File No. S360-31
GY28-2030-2

Program Logic

Version 8.1

IBM System/360 Time Sharing System

Linkage Editor

Describes the internal logic of the linkage editor for the IBM System/360 Time Sharing System (TSS/360). (Another publication, IBM System/360 Time Sharing System: Linkage Editor, GC28-2005, explains how the linkage editor is used.) The linkage editor is an optional facility; its use is not required to successfully run programs in TSS/360.

- Explains the structure of an object module and its program module dictionary.
- Describes the relationship between the linkage editor and TSS/360.
- Provides details on the three phases of linkage editor processing - control statement, output, and early-end. (Flowcharts are also provided.)

In TSS/360, the output of a language processor or the linkage editor is an object module; it is input to the dynamic loader. With the linkage editor, a user may permanently link separate but related object modules into one object module, reducing dynamic loader processing time. He may also, without having to reassemble or recompile his program, use the linkage editor to: combine control sections within a module (possibly saving storage and reducing paging activity during execution); change control section attributes; change or delete control section and entry point names; or change external references.

This book is for customer engineers, system engineers, and programmers who need to pinpoint problems, and system programmers involved in altering the linkage editor design.

Before using, be familiar with the contents of:

IBM System/360 Operating System: Principles of Operation, GA22-6821
IBM System/360 Time Sharing System: Concepts and Facilities, GC28-2003

Third Edition (September 1971)

This publication has been revised to incorporate cumulative changes added by Technical Newsletters and to follow new IBM guidelines for program logic manuals. The book has been reorganized and much of it has been rewritten. (The Preface explains the new organization.) Significant changes will be indicated by a vertical line beside the changed text. A glossary has been added.

Inclusion of the DXD and CXD instructions and the Q-type address constant to the TSS/360 assembler (to allow assembler-produced object modules to interface with PL/I-produced modules) requires the use of new fields in the program module dictionary (PMD) and new lines in the PMD listing. These changes to TSS/360 are reflected in the linkage editor in the addition of MD500, a subroutine of the LSTPMD (PMD Listing) routine. This subroutine is described as well as new changes to the PMD format (Appendix A).

This edition is current with Version 8, Modification 1, of the IBM System/360 Time Sharing System (TSS/360), and remains in effect for all subsequent versions or modifications of TSS/360 unless otherwise noted. Significant changes or additions to this publication will be provided in new editions or Technical Newsletters. Before using this publication, refer to the latest edition of IBM System/360 Time Sharing System: Addendum, GC28-2043, which may contain information pertinent to the topics covered in this edition. The Addendum also lists the editions of TSS/360 publications that are applicable and current.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

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

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to the IBM Corporation, System/360 Time Sharing System Programming Publications, Department 643, Neighborhood Road, Kingston, New York 12401.

PREFACE

THE PURPOSE OF THIS BOOK

This book is one of a series of TSS/360 program logic manuals; it describes in general and in detail how the TSS/360 linkage editor works. The book is intended for use by programmers and customer engineers who need detailed information about the linkage editor and system programmers responsible for changing it. Using this book for guidance, even more detailed information about the linkage editor can be found in a printed or microfiche listing of the program. (The linkage editor is contained in the TSS/360 object module CEYTS.)

TO USE THIS BOOK, YOU NEED ...

An understanding of the general principles of System/360 and the main concepts of TSS/360, such as virtual storage. This information is available in:

> IBM System/360 Operating System: Principles of Operation, GA22-6821.
> IBM System/360 Time Sharing System: Concepts and Facilities, GC28-2003.

OTHER BOOKS YOU MAY NEED ...

This book contains a brief summary of how the linkage editor is used. The basic how-to-use-it book, containing more extensive information, is:

> IBM System/360 Time Sharing System: Linkage Editor, GC28-2005.

Input to the linkage editor comes from language processor control (LPC). This book describes the LPC interface; however, a more thorough treatment is provided in:

> IBM System/360 Time Sharing System: Command System Program Logic Manual, GY28-2013.

Output from the linkage editor -- the object module -- becomes input, prior to execution, to the dynamic loader. How the dynamic loader processes the object module is described in:

> IBM System/360 Time Sharing System: Dynamic Loader Program Logic Manual, GY28-2031.

HOW THIS BOOK IS ORGANIZED

The book is divided into these sections:

- "Section 1: Introduction" -- Defines the linkage editor, describes input and output, and explains the general structure of an object module.
- "Section 2: Method of Operation" -- Describes the linkage editor's relationship to the user and TSS/360, and discusses its three phases of processing.
- "Section 3: Program Organization" -- Describes individual routines and subroutines within the linkage editor, and contains flowcharts for the routines.
- "Section 4: Directory" -- Contains a directory to routines, correlating entry point and routine names to flowchart designation.
- "Section 5: Data Areas" -- Describes tables, lists, and work areas used by linkage editor routines; also contains a PSECT organization table.
- "Section 6: Diagnostic Aids" -- Contains suggestions for debugging (for instance, where to take dynamic dumps).
- "Appendix A: The Program Module Dictionary" -- Illustrates and describes the format of the PMD.
- "Appendix B: The Internal Symbol Dictionary" -- Discusses the ISD, and illustrates and describes the format of the composite ISD directory.
- "Appendix C: Diagnostic Messages" -- Contains a numbered list, with error levels, of messages the linkage editor may issue to the user.
- "Appendix D: Glossary" -- Defines special terms used in this book.

CONTENTS

SECTION 1: INTRODUCTION 1
What the Linkage Editor Does 1
How the Linkage Editor is Used 1
Input 2
Output 3
What An Object Module is 3
Structure of an Object Module 3
Program Module Dictionary 4
Text 4
Internal Symbol Dictionary 4
Control Sections 5
Attributes of Control Sections 5
Common Control Sections 5
Mixing Object Modules 5
Managing Program Libraries 5
Comparison Between TSS/360 and OS/360 Linkage Editors 6
Linkage Editor Size Requirements and Limitations 6

SECTION 2: METHOD OF OPERATION 7
Relationship to the System 7
The Three Phases of the Linkage Editor 8
Control Statement Processing 8
Output Processing 9
Early-End Processing 9
The Language Processor Control Interface 9

SECTION 3: PROGRAM ORGANIZATION 10
Control Statement Processor 10
Function Summary 10
APENCX, APENEX, APENIN -- Append RLD (CEYCX, CEYEX, CEYIN) 10
BRING -- Bring PMD, Text, and ISD from Library (CEYBR) 16
COMBINE -- COMBINE Statement Processor (CEYCO) 17
END -- END Statement Processor (CEYEN) 18
ERROR -- Error Message Processor (CEYER) 20
EXTREF -- External Reference Search (CEYXR) 20
GETCSD -- Locate Control Section Dictionary (CEYGC) 21
GETLINE Routine (CFADB) 21
GTCSAD -- Get CSD Table Addresses (CEYGA) 21
INANAL -- Control Statement Input/Analyze Processor (CEYIA1) 22
INCLUDE -- INCLUDE Statement Processor (CEYIC) 23
LINK -- Link Modules (CEYLK) 24
PUTDIAG (CFADC1 in Module CFADC) 26
RENAME -- RENAME Statement Processor (CEYRN) 26
SCAN -- Scan (CEYSC) 27
TRAITS -- TRAITS Statement Processor (CEYTR) 28
Output Processor 29
Function Summary 29
OUTPUT -- Output Processor (CEYOP) 31
Early-End Processor 34
Function Summary 34
EARLY END -- Early-End Processor (CEYEE1) 34

FLOWCHARTS 35

SECTION 4: DIRECTORY 68
Linkage Editor Routine Directory 68

SECTION 5: DATA AREAS 69
Common Areas (Storage Areas, Tables, and Lists) 69
Common Internal Storage Areas 69
Work Areas 69
TEMP Storage 69

Common Internal Tables and Lists 69
Exclusion Table (EXCLUD) 69
External Name List (NAMES) 69
Hash Table (HASHTB) 69
Rename/Combine Table (RCTBL) 70
Stack Table (STACK) 70
Reference to Tables and Lists 71
PSECT Organization 71

SECTION 6: DIAGNOSTIC AIDS 73
General Debugging Aids 73

APPENDIX A: THE PROGRAM MODULE DICTIONARY 75
PMD HEADING 75
Control Section Dictionary (CSD) 77
CSD Heading 78
Definition Table 79
Reference Table 80
Relocation Dictionary (RLD) 81
Modifier Pointer 81
Modifier 81
RLD for Complex Definitions 82
RLD for Text External Reference 82
RLD for Text Internal Reference 82
Virtual Memory Page Table (VMPT) 82

APPENDIX B: THE INTERNAL SYMBOL DICTIONARY 83
The Composite ISD 83
The Composite ISD Directory 84
Directory Heading 84
Entries for Each Input Module 84
Composite ISDs as Input 84

APPENDIX C: DIAGNOSTIC MESSAGES 85

APPENDIX D: GLOSSARY 87

INDEX 93

ILLUSTRATIONS

Figure 1. Structure of an object module 4
Figure 2. Overview of linkage editor processing 7
Figure 3. Relationship between linkage editor and LPC 8
Figure 4. Control statement processing flow 9
Figure 5. Overview of the control statement processor 11
Figure 6. Overview of the output processor 29
Figure 7. Overview of the early-end processor 34
Figure 8. Exclusion table (EXCLUD) format 69
Figure 9. External name list format 69
Figure 10. Hash table (HASHTB) format 70
Figure 11. Rename/combine table (RCTBL) format 70
Figure 12. Stack table (STACK) format 70
Figure 13. Program module dictionary format 76
Figure 14. Composite ISD directory format 83

Table 1. Controlling the linkage editor 2
Table 2. Control statement processing hierarchical table 12
Table 3. Output processing hierarchical table 30
Table 4. Early-End processing hierarchical table 34
Table 5. Linkage editor routine directory 68
Table 6. Data references by routines 71
Table 7. PSECT organization 72
Table 8. Register usage 73
Table 9. Size limitations and requirements 74

Chart AA. INANAL, INCLUDE, COMBINE, RENAME/DELNAME, TRAITS, and END statement processors 36
Chart AD. COMSUB - Combine Control Section subroutine 45
Chart AH. CLEANUP - Cleanup Final Module subroutine 46
Chart AI. COLLECT - Collect Common Requirements subroutine and FIXISD - Fix ISD subroutine 47
Chart AJ. LINK - Link Modules subroutine 48
Chart AK. UPISD - Update ISD subroutine 53
Chart AL. APENDF - Append Definition Table subroutine 54
Chart AM. APENCX, APENEX, and APENIN 55
Chart AN. BRING - Bring PMD, Text, and ISD from Library 56
Chart AO. ERROR - Error Message Processor 58
Chart AP. EXTREF - External Reference Search and GETCSD - Locate Control Section Dictionary 59
Chart AQ. GTCSAD - Get CSD Table Addresses and SCAN - Scan subroutine 60
Chart BA. OUTPUT - Output routine 61
Chart BB. LSTPMD - Program Module Dictionary Listing subroutine and MD240 subroutine 62
Chart BC. MD300 and MD350 subroutines 64
Chart BD. MD450 and MD500 subroutines 65
Chart BE. MD600 subroutine 66
Chart CA. EARLY END - Early-End routine 67

The linkage editor is a TSS/360 program called into operation when a user issues an LNK command. The LNK command causes the linkage editor, contained in a single object module named CEYTS, to be loaded into the user's virtual storage.

While the linkage editor can be considered a service program, it has some characteristics of a language processor. As a result of linkage editor processing, a source data set is created consisting of linkage editor control statements, and, as with language processing, the primary output is an object module.

WHAT THE LINKAGE EDITOR DOES

The linkage editor has two distinct functions:

1. It links two or more existing object modules into one new object module.
2. It edits, that is, changes, control information in an object module, obviating the need to reassemble or recompile the source program. The linkage editor can change or delete control section names or entry point names, change external references, change control section attributes, or cause control sections within a module to be combined.

Besides these two primary functions, the linkage editor can also be used to recreate an existing object module in another program library.

The linkage editor, on completion of successful processing, has produced a new object module; the object module or modules used as input still exist, to be retained or erased as the programmer desires.

HOW THE LINKAGE EDITOR IS USED

A TSS/360 user may wish to link two or more related object modules into one module. He may have written a large program and, for convenience, divided it into parts that he assembled or compiled separately. These separate object modules contain control information (provided as the result of the user's language statements) that relates them to each other. When the user calls one object module to be executed, the TSS/360 dynamic loader will load not only that module but all others that it references. If the linkage editor is used to permanently join these object modules into one object module, dynamic loader processing time will be saved. The TSS/360 user must decide whether the loading time saved on subsequent runs is worth the one-time investment in linkage editor processing.

The user may wish to combine control sections within a module. Since in TSS/360 each control section, regardless of length, begins on a page boundary, combining short control sections (those much less than 4096 bytes) can mean better utilization of external storage and less paging activity during program execution. With the linkage editor, the user can combine control sections without having to rewrite and then reassemble or recompile his program.

The user may also use the linkage editor, without having to reassemble or recompile, to change or delete control section names and entry point names, to change external references, or to change control section attributes.

The user may find it convenient to use the linkage editor for several purposes, and he can do so within a single linkage editor run.

The linkage editor is invoked in either a conversational or nonconversational task with the LNK command. The user follows immediately with a sequence of linkage editor control statements which must terminate with an END statement. (These control statements are described in Table 1.) The linkage editor provides diagnostic messages as the statements are entered; the terminal user may make immediate corrections. Following the END statement, the linkage editor provides a list of unresolved external references (presumably these references will either be resolved in subsequent linkage editor processing or be left for resolution by the dynamic loader).

Table 1. Controlling the linkage editor

TSS/360 user issues LNK command specifying:	Name of Output Object Module	Whether Control Statements Prestored	Program Library in Which to Place Output	Version ID	Whether to Produce ISD	Whether to Produce PMD Listing	Where to Put PMD Listing	Line Number, Increment
Default:	Must specify	Not prestored	Latest library created	System-provided	Produce ISD	Don't produce PMD listing	Conv: Store Name: Print	100,100

Then issues any of the control statements below:	Also specifying:	Placing the control statement:*	As a result, the linkage editor:
COMBINE	Names of control sections to be combined.	Ahead of a form-1 INCLUDE which will specify the object module containing the control sections.	At the next form-1 INCLUDE, combines all control sections named into the first control section named in the COMBINE statement.
RENAME	Old and new external references, entry point names, control section names.	Ahead of a form-1 INCLUDE which will specify object module containing names.	Changes or deletes specified names in first object module named in next form-1 INCLUDE.
TRAITS	Name of control section and new attributes.	Ahead of a form-1 INCLUDE which will specify object containing control section.	Changes attributes of specified control section in first module named in next form-1 INCLUDE.
INCLUDE (form-1)	The ddname of a program library and object module in it.	Before END.	Causes the specified object module to be included in the output object module and any stacked COMBINE, RENAME, and TRAITS statements to be processed.
INCLUDE (form-2)	The ddname of a program library to be searched.	After at least one previous form-1 INCLUDE.	Searches library, includes in output module all modules referenced by previously included modules.
INCLUDE (form-3)	The ddname of a program library followed by a minus sign; names of external references not to be resolved.	After at least one previous form-1 INCLUDE.	Searches library, includes in output module all modules referenced by previously included modules, except those containing the external references specified.
END	Blank.	Last.	Searches all libraries on the current program library list for modules satisfying unresolved references, includes modules in output module. Furnishes message listing unresolved references and those resolvable in SYSLIB.

*Placement Rules:

1. A COMBINE, TRAITS, or RENAME statement may precede any other COMBINE, TRAITS, or RENAME statement, a form-1 INCLUDE, or END (it will not be processed if it is stacked when the END statement occurs). It cannot be immediately followed by a form-2 or -3 INCLUDE.
2. A form-1 INCLUDE can be placed before, between, or after any other statement except END (it may not be placed after END).
3. A form-2 or -3 INCLUDE must have been immediately preceded by a form-1, -2, or -3 INCLUDE, and at least one previous form-1 INCLUDE must have occurred.
4. The END statement must be last.

INPUT

Input to the linkage editor consists of:

1. LNK command parameters.
2. Control statements.
3. One or more object modules.

Following an LNK command, the user provides control statements specifying actions for the linkage editor to take. A summary of LNK command parameters and linkage editor control statements is provided in Table 1. More detailed information on the command and control statements is in IBM System/360 Time Sharing System: Linkage Editor, GC28-2005.

The names of one or more input object modules are specified by the user as operands in his control statements.

OUTPUT

Output from the linkage editor consists of:

1. A new object module.
2. A listing of the program module dictionary (PMD), if requested.
3. An external name list.

The object module is placed in the program library (virtual partitioned data set) named by the user in his LNK command. It must be a different library from those containing the input modules.

The PMD listing shows the contents of the program module dictionary (the control information) of the new object module. The format of this listing is illustrated in Appendix A of IBM System/360 Time Sharing System: Linkage Editor. (The format of the PMD itself is shown in Appendix A of this book.)

The PMD listing must be requested in the LNK command. The listing is either stored as a list data set for future access or printed immediately and not retained in storage. Unless the user specifies the opposite, the PMD listing for a terminal (conversational) user is written as a data set on external storage; the PMD listing for a batch (nonconversational) user is printed. If a terminal user specifies LISTDS=N, the listing is typed out at his terminal (and is not stored as a data set).

In addition to the output object module and optional PMD listing, the linkage editor produces an external name list. The user does not see the external name list (which contains a list of external definitions in the output module); it is passed to the system for use in storing the module so that it may later be accessed by reference to any of the external names.

In addition to the linkage editor output, a source data set containing the linkage editor control statements is created by TSS/360's language processor control as a result of linkage editor processing. The user thus has a stored set of control statements which he may use in or modify for later linkage editor processing.

WHAT AN OBJECT MODULE IS

In TSS/360, the primary output of a language processor (assembler, FORTRAN compiler, or PL/I compiler) or the linkage editor is an object module. (Other TSS/360 publications may use the terms "program module" and "object program module" for "object module.") The object module contains the user's program (instructions, data constants, and reserved areas) plus information that the system requires to inspect the program or set it up for execution. The user may think of this as his program; the system sees the object module.

After the object module has been produced by language processing or linkage editing, the module is stored (until erased) as a named member of a program library (a virtual partitioned data set containing object modules as members). The user runs his program by issuing the CALL command or by using a direct call (that is, by simply using the object module name as a command); the system responds by locating the object module, loading it (and any other modules to which the called module refers) into the user's virtual storage, and executing those pages of the object module(s) that are required.

STRUCTURE OF AN OBJECT MODULE

An object module is divided into:

1. A program module dictionary (PMD).
2. Text (the program itself, consisting of machine-coded instructions, data constants, and reserved data areas).
3. An internal symbol dictionary (ISD).

Figure 1 illustrates this structure.

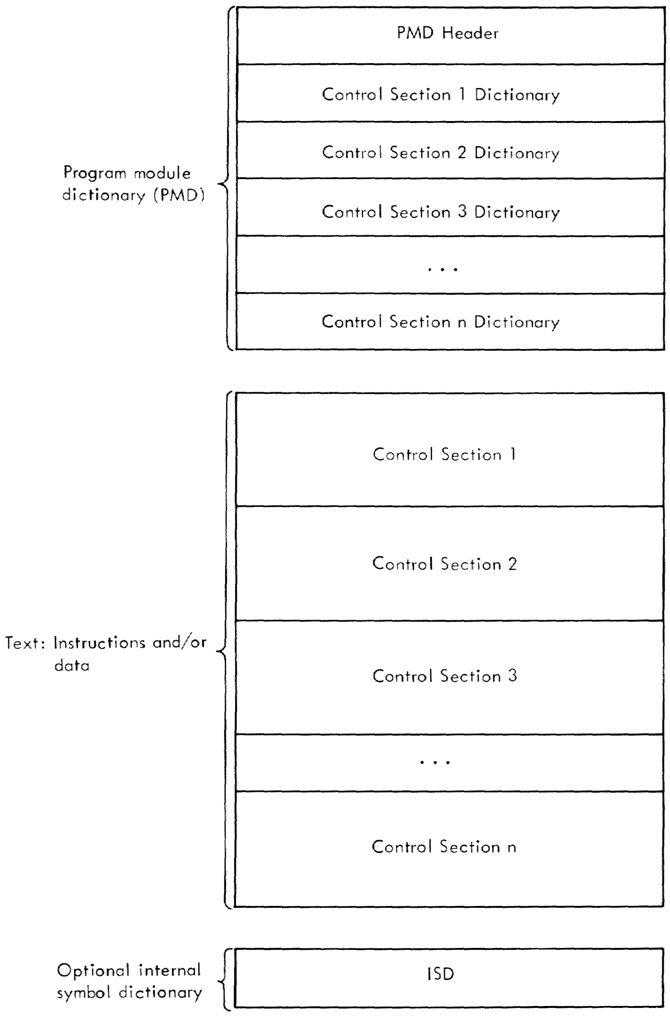

PMD Header
Control Section 1 Dictionary
Control Section 2 Dictionary
Control Section 3 Dictionary
. . .
Control Section n Dictionary
Program module dictionary (PMD)
Control Section 1
Control Section 2
Control Section 3
. . .
Control Section n
Text: Instructions and/or data
ISD
Optional internal symbol dictionary

Figure 1. Structure of an object module

Program Module Dictionary

A program module dictionary consists of:

1. A heading containing the standard entry point to the module, version identifier, and other information common to the entire module. The heading begins on a page boundary.

2. One control section dictionary (CSD) for each control section in the module. Each CSD contains information regarding the CSECT version, external symbol definitions and references, relocation pointers, and information relating text pages in the control section to virtual storage pages.

When linkage editing input modules, the program module dictionary of each module is referenced to produce a single program module dictionary for the output module.

A picture of the program module dictionary and detailed explanation of each field in the PMD is contained in Appendix A.

Text

The text contains the program itself; the instructions, data constants, and reserved but non-initialized data areas. A control section is composed of one or more pages (units of 4096 contiguous bytes); these pages may be text pages or empty pages (pages that simply reserve 4096 bytes as the result of some DS or ORG assembler instruction). Text pages actually exist on some external storage medium; empty pages exist only within virtual storage allocated to the control section. Information in the CSD relates text pages to the control section's total virtual storage.

The text is not changed during linkage editor processing.

Internal Symbol Dictionary

The internal symbol dictionary (ISD) contains the location, length, and type of all internal symbols. The ISD enables the user to debug his program using TSS/360's program control system (PCS). The ISD begins on a page boundary.

The assembler, FORTRAN, or linkage editor user will automatically receive an object module containing an ISD, unless he explicitly states otherwise in his language command. If no ISD is present, internal symbols in the module will not be accessible to the user with PCS commands. (The PL/I user does not have an option; the PL/I compiler does not produce an ISD.)

The linkage editor simply includes as part of its output module each ISD present in input modules; it does not develop its own list of internal symbols. This linkage editor-generated ISD is called a composite ISD, and consists of:

1. A directory pointing to each included ISD in the output module.

2. A chain of the included ISDs.

If one of several input modules does not have an ISD, the composite ISD will contain all other input ISDs. The user will simply not be able to access internal symbols in the part of the output module for which there was no input ISD.

The composite ISD can contain both compiler- and assembler-produced ISDs as well as composite ISDs produced by previous passes through the linkage editor.

Appendix B illustrates and describes the format of the directory portion of the composite ISD. The ISDs produced by the assembler and the FORTRAN compiler differ slightly and are shown, respectively, in IBM System/360 Time Sharing System: Assembler Program Logic Manual, GY28-2021, and FORTRAN Compiler Program Logic Manual, GY28-2019.

Unlike the assembler, the linkage editor does not produce an ISD listing.

CONTROL SECTIONS

Programs are divided into control sections, the basic logical programming unit. It is by the use of a dictionary associated with a control section that programmers can make changes to the object module without recompiling. Each control section in the module is assigned attributes which can be overridden with the TRAITS control statement.

ATTRIBUTES OF CONTROL SECTIONS

A control section may have one or more of the following attributes:

- Variable length -- a number of pages is allocated in addition to the declared length (a control section without this attribute is considered fixed-length).
- Read-only -- data may not be stored into the control section.
- Common -- common to all object modules in which it is declared.
- Privileged -- eligible to be classed as a privileged system program when placed into the system library (SYSLIB).
- System -- part of a system (as opposed to user) object module.
- Public -- assigned storage so it is available to other tasks.
- Prototype (PSECT) -- storage containing the private copy of modifiable storage made available to each task for public routines.

COMMON CONTROL SECTIONS

Common control sections are created by the assembler and FORTRAN compiler to allow separate programs to access the same storage area. The sizes of blank (unnamed) common areas are examined by the linkage editor. If more than one blank common area is found in the input, the linkage editor reserves an area equivalent to the largest blank common area encountered and ignores the rest; the data content of the first common control section encountered is retained. Named common control sections are treated similarly to noncommon control sections, except that common control section names are not placed in the external name list and hence do not become part of the partitioned organization directory (POD) of a library.

MIXING OBJECT MODULES

Object modules produced by different language processors can be linked into one object module by the linkage editor. This is because the PMDs produced by the assembler, the FORTRAN compiler, the PL/I compiler, and the linkage editor are identical, except for flags indicating which processor produced them. The user must ensure, however, that he has followed linkage conventions and provided compatible data forms. He must also be familiar with the control section names and functions of PL/I and FORTRAN produced modules.

MANAGING PROGRAM LIBRARIES

Object modules in TSS/360 are stored in libraries. A library is a partitioned data set whose members are object modules. TSS/360 users have access to four kinds of libraries:

- System library (with the data set name SYSLIB) -- This library contains programs accessible to all users, both TSS/360's programs (including the linkage editor) and the installation's own standard subroutines and functions.
- User library (with the data set name USERLIB) -- This is the private library assigned to each user when he is joined to the system. If the user does not specify a library to receive his linkage editor output, the output object module is automatically placed in his user library.
- Job library (with a user-defined data set name) -- This is a library that the user defines during a task by using the JOBLIB option of the DDEF command. Commonly, job libraries are used to hold object modules temporarily; when debugged, the object module, especially if it is to be run often, may be transferred to the user library. A user may designate the name of a job library in which the output of linkage editor processing is to be placed.

- Other library (with a user-defined data set name) -- A user may also place the output of the linkage editor in a library that is not a job library, by using the DDEF command without specifying the JOBLIB option. Such a library may be designated as the source of input modules or as the destination for the output module. The linkage editor does not search this kind of library for modules containing external definitions that would resolve unresolved references in the module being developed by the linkage editor. Modules in this kind of library cannot be loaded or called for execution. This kind of library is not on the user's program library list.

A more detailed discussion of program libraries as they relate to the linkage editor is contained in IBM System/360 Time Sharing System: Linkage Editor.

Prior to issuing the LNK command, the user must have defined any libraries from which he wishes to obtain object modules or which he wishes to be searched to resolve external references. He must also define the library in which the output is to be placed. (He does not, however, have to define USERLIB.) Since the output must go in a library different from any that contained input modules, at least one DDEF command will be required prior to linkage editing.

COMPARISON BETWEEN TSS/360 AND OS/360 LINKAGE EDITORS

Although similar in function, TSS/360 and OS/360 (IBM System/360 Operating System) linkage editors are different programs. Control statements for one linkage editor are incompatible with the other. In OS/360, a distinction is made between the output of a language processor (an "object module") and the output of the linkage editor (a "load module"). TSS/360 does not make this distinction; the output of both kinds of processor is called an "object module." OS/360 object modules containing external references (references to other modules) must be linked into a load module prior to being input to the loader. In TSS/360, the linkage editor is always an optional facility; any object module may become input to the dynamic loader.

LINKAGE EDITOR SIZE REQUIREMENTS AND LIMITATIONS

Table 9 in "Section 6: Diagnostic Aids" lists size requirements and limitations on the number and size of input modules.

RELATIONSHIP TO THE SYSTEM

When the user issues an LNK command, TSS/360 passes control to:

The command analyzer and executor (CA&E), which passes control to:

Language processor control (LPC), which passes control to:

The linkage editor.

During linkage editor processing, the linkage editor calls LPC:

- To get each control statement.
- When a diagnostic message is to be issued.
- When a phase of processing is completed.
- If, for some reason, it cannot continue processing.

The linkage editor also uses system routines called as the result of OPEN, CLOSE, FIND, GETMAIN, and FREEMAIN macro instructions, and directly calls the dynamic loader for library searches.

This general relationship between the linkage editor and the system is shown in Figure 2. The relationship between the linkage editor and LPC is shown in Figure 3.

