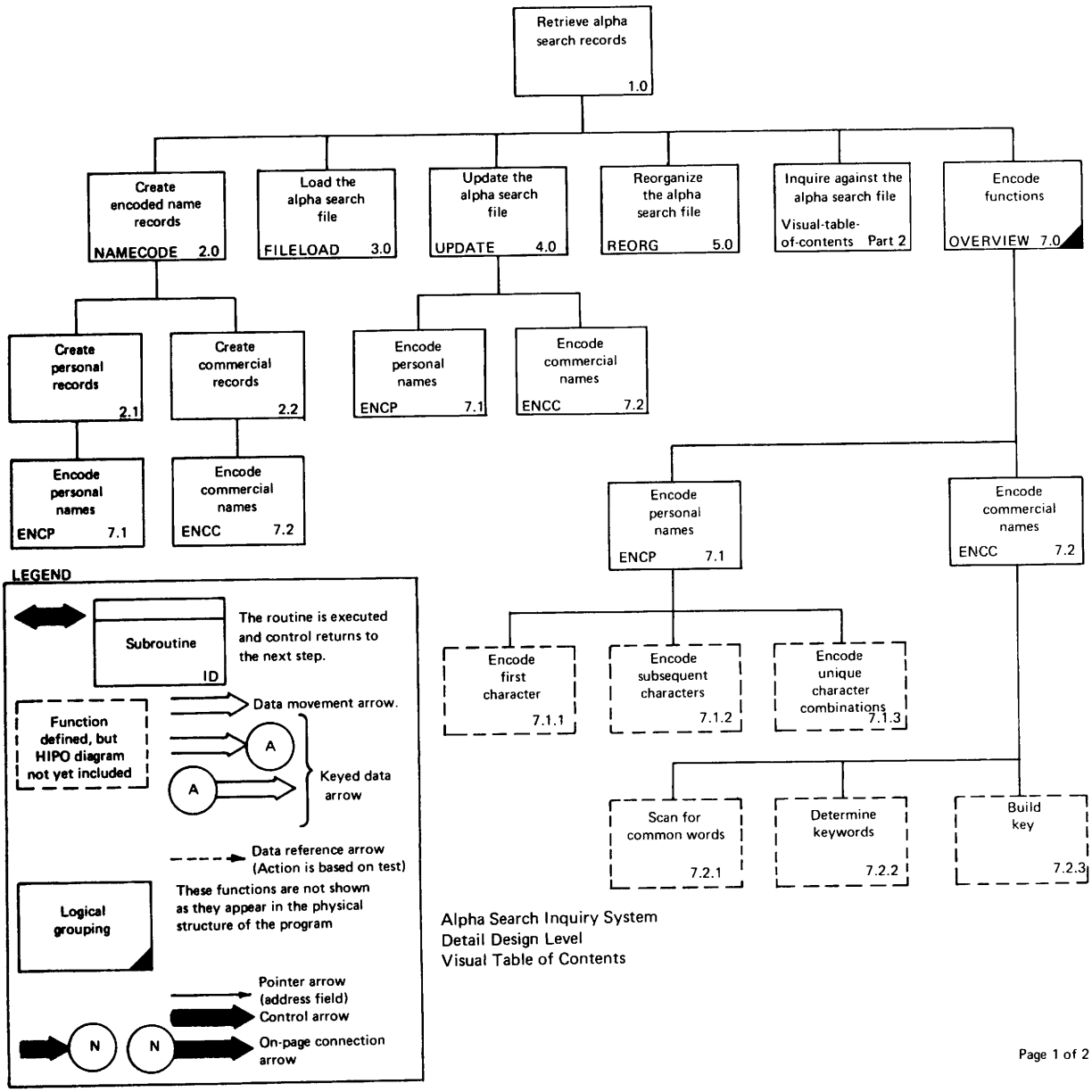


Figure 14. The detail design level visual table of contents of the Alpha Search Inquiry System (1 of 2)

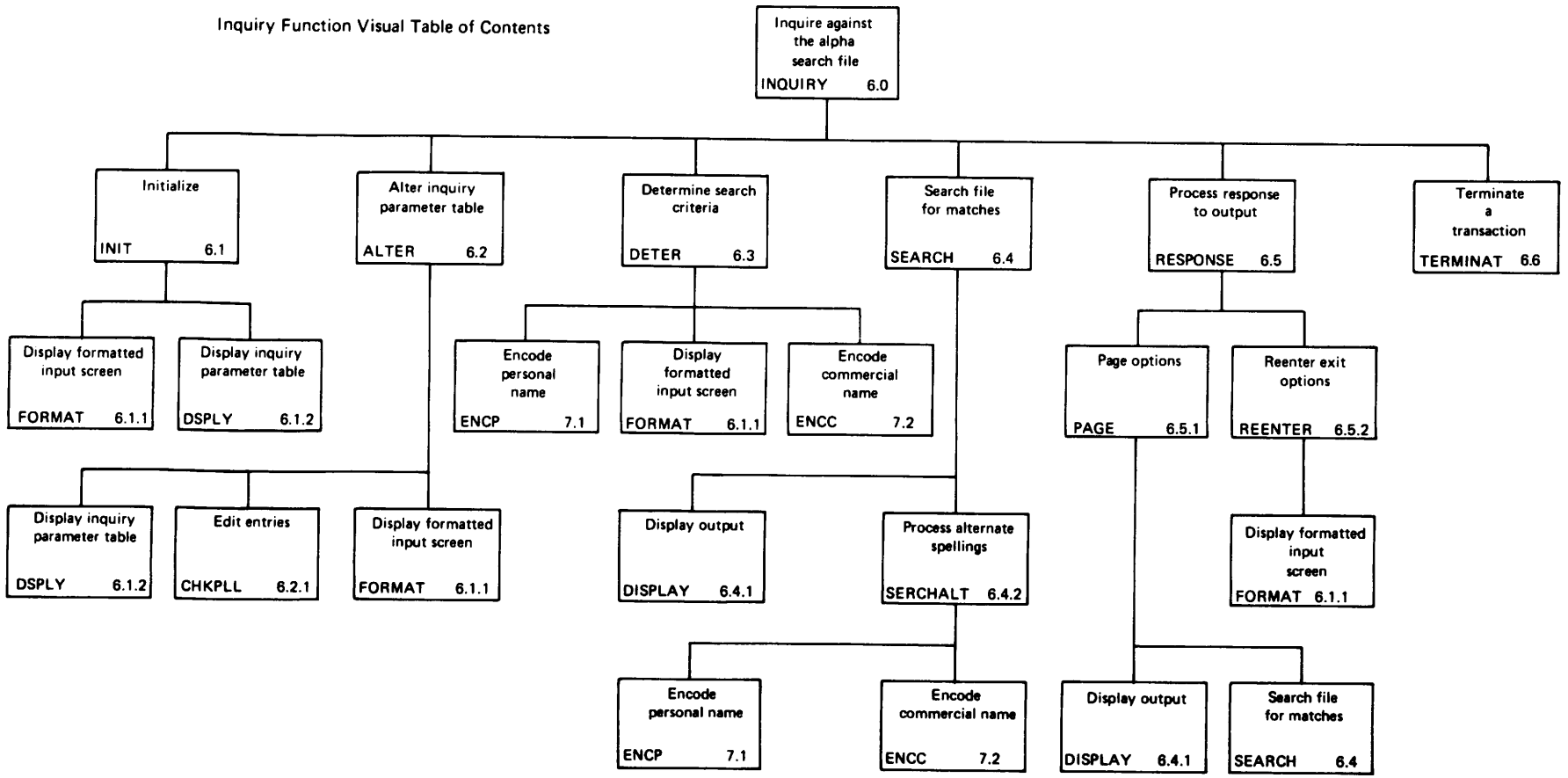


Figure 14. The detail design level visual table of contents of the Alpha Search Inquiry System (2 of 2)

The detail design level visual table of contents of the Alpha Search Inquiry System (see Figure 14) reflects the changes made in structure as programming decisions were made. The initialize (Diagram 6.1) and alter inquiry parameter table (Diagram 6.2) functions of the inquiry program have been fully expanded. The remaining functions of the inquiry program have been expanded one level to show the changes. Some of these changes were due to programming considerations; others were needed to specify interaction with CICS as a terminal-oriented system. Also, when expanding the initial design package, some more logical arrangements of function became apparent. For example, the display names function was found to fit better as a subfunction of the search alpha search file for names function. This type of rearrangement is to be expected as actual program structure is determined.

Another structural change in the detail design level visual table of contents reflects a logical grouping of functions (box 7.0). Certain functions may be related logically even though they are referenced both collectively and independently within the the program structure. For example, encode personal name and encode commercial name are a logically related group of functions, each of which has a separate HIPO diagram. They may be referenced collectively by referring to Diagram 7.0, or individually by referring to each diagram identification number. This centralization of related functions allows easier location without searching down other functional paths. To indicate that box 7.0 is a logical grouping of functions rather than a reference to a diagram depicting the physical program structure, the lower right-hand corner of the box is shaded (see Figure 15).

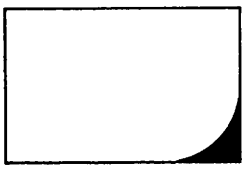


Figure 15. Box showing a logical grouping of functions

A second reason for the logical grouping is to minimize the congestion in the visual table of contents. Both of the encode functions have lower-level diagrams further describing the functions. Where these encode functions are referenced (under boxes 2.1, 2.2, 4.0, 6.3, and 6.4.2), only the highest-level identification number of the encode function (7.1 or 7.2) need be referenced; it is implied that the lower-level diagrams are also included in this reference. Thus the main functional paths in the visual table of contents are not obscured.

As the detail design is being developed, some functions may be identified as requiring further expansion in a lower-level diagram. When they are identified they may be shown in the visual table of contents enclosed in a dashed-line box until included in the package. The use of this dashed-line box is shown in Figure 14; those subfunctions of encode functions not fully expanded for the purpose of the case study are identified in this manner.

The shaded-corner and dashed-line boxes are conventions used in this HIPO package. When such conventions are used, they should be fully explained in the legend.

For any further maintenance effort, the hierarchy diagram provides a functional description of the system as it is programmed, as well as a directory to the diagrams explaining the various processes.

Expanding and Adding HIPO Diagrams

In the detail design and implementation phase, the initial design detail diagrams are expanded and rearranged as necessary to specify the sequence of execution of the functions to be contained in the units of code. Not all functions are broken down to the same level in the hierarchy, as some functions are necessarily more complex than others.

The inquiry function overview diagram of the Alpha Search Inquiry System has been expanded to the detail design level (see Figure 16). The steps of the initial design overview (see Figure 11) have been altered, new steps added, and more specific I/O items included. Actual names have been given to modules, fields, tables, switches, and so forth, and should be used consistently throughout the package. The extended description has been expanded to include the following items:

- Additional information about the processing shown in the corresponding step
- Additional details about data items
- Implementation information needed for code development
- Program labels for code
- References to lower-level HIPO diagrams or non-HIPO documentation

Note that the process step should not be restated in the extended description; the reasoning for the step should.

When expanding an initial design package, many options must be considered and decisions made in order to proceed with the detail design. Below are some activities essential to the detail design effort; also included are references to the detail design HIPO diagrams where they are applicable. (See Appendix B for the detail design HIPO package.) Note that the high-level diagrams of the functions of the inquiry program have been included in the HIPO package to show the rearrangement of these functions. Only the initialize (Diagram 6.1) and alter inquiry parameter table (Diagram 6.2) functions have been fully expanded.

- Select the data management technique and teleprocessing system to be used. CICS is used in the implementation of the inquiry function. Many references to CICS facilities and macros are made throughout the HIPO package. (Diagram 6.1.2, steps 2 and 5, and Diagram 6.1, steps 1 and 3, are examples.)
- Design the data layouts to interface with and meet the requirements of the data management technique, teleprocessing system, and specific devices used. A layout of the transaction work area (TWA) is included as a Figure (6.0A) in the detail design package and is referenced in the extended description sections.
- Include programming activities of a housekeeping nature, that is, open and close files, acquire storage, etc. The initialization and termination routines (Diagrams 6.1., step 1, and 6.6, steps 1 and 2) have been added to the HIPO package.
- Break up major functions so they can be fully understood.

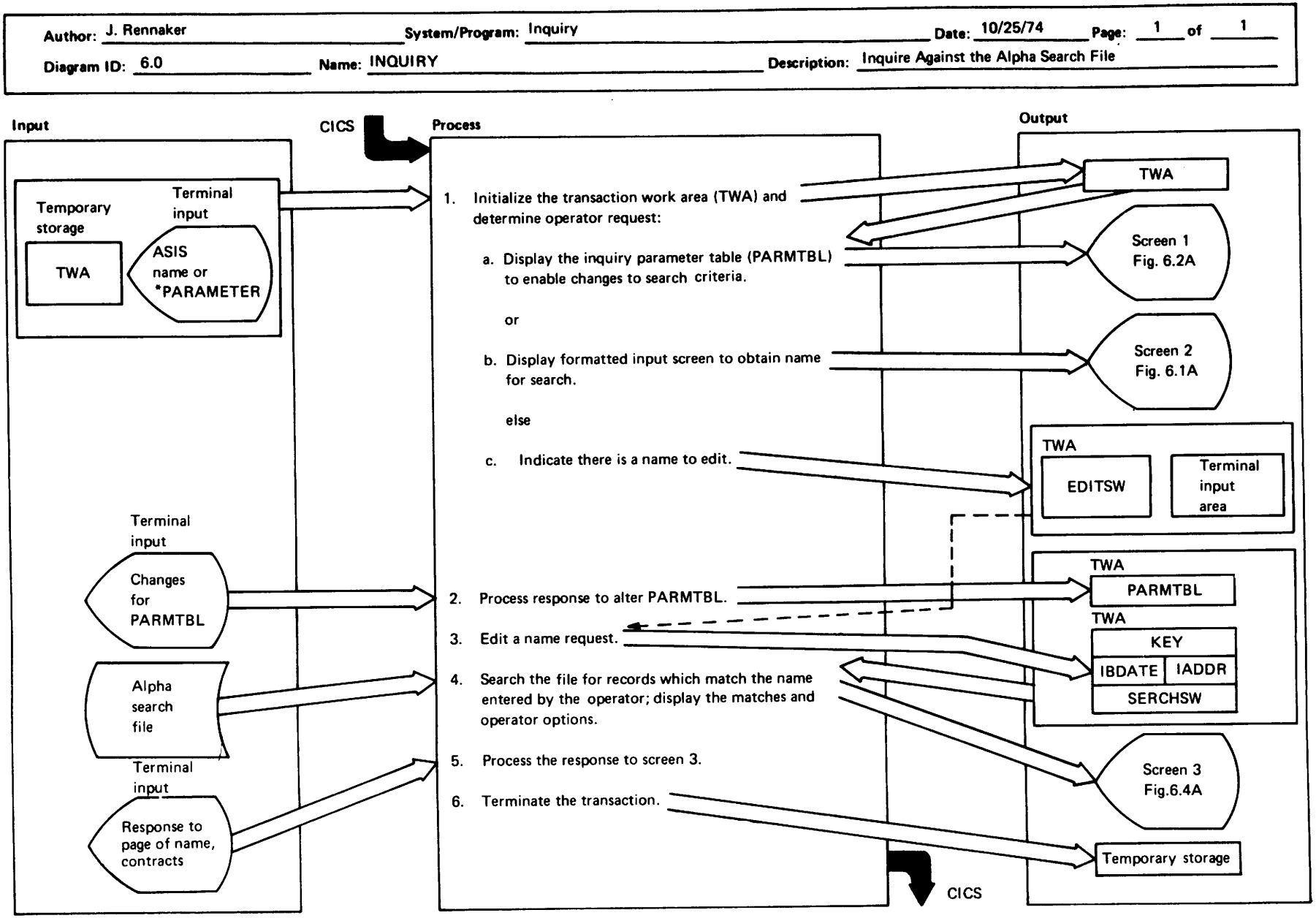


Figure 16. Inquire against the alpha search file overview diagram (6.0)

Extended Description

Notes	Routine	Label	Ref.
All files are opened by CICS. The TWA contains	INQUIRY		6.0A
all transaction dependent data.			
1. Temporary storage will contain the TWA by	INQUIRY	INIT	6.1
terminal ID if this is a continuation of a			Fig. 6.1A
transaction.			
a. The operator may wish to change the			
first or last name degrees of likeness or the			
alternate spelling option which are the			
search criteria in the PARMTBL-ALTERSW			
and CICSRET are then set on.			
b. RFORMSW and CICSRET are set on.			
2. This routine is called if CICSRET is off and	ALTER		6.2
ALTERSW is on.			
3. This routine is called if EDITSW is on. The	DETER		6.3
name is phonetically encoded to build a key			
SERCHSW is set on if there are no edit errors.			
4. The search is performed if SERCHSW is	SEARCH		6.4
on until end of limits for original key and			
alternate spellings or end of time or full			
buffer. The names and options of: PAGE			

Extended Description

Notes	Routine	Label	Ref.
FORWARD, PAGE BACK, PAGE START,			
REENTER, EXIT are displayed; RESPSW			
and CICSRET are set on.			
5. This routine is called if CICSRET is off	RESPONSE		6.5
and RESPSW is on.			
6. Set off CICSRET. The TWA is saved in	INQUIRY	TERMINAT	6.6
temporary storage to enable processing by			
the proper routine after CICS returns			
control to INQUIRY with the operator			
response.			

Figure 16. Inquire against the alpha search file overview diagram (6.0) extended description

- Include the details of input/output items, such as operator entries, not detailed in the initial design. As an example, Diagram 6.1 (steps 3A and 3B) has defined the operator entry.
- Include layouts of files, records, fields, tables, switches, control blocks, screens, etc., in the HIPO package. For example, Figures 6.1A, 6.2A, and 6.4A in Appendix B are designs of display screens to be used and Figure 6.0A is a layout of the CICS transaction work area (TWA), which contains, among other things, the many switches used by the various modules.
- Define program labels for all input/output items and modules so they can be used as "shorthand" notation consistently throughout the package. A good example of this need is the many switches required by these modules. They are named and defined in the TWA.
- Expand error processing not included in initial design. For example, Diagram 6.1.2, step 3, references error switches and messages.

Not included in a HIPO package but also stated during this phase of development are such items as performance requirements and run book information, that is, computer and terminal operator instructions, job control, etc. These items, plus the detail design HIPO package, provide complete documentation for any further maintenance of the system.

Reviews of the Detail Design HIPO Package

The reviews of the detail design HIPO package are a continuation of the type of reviews performed during the initial design and requirements phase. Reviews, both formal and informal, should be conducted throughout this phase and should involve many people with varying expertise. The points discussed under "Reviews of the Initial Design HIPO Package" are also applicable in the reviews of the detail design package. The following items are also of major importance:

- It is crucial that programs, files, data items, tables, and so forth, be given appropriate names that are used consistently throughout the package. Particular attention should be given to the input and output interfaces.
- All detail diagrams should be subsets of the next-higher-level diagram both in functional content and major input and output items.
- Detail diagrams should be compared to initial design diagrams to ensure that nothing has been added, omitted, or inadvertently changed.
- Code should be compared to detail diagrams to ensure that the function is being properly performed.

The links provided by HIPO from requirements to initial design, detail design, and subsequently to code have not typically been available in system design. In the past, requirements have been rewritten into programming statements; much of the information gathered during design has been thrown out or is not properly organized once the system is operational. HIPO provides a means of consistently preserving all the information, both general and detailed, gathered in the process.

Uses of the Detail Design HIPO Package

The detail design HIPO package has many uses both for the data processing department and for people who interface with that department. It can be used by:

- Data processing management, to educate new analysts and programmers as they are added to the project
- Programming management, to make programming assignments (they can assign overview diagrams to programmers for development detail of additional diagrams and corresponding code)

- Reviewers, to determine the consistency between the initial design package and the detail design package, and the accuracy of the code as compared to the detail package
- Programmers, to ensure that the data interfaces between programs are consistent
- Programmers, to record standard naming conventions, error message numbers, record layouts, etc.
- Analysts and programmers, to design test cases (the input/output items described in the diagrams aid in this task).

Summary

This section has shown how the initial design HIPO package for the case study was expanded to become the detail design package. Certain subfunctions of the inquiry function were fully expanded to the detail level. Reviews and uses of the HIPO package have also been discussed. A detail design package should provide clear, accurate, detailed documentation for any future maintenance required in the system. (See Appendix B for the detail design HIPO package for the Alpha Search Inquiry System.)

Chapter 4: Maintenance Phase

The objectives of the maintenance phase are to:

- Correct any errors in the developed system as rapidly as possible to minimize disruption
- Add functions to the developed system
- Make corrections or additions without disrupting the system structure to ease future maintenance

The programmers responsible for maintaining a system are often not those who were involved in its design or implementation. In many cases the designers and implementers are not available for consultation on details, or so much time has elapsed since the original implementation that total recall is improbable. Yet the maintenance programmers must be taught the system. For large systems this is a difficult task. To learn an entire system well enough to be prepared in advance to look at any segment of code in which an error is found may be an unrealistic goal. What is needed instead is a means of locating the code requiring changes without necessarily knowing the whole system in detail. This section discusses how the HIPO package is used for this purpose.

Reviews of the Maintenance HIPO Package

The detail design HIPO package is input to the maintenance phase. Reviews of the maintenance phase HIPO package may be conducted for the following three purposes:

1. To educate new personnel, thus ensuring their understanding of the system.
2. To check the HIPO package for clarity, legibility, and completeness. The logic paths must be easily followed and understood. Also, all last-minute changes must be properly reflected in the diagrams, particularly in the extended description sections.
3. To optionally delete some low-level diagrams, thus reducing maintenance of this HIPO package.

Since code now exists for the system, some of the lowest-level diagrams containing specific instructions for implementation may not be required for maintenance activities. References may be made directly to the code from the extended descriptions in higher-level diagrams. Some factors affecting the decision to keep or delete these diagrams are as follows:

- Does the HIPO diagram provide an understanding of the "why" of the function not available in the code alone? This can be seen during the reviews with the maintenance personnel. Also evident at this time is the diagram level at which the reviewers feel sufficiently knowledgeable about function to begin looking at the code.
- Is the code readable?
- Does each segment of code have a straightforward relationship to the function on a HIPO diagram? If the implementation of a function is scattered, the diagram should be kept to describe the function in its entirety and the extended description should serve as a directory to the various segments of code.
- How detailed are the lowest-level diagrams? During implementation some processes may have been expanded to a level of detail where each step in a diagram represents as few as five statements of code. In such cases, the next-higher-level diagram may provide an adequate stopping point. If process steps in a diagram represent 50 statements of code, the diagram probably should be retained.

When a decision is made to delete a diagram, the next-higher-level diagram should be checked to ensure that all the information needed is contained in the extended description, particularly the references to the code.

Uses of the Maintenance HIPO Package

The uses of the maintenance HIPO package are to find and fix program errors and to add new features to the system. These are discussed below, using some diagrams of the case study detail design HIPO package (see Appendix B) as examples.

Finding and Correcting Program Errors

The hierarchy depicted in the visual table of contents should enable the programmer to locate the proper functional path of the programming system. His familiarity with the system will determine the level at which he can begin to trace the path. Both functions and input/output items will probably have to be examined. Consider the following example:

In the case study, assume that the wrong default is being used for the last name limit. The programmer would need to determine what code processes these parameters. Since the parameters are in the table called PARMTBL, step 2 of Figure 16 indicates that Diagram 6.2 should be examined. This diagram describes the functions of altering the PARMTBL values.

Step 3 of Diagram 6.2 describes the parameter processing for this field and step 6 indicates that Diagram 6.1.2, which displays the parameter screen and checks for edit errors, should be referenced. Step 3 of Diagram 6.1.2 describes the process, and the extended description points to the code to be examined. At this point the programmer can refer to the code to determine the problem in establishing the first and last name defaults.

Adding Program Functions

The addition of a new function to an existing system presents the problem of fitting that function within the existing programs and data structure. The HIPO package is used in this activity to identify any areas affected by the proposed change.

Summary

This section has defined the objectives of the maintenance phase and described how the HIPO technique can be used to meet these objectives. It has also provided some guidelines for deleting low-level HIPO diagrams from the detail design HIPO package.

Chapter 5: Sample HIPO Conventions

In a given installation:

- Several designers, analysts, and programmers may be working on one system
- A designer, analyst, or programmer may be working on several systems
- Several people with different professional backgrounds and interests may be reviewing a system

Providing HIPO diagrams that are consistent within a package increases the readability of the entire package; providing consistency within an installation greatly improves the usability of the documentation technique. Each installation should, therefore, develop its own HIPO conventions.

This section provides definitions of frequently used HIPO terms and symbols, sample HIPO conventions, and examples of these. The reader should feel free to adapt these conventions to his particular installation. This section is organized so that references may be made to individual conventions independently. Information relevant to more than one topic appears more than once. The HIPO aids—namely, the HIPO Template (GX20-1971), the Template jacket, and the HIPO Worksheet (GX20-1970)—are shown in Figures 17, 18 and 19 respectively. These aids may be used to prepare HIPO diagrams.

Definitions of HIPO Terms

1. HIPO package—A set of HIPO diagrams containing a visual table of contents, overview HIPO diagrams, and detail HIPO diagrams. It may also contain related non-HIPO documentation, such as file and record layouts, system flowcharts, screen designs, decision tables, or detailed flowcharts.
2. Visual table of contents—The section of a HIPO package containing the names, identification numbers, and descriptions of the HIPO diagrams and showing their functional relationship in a hierarchical fashion. It also contains a legend.
3. Legend—A box on the visual table of contents section defining the symbols used in the HIPO diagrams.
4. HIPO diagram—A graphic representation of the functions or processes of a system. It contains three major sections:
   a. Input—The data items used by the process steps and connected to them by arrows
   b. Process—A series of numbered statements describing a given function. These statements are connected by arrows to the input needed to perform the function and the output created by the function.
   c. Output—The data items created or modified by the process steps and connected to them by arrows
5. Extended description—An amplification of the process steps and input and output data items which identifies modules and labels associated with the process step(s), and references other HIPO diagrams or non-HIPO documentation. Not all steps may require amplification. If no steps of a diagram require amplification, no extended description section is necessary. The extended description always appears with its associated HIPO diagram.
6. Overview diagram—A high-level HIPO diagram that provides a general description of functions and input and output data items described in further detail in lower-level diagrams.

7. Detail diagram—A lower-level HIPO diagram that provides detail of the functions (processes) and shows specific input and output data items making up the higher-level functions.
8. Data item—Any symbol appearing in the input or output area of a HIPO diagram. It may be:
   - Any input/output symbol
   - A box of any size (there are two "data item" boxes provided on the HIPO Template)
9. Data group—Any number of data items enclosed in a single box within the input or output area.
10. Arrow—Any one of the four types of arrows (data movement, data reference, control, pointer) which may appear in a diagram.
11. Input (output) path—The area between the input (output) and process sections of a HIPO diagram which contains the arrows connecting the two sections.
12. Diagram identification (ID)—The HIPO diagram number.

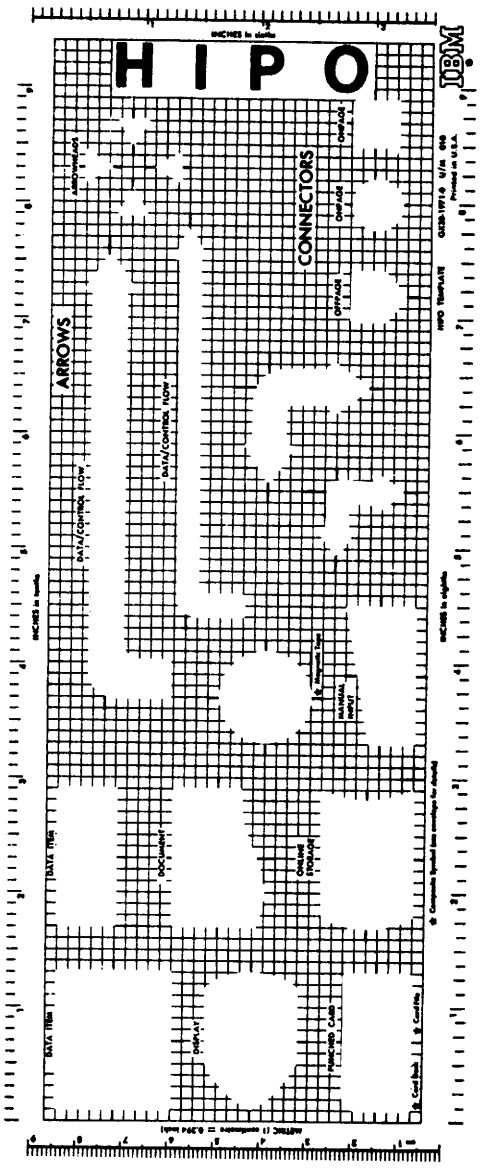


Figure 17. HIPO Template

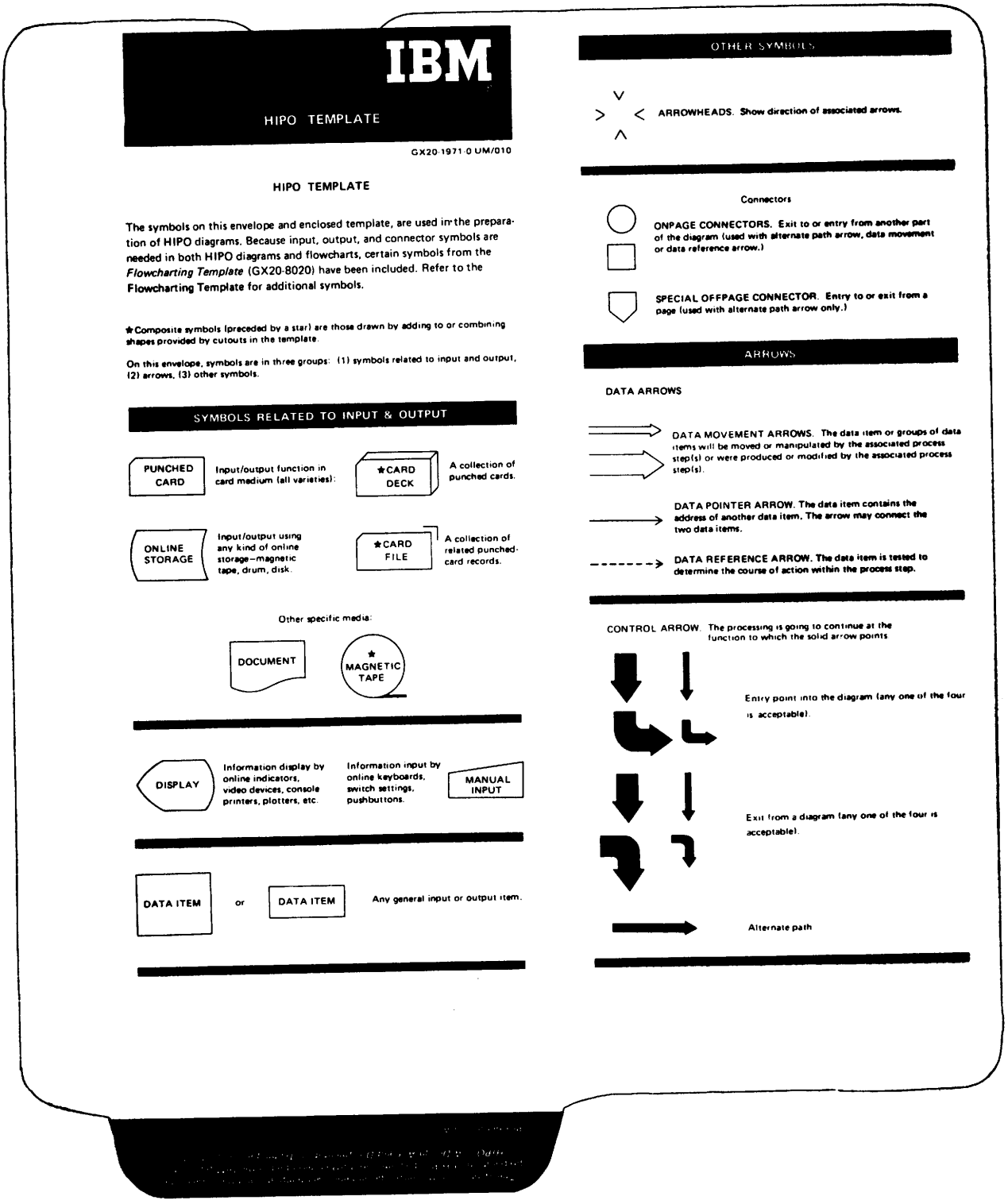


Figure 18. HIPO Template jacket

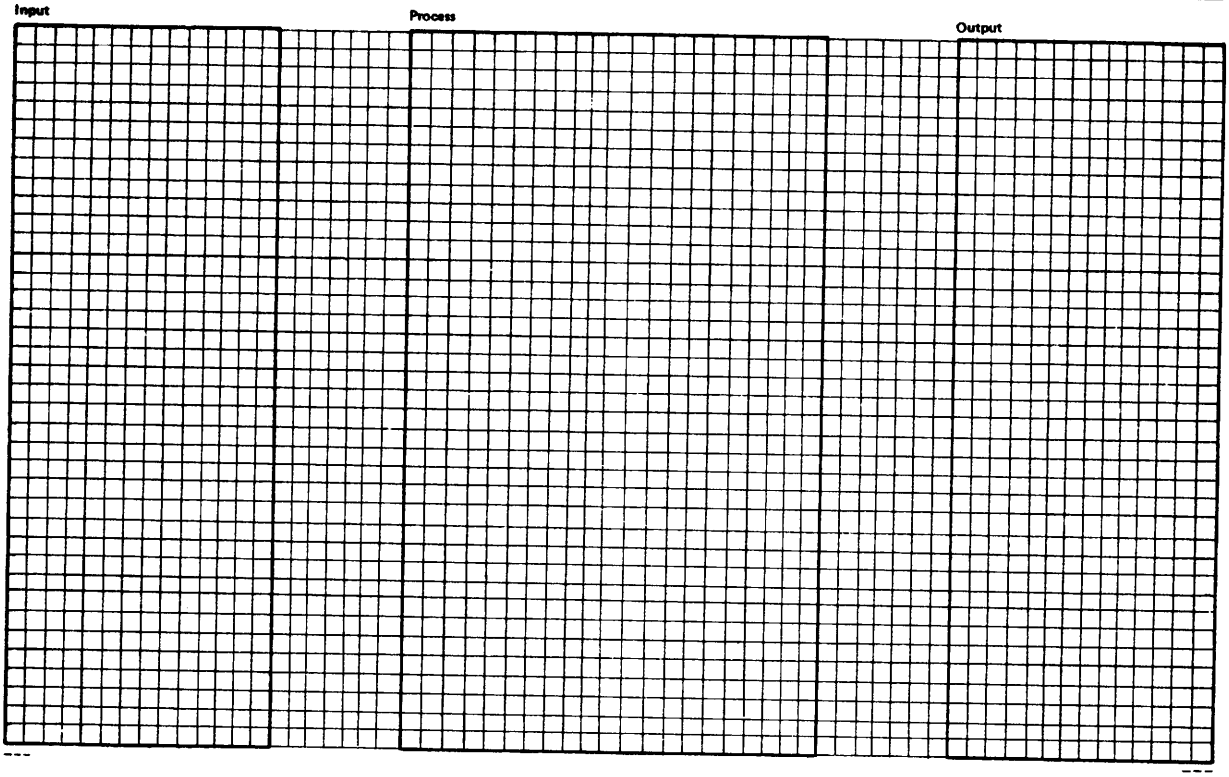
IBM® HIPO WORKSHEET

GX20-1970-0 U/M 025 *
Printed in U.S.A.

Author: _____ System/Program: _____ Date: _____ Page: _____ of _____

Diagram ID: _____ Name: _____ Description: _____

Input | Process | Output



Extended Description

Notes		Ref.

Extended Description

Notes		Ref.

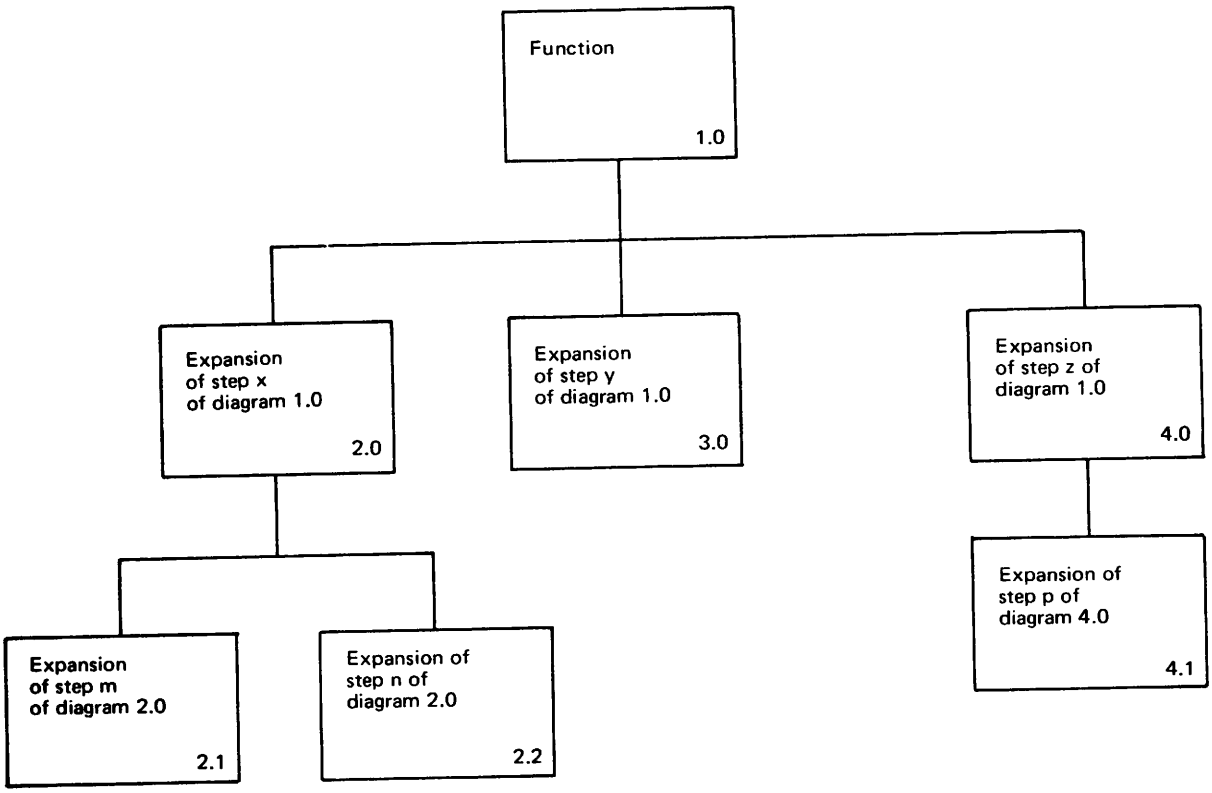
* The number of sheets per pad may vary slightly

Figure 19. HIPO Worksheet

Use of the Visual Table of Contents

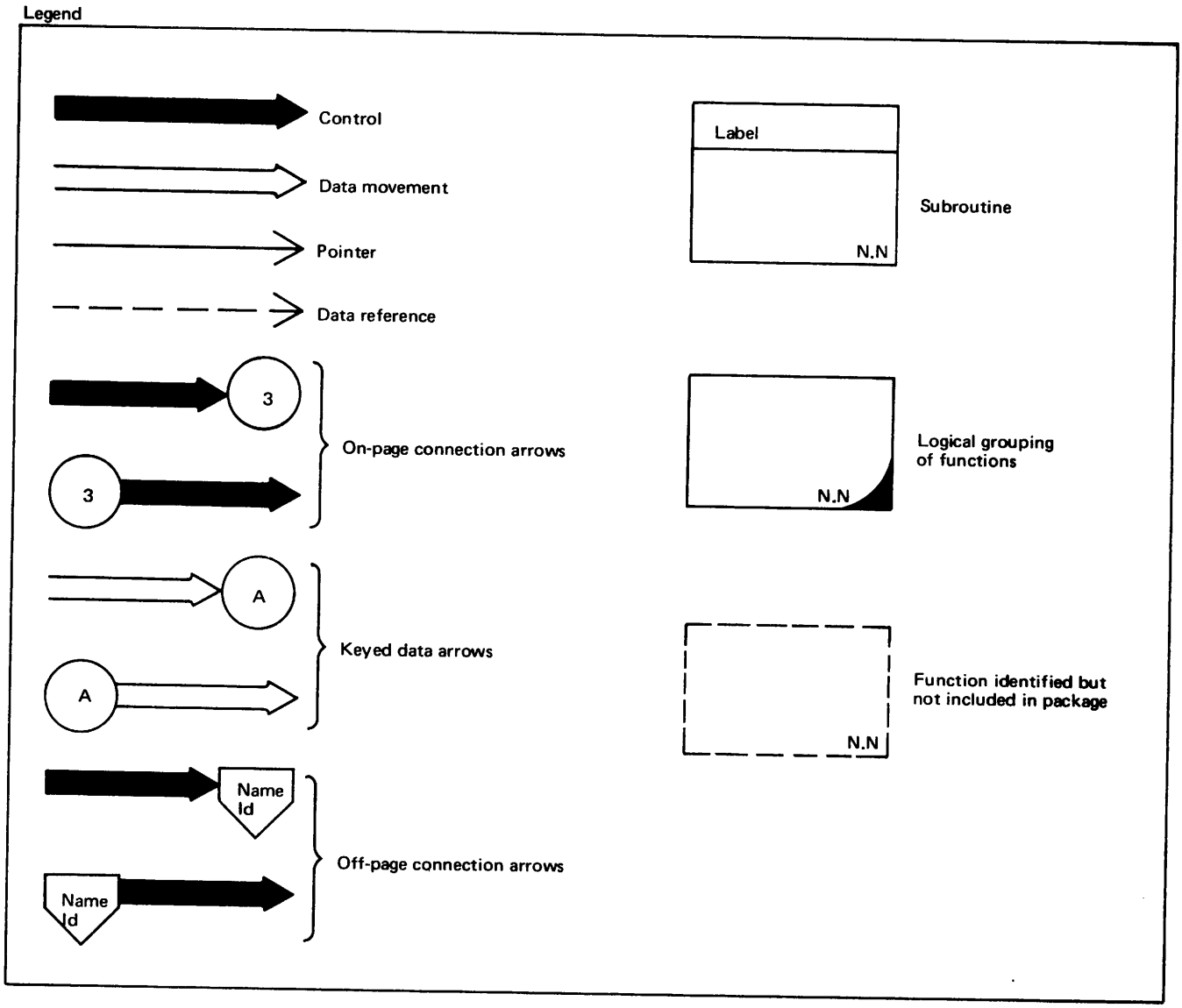
The following suggestions are made for the use of the visual table of contents in a HIPO package:

- Each set of HIPO diagrams should be accompanied by a visual table of contents diagram that shows the relationship of each diagram to the others in a set. It should be the first item in a HIPO package.
- The visual table of contents should be shown in a hierarchy structure as in the following example. The hierarchy structure should represent the functional structure of the program or system. Overviews should appear at the highest levels, and the lower-level diagrams should contain more detail.



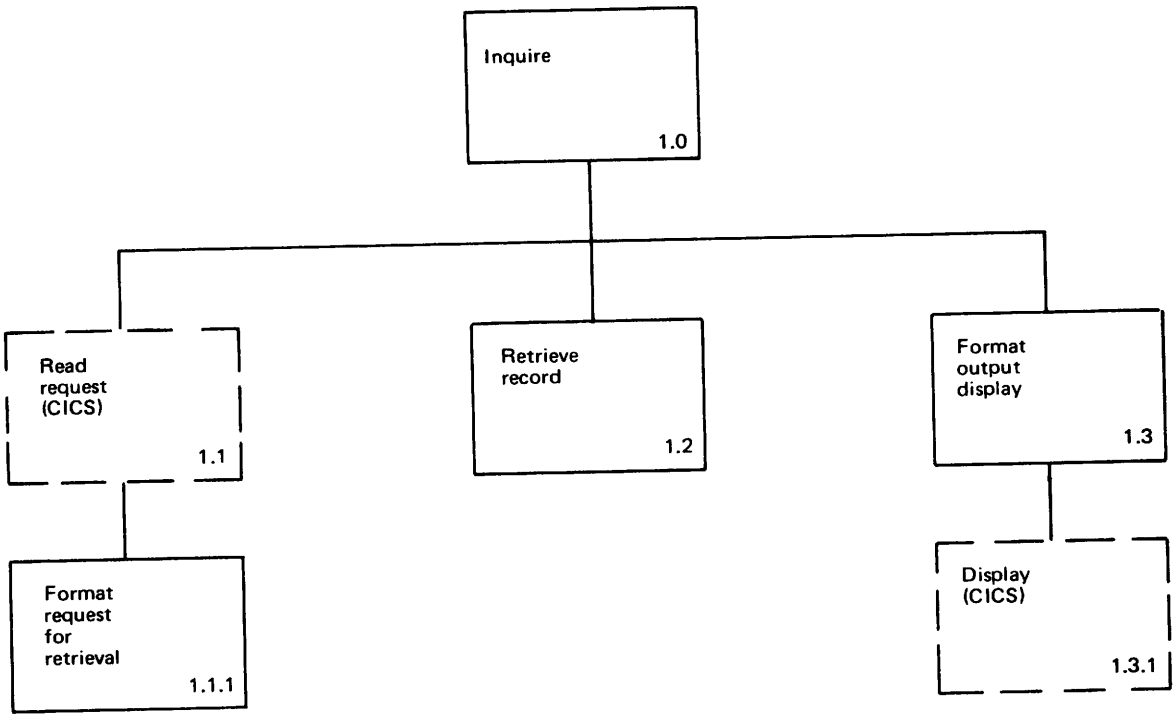
- Each box in the hierarchy should contain the identification number and name of the process described in the corresponding HIPO diagram. The page on which the diagram appears may also be within the box.
- The recommended numbering scheme for HIPO diagrams is that shown in the visual table of contents diagram above. Numbering in this manner facilitates additions to the package without disturbing the numbering scheme. Sequential numbering can be used for packages with few diagrams and few expected additions. Using the hierarchical numbering scheme, other types of documentation (flowcharts, file layouts, screen displays, etc.) can be added to the package by giving them the same number as the HIPO diagram that initially references them, with an appended alphabetic character, such as 6.4.2A. If they pertain to the entire HIPO package, they may be given the same number as the highest-level HIPO diagram in the package, with an appended alphabetic character.

- Each visual table of contents diagram should contain a legend describing to the user how the diagrams are to be read. For example:



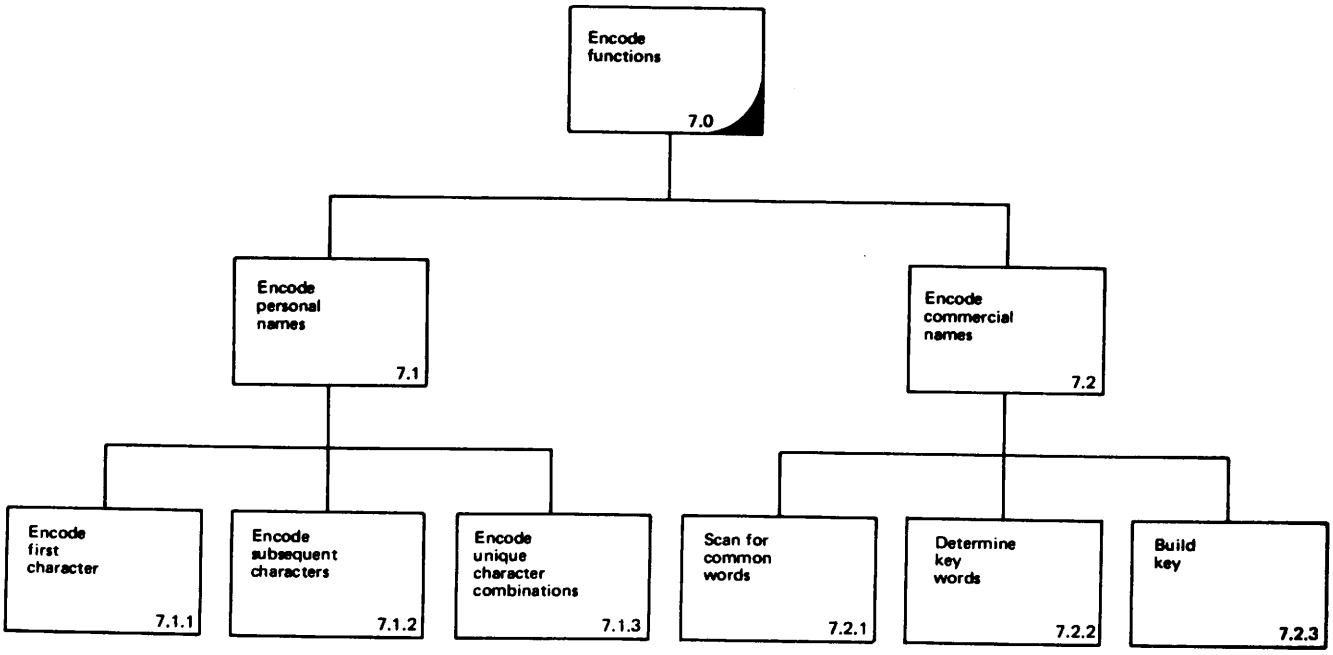
- If the HIPO package is to be read by persons totally unfamiliar with the HIPO technique and/or data processing, a comprehensive legend (or reading guide) may be added to the visual table of contents. Some items that may be included in a reading guide are:
  Input/output symbols
  Arrow usage and conventions
  Abbreviations
  Extended description conventions

- If processes performed by another system are included in a visual table of contents to provide clarity, they may be indicated by a dashed-line box. For example:



Another use of the dashed-line box is to show functions that have been identified but not yet included in the package. A solid-line box is drawn when the function is included. Whatever the use, it must be explained in the legend.

- If some functions are grouped into a logical collection as opposed to being described where they are used within the implemented program structure, the logical grouping box may be designated by a shaded corner. For example:



- The logical grouping may be used to collect diagrams of very frequently used routines, such as utility routines, which, if placed beneath each diagram that uses them, would unnecessarily enlarge the visual table of contents. These diagrams should not have any references in the visual table of contents other than in the logical grouping. However, the extended description accompanying the diagrams that use these functions should contain the HIPO identification number in the reference column. In each diagram in the logical grouping the identification numbers of the calling diagrams should be adjacent to the entry control arrow. This technique, useful for error message printing and other frequently used routines, is recommended only where it is impractical to show all subfunctions where they are used in the visual table of contents.

General Format of a HIPO Diagram

The following points are suggested for maintaining consistency in HIPO diagram format:

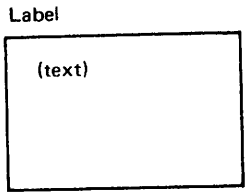
- Each section (input, process, and output) should be outlined and labeled. Each section may vary or be fixed in both width and length within a set of HIPO diagrams, depending on user preference. The HIPO Worksheet provides three fixed-size sections labeled and outlined, and extended description sections.
- Shading may be used to highlight areas if desired, but it has no functional meaning. The user should decide whether the time involved in shading is worth the resulting effect. Note that shading does not reproduce well.
- HIPO diagrams limited to one page are more readable than multiple-page diagrams. If multiple-page diagrams are used, the process steps should be numbered sequentially throughout the diagram.
- Each diagram should contain certain information for identification. The HIPO Worksheet provides a heading for the following information:
  The author of the diagram
  The name of the system or program being documented
  The date
  The page number (if the diagram is the nth page of an m-page HIPO diagram, it should be so labeled)
  The diagram identification number
  The name of the function being described in this diagram (if a program label has been assigned, it should be used)
  The description of the function (this should be the same description contained in the visual table of contents)

Use of Data Items and Data Groups

The following suggestions are made for the use of data items and data groups within HIPO diagrams:

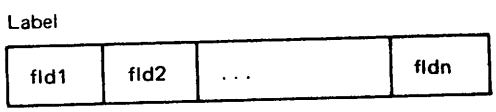
- The following are examples of data items that may be shown within the input or output section of a HIPO diagram. Each item should be labeled. Note that the size of a data item box has no functional significance; it is determined by the amount of information to be placed in that box.

Example 1

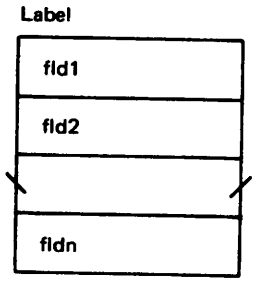


A single data item symbol. This may be a field, register, record, storage area, etc. The label identifies the item and should be the name actually used in the program if one has been assigned. If this is the case, it should be in capital letters. The text can briefly describe the data item. The extended description section may identify important attributes.

Example 2

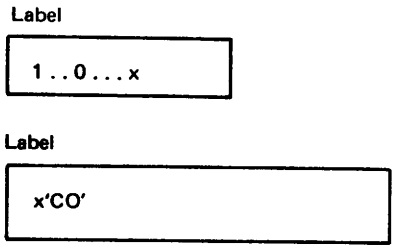


A record with a layout of fields within it, shown several to a line.



A record with a layout of fields within it, shown one to a line. This format can facilitate drawing the arrows and updating the data item.

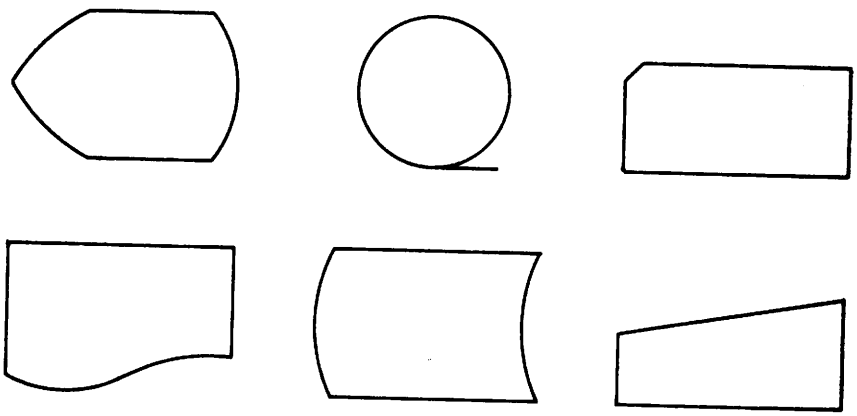
Example 3



A flag block. A data item that has a length of one or more bytes. Each of the 8 bits within each byte may be tested or modified. The significant bits should be shown as a 1 or 0. The remaining bits should be shown as periods. If a bit may be significant but may be 0 or 1, an X may be used.

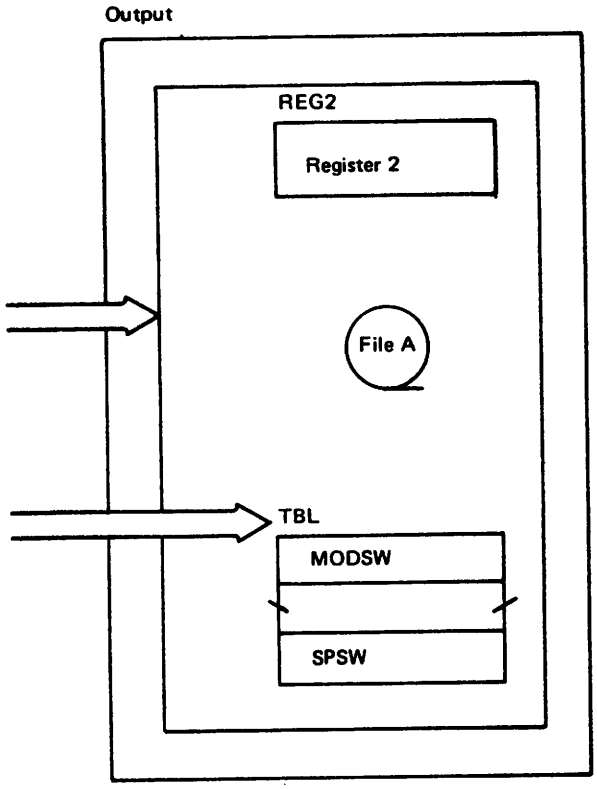
Preferably, the layout of a flag block is shown in the extended description or a separate figure to permit easier updating, should the contents change.

Example 4



External devices. The standard flowchart symbols may be used to depict external media such as cards, documents, tape, online storage, and display devices. When physical device dependency is not important or does not exist, a data item symbol may be used.

- A data group is one or more of the data items discussed above which are enclosed in a box. The purpose of grouping data items is to indicate that all of the items within the box are included in the data flow of arrows that *touch* the edge of this box. The value of placing data items within a data group is to reduce the number of arrows. For example:



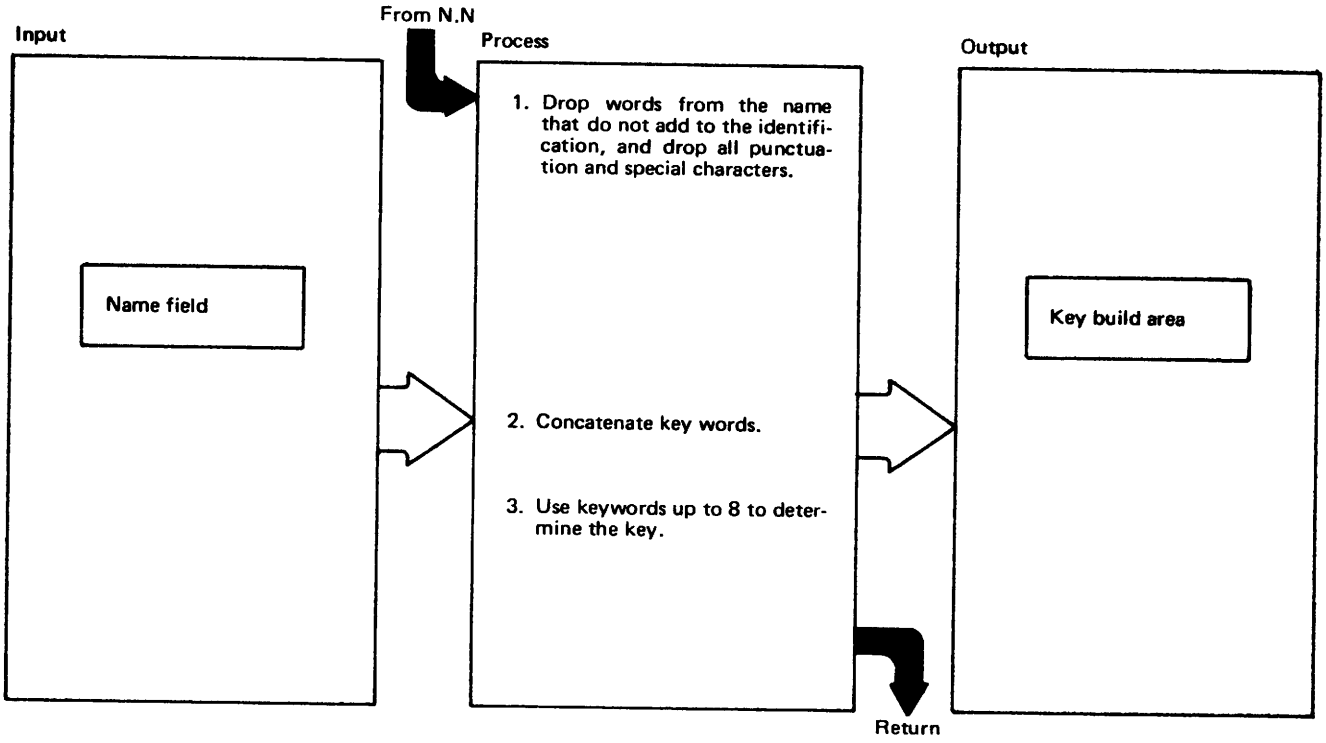
- Only the fields used should be shown in the input. A space left before or after a field in a data item implies that the item contains other fields not relevant to this input. A space does not necessarily indicate how many fields have been omitted or their lengths. Also, the ordering of fields within a data item does not necessarily imply the actual order in the record, table, etc. (unless it is a program dependency, in which case it must be so stated in the extended description). It is often useful to rearrange fields or groups of data items to facilitate a better layout of data related to process steps and avoid the crossing of data movement and data reference arrows.

Use of Arrows

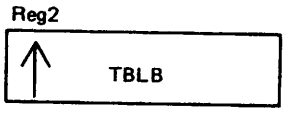
The following suggestions are made for the use of the various types of arrows in HIPO diagrams:

- The width of the arrow does not depict a functional difference. A wide arrow of a given arrow type may be used for emphasis; two sizes of arrows are shown on the HIPO template. While the narrow arrow is the one most frequently used, the wide arrow may be used when there is only one arrow in the input and output paths. Any other use of different-width arrows should be explained in the legend and extended descriptions, as appropriate. If required, arrows may join each other or may cross over each other, as described later in this section.

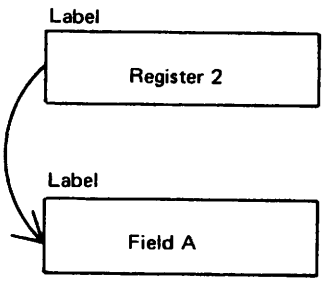
- The data movement arrow indicates which data item or group of data items is moved or manipulated, or produced or modified, by the associated process step(s).



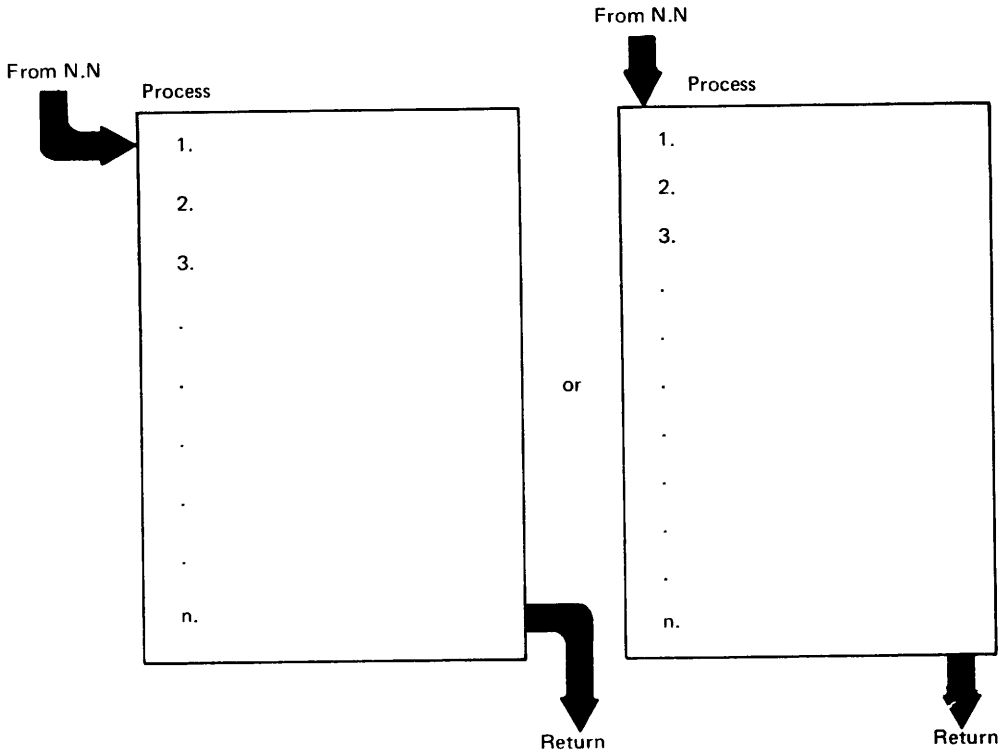
- The data pointer arrow indicates the address of another data item.



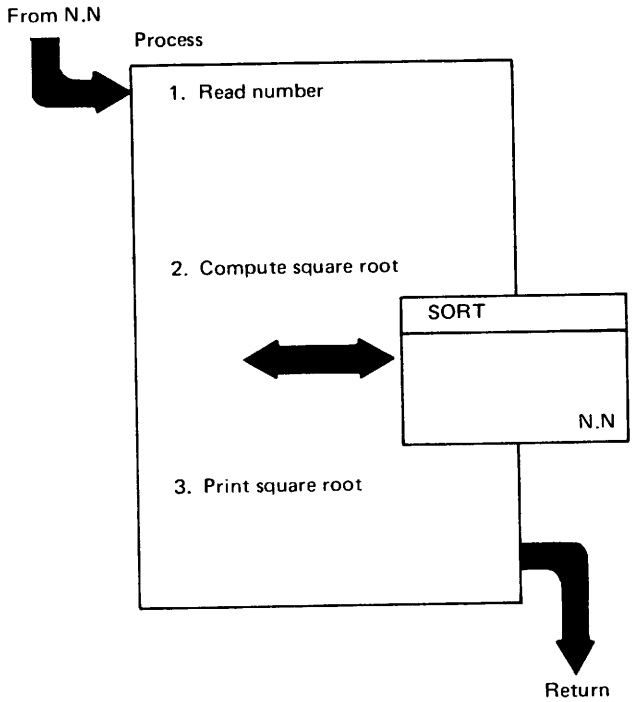
The arrow may connect the data items:



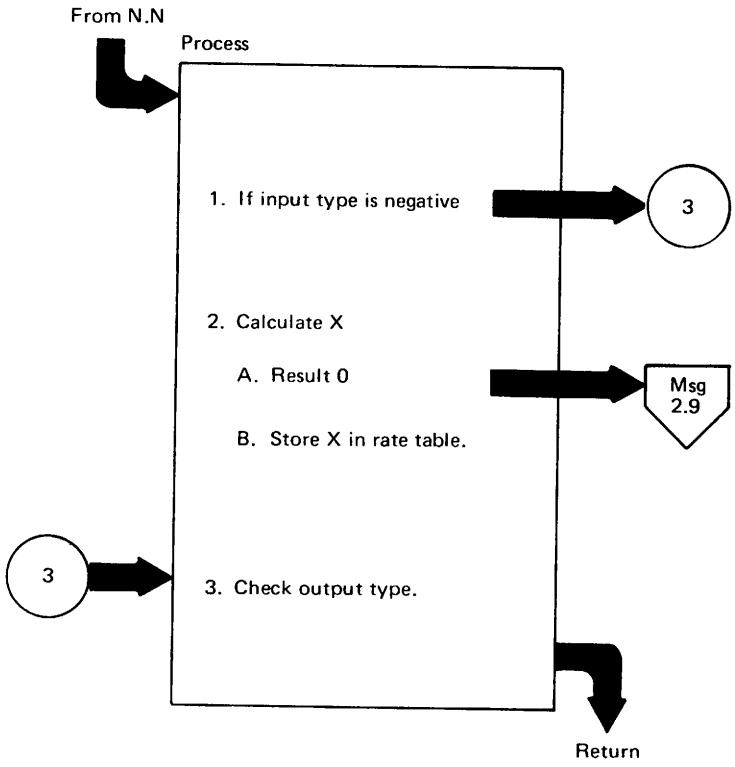
- The control arrow indicates that the processing is going to continue at the function to which the arrow points. Following are some examples:
  a. The entry point into the diagram and the exit point from the diagram may be shown in either of the following two ways with the identification number(s) of the calling and called (if it is different from the calling diagram) HIPO diagram(s) adjacent to these arrows:



  b. This example shows that control is given to another function, such as a subroutine, and returns to the next operation once the function has been performed:

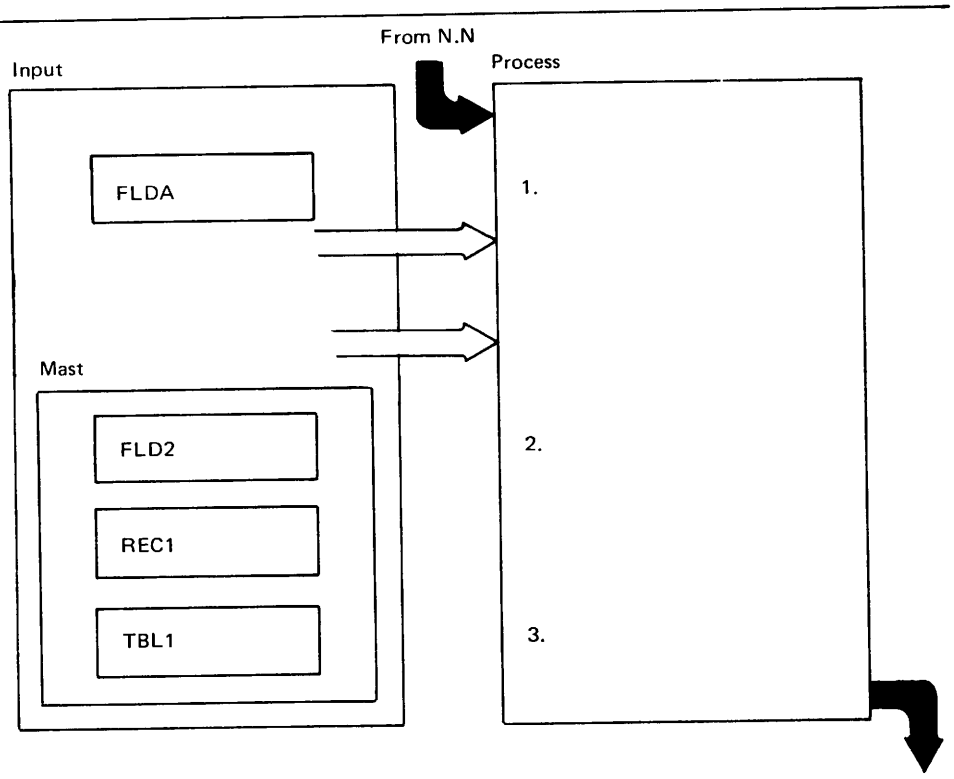


c. This control arrow indicates that an alternate path is to be taken. This use of the control arrow should be kept to a minimum since the reader typically must follow this path and return before continuing to read the diagram. This arrow will probably not be used when structured code is being developed and when HIPO diagrams are limited to one page.