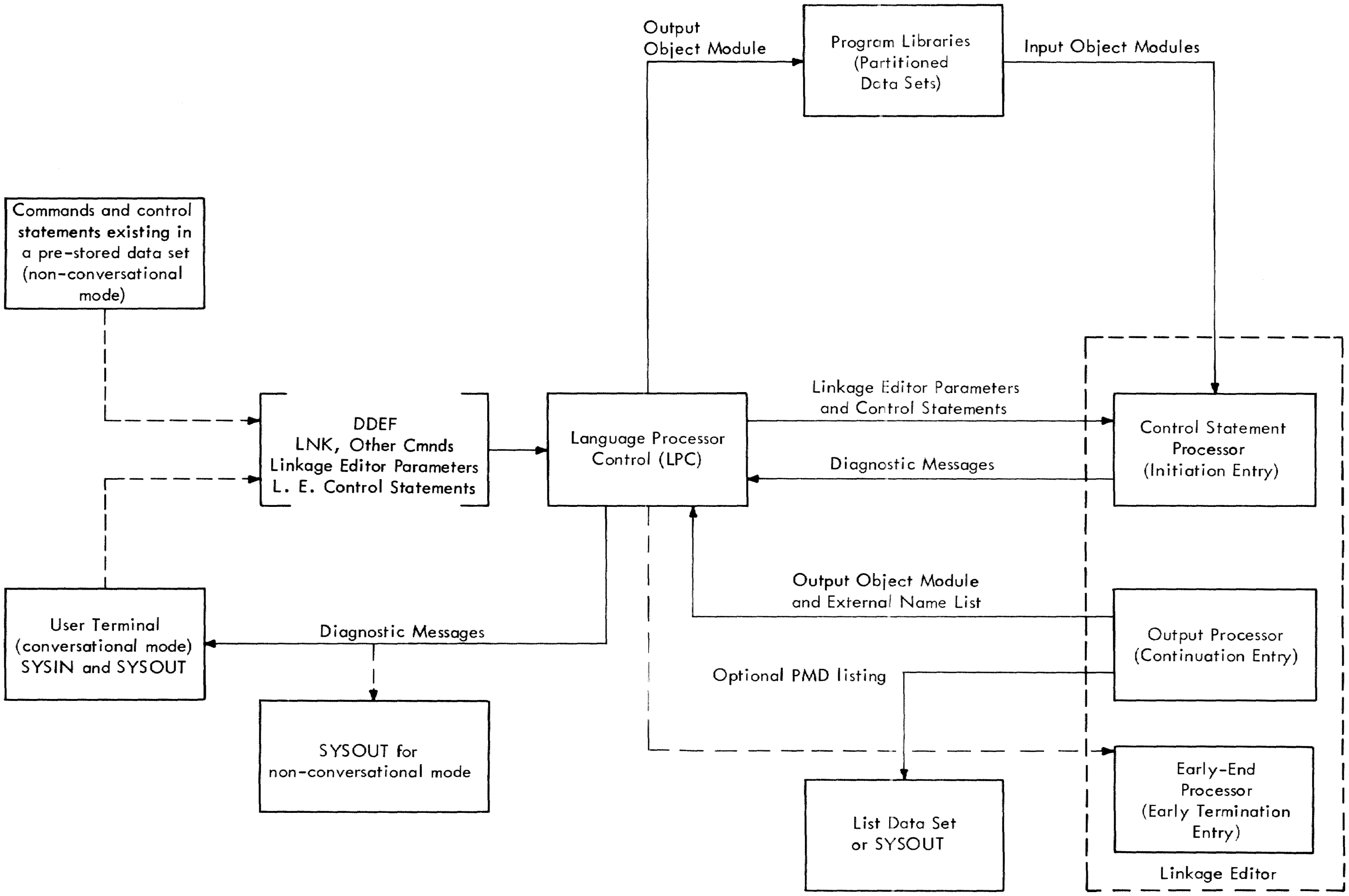

Output Object Module
Program Libraries (Partitioned Data Sets)
Input Object Modules
Commands and control statements existing in a pre-stored data set (non-conversational mode)
DDEF
LNK, Other Cmnds
Linkage Editor Parameters
L. E. Control Statements
Language Processor Control (LPC)
Linkage Editor Parameters and Control Statements
Diagnostic Messages
Control Statement Processor (Initiation Entry)
Output Object Module and External Name List
Output Processor (Continuation Entry)
User Terminal (conversational mode) SYSIN and SYSOUT
Diagnostic Messages
Optional PMD listing
SYSOUT for non-conversational mode
Early-End Processor (Early Termination Entry)
List Data Set or SYSOUT
Linkage Editor

Figure 2. Overview of linkage editor processing

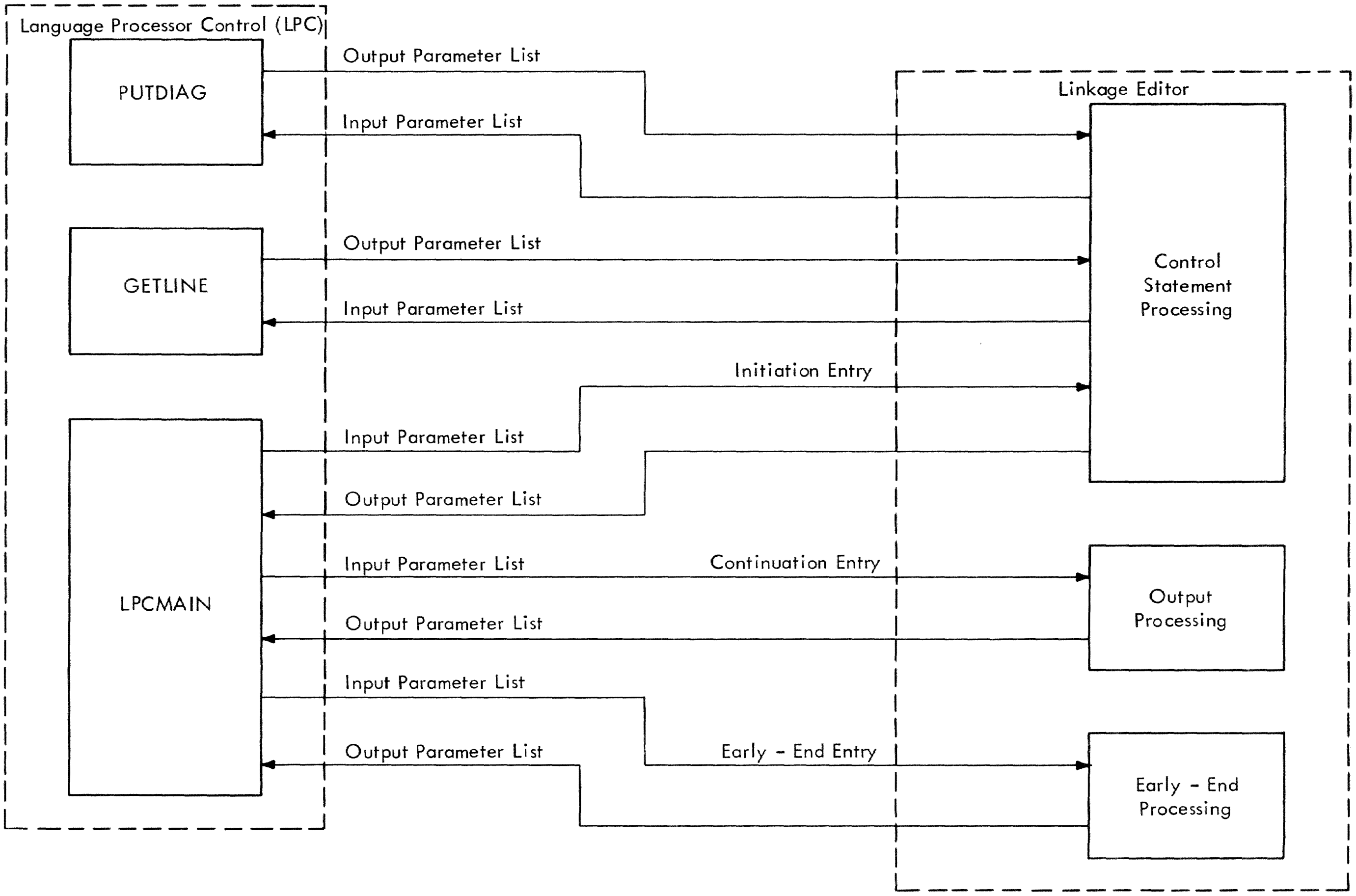


Figure 3. Relationship between linkage editor and LPC

THE THREE PHASES OF THE LINKAGE EDITOR

The linkage editor is divided into the three main phases or routines:

- Control Statement Processor (entry point CEYIA1)
- Output Processor (entry point CEYOP1)
- Early-End Processor (entry point CEYEE1)

When a user requests the services of the linkage editor, LPC calls the linkage editor at the initiation (Control Statement Processor) entry point. Control is returned to LPC after all control statement processing has been performed (an END statement has been received). LPC then calls the linkage editor at the continuation (Output Processor) entry point to deliver the final module, the external name list, and, if the user has requested it, a PMD listing. After this, control is returned to LPC and linkage processing is complete. If it is necessary to prematurely terminate processing, LPC enters the linkage editor at its early-end (Early-End Processor) entry point for necessary clean-up.

CONTROL STATEMENT PROCESSING

This phase of linkage editor processing is illustrated in Figure 4 and is summarized below.

During initialization, work areas are obtained, and switches and tables are initialized. Control statements are then requested from LPC, one at a time. Each statement is checked for accuracy and correct sequence in relation to other control statements. (See the Placement Rules footnote at the bottom of Table 1 in Section 1.) When a COMBINE, RENAME, or TRAITS statement is received, the statement is stacked until the next form-1 INCLUDE statement is received, at which time the INCLUDE statement processing routine calls individual subroutines to process each stacked COMBINE, RENAME, or TRAITS on a first-in, first-out basis. When an INCLUDE

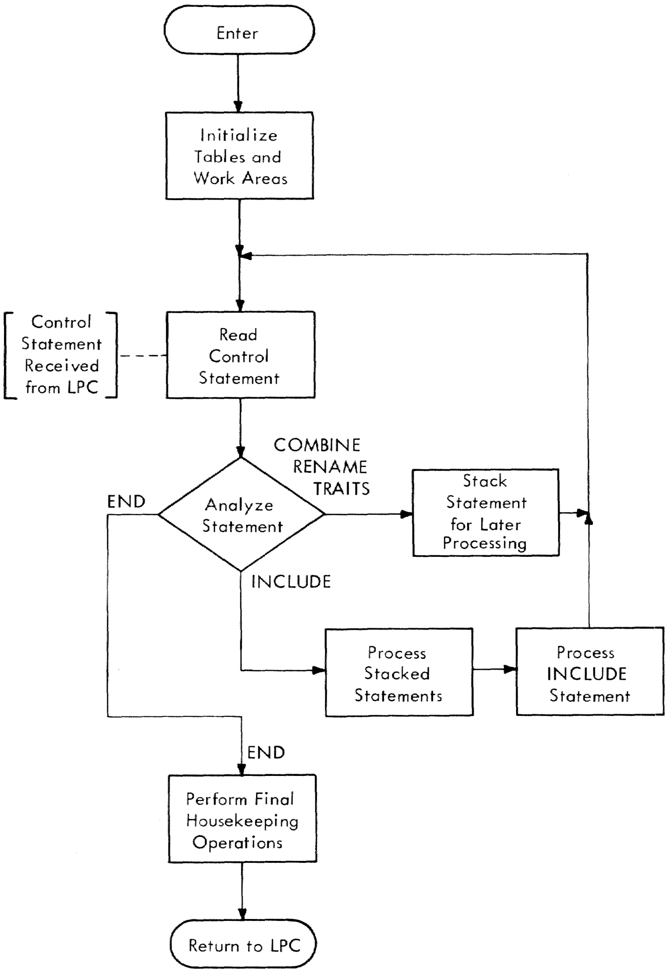

Enter
Initialize Tables and Work Areas
Control Statement Received from LPC
Read Control Statement
COMBINE RENAME TRAITS
END
Analyze Statement
Stack Statement for Later Processing
INCLUDE
Process Stacked Statements
Process INCLUDE Statement
END
Perform Final Housekeeping Operations
Return to LPC

Figure 4. Control statement processing flow

or END control statement is received, the appropriate routine to process that statement is called immediately.

Each time processing of a statement is completed, control returns to the main control statement processing routine, INANAL, which then requests the next statement from LPC.

The ERROR processor routine is used by INANAL and other control statement processor routines to set up diagnostic messages to be sent to the user via LPC.

Control statement processing is terminated when, after processing an END statement, control is returned to LPC.

OUTPUT PROCESSING

After all statements have been processed, LPC calls the linkage editor at the entry point for output processing, which produces the final module, including PMD, text, and ISD (if required). The output processing function also prepares the external name list and PMD listing; on completion, it passes to LPC the location of the output module and the external name list.

EARLY-END PROCESSING

Early-end processing releases storage areas and closes any open libraries. It is entered if the linkage editor is to be terminated before normal completion.

THE LANGUAGE PROCESSOR CONTROL INTERFACE

LPC serves as the system link between the user and the linkage editor; LPC action is not evident to the user. LPC gathers the input parameters for the linkage editor, loads the linkage editor, and passes on the parameters. The linkage editor calls upon LPC to issue diagnostic messages.

LPC consists of three routines:

1. LPCMAIN -- collects input parameters and stores the object module in a program library.

2. GETLINE -- receives the linkage editor control statement source lines, one at a time, from the user and creates a source data set (or takes the source lines one at a time from a pre-stored data set), and passes them on to the linkage editor upon request. When necessary, GETLINE issues diagnostic messages stacked by PUTDIAG and prompts for corrections before getting the next source line.

3. PUTDIAG -- collects and stacks diagnostic messages from the linkage editor.

Detailed descriptions of these routines can be found in Command System Program Logic Manual, GY28-2013.

This section is divided into three parts, corresponding to the three main routines of the linkage editor:

- Control Statement Processor (entry point CEYIA1)
- Output Processor (entry point CEYOP1)
- Early-End Processor (entry point CEYEE1)

Each part contains a function summary, an overview figure, a hierarchical table, and individual routine descriptions. The routine descriptions for the Control Statement Processor are arranged alphabetically; the routine descriptions for the Output Processor are arranged alphabetically within an overall Output Processor routine description. There is only one Early-End routine.

Flowcharts for the routines appear at the end of this section.

Entry point names provided in the figures, hierarchical tables, routine descriptions, and flowcharts enable quick reference, through use of the cross-reference listing, to any desired section of code in the object program listing of the linkage editor (object module CEYTS).

CONTROL STATEMENT PROCESSOR

Function Summary

The routines that constitute the Control Statement Processor do the following:

- Get a line of input from LPC (language processor control).
- Scan it for validity. Call LPC to tell the user if the line is invalid.
- Stack any COMBINE, RENAME, or TRAITS statements until the next form-1 INCLUDE; process them when the form-1 INCLUDE is received.
- Process any form-2 or -3 INCLUDE statements.
- Link specified input modules to the output.
- Call the dynamic loader via the LIBESRCH macro instruction to search for modules to satisfy unresolved external references.
- Perform final linkage and cleaning up when an END statement is received.

On completion of END statement processing, control statement processing is concluded, and control is returned to LPC, which then invokes the linkage editor at its second or continuation entry point for output processing.

Routines shown in blocks in Figure 5 correspond to entries in Table 2. The level number in the blocks (upper right-hand corner) corresponds to the level the routine occupies in Table 2. Routines in LPC and the dynamic loader and macro instructions used to call other system routines are shown under called routines in Table 2, but are not otherwise described.

Descriptions of the individual routines follow Table 2 and are arranged alphabetically by routine name.

APENCX, APENEX, APENIN -- Append RLD (CEYCX, CEYEX, CEYIN)

This routine appends the RLD for complex definitions and references of a CSD in the input PMD to the output PMD. (See Chart AM.)

Entry: Entry points CEYCX, CEYEX, CEYIN; entry parameters:

Registers:
2, 4, 5
pseudo parameters whose values are preserved.

3
location of next available position in task dictionary.

12
common register which contains a pointer to the CSD heading in the task dictionary.

Calling Sequences: INVOKE ACEYCX; INVOKE ACEYEX; INVOKE ACEYIN.

Routines Called: None.

Exit: Normal; exit parameter:

Register 3
pointer to next position in output PMD.

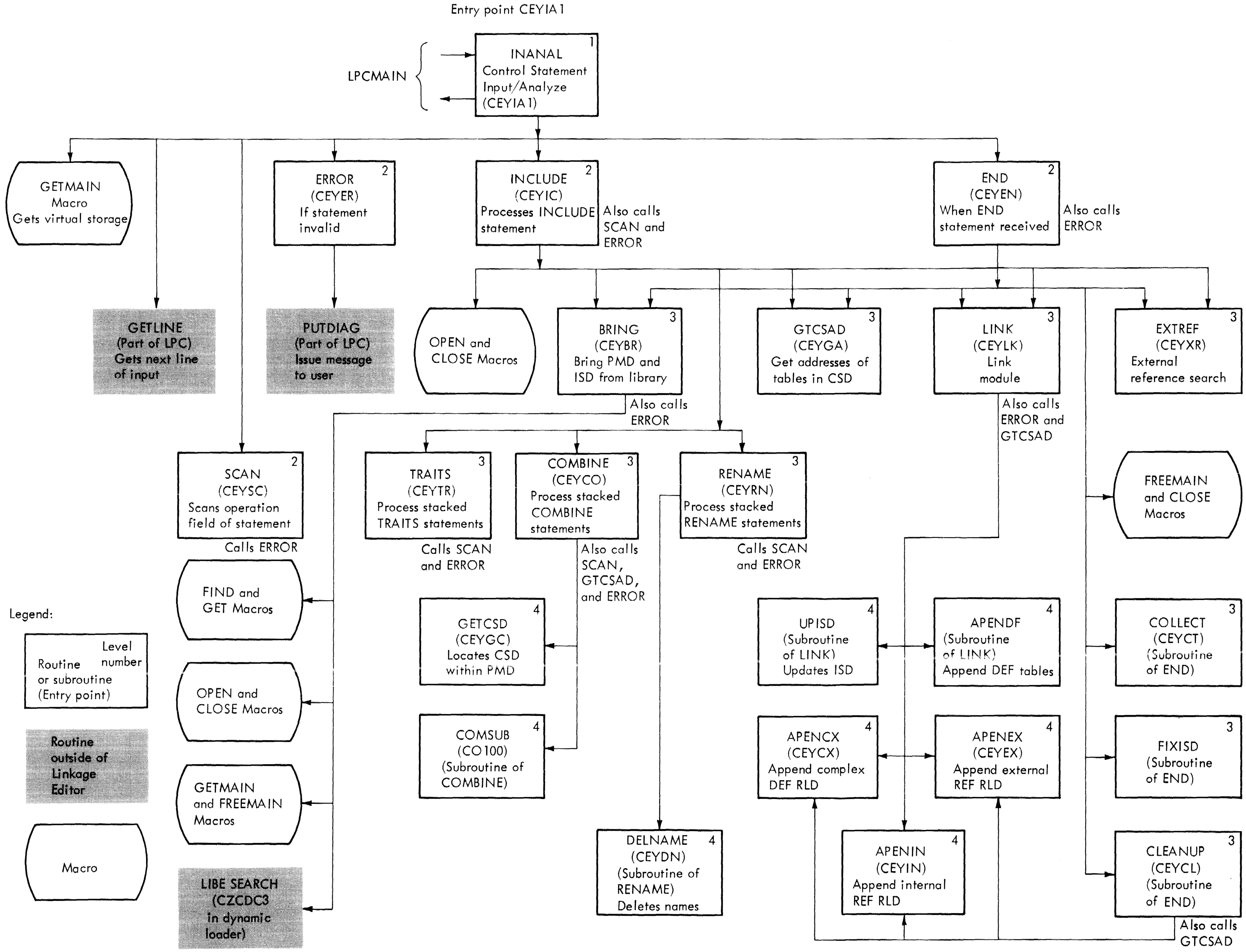

Entry point CEYIA1
LPCMAIN
INANAL 1 Control Statement Input/Analyze (CEYIA1)
GETMAIN Macro Gets virtual storage
ERROR 2 (CEYER) If statement invalid
INCLUDE 2 (CEYIC) Processes INCLUDE statement
Also calls SCAN and ERROR
END 2 (CEYEN) When END statement received
Also calls ERROR
GETLINE (Part of LPC) Gets next line of input
PUTDIAG (Part of LPC) Issue message to user
OPEN and CLOSE Macros
BRING 3 (CEYBR) Bring PMD and ISD from library
Also calls ERROR
GTCSAD 3 (CEYGA) Get addresses of tables in CSD
LINK 3 (CEYLK) Link module
Also calls ERROR and GTCSAD
EXTREF 3 (CEYXR) External reference search
SCAN 2 (CEYSC) Scans operation field of statement
Calls ERROR
TRAITS 3 (CEYTR) Process stacked TRAITS statements
Calls SCAN and ERROR
COMBINE 3 (CEYCO) Process stacked COMBINE statements
Also calls SCAN, GTCSAD, and ERROR
RENAME 3 (CEYRN) Process stacked RENAME statements
Calls SCAN and ERROR
FREEMAIN and CLOSE Macros
Legend:
Routine or subroutine (Entry point) Level number
Routine outside of Linkage Editor
Macro
FIND and GET Macros
OPEN and CLOSE Macros
GETMAIN and FREEMAIN Macros
LIBE SEARCH (CZCDC3 in dynamic loader)
GETCSD 4 (CEYGC) Locates CSD within PMD
COMSUB 4 (CO100) (Subroutine of COMBINE)
DELNAME 4 (CEYDN) (Subroutine of RENAME) Deletes names
UPISD 4 (Subroutine of LINK) Updates ISD
APENDF 4 (Subroutine of LINK) Append DEF tables
COLLECT 3 (CEYCT) (Subroutine of END)
APENCX 4 (CEYCX) Append complex DEF RLD
APENEX 4 (CEYEX) Append external REF RLD
FIXISD 3 (Subroutine of END)
APENIN 4 (CEYIN) Append internal REF RLD
CLEANUP 3 (CEYCL) (Subroutine of END)
Also calls GTCSAD

Figure 5. Overview of the control statement processor

Table 2. Control statement processing hierarchical table (part 1 of 4)

Routine	Purpose	Called Routines	Calling Conditions
Routine: Control Statement Processor -- Level: 1			
INANAL - Control Statement Input/Analyze (CEYIA1; CEYIA)	Initiation entry point from LPC. Allocates virtual storage. Reads and analyzes statement, branches to appropriate processor. Stacks RENAME, TRAITS, and COMBINE statements.	GETMAIN macro instruction.	Always called.
		GETLINE (CFADB) (routine of LPC)	To get next line of input.
		SCAN (CEYSC)	To scan linkage editor statement operation field.
		ERROR (CEYER)	For invalid delimiter or invalid statement.
		INCLUDE Statement (CEYIC)	When INCLUDE statement is received as input.
		END Statement (CEYEN)	When END statement is received as input.
Routine: Control Statement Processor -- Level: 2			
SCAN (CEYSC)	Scans a name of 8 or fewer characters until a delimiter is found.	ERROR (CEYER) (also a level 2 routine)	If name contains more than 8 characters.
ERROR (CEYER)	Delivers a diagnostic message.	PUTDIAG (CFADC) (routine of LPC)	Always called.
INCLUDE - Statement Processor (CEYIC)	Processes INCLUDE statement.	SCAN (CEYSC) (also a level 2 routine)	For each name appearing in the INCLUDE statement operand.
		ERROR (CEYER) (also a level 2 routine)	For reference to SYSLIB, nonexistent module, no form-1 INCLUDE given, or invalid delimiter in operand.
		OPEN macro instruction	For the library name in a form-1 INCLUDE statement.
		CLOSE macro instruction	After form-1 INCLUDE statement processing completed.
		BRING (CEYBR)	For each external name appearing in a form-1 INCLUDE statement, or for each unresolved external reference in the output module (for form-2 and -3 INCLUDE statements) which is not in EXCLUD table.
		TRAITS Statement (CEYTR)	For each TRAITS statement appearing in STACK table.
		COMBINE Statement (CEYCO)	For each COMBINE statement appearing in STACK table.
		RENAME Statement (CEYRN)	For each RENAME statement appearing in a STACK table.
		LINK (CEYLK)	For each external name appearing in a form-1 INCLUDE statement; or for each unresolved external reference in the output module (for form-2 and -3 INCLUDE statements), which is not in EXCLUD table.
		GTCSAD (CEYGA)	To get address of REF table in first CSD of input PMD (for form-2 and -3 INCLUDE statements only).
		EXTREF (CEYXR)	For each unresolved external reference in the output module (for form-2 and -3 INCLUDE statements only).

Table 2. Control statement processing hierarchical table (part 2 of 4)

<table>
<tr><td colspan="4">Routine: Control Statement Processor -- Level: 2 (cont'd)</td></tr>
<tr><th>Routine</th><th>Purpose</th><th>Called Routines</th><th>Calling Conditions</th></tr>
<tr><td>END Statement Processor (CEYEN)</td><td>Resolves references, attaches ISD if required.</td><td>FREEMAIN macro instruction</td><td>Always called.</td></tr>
<tr><td></td><td></td><td>EXTREF (CEYXR)</td><td>Always called. (Searches PMD for external references.)</td></tr>
<tr><td></td><td></td><td>GTCSAD (CEYGA)</td><td>Always called. (Calculates location of CSD tables.)</td></tr>
<tr><td></td><td></td><td>BRING (CEYBR)</td><td>If there are unresolved references in the output module.</td></tr>
<tr><td></td><td></td><td>COLLECT (CEYCT) (subroutine of END)</td><td>If an input module that satisfies an unresolved reference is in SYSLIB.</td></tr>
<tr><td></td><td></td><td>LINK (CEYLK)</td><td>To link modules (not in SYSLIB) that resolve references in the output module and do not satisfy names in the EXCLUD table.</td></tr>
<tr><td></td><td></td><td>ERROR (CEYER) (also a level 2 routine)</td><td>If there are unresolved references in the output module.</td></tr>
<tr><td></td><td></td><td>CLEANUP (CEYCL) (subroutine of END)</td><td>Always called.</td></tr>
<tr><td></td><td></td><td>FIXISD (subroutine of END)</td><td>If ISD is required.</td></tr>
<tr><td></td><td></td><td>CLOSE macro instruction</td><td>If there is an open library.</td></tr>
<tr><td colspan="4">Routine: Control Statement Processor -- Level: 3</td></tr>
<tr><td>BRING (CEYBR)</td><td>Gets PMD, text, and ISD from library.</td><td>FIND macro instruction</td><td>If library name given.</td></tr>
<tr><td></td><td></td><td>OPEN macro instruction</td><td>If library is not open.</td></tr>
<tr><td></td><td></td><td>CLOSE macro instruction</td><td>If open library is not library name given.</td></tr>
<tr><td></td><td></td><td>LIBE SEARCH (CZDC3 in dynamic loader)</td><td>If library name is not given.</td></tr>
<tr><td></td><td></td><td>GET macro instruction</td><td>Always called.</td></tr>
<tr><td></td><td></td><td>FREEMAIN macro instruction</td><td>To free old virtual storage if the input module exceeds the size allotted.</td></tr>
<tr><td></td><td></td><td>GETMAIN macro instruction</td><td>To get virtual storage for the input module if its size is larger than that allotted.</td></tr>
<tr><td></td><td></td><td>ERROR (CEYER) (a level 2 routine)</td><td>If the input or output module size exceeds available virtual storage.</td></tr>
<tr><td>TRAITS Statement Processor (CEYTR)</td><td>Processes TRAITS statement.</td><td>SCAN (CEYSE) (a level 2 routine)</td><td>Always called.</td></tr>
<tr><td></td><td></td><td>ERROR (CEYER) (a level 2 routine)</td><td>If a name does not exist, or is invalid, or if an invalid delimiter is used.</td></tr>
<tr><td>COMBINE Statement Processor (CEYCO)</td><td>Processes COMBINE statement.</td><td>SCAN (CEYSC) (a level 2 routine)</td><td>Always called.</td></tr>
<tr><td></td><td></td><td>GETCSD (CEYGC)</td><td>Always called unless statement errors are discovered.</td></tr>
<tr><td></td><td></td><td>GTCSAD (CEYGA) (also a level 3 routine)</td><td>Always called unless statement errors are discovered.</td></tr>
</table>

Table 2. Control statement processing hierarchical table (part 3 of 4)

Routine: Control Statement Processor -- Level: 3 (cont'd)			
Routine	Purpose	Called Routines	Calling Conditions
		COMSUB (CO100) (a COMBINE subroutine)	Always called unless statement errors are discovered.
		ERROR (CEYER) (a level 2 routine)	Called if a nonexistent name used in operand, if CSECTs to be combined have unlike attributes, or if an invalid delimiter is used.
RENAME Statement Processor (CEYRN)	Processes RENAME statement.	SCAN (CEYSE) (a level 2 routine)	Always called.
		ERROR (CEYER) (a level 2 routine)	If a duplicate or nonexistent name is supplied or if invalid delimiter is used.
		DELNAME (subroutine of RENAME)	To delete entry point names.
LINK (CEYLK)	Links input module to output module, deletes duplicate CSECTs and those marked for deletion. Updates ISD, CSDs, and HASHTB.	ERROR (CEYER) (a level 2 routine)	Called if there are duplicate entry names, or if attribute conflicts existed during CSECT rejection.
		UPISD (subroutine of LINK)	If ISD is required.
		APENDF (subroutine of LINK)	Called to append Definition Table to output CSD if at least one CSECT from the input module is linked.
		APENCX (CEYCX)	Called to append complex RLD to output CSD if at least one CSECT from the input module is linked.
		APENEX (CEYEX)	Called to append external RLD to output CSD if at least one CSECT from the input module is linked.
		APENIN (CEYIN)	Called to append internal RLD to output CSD if at least one CSECT from the input module is linked.
		GTCSAD (CEYGA) (also a level 3 routine)	Called to get CSD table addresses if at least one CSECT from the input module is linked.
GTCSAD (CEYGA)	Calculates locations of the six tables in a CSD: Definition Table Reference Table Complex DEF RLD External REF RLD Internal REF RLD Virtual Memory Page Table	None	
EXTREF (CEYXR)	Searches output module for next unresolved reference.	None	
COLLECT (CEYCT) (subroutine of END)	Update blank common CSECT size.	None	
CLEANUP (CEYCL) (subroutine of END)	Deletes entries marked for deletion in the output module, moves output module to final output area, updates sizes of common CSECTs.	GTCSAD (CEYGA) (also a level 3 routine)	Always called to get location of tables in CSD.
		APENCX (CEYCX)	Always called to append the complex RLD to the output area.
		APENEX (CEYEX)	Always called to append the external reference RLD to the output area and delete marked RLDs.

Table 2. Control statement processing hierarchical table (part 4 of 4)

Routine: Control Statement Processor -- Level: 3 Cont'd			
Routine	Purpose	Called Routines	Calling Conditions
		APENIN (CEYIN)	Always called to delete marked RLDs and append the internal reference RLD to the output area.
FIXISD (subroutine of END)	Completes table at beginning of composite ISD and appends input ISDs.	None	
Routine: Control Statement Processor -- Level: 4			
GETCSD (CEYGC)	Locates a CSD within the PMD.	None	
COMSUB (CO100) (a subroutine of COMBINE)	Combines two CSDs and text into a working area and updates combined CSD.	None	
DELNAME (subroutine of RENAME)	Deletes entry point names.	None	
UPISD (a subroutine of LINK)	Updates the ISD as each CSECT is linked to the output module.	None	
APENDF (a subroutine of LINK)	Appends the DEF table of a CSD to the output module.	None	
APENCX (CEYCX)	Appends the RLD for complex DEFs of a CSD to the output module.	None	
APENEX (CEYEX)	Appends the RLD for external REFs of a CSD to the output module.	None	
APENIN (CEYIN)	Appends the RLD for internal REFs of a CSD to the output module.	None	

Operation:

APENCX: RLDs for complex definitions that have previously been marked for deletion are deleted. Additions or deletions to any of the previous control sections that have already been moved to the output PMD change the relative locations of complex definitions for the CSD being processed, or the number of entry pages in the RLD for complex definitions.

If such additions or deletions have been made, the RLD entry pages and the byte displacement in the modifier entry are adjusted in a work area, and the RLD for complex definitions is moved to the output module.

APENEX: Modifiers for external RLD references are checked. Those that have previously been marked for deletion are deleted; those that have not been marked for deletion are moved to the output module.

APENIN: Modifiers for internal RLD references are checked. Those that have previously been marked for deletion are deleted; those that have not been marked for deletion are moved to the output module.

APENDF -- Append Definition Table Subroutine (Chart AL): APENDF, an open subroutine used exclusively by LINK, is entered by a direct branch. It appends the CSD's definition table to the output module and updates the external name list (NAMES).

This subroutine is entered with a pointer to the next available position in the output module, a pointer to the CSD's definition table, and a pointer to the CSD heading in the output module. All definitions previously marked for deletion are deleted. All RLDs for complex definitions whose byte value must be decremented because of the deletion of one or more definitions are adjusted accordingly. As each definition is moved from the PMD to the output module, the definition name is added to the external name list (NAMES).

Control is returned to LINK with a pointer to the next available position in the output module.

BRING -- Bring PMD, Text, and ISD from Library (CEYBR)

This routine fetches the PMD and text from a library, places them in the designated areas of storage, and, if required, brings the ISD to the next position in the ISD chain. (See Chart AN.)

Entry: Entry point CEYBR; entry parameters:

Registers:

0, 1
name to be found, left-justified, blank-filled.

2, 3
ddname for library to be searched, left-justified, blank-filled; zero if entire program library list is to be searched.

4, 5, 6, 7
pseudo parameters, whose values are preserved.

Calling Sequence: INVOKE ACEYBR.

Routines Called: ERROR, OPEN (VAM), FIND (VAM), CLOSE (VAM), GET (VPAM), LIBE SEARCH (part of Dynamic Loader).

Exit:

To Calling Routine: Normal for a "not found" exit, or with register 14 incremented by 4 for a "found" exit.

To LPC: The "can't continue" return is made to LPC in the event of an abnormal end return from GETLINE or in the event of storage overflow of a PMD, text, or ISD. The return code is set to 4.

Operation: This routine is entered with the name to be found, the ddname for the library to be searched, or a zero code if the entire program library list is to be searched.

If the ddname is given on entry, the FIND macro instruction is used to locate the module in the named library that satisfies the symbol. Control is returned to the calling routine via its "not found" exit if the named module cannot be found.

When a ddname is not given, the Library Search (LIBE SEARCH) subroutine is called through restricted linkage to search the entire program library list. LIBE SEARCH searches the job libraries, user library, and the system library to get the ddname of the library containing the module that will define a given symbol. If LIBE SEARCH does not find the module, it returns control to BRING via its "not found" exit, and control is returned to the calling module. If LIBE SEARCH finds the module, and it resides in SYSLIB, the SYSSW switch is set to "yes." The ddname for the library is placed in the DCB, and the DCB is opened.

If the module is in SYSLIB, the PMD is obtained using GET so that the size of any blank common CSECTs can be found. Text and ISD for modules in SYSLIB are ignored, and the module is not linked to the output module.