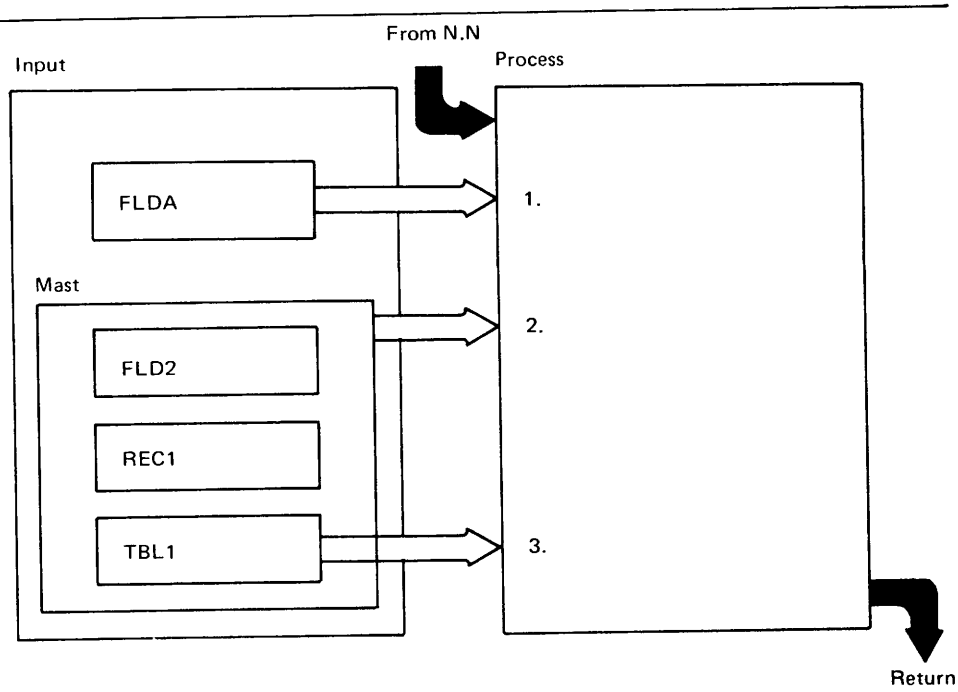


- A data arrow in the input path should begin at the right-hand *edge* of a data item or data group and terminate at the number of the process step that uses this item(s). For output it should begin at the right-hand edge of the text for the process step and terminate at the left-hand edge of a data item or data group in the output section. The beginning of all output arrows may be aligned. The exception to this is that arrows may be drawn in such a manner as to prevent crossing text with an arrow.

UNCLEAR



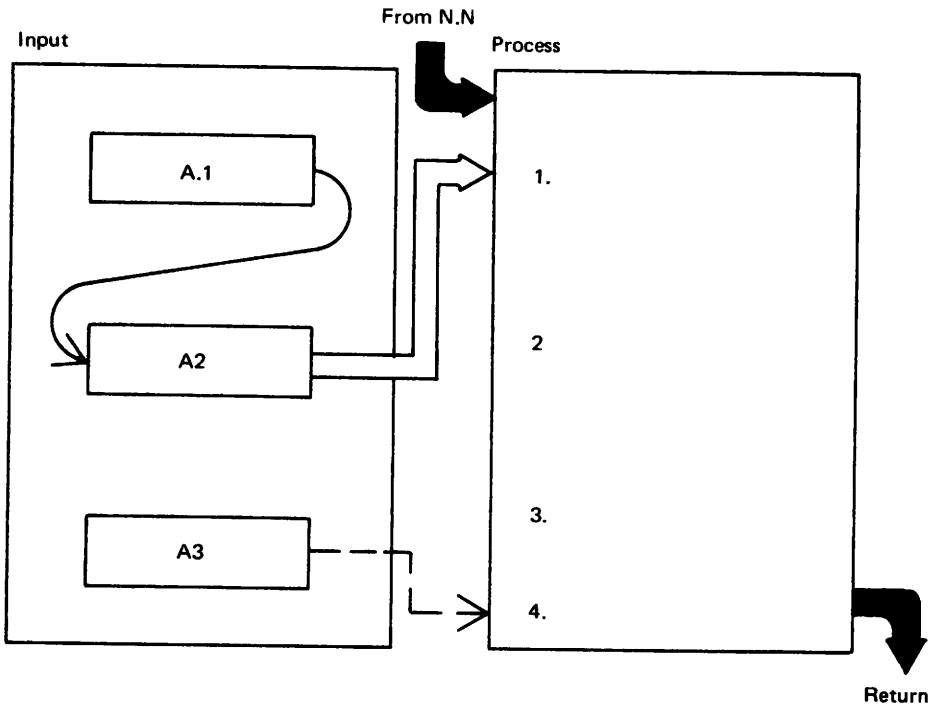
CORRECT



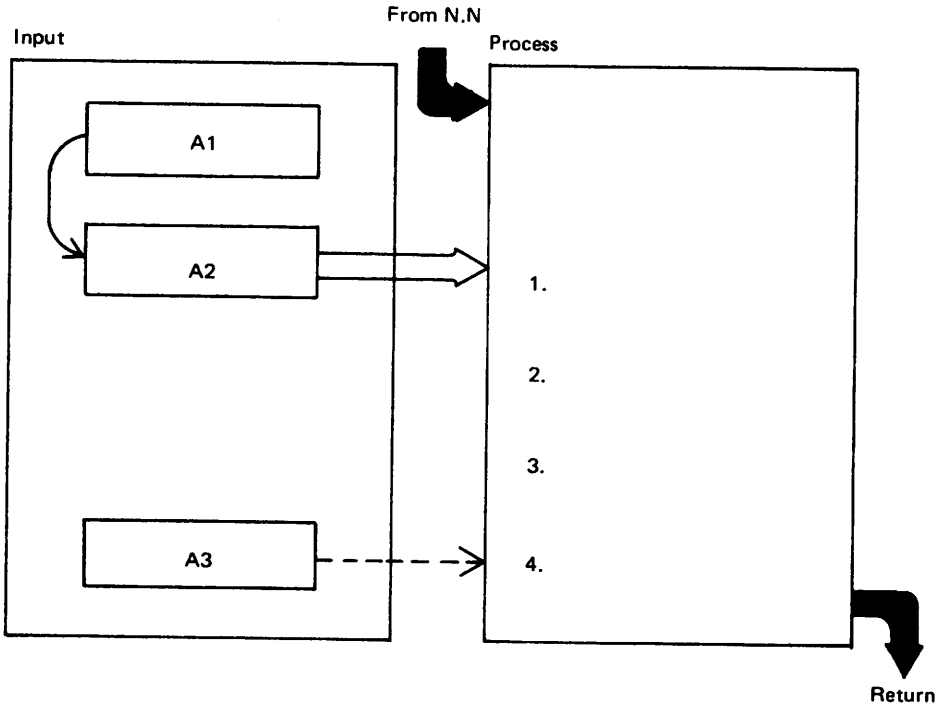
It is unclear in the first example whether the input was FLDA or all data, or whether the input was for step 1 or 2.

- Arrows should be as straight as possible so the eye can easily follow them. For example:

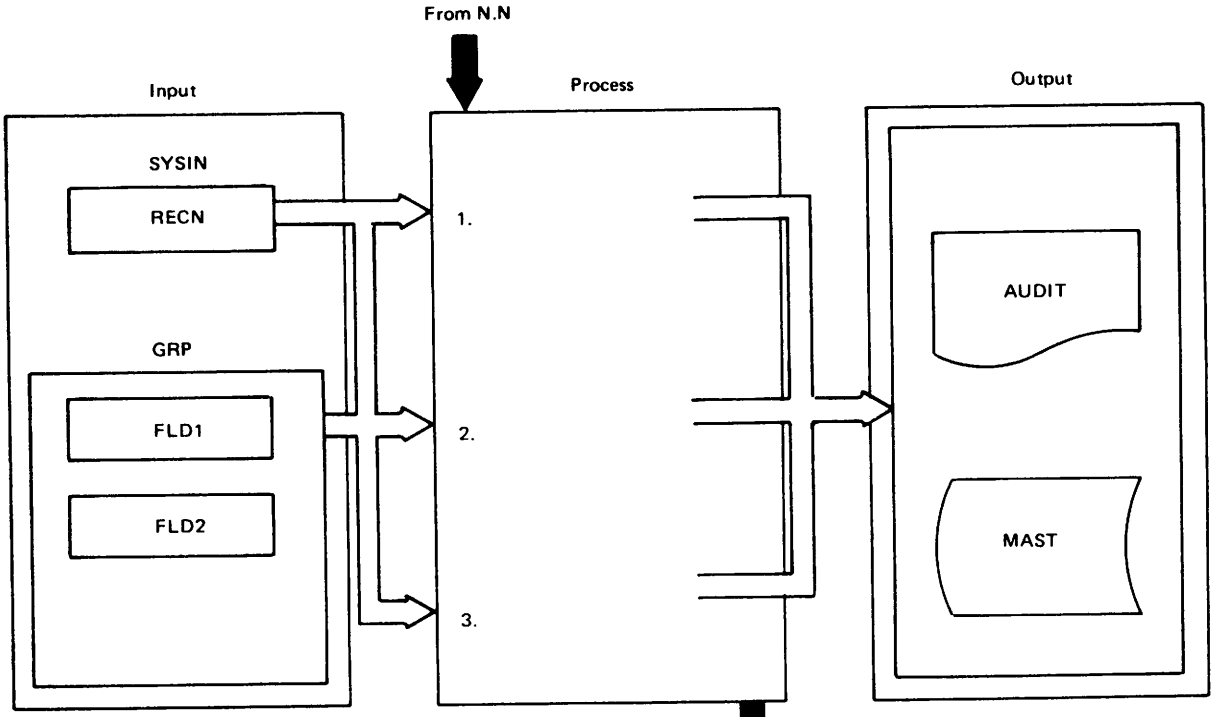
POOR



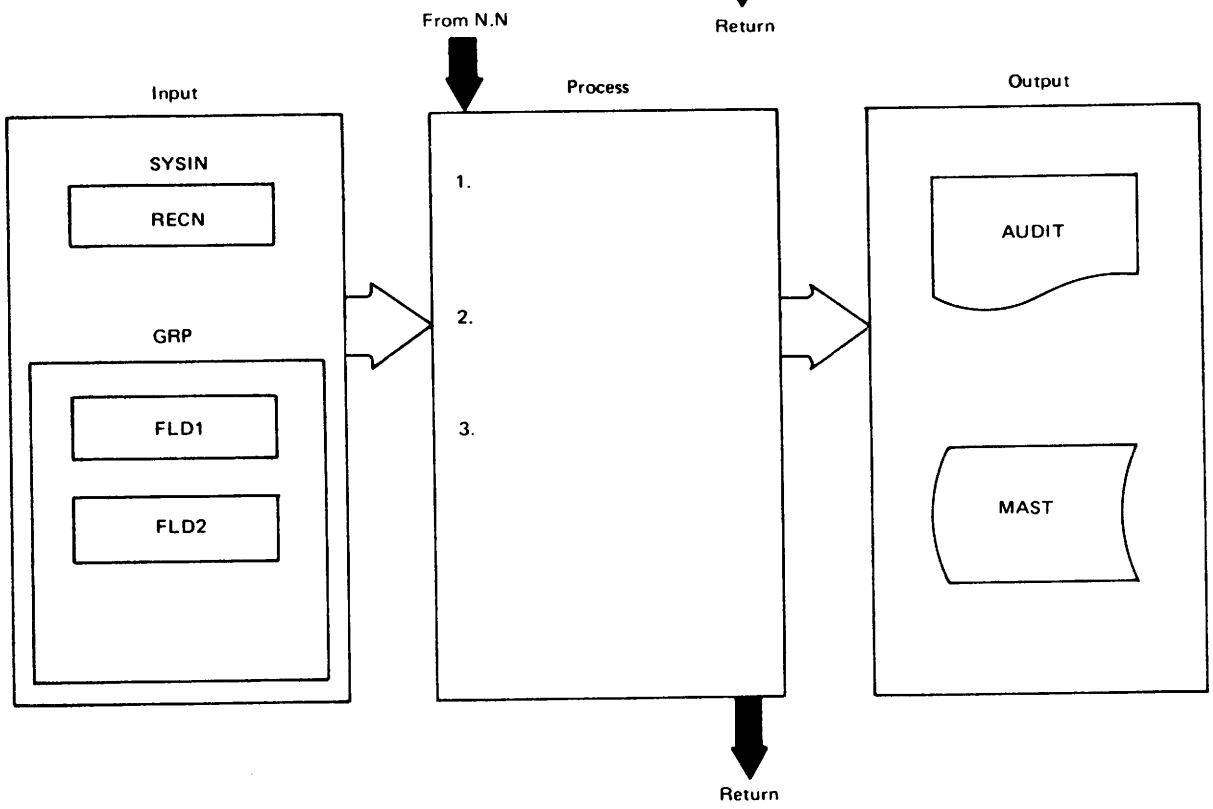
IMPROVED



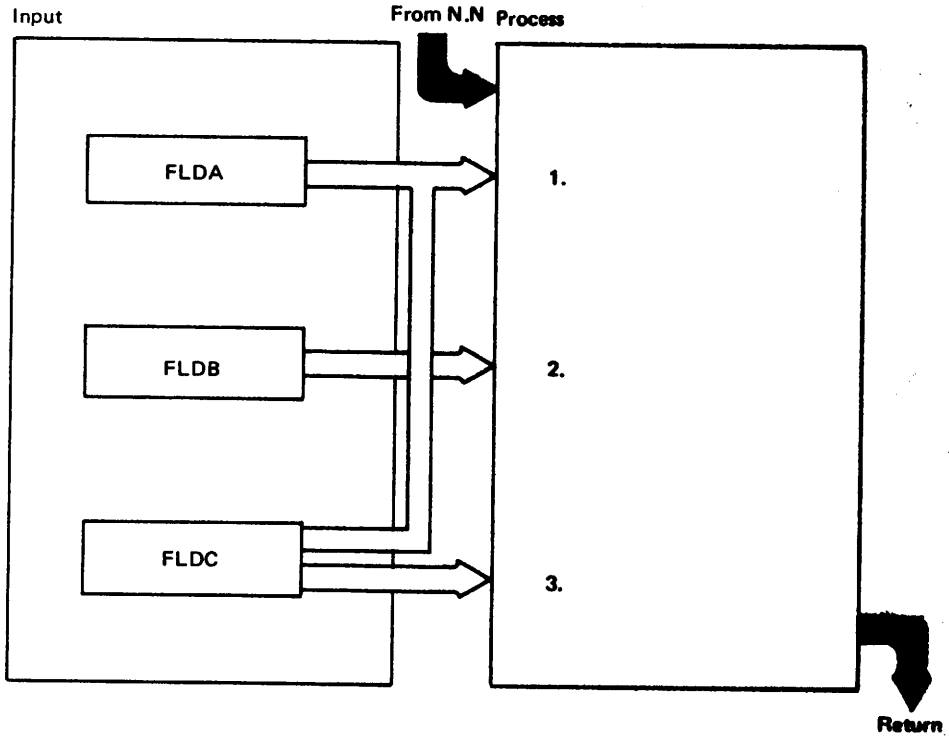
- If there is only one data movement arrow in a given diagram, it may go from the edge of the entire input section to the edge of the entire process section and from the edge of the entire process section to the edge of the entire output section. Frequently a wide arrow is used for emphasis here. This is particularly useful for overview diagrams.



or

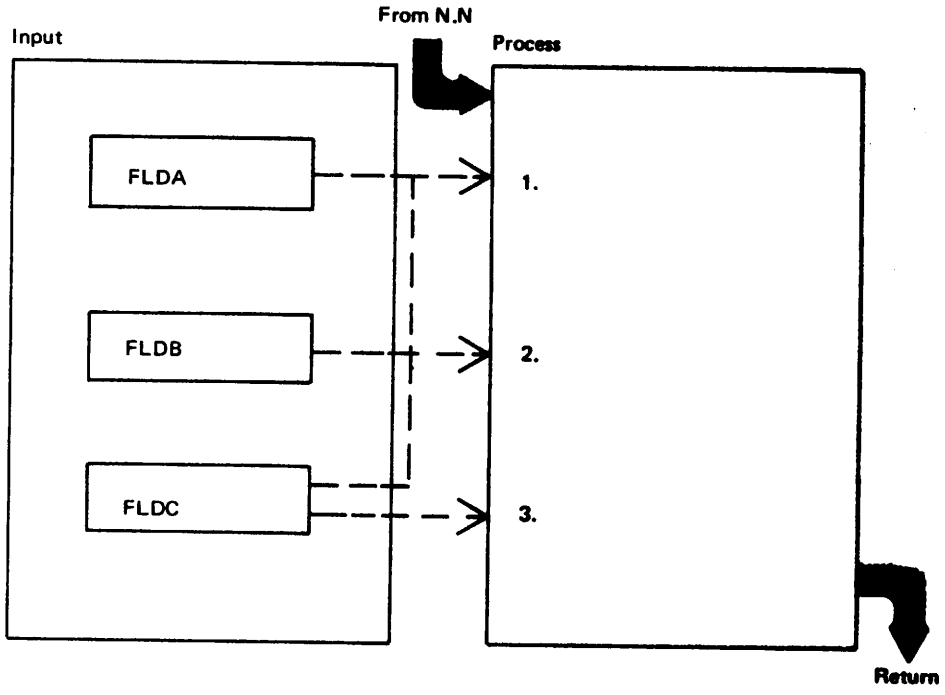


- Arrows may cross over each other or may join to show that they point to the same process step number. Crossing arrows should be kept to a minimum. Many times they can be avoided by rearranging the data items, enclosing data items within a box (data group), leaving space between process steps, or using keyed data arrows. The following example shows crossing arrows:

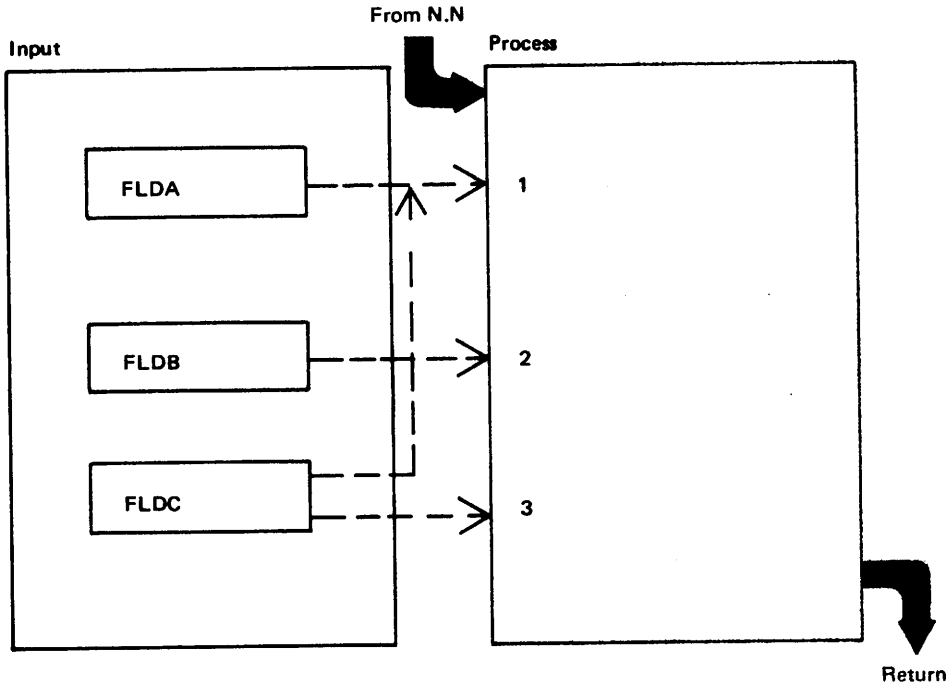


From the above, a user can tell that the arrow to step 1 is from FLDA and FLDC, and that the arrow to step 2 crosses the other arrow and is from data FLDB.

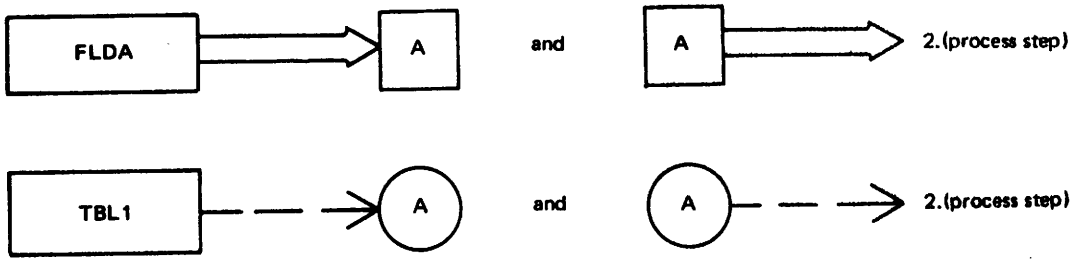
- It is more difficult to illustrate crossing arrows with data reference arrows.



In the above example, FLDC may go to both steps 1 and 2. In order to clarify that the arrow from FLDB goes to step 2 and the arrows from FLDA and FLDC go to step 1, the following technique is suggested. When an arrow merges with another arrow, the merging arrow should terminate with an arrowhead; if it crosses another arrow, there should not be an arrowhead at the intersection.



- Data movement or data reference arrows should not point from the process steps to the input area.
- Keyed data arrows are associated by means of the same letter and are of the form:

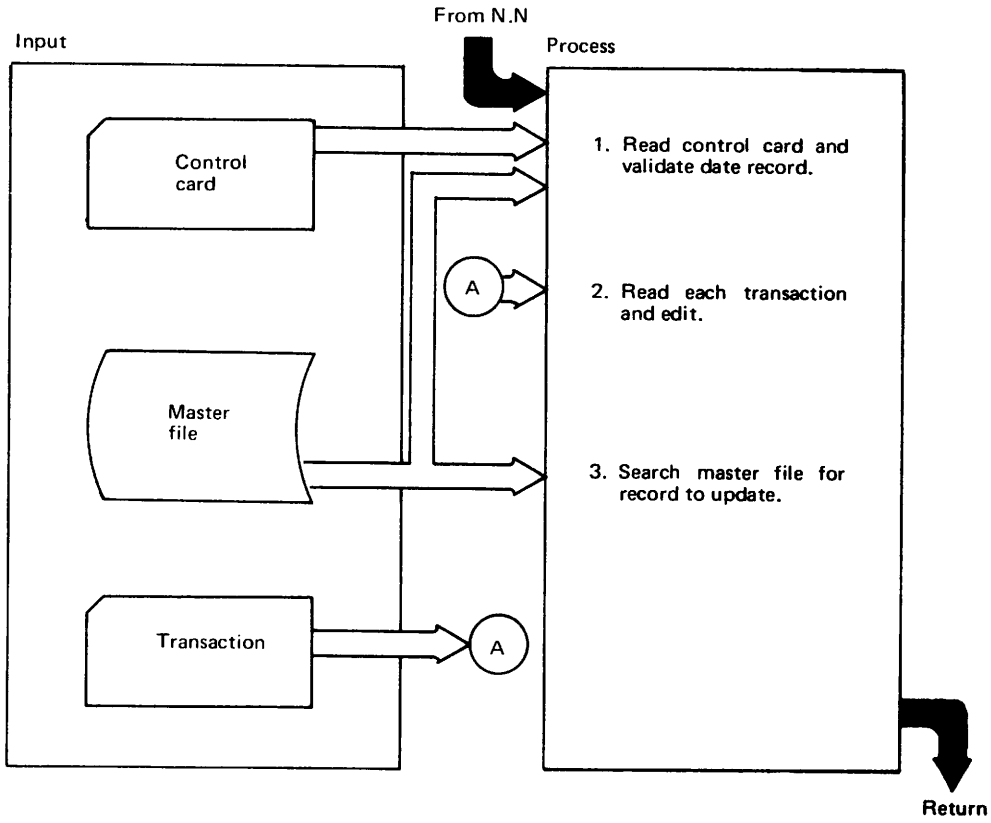


The letters may be circled or boxed. Keyed data arrows are used when multiple arrows in the input and output paths would be confusing to the user. Their use should be kept to a minimum.

- When data is used repeatedly, the following may be done:
  1. If a data item is input to several steps, but not output from them, the following alternatives are given:
     a. Use a single arrow from the data to the steps (which must for this approach be consecutive and grouped by boxing).
     b. Use a multiple arrow (one tail, several heads) from the data to each step.

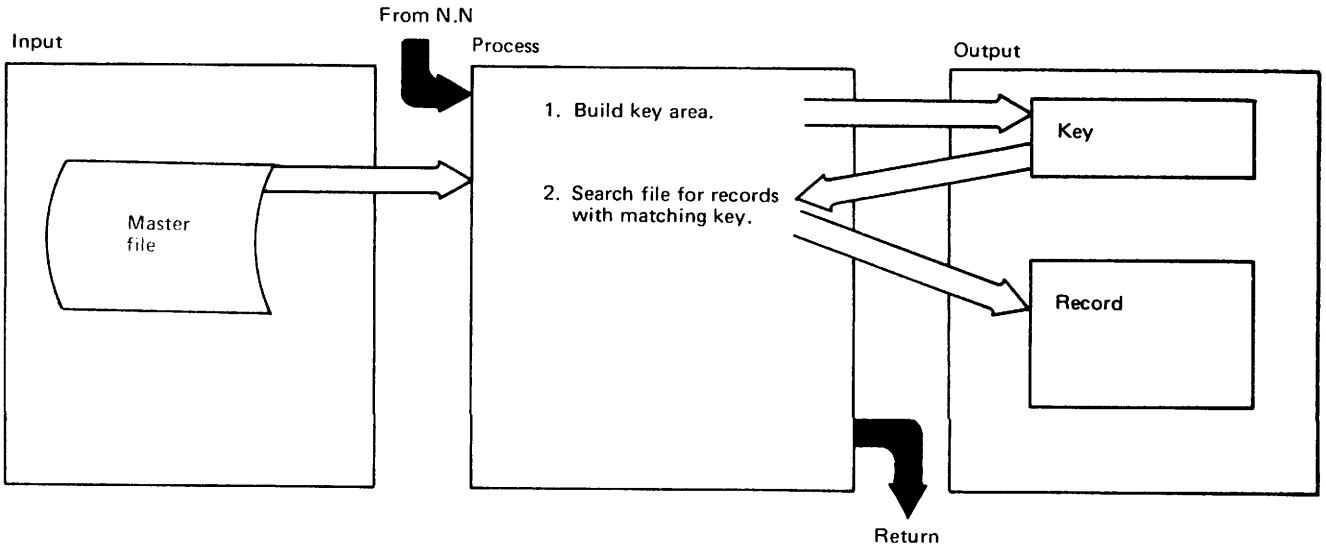
c. Use a keyed data arrow.
d. Repeat the data item as necessary, with different arrows.

These approaches may be combined.

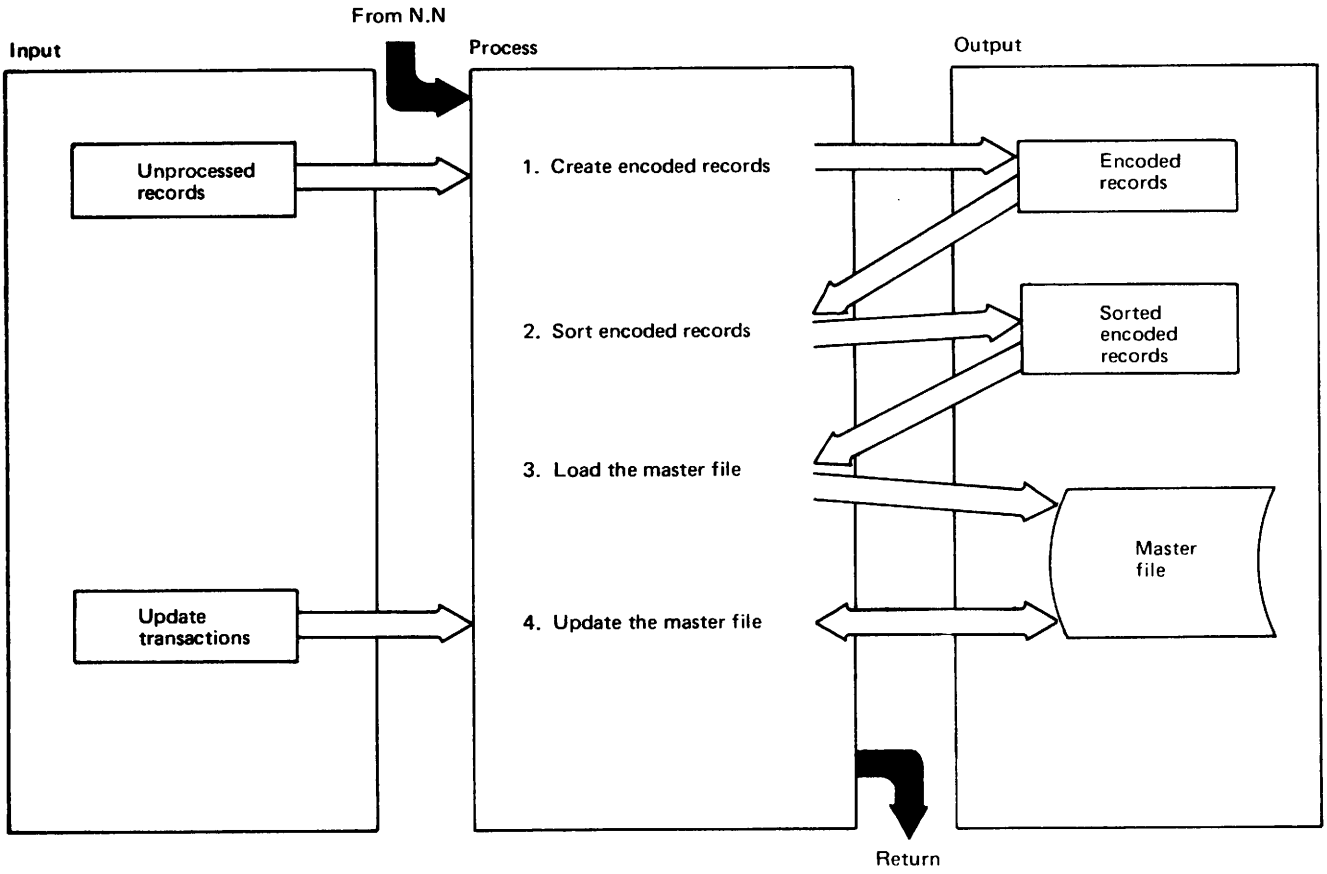


2. If a data item is output from one step and then becomes input to a later step(s):
   a. Use a single or multiple arrow back from the output data to the later step(s), using a keyed data arrow if necessary.
   b. Repeat the data on the input side. In this case the reader must be told that the data item is not original input, by adding text (for example, "created in step 3" or "from step 3") to the data on the input side.

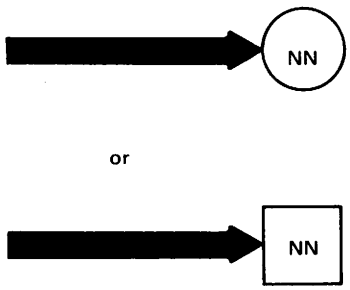
   The first option should be used except where it would cause confusion of arrows in the output path.



3. If a data item is input to a step that modifies it, and then is input (in its modified form) to a later step(s), the same techniques as in (2) above may be used, but the word "updated" rather than "created" should be used. Note that the use of the words "created" and "updated" with output data items may be helpful even when they are not used as input later.



- When data shown in the output area is used as both input and output in the same process step, a double-headed arrow may be used.
- On-page connection arrows are used to indicate that control is transferred from one process step to another within the same diagram. The format is:



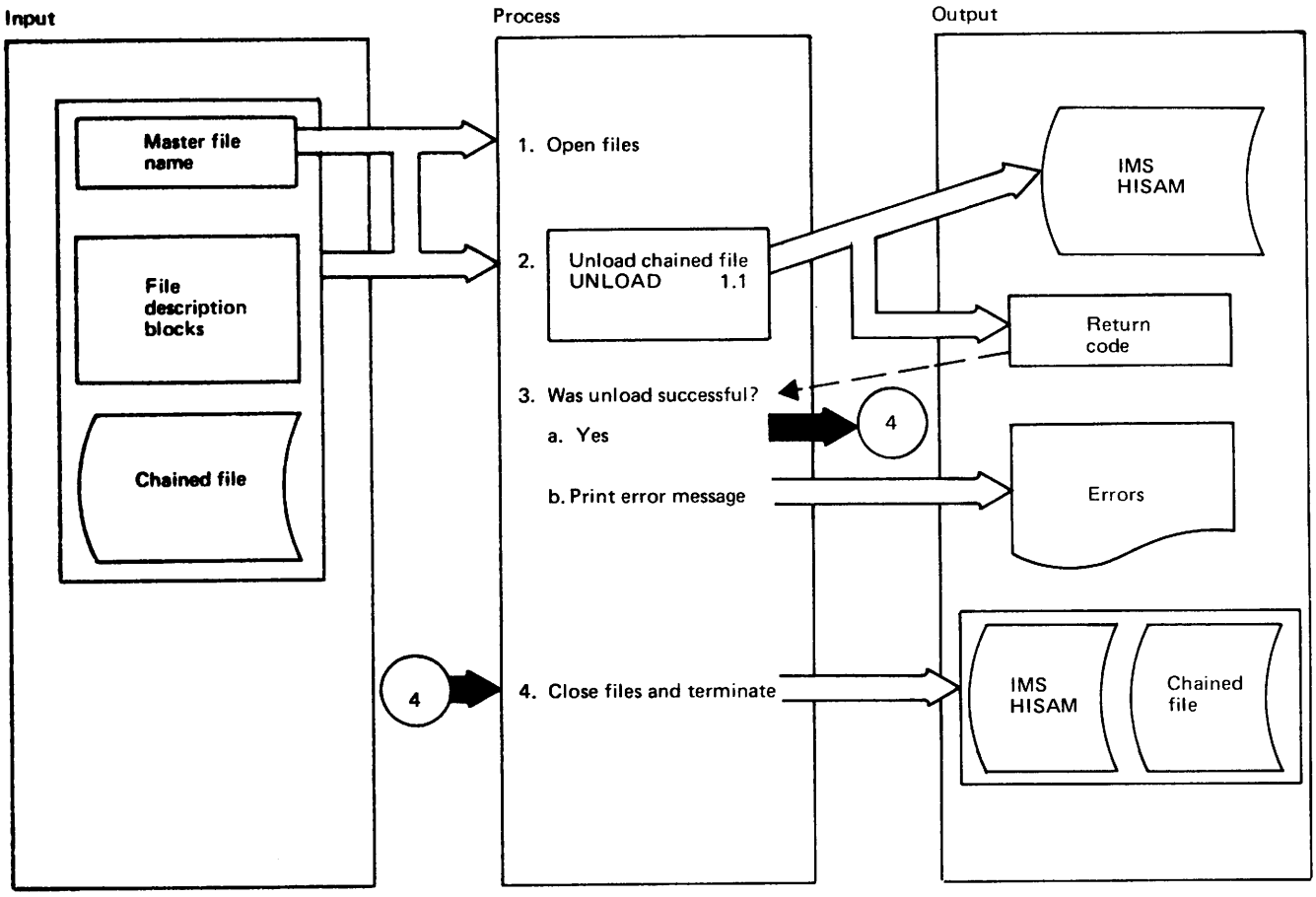
where NN is the step number to which control is given. At step NN there should be an arrow of the following format:

The number may be circled or boxed. Diagram 6.3 in the initial design package (see Appendix A) contains an example of an on-page connection arrow.

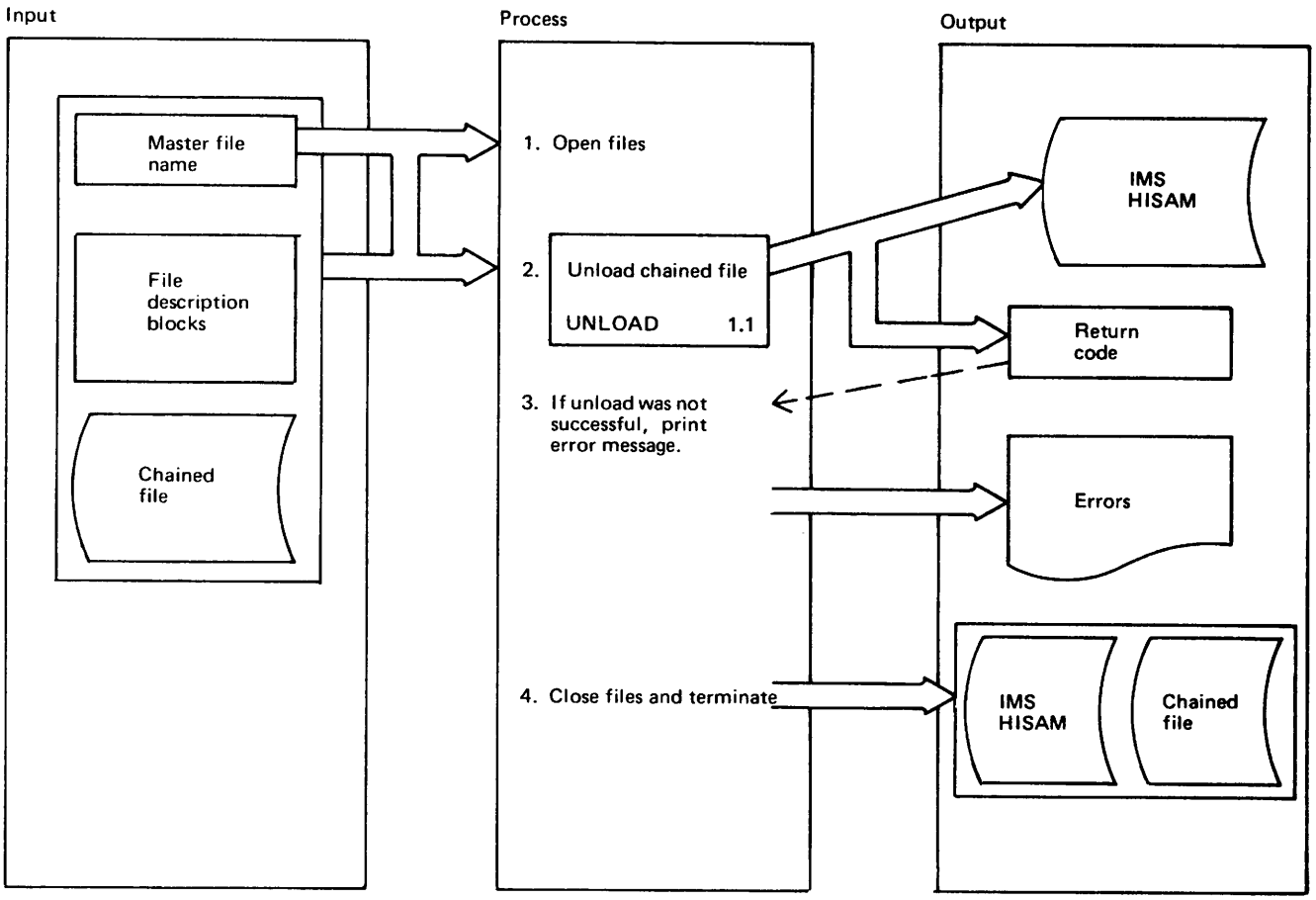
- Off-page connection arrows are used when control is transferred from a process step on one diagram to a process step on another diagram. The format is:

where the name is that of the function to which control is transferred and the ID is the HIPO diagram number. If, within that HIPO diagram, the step number to which control is given is other than 1, the step number should also be indicated. At the entry point associated with this off-page connection arrow, the following should appear:

- On-page and off-page connection arrows may often be avoided by rewording the process step. The example below requires an on-page connection arrow:



Rephrasing process step 3 in the above example eliminates the need:



Another method of minimizing control arrows is to describe secondary actions in the extended description section rather than in the process section.

- Entry and exit control arrows are usually meaningless in system overview diagrams that show processes performed by independent programs. The case study overview diagram (1.0) is an example of this (see Appendix A). If top-down development and structured programming are used, there will be one entry at the top of a diagram and one exit at the bottom. The identification numbers of the calling routines should be adjacent to the entry arrow.

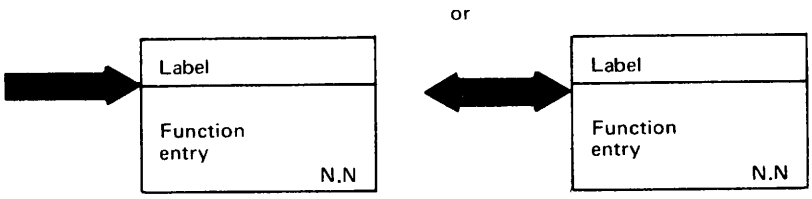
Use of the Process Section

The following suggestions are made for the use of the process section in HIPO diagrams:

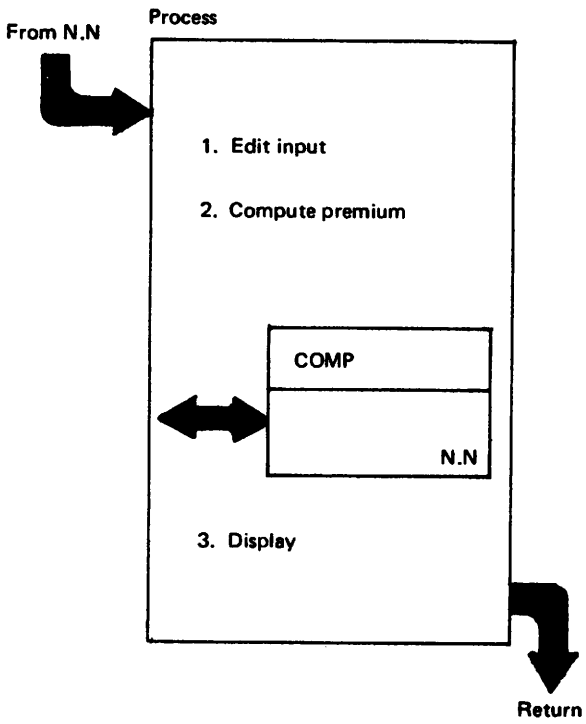
- This section should contain a single column of process steps. The steps should be sequentially numbered beginning with 1 as the number of the first step on the first page of a HIPO diagram. The numbers may be circled or boxed. Substeps should be lettered sequentially.
- There is no maximum to the number of steps per diagram; however, when the number is above the range of 7 to 10, the user has more difficulty comprehending the material quickly. Putting too many steps or too much detail in the process section is often the cause of a problem. The problem may be corrected by putting some of the information in the extended description section or by subdividing the functions into lower-level HIPO diagrams.
- To indicate that a process step is described in further detail in another diagram, two alternatives are offered:
  1. Place the identification number of the lower-level diagram in the reference column for this step in the extended description.
  2. Draw a box around the process step and place the identification number of the lower-level diagram in the lower right-hand corner of the box.
- The content of the process steps varies by development stage, complexity of the process, and intended audience. In general it is best to begin the process statement with the imperative form of a verb, such as:

accept	delete	get	perform	select
add	dequeue	handle	place	set
allocate	detach	identify	position	specify
alter	determine	increment	post	start
analyze	display	initialize	process	stop
assign	do until	insert	provide	store
begin	do while	issue	purge	supply
build	edit	locate	put	suspend
calculate	encode	link	queue	switch
check	enqueue	load	read	terminate
clear	enter	look up	record	test
close	establish	maintain	reinstate	transfer
complete	examine	make	release	translate
construct	execute	merge	resolve	update
control	exit	modify	restore	use
convert	extract	monitor	return	validate
copy	find	move	scan	verify
create	fix	obtain	schedule	write
decrement	format	open	search	

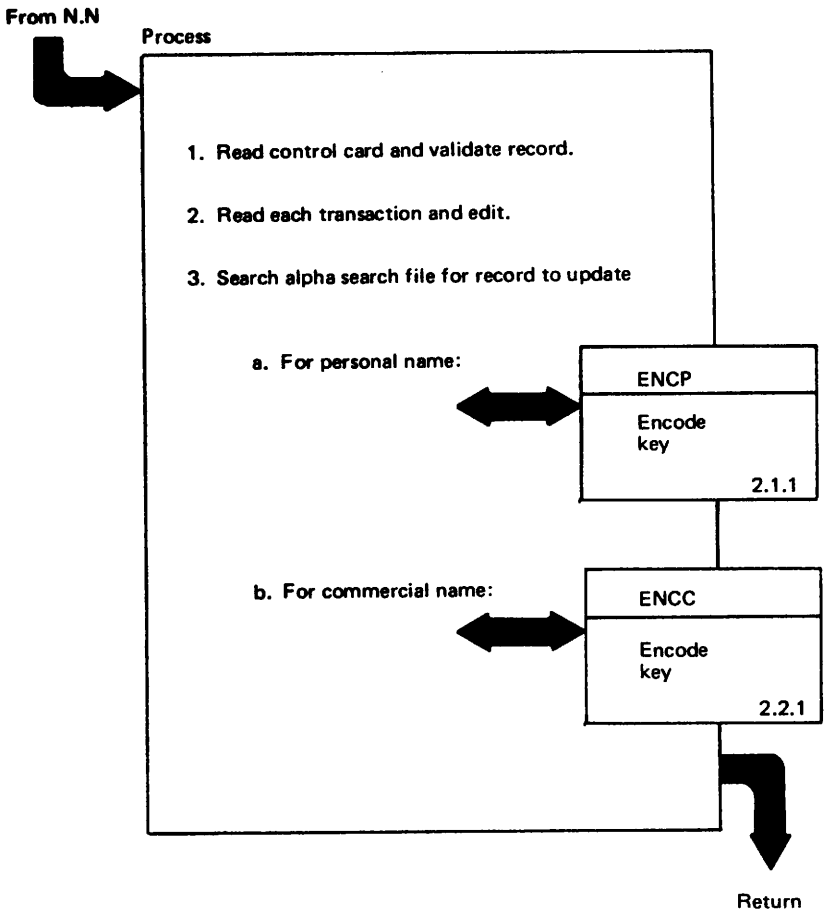
- The content of the process section should show function rather than code statements, for example, what the loop, branch, or condition test is intended to accomplish.
- No decision symbols should be contained within the HIPO diagram.
- Process steps may contain a subroutine function. The subroutine block is shown as follows:



where the label is the name of the subroutine, and NN is the identification number of another HIPO diagram describing this function. The control arrow indicates whether control is given to the subroutine with no return or with a return to the next step in the process. The subroutine may be internal to the process described or be an external routine. The internal routine should be shown as follows:

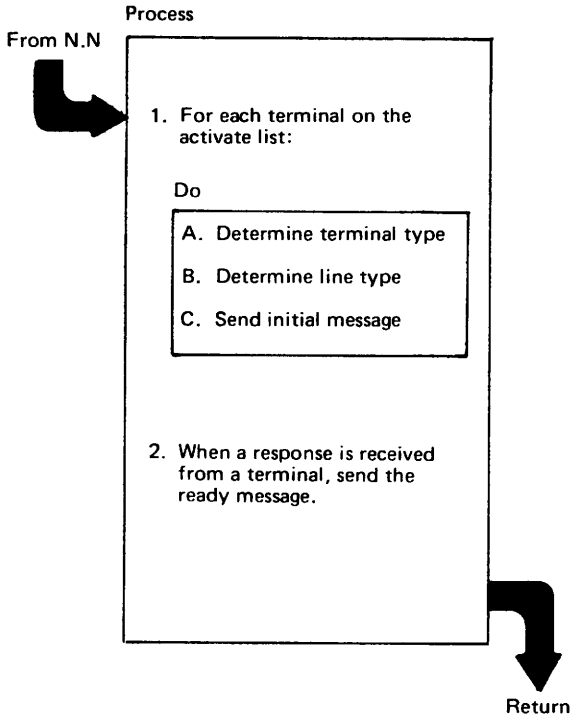


The external routine should be shown as follows:



Note: *As an alternative, subroutines within a set of HIPO diagrams may be indicated within the extended description section. (The function described may then become an internal routine, external routine, or in-line code.)*

- Iterative functions may be shown by enclosing these functions within a box. For example:



- There is a practical limit to the levels of nesting that can be seen easily on a HIPO diagram. If more than two levels are needed, the innermost nested items can be shown in a lower-level HIPO diagram or referenced in the extended description. Note that the intent of the diagram is not to show program statements, but the detail needed for coding purposes.
- Care should be taken within process step text to prevent ambiguity. Consider the following example:

  If the transaction code is change or the transaction code is delete and if the expiration date has passed, delete the master record.

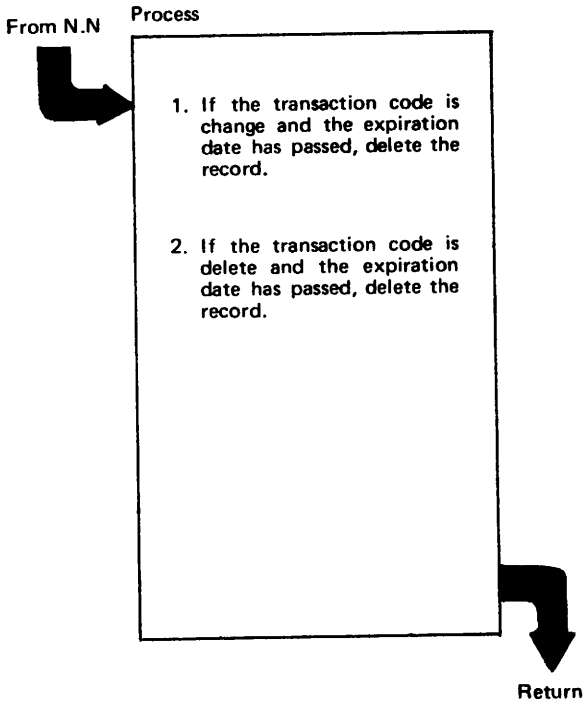
  To eliminate different interpretations, only one of the following statements should be used:

  1. If the transaction code is change or if (the transaction code is delete and if the expiration date has passed), delete the master record.

  or

  2. If (the transaction code is change or delete) and the expiration date has passed, delete the master record.

A second method of clarifying this example is to write a separate process step for each set of conditions. For example:



Use of the Extended Description

The following suggestions are made for the use of the extended description or notes section of a HIPO diagram:

- The extended description section should appear with the associated HIPO diagram; however, it may not be required with some diagrams. It contains notes, program labels, and references to other HIPO diagrams or figures. Each column of information should be appropriately labeled. Some examples of column headings are notes, routine, module label, segment, message, and reference. The HIPO Worksheet provides an extended description section with a notes area, two unlabeled columns that can be used for program labels, and a reference column. The first column, the notes area, should contain additional information required for the process step of the same number.

Extended Description

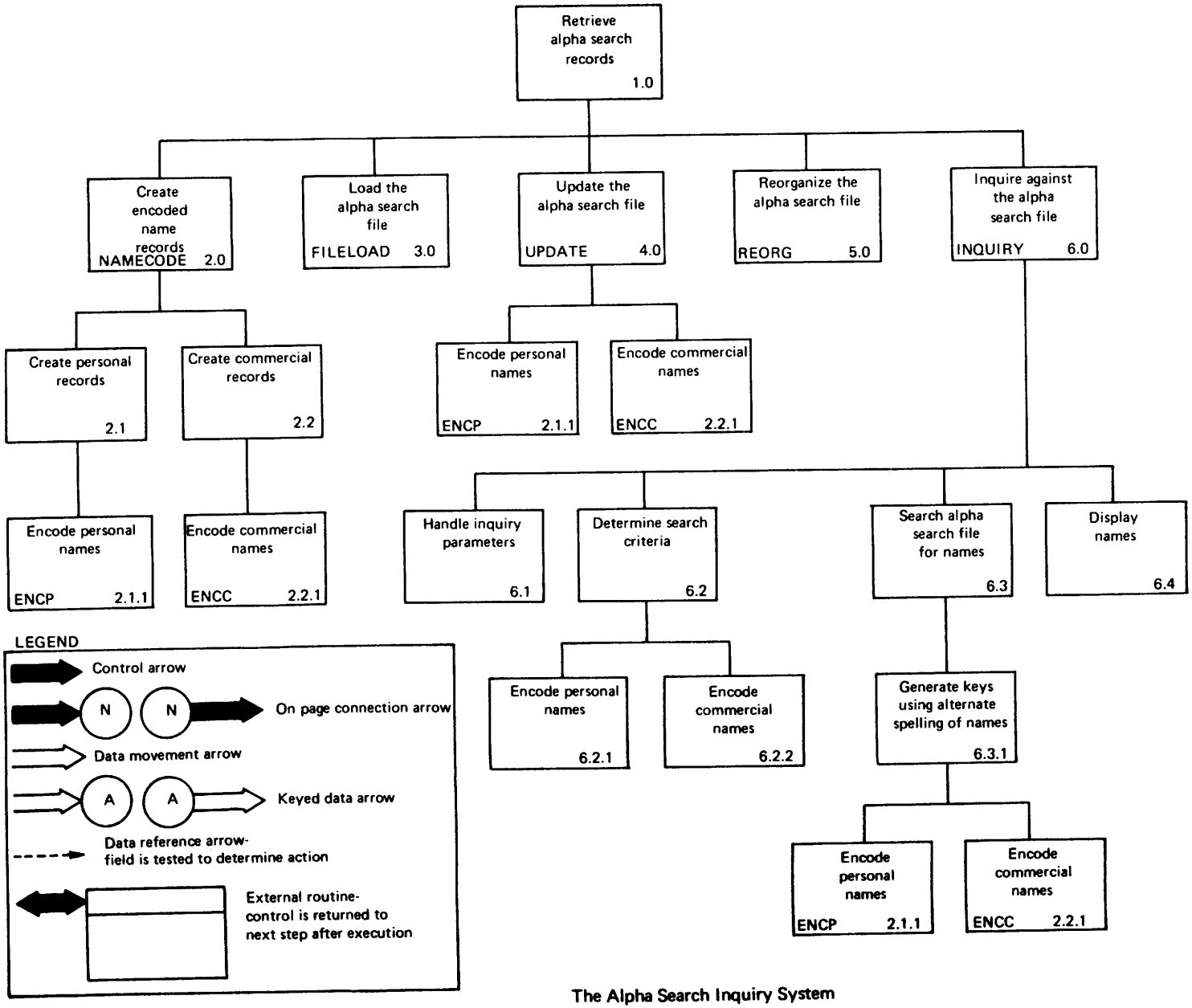
Notes	Module	Segment	Ref.
1. If invalid, job records are bypassed and error message is printed.	ICDNA	DOFRM	2.2.1
2. If type of work is incorrect, job records are bypassed and error message is printed.	ICDNA	TYPWR	2.2.2
3. Use pay rate table (RATETAB) to find correct rate.	ICDNA	PRTLU	2.2.3
4. Check for overtime, shift pay, or holiday pay and add to rate.	ICDNA	CKSPC	2.2.4

- The notes area may also contain information on input and output items. The remaining columns should contain information to assist the user in finding the code for this step or additional HIPO diagrams. It should not be assumed that information in a column carries forward until a new entry is encountered. During the implementation phase these columns should be updated as the information is defined.
- The first column may also contain an introductory paragraph pertinent to the entire process, along with additional paragraphs labeled "Entry", "Exit", "Error Processing", etc., for added information.

Appendix A: Alpha Search Inquiry System Initial Design HIPO Package

Included in this appendix are some diagrams that have appeared previously in the text.

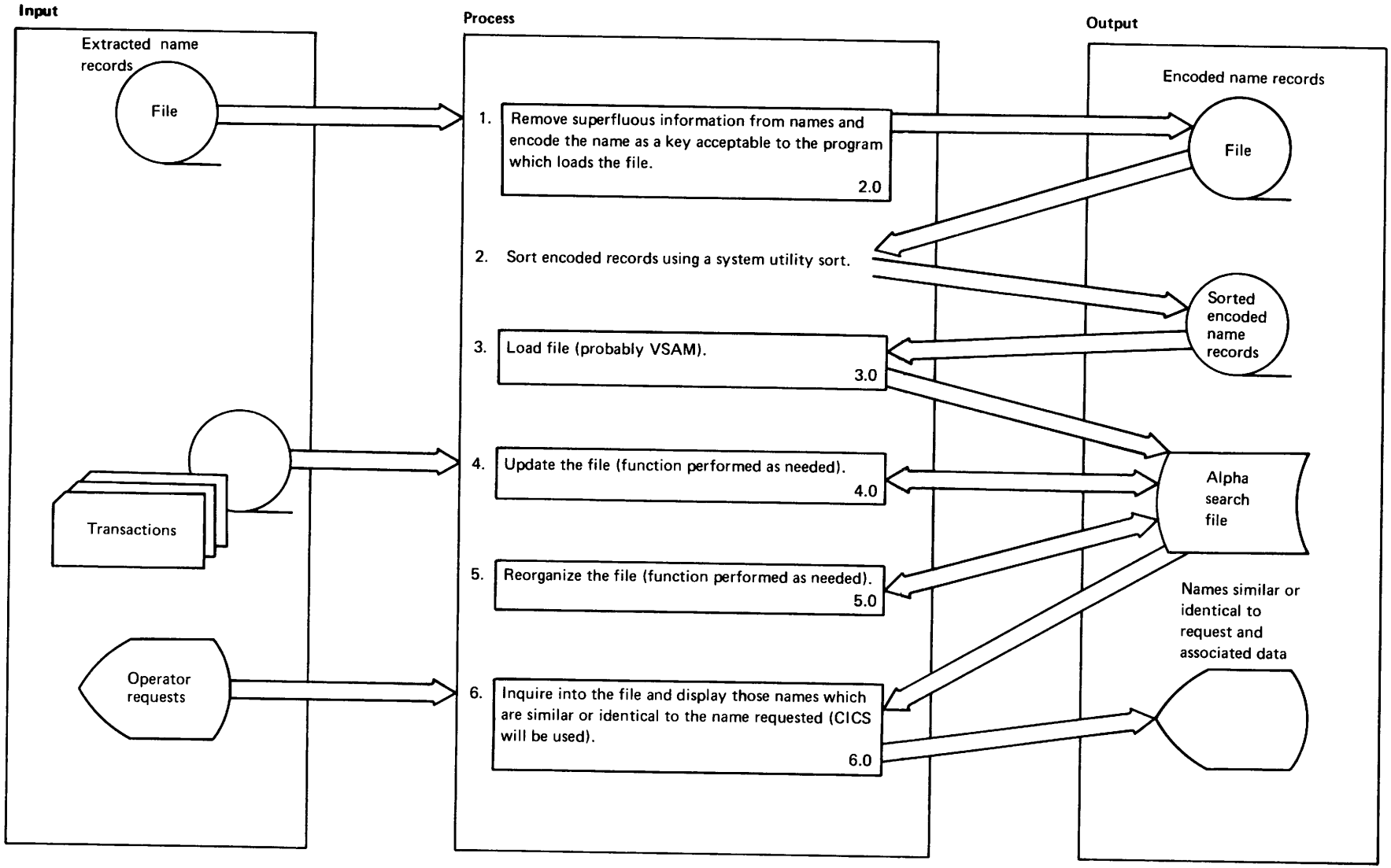
This page intentionally left blank.



The Alpha Search Inquiry System
Initial Design Level
Visual Table of Contents

Author: T.Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 1.0 Name: Description: Retrieve Alpha Search Records System Overview



Extended Description

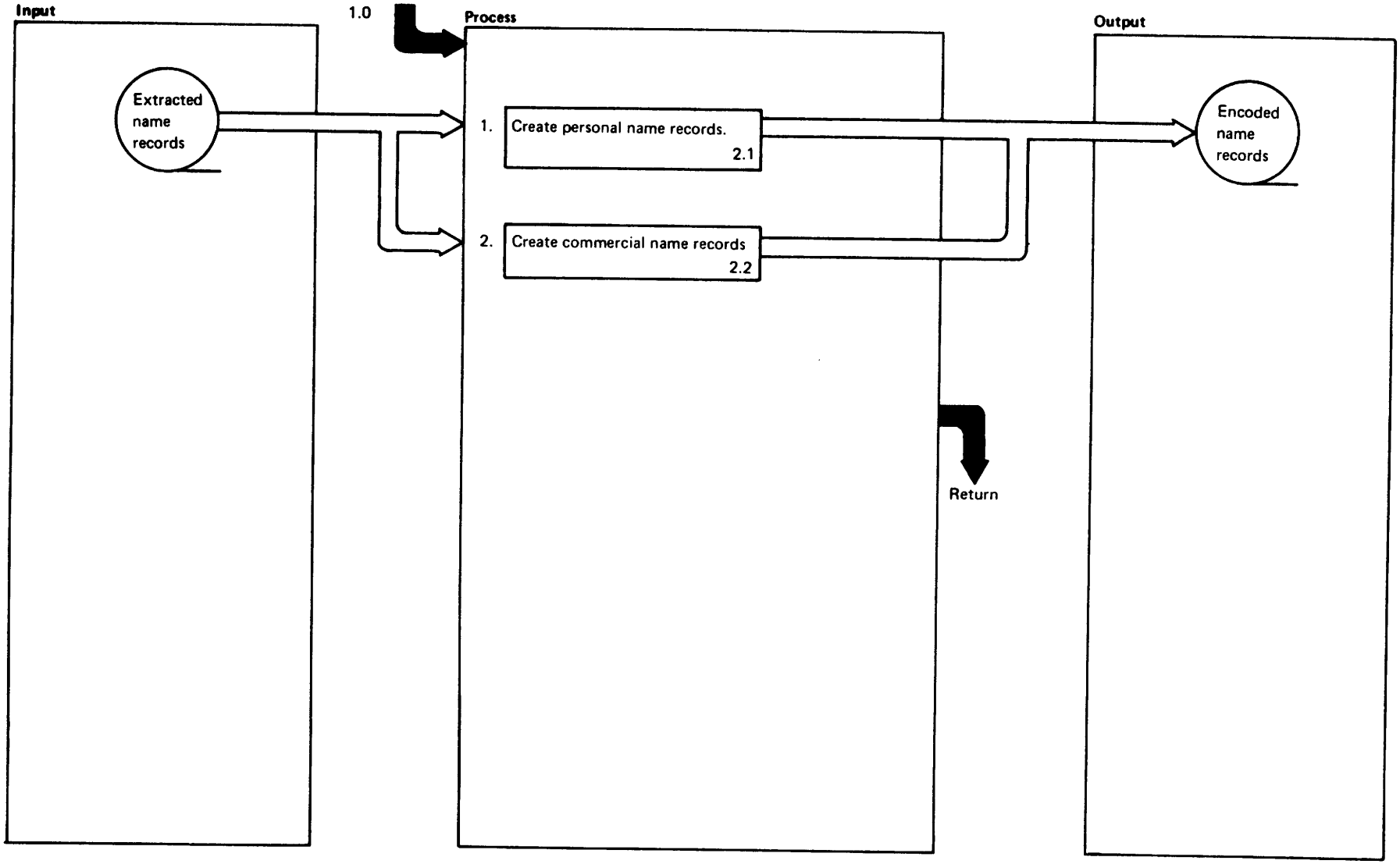
Notes	Routine	Label	Ref.
The extracted name records are created from			
existing customer files by a separate program			
which is not discussed in this package.			

Extended Description

Notes	Routine	Label	Ref.

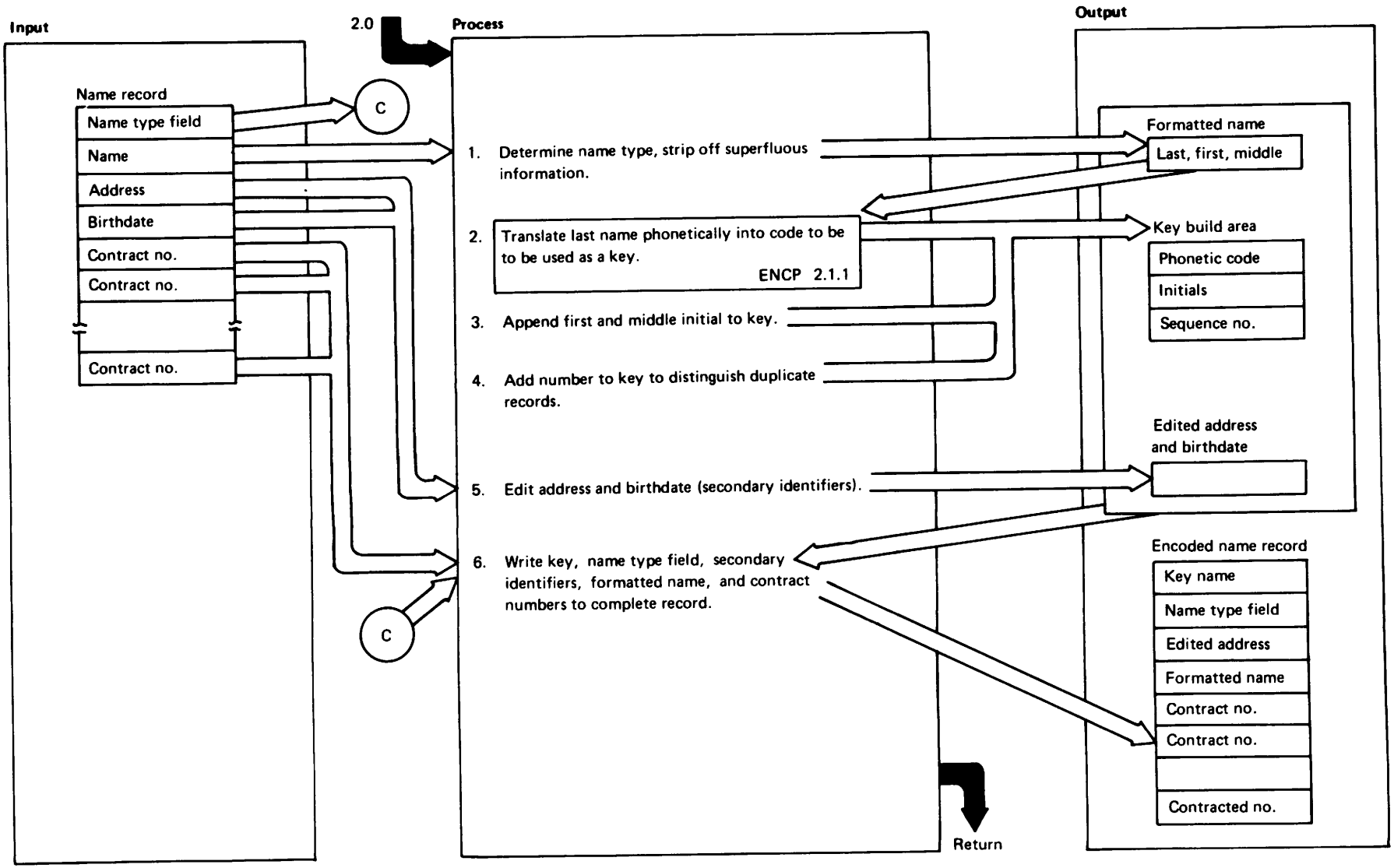
Author: J. Rennaker System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 2.0 Name: NAMECODE Description: Create Encoded Name Records



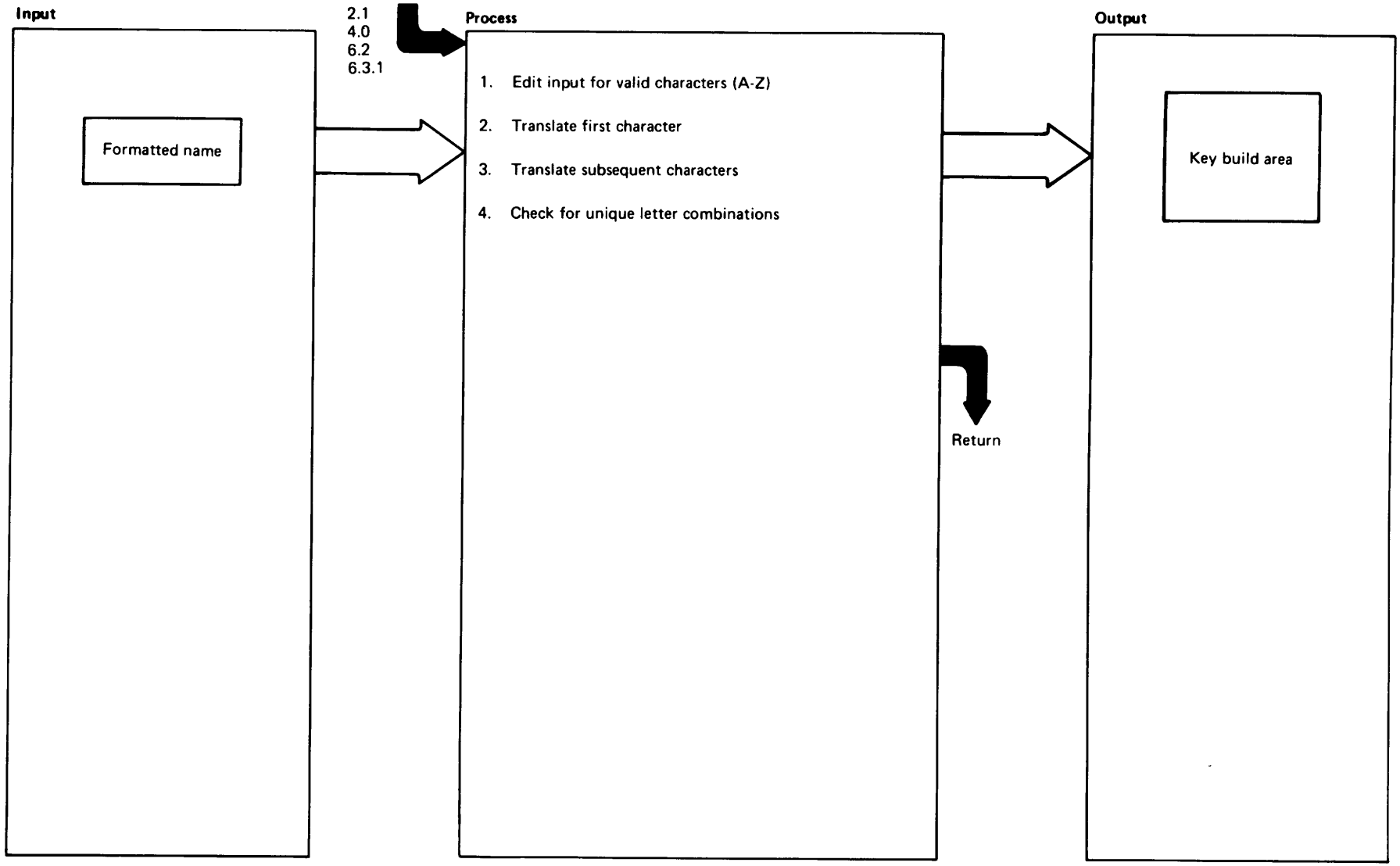
Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 2.1 Name: Description: Create Personal Records