When the FIND macro instruction returns control to BRING via its "found" exit, BRING has a pointer which gives the length in bytes of the PMD, text, and ISD (if present).

Upon return from FIND, the size of the PMD is checked. If the PMD is too large for the current GETMAIN area, the FREEMAIN macro instruction releases areas occupied by the old PMD and text, and the GETMAIN macro instruction obtains space for the input PMD. If GETMAIN returns control to BRING via the error exit, ERROR is invoked to issue message 12, the FREEMAIN macro instruction releases all storage areas, and BRING returns control to LPC via the "can't continue" exit.

If the PMD is not too large or if GETMAIN returns via its normal exit, the estimated size of the output module is checked. If it is not too large, the GET macro instruction places the PMD in the module area. If the input text is too large, FREEMAIN releases the old text area, and GETMAIN obtains storage for the input text. If GETMAIN returns control to BRING via the error exit, ERROR is invoked to issue message 13, FREEMAIN releases all storage, and BRING returns control to LPC via the "can't continue" exit.

If the combined area of the input text and the old text is not too large for the total storage area, GET places the next pages of text in the work area. If the combined area of the input text and the old text is too large for the total storage area, ERROR is invoked to issue message 13, FREEMAIN releases all storage areas, and BRING returns control to LPC via the "can't continue" exit.

After GET places all text pages in the work area, BRING returns control to the caller via the "found" exit if the ISD is not required; if the ISD is required and there is no ISD input, control is returned via the "found" exit.

A check is made to determine if the new ISD combined with the previous ISD is too large for the storage area; if the combined ISDs are too large, ERROR is invoked to issue diagnostic message 12, FREEMAIN releases all storage areas, and BRING returns control to LPC via the "can't continue" exit. When the combined ISDs are not too large, GET chains the new ISD to previous ISDs.

If the estimated size of the ISD to be generated is too large for the storage area, ERROR issues message 13, FREEMAIN releases all storage areas, and BRING returns control to LPC via the "can't continue" exit. If the estimated size with the old PMD fits within the storage area, BRING returns control to the caller via the "found" exit. Any library opened in the BRING routine is closed before control is returned to the caller.

CLEANUP -- Cleanup Final Module Subroutine (Chart AH): The Cleanup Final Module (CLEANUP) subroutine deletes marked entries for entry point references and modifiers and moves the output PMD to the final output area. CLEANUP locates the first CSD in the output module. If no CSDs remain to be processed in the output module, control is returned to END. When a CSD is found, CLEANUP calls GTCSAD.

GTCSAD is entered at CEYGA via restricted linkage, with a pointer to the CSD heading. GTCSAD calculates the locations of the six tables for the CSD. GTCSAD returns control to CLEANUP with the address of the TABLE for CSD addresses. CLEANUP moves the CSD heading and definitions to the next position in the output area, clears the definition search and CSD links, moves the reference table to the next position in the work area, and clears the reference CSD links.

CLEANUP calls APENCX, via restricted linkage, to append the CSD's RLD for complex definitions to the final output area and to update the byte address modifiers. APENCX is entered with a pointer to the next available position in the PMD and a pointer to the CSD heading in the output module.

CLEANUP calls APENEX via restricted linkage. APENEX deletes RLDs for text (external reference) that have been marked and appends the CSD's RLD for text (external reference) to the output area.

CLEANUP calls the APENIN subroutine via restricted linkage. APENIN deletes RLDs for text (internal reference) that have been marked and appends the CSD's RLD for text (internal reference) to the output area. If there is text for the control section, the page table is moved to the output area; if the control section is blank COMMON, its size is updated in the CSD.

CLEANUP then checks to determine whether there are other CSDs in the module. When other CSDs are present, they are processed in a manner identical to the first CSD. After all CSDs are processed, control is returned to END.

COLLECT -- Collect Common Requirements Subroutine (Chart AI): COLLECT is entered by a direct branch from END when an external reference is resolved by a module in the system library. It steps through each CSD of the module; when it finds a blank common section, its size is checked, and CSIZE is updated if required. CSIZE thus holds the largest blank common size required by the output module. Exit is back to END.

COMBINE -- COMBINE Statement Processor (CEYCO)

This routine processes the COMBINE statement. (See Chart AA, Part 5.)

Entry: Entry point CEYCO; entry parameters:

Registers:
8
a common register pointing to the first character position of the statement operand.

9
a common register pointing to the input PMD.

Calling Sequence: Direct branch to location CEYCO from the INCLUDE routine.

Routines Called: GETSCD, SCAN, ERROR, GTCSAD.

Exit: Direct branch back to location ICRET in the INCLUDE routine.

Operation: This routine is entered with a pointer to the first character position of the operand and a pointer to the input module's PMD. The COMBINE statement is checked for an invalid delimiter and to determine if an operand exists. A call is made to ERROR to issue diagnostic message 11 if an invalid delimiter is found, or message 6 if the statement contains a name not present in the module. ERROR is called to issue diagnostic message 8 if the control sections have different attributes. SCAN is entered at CEYSC with a pointer to the first byte of the statement's first operand. SCAN ensures that the operand is correct (contains fewer than nine characters) and returns control. GETSCD is entered at CEYGC via restricted linkage, with a pointer to the PMD that contains the CSD and the name, located in TEMP, of the control section to be combined. Each control section name in the PMD is checked to see if it matches the name in TEMP. When a matching name is found, exit is taken with a pointer to the CSD heading. The affected CSD is then marked for combining, and GTCSAD is called.

GTCSAD is entered at CEYGA, via restricted linkage, with a pointer to the CSD heading. GTCSAD calculates the locations of the six CSD tables. Locations for these tables are stored in a 6-fullword area, TABLE (for CSD addresses). If an ISD is required, the control section name and its text displacement is placed in the Rename/Combine Table (RCTBL).

Note: All control sections to be combined cause the name and text displacement to be placed in the RCTBL, which contains entries that are to be placed in the ISD when the control section is linked to the output module. RCTBL is used by UPISD.

Processing for a multiple-entry statement is indicated by a comma; the following processing is required. GTCSAD is entered at CEYGA, along with a pointer to the first byte of the CSD heading of the first CSECT to be combined. GTCSAD calculates the locations for the six tables associated with this CSD. The location of each table is stored in TABLE. A call is made to SCAN to check the next control section name. After SCAN returns control, GTCSAD is again called to compute the location of the tables associated with the next control section to be combined. The control section name is marked for deletion. If an ISD is required, the control section name and its associated text displacement is placed in RCTBL; otherwise, such an entry is unnecessary. The COMSUB subroutine (described below) is entered by a direct branch to combine the two control sections and their applicable text. COMSUB also adjusts the tables associated with these two control sections. Identical processing is provided for each remaining control section name in the operand until a blank delimiter is encountered.

Termination of processing is denoted by a blank delimiter, in which case the pointers for the work areas are updated to indicate the next available position, and the pointers in the RCTBL are also updated, if the ISD is required. Control is then returned to location ICRET in the INCLUDE statement processor.

COMSUB -- Combine Control Section Subroutine (Chart AD): COMSUB, an open subroutine, is used exclusively by COMBINE. It performs the mechanics of combining two CSDs and their associated text and provides for the updating of the tables within the combined CSD.

This subroutine uses a work area for combining the CSDs, which is referred to in the description below and in the flowchart as work area A. It also uses a second work area for combining text; this is referred to as work area B.

Upon entry, COMSUB merges in work area A the CSD headings for each control section and updates the entries in the headings. Associated text for each control section is relocated to work area B, where the text for the first control section is placed before that for the second control section. Relocation values are applied to relocatable definitions and to relocatable and complex definitions displacements for R-values located in the second control section CSD. The definition and reference tables for both control sections are combined in work area A. Also, for the second control section, relocation values are applied to entries in the relocation dictionary (RLD) for complex definitions, the RLD for text external references, and the RLD for text internal references. The RLD entries, with relocation values applied, are combined with their counterpart entries in the RLDs for the first control section; combined entries are moved to work area A. The virtual memory page tables for both control sections are also combined and placed in a text work area. COMSUB then calculates and stores the length of the resultant CSD into the CSD heading.

DELNAME -- Delete Entry Name Subroutine (Chart AA, Part 7): DELNAME, an open subroutine used solely by RENAME, is entered by a direct branch.

This subroutine is entered with a pointer to the entry name to be deleted and a pointer to the associated CSD heading. The entry name is located in the definition table and marked for deletion. The RLD for complex definitions is found, and a search is made for modifiers referencing the deleted definition entry. All modifiers referencing a deleted definition are marked for deletion, and control is returned to the RENAME routine.

END -- END Statement Processor (CEYEN)

This routine processes the END statement. (See Chart AA, Part 9.)

Entry: Entry point CEYEN; entry parameters:

Register 10
a common register pointing to the output PMD.

Calling Sequence: Direct branch to location CEYEN from the INANAL module.

Routines Called: EXTREF, BRING, ERROR, LINK and GTCSAD.

Exit: Normal RETURN to LPC from the initiation entry. The exit parameters are:

Register 15 contains an exit code, as follows:

Exit code 0
normal return. It is assumed that LPC will respond with a call to either the linkage editor's continuation entry point or its early-end entry point.

Exit code 4
linkage editor cannot continue. The assumption is that LPC will respond with a call to either the linkage editor's initiation entry point, or its early-end entry point.

Operation: This routine is entered by INANAL with a pointer to the output module. The form-1 switch (FRM1SW) is checked for a state of one or zero. Control is returned to the LPC when the form-1 switch is zero, since no output module exists. When a form-1 INCLUDE statement has been processed (form-1 switch is one), EXTREF is invoked to search the output module for unresolved references.

EXTREF is entered at CEYXR via restricted linkage, with the following information: a pointer to the first reference to be checked, a count of the number of references remaining in the CSD's reference table, a pointer to the CSD heading, and a pointer to the output PMD. When an unresolved external reference is found, control is returned to END with a pointer to the unresolved reference, a count of the remaining references in the CSD, and a pointer to the CSD heading.

The CLEANUP subroutine is called when EXTREF returns control via its "not found" exit. CLEANUP is entered via direct branch to delete entries so marked by other processors, to update a blank common section size, and to move the output PMD to the final output area.

If the linkage editor's ISD has been generated, END then calls the Fix ISD (FIXISD) subroutine. FIXISD is entered by a direct branch and is used to place in the output ISD the length of the ISD, the name of the output module, and the displacement from each module heading to the corresponding input ISD. Also, FIXISD strings the input ISDs together, and returns control to END. After FIXISD returns control, or when the ISD is not required, the NONAME table is checked for entries, and ERROR is called to issue diagnostic message 7.

The NONAME table consists of 8-byte entries, representing the alphameric names of external references, which cannot be resolved from the program library list. These names are sent to the user via message 7.

The SLBNAM table is also checked for name entries (references resolvable from SYSLIB) and, when a name entry is found, ERROR is called to issue diagnostic message 19. If a name entry does not exist in SLBNAM, or after processing by ERROR, the diagnostic code is set in the module heading, the return code is set, and control is returned to LPC.

During the initial part of END activities, EXTREF searches the output module for unresolved external references; if such references exist during that time, a check is made to determine if there is an RLD modifier. If there is no modifier, the unresolved reference is deleted, and the RLD reference numbers are updated. In any case, the following processing occurs.

BRING is called via restricted linkage, with the following information: a pointer to the name (unresolved external reference) to be found, and the ddname set to zero, denoting a search of the entire program library list. BRING, a closed subroutine entered at CEYBR, obtains the PMD, text, and ISD (if required) for the module that will satisfy the unresolved external reference. BRING returns control to END whether or not a name satisfying the reference is found. A check is made to determine whether the "found" reference is present in the EXCLUD table, when BRING returns control via the "found" exit. If the found reference is in the EXCLUD table, and the module in which the reference was located is in SYSLIB, the reference is entered in the SLBNAM table, EXTREF is again called to search the output module for the next unresolved external reference, and processing identical to that for the first reference is repeated.

When BRING returns via its "not found" exit, or when the found reference is in the EXCLUD table but not in a SYSLIB, the reference is placed in the NONAME table, and EXTREF is again called to search the output module for the next unresolved external reference, and processing identical to that for the first reference is repeated.

When the name satisfying the unresolved reference is not in the EXCLUD table, and when names in the module do not satisfy any of those in the EXCLUD table, a check is made to determine if the module resolving the reference is contained within SYSLIB.

If the module is not in SYSLIB, and it is not the second attempt to link a FORTRAN main program, LINK is entered at CEYLK via restricted linkage with pointers to the input PMD and the output module. LINK joins the module that resolves the unresolved external reference to the linkage editor's output module. LINK deletes duplicated control section names and control sections previously marked for deletion, ensures that there is only one COMMON control section, and updates the module's CSDs and their associated text. LINK then returns control to END. The reference is checked to determine if it was resolved; if it was resolved, EXTREF is again called to search the output module for the next unresolved external reference, and processing identical to that for the first reference is repeated. If the reference was not resolved, the reference is placed in the NONAME table, and EXTREF is again called to provide processing identical to that provided when the first reference was not resolved.

When the module returned by BRING is contained in SYSLIB, the Collect Common Requirements (COLLECT) subroutine is called. COLLECT, entered by a direct branch, locates a blank common control section in a SYSLIB module, and after checking the common size of the section, updates CSIZE.

ERROR -- Error Message Processor (CEYER)

This routine provides error messages during linkage editor activity, via the PUTDIAG routine of LPC. (See Chart AO.)

Entry: Entry point CEYER; entry parameters:

Registers:

0
a parameter indicating the type of exit. Zero signifies exit to INANAL, and nonzero signifies return to the caller.

1
error code, identifying the message to be delivered.

Calling Sequence: INVOKE ACEYER, or direct branch to location CEYER.

Routines: Called PUTDIAG routine of LPC.

Exit:

To Calling Routine: If register 0 contains 0, exit is via a direct branch to location IARET (CEYIA) in INANAL. If register 0 is nonzero, return is made to the caller. There are no output parameters.

To LPC: A "can't continue" return is made to LPC in the event of an abnormal end return from PUTDIAG.

Operation: In nonconversational mode, switch ERDIAG is set to denote a level-2 diagnostic code. If the mode is conversational, a switch (MSGSW) is set which notifies INANAL that an altered line is expected on the next entrance to the GETLINE routine of LPC.

The PUTDIAG routine of LPC is called with type-I linkage when an error message is to be delivered. Upon entry, ERROR stores the user response indicator, and the appropriate error message is formatted and moved to the buffer area. The LPC parameter is set to indicate user response, and the PUTDIAG routine of LPC is called with type-I linkage. PUTDIAG stacks the message until the next time the linkage editor calls the GETLINE routine of LPC; at that time the message is delivered. ERROR returns control to LPC MAIN with a "can't continue" indication on an ABEND return from PUTDIAG. Otherwise, it returns to the calling routine or to INANAL.

EXTREF -- External Reference Search (CEYXR)

This routine searches the output PMD for the next unresolved external reference. (See Chart AP.)

Entry: Entry point CEYXR; entry parameters:

Registers:

0, 1, 6, 7
pseudo parameters whose values are preserved.

2
pointer to the first reference to be checked.

3
count of the number of references remaining in the reference table.

4
pointer to the CSD heading.

5
pseudo parameter whose value is preserved.

Calling Sequence: INVOKE ACEYXR.

Routines Called: None.

Exit: Normal for a "not found" exit. On a "found" exit register 14 is incremented by 4. Exit parameters:

Registers:
2
pointer to an unresolved reference, if any, or to the end of the PMD.

3
updated count of remaining references in the CSD.

Operation: References located in the CSDs are checked to see if they are unresolved external references; i.e., references that have not been satisfied by a definition. If such a reference is found, control is returned to the calling module via the found exit, along with a pointer to the unresolved reference and the count of remaining references in the CSD. All references in the CSD are checked in the same way the first reference is checked. A "not found" exit is taken when all the CSDs in the output module have been checked for unresolved references.

FIXISD -- Fix ISD Subroutine (Chart AI): FIXISD is entered by a direct branch from END when an ISD has been generated. FIXISD completes the ISD by filling in the length of the ISD, the name of the output module, and displacements from each module heading to the corresponding input ISD. It then attaches the string of input ISDs, thus forming a composite ISD. Exit is back to END.

GETCSD -- Locate Control Section Dictionary (CEYGC)

This routine locates a CSD in a PMD. (See Chart AP.)

Entry: Entry point CEYCG; entry parameters:

Registers:
0-2, 4, 5, 6, 7
pseudo parameters, whose values are preserved.

9
common register which contains a pointer to the PMD.

Calling Sequence: INVOKE ACEYGC.

Routines Called: None.

Exit: Normal for the "not found" exit. On a "found" exit, register 14 is incremented by 4. Exit parameters:

Register 3
pointer to the CSD heading on a "found" exit, or a pointer to the end of the PMD on a "not found" exit.

Operation: GETCSD is entered with the name of the control section in the PMD that is to be located and a pointer to the PMD. The control section name is located in a temporary storage (TEMP) of eight characters in the PSECT. Each control section name in the PMD is checked to see if it matches the name in TEMP. If a matching name is found, the normal exit is taken; otherwise, the "not found" error exit is taken.

GETLINE Routine (CFADB)

This routine is part of language processor control and is described in Command System Program Logic Manual, GY28-2013.

GTCSAD -- Get CSD Table Addresses (CEYGA)

This routine calculates the locations of the six tables for a given CSD. (See Chart AQ.)

Entry: Entry point CEYGA; entry parameters:

Registers:
3
a pointer to the first byte of the CSD heading.

0-2, 4, 5, 6, 7
pseudo parameters, whose value is preserved.

Calling Sequence: INVOKE ACEYGA.

Routines Called: None.

Exit: Normal; exit parameters:

The following information is placed in location TABLE:

Word 1	Location of Definition Table
Word 2	Location of Reference Table
Word 3	Location of RLD for Complex Definitions
Word 4	Location of RLD for External References
Word 5	Location of RLD for Internal References
Word 6	Location of Virtual Memory Page Table

Operation: This routine is entered with a pointer to a CSD heading in a PMD and calculates the location of the six tables for that CSD; that is, the Definition and Reference Tables, the RLDs for Complex Definitions, External References, and Internal References, and the Virtual Memory Page Table. This information is stored in a temporary save area of six fullwords called TABLE, in a PSECT.

INANAL -- Control Statement Input/Analyze Processor (CEYIA1)

This routine performs initialization, receives linkage editor statements from LPC, and transfers control to the routine that processes the particular statement. (See Chart AA, Part 1.)

Entry: Entry point CEYIA1; entry parameters:

Register 1
address of a parameter list.

The parameter list pointed to by register 1 consists of a series of address constants aligned on word boundaries as follows:

Word 1 Address of a field containing the output module name.

Word 2 Address of a 1-byte field that contains 00000000 if batch mode applies; 00000001 if conversational.

Word 3 Address of a 3-byte option table with the following significance:

Byte 1 - Produce ISD
Byte 2 - Produce PMD Listing

Each of the above bytes contains the EBCDIC character Y if the option is desired; N, if it is not. The presence of any other character causes the linkage editor to take the built-in default action for the specified option.

Byte 3 - Produce List Data Set

The above byte contains Y if the selected PMD listing is to be stored in a list data set, and N if it is to go to SYSOUT.

Word 4 Address of the data control block (DCB) for the PMD list data set. Unless the data set is to be written on SYSOUT (byte 3 above contains N), the linkage editor opens, uses, and closes this data set.

Routines Called: CZCGA (via GETMAIN macro), GETLINE (in LPC), SCAN, ERROR, INCLUDE, and END.

Exits:

Normal -- Returns to LPC from the END routine with a return code of 0.

Abnormal -- Returns to LPC with a return code of 4 if the GETLINE routine returns an abnormal end indication.

Operation: This routine is entered with a parameter list containing address pointers to:

- The name of the list data set that will ultimately be the name of the output module.
- Data stating mode of linkage editor operation, conversational, or nonconversational.
- Data indicating whether to produce the Internal Symbol Dictionary and PMD listing, and whether the PMD listing, if requested, is to go into a list data set or on SYSOUT.
- The address of the data control block (DCB) for the list data set. (Ignored if requested PMD listing goes to SYSOUT.)

Program switches are set according to these input parameters.

The following major work areas and tables are initialized: Common Internal Storage Areas, Rename/Combine Table (RCTBL), ISD, and Exclusion Table (EXCLUD). The name assigned to the output module is placed in the NAMES table. The GETMAIN macro instruction is used to obtain storage for the input and output PMD, text, and ISD, as well as for work areas WORKC1, WORKC2, and WORKT.

After initialization the GETLINE macro instruction is used to read a control statement from LPC. The return code is analyzed to determine if there is an abnormal end, batch end of data set, or altered line return. Processing is terminated if GETLINE returns a code of 12 (abnormal end), and the linkage editor returns control to LPCMAIN with the return code set to 4 (can't continue). If GETLINE returns a code of 8 (batch end of data set), END is entered. If the user makes a correction in his statements (GETLINE return code = 4), the corrected set of statements is processed again; if statements already completely processed are changed by the user, or if an error occurs in an INCLUDE statement which links more than one module, processing starts over from the beginning. GETLINE is again used to get the line following the first statement of the statement group which is in error; i.e., the previous INCLUDE statement. If no corrections are made by the user to incorrect statements, the output module is marked with a diagnostic code.

Extraneous blank characters are stripped from the control statement, and after a check that the statement is within the 256-character length limit, it is placed in a statement store area (SAVLN1). If the operand of the input statement contains more than one entry, GETLINE is used to obtain the next statement from LPC. SCAN is invoked, using restricted linkage, to verify that the operands contain fewer than nine characters. SCAN processing is terminated when a delimiter (which is saved for later reference) is encountered. The recorded delimiter is then examined, and ERROR is invoked for a nonblank delimiter. ERROR issues message 11 and returns control. COMBINE, RENAME, and TRAITS statements are placed in the stack table (STACK) to await processing of an INCLUDE statement. INCLUDE and END statements cause a direct branch to the routine that processes the particular statement. Invalid statement verbs cause ERROR to be invoked to issue message 3.

INCLUDE -- INCLUDE Statement Processor (CEYIC)

This routine processes the three forms of the INCLUDE statement. (See Chart AA, Part 2.)

Entry: Entry point CEYIC; entry parameters:

Registers:

8
: a common register pointing to the first unprocessed character position of the statement operand.

9
: a common register pointing to the input PMD.

10
: a common register pointing to the output PMD.

Calling Sequence: Direct branch to location CEYIC from INANAL.

Routines Called: LINK, SCAN, BRING, ERROR, EXTREF, RENAME, TRAITS, and COMBINE.

Exit: Direct branch to INANAL (location IARET).

Operation: Register 8 points to the first unprocessed character position of the statement operand. Upon entry, SCAN is invoked at CEYSC, using restricted linkage. SCAN is entered with a pointer to the first-byte position of the library name. SCAN verifies that the library name contains fewer than nine characters. The delimiter following library name is recorded, the library name is placed in TEMP, and SCAN returns control, with pointers to the first-byte position after the delimiter, to TEMP and to the library name.

If the library name in the INCLUDE statement is the name SYSLIB, ERROR is invoked to print message 10, and INCLUDE statement processing is terminated. When the library name indicates a library other than SYSLIB, the ddname of the specified library name is planted in the DCB. The OPEN macro instruction is used to open the indicated partitioned library. The delimiter provided by SCAN is used to determine the form of the INCLUDE statement being processed.

Form-1 INCLUDE Processing: SCAN is invoked to process each module or entry name to ensure the correctness of the operand. If there is any delimiter between module names other than a comma, a direct branch is made to ERROR to generate message 11.

SCAN is invoked to process the first module or entry name in the control statement operand. BRING is invoked at CEYBR, using restricted linkage, with the name to be found and the ddname for the library to be searched. BRING fetches the PMD and text from the indicated library and places them in a work area. If required, the ISD is moved to the next position in the ISD chain. When the object module is not found in the indicated library or is found in SYSLIB, ERROR is invoked to generate message 1 or 2. If this is the end of the INCLUDE statement processing, control is returned to INANAL; otherwise, processing identical to that for the first module or entry name continues for the other entries in the operand of the statement.

If BRING finds the module, the STACK table is searched to determine whether a RENAME, COMBINE, or TRAITS statement exists; if any of these statements is found, a direct branch is made to the appropriate control statement processor. When the STACK table is empty (no RENAME, COMBINE, TRAITS statements), the work areas are initialized.

If the module being included is a FORTRAN main program and a FORTRAN main program has already been included, ERROR is invoked to generate diagnostic message 22. The input parameters will indicate to ERROR that user response is expected and that it shall exit to INANAL. If there is no attempt to linkage edit two FORTRAN main programs, LINK is invoked at CEYLK.

LINK is invoked, using restricted linkage; input parameters are pointers to the PMD and the output module. LINK links the module indicated in the form-1 INCLUDE statement to the linkage output module. After the module is linked, the form-1 switch (FRM1SW) is set to 1, and the delimiter is again checked. A delimiter other than a right parenthesis or comma causes

ERROR to issue message 11. A comma indicates another name in the operand of the current form-1 INCLUDE statement; therefore, processing identical to that for the first name is repeated. A right parenthesis indicates no other entries in the current form-1 INCLUDE statement. Control is returned to INANAL, indicating the end of form-1 INCLUDE statement processing.

Form-2 INCLUDE Processing: Form-2 processing is allowed only after at least one form-1 statement has been processed; if a form-2 statement appears before a form-1, ERROR will issue message 9. The STACK table is then searched for entries; any entry causes ERROR to issue message 9. If no entries exist in the STACK table, GTCSAD is invoked at CEYGA through restricted linkage with a pointer to the CSD heading in the PMD. GTCSAD builds a table that contains the location of the six tables in the CSD. These tables are definition and reference tables, the relocation dictionaries (RLDs) for complex definitions and external and internal references, and the virtual memory page table. EXTREF is invoked at CEYXR using restricted linkage with a pointer to the first reference to be checked, a count of the number of references remaining in the reference table, and a pointer to the CSD heading.

EXTREF searches the output module for unresolved references and provides a pointer to an unresolved reference if any is found, a pointer to the CSD heading, and the updated count of remaining references in the CSD. If the unresolved reference is not in the EXCLUD table, BRING is invoked to fetch the PMD and text of the module that satisfies the unresolved reference and to bring the ISD to the next position in the ISD chain. The module is contained in the library specified by the ddname in the form-2 INCLUDE statement. LINK is called to link the module, found by BRING, to the output module when the following conditions are satisfied: external names in the retrieved module do not match entries in the EXCLUD table, the retrieved module is not located in SYSLIB, none of the names in the module are to be excluded, and it is not an attempt to linkage edit more than one FORTRAN main program.

When EXTREF returns control with an unresolved reference, processing identical to that for the first reference is provided. If an unresolved reference is not found, the CLOSE macro instruction closes the partitioned library, the library name is removed from LBOPEN, and control is returned to INANAL, indicating the termination of processing for the form-2 INCLUDE statement.

Form-3 INCLUDE Processing: During form-3 processing, the form-1 switch (FRM1SW) must be set to 1; if FRM1SW is not 1, ERROR issues message 9. The first character in the statement's operand is checked; a character other than a left parenthesis causes ERROR to issue message 11. SCAN is invoked to ensure that the external reference name specified in the statement's operand contains fewer than nine characters. The external reference name is recorded in the EXCLUD list (defined in the introduction). Upon checking the delimiter following "external reference name," ERROR is called to issue message 11 if the delimiter is other than a comma or a right parenthesis. If the delimiter is a comma, indicating more than one entry in the statement, processing identical to that performed for the statement's first entry (external reference name) is repeated. If the delimiter is a right parenthesis, indicating no more entries in the statement's operand, processing identical to that provided for the form-2 INCLUDE statement is provided.

LIBE SEARCH -- Library Search Subroutine (CZCDC3): LIBE SEARCH is a closed subroutine in the dynamic loader that locates an object module which contains a particular symbol, and is used to supply BRING with the ddname of the library containing the module. LIBE SEARCH is described in Dynamic Loader Program Logic Manual, GY28-2031.

LINK -- Link Modules (CEYLK)

This routine attaches an input module to the linkage editor output module. (See Chart AJ.)

Entry: Entry point CEYLK; entry parameters:

Registers:

0-5, 6-7
pseudo parameters whose values are preserved.

9
common register which contains a pointer to the input PMD.

10
common register which contains a pointer to the output PMD.

Calling Sequence: INVOKE ACEYLK.

Routines Called: APENEX, APENIN, APENCX, GTCSAD, ERROR.

Operation: This routine is entered with a pointer to the input module's PMD and a pointer to the output module. If it is the first module to be attached, its PMD heading is moved to the output module, the PCS communication indicator (TDYPCS) is set, and the name to be assigned to the output module is placed in the output module.

When a FORTRAN main program is linkage edited, the standard entry point of the output module will correspond to the existing standard entry point of the FORTRAN main program. The LINK routine will achieve this standard entry point similarity by saving the main FORTRAN header and by returning to INANAL to reinitiate the INCLUDE statement processing. When the LINK routine is again called to link the first input module, it will determine whether a main FORTRAN header has been previously saved. If a saved header exists, it will be stored into the output PMD header.

The CSD pointer is set to the first CSD in the module, and the COMBINE switch (LKCMSW) is set to zero. The CSD in the input module's PMD is checked to determine if it is to be deleted or combined, or if it is a blank common control section. Control sections that were combined are placed in the combine section of Work Area by COMBINE. Blank common control sections, other than the first encountered, are marked for deletion. The size of the largest blank common section encountered becomes the size of the retained control section.

The following checks are made for each CSD; if necessary, ERROR is called to issue each message mentioned:

- A nonblank definition name which is not a control section name, and which matches a definition name in the output module, results in message 4; the input PMD definition is marked for deletion.
- A nonblank definition name which is a control section name, and which matches a definition name in the output module, results in message 14; the duplicate (input) control section is marked for deletion.
- Names in the PMD and output module that name control sections without common attributes result in message 15.
- When the output module CSECT is read-only or privileged and the input PMD CSECT is nonread-only or nonprivileged, respectively, message 16 is delivered.
- A nonprivileged output module CSECT and a privileged input PMD CSECT result in message 17.
- An input PMD CSECT with a length greater than the output module CSECT results in message 18.

After the above error checks have been completed for each CSD in the input PMD, actual linking of CSECTs begins. The output ISD module heading is updated only if the ISD is required.

The LINK routine now determines whether the standard entry point (SEP) of the input module may be included as a DEF in the output module's PMD. To be retained in one of the CSDs of the output module's PMD, the DEF created must be able to retain V- and R-values which will reference exactly the same areas of text referenced by the standard entry point DEF in the input module. If LINK determines this is not possible, warning is provided with message 24.

LINK records all information needed by subroutine APENDF to preserve the original standard entry point as either a relocatable or complex DEF. The DEF retained will be relocatable if the input module has only PSECTs or only CSECTs and the standard entry point's REF is the first control section's name. The DEF will be complex if the text and DEF still exist for both the first PSECT of the input module and the CSECT named in the standard entry point REF. It will also be complex if the input module has more than one control section, all of which are the same type, and if the standard entry point REF name of the input module is not the name of the module's first control section.

If the control section name DEF in which the standard entry point DEF is to be retained has been marked for deletion in the output module, LINK issues warning message 24, unless the control section is found to have been combined with another. In this event, the standard entry point DEF will be placed in the combined CSD which contains the deleted control section name DEF.

LINK now begins processing each control section in the input module by calculating the number of text pages for each section. If the end of the input PMD containing the CSD has not been reached, a check is made to determine whether the control section has been marked for deletion or combining. If the control section is marked for deletion, the CSD pointer is bumped to the next CSD in the PMD; the text pointer is bumped to the next control section text page.

When the control section is marked for combining, the CSD pointer is saved in the combine switch (LKCMSW), the text page count and text pointer are saved; the CSD pointer and text pointer to the combined section in work area are set; and the number of text pages that have been combined is computed.