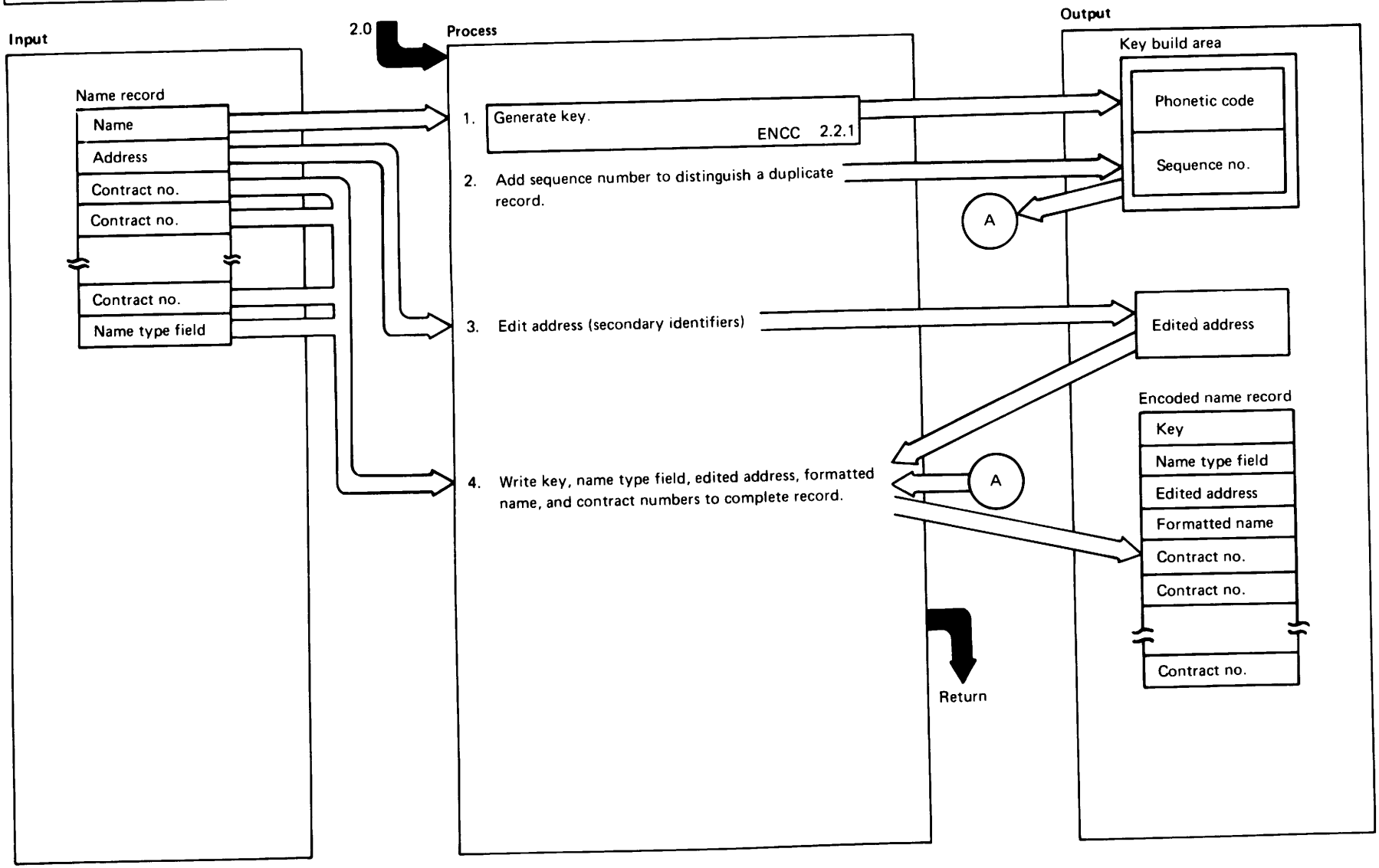
Author: T. Baloun **System/Program:** Alpha Search Inquiry System **Date:** 10/25/74 **Page:** 1 of 1

Diagram ID: 2.1.1 **Name:** ENCP **Description:** Encode Personal Names



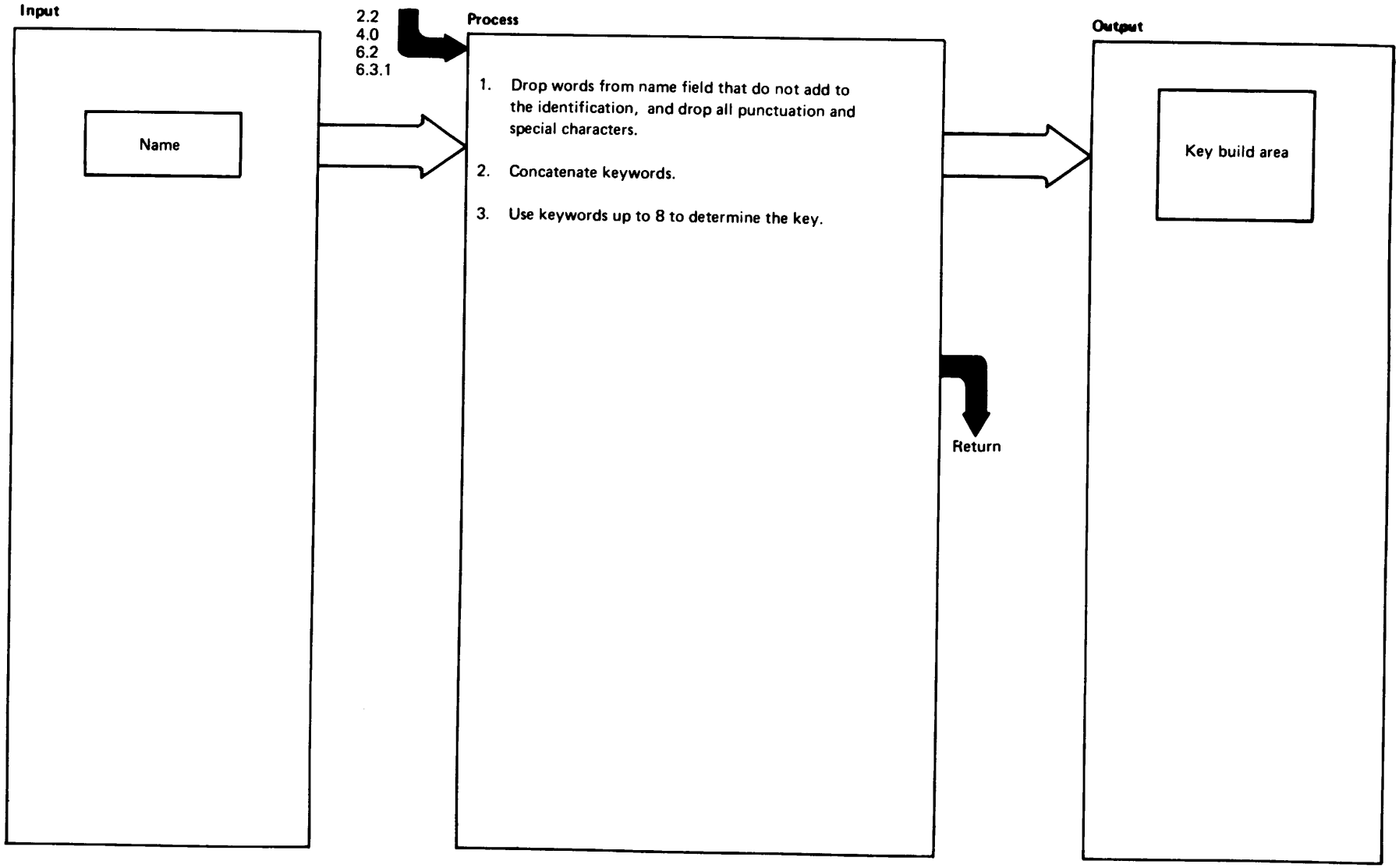
Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 2.2 Name: Description: Create Commercial Records



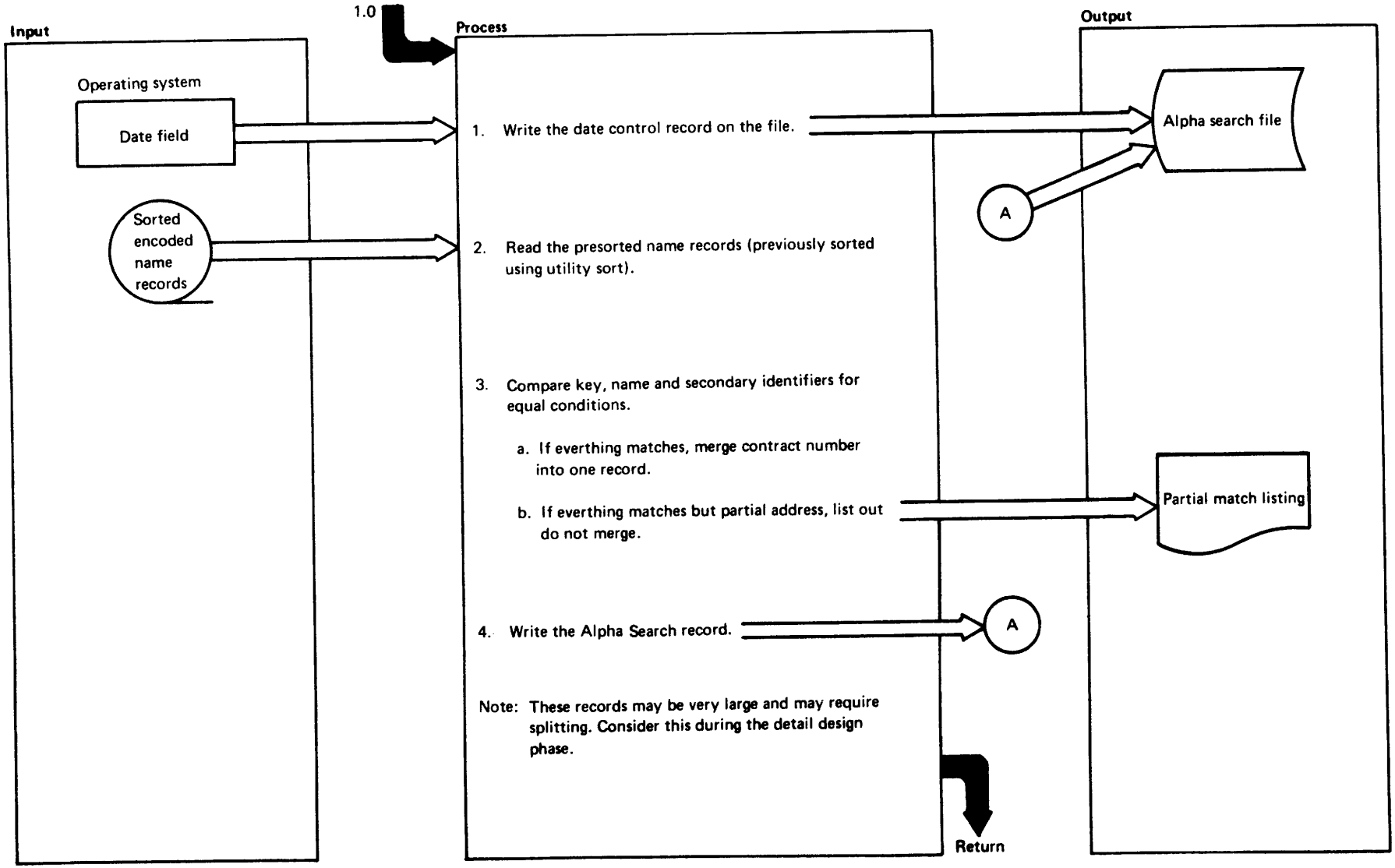
Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 2.2.1 Name: ENCC Description: Encode Commercial Names



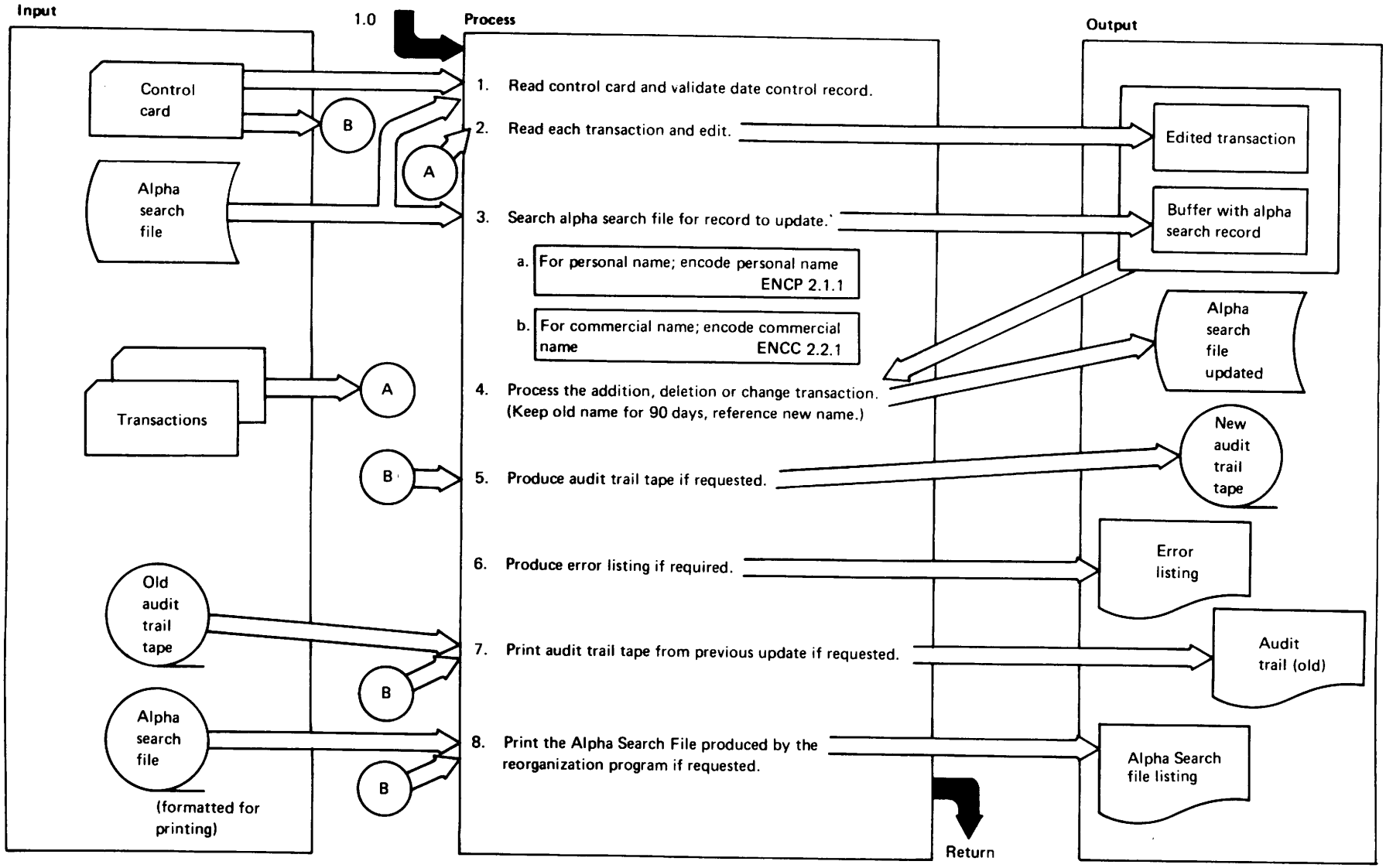
Author: J. Rennaker System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 3.0 Name: FILELOAD Description: Load the Alpha Search File



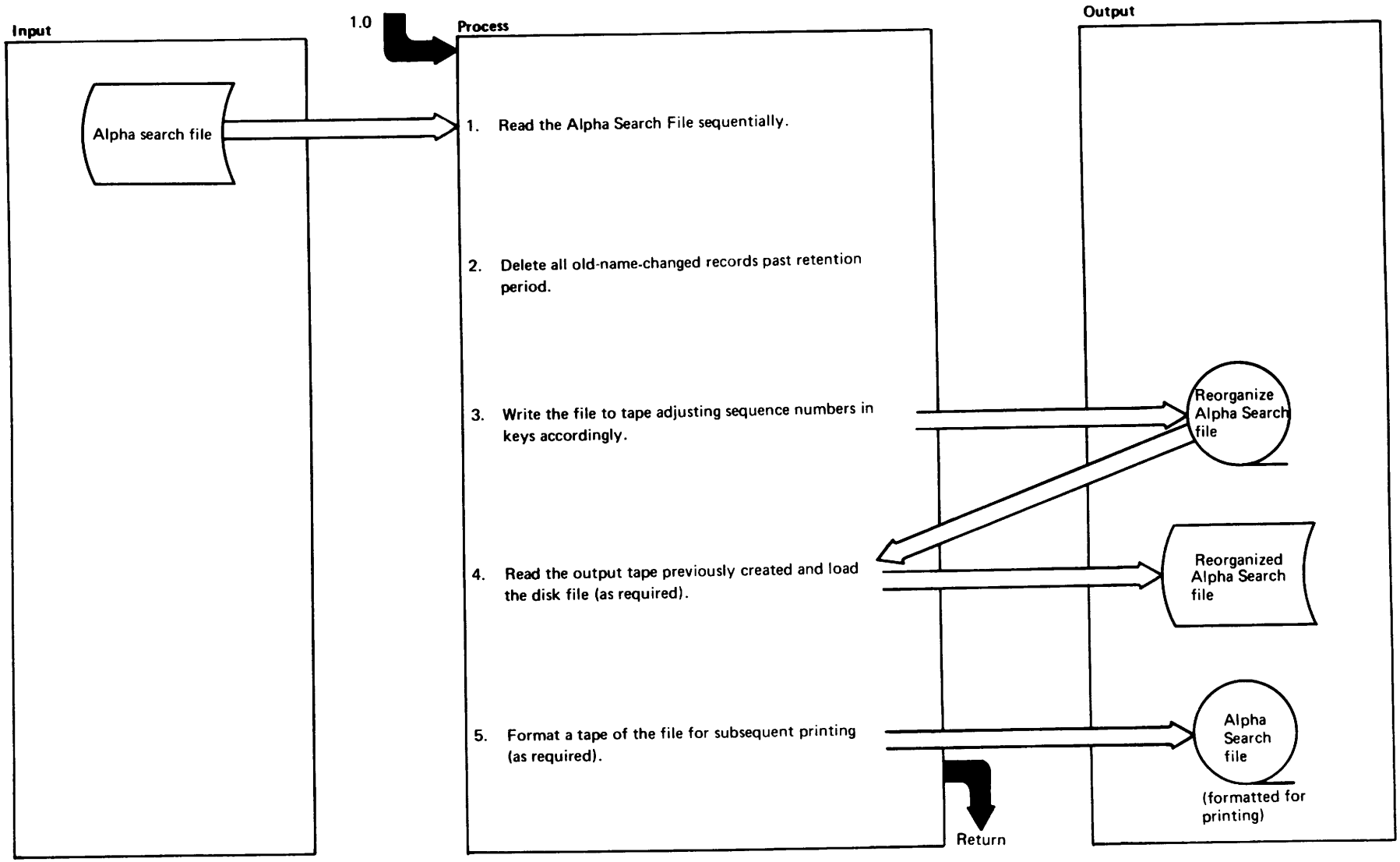
Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 4.0 Name: UPDATE Description: Update the Alpha Search File



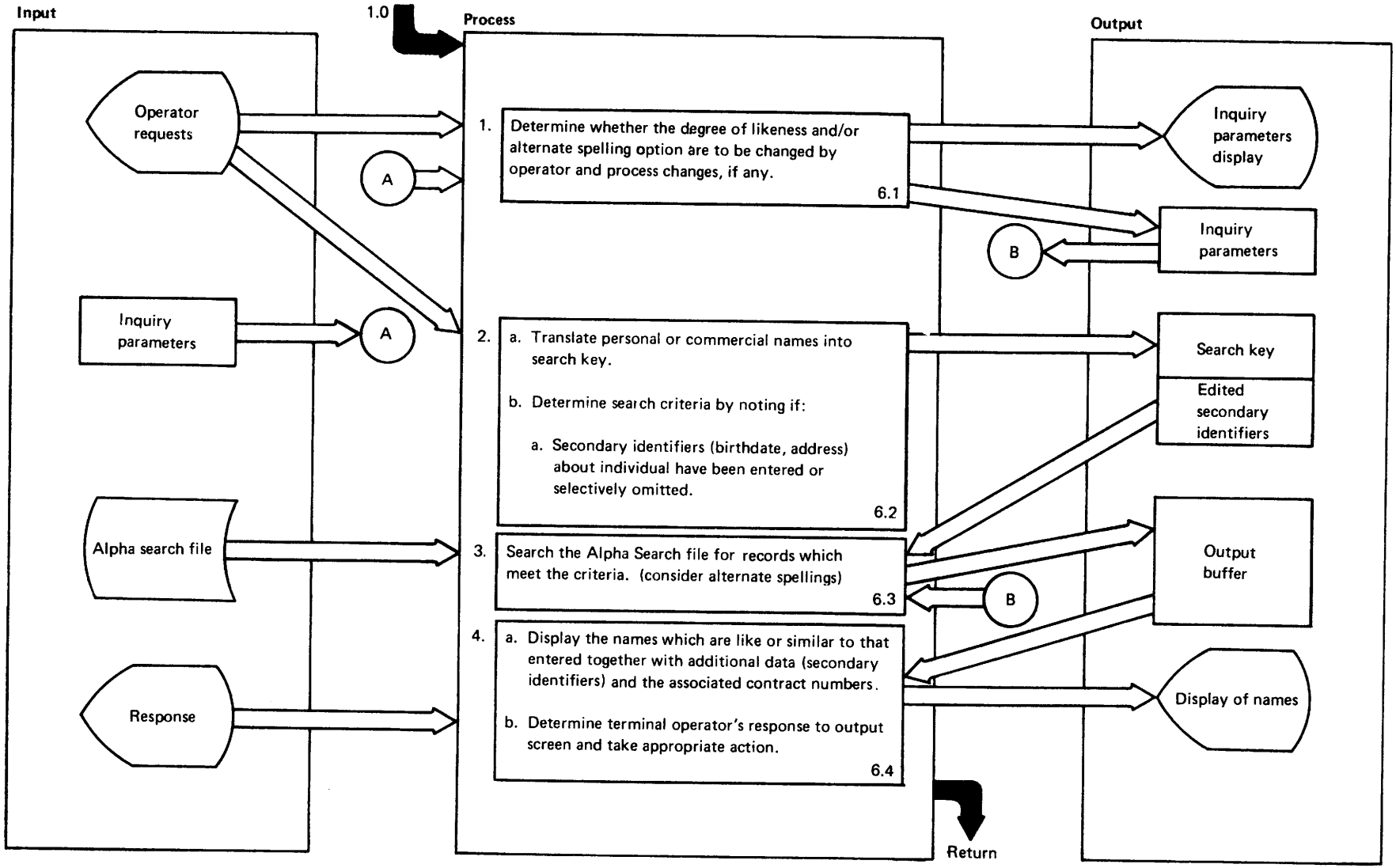
Author: J. Rennaker System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 5.0 Name: REORG Description: Reorganize the Alpha Search File



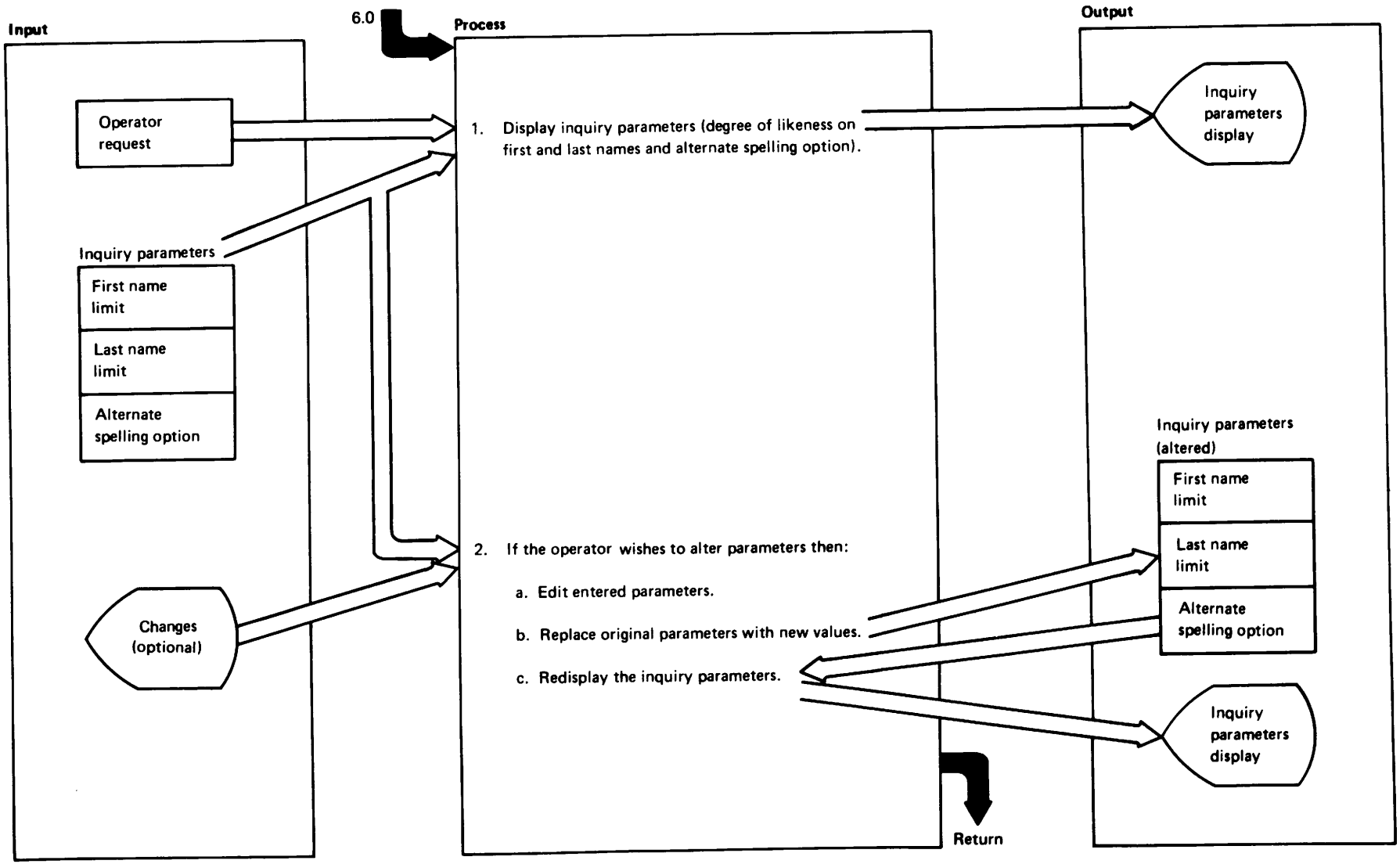
Author: T. Baloun | System/Program: Alpha Search Inquiry System | Date: 10/25/74 | Page: 1 of 1

Diagram ID: 6.0 | Name: INQUIRY | Description: Inquire Against the Alpha Search File



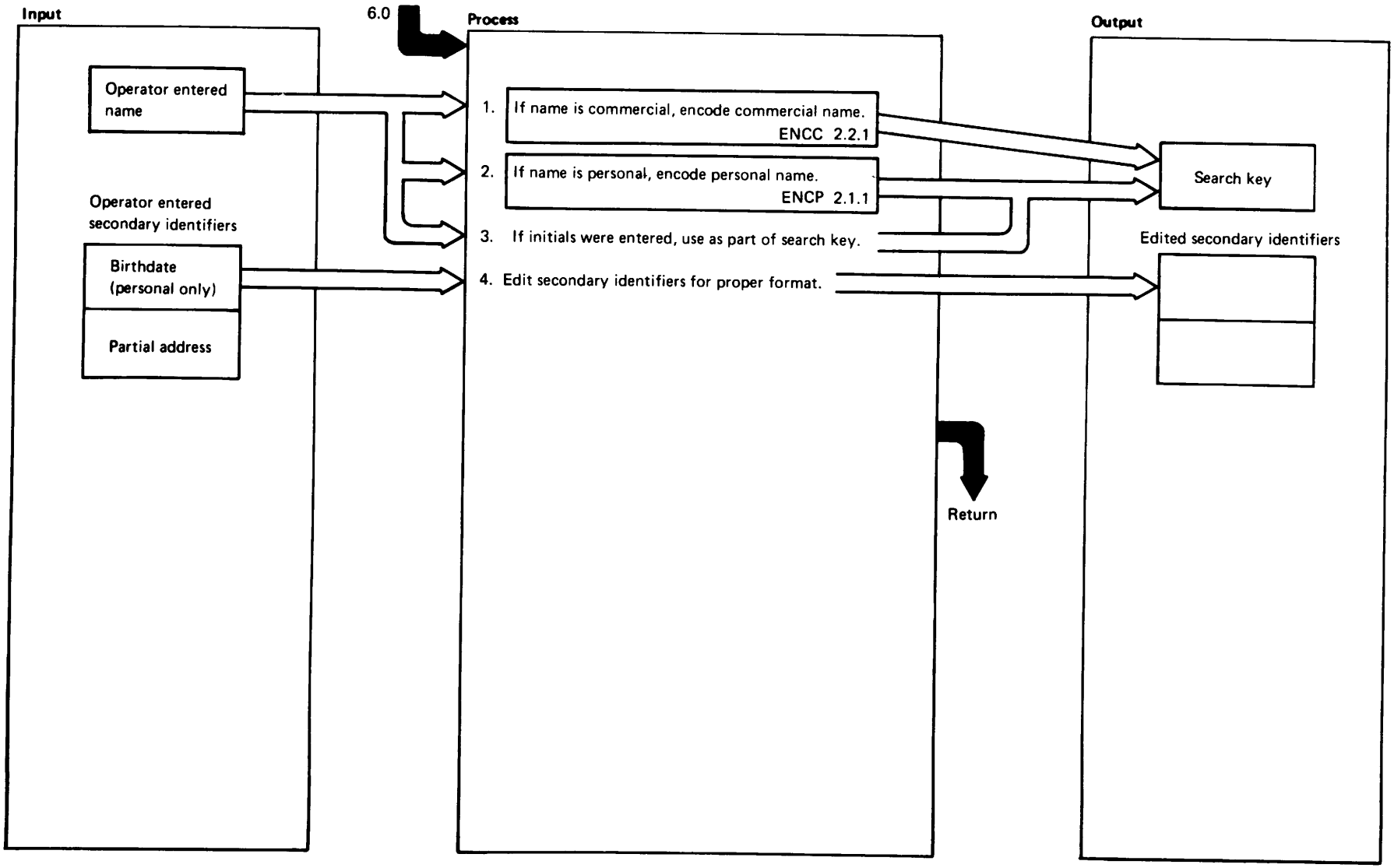
Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.1 Name: Description: Handle Inquiry Parameters



Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

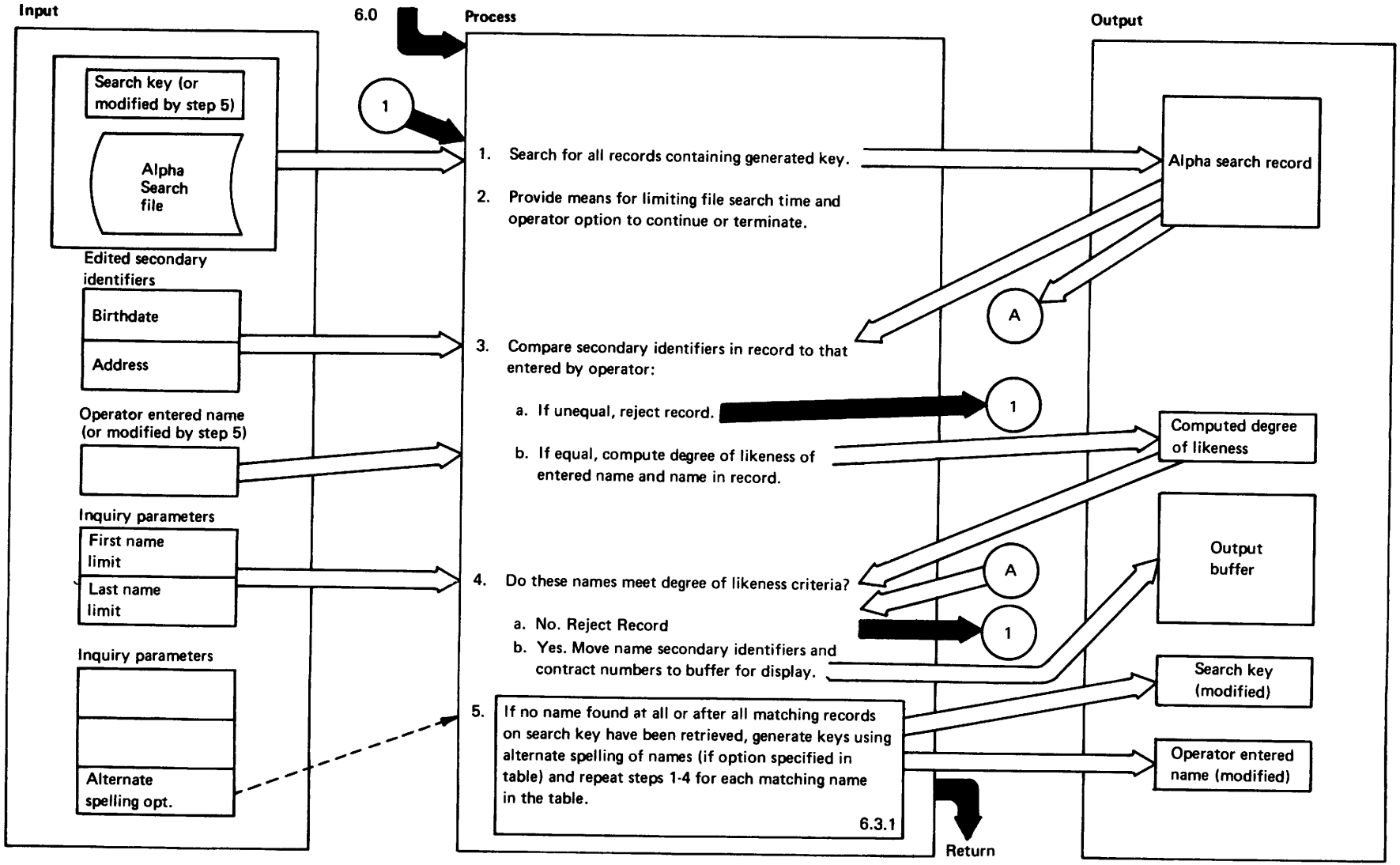
Diagram ID: 6.2 Name: Description: Determine Search Criteria



This page intentionally left blank.

Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.3 Name: Description: Search Alpha Search File For Names



Extended Description

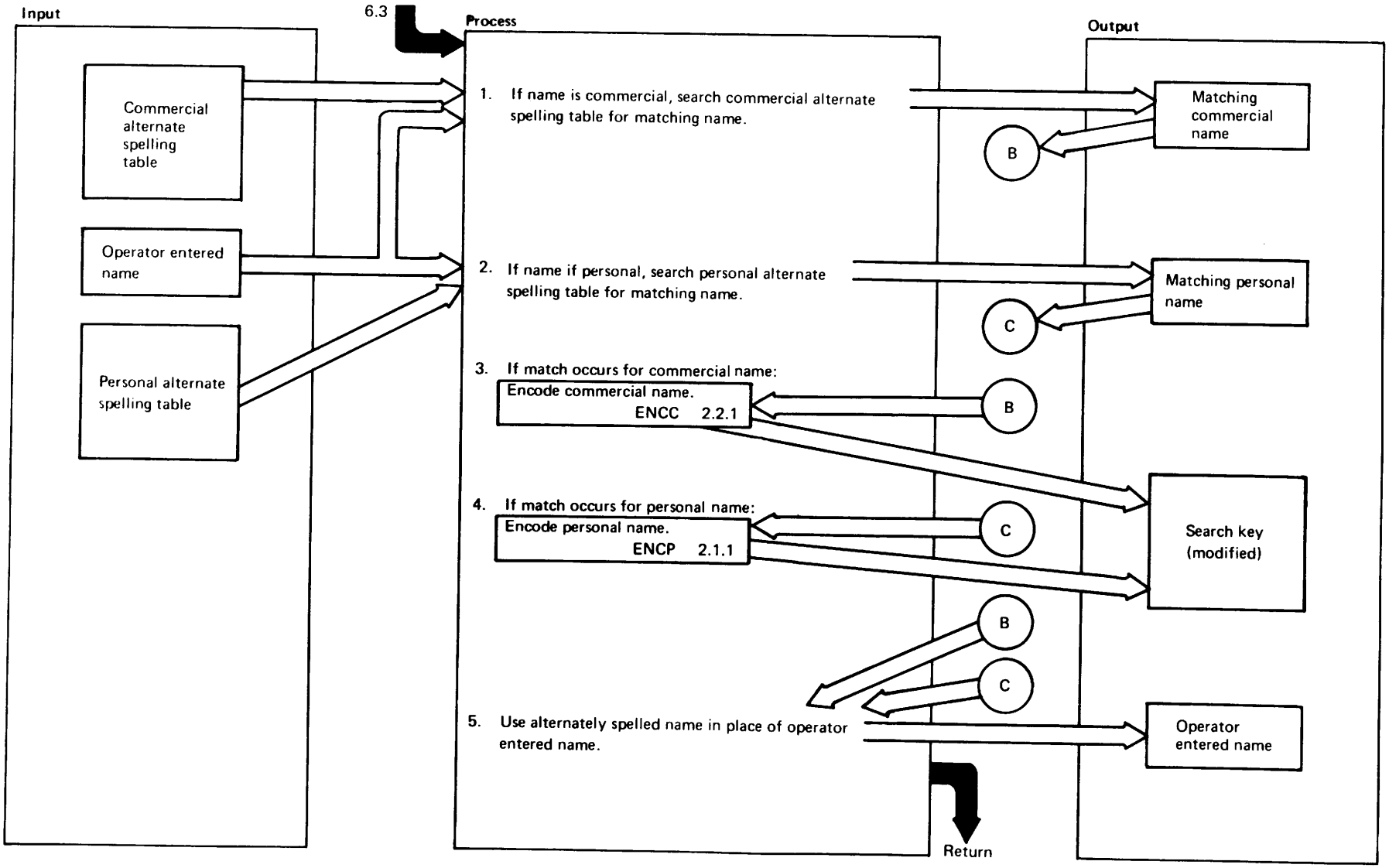
Notes	Routine	Label	Ref.
1. This routine is entered from a request to	DSPLY		6.0A
change the inquiry parameters or after the			TWA
operator has entered data to alter the inquiry			
parameters.			
3. If there is an error in last name limit, move the	DSPLY		
value in error to buffer then move the default			
(075) to PARMTBL LL, (reset LLERR to 0)			
also move edit error message to buffer. Do			
likewise for first name limit error (FLERR)			
FL default is 075. For alternate spelling			
error (CRERR) CR default is N (no).			
Note: These errors were detested in ALTER			
5. Use CICS macro DFHTC.	DSPLY		Fig. 6.2A

Extended Description

Notes	Routine	Label	Ref.

Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

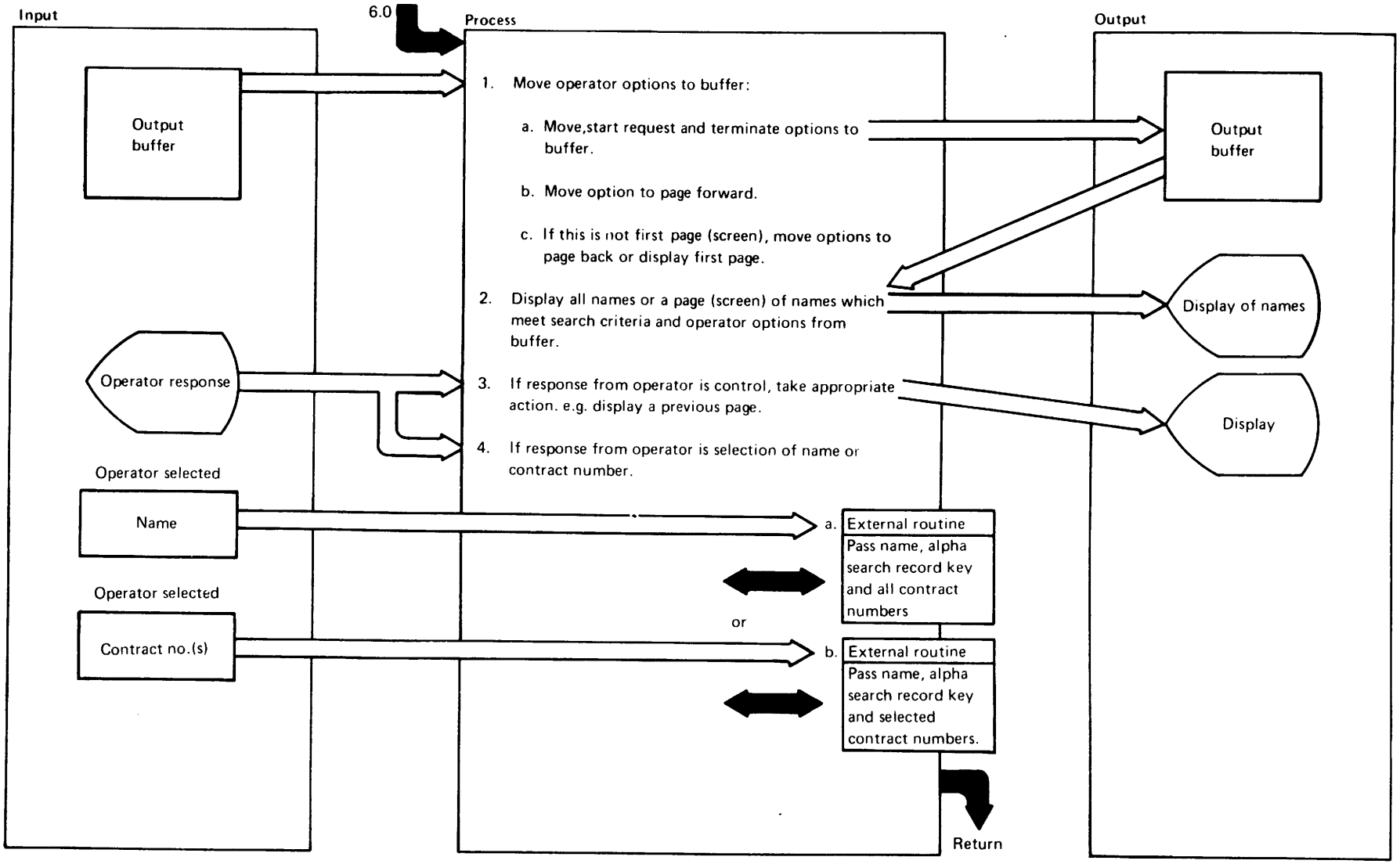
Diagram ID: 6.3.1 Name: Description: Generate Keys Using Alternate Spellings of Names



This page intentionally left blank.

Author: T. Baloun System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.4 Name: Description: Display Names



Extended Description

Notes	Routine	Label	Ref.
When a display screen (buffer) is full, the screen			
will be displayed and the operator will choose			
among these options:			
a. enter another name			
b. stop			
c. search further for more matches			
d. display previously found names which			
match this name			
e. display the first names which were			
found which match this name.			
f. pass a name and selected contract			
numbers to an external routine.			
g. pass a name and all contract numbers			
to an external routine.			

Extended Description

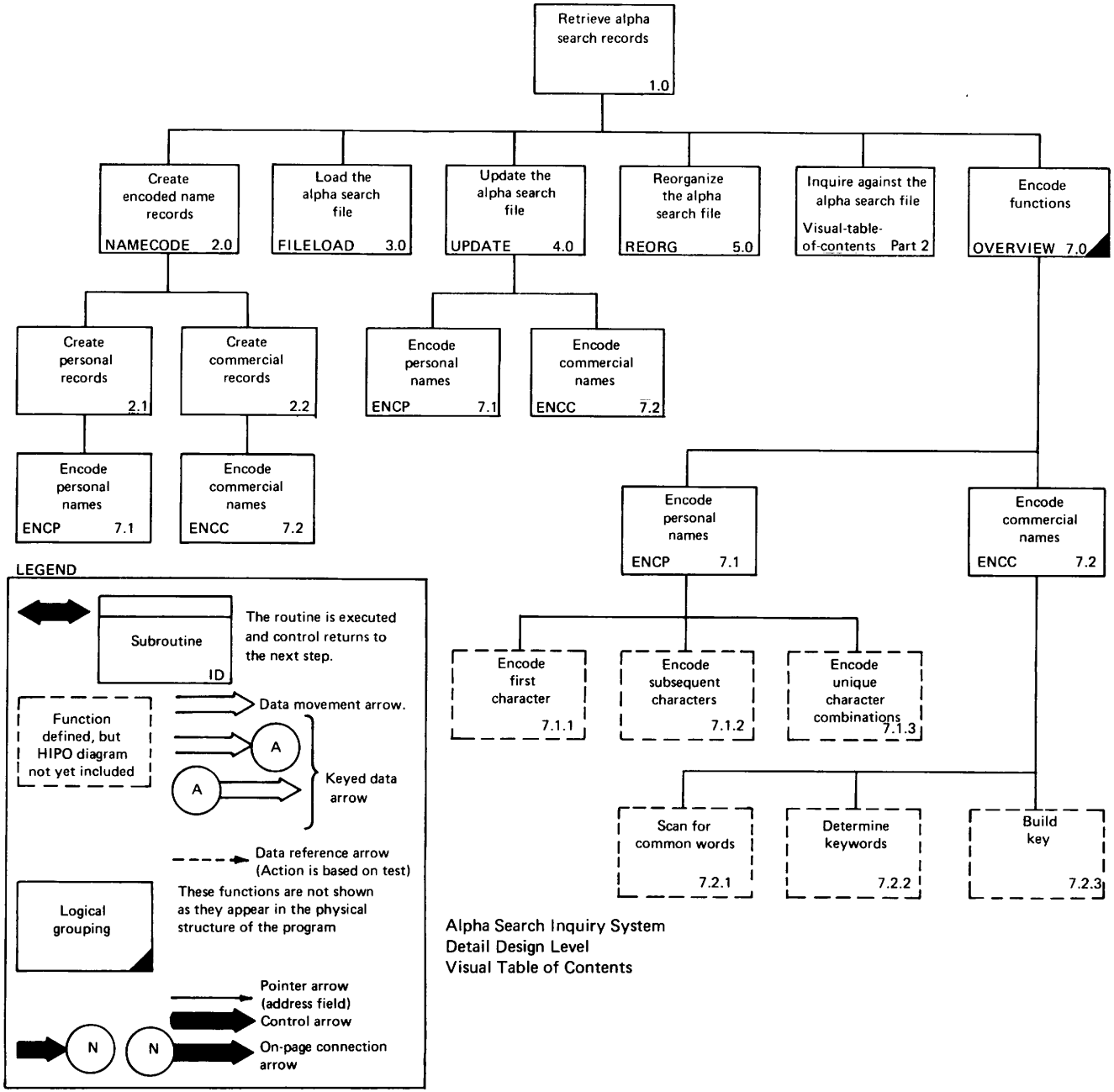
Notes	Routine	Label	Ref.

This page intentionally left blank.

Appendix B: Alpha Search Inquiry System Detail Design HIPO Package

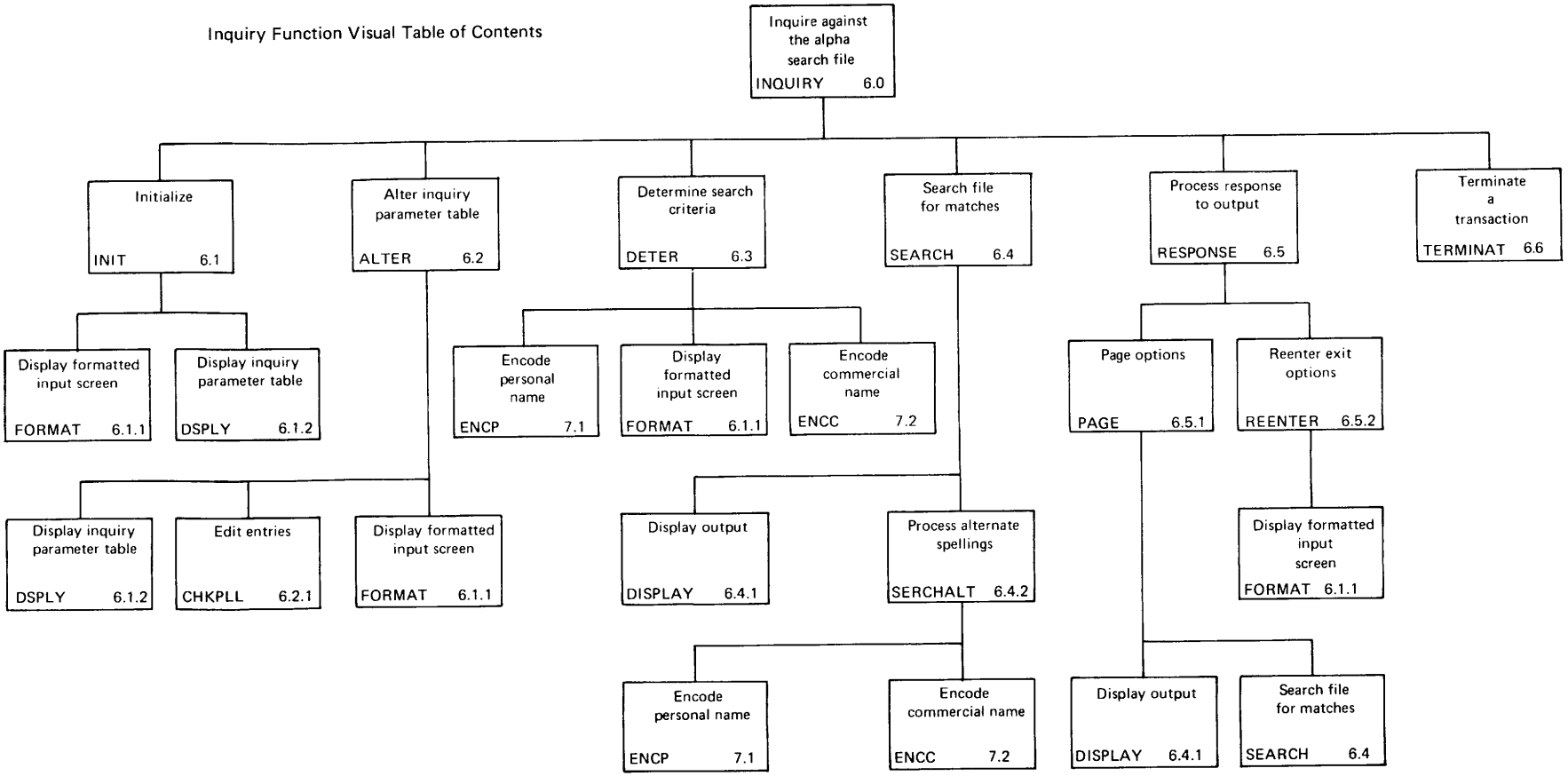
Included in this appendix are some diagrams that have appeared previously in the text.

Only the inquiry function of the initial design HIPO package in Appendix A—and only certain subfunctions of that function—have been expanded here. The fully expanded subfunctions are initialize (Diagram 6.1) and alter inquiry parameter table (Diagram 6.2).



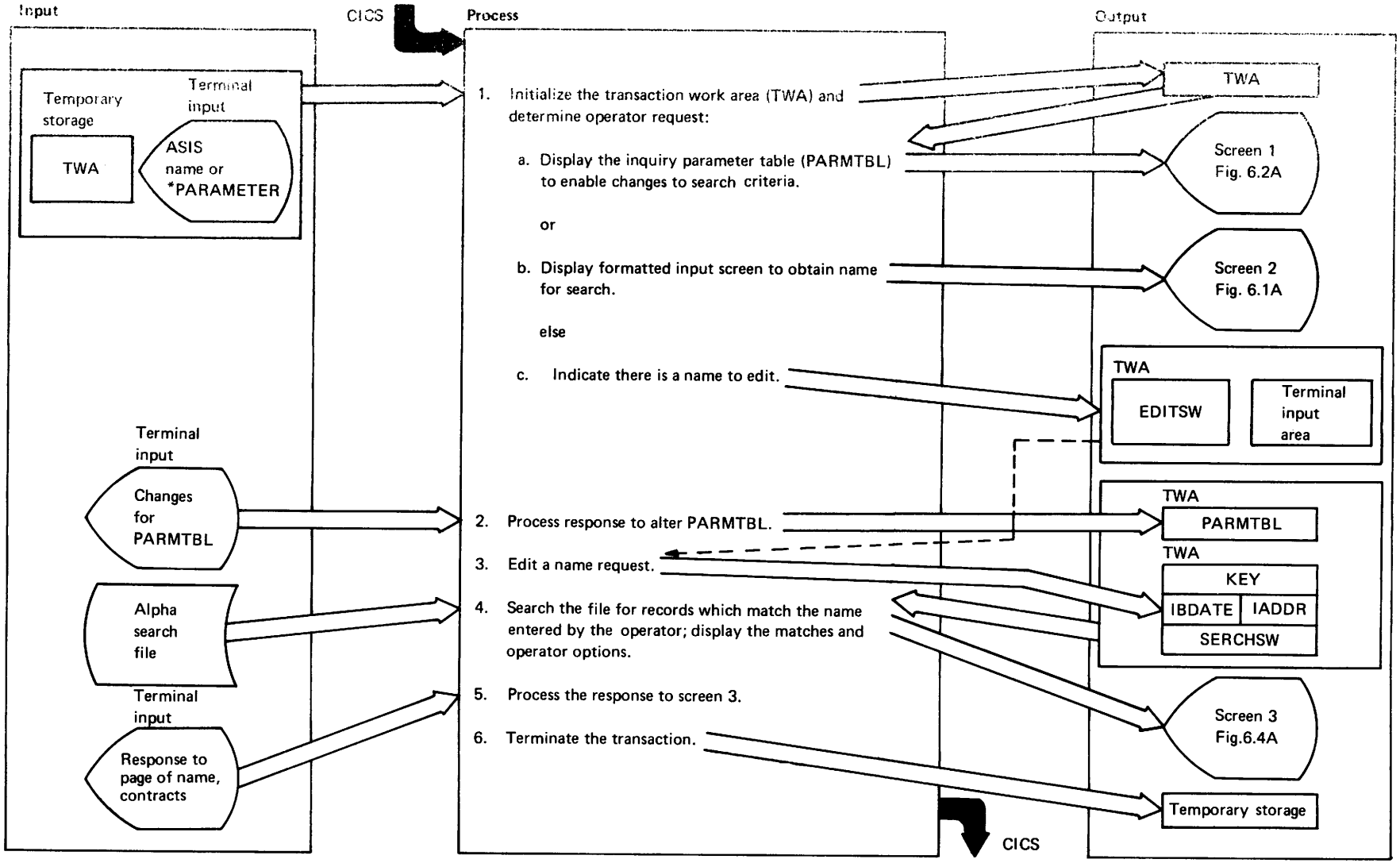
Alpha Search Inquiry System
Detail Design Level
Visual Table of Contents

Inquiry Function Visual Table of Contents



Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.0 Name: INQUIRY Description: Inquire Against the Alpha Search File



Extended Description

Notes	Routine	Label	Ref.
All files are opened by CICS. The TWA contains	INQUIRY		6.0A
all transaction dependent data.			
1. Temporary storage will contain the TWA by	INQUIRY	INIT	6.1
terminal ID if this is a continuation of a			Fig. 6.1A
transaction.			
a. The operator may wish to change the			
first or last name degrees of likeness or the			
alternate spelling option which are the			
search criteria in the PARMTBL-ALTERSW			
and CICSRET are then set on.			
b. RFORMSW and CICSRET are set on.			
2. This routine is called if CICSRET is off and	ALTER		6.2
ALTERSW is on.			
3. This routine is called if EDITSW is on. The	DETER		6.3
name is phonetically encoded to build a key			
SERCHSW is set on if there are no edit errors.			
4. The search is performed if SERCHSW is	SEARCH		6.4
on until end of limits for original key and			
alternate spellings or end of time or full			
buffer. The names and options of: PAGE			

Extended Description

Notes	Routine	Label	Ref.
FORWARD, PAGE BACK, PAGE START,			
REENTER, EXIT are displayed; RESPSW			
and CICSRET are set on.			
5. This routine is called if CICSRET is off	RESPONSE		6.5
and RESPSW is on.			
6. Set off CICSRET. The TWA is saved in	INQUIRY	TERMINAT	6.6
temporary storage to enable processing by			
the proper routine after CICS returns			
control to INQUIRY with the operator			
response.			

FIGURE 6.0A
TRANSACTION WORK AREA (TWA)

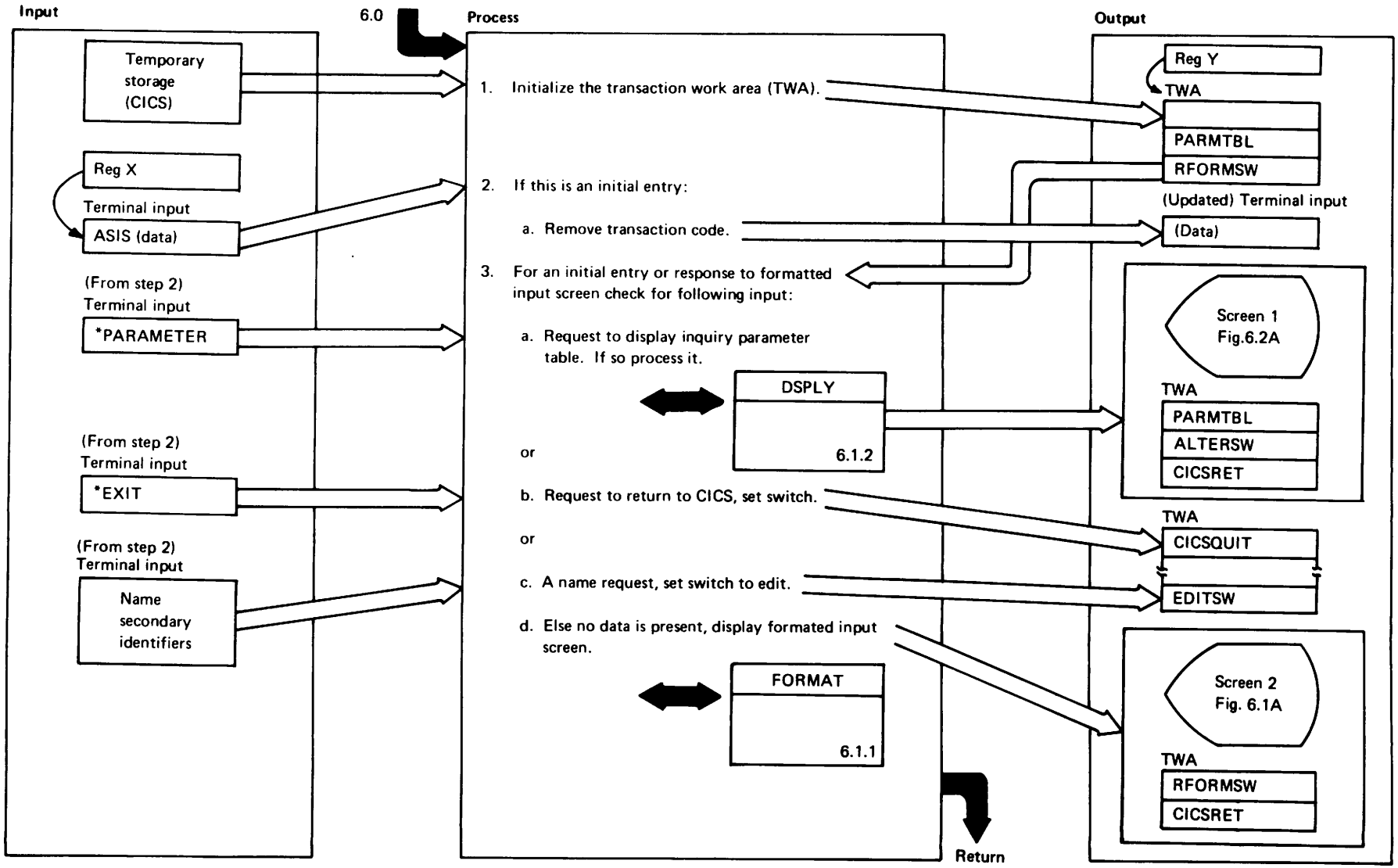
LABEL		USE
CICSRET	CL1	Switch - Set to indicate that control should be returned to CICS and TWA should be saved in temporary storage. When operator enters any data control will return to this transaction.
CICSQUIT	CL1	Switch - Set to indicate release all temporary storage and return control to CICS unconditionally. (End of Transaction).
RFORMSW	CL1	Switch - Set to indicate control should be given to edit the operator's search request in response to the formatted input screen when control is returned from CICS. It is set by FORMAT.
ALTERSW	CL1	Switch - Set to indicate control should be given to routine (ALTER) to alter inquiry parameter table whenever operator enters any data and control is returned from CICS.
PARMTBL	OCL9	Inquiry Parameter Table
LL	CL4	000-100 - % of likeness acceptable in last name.
FL	CL4	000-100 - % of likeness acceptable for first name.
CR	CL1	Y or N alternate spelling table search option.
LLERR	CL1	Error by operator for last name limit.
FLERR	CL1	Error by operator for first name limit.
CRERR	CL1	Error by operator alternate spelling.
EDITERR	CL1	Switch - Set to indicate errors were encountered in editing operator search request.
CNAME	CL30	Name used for compare (This is either INAME or an alternate spelling).
INAME	CL30	Operator entered name.
IBDATE	CL16	Operator entered birthdate.
IADDR	CL30	Operator entered address.
ALNAME	CL30	Current name found in alternate table.
TIOBUF	1F	Address of output buffer for names.
TIADDR	1F	Address of terminal input area.
TIOALEN	1F	Length of data in terminal input area.
EDITSW	CL1	Set by initialize to indicate the operator entered a name request which is to be edited.
UPPERLIM	CL9	Set by encode routine.

LABEL		USE
LOWERLIM	CL9	Set by encode routine.
KEY	CL9	Encoded key.
SERCHSW	CL1	Switch - Set to 1 to indicate that the operator request has been successfully edited and the file should be searched. After processing original data, set to 2 to indicate alternate names are being searched.
SPAGENO	H	Current number of the page being displayed.
PAGENO	H	Number of display pages which have been built containing records which match the search criteria.
NORF	CL1	Switch - Set to indicate records were found which match the request entered by the operator and which passed the degree of likeness tests.
EOFSW	CL1	Switch - Set when all possible records have been searched.
LINECNT	H	The current line number built in the buffer to display the names and data.
CURRKEY	CL9	The key of the current alpha search record being processed.
RESPSW	CL1	Switch - Set on when a response is expected from the operator after a display of a buffer of names.
TIMESW	CL1	Switch - Set on if time expires before search has found a match. This is set by checking an address provided by CICS prior to each file read.
TIMEADR	1F	Address provided by CICS to test end of time limit for search.
CONTRACT	1F	Displacement to the next contract number to be moved to output. 0 if completed all contract numbers of current record.

NOTES:
1. A switch is 0 if it is off, 1 if it is on.
2. Reg X points to the terminal input area.
3. Reg Y points to the transaction work area (TWA).
4. Fields are initialized to zeroes.

Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.1 Name: INIT Description: Initialize

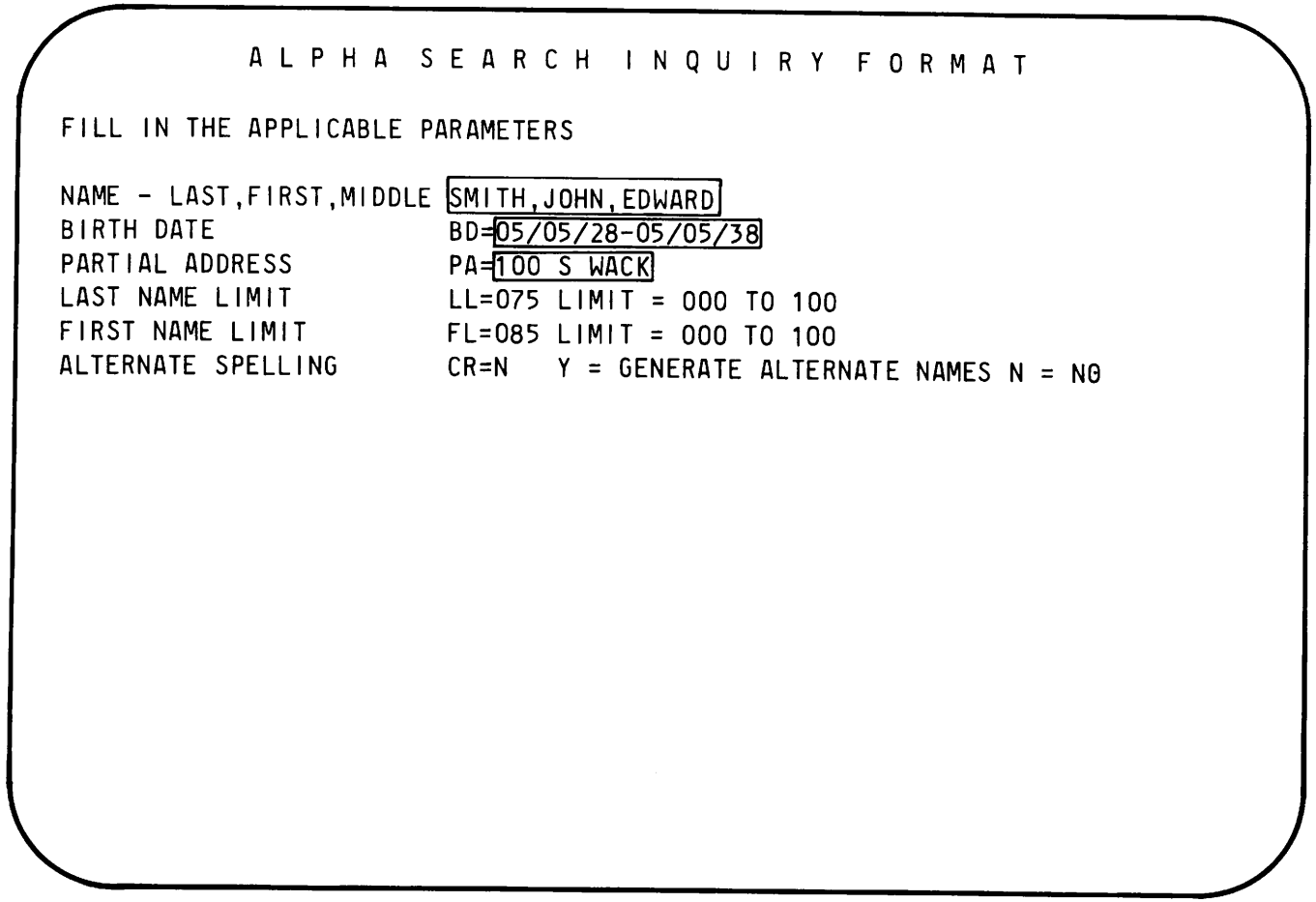


Extended Description

Notes	Routine	Label	Ref.
1. Read CICS temporary storage by terminal ID.	INQUIRY	INIT	TWA
If a TWA is found, it will contain a switch			Fig. 6.0A
to indicate which routine is to process the			
operator input (ALTERSW, RESPSW,			
RFORMSW). If a TWA is not found use			
DFHSC macro to acquire one. Save input			
address in TIADDR and length in TIOALEN			
of TWA.			
3a.	DSPLY		6.1.2
3d.	FORMAT		6.1.1

Extended Description

Notes	Routine	Label	Ref.

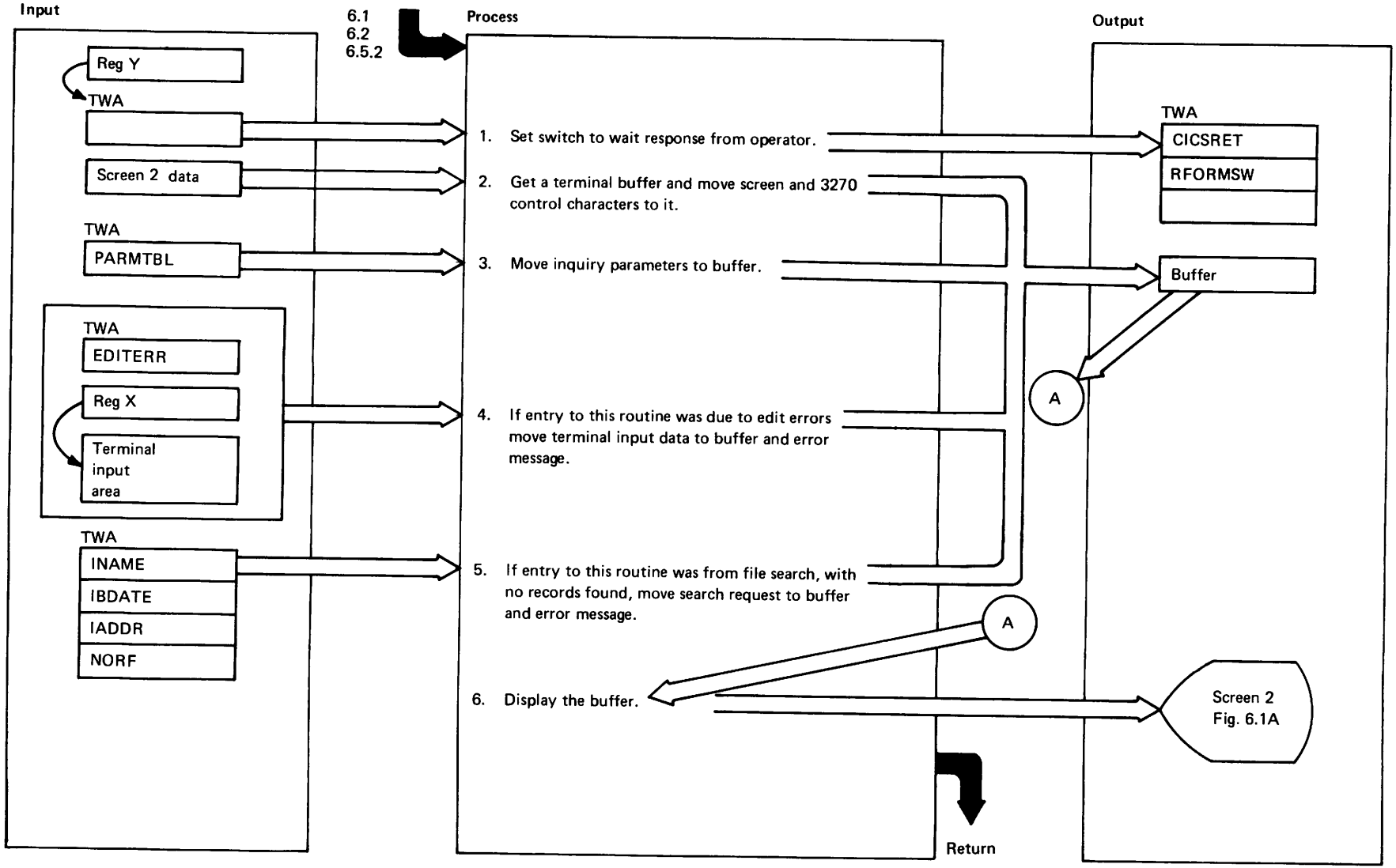


SCREEN 2 FIGURE 6.1A

This page intentionally left blank.