If necessary, the Update ISD (UPISD) subroutine is called by a direct branch to update the ISD as each control section in the PMD is processed. If it is not necessary to update the ISD, GTCSAD is called. GTCSAD, entered at CEYGA via restricted linkage, calculates the locations of the following six tables for the CSD: the Definition and Reference Tables; the RLDs for Complex Definitions, External References, and Internal References; and, the Virtual Memory Page Table. GTCSAD places these locations in the TABLE (CSD Addresses) and returns control to LINK. LINK appends the CSD heading to the output module and calls the Append Definition Table (APENDF) Subroutine.

APENDF is entered with a pointer to the next available position in the output module, a pointer to the CSD definition table, and a pointer to the CSD heading in the output module. LINK also passes information indicating whether the input module's standard entry point DEF is to be included as a DEF in this CSD. APENDF first appends the CSD's definition table to the output module. Then, if the standard entry point DEF is to be saved as a relocatable DEF, it is added following the DEF for the control section name, and the count of relocatable DEFs kept in the output CSD is updated. If the SEP DEF is to be complex, it is added as the last DEF in the definition table, and the count of complex DEFs is updated. APENDF then updates the external name list (NAMES) and returns control to LINK. LINK appends the CSD reference table to the output module and calls subroutine APENCX.

APENCX is entered at CEYCX via restricted linkage, and is used to append the CSD's complex RLD to the output module. LINK prepares to call APENCX (Append Complex DEF RLD) by determining if the present CSD contains the standard entry point DEF which has been saved. If it has and is a complex DEF, then any new modifier pointers needed are added to the complex DEF RLD modifier pointer list, previous pointers are updated, and a new modifier is added for the new complex DEF. APENEX is entered at CEYEX to append the CSD's RLD for text (external references) to the output module; and APENIN is entered at CEYIN to append the CSD's RLD for text (internal reference) to the output module. After control is returned to LINK, the virtual memory page table entries are appended to the output module. If there is text for the CSD, it is appended to the output module. If the control section just appended to the output module was a combined control section, the CSD and text pointers and page count are restored to their former values, and the pointers are advanced to the next control section.

When the end of the PMD is reached, the cumulative length of the input ISDs is incremented by the length of the current ISD, the ISD pointer is moved to the next vacancy, and the RCTBL table heading is initialized. Control is returned to the calling module if all control sections from the PMD have been deleted. If all control sections in the PMD have not been deleted, the number of definitions in the table is computed, all new definitions are chained to the hash table via the definition search links, and the index to the CSD heading is placed in the new definition CS links. The pointer to the next CSD is obtained, and the processing for its definitions identical to the first CSD is performed if the end of the output module has not been reached. If the end of the output module has been reached, the CSD pointer is set to the first CSD in the output module, and GTCSAD is called.

GTCSAD is entered at CEYGA via restricted linkage, with a pointer to the CSD's Definitions, References, RLD for Complex Definitions, RLD for External References, RLD for Internal References, and the Virtual Memory Page Tables. GTCSAD returns control to LINK, and LINK chains as many references in the CSD as possible to the corresponding definitions in the CSD via the reference use links.

After all references in this CSD are processed, the CSD pointer is advanced to the next CSD; if the end of the output module has been reached, control is returned to the calling module; if the end of output module has not been reached, remaining CSDs in the output module are processed similarly to the first CSD.

PUTDIAG (CFADC1 in Module CFADC)

This routine is part of LPC and is explained in Command System Program Logic Manual, GY28-2013.

RENAME -- RENAME Statement Processor (CEYRN)

This routine processes the RENAME statement. (See Chart AA, Part 6.)

Entry: Entry point CEYEN; entry parameters:

Registers:

8
: a common register pointing to the first character position of the statemen operand.

9
: a common register pointing to the input PMD.

Calling Sequence: Direct branch to location CEYEN from the INCLUDE routine.

Routines Called: ERROR, SCAN.

Exit: Exit from this routine is made by a direct branch back to location ICRET of the INCLUDE routine.

Operation: This routine is entered with a pointer to the input PMD and a pointer to the first character position of the statement operand. Using restricted linkage, SCAN is entered at CEYSC with a pointer to the first byte of the statement's old name. The old name is scanned until a delimiter character (left or right parenthesis, minus sign, comma, or blank) is encountered. If the old name contains more than eight characters, SCAN calls ERROR which issues message 10. SCAN returns control with a pointer to the first byte that follows the delimiter. The delimiter is examined to determine the processing required for the RENAME statement:

- A left parenthesis for a delimiter causes a control section, entry name, or an external reference to be renamed.
- A blank or comma for a delimiter causes a control section or entry name to be deleted.

Delimiters other than a blank, comma, or left parenthesis causes ERROR to issue message 11.

If the delimiter is a left parenthesis, SCAN is called again to process the new name in the RENAME statement in the same way provided for the old name. If the next delimiter is other than a right parenthesis, ERROR is called to issue message 11.

A pointer is set to the first control section dictionary in the module denoted by the old name, and another pointer is set to the combined section (which is in a work area) if the control section has been combined. When the old name is a definition and the new name is already defined in the output module, ERROR is called to issue message 5, after which control is returned to INCLUDE. If the old name is a definition and the new name is not defined in the output module, the old name in the input PMD is replaced by the new name.

When the old name is a definition denoting a control section name and the ISD is required, the control section name is placed in the Rename/Combine Table (RCTBL). (Note that all renamed control sections cause entries to be placed in the RCTBL.) These entries will be placed in the ISD when the control section is linked to the output module. RCTBL is used by the Update ISD (UPISD) subroutine, which is a part of the LINK module. The old name in the RENAME statement may also denote an external reference; if so, that external reference is replaced by the external reference denoted by the new name. After the first control section is processed, the input module's PMD is checked for remaining control sections; if additional control sections are present, they are processed in the same way the first control section was processed.

If, at the completion of the PMD search, the external name denoted by the old name did not match a name in the PMD, ERROR is called to issue message 6.

The delimiter following the old name operand in the RENAME statement is checked; delimiters other than a comma or blank cause ERROR to issue message 11. RENAME statement processing is terminated when a blank delimiter is encountered, at which time control is returned to the INCLUDE routine.

If a control section or entry name is to be deleted (delimiter was a comma or blank during the time the old name was first scanned), a pointer is set to the first CSD in the input PMD. A CSD pointer is also set to the combined section (which is in a work area) when the control section is marked for combining. The control section denoted by the old name causes that control section to be marked for deletion. A direct branch is made to the Delete Entry Name (DELNAME) subroutine if the old name denotes an entry name. DELNAME is entered with a pointer to the entry name to be deleted and a pointer to the associated CSD heading. DELNAME locates and marks for later deletion the desired entry name, and updates the related CSD tables. On return to RENAME, the input PMD is checked for remaining CSDs. Remaining CSDs are processed in the same way the first CSD was processed. ERROR is called to issue message 6 if no CSDs remain in the PMD, and when the name denoted by the old name has not been matched with an identical definition in the input PMD. A check is made of the delimiter following the old name; processing is terminated when a blank delimiter is encountered, and control is returned to the INCLUDE routine.

SCAN -- Scan (CEYSC)

This routine scans a name in search of a delimiter. (See Chart AQ.)

Entry: Entry point CEYSC; entry parameters:

Registers:
0, 1, 3, 4, 5, 6, 7
pseudo parameters, whose values are preserved.

8
common register which points to the first byte position of a name.

Calling Sequence: INVOKE ACEYSC.

Routines Called: ERROR.

Exit: Returns to calling routine unless more than eight characters appear before a delimiter. In this case, a direct branch exit is made to ERROR at location CEYER. Exit parameters:

Registers:
2
delimiting character in low-order eight bits.

8
common register which points to the byte position following the delimiter.

Operation: The name passed to this routine is examined (by means of a translate and test table) for a delimiting character. If more than eight characters are encountered before a proper delimiter is found, a direct branch is made to ERROR to issue message 10. The delimiter character is recorded and the scanned name is placed in TEMP (a temporary save area of eight characters). TEMP is filled with blanks when the first character scanned is a delimiter.

Exit from SCAN is made using restricted linkage. SCAN provides the calling routine with the delimiter character and pointers to the byte position following the delimiter.

Allowable delimiters are left parenthesis, right parenthesis, hyphen, comma, and blank.

TRAITS -- TRAITS Statement Processor (CEYTR)

This routine processes the TRAITS statement. (See Chart AA, Part 8.)

Entry: Entry point CEYTR; entry parameters:

Registers:
8
a common register pointing to the first character position of the statement operand.

9
a common register pointing to the input PMD.

Calling Sequence: Direct branch to CEYTR from the INCLUDE routine.

Routines Called: SCAN, ERROR.

Exit: Direct branch back to location ICRET in the INCLUDE routine.

Operation: This routine is entered with a pointer to the first character position of the TRAITS statement operand and a pointer to the input module's PMD. SCAN is entered at CEYSC with a pointer to the first byte of the statement operand, which indicates the control section to be processed. SCAN scans the control section name until a delimiter character is found. When the control section name contains more than eight characters, SCAN calls ERROR, which issues message 10. A delimiter other than a left parenthesis or blank causes TRAITS to branch to ERROR to issue message 11.

The input PMD is searched for the control section's CSD by comparing the control section name defined by the TRAITS statement to the control section name in the CSD header. If a CSD is marked for combining, the control section name of the CSD header in the work area (not in the input PMD) is compared to the control section name defined by the TRAITS statement. (The name is stored in TEMP.) ERROR is called to issue message 11 if the control section name is not found in the input PMD, or the work area when processing combined control sections. The attributes word is cleared; that is the FIXED attribute is assigned. The delimiter following the statement's operation code is then checked. If the delimiter is a blank, control is returned to INCLUDE, and the FIXED attribute is thus automatically assigned to the control section. If the delimiter is other than a blank, SCAN is called to process the TRAITS statement's trait operand. ERROR is called to issue message 10 if the attribute word is invalid. When a valid attribute is detected, the appropriate trait code is assigned to the applicable attributes word in the CSD heading. A check is again made of the delimiter to determine whether or not it is a comma, right parenthesis, or neither of these. A comma is indicative of more than one operand; consequently, processing for the additional operand or operands is identical to that processing provided for the first operand. Control is returned to INCLUDE for a right parenthesis, which indicates end-of-statement processing. Delimiters other than a comma or a right parenthesis cause ERROR to be called to deliver message 11.

UPISD -- Update ISD Subroutine (Chart AK): UPISD is an open subroutine, used exclusviely by LINK, that updates the ISD when the input module control section is being attached to the output module. After being entered by a direct branch, the RCTBL table is searched for an output control section name that matches the current PMD control section name. If the name found is a RENAME entry, RCTBL is searched for an output control section name matching the input control section name. In essence, this is a check for a COMBINE entry with its name subsequently renamed. If such a name is found, the output control section name (from the RENAME entry) is moved from RCTBL to the COMBINE entry in the ISD.

If a match is not found in RCTBL, the ISD receives the existing output control section name, and control is returned to LINK.

OUTPUT PROCESSOR

Function Summary

After all linkage editor control statements (concluding with an END statement) have been successfully handled by the Control Statement Processor and return has been made to LPCMAIN, LPCMAIN invokes the linkage editor at its second (continuation) entry point to produce the output module and, if necessary, a PMD listing. This output processing is performed by the Output Processor routine (OUTPUT) and a number of subroutines.

OUTPUT puts together the components of the module (PMD, text, and, if requested, the composite ISD), produces an external name list (a list of alias names by which the module to be stored may later be found), and prepares a PMD listing, if the user has requested it. At the completion of this processing, OUTPUT returns to LPCMAIN, having filled in a parameter list containing the locations of the output module's components and the external name list.

A more detailed description of the linkage editor's OUTPUT Routine follows Figure 6 and Table 3.

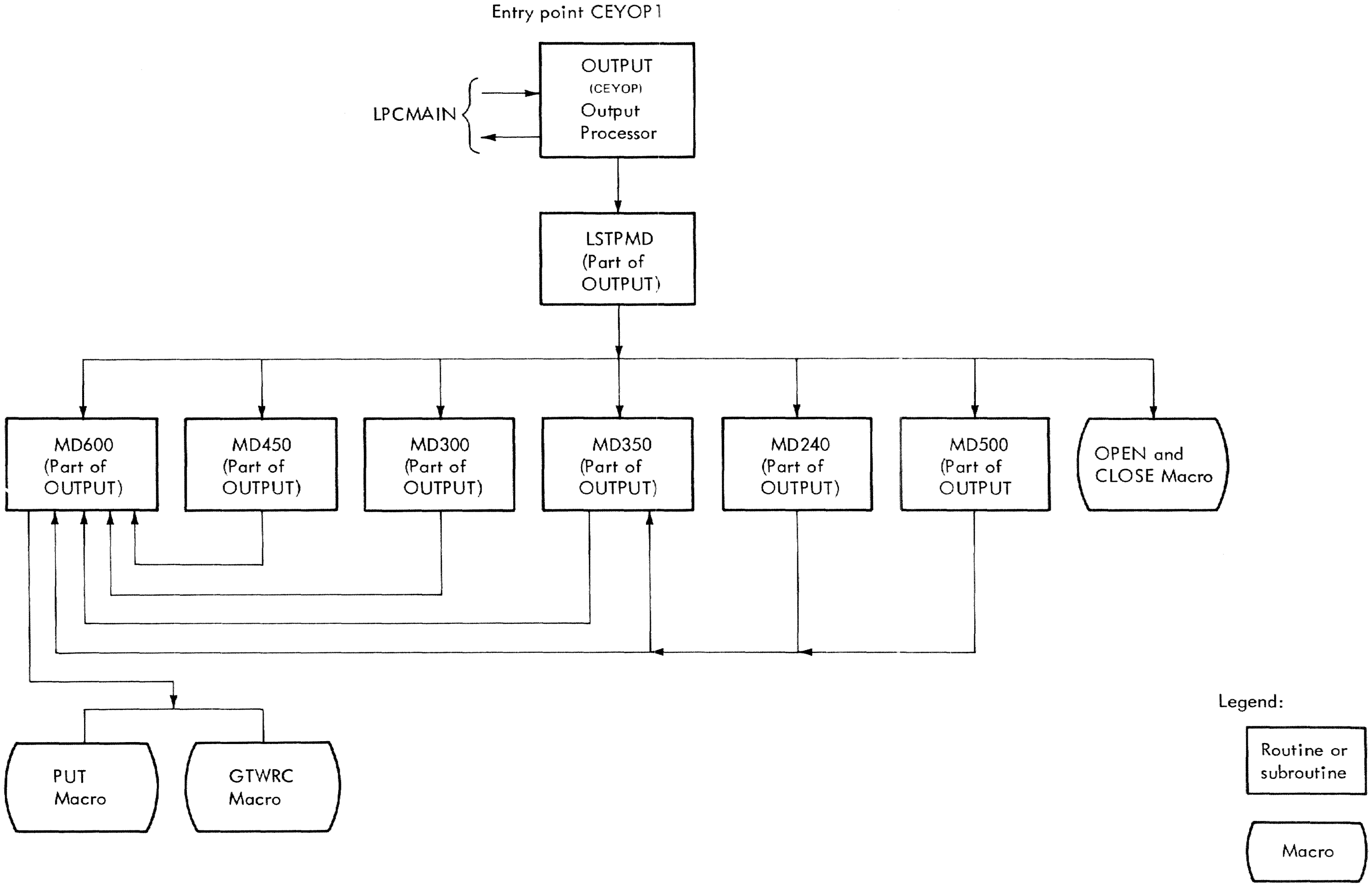


Figure 6. Overview of the output processor

Table 3. Output processing hierarchical table

<table>
<tr><td colspan="4">Routine: Output Processor -- Level: 1</td></tr>
<tr><td>Routine</td><td>Purpose</td><td>Called Routines</td><td>Calling Conditions</td></tr>
<tr><td>Output Processor (CEYOP1)</td><td>Continuation entry point from LPC. Delivers output module, prepares PMD listing.</td><td>LSTPMD (CEYLP)</td><td>If PMD listing is required.</td></tr>
<tr><td colspan="4">Routine: Output Processor -- Level: 2</td></tr>
<tr><td>LSTPMD (CEYLP)</td><td>Prepares PMD listing, place it in a VISAM data set or SYSOUT.</td><td>MD600</td><td>For each line of output.</td></tr>
<tr><td></td><td></td><td>MD450</td><td>When 'NAME' and 'VALUE' lines are to be listed for a DEF table.</td></tr>
<tr><td></td><td></td><td>MD300</td><td>When 'REF#' and 'NAME' lines are to be listed for a REF table.</td></tr>
<tr><td></td><td></td><td>MD350</td><td>When 'LENGTH', 'REF#', 'TYPE' and 'BYTE' lines are to be listed for an RLD table.</td></tr>
<tr><td></td><td></td><td>MD240</td><td>When external or internal REF RLD modifiers are to be listed.</td></tr>
<tr><td></td><td></td><td>MD500</td><td>When Q REFs or CXD REFs are to be listed.</td></tr>
<tr><td></td><td></td><td>OPEN macro instruction</td><td>When listing goes to list data set.</td></tr>
<tr><td></td><td></td><td>CLOSE macro instruction</td><td>When listing goes to list data set.</td></tr>
<tr><td colspan="4">Routine: Output Processor -- Level: 3</td></tr>
<tr><td>MD600</td><td>Places line of PMD listing into VISAM data set or SYSOUT.</td><td>PUT or GTWRC instruction</td><td>PUT: for list data set.
GTWRC: listing to SYSOUT.</td></tr>
<tr><td>MD450</td><td>Writes 'NAME' and 'VALUE' lines for a DEF table in the PMD.</td><td>MD600</td><td>Always called.</td></tr>
<tr><td>MD300</td><td>Writes 'REF#' and 'NAME' lines for a REF table in the PMD.</td><td>MD600</td><td>Always called.</td></tr>
<tr><td>MD350</td><td>Writes 'LENGTH', 'REF#', 'TYPE' and 'BYTE' lines for an RLD table.</td><td>MD600</td><td>Always called.</td></tr>
<tr><td>MD240</td><td>Writes external or internal REF RLD modifiers.</td><td>MD350</td><td>Always called.</td></tr>
<tr><td></td><td></td><td>MD600</td><td>Always called.</td></tr>
<tr><td>MD500</td><td>Writes detail lines for Q REFs and CXD REFs.</td><td>MD600</td><td>Always called.</td></tr>
</table>

OUTPUT -- Output Processor (CEYOP)

This routine delivers the final output module (PMD, text, and ISD), external name list, and the necessary return codes to the LPC. It also prepares a PMD listing, if the user has requested it. (See Chart BA.)

Entry: Entry point CEYOP1; entry parameters:

Register 1
address of a parameter list.

The parameter list consists of a series of address constants aligned on word boundaries:

Word 1 Address of a 1-byte field. The linkage editor fills this field with 00000001 if the list data set (PMD listing) contains lines to be listed; with 00000000, if the list data set is empty.

Word 2 Address of a 1-word field. The linkage editor fills this field with the number of bytes in the PMD.

Word 3 Address of a 1-word field. The linkage editor fills this field with the location of the first byte of the PMD.

Word 4 Address of a 1-word field. The linkage editor fills this field with the number of bytes in the output text.

Word 5 Address of a 1-word field. The linkage editor fills this field with the location of the first byte of the output text.

Word 6 Address of a 1-word field. The linkage editor fills this field with the number of bytes in the ISD.

Word 7 Address of a 1-word field. The linkage editor fills this field with the location of the first byte of the ISD.

Word 8 Address of a 1-word field. The linkage editor fills this field with the location of the first byte of the list of external names.

Calling Sequence: CALL CEYOP1.

Routines Called: LSTPMD, if PMD listing required.

Exit: Normal; exit parameters:

Register 15 contains a condition code, as follows:

Condition code 0
no errors. The field specified by parameter words 1-8 is filled in. If no ISD was produced, the fields specified by words 6 and 7 of the parameter list are zero.

Condition code 4
minor errors, parameter output as for code 0.

Condition code 8
major errors, parameter output as for code 0.

Condition code 12
no object module. Only the field specified by parameter word 1 is filled.

Condition code 16
abnormal end condition; parameter output not currently defined.

Operation: This routine is entered by LPC MAIN at location CEYOP1 with the addresses of locations where the following information is to be stored:

- Lines from the list data set (PMD listing).
- Number of bytes in the PMD.
- Location of first byte in the PMD.
- Number of bytes in the output text.
- Location of first byte in output text.
- Number of bytes in the ISD.
- Location of first byte in the ISD.
- Location of first byte in the external name list.

Initially, a check is made of the form-1 switch (FRM1SW), maintained by INANAL, to determine if at least one form-1 INCLUDE statement has been given. If a form-1 INCLUDE statement has not been given, the "lists exists" indicator is set to zero, the return code is set to 12 (no object module), and control is returned to the LPC. If a form-1 INCLUDE statement has been given and a PMD listing is required, the Program Module Dictionary Listing (LSTPMD) subroutine (described below) is called. LSTPMD is entered at CEYLP via restricted linkage with pointers to the location of the PMD, the length of the PMD, page number to be assigned to the first page of output, and the address of the DCB for the list data set. LSTPMD prepares a PMD listing, places it in the list data set or on SYSOUT, and returns control to this routine.

Before returning control to LPC, this routine prepares a return code and a parameter list for the LPC. If no errors were encountered during linkage editor processing, the return code is set to zero (no errors). If one or more errors were encountered, the return code is set to 8 (major error). The following parameter information is provided: a code specifying whether the list data set contains lines to be listed; the number of bytes in the PMD and the location of the first byte; the number of bytes in the text and the location of the first byte; the number of bytes in the ISD and the location of the first byte (if the ISD is to be generated); and the location of the first byte in the external name list.

LSTPMD -- Program Module Dictionary Listing Subroutine (Chart BB): LSTPMD (CEYLP), a closed subroutine, is entered with INVOKE ACEYLS if the PMD listing option was taken. Information required for the listing is extracted from:

Register 1
address of a five-word list that contains the following information:

Word 1 Location of program module dictionary.

Word 2 Length of PMD.

Word 3 Page of number -1 to be assigned to first page of output.

Word 4 Address of 720-byte work area.

Word 5 Address of the DCB for the list data set.

Upon entry to this subroutine, the information listed above is stored, and a switch (GATESW) tested to determine the destination of the PMD listing (whether to a list data set or to SYSOUT). The output buffer is initialized, and the MD600 subroutine (described below) is entered to write the following module and control section header lines:

- Module Header Lines

  MODULE NAME
  LENGTH
  DIAG SEVERITY

- Control Section Header Lines

  CONTROL SECTION NUMBER
  NAME
  TYPE
  ATTRIBUTES
  VERSION
  CSD LENGTH
  TEXT LENGTH

If relocatable definitions exist for the PMD listing, the MD600 subroutine is entered to write the RELOCATABLE DEFINITIONS header line, and the MD450 subroutine (described below) is entered to list the detail (NAME and VALUE lines). If absolute definitions exist for the PMD listing, the MD600 subroutine is entered to write the ABSOLUTE DEFINITIONS header line, and the MD450 subroutine is entered to list the detail lines. If complex definitions exist for the PMD listing, the MD600 and MD450 subroutines are entered to list the COMPLEX DEFINITIONS header line and the detail lines. If references exist for the PMD listing, the MD600 and MD300 subroutines are entered to list the REFERENCES header line and the detail lines. If Q REFs or a CXD REF exist, they are listed separately from other references; MD600 and MD500 are entered to list header and detail lines.

After all REFs have been listed, LSTPMD determines if the complex DEF RLD contains any modifiers. If so, MD600 is called to write the MODIFIERS FOR COMPLEX DEFs heading. For each modifier pointer with modifiers, MD600 is called to list the header for the corresponding virtual page of the PMD:

PAGE xx # MODIFIERS xxxx

and MD350 is called to list the contents of each modifier:

LENGTH
REF #
TYPE
BYTE

The external REF (including Q REFs and CXD REF) and internal REF RLDs are listed in a similar fashion. The header for each modifier pointer's modifier, however, reads:

TEXT PAGE xx VIRTUAL PAGE xx # MODIFIERS xxxx

The text page is relative to the first page of text produced by the language processor; the virtual page is relative to the first page of virtual storage allocated to the control section when the module is loaded. (Text pages plus "empty" pages - those resulting from DS or ORG instructions and not actually on external storage - equal virtual storage pages.)

After the first control section has been processed, the address of the next control section is obtained and processing continues. When the PMD has been completely processed, the CLOSE macro instruction is used to close the list data set (unless the listing went to SYSOUT) and control is returned to the caller.

MD240 Subroutine (Chart BB): Upon entry, the MD240 subroutine finds the virtual memory page in which the text page for the control section is located. MD240 writes the PMD listing header lines for both the modifiers for text (external and internal refs) entries (text page, virtual page, and # modifiers). After writing the detail header lines, MD600 is entered to list the detail lines applicable to the header lines. MD350 is also entered to complete processing for the PMD listings' modifiers for text (external REFs and internal REFs) entries. The text page count is incremented and the address of the next modifier for text (external REFs or internal REFs) pointer is obtained. If the pointer does not point to an external or internal reference, control is returned to LSTPMD. If the pointer points to an external or internal reference, and there are modifiers for this pointer, remaining pointers for external and internal references are checked in the same manner as the previous pointers. But if there are modifiers for this pointer, they are processed in a manner identical to that for the first internal or external reference modifier.

MD300 Subroutine (Chart BC): The MD300 subroutine is used by LSTPMD to list detail lines for the references entries in the PMD listing. Upon entry, MD300 sets the REF # and NAME heading in their positions, assigns a reference number (beginning with zero) to the entry, and gets the associated name. The MD600 subroutine is entered to write the header and detail lines as formatted, and if all entries have been processed, control is returned to LSTPMD. MD600 is entered to write the lines if the print line is filled, even if all entries have not been processed. After MD600 writes the lines or if the print line is not filled, a reference number is assigned to the next entry, and processing for this is identical to that for the previous entry.

MD350 Subroutine (Chart BC): The MD350 subroutine is used by LSTPMD and MD240, respectively, to list detail lines for the modifiers for complex DEFs and to complete processing for the modifiers associated with text (external REFs and internal REFs) pointers. Initially, the LENGTH and REF # heading is set into position and the reference number associated with the present modifier entry is obtained. After the reference number for all entries is obtained, MD600 is entered to write the lines as currently formatted. The TYPE and BYTE heading are set into position, and the "type" and "byte" information for the first entry, whose length and number were just written, is obtained. The type and byte information for the remaining entries is obtained, and MD600 is entered to write the lines as formatted. When all modifiers are processed, control is returned to the calling subroutine. (If all modifiers have not been processed, MD350 starts at its beginning to provide the same processing for the subsequent modifier.)

MD450 Subroutine (Chart BD): The MD450 subroutine is used by LSTPMD to list detail lines for relocatable, absolute, and complex definitions entries in the PMD listing. The NAME and VALUE heading are set in position, and the name and value for the associated entry is placed into the print area. When all entries have been processed, MD600 is entered to write the lines as currently formatted, after which control is returned to LSTPMD. MD600 is entered to write the lines if the print line is filled, even if all entries have not been processed.

MD500 Subroutine (Chart BD): This subroutine is used by LSTPMD to list detail lines of Q REF and CXD REF types of external reference; these are listed separately from other references in the PMD listing. The REF number and name is set into position for each Q REF or CXD REF; then a call is made to MD600 to write the formatted lines. Next, length and align values are set up and MD600 is called to write these lines. Control is then returned to LSTPMD.

MD600 Subroutine (Chart BE): The MD600 subroutine is used by LSTPMD, MD240, MD300, MD350, MD450, and MD500, to list detail lines for entries in the PMD listing. Initially, the detail line is checked to determine whether it is being ejected; if so, either a PUT (to a list data set) or GTWRC (to SYSOUT) macro instruction is used to set and write the page header. The line count is then reset. Another test is made to determine the listing destination, whether list data set or SYSOUT, and an appropriate macro instruction (PUT or GTWRC) is issued to write the current line. The line count is incremented, and control is returned to the calling subroutine.

EARLY-END PROCESSOR

Function Summary

The Early-End Processor phase of the linkage editor, entered at CEYEE1 from LPCMAIN, is called as a result of early termination of linkage editor processing in the Control Statement Processor phase. The Early-End Processor consists of one routine, EARLY END; this routine closes the list data set if open, closes any libraries that may be open, and frees virtual storage secured previously for work areas. Figure 7 and Table 4 illustrate the Early-End Processor's general operation and hierarchy, respectively.

EARLY END -- Early-End Processor (CEYEE1)

This routine performs all activities required if linkage editor processing must be terminated before normal completion of a task. (See Chart CA.)

Entry: Entry point CEYEE1; entry parameters:

Register 1
address of a one-byte field. EARLY END will fill this field with X'01' if the list data set contains lines to be listed; with X'00', if the list data set is empty.

Calling Sequence: CALL CEYEE1.

Routines Called: Via CLOSE and FREEMAIN macros.

Exit: Normal; exit parameters:

Register 15

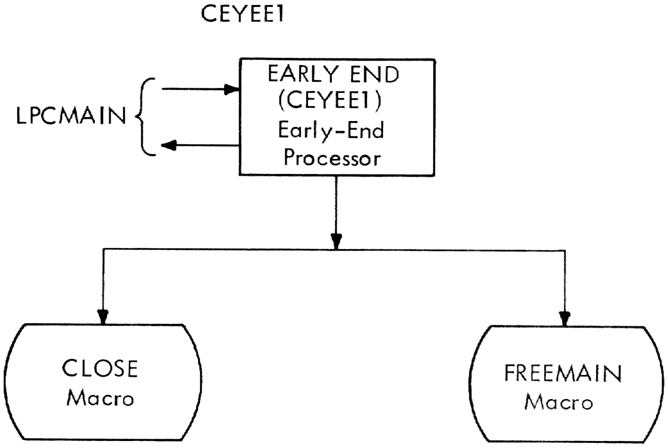


Figure 7. Overview of the early-end processor

End Code 0
normal end; the field specified by register 1 is completed.

Operation: Upon entry, a check is made to determine whether the list data set contains lines (PMD listing) to be listed; if lines are to be listed, the field whose address has been passed in register 1, is set to one, and the CLOSE macro instruction is used to close the list data set if it was opened. If the list data set contains no lines to be listed, the field is set to zero. A check is made to determine whether any library is still open; if so, the CLOSE macro instruction is again used to close the open library. If no lines are to be listed, or no library is open, or after CLOSE closes the open library, the FREEMAIN macro instruction is used to release all work areas, and control is returned to LPC.

Table 4. Early-End processing hierarchical table

Routine: Early End -- Level: 1			
Routine	Purpose	Called Routines	Calling Conditions
Early End (CEYEE1)	Early-end entry point from LPC. Terminates linkage editor processing prior to completion.	CLOSE macro instruction	Always called to close opened libraries.
		FREEMAIN macro instruction	Always called to free virtual storage areas.

The flowcharts in this manual have been produced by an IBM program, using ANSI symbols. The symbols are defined in the left column below, and examples of their use are shown at the right.

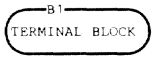
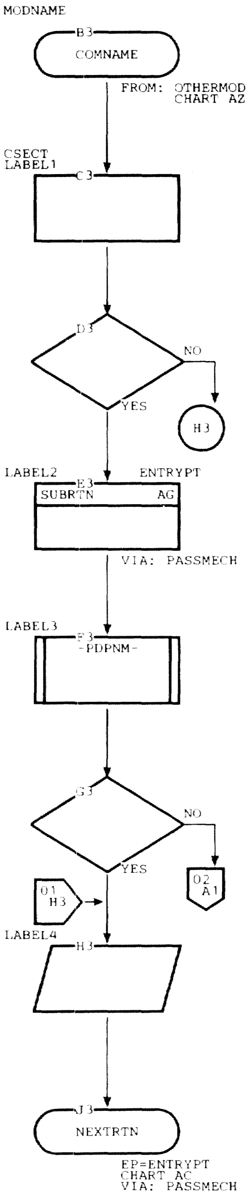
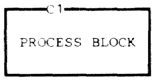
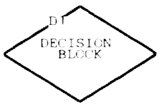
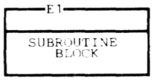
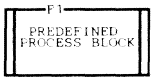
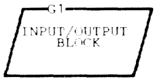
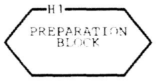
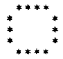
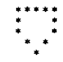
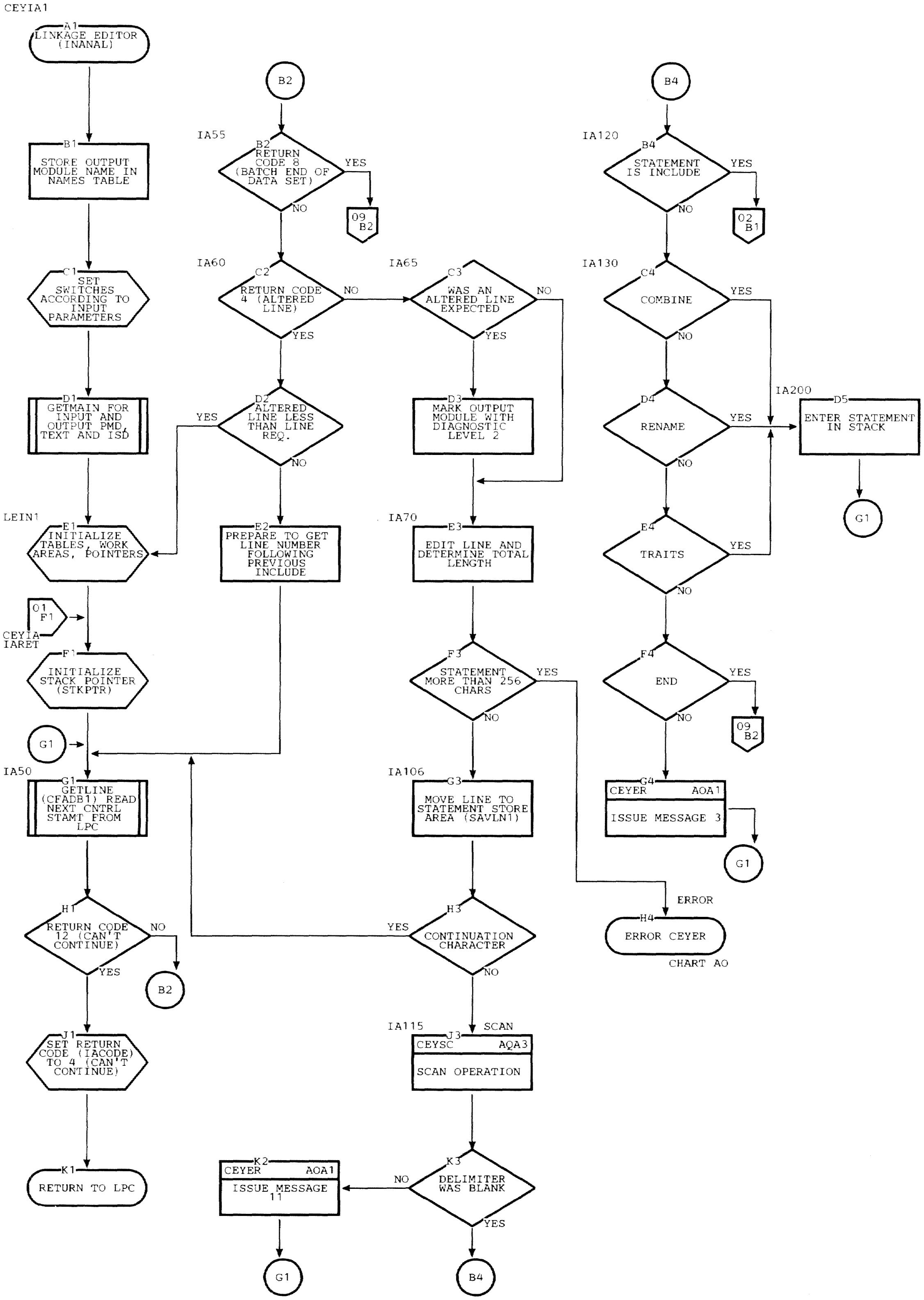
SYMBOL	DEFINITION	EXAMPLE	COMMENTS
	INDICATES AN ENTRY OR TERMINAL POINT IN A FLOWCHART; SHOWS START, STOP, HALT, DELAY, OR INTERRUPTION. MAY ALSO INDICATE RETURN TO THE CALLING PROGRAM.	 MODNAME B3 COMNAME FROM: OTHERMOD CHART AZ CSECT LABEL1 C3 D3 NO YES H3 LABEL2 E3 ENTRYPT SUBRTN AG VIA: PASSMECH LABEL3 F3 -PDPNM- G3 NO YES 01 H3 02 A1 LABEL4 H3 J3 NEXTRTN EP=ENTRYPT CHART AC VIA: PASSMECH	B3: MODNAME IS THE LOAD MODULE OR LIBRARY NAME OF THE ROUTINE DESCRIBED BY THIS FLOWCHART. COMNAME IS THE COMMON NAME OF THE ROUTINE. OTHERMOD INDICATES THE MODULES PASSING CONTROL TO THIS MODULE AND THEIR FLOWCHARTS.
	INDICATES A PROCESSING FUNCTION OR A DEFINED OPERATION CAUSING CHANGE IN VALUE, FORM OR LOCATION OF INFORMATION.		C3: CSECT IS THE CSECT NAME OR OTHER ENTRY POINT AT WHICH PROCESSING BEGINS. LABEL1 IS THE LABEL OF THE FIRST INSTRUCTION.
	INDICATES A DECISION OR SWITCHING-TYPE OPERATION THAT DETERMINES WHICH OF A NUMBER OF ALTERNATE PATHS SHOULD BE FOLLOWED.		D3: PROGRAM EXECUTION CONTINUES WITH BLOCK H3 WHEN THE DECISION IS NO, OR BLOCK E3 WHEN THE DECISON IS YES.
	INDICATES A SUBROUTINE OR MODULE THAT IS DESCRIBED IN THIS MANUAL		E3: LABEL2 IS THE LABEL OF THE SECTION OF CODE IN THIS ROUTINE FROM WHICH CONTROL IS PASSED TO THE SUBROUTINE. CONTROL RETURNS TO THE NEXT INSTRUCTION FOLLOWING THE SUBROUTINE CALL. ENTRYPT IS THE ENTRY POINT. SUBRTN IS THE COMMON NAME OF THE SUBROUTINE IN FLOWCHART AG. VIA: PASSMECH INDICATES HOW CONTROL PASSES FROM COMNAME TO SUBRTN.
	INDICATES A SUBROUTINE OR MODULE THAT IS INCLUDED IN THE FLOWCHARTS OF ANOTHER MANUAL.		F3: LABEL3 IS THE LABEL OF THE SECTION OF CODE FROM WHICH CONTROL IS PASSED TO THE PREDEFINED PROCESS PDPNM, WHICH IS DOCUMENTED IN ANOTHER PUBLICATION (-PDPNM- MAY ALSO BE USED IN A PROCESSING BLOCK).
	INDICATES GENERAL I/O FUNCTIONS, SUCH AS GET, PUT, READ, WRITE, SIO, AND DEVICE-CONTROL MACRO INSTRUCTIONS.		G3: EXECUTION CONTINUES WITH BLOCK H3 WHEN THE DECISION IS YES, OR WITH BLOCK A1 ON PAGE 2 OF THIS SET OF FLOWCHARTS WHEN THE DECISION IS NO. THE OFFPAGE CONNECTOR MARKED 01H3 INDICATES THAT EXECUTION CONTINUES WITH BLOCK H3 FROM ANOTHER PAGE OF THIS SET OF FLOWCHARTS. THIS CONNECTOR IS ALSO PAIRED WITH THE ONPAGE CONNECTOR FROM BLOCK D3.
	INDICATES A PROCESS THAT CHANGES SYSTEM OPERATION, FOR EXAMPLE, SETS A SWITCH, MODIFIES AN INDEX REGISTER, OR INITIALIZES A ROUTINE.		H3: LABEL4 IS THE LABEL OF A SECTION OF CODE OF THIS ROUTINE THAT INITIATES I/O.
ONPAGE CONNECTOR 	INDICATES ENTRY TO OR EXIT FROM ANOTHER BLOCK ON THE SAME FLOWCHART PAGE.		J3: NEXTRTN IS THE COMMON NAME OF THE ROUTINE THAT EXECUTES AFTER THIS ROUTINE. ENTRYPT IS THE ENTRY POINT OF NEXTRTN, WHICH IS DESCRIBED IN CHART AC. VIA: PASSMECH INDICATES HOW CONTROL PASSES FROM COMNAME TO NEXTRTN.
OFFPAGE CONNECTOR 	INDICATES ENTRY TO OR EXIT FROM A BLOCK ON ANOTHER PAGE OF THE SAME SET OF FLOWCHARTS.		

Chart AA. Linkage Editor: INANAL - Control Statement Input/Analyze processor (page 1 of 9)



CEYIA1
A1 LINKAGE EDITOR (INANAL)
B1 STORE OUTPUT MODULE NAME IN NAMES TABLE
C1 SET SWITCHES ACCORDING TO INPUT PARAMETERS
D1 GETMAIN FOR INPUT AND OUTPUT PMD, TEXT AND ISD
LEIN1
E1 INITIALIZE TABLES, WORK AREAS, POINTERS
01 F1
CEYIA IARET
F1 INITIALIZE STACK POINTER (STKPTR)
G1
IA50
G1 GETLINE (CFADB1) READ NEXT CNTRL STAMT FROM LPC
H1 RETURN CODE 12 (CAN'T CONTINUE)
NO
YES
B2
J1 SET RETURN CODE (IACODE) TO 4 (CAN'T CONTINUE)
K1 RETURN TO LPC
B2
IA55
B2 RETURN CODE 8 (BATCH END OF DATA SET)
YES
NO
09 B2
IA60
C2 RETURN CODE 4 (ALTERED LINE)
NO
YES
D2 ALTERED LINE LESS THAN LINE REQ.
YES
NO
E2 PREPARE TO GET LINE NUMBER FOLLOWING PREVIOUS INCLUDE
K2 CEYER AOA1
ISSUE MESSAGE 11
G1
IA65
C3 WAS AN ALTERED LINE EXPECTED
NO
YES
D3 MARK OUTPUT MODULE WITH DIAGNOSTIC LEVEL 2
IA70
E3 EDIT LINE AND DETERMINE TOTAL LENGTH
F3 STATEMENT MORE THAN 256 CHARS
YES
NO
IA106
G3 MOVE LINE TO STATEMENT STORE AREA (SAVLN1)
H3 CONTINUATION CHARACTER
YES
NO
IA115
SCAN
J3 CEYSC AQA3
SCAN OPERATION
K3 DELIMITER WAS BLANK
NO
YES
B4
B4
IA120
B4 STATEMENT IS INCLUDE
YES
NO
02 B1
IA130
C4 COMBINE
YES
NO
D4 RENAME
YES
NO
E4 TRAITS
YES
NO
F4 END
YES
NO
09 B2
G4 CEYER AOA1
ISSUE MESSAGE 3
G1
ERROR
H4 ERROR CEYER
CHART AO
IA200
D5 ENTER STATEMENT IN STACK
G1

Chart AA. Linkage Editor: INCLUDE - INCLUDE statement processor (page 2 of 9)

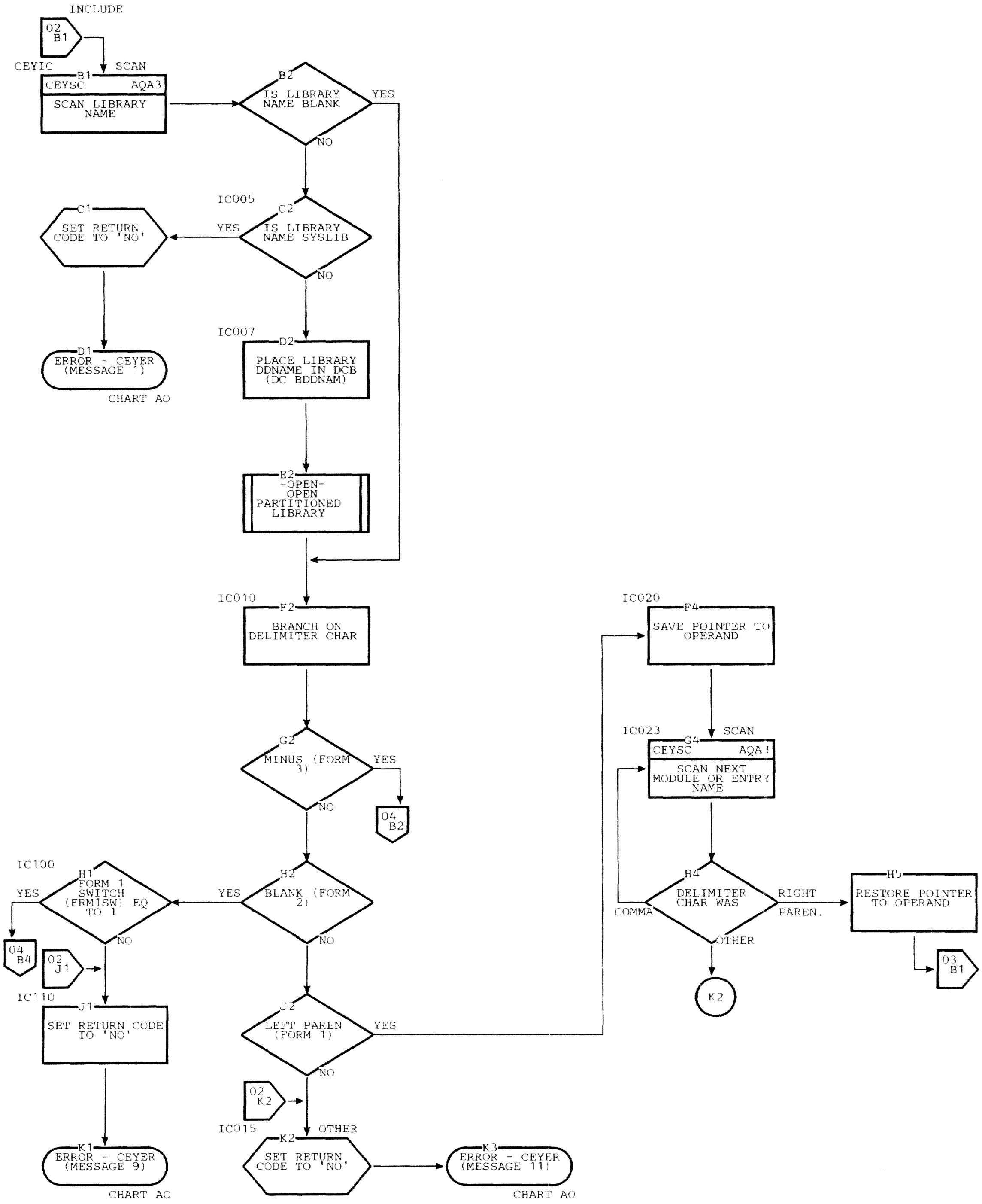
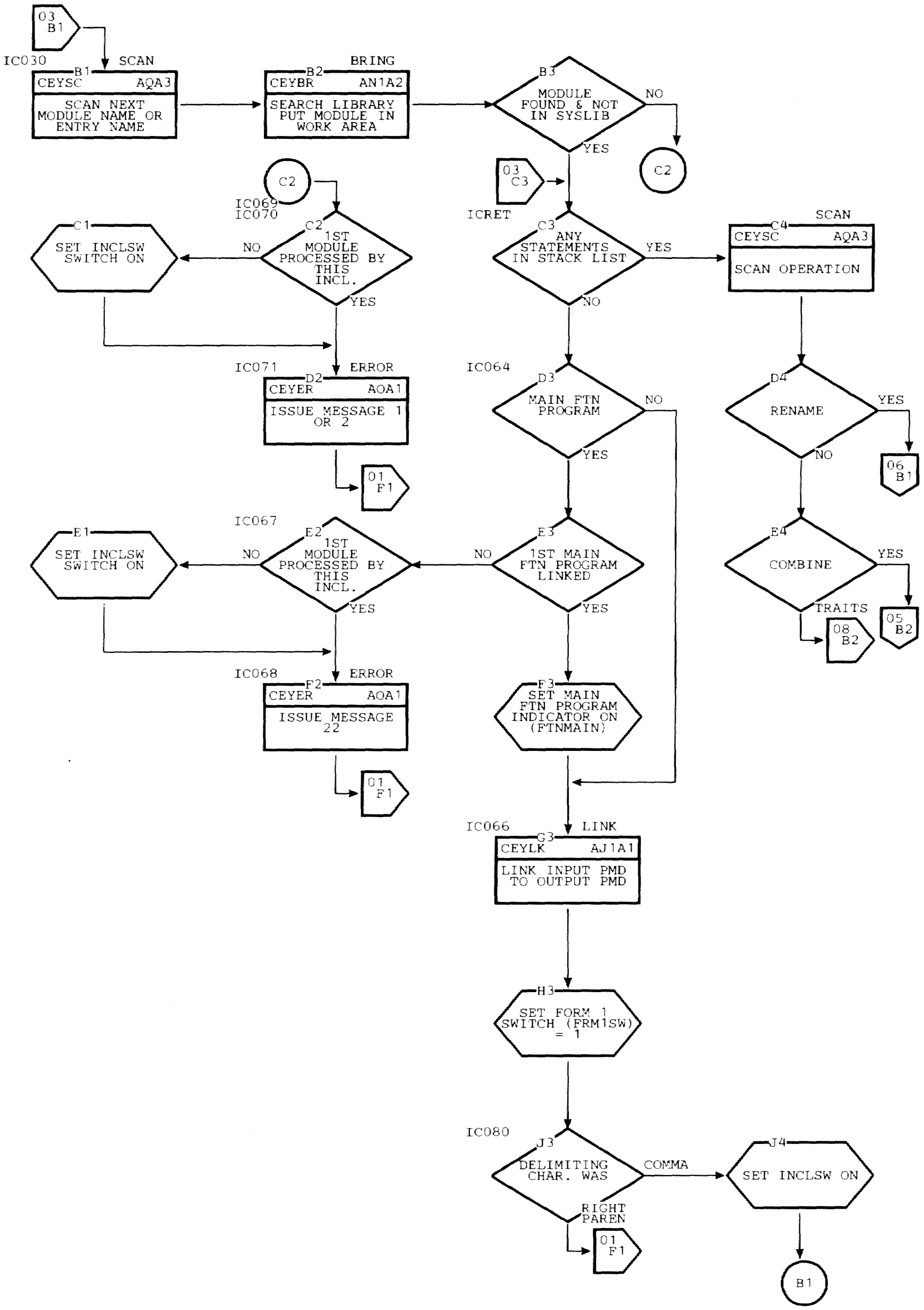


Chart AA. Linkage Editor: INCLUDE - INCLUDE statement processor (page 3 of 9)



NOTE: J4. #INDICATES MORE THAN ONE MODULE LINKED BY THIS INCLUDE

Chart AA. Linkage Editor: INCLUDE - INCLUDE statement processor (page 4 of 9)

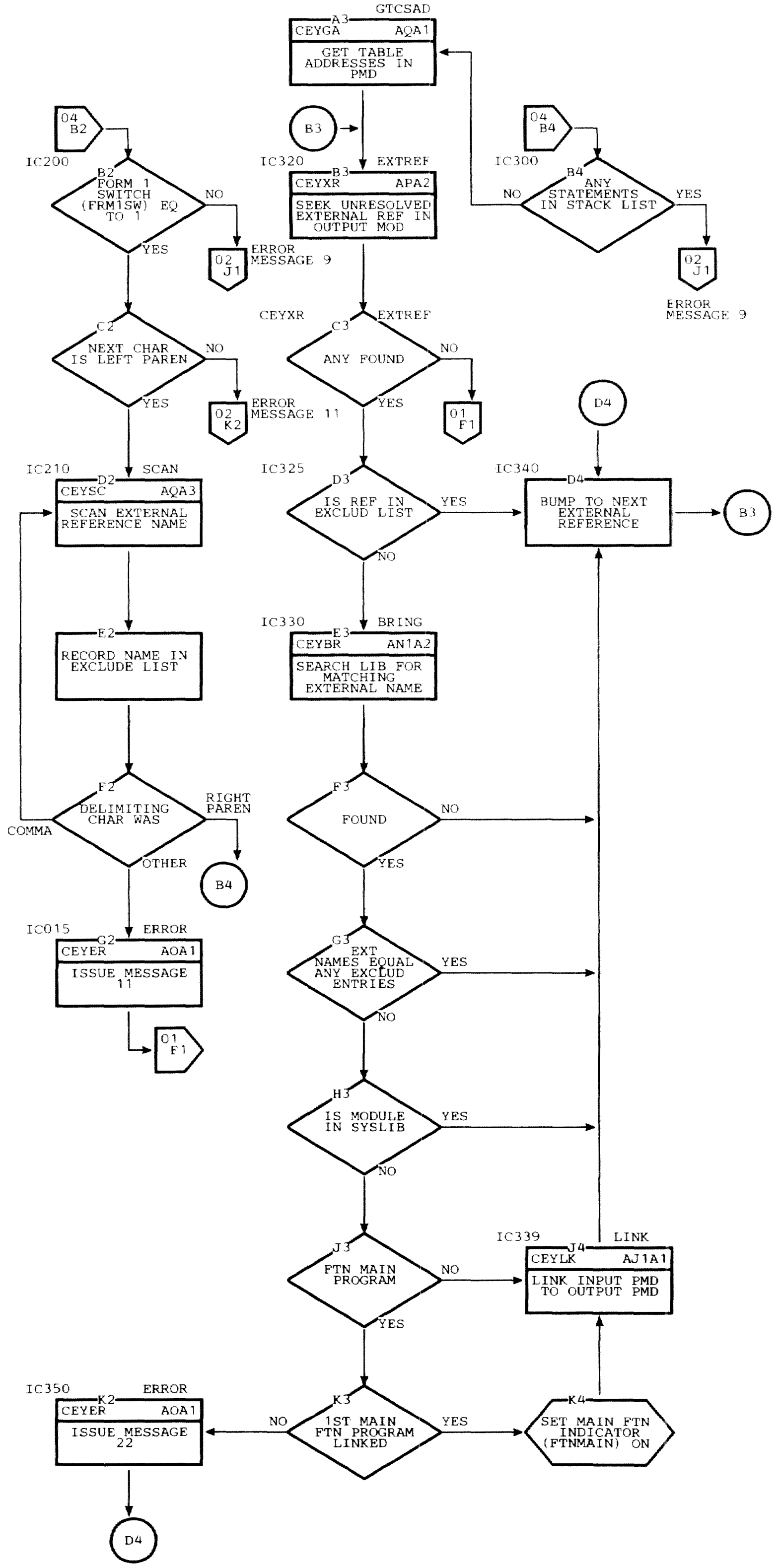


Chart AA. Linkage Editor: COMBINE - COMBINE statement processor (page 5 of 9)

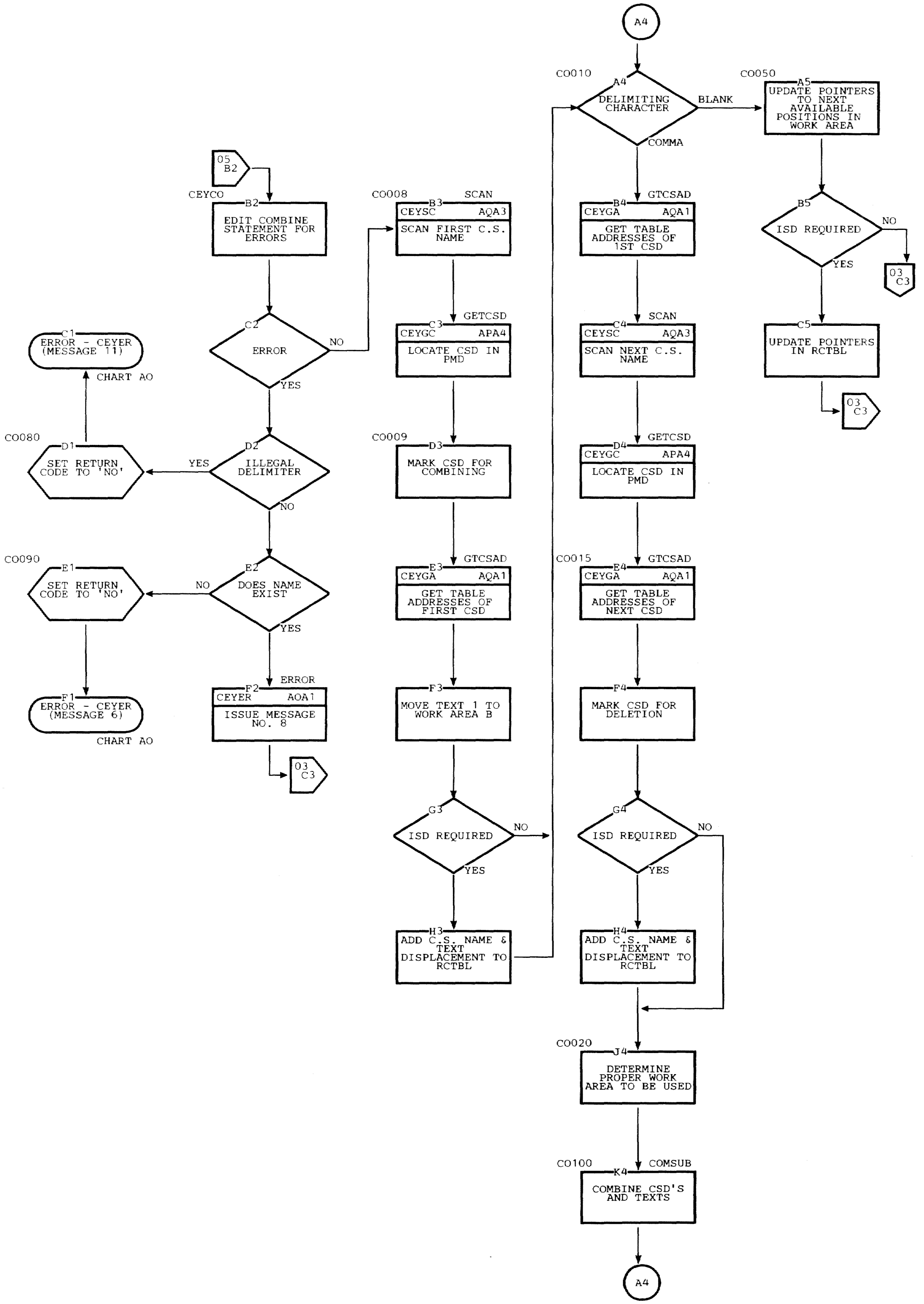


Chart AA. Linkage Editor: RENAME/DELNAME - RENAME statement processor and Delete Entry Name subroutine (page 6 of 9)

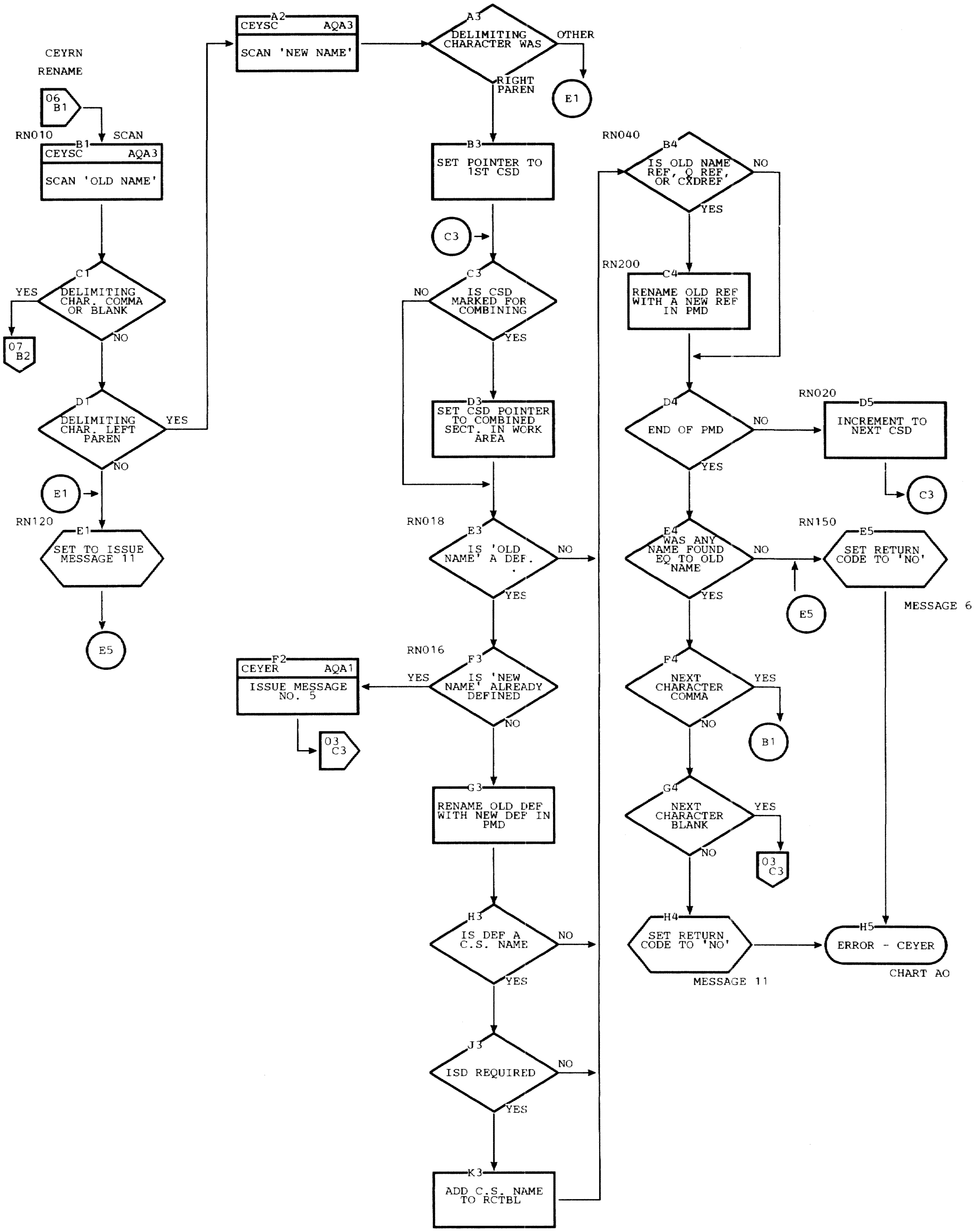
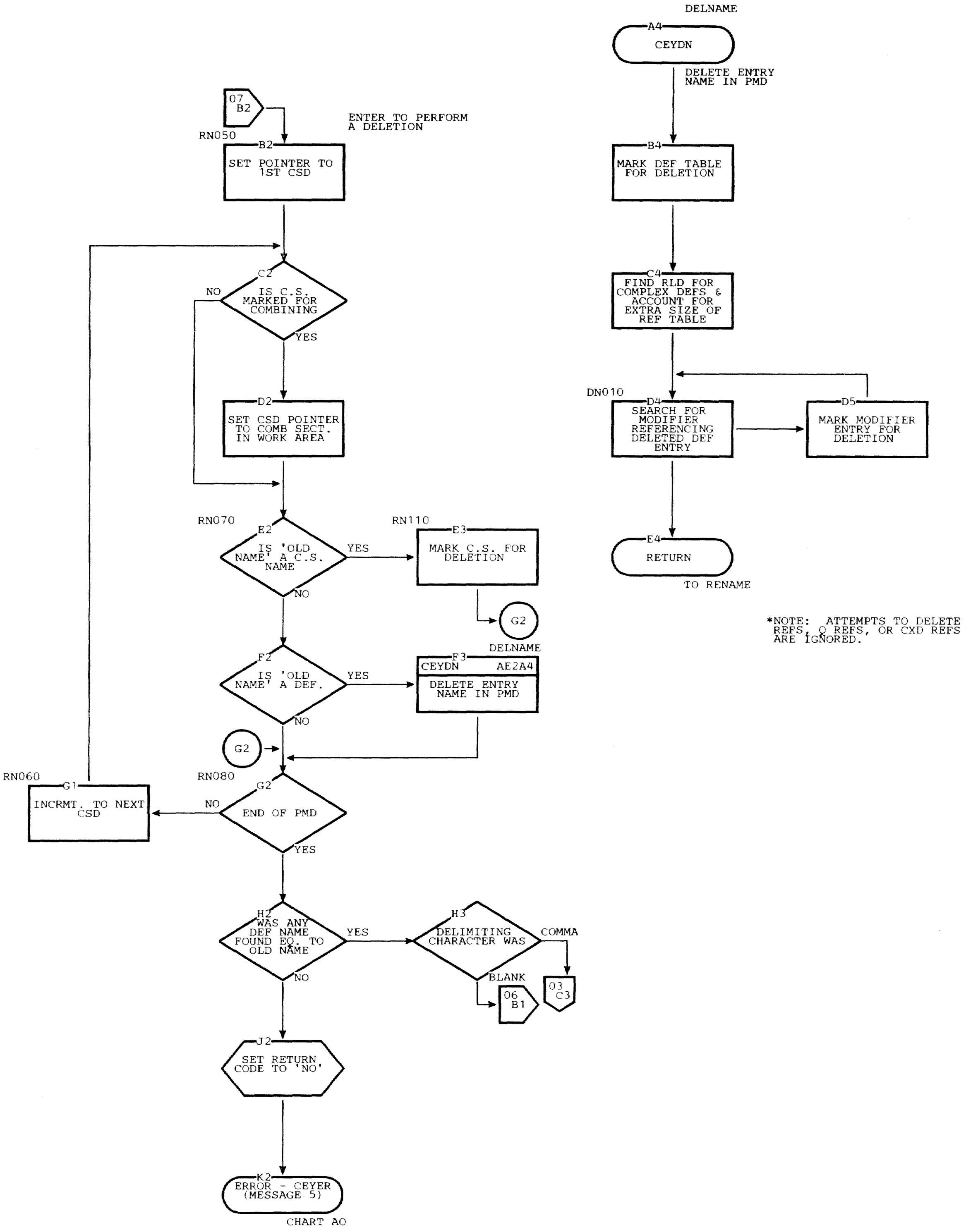


Chart AA. Linkage Editor: RENAME/DELNAME - RENAME statement processor and Delete Entry Name subroutine (page 7 of 9)



*NOTE: ATTEMPTS TO DELETE REFS, Q REFS, OR CXD REFS ARE IGNORED.