Author: R. Bloon System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.1.1 Name: FORMAT Description: Display Formatted Input Screen



Extended Description

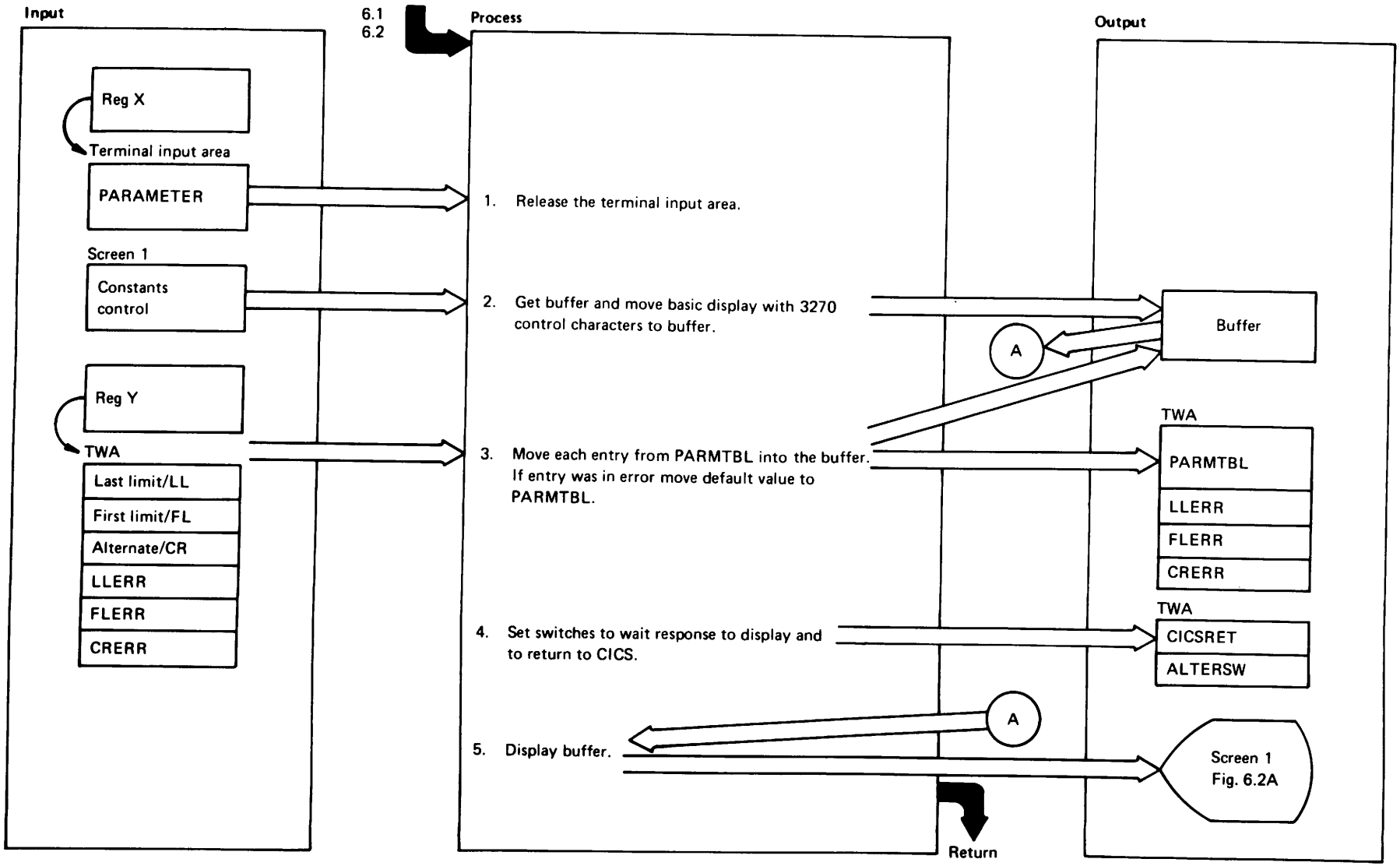
Notes	Routine	Label	Ref.
1. Setting CICSRET and RFORMSW will cause	FORMAT		Fig 6.1A
all further processing to be bypassed and			Fig 6.0A
control to be given to CICS to wait for			(TWA)
operator to enter a name.			
4. Reset EDITERR switch. This was set in	FORMAT		
DETER.			
5. Reset no-record-found switch (NORF). This	FORMAT		
was set in SEARCH.			
6. Reset all information for the search	FORMAT		
SPAGENO			
KEY			
INAME			
EOFSW			
Release current terminal input area.			

Extended Description

Notes	Routine	Label	Ref.

Author: D. Clark System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.1.2 Name: DSPLY Description: Display Inquiry Parameter Table



Extended Description

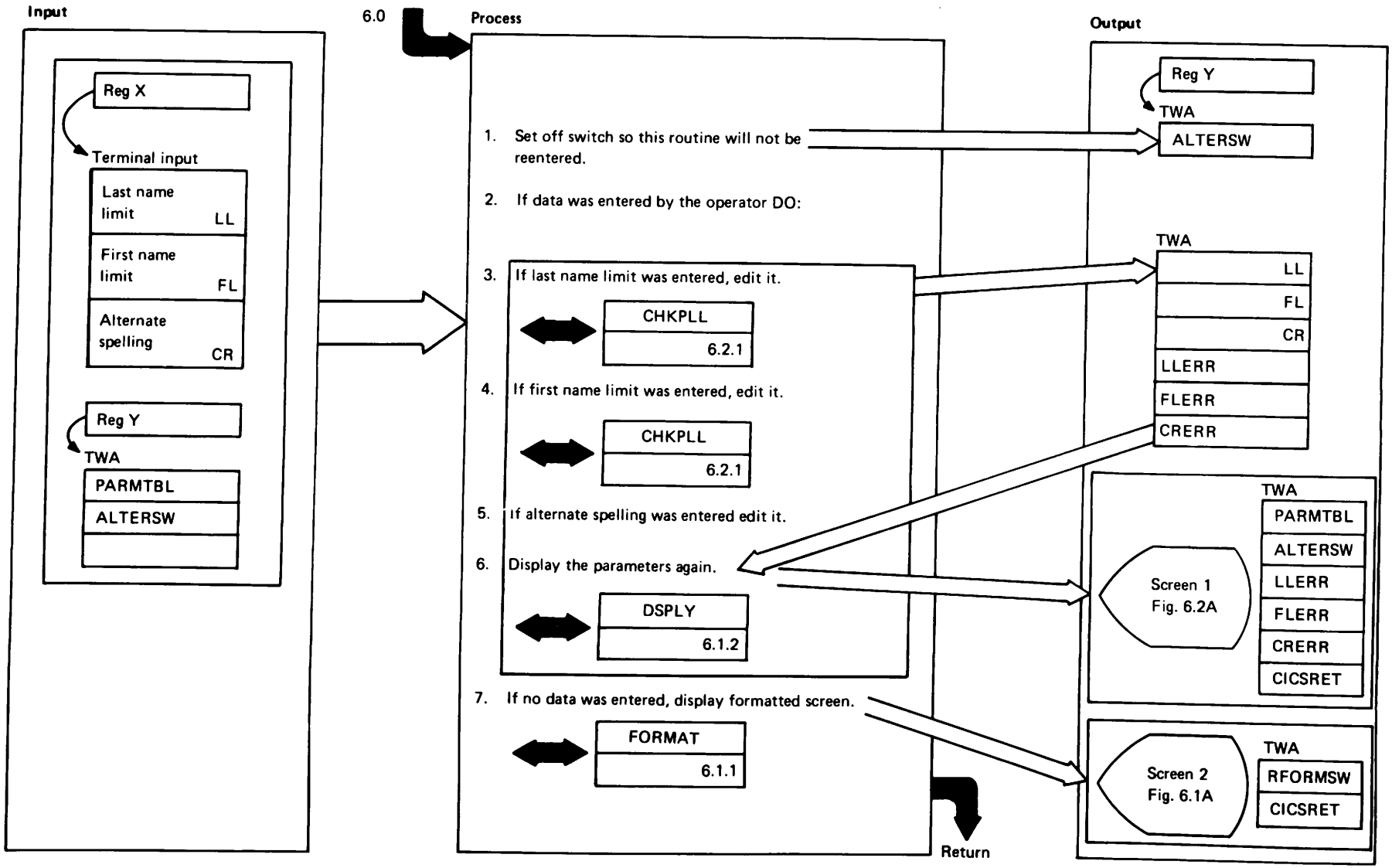
Notes	Routine	Label	Ref.
1. This routine is entered from a request to	DSPLY		6.0A
change the inquiry parameters or after the			TWA
operator has entered data to alter the inquiry			
parameters.			
3. If there is an error in last name limit, move the	DSPLY		
value in error to buffer then move the default			
(075) to PARMTBL LL, (reset LLERR to 0)			
also move edit error message to buffer. Do			
likewise for first name limit error (FLERR)			
FL default is 075. For alternate spelling			
error (CRERR) CR default is N (no).			
Note: These errors were detested in ALTER			
5. Use CICS macro DFHTC.	DSPLY		Fig. 6.2A

Extended Description

Notes	Routine	Label	Ref.

Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.2 Name: ALTER Description: Alter Inquiry Parameter Search Table

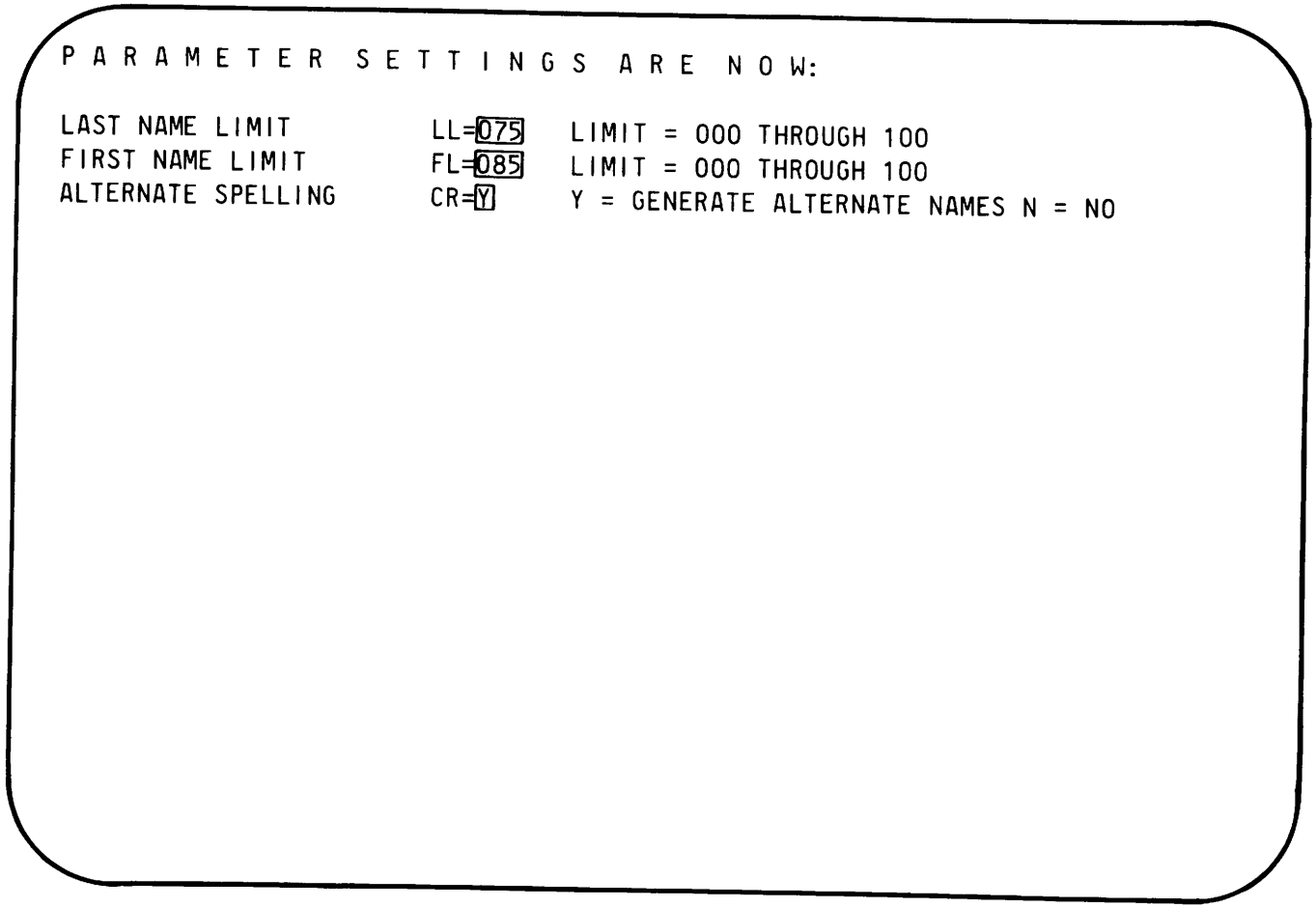


Extended Description

Notes	Routine	Label	Ref.
1. Switch was set by DSPLY to cause entry after	ALTER		
operator response.			
3 & 4	ALTER	CHKPLL	6.2.1
Reg. X points to input register Z points to			
appropriate field in the PARMTBL Register Z			
on exit will have the original input value if			
it was in error or 0 if it is not. PARMTBL			
will be changed whether or not there was an			
edit error. Set LLERR or FLERR on as			
appropriate.			
5. Alternate spelling entry must be a single	ALTER		
character of Y or N, else move entered data			
to CR field and set CRERR on.			
6. Always call DSPLY to show changed	DSPLY		6.1.2
PARMTBL & reset errors (until operator			6.2A
enters no data)			
7. If no data is detected (operator hit enter	FORMAT		6.1.1
key without data).			6.1A

Extended Description

Notes	Routine	Label	Ref.

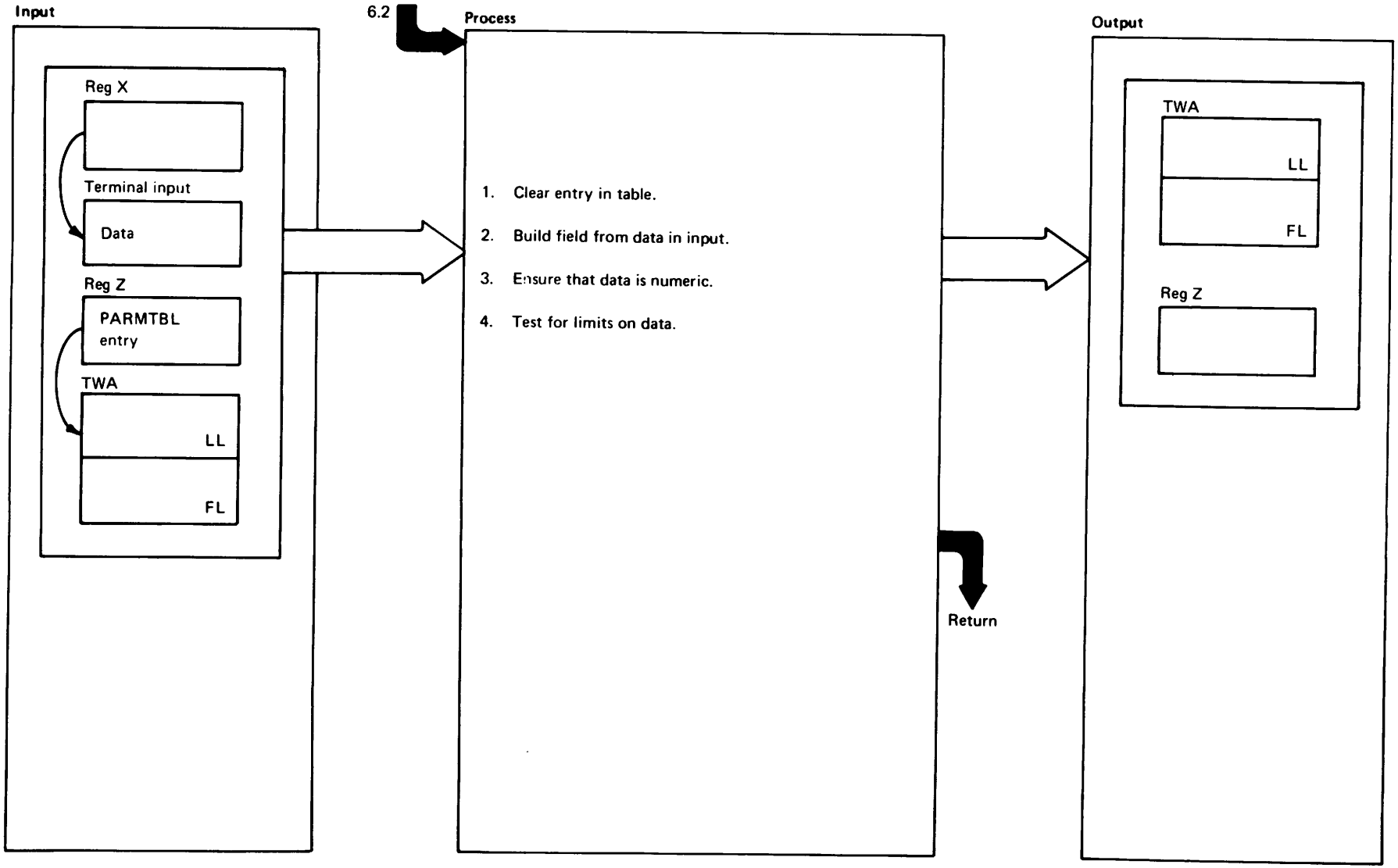


SCREEN 1 FIGURE 6.2A

This page intentionally left blank.

Author: C.Valenti System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.2.1 Name: CHKPLL Description: Edit Entries



Extended Description

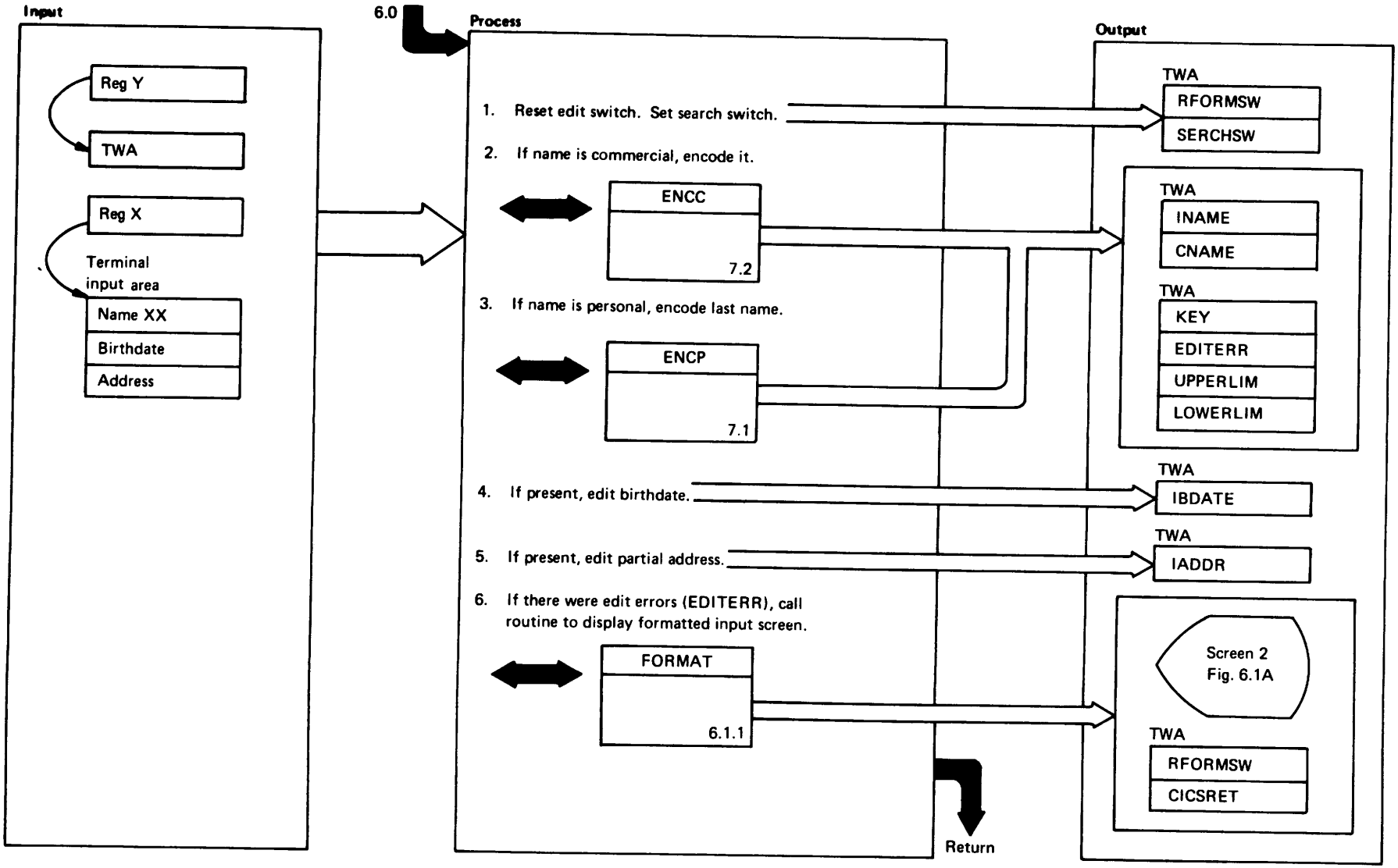
Notes	Routine	Label	Ref.
2. The maximum length for the last name limit	ALTER	CHKPLL	
and first name limit is 4. If data exceeds			
this, return input value in Register Z. The			
data is terminated by a 3270 SBAO character,			
comma or end of data.			
3. If entry is not numeric return entered value in	ALTER	CHKPLL	
Register Z.			
4. If data is not a value between 000-100 return	ALTER	CHKPLL	
entered value in Register Z. If no errors were			
found return 0 in Register Z.			
2-4.			
Always move entered value to the field in			
PARMTBL.			

Extended Description

Notes	Routine	Label	Ref.

Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.3 Name: DETER Description: Determine Search Criteria



Extended Description

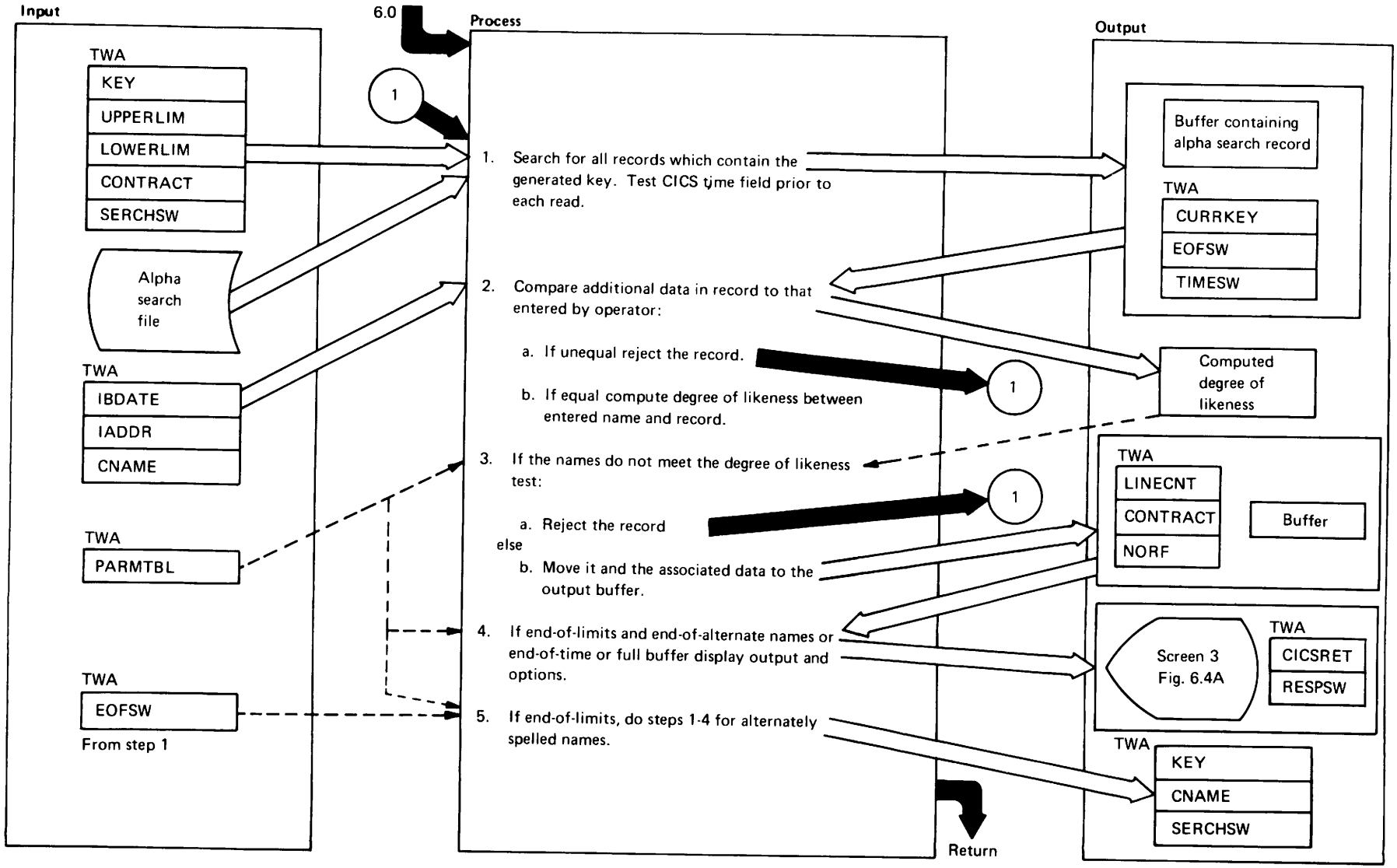
Notes	Routine	Label	Ref.
1. Set SERCHSW=1.	DETER		6.0A
			TWA
2. Save the original field in INAME and CNAME	DETER		
before encoding it. If the name is preceeded by	ENCC		7.2
a number it is commercial. Encode translates	ENCP		7.1
the name to a key and sets upper and lower			
limits for the search. If there is an error,			
set EDITERR on after return and set SERCHSW			
off.			
3. If first and/or middle initial are not entered	DETER		
set lower limit to blank and upper limit to			
Z for initials in key. Otherwise use first and			
middle initials provided. This is used to limit			
search to those names with the same initials			
as those entered. If there is an error set			
EDITERR on and set off SERCHSW.			
4 & 5. Release the terminal input area.	FORMAT		6.1.1
			6.1A

Extended Description

Notes	Routine	Label	Ref.

Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.4 Name: SEARCH Description: Search File for Matches

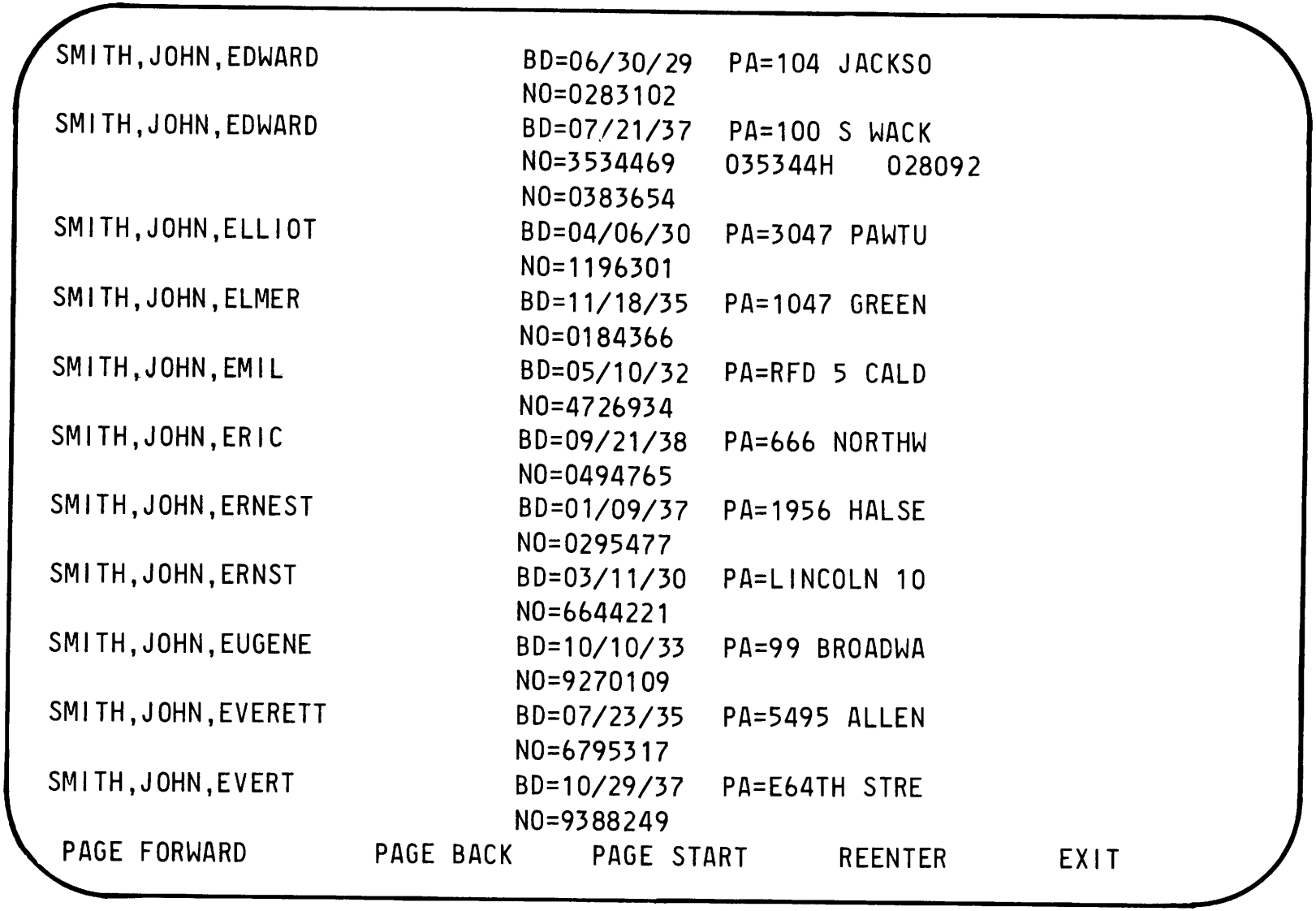


Extended Description

Notes	Routine	Label	Ref.
1. Prior to beginning the search the file limits are	SEARCH		
set and CICS is provided the time increment.			
CICS returns the address of the field in which			
the time will be posted. If this routine is			
entered after a PAGE FORWARD and			
CONTRACT is not zero step 3b is executed			
beginning with this contract number. (CICS			
updates CURRKEY and EOFSW).			
2. Birthdate and partial address are optional	SEARCH		
If they are present and the fields in the record			
are blank, the compare is considered equal.			
3. Before moving to be buffer check LINECNT	SEARCH		
If it is 0 acquire an output buffer. Else move			
the data from the record to the buffer. Set			
on NORF.			
4. Display the output and set on CICSRET and	DISPLAY		6.4.1
RESPSW to await operator response.			Fig 6.4A
5. Alternate spellings are checked if CR=Y in	SERCHALT		6.4.2
PARMTBL. Set SERCHSW to 2.			

Extended Description

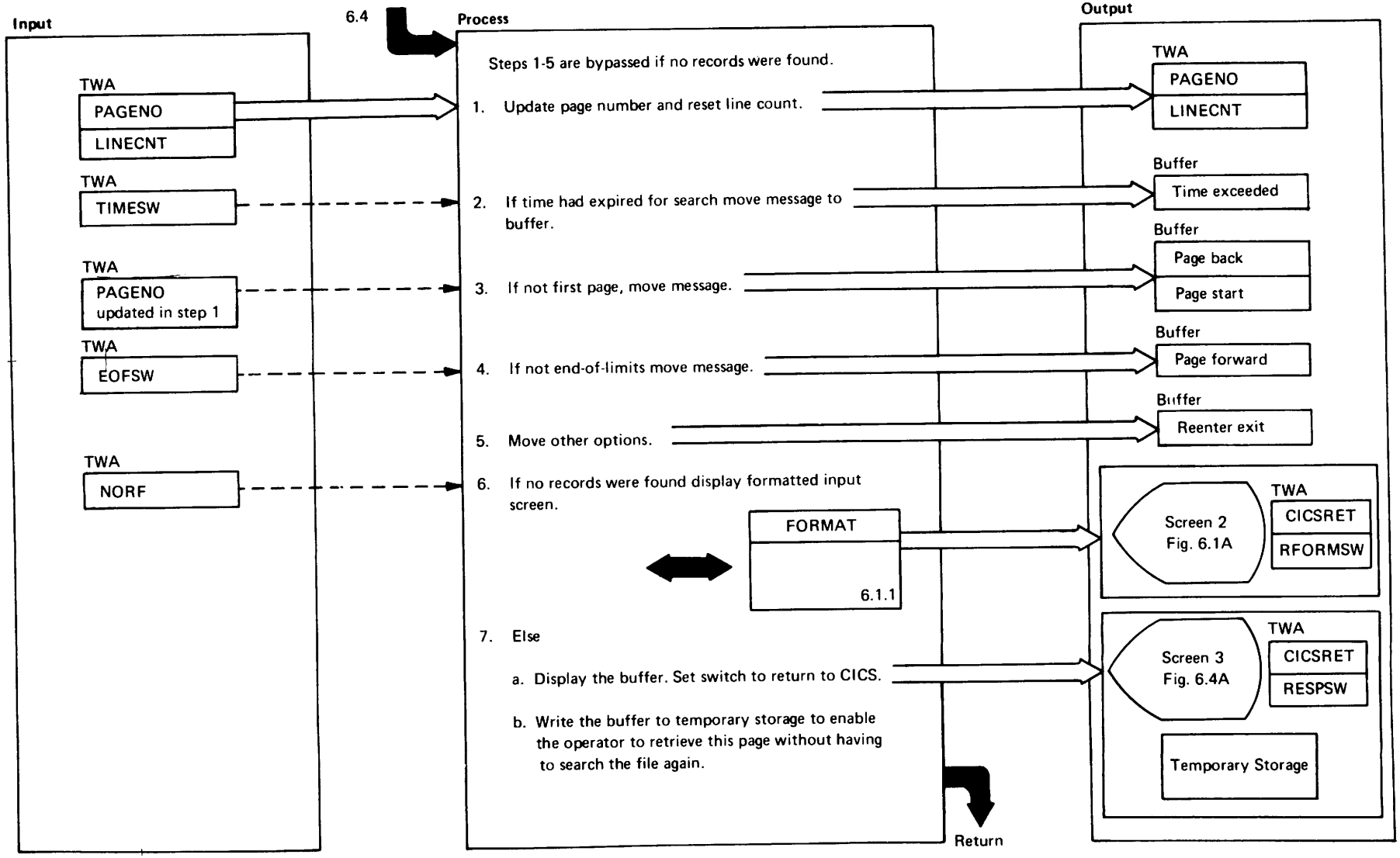
Notes	Routine	Label	Ref.