Chart AA. Linkage Editor: TRAITS - TRAITS statement processor (page 8 of 9)

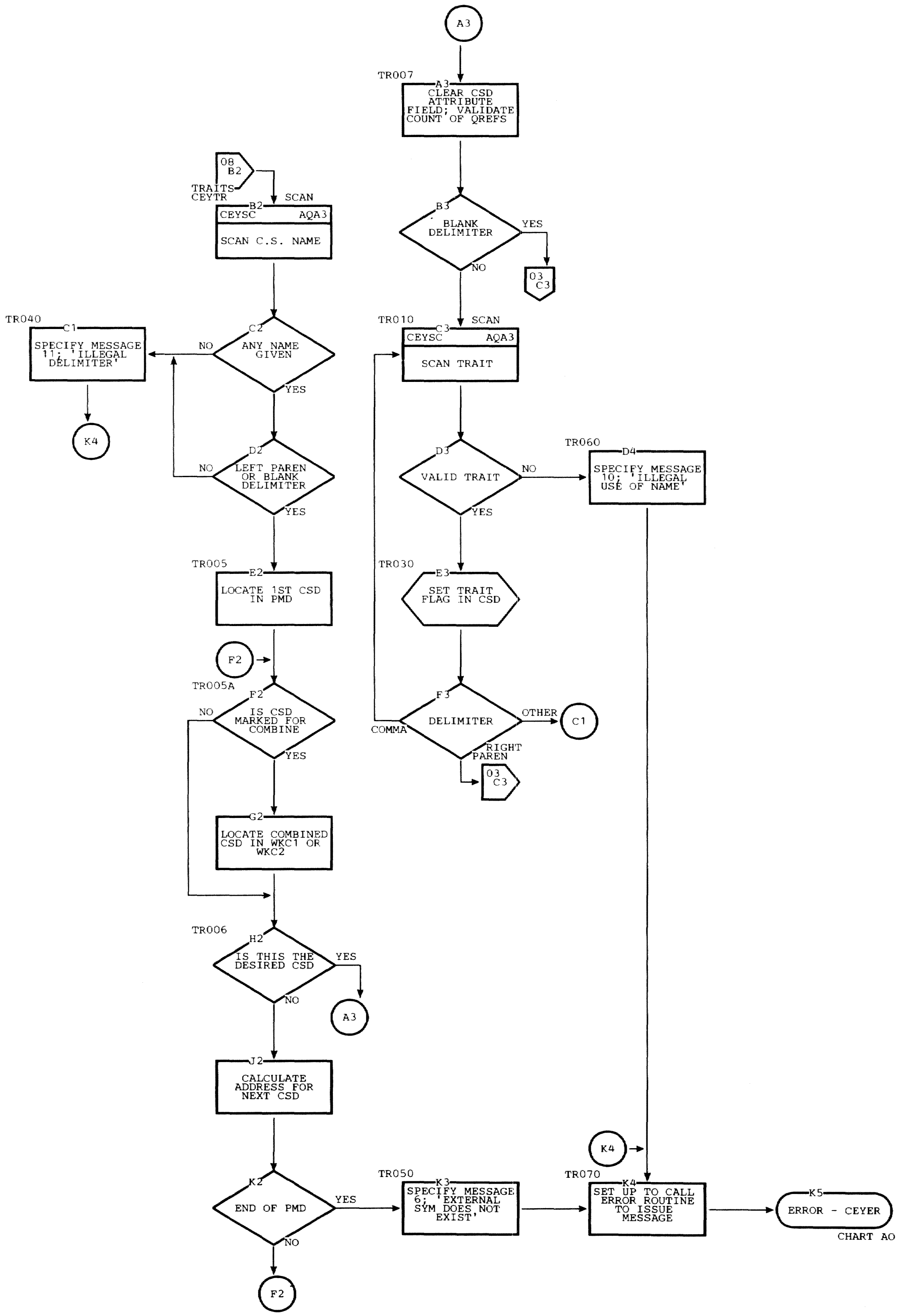


Chart AA. Linkage Editor: END - END statement processor (page 9 of 9)

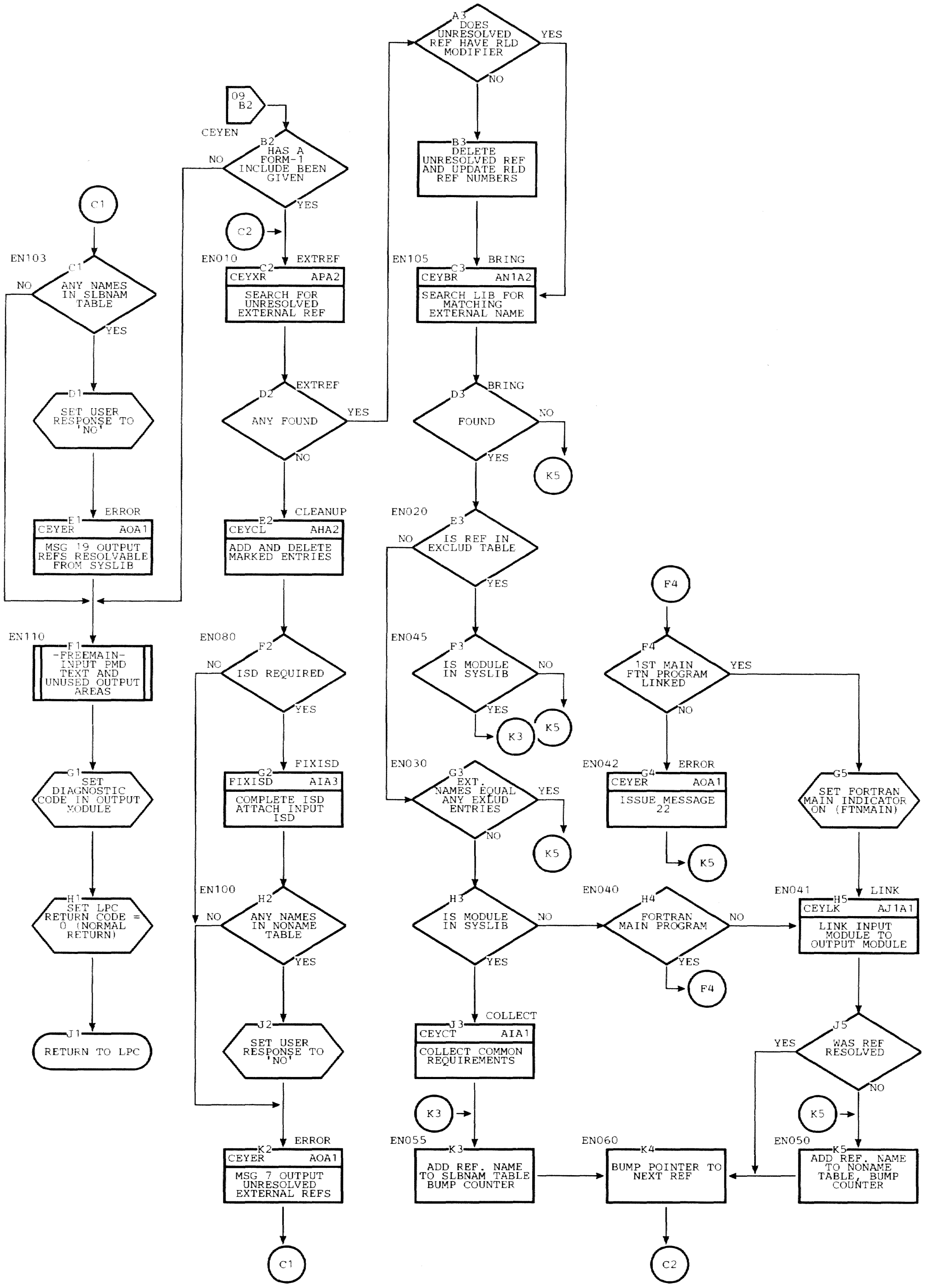
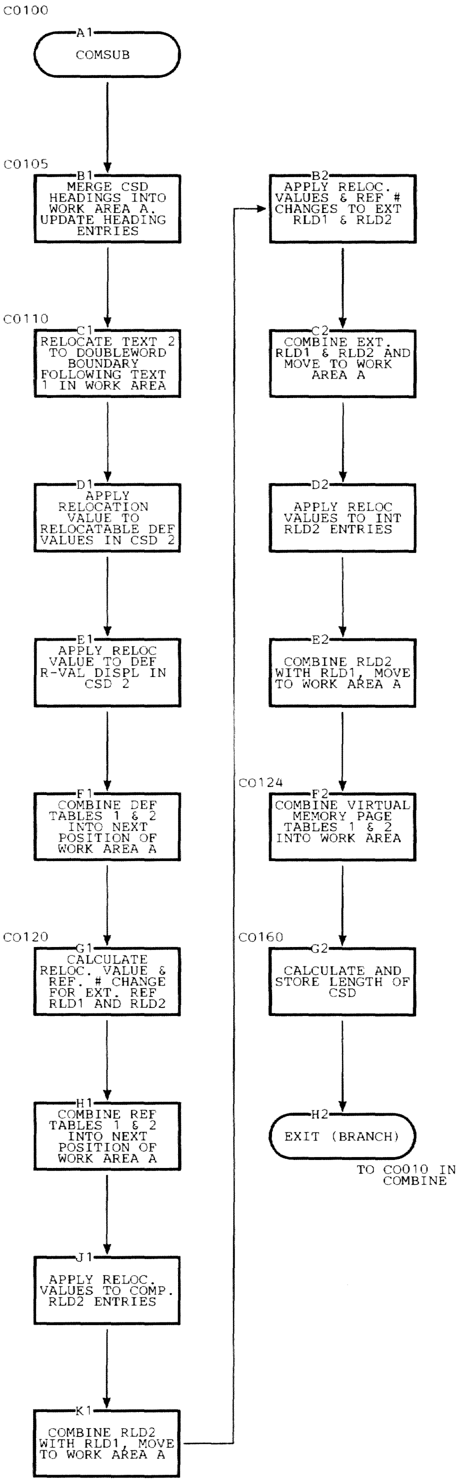

A3 DOES UNRESOLVED REF HAVE RLD MODIFIER
YES
NO
B3 DELETE UNRESOLVED REF AND UPDATE RLD REF NUMBERS
EN105
C3 BRING
CEYBR AN1A2
SEARCH LIB FOR MATCHING EXTERNAL NAME
D3 BRING FOUND
NO
YES
K5
09 B2
CEYEN
B2 HAS A FORM-1 INCLUDE BEEN GIVEN
NO
YES
C2
EN010
C2 EXTREF
CEYXR APA2
SEARCH FOR UNRESOLVED EXTERNAL REF
D2 EXTREF ANY FOUND
YES
NO
C1
EN103
C1 ANY NAMES IN SLBNAM TABLE
NO
YES
D1 SET USER RESPONSE TO 'NO'
E1 ERROR
CEYER AOA1
MSG 19 OUTPUT REFS RESOLVABLE FROM SYSLIB
EN110
F1 -FREEMAIN- INPUT PMD TEXT AND UNUSED OUTPUT AREAS
G1 SET DIAGNOSTIC CODE IN OUTPUT MODULE
H1 SET LPC RETURN CODE = 0 (NORMAL RETURN)
J1 RETURN TO LPC
E2 CLEANUP
CEYCL AHA2
ADD AND DELETE MARKED ENTRIES
EN080
F2 ISD REQUIRED
NO
YES
G2 FIXISD
FIXISD AIA3
COMPLETE ISD ATTACH INPUT ISD
EN100
H2 ANY NAMES IN NONAME TABLE
NO
YES
J2 SET USER RESPONSE TO 'NO'
K2 ERROR
CEYER AOA1
MSG 7 OUTPUT UNRESOLVED EXTERNAL REFS
C1
EN020
E3 IS REF IN EXCLUD TABLE
NO
YES
EN045
F3 IS MODULE IN SYSLIB
NO
YES
K3
K5
EN030
G3 EXT. NAMES EQUAL ANY EXLUD ENTRIES
YES
NO
K5
H3 IS MODULE IN SYSLIB
NO
YES
J3 COLLECT
CEYCT AIA1
COLLECT COMMON REQUIREMENTS
K3
EN055
K3 ADD REF. NAME TO SLBNAM TABLE BUMP COUNTER
F4
F4 1ST MAIN FTN PROGRAM LINKED
YES
NO
EN042
G4 ERROR
CEYER AOA1
ISSUE MESSAGE 22
K5
EN040
H4 FORTRAN MAIN PROGRAM
NO
YES
F4
EN060
K4 BUMP POINTER TO NEXT REF
C2
G5 SET FORTRAN MAIN INDICATOR ON (FTNMAIN)
EN041
H5 LINK
CEYLK AJ1A1
LINK INPUT MODULE TO OUTPUT MODULE
J5 WAS REF RESOLVED
YES
NO
K5
EN050
K5 ADD REF. NAME TO NONAME TABLE, BUMP COUNTER

Chart AD. COMSUB - Combine Control Section subroutine



C0100
A1
COMSUB
C0105
B1
MERGE CSD HEADINGS INTO WORK AREA A. UPDATE HEADING ENTRIES
C0110
C1
RELOCATE TEXT 2 TO DOUBLEWORD BOUNDARY FOLLOWING TEXT 1 IN WORK AREA
D1
APPLY RELOCATION VALUE TO RELOCATABLE DEF VALUES IN CSD 2
E1
APPLY RELOC VALUE TO DEF R-VAL DISPL IN CSD 2
F1
COMBINE DEF TABLES 1 & 2 INTO NEXT POSITION OF WORK AREA A
C0120
G1
CALCULATE RELOC. VALUE & REF. # CHANGE FOR EXT. REF RLD1 AND RLD2
H1
COMBINE REF TABLES 1 & 2 INTO NEXT POSITION OF WORK AREA A
J1
APPLY RELOC. VALUES TO COMP. RLD2 ENTRIES
K1
COMBINE RLD2 WITH RLD1, MOVE TO WORK AREA A
B2
APPLY RELOC. VALUES & REF # CHANGES TO EXT RLD1 & RLD2
C2
COMBINE EXT. RLD1 & RLD2 AND MOVE TO WORK AREA A
D2
APPLY RELOC VALUES TO INT RLD2 ENTRIES
E2
COMBINE RLD2 WITH RLD1, MOVE TO WORK AREA A
C0124
F2
COMBINE VIRTUAL MEMORY PAGE TABLES 1 & 2 INTO WORK AREA
C0160
G2
CALCULATE AND STORE LENGTH OF CSD
H2
EXIT (BRANCH)
TO C0010 IN COMBINE

Chart AH. CLEANUP - Cleanup Final Module subroutine

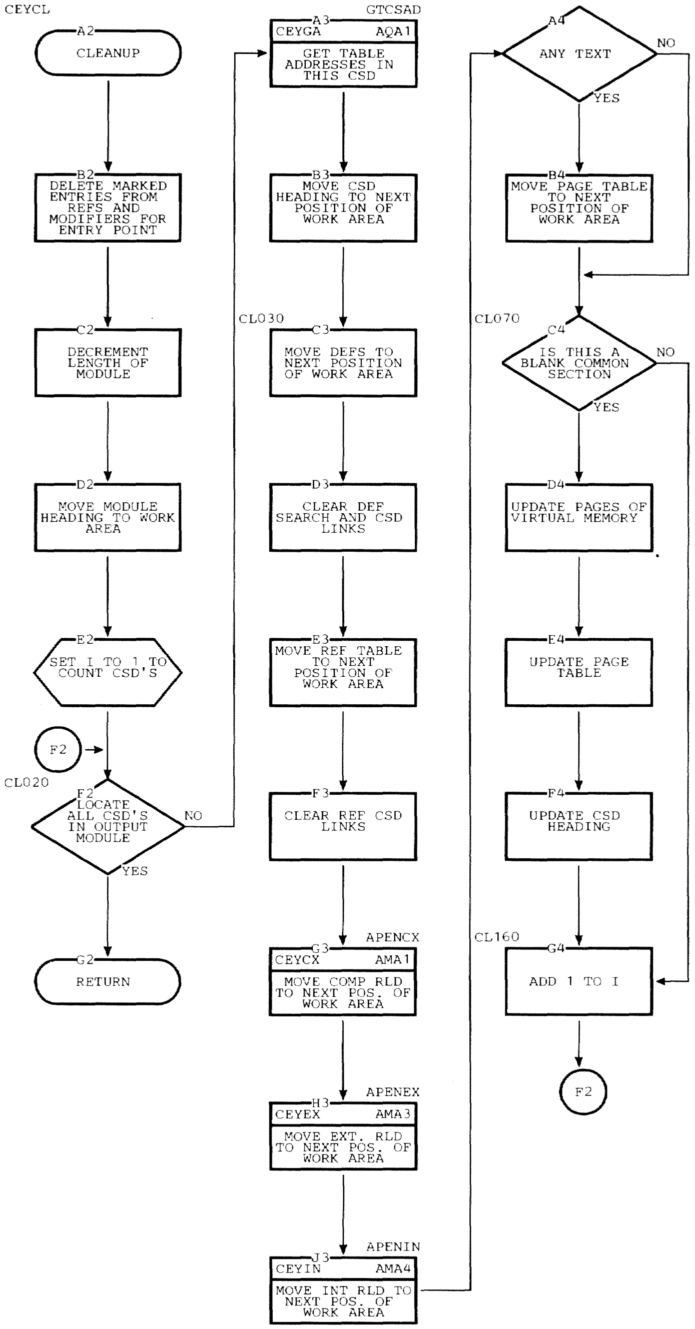

CEYCL
A2
CLEANUP
B2
DELETE MARKED ENTRIES FROM REFS AND MODIFIERS FOR ENTRY POINT
C2
DECREMENT LENGTH OF MODULE
D2
MOVE MODULE HEADING TO WORK AREA
E2
SET I TO 1 TO COUNT CSD'S
F2
CL020
F2
LOCATE ALL CSD'S IN OUTPUT MODULE
NO
YES
G2
RETURN
GTCSAD
A3
CEYGA
AQA1
GET TABLE ADDRESSES IN THIS CSD
B3
MOVE CSD HEADING TO NEXT POSITION OF WORK AREA
CL030
C3
MOVE DEFS TO NEXT POSITION OF WORK AREA
D3
CLEAR DEF SEARCH AND CSD LINKS
E3
MOVE REF TABLE TO NEXT POSITION OF WORK AREA
F3
CLEAR REF CSD LINKS
APENCX
G3
CEYCX
AMA1
MOVE COMP RLD TO NEXT POS. OF WORK AREA
APENEX
H3
CEYEX
AMA3
MOVE EXT. RLD TO NEXT POS. OF WORK AREA
APENIN
J3
CEYIN
AMA4
MOVE INT RLD TO NEXT POS. OF WORK AREA
A4
ANY TEXT
NO
YES
B4
MOVE PAGE TABLE TO NEXT POSITION OF WORK AREA
CL070
C4
IS THIS A BLANK COMMON SECTION
NO
YES
D4
UPDATE PAGES OF VIRTUAL MEMORY
E4
UPDATE PAGE TABLE
F4
UPDATE CSD HEADING
CL160
G4
ADD 1 TO I
F2

Chart AI. COLLECT - Collect Common Requirements subroutine and FIXISD - Fix ISD subroutine

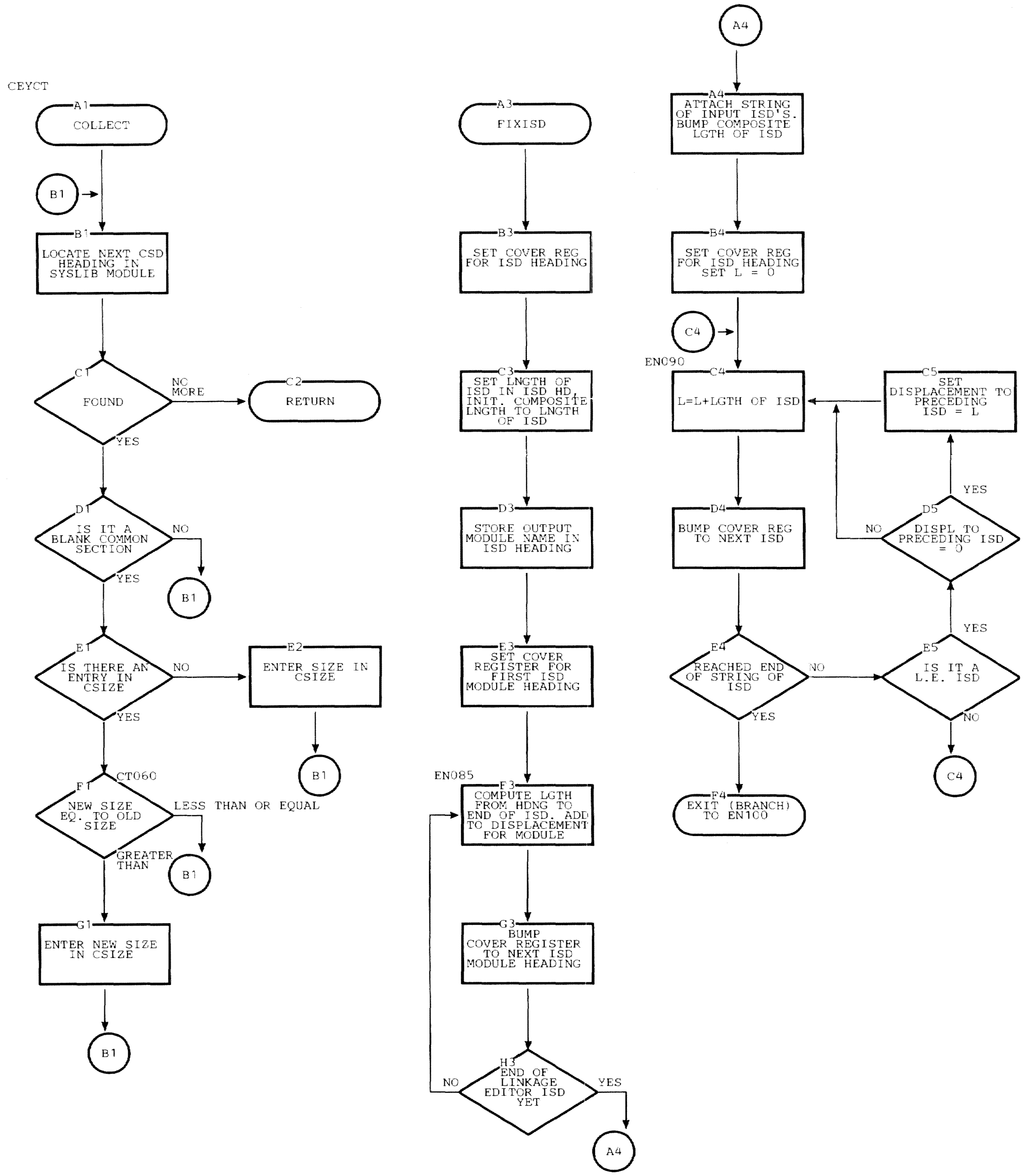
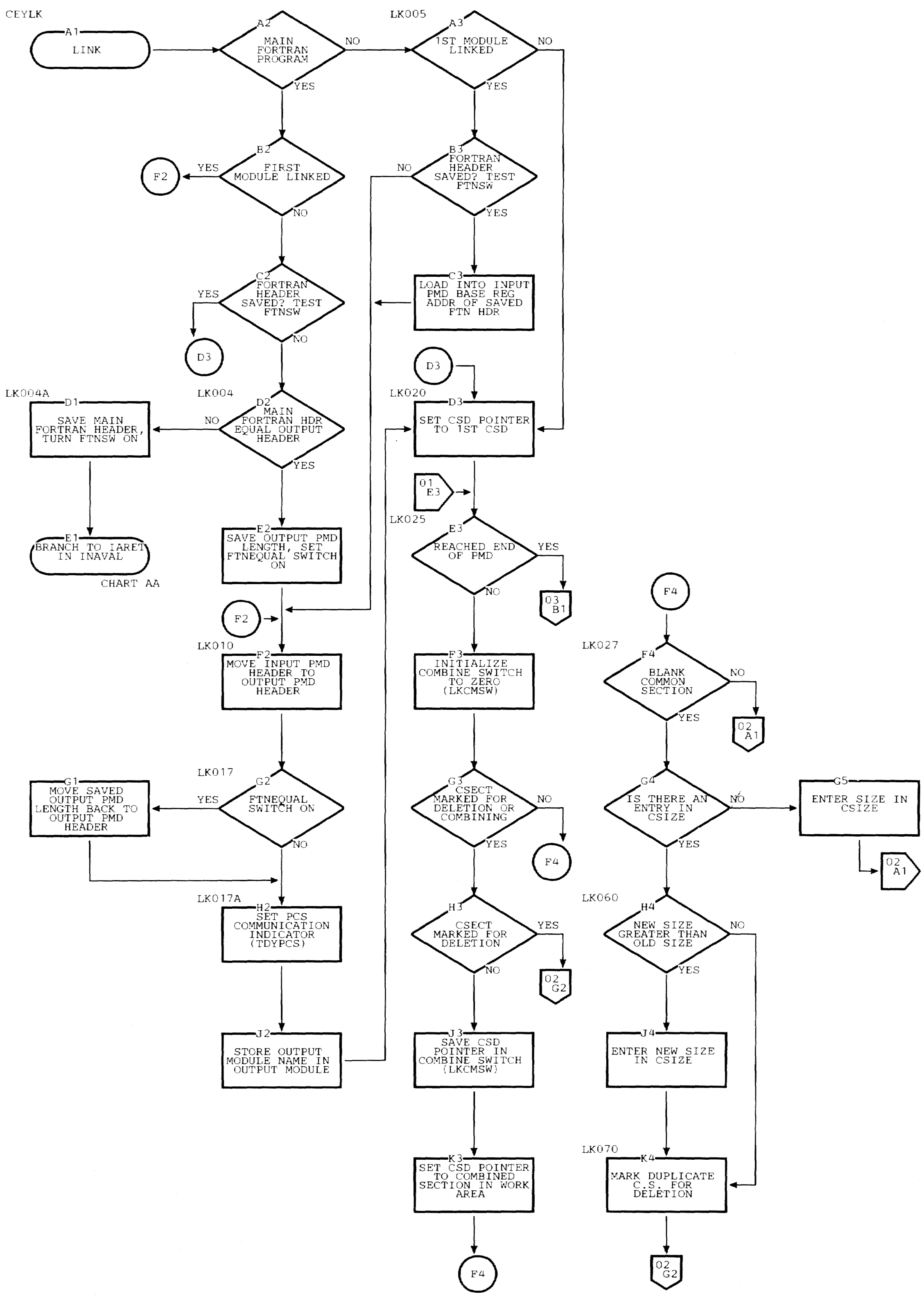

CEYCT
A1
COLLECT
B1
B1
LOCATE NEXT CSD HEADING IN SYSLIB MODULE
C1
FOUND
NO MORE
C2
RETURN
YES
D1
IS IT A BLANK COMMON SECTION
NO
B1
YES
E1
IS THERE AN ENTRY IN CSIZE
NO
E2
ENTER SIZE IN CSIZE
B1
YES
CT060
F1
NEW SIZE EQ. TO OLD SIZE
LESS THAN OR EQUAL
GREATER THAN
B1
G1
ENTER NEW SIZE IN CSIZE
B1
A3
FIXISD
B3
SET COVER REG FOR ISD HEADING
C3
SET LNGTH OF ISD IN ISD HD, INIT. COMPOSITE LNGTH TO LNGTH OF ISD
D3
STORE OUTPUT MODULE NAME IN ISD HEADING
E3
SET COVER REGISTER FOR FIRST ISD MODULE HEADING
EN085
F3
COMPUTE LGTH FROM HDNG TO END OF ISD. ADD TO DISPLACEMENT FOR MODULE
G3
BUMP COVER REGISTER TO NEXT ISD MODULE HEADING
H3
END OF LINKAGE EDITOR ISD YET
NO
YES
A4
A4
A4
ATTACH STRING OF INPUT ISD'S. BUMP COMPOSITE LGTH OF ISD
B4
SET COVER REG FOR ISD HEADING SET L = 0
C4
EN090
C4
L=L+LGTH OF ISD
C5
SET DISPLACEMENT TO PRECEDING ISD = L
D4
BUMP COVER REG TO NEXT ISD
YES
D5
NO
DISPL TO PRECEDING ISD = 0
YES
E4
REACHED END OF STRING OF ISD
NO
E5
IS IT A L.E. ISD
YES
NO
C4
F4
EXIT (BRANCH) TO EN100

Chart AJ. LINK - Link Modules subroutine (page 1 of 5)



CEYLK
A1
LINK
A2
MAIN FORTRAN PROGRAM
NO
YES
LK005
A3
1ST MODULE LINKED
NO
YES
B2
FIRST MODULE LINKED
YES
F2
NO
B3
FORTRAN HEADER SAVED? TEST FTNSW
NO
YES
C2
FORTRAN HEADER SAVED? TEST FTNSW
YES
D3
NO
C3
LOAD INTO INPUT PMD BASE REG ADDR OF SAVED FTN HDR
D3
LK004A
D1
SAVE MAIN FORTRAN HEADER, TURN FTNSW ON
LK004
D2
MAIN FORTRAN HDR EQUAL OUTPUT HEADER
NO
YES
LK020
D3
SET CSD POINTER TO 1ST CSD
01 E3
LK025
E3
REACHED END OF PMD
YES
03 B1
NO
E1
BRANCH TO IARET IN INAVAL
CHART AA
E2
SAVE OUTPUT PMD LENGTH, SET FTNEQUAL SWITCH ON
F2
LK010
F2
MOVE INPUT PMD HEADER TO OUTPUT PMD HEADER
F3
INITIALIZE COMBINE SWITCH TO ZERO (LKCMSW)
F4
LK027
F4
BLANK COMMON SECTION
NO
02 A1
YES
LK017
G1
MOVE SAVED OUTPUT PMD LENGTH BACK TO OUTPUT PMD HEADER
G2
FTNEQUAL SWITCH ON
YES
NO
G3
CSECT MARKED FOR DELETION OR COMBINING
NO
F4
YES
G4
IS THERE AN ENTRY IN CSIZE
NO
YES
G5
ENTER SIZE IN CSIZE
02 A1
LK017A
H2
SET PCS COMMUNICATION INDICATOR (TDYPCS)
H3
CSECT MARKED FOR DELETION
YES
02 G2
NO
LK060
H4
NEW SIZE GREATER THAN OLD SIZE
NO
YES
J2
STORE OUTPUT MODULE NAME IN OUTPUT MODULE
J3
SAVE CSD POINTER IN COMBINE SWITCH (LKCMSW)
J4
ENTER NEW SIZE IN CSIZE
K3
SET CSD POINTER TO COMBINED SECTION IN WORK AREA
F4
LK070
K4
MARK DUPLICATE C.S. FOR DELETION
02 G2

Chart AJ. LINK - Link Modules subroutine (page 2 of 5)

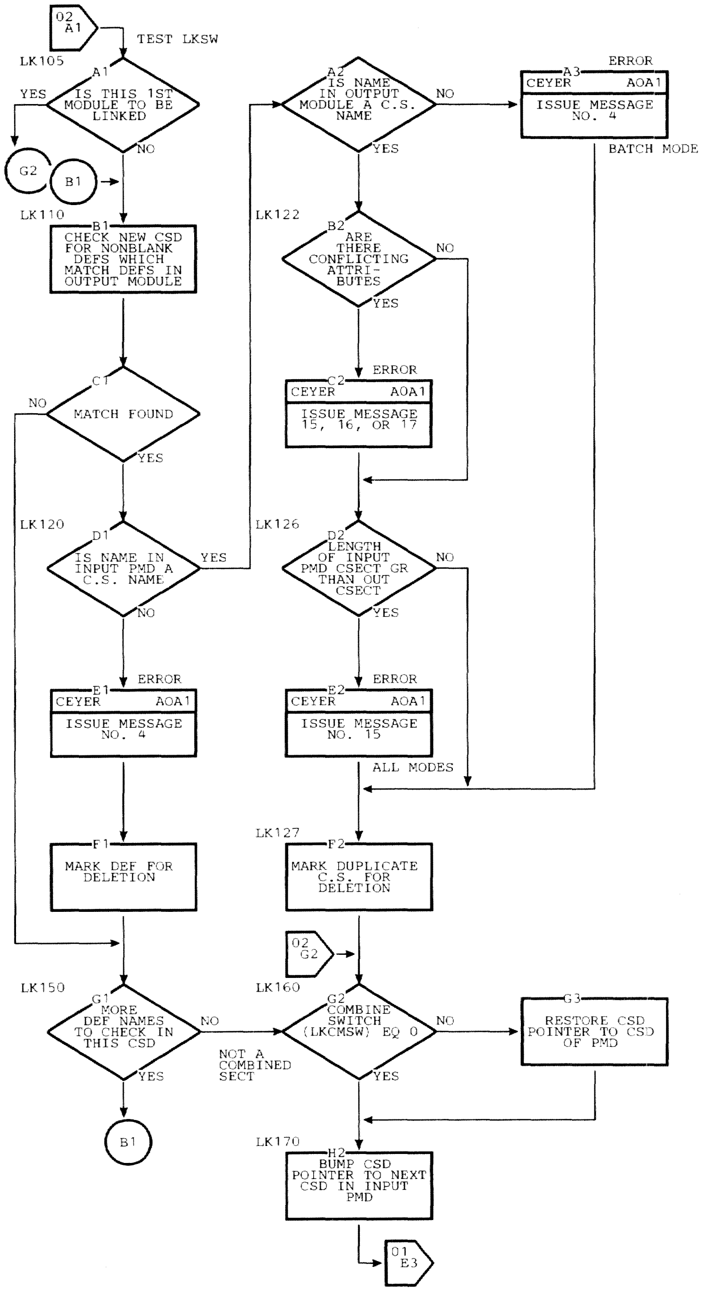

02 A1
TEST LKSW
LK105
A1
IS THIS 1ST MODULE TO BE LINKED
YES
NO
G2
B1
LK110
B1
CHECK NEW CSD FOR NONBLANK DEFS WHICH MATCH DEFS IN OUTPUT MODULE
C1
MATCH FOUND
NO
YES
LK120
D1
IS NAME IN INPUT PMD A C.S. NAME
YES
NO
ERROR
E1
CEYER AOA1
ISSUE MESSAGE NO. 4
F1
MARK DEF FOR DELETION
LK150
G1
MORE DEF NAMES TO CHECK IN THIS CSD
NO
YES
B1
NOT A COMBINED SECT
A2
IS NAME IN OUTPUT MODULE A C.S. NAME
NO
YES
ERROR
A3
CEYER AOA1
ISSUE MESSAGE NO. 4
BATCH MODE
LK122
B2
ARE THERE CONFLICTING ATTRI-BUTES
NO
YES
ERROR
C2
CEYER AOA1
ISSUE MESSAGE 15, 16, OR 17
LK126
D2
LENGTH OF INPUT PMD CSECT GR THAN OUT CSECT
NO
YES
ERROR
E2
CEYER AOA1
ISSUE MESSAGE NO. 15
ALL MODES
LK127
F2
MARK DUPLICATE C.S. FOR DELETION
02 G2
LK160
G2
COMBINE SWITCH (LKCMSW) EQ 0
NO
YES
G3
RESTORE CSD POINTER TO CSD OF PMD
LK170
H2
BUMP CSD POINTER TO NEXT CSD IN INPUT PMD
01 E3

Chart AJ. LINK - Link Modules subroutine (page 3 of 5)

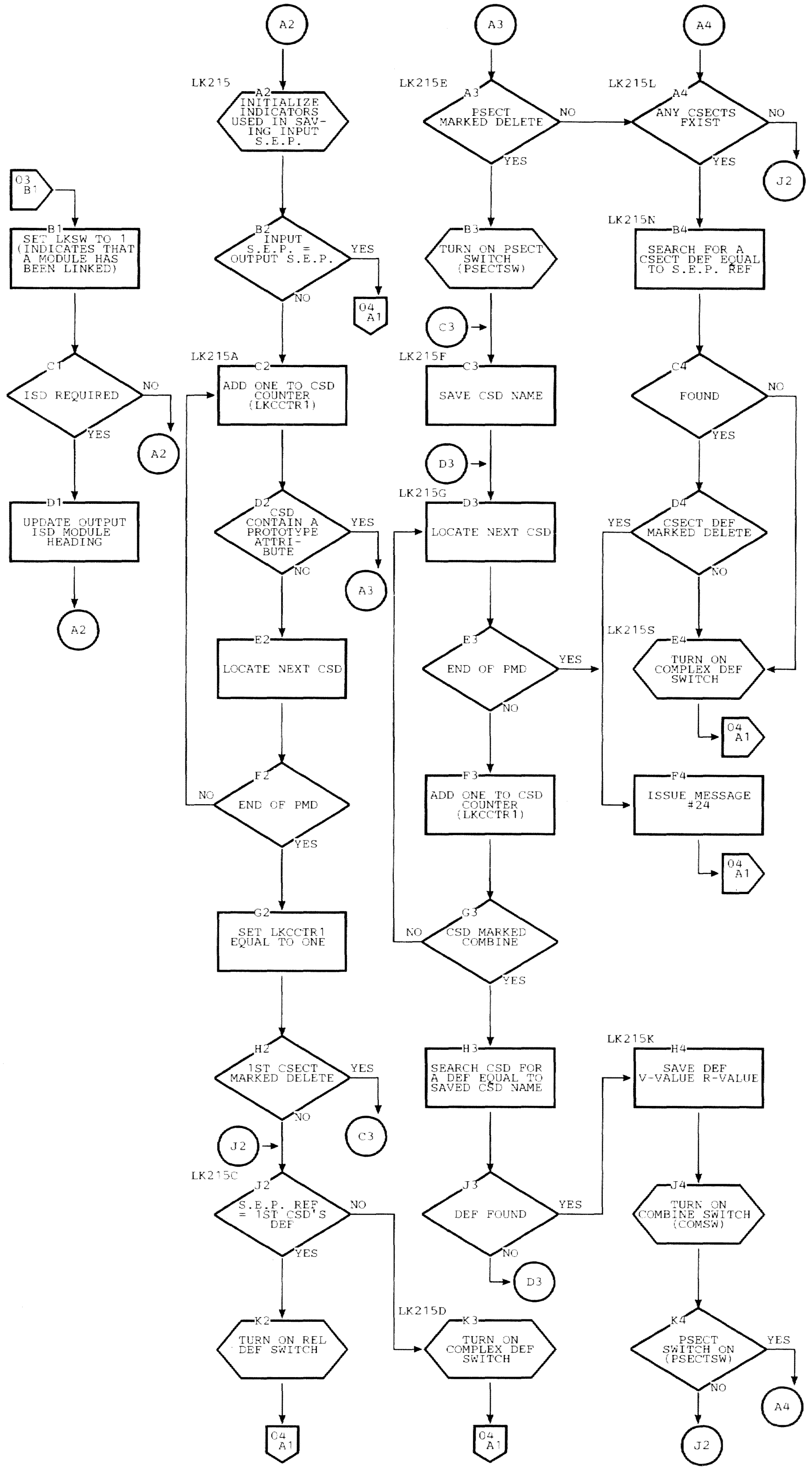


Chart AJ. LINK - Link Modules subroutine (page 4 of 5)

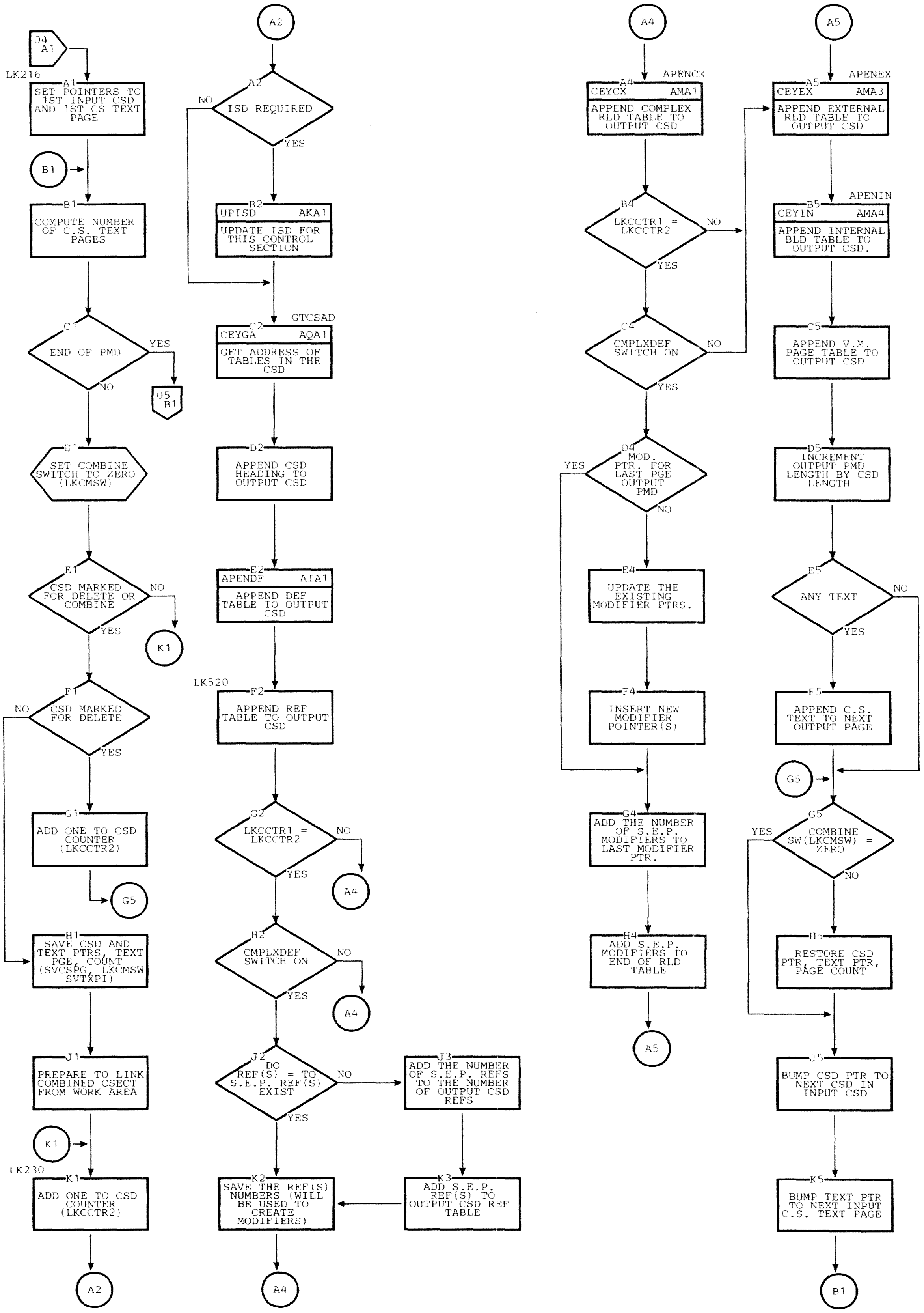

04 A1
LK216
A1 SET POINTERS TO 1ST INPUT CSD AND 1ST CS TEXT PAGE
B1
B1 COMPUTE NUMBER OF C.S. TEXT PAGES
C1 END OF PMD
YES
05 B1
NO
D1 SET COMBINE SWITCH TO ZERO (LKCMSW)
E1 CSD MARKED FOR DELETE OR COMBINE
NO
K1
YES
F1 CSD MARKED FOR DELETE
NO
YES
G1 ADD ONE TO CSD COUNTER (LKCCTR2)
G5
H1 SAVE CSD AND TEXT PTRS, TEXT PGE, COUNT (SVCSPG, LKCMSW SVTXPI)
J1 PREPARE TO LINK COMBINED CSECT FROM WORK AREA
K1
LK230
K1 ADD ONE TO CSD COUNTER (LKCCTR2)
A2
A2
A2 ISD REQUIRED
NO
YES
B2 UPISD AKA1
UPDATE ISD FOR THIS CONTROL SECTION
GTCSAD
C2 CEYGA AQA1
GET ADDRESS OF TABLES IN THE CSD
D2 APPEND CSD HEADING TO OUTPUT CSD
E2 APENDF AIA1
APPEND DEF TABLE TO OUTPUT CSD
LK520
F2 APPEND REF TABLE TO OUTPUT CSD
G2 LKCCTR1 = LKCCTR2
NO
A4
YES
H2 CMPLXDEF SWITCH ON
NO
A4
YES
J2 DO REF(S) = TO S.E.P. REF(S) EXIST
NO
J3 ADD THE NUMBER OF S.E.P. REFS TO THE NUMBER OF OUTPUT CSD REFS
YES
K2 SAVE THE REF(S) NUMBERS (WILL BE USED TO CREATE MODIFIERS)
K3 ADD S.E.P. REF(S) TO OUTPUT CSD REF TABLE
A4
A4
APENCX
A4 CEYCX AMA1
APPEND COMPLEX RLD TABLE TO OUTPUT CSD
B4 LKCCTR1 = LKCCTR2
NO
YES
C4 CMPLXDEF SWITCH ON
NO
YES
D4 MOD. PTR. FOR LAST PGE OUTPUT PMD
YES
NO
E4 UPDATE THE EXISTING MODIFIER PTRS.
F4 INSERT NEW MODIFIER POINTER(S)
G4 ADD THE NUMBER OF S.E.P. MODIFIERS TO LAST MODIFIER PTR.
H4 ADD S.E.P. MODIFIERS TO END OF RLD TABLE
A5
A5
APENEX
A5 CEYEX AMA3
APPEND EXTERNAL RLD TABLE TO OUTPUT CSD
APENIN
B5 CEYIN AMA4
APPEND INTERNAL BLD TABLE TO OUTPUT CSD.
C5 APPEND V.M. PAGE TABLE TO OUTPUT CSD
D5 INCREMENT OUTPUT PMD LENGTH BY CSD LENGTH
E5 ANY TEXT
NO
YES
F5 APPEND C.S. TEXT TO NEXT OUTPUT PAGE
G5
G5 COMBINE SW(LKCMSW) = ZERO
YES
NO
H5 RESTORE CSD PTR, TEXT PTR, PAGE COUNT
J5 BUMP CSD PTR TO NEXT CSD IN INPUT CSD
K5 BUMP TEXT PTR TO NEXT INPUT C.S. TEXT PAGE
B1

Chart AJ. LINK - Link Modules subroutine (page 5 of 5)

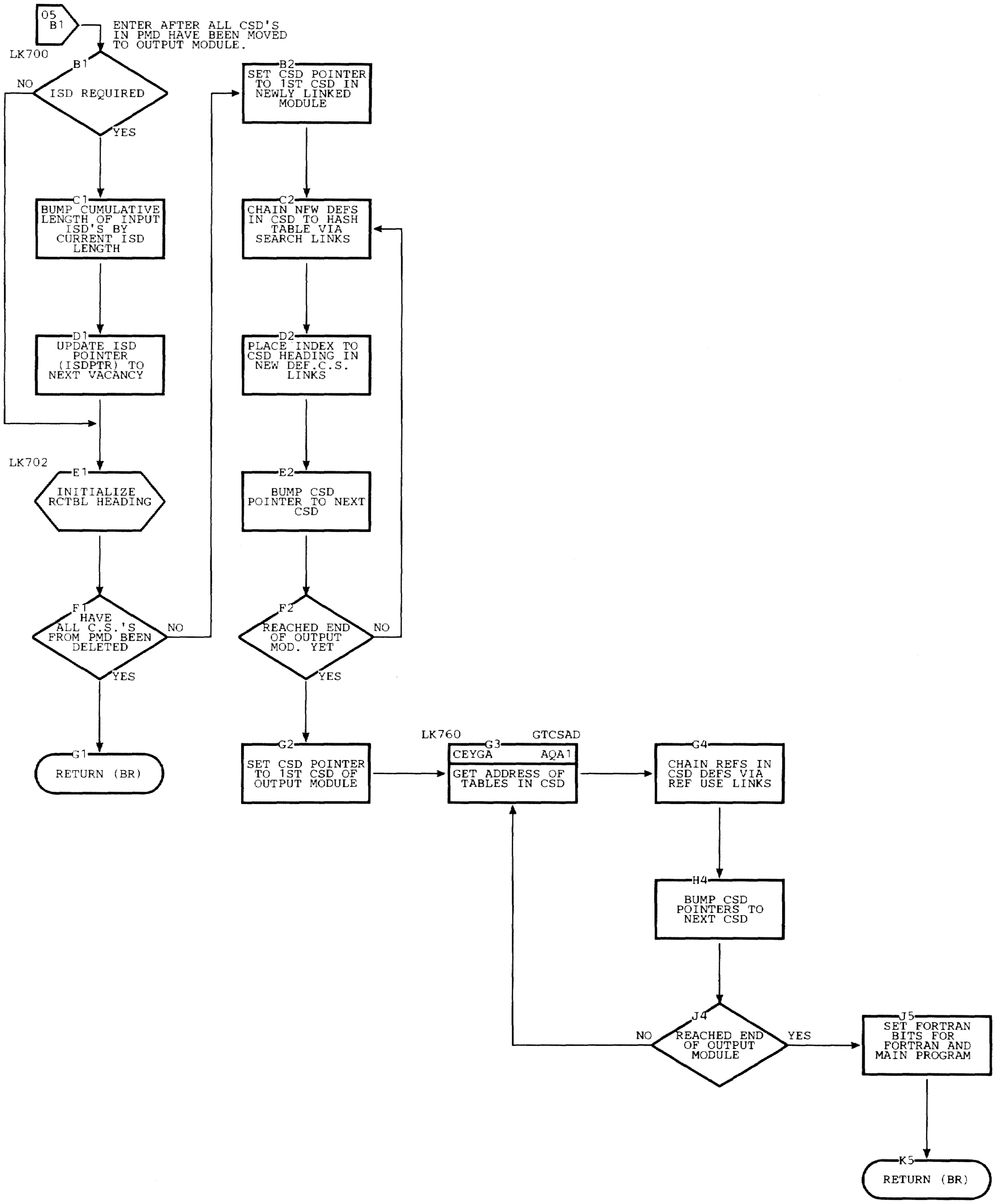

05
B1
ENTER AFTER ALL CSD'S IN PMD HAVE BEEN MOVED TO OUTPUT MODULE.
LK700
B1
ISD REQUIRED
NO
YES
C1
BUMP CUMULATIVE LENGTH OF INPUT ISD'S BY CURRENT ISD LENGTH
D1
UPDATE ISD POINTER (ISDPTR) TO NEXT VACANCY
LK702
E1
INITIALIZE RCTBL HEADING
F1
HAVE ALL C.S.'S FROM PMD BEEN DELETED
NO
YES
G1
RETURN (BR)
B2
SET CSD POINTER TO 1ST CSD IN NEWLY LINKED MODULE
C2
CHAIN NEW DEFS IN CSD TO HASH TABLE VIA SEARCH LINKS
D2
PLACE INDEX TO CSD HEADING IN NEW DEF.C.S. LINKS
E2
BUMP CSD POINTER TO NEXT CSD
F2
REACHED END OF OUTPUT MOD. YET
NO
YES
G2
SET CSD POINTER TO 1ST CSD OF OUTPUT MODULE
LK760
GTCSAD
G3
CEYGA
AQA1
GET ADDRESS OF TABLES IN CSD
G4
CHAIN REFS IN CSD DEFS VIA REF USE LINKS
H4
BUMP CSD POINTERS TO NEXT CSD
J4
REACHED END OF OUTPUT MODULE
NO
YES
J5
SET FORTRAN BITS FOR FORTRAN AND MAIN PROGRAM
K5
RETURN (BR)

Chart AK. UPISD - Update ISD subroutine

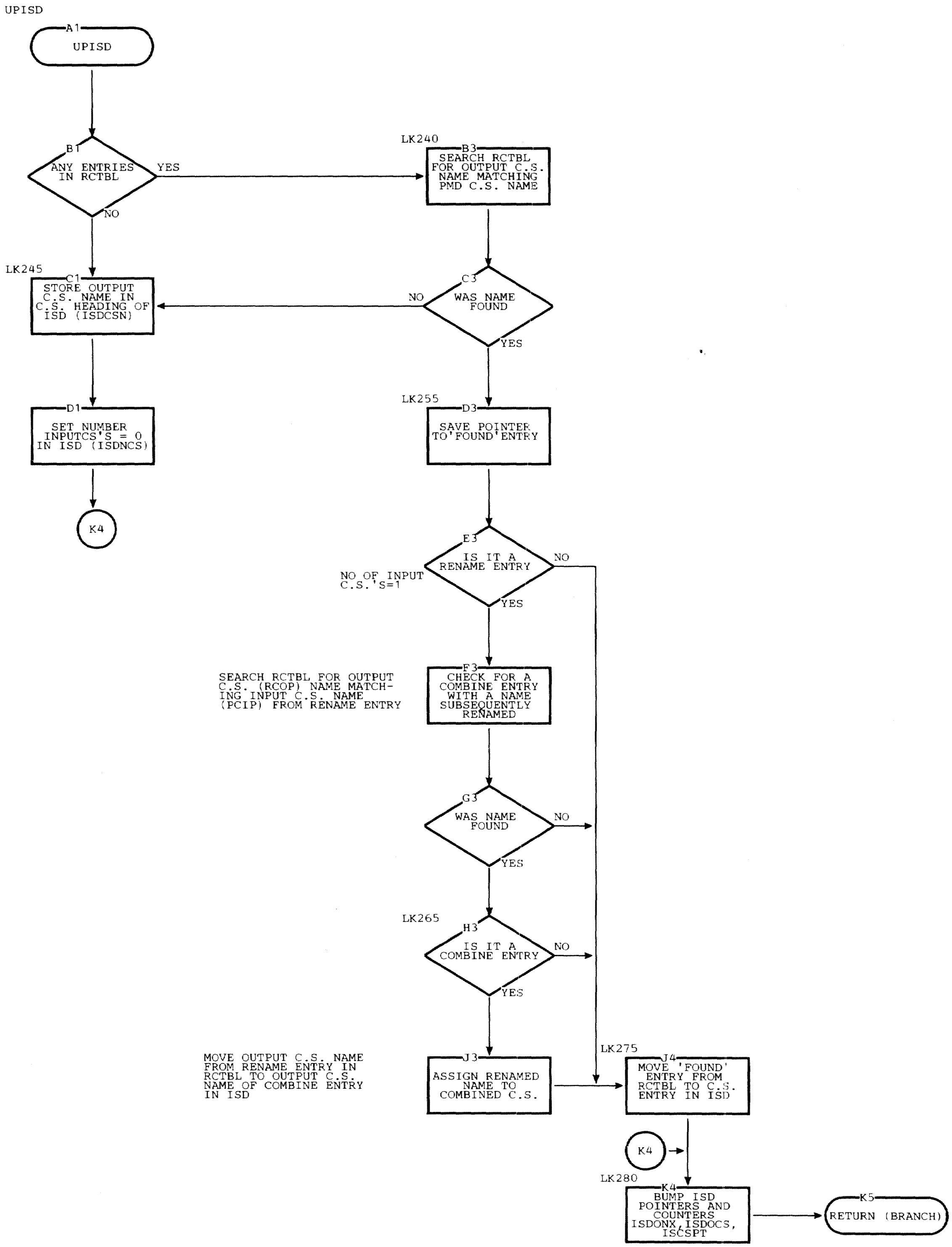

UPISD
A1
UPISD
B1
ANY ENTRIES IN RCTBL
YES
NO
LK240
B3
SEARCH RCTBL FOR OUTPUT C.S. NAME MATCHING PMD C.S. NAME
LK245
C1
STORE OUTPUT C.S. NAME IN C.S. HEADING OF ISD (ISDCSN)
C3
WAS NAME FOUND
NO
YES
D1
SET NUMBER INPUTCS'S = 0 IN ISD (ISDNCS)
K4
LK255
D3
SAVE POINTER TO'FOUND'ENTRY
E3
IS IT A RENAME ENTRY
NO OF INPUT C.S.'S=1
NO
YES
SEARCH RCTBL FOR OUTPUT C.S. (RCOP) NAME MATCH-ING INPUT C.S. NAME (PCIP) FROM RENAME ENTRY
F3
CHECK FOR A COMBINE ENTRY WITH A NAME SUBSEQUENTLY RENAMED
G3
WAS NAME FOUND
NO
YES
LK265
H3
IS IT A COMBINE ENTRY
NO
YES
MOVE OUTPUT C.S. NAME FROM RENAME ENTRY IN RCTBL TO OUTPUT C.S. NAME OF COMBINE ENTRY IN ISD
J3
ASSIGN RENAMED NAME TO COMBINED C.S.
LK275
J4
MOVE 'FOUND' ENTRY FROM RCTBL TO C.S. ENTRY IN ISD
K4
LK280
K4
BUMP ISD POINTERS AND COUNTERS ISDONX,ISDOCS, ISCSPT
K5
RETURN (BRANCH)

Chart AL. APENDF - Append Definition Table subroutine

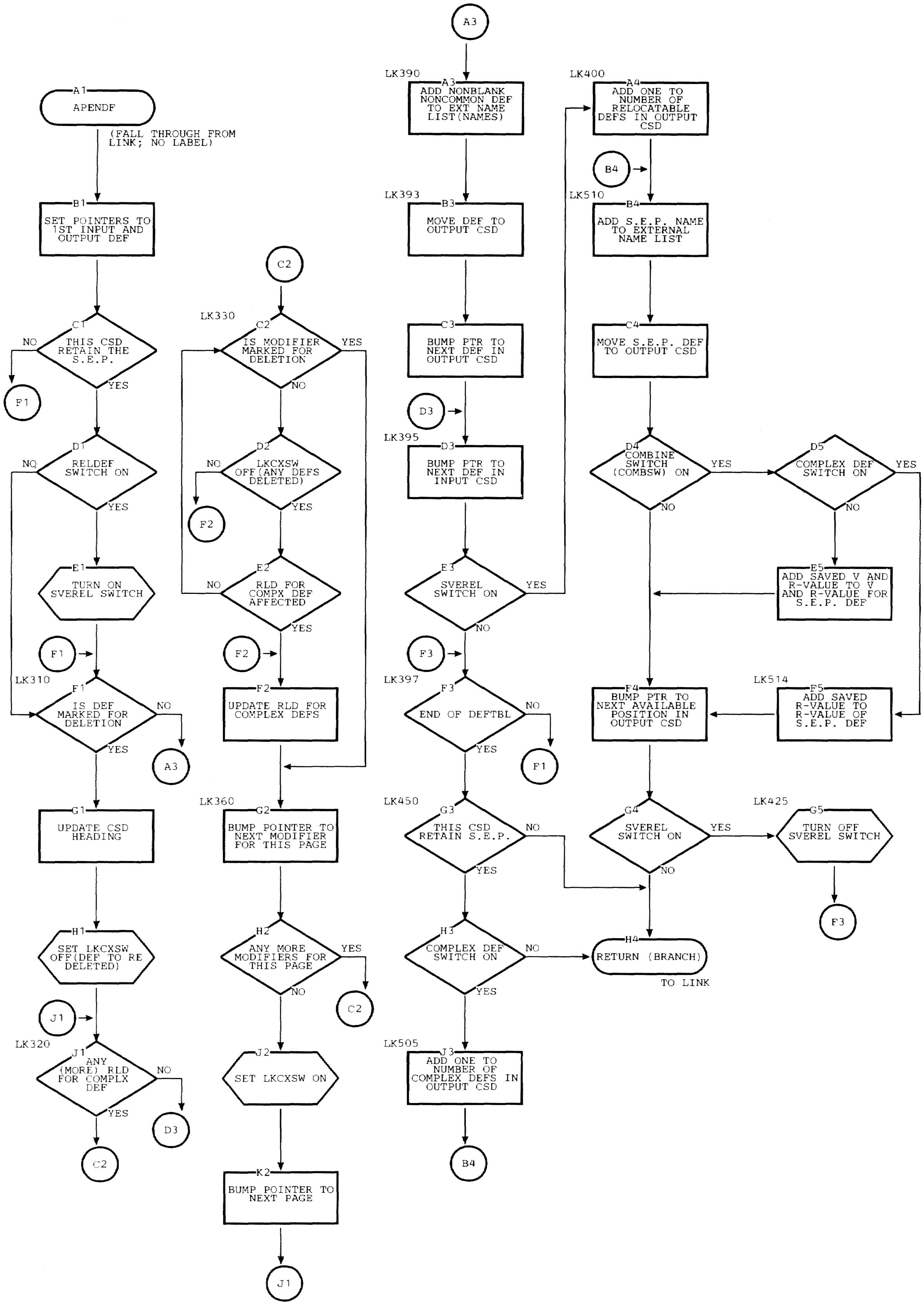

A1
APENDF
(FALL THROUGH FROM LINK; NO LABEL)
B1
SET POINTERS TO 1ST INPUT AND OUTPUT DEF
C1
THIS CSD RETAIN THE S.E.P.
NO
F1
YES
D1
RELDEF SWITCH ON
NO
YES
E1
TURN ON SVEREL SWITCH
F1
LK310
F1
IS DEF MARKED FOR DELETION
NO
A3
YES
G1
UPDATE CSD HEADING
H1
SET LKCXSW OFF(DEF TO RE DELETED)
J1
LK320
J1
ANY (MORE) RLD FOR COMPLX DEF
NO
D3
YES
C2
C2
LK330
C2
IS MODIFIER MARKED FOR DELETION
YES
NO
D2
LKCXSW OFF(ANY DEFS DELETED)
NO
F2
YES
E2
RLD FOR COMPX DEF AFFECTED
NO
YES
F2
F2
UPDATE RLD FOR COMPLEX DEFS
LK360
G2
BUMP POINTER TO NEXT MODIFIER FOR THIS PAGE
H2
ANY MORE MODIFIERS FOR THIS PAGE
YES
C2
NO
J2
SET LKCXSW ON
K2
BUMP POINTER TO NEXT PAGE
J1
A3
LK390
A3
ADD NONBLANK NONCOMMON DEF TO EXT NAME LIST(NAMES)
LK393
B3
MOVE DEF TO OUTPUT CSD
C3
BUMP PTR TO NEXT DEF IN OUTPUT CSD
D3
LK395
D3
BUMP PTR TO NEXT DEF IN INPUT CSD
E3
SVEREL SWITCH ON
YES
NO
F3
LK397
F3
END OF DEFTBL
NO
F1
YES
LK450
G3
THIS CSD RETAIN S.E.P.
NO
YES
H3
COMPLEX DEF SWITCH ON
NO
YES
LK505
J3
ADD ONE TO NUMBER OF COMPLEX DEFS IN OUTPUT CSD
B4
LK400
A4
ADD ONE TO NUMBER OF RELOCATABLE DEFS IN OUTPUT CSD
B4
LK510
B4
ADD S.E.P. NAME TO EXTERNAL NAME LIST
C4
MOVE S.E.P. DEF TO OUTPUT CSD
D4
COMBINE SWITCH (COMBSW) ON
YES
NO
F4
BUMP PTR TO NEXT AVAILABLE POSITION IN OUTPUT CSD
G4
SVEREL SWITCH ON
YES
NO
H4
RETURN (BRANCH)
TO LINK
D5
COMPLEX DEF SWITCH ON
YES
NO
E5
ADD SAVED V AND R-VALUE TO V AND R-VALUE FOR S.E.P. DEF
LK514
F5
ADD SAVED R-VALUE TO R-VALUE OF S.E.P. DEF
LK425
G5
TURN OFF SVEREL SWITCH
F3