SCREEN 3 FIGURE 6.4A

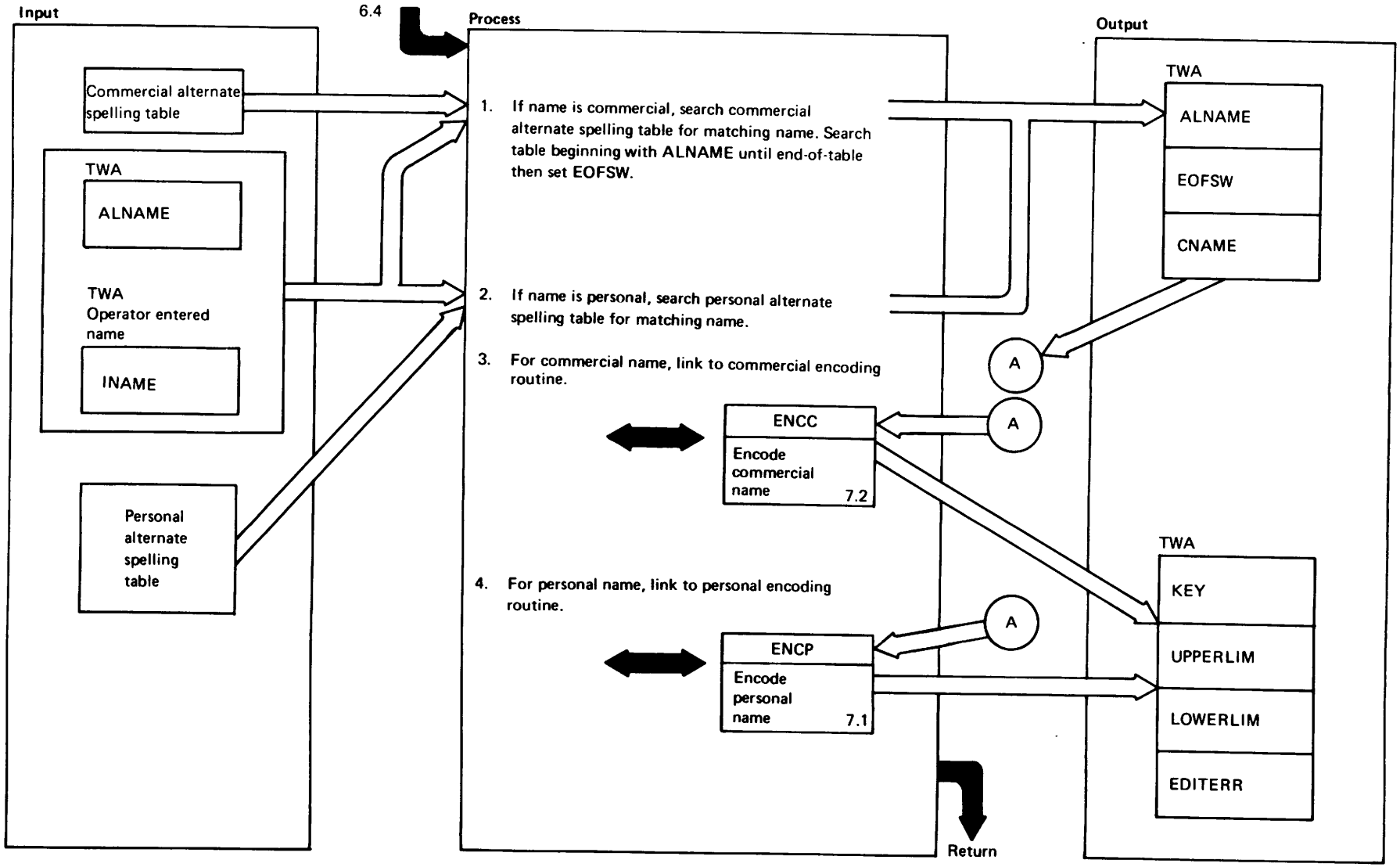
Author: J. Woods | System/Program: Inquiry | Date: 10/25/74 | Page: 1 of 1

Diagram ID: 6.4.1 | Name: DISPLAY | Description: Display Output



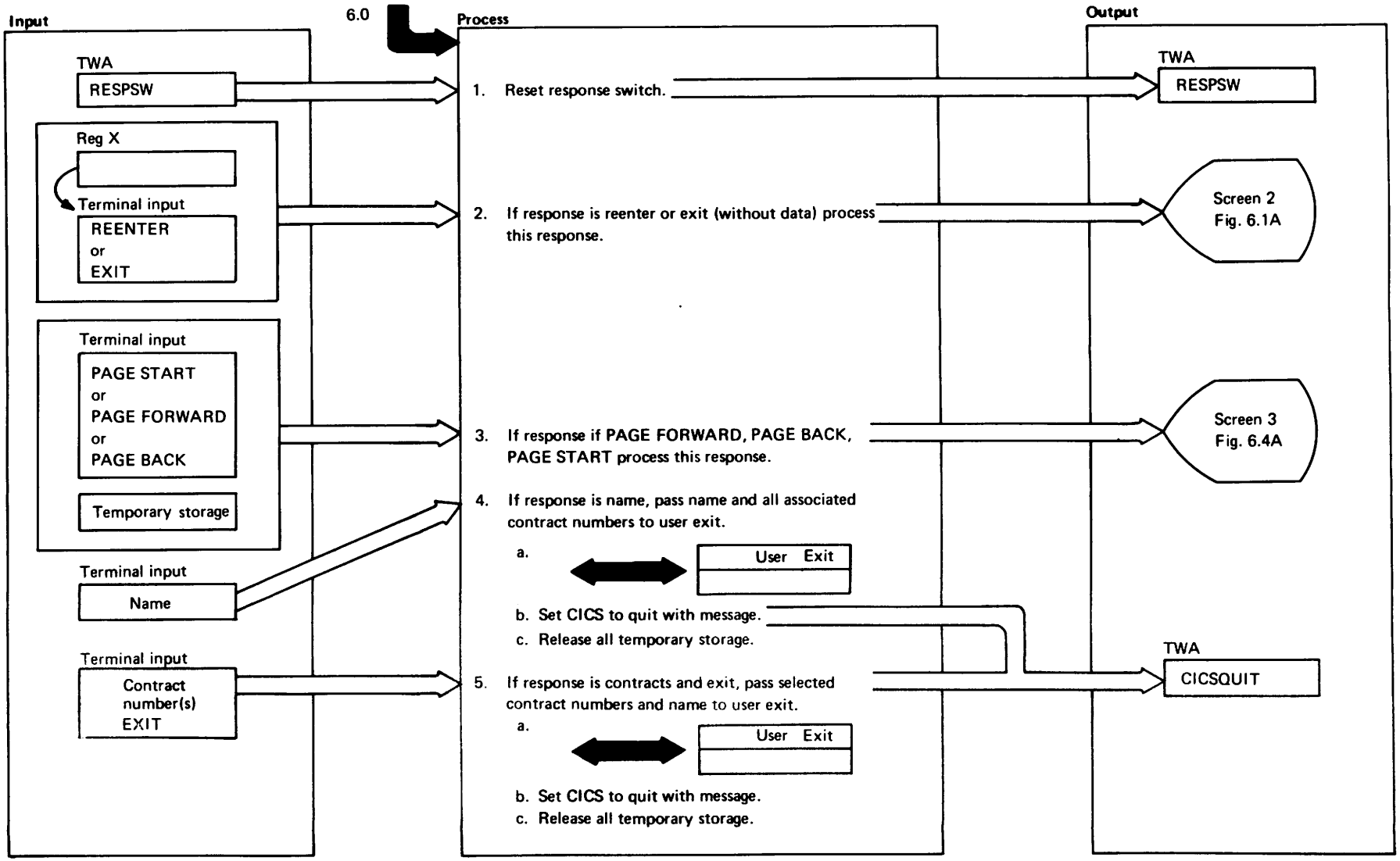
Author: K. Otto System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.4.2 Name: SERCHALT Description: Process Alternate Spellings



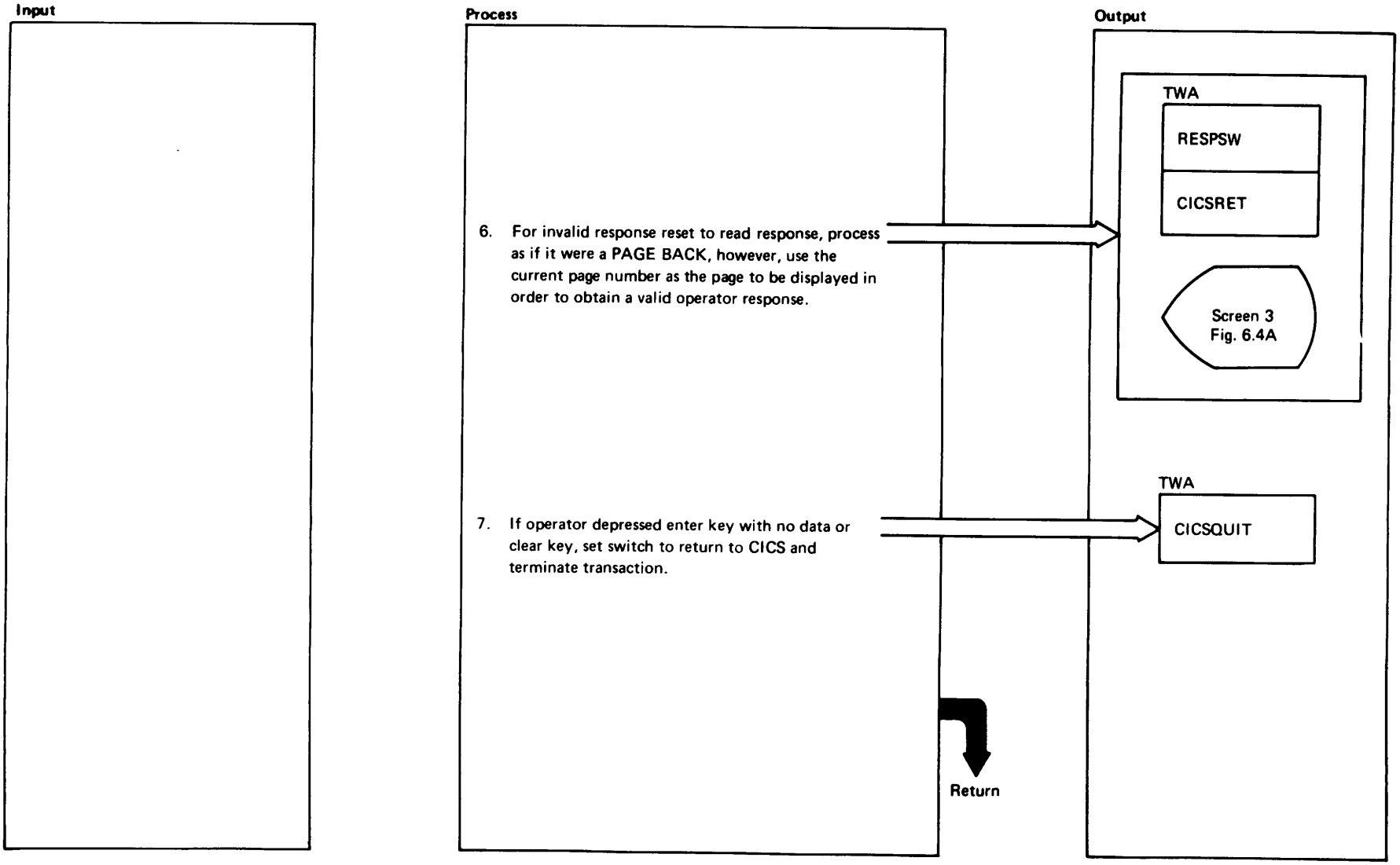
Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 2

Diagram ID: 6.5 Name: RESPONSE Description: Process Response to Output



Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 2 of 2

Diagram ID: 6.5 Name: RESPONSE Description: Process Response to Output



Extended Description

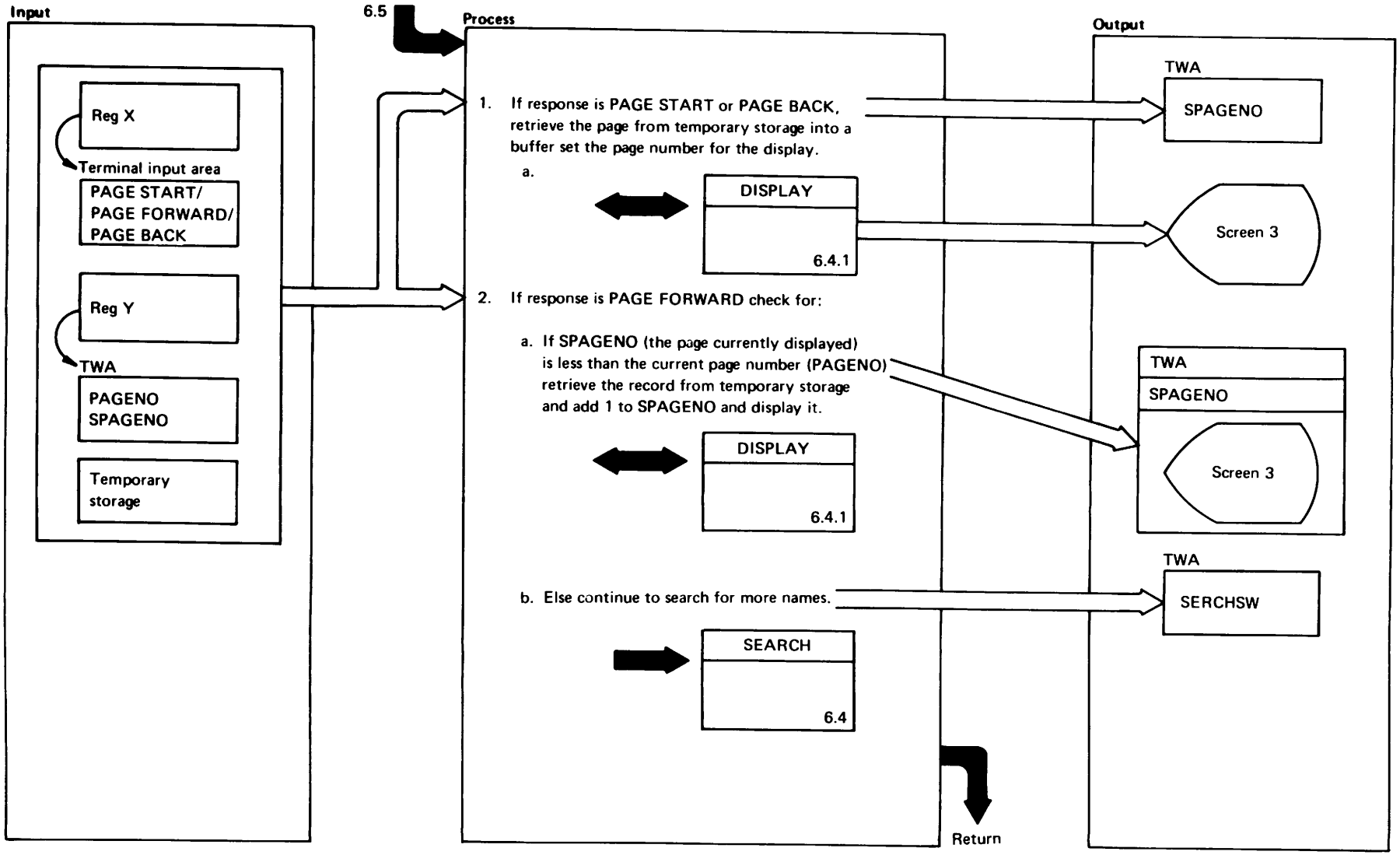
Notes	Routine	Label	Ref.
2.	RESPONSE	REENTER	6.5.2
3. For PAGE BACK and PAGE START retrieve	RESPONSE	PAGE	6.5.1
the page from temporary storage. For PAGE			
FORWARD the search routine will be called to			
retrieve more names.			
4. The name of the routine must be in the CICS	INQUIRY 4		N/A
program processing table. A CICS LINK is			
done to this program. (Same is true for Step 5)			
Release the input buffer.			
5. The contract number fields are not immediate	INQUIRY 4		N/A
data (one or more numbers may be selected).			
The operator should select EXIT after all			
desired numbers (or depress the enter key).			
If any other options are chosen before the			
EXIT, these take precedence. Release the			
input buffer.			
6.	RESPONSE	PAGE	6.5.1

Extended Description

Notes	Routine	Label	Ref.

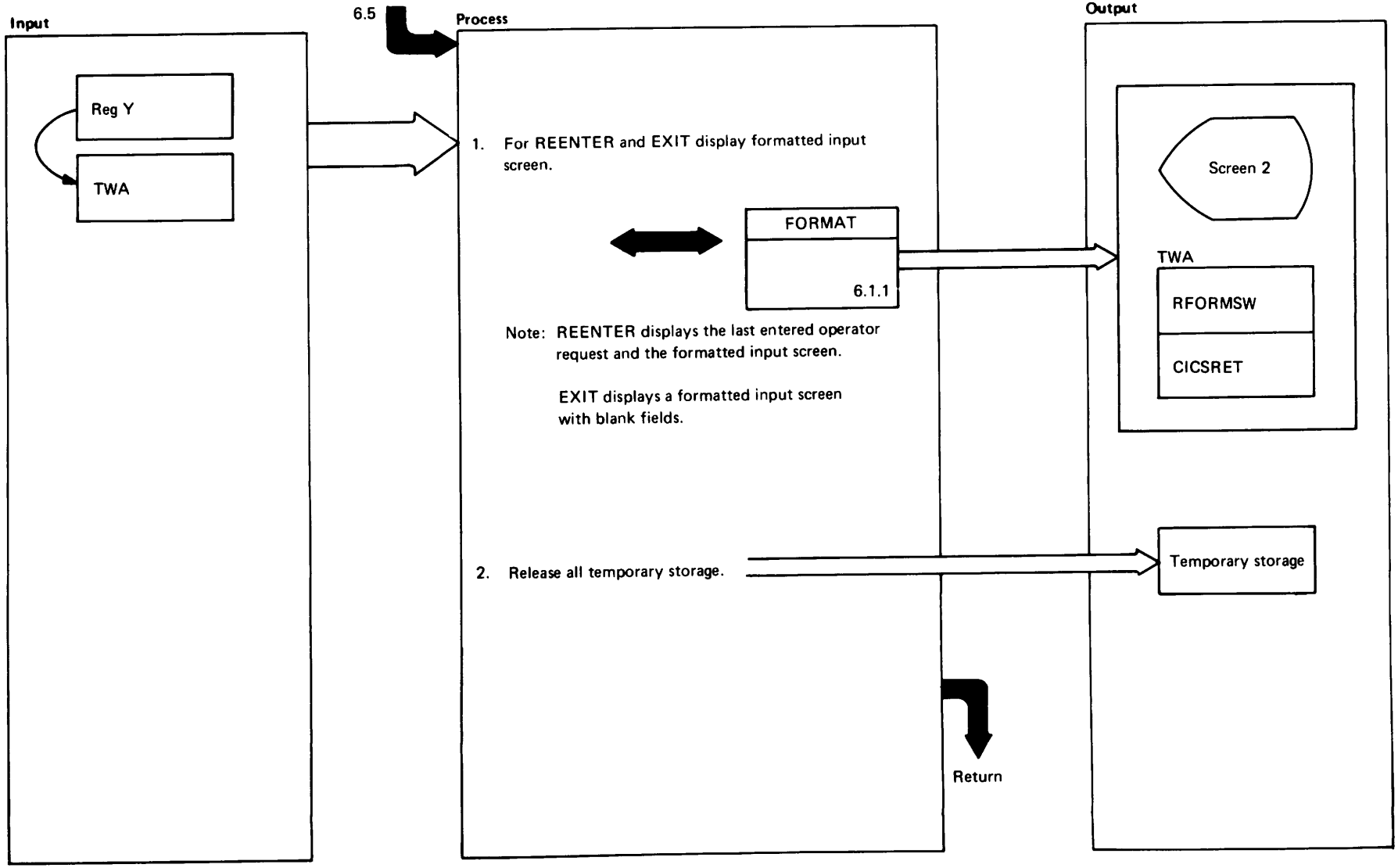
Author: R. Ballow System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.5.1 Name: PAGE Description: Page Options



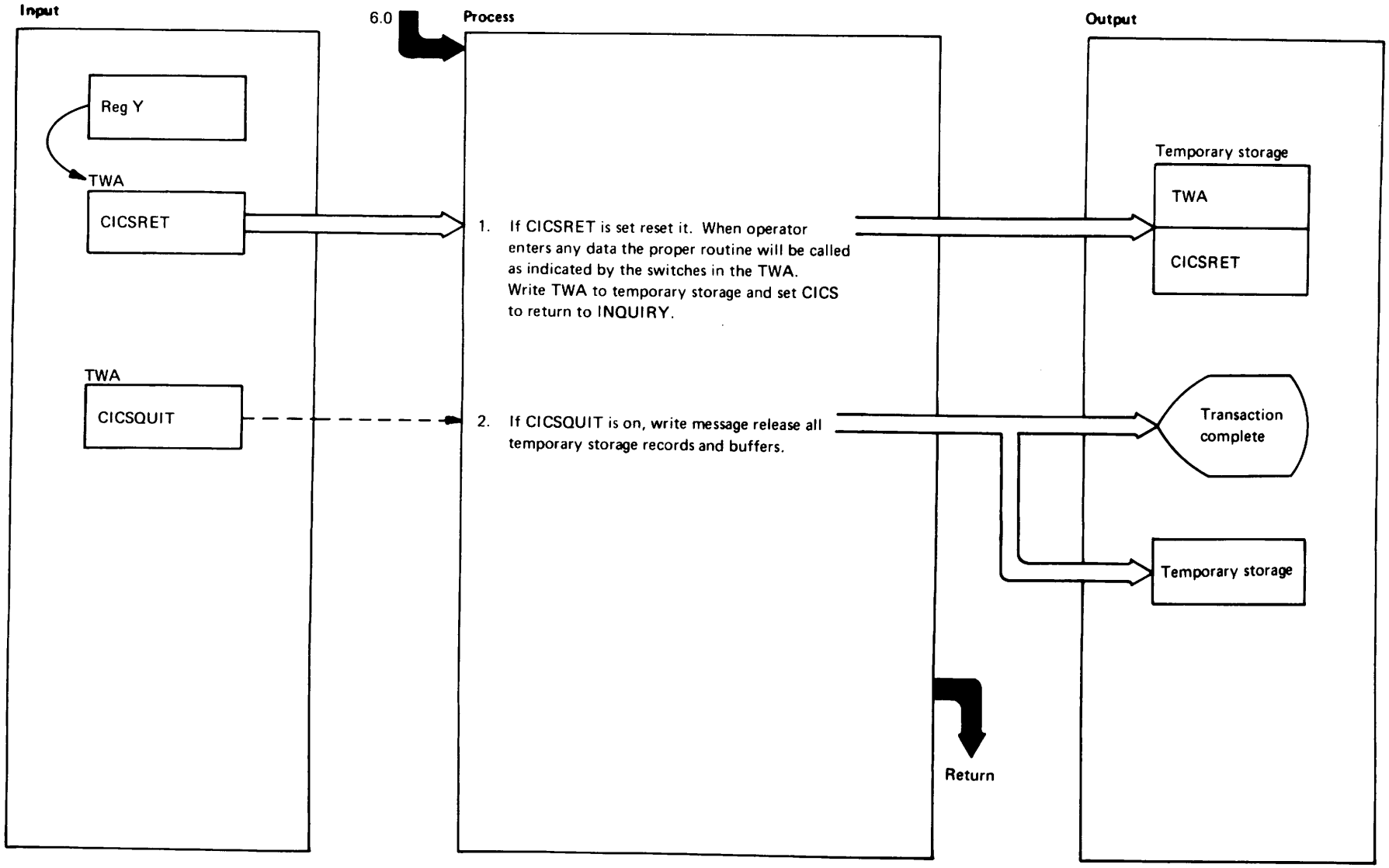
Author: R. Ballow System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.5.2 Name: REENTER Description: Reenter Exit Options



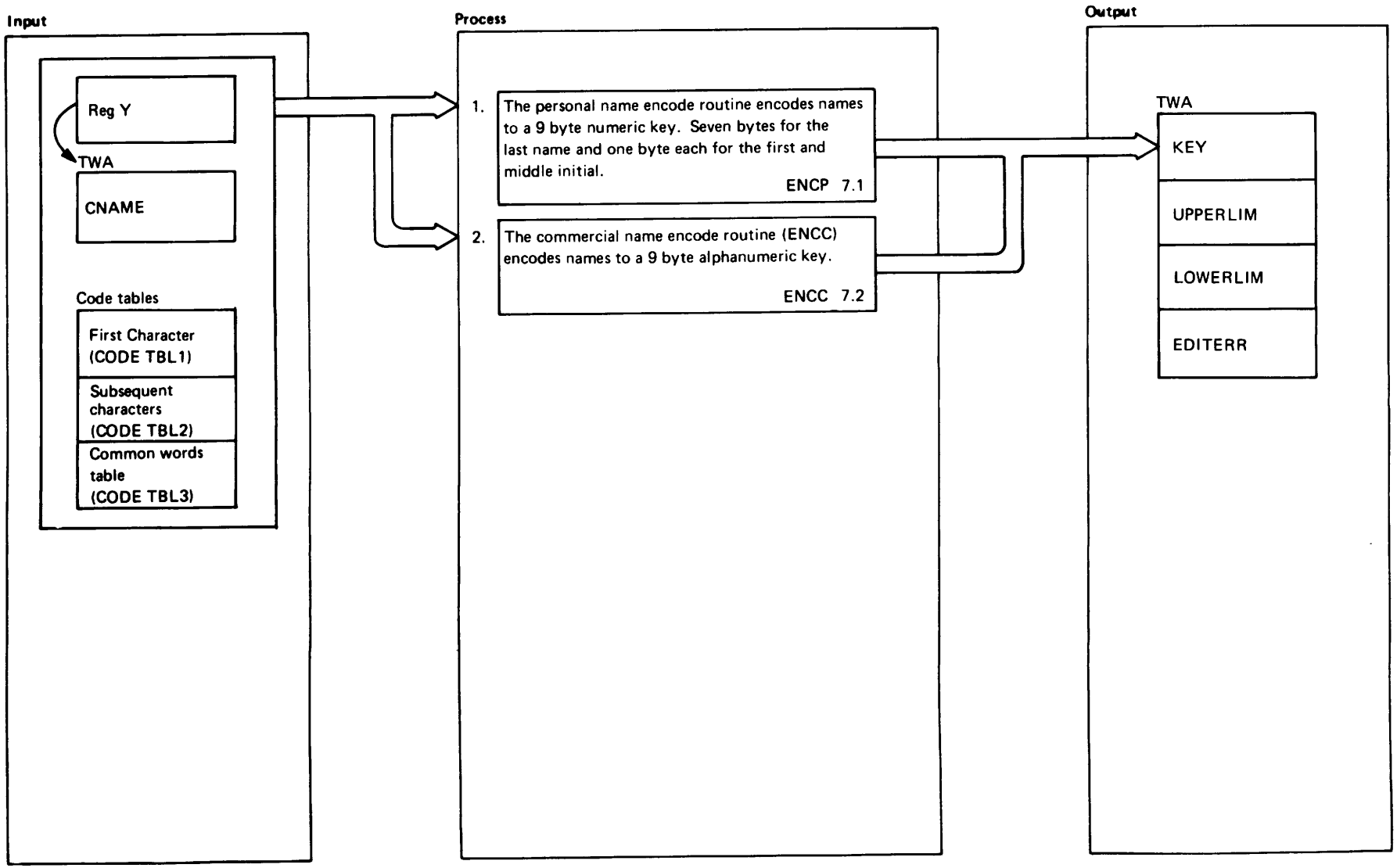
Author: J. Rennaker System/Program: Inquiry Date: 10/25/74 Page: 1 of 1

Diagram ID: 6.6 Name: TERMINAT Description: Terminate a Transaction



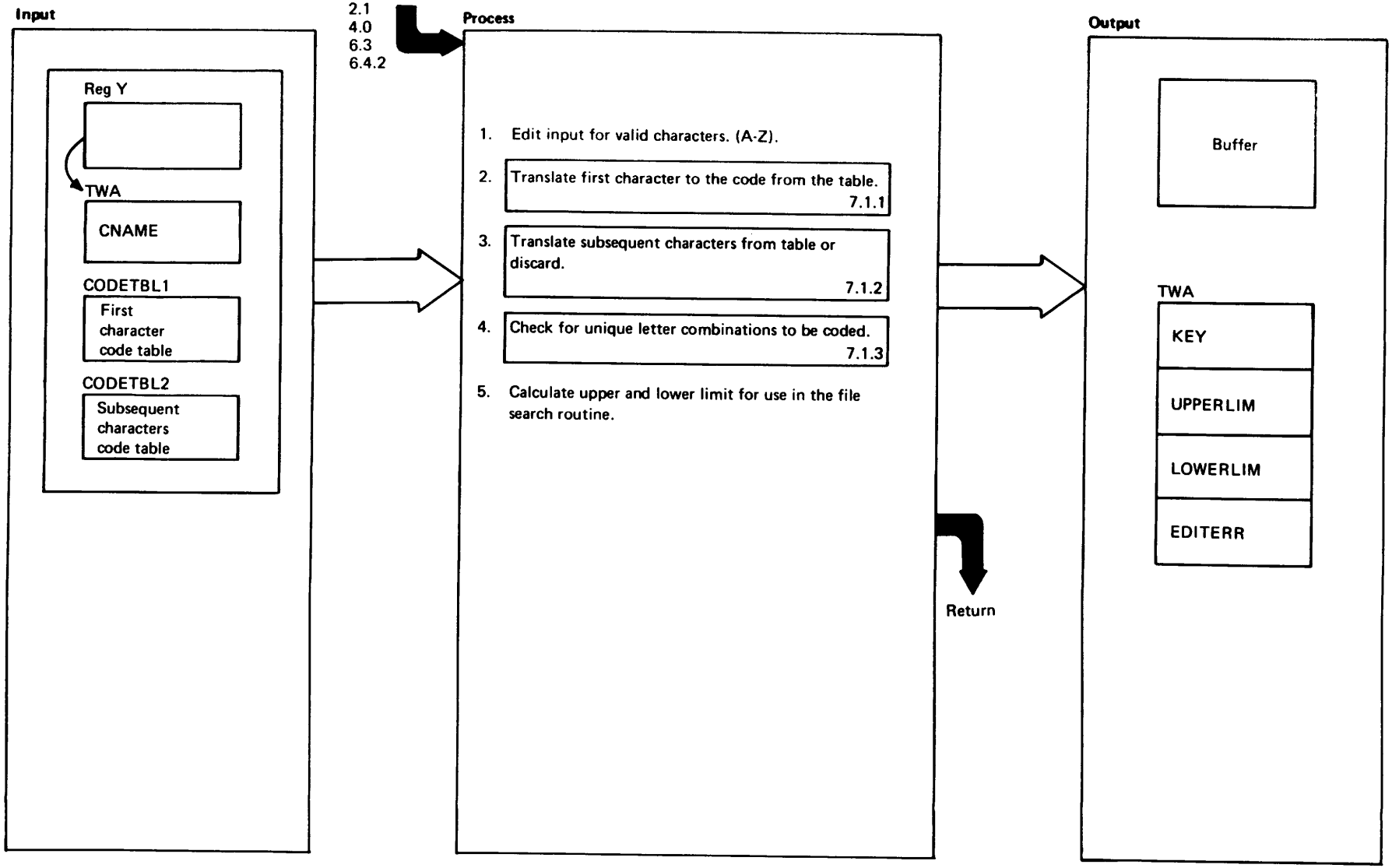
Author: J. Rennaker System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 7.0 Name: Description: Encode Functions



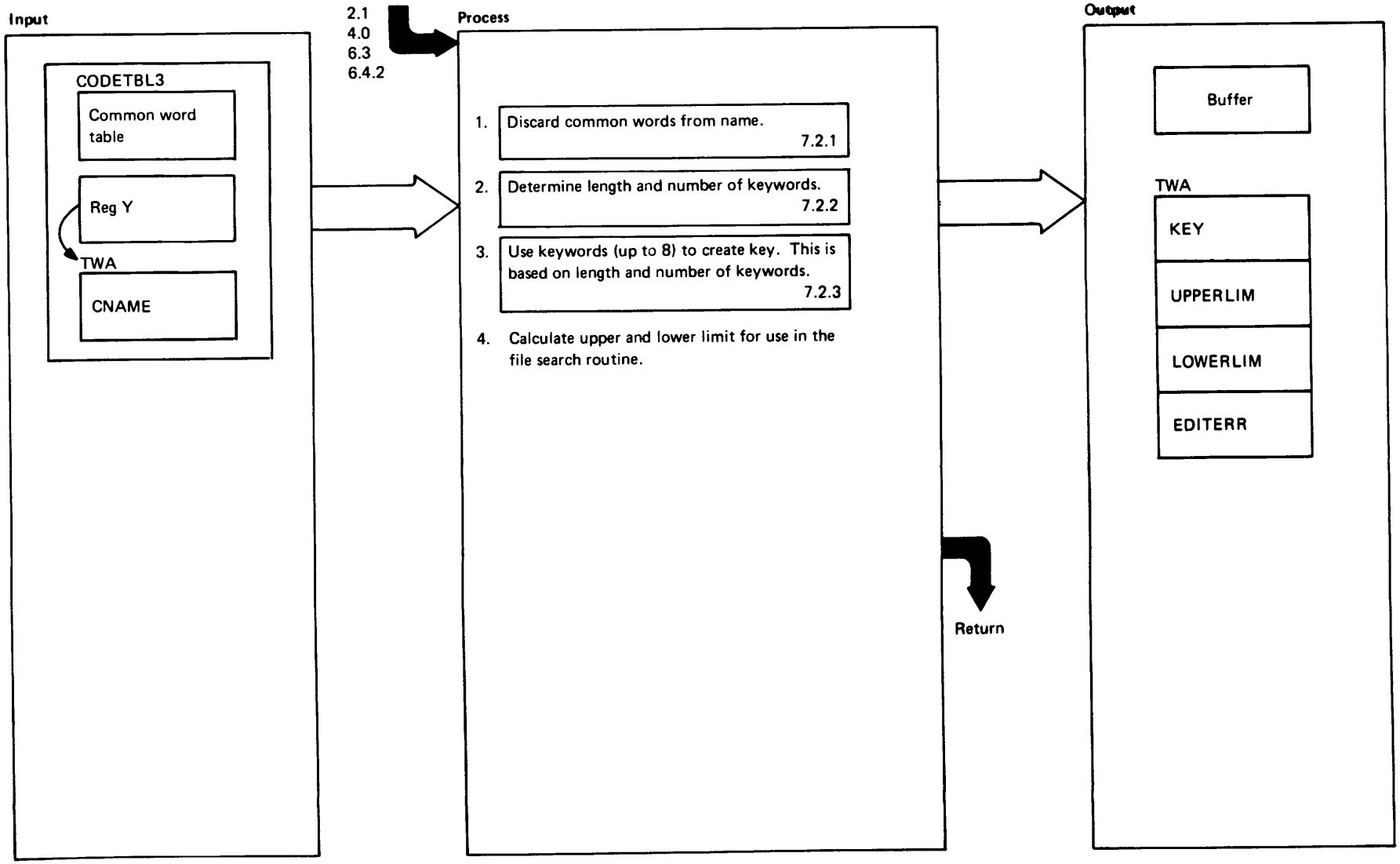
Author: C. Stith **System/Program:** Alpha Search Inquiry System **Date:** 10/25/74 **Page:** 1 of 1

Diagram ID: 7.1 **Name:** ENCP **Description:** Encode Personal Name Overview



Author: C. Stith System/Program: Alpha Search Inquiry System Date: 10/25/74 Page: 1 of 1

Diagram ID: 7.2 Name: ENCC Description: Encode Commercial Name



Bibliography

Baker, F.T. *System Quality through Structured Programming.* Proceedings of FJCC. December 1972.

Bohm, Corrado and Jacopini, Giuseppe (1966) *Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules.* Comm. ACM Vol. 9 No. 5 May 1966.

Dijkstra, Edsger W. *Go To Statement Considered Harmful.* Letter to Editor, Comm. ACM Vol. 11 No. 3 March 1968.

Dijkstra, Edsger W. *The Humble Programmer.* Comm. ACM Vol. 15 No. 10. October 1972.

Stevens, W.P. and Myers, G.J. and Constantine, L.L. *Structured Design.* IBM Systems Journal Vol. 13 No. 2 1974. (Reprints are available under order number G320-5323.)

Improved Programming Technologies—An Overview. IBM Corporation, GC20-1850.

READER'S COMMENT FORM

HIPO – A Design Aid and Documentation Technique

GC20-1851-1

Please comment on the usefulness and readability of this publication, suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM. If you wish a reply, be sure to include your name and address.

COMMENTS

fold

fold

fold

fold

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.
FOLD ON TWO LINES, STAPLE AND MAIL.

Your comments, please . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Fold

Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail

No postage stamp necessary if mailed in the U.S.A.

Postage will be paid by:

International Business Machines Corporation
1133 Westchester Avenue
White Plains, New York 10604

Att: Technical Publications/Systems – Dept. 824

Fold

Fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

HIPO – A Design Aid and Documentation Technique Printed in U.S.A. GC20-1851-1