Chart AM. APENCX, APENEX, and APENIN

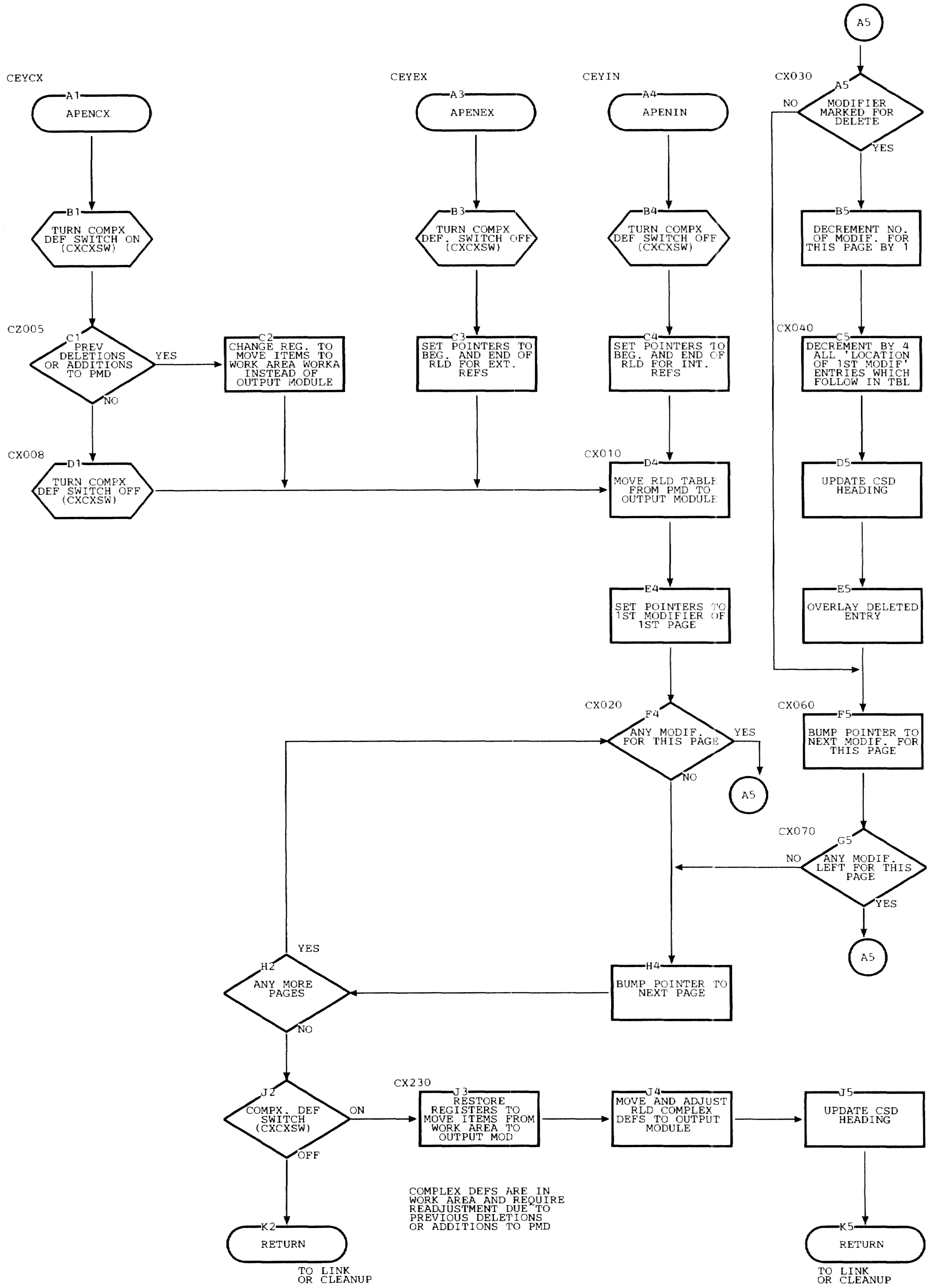

CEYCX
A1
APENCX
B1
TURN COMPX DEF SWITCH ON (CXCXSW)
CZ005
C1
PREV DELETIONS OR ADDITIONS TO PMD
YES
NO
C2
CHANGE REG. TO MOVE ITEMS TO WORK AREA WORKA INSTEAD OF OUTPUT MODULE
CX008
D1
TURN COMPX DEF SWITCH OFF (CXCXSW)
CEYEX
A3
APENEX
B3
TURN COMPX DEF. SWITCH OFF (CXCXSW)
C3
SET POINTERS TO BEG. AND END OF RLD FOR EXT. REFS
CEYIN
A4
APENIN
B4
TURN COMPX DEF SWITCH OFF (CXCXSW)
C4
SET POINTERS TO BEG. AND END OF RLD FOR INT. REFS
CX010
D4
MOVE RLD TABLE FROM PMD TO OUTPUT MODULE
E4
SET POINTERS TO 1ST MODIFIER OF 1ST PAGE
CX020
F4
ANY MODIF. FOR THIS PAGE
YES
NO
A5
A5
CX030
A5
MODIFIER MARKED FOR DELETE
NO
YES
B5
DECREMENT NO. OF MODIF. FOR THIS PAGE BY 1
CX040
C5
DECREMENT BY 4 ALL 'LOCATION OF 1ST MODIF' ENTRIES WHICH FOLLOW IN TBL
D5
UPDATE CSD HEADING
E5
OVERLAY DELETED ENTRY
CX060
F5
BUMP POINTER TO NEXT MODIF. FOR THIS PAGE
CX070
G5
ANY MODIF. LEFT FOR THIS PAGE
NO
YES
A5
H4
BUMP POINTER TO NEXT PAGE
H2
ANY MORE PAGES
YES
NO
J2
COMPX. DEF SWITCH (CXCXSW)
ON
OFF
CX230
J3
RESTORE REGISTERS TO MOVE ITEMS FROM WORK AREA TO OUTPUT MOD
J4
MOVE AND ADJUST RLD COMPLEX DEFS TO OUTPUT MODULE
J5
UPDATE CSD HEADING
COMPLEX DEFS ARE IN WORK AREA AND REQUIRE READJUSTMENT DUE TO PREVIOUS DELETIONS OR ADDITIONS TO PMD
K2
RETURN
TO LINK OR CLEANUP
K5
RETURN
TO LINK OR CLEANUP

Chart AN. BRING - Bring PMD, Text, and ISD from Library (page 1 of 2)

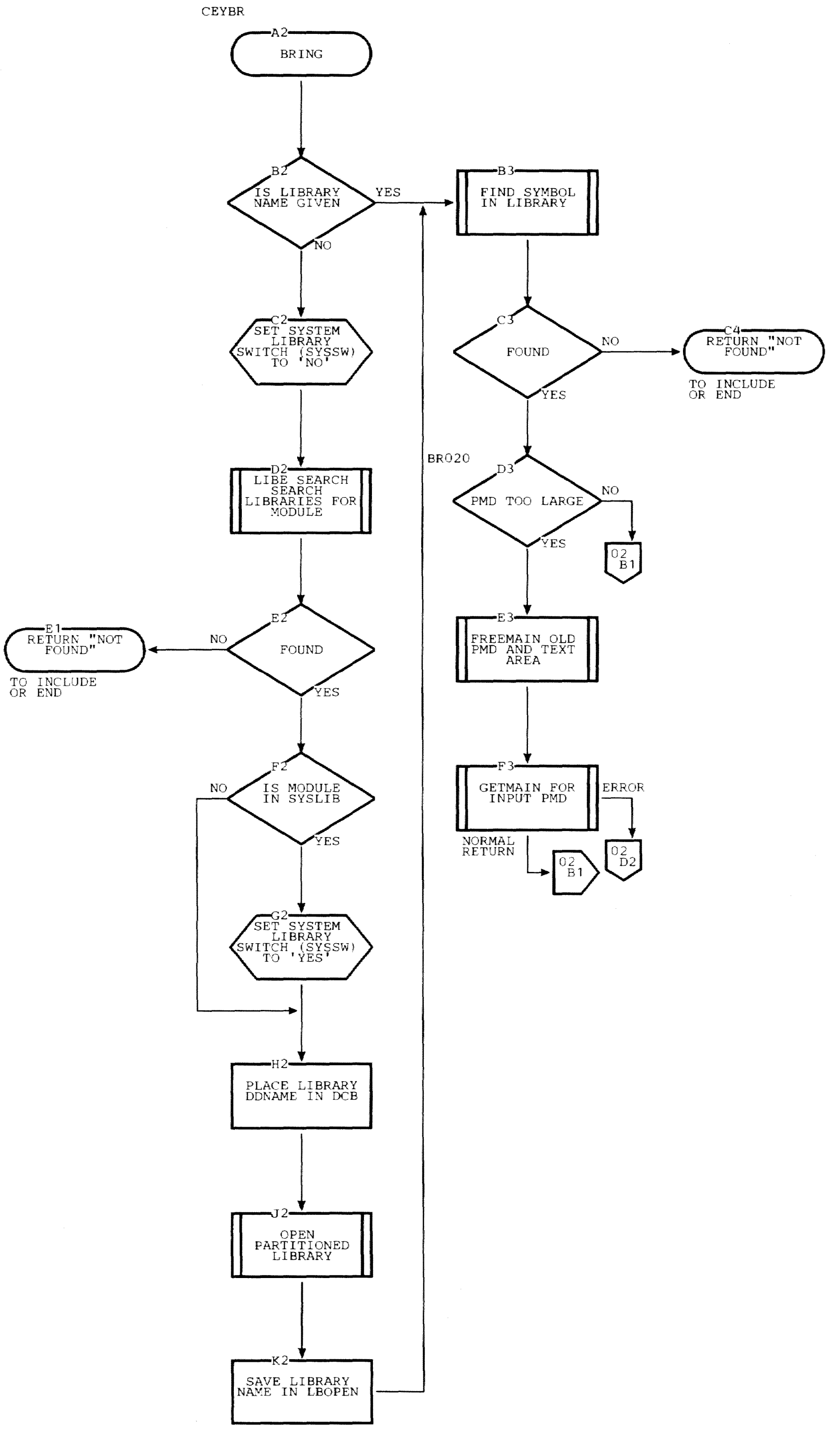


Chart AN. BRING - Bring PMD, Text, and ISD from Library (page 2 of 2)

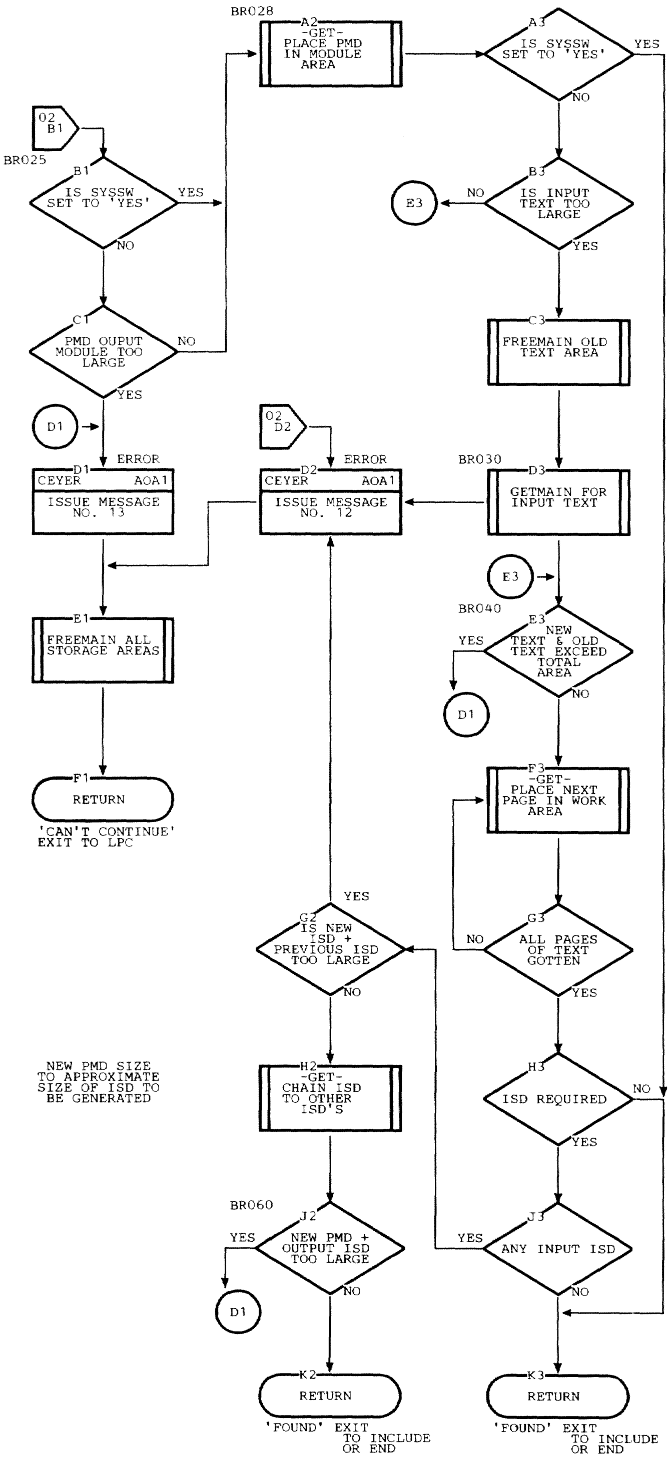


Chart AO. ERROR - Error Message Processor

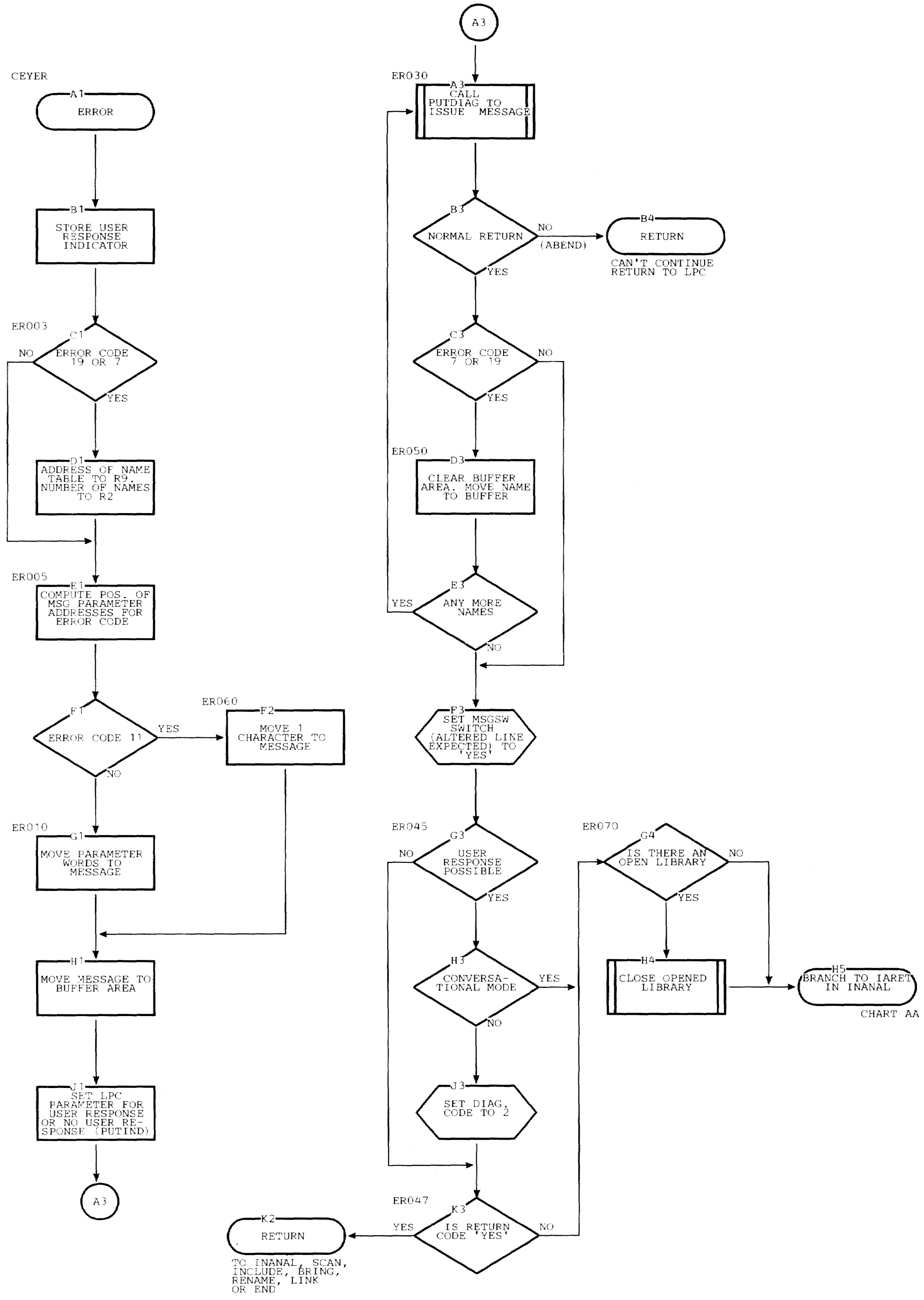

CEYER
A1
ERROR
B1
STORE USER RESPONSE INDICATOR
ER003
C1
ERROR CODE 19 OR 7
NO
YES
D1
ADDRESS OF NAME TABLE TO R9. NUMBER OF NAMES TO R2
ER005
E1
COMPUTE POS. OF MSG PARAMETER ADDRESSES FOR ERROR CODE
F1
ERROR CODE 11
YES
NO
ER060
F2
MOVE 1 CHARACTER TO MESSAGE
ER010
G1
MOVE PARAMETER WORDS TO MESSAGE
H1
MOVE MESSAGE TO BUFFER AREA
J1
SET LPC PARAMETER FOR USER RESPONSE OR NO USER RESPONSE (PUTIND)
A3
A3
ER030
A3
CALL PUTDIAG TO ISSUE MESSAGE
B3
NORMAL RETURN
NO
(ABEND)
YES
B4
RETURN
CAN'T CONTINUE RETURN TO LPC
C3
ERROR CODE 7 OR 19
NO
YES
ER050
D3
CLEAR BUFFER AREA. MOVE NAME TO BUFFER
E3
ANY MORE NAMES
YES
NO
F3
SET MSGSW SWITCH (ALTERED LINE EXPECTED) TO 'YES'
ER045
G3
USER RESPONSE POSSIBLE
NO
YES
H3
CONVERSA-TIONAL MODE
YES
NO
J3
SET DIAG. CODE TO 2
ER047
K3
IS RETURN CODE 'YES'
YES
NO
K2
RETURN
TO INANAL, SCAN, INCLUDE, BRING, RENAME, LINK OR END
ER070
G4
IS THERE AN OPEN LIBRARY
NO
YES
H4
CLOSE OPENED LIBRARY
H5
BRANCH TO IARET IN INANAL
CHART AA

Chart AP. EXTREF - External Reference Search and GETCSD - Locate Control Section Dictionary

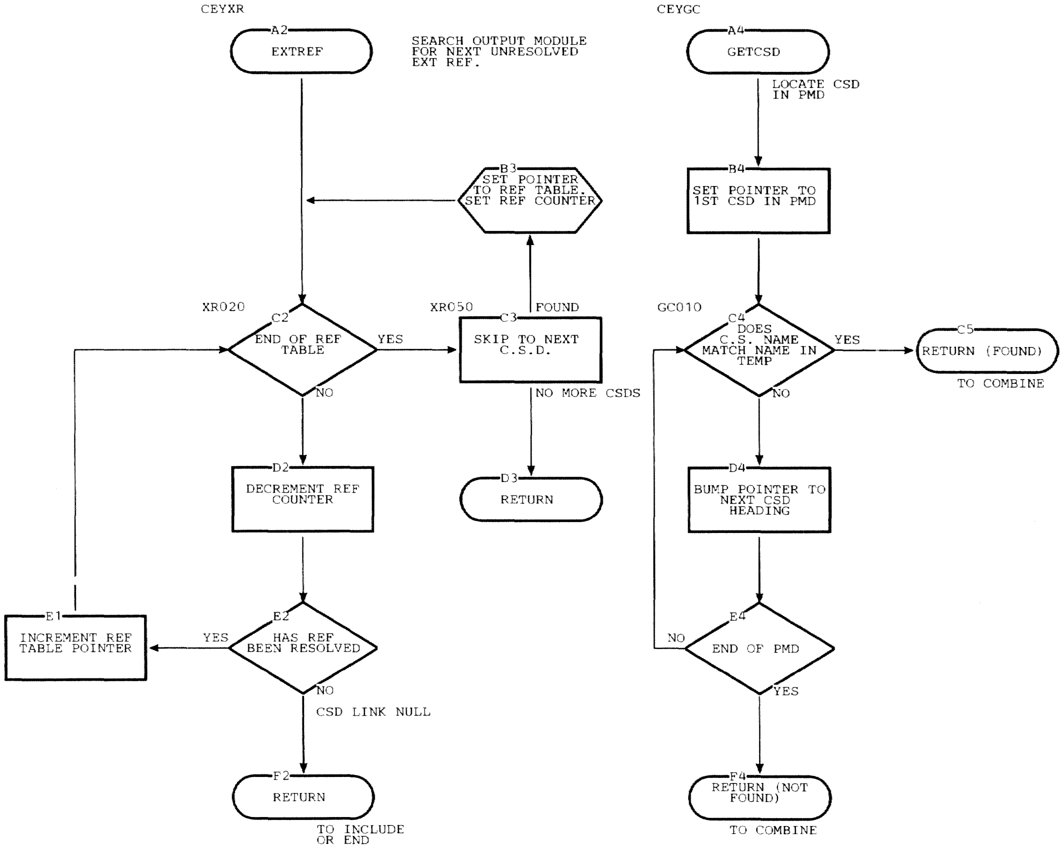


Chart AQ. GTCSAD - Get CSD Table Addresses and SCAN - Scan subroutine

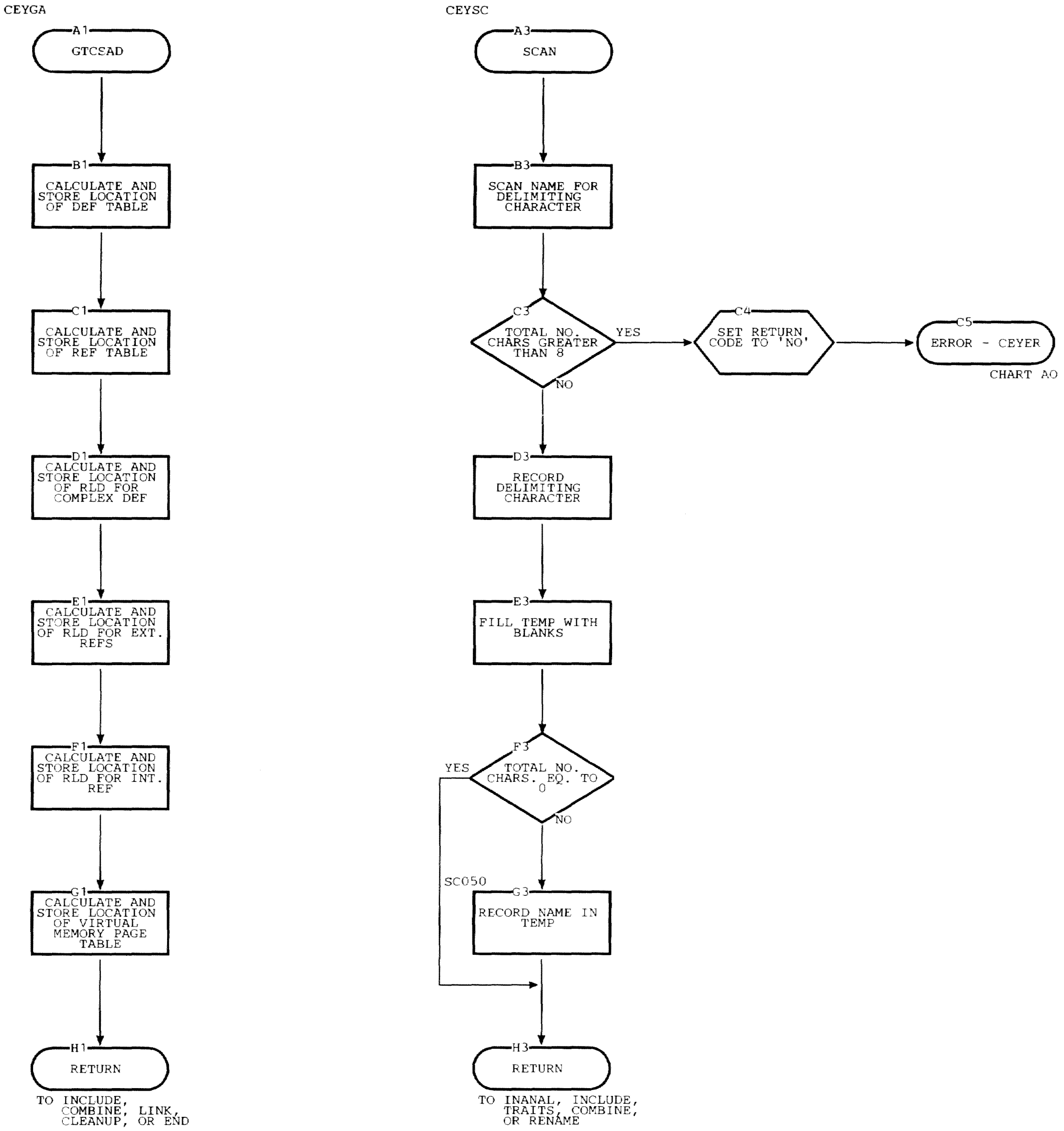

CEYGA
A1
GTCSAD
B1
CALCULATE AND STORE LOCATION OF DEF TABLE
C1
CALCULATE AND STORE LOCATION OF REF TABLE
D1
CALCULATE AND STORE LOCATION OF RLD FOR COMPLEX DEF
E1
CALCULATE AND STORE LOCATION OF RLD FOR EXT. REFS
F1
CALCULATE AND STORE LOCATION OF RLD FOR INT. REF
G1
CALCULATE AND STORE LOCATION OF VIRTUAL MEMORY PAGE TABLE
H1
RETURN
TO INCLUDE, COMBINE, LINK, CLEANUP, OR END
CEYSC
A3
SCAN
B3
SCAN NAME FOR DELIMITING CHARACTER
C3
TOTAL NO. CHARS GREATER THAN 8
YES
NO
C4
SET RETURN CODE TO 'NO'
C5
ERROR - CEYER
CHART AO
D3
RECORD DELIMITING CHARACTER
E3
FILL TEMP WITH BLANKS
F3
TOTAL NO. CHARS. EQ. TO 0
YES
NO
SC050
G3
RECORD NAME IN TEMP
H3
RETURN
TO INANAL, INCLUDE, TRAITS, COMBINE, OR RENAME

Chart BA. OUTPUT - Output routine

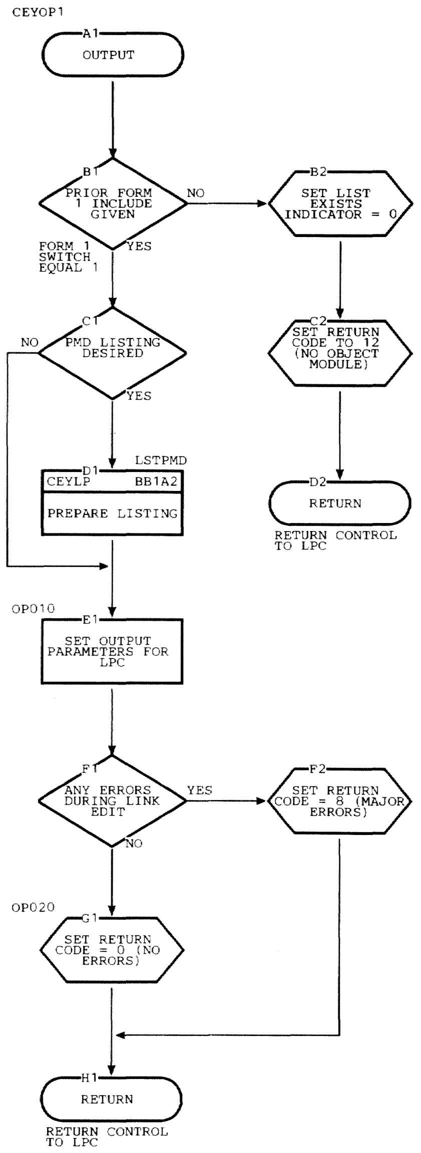

CEYOP1
A1
OUTPUT
B1
PRIOR FORM 1 INCLUDE GIVEN
NO
B2
SET LIST EXISTS INDICATOR = 0
FORM 1 SWITCH EQUAL 1
YES
C1
NO
PMD LISTING DESIRED
C2
SET RETURN CODE TO 12 (NO OBJECT MODULE)
YES
LSTPMD
D1
CEYLP BB1A2
PREPARE LISTING
D2
RETURN
RETURN CONTROL TO LPC
OP010
E1
SET OUTPUT PARAMETERS FOR LPC
F1
ANY ERRORS DURING LINK EDIT
YES
F2
SET RETURN CODE = 8 (MAJOR ERRORS)
NO
OP020
G1
SET RETURN CODE = 0 (NO ERRORS)
H1
RETURN
RETURN CONTROL TO LPC

Chart BB. LSTPMD - Program Module Dictionary Listing subroutine (page 1 of 2)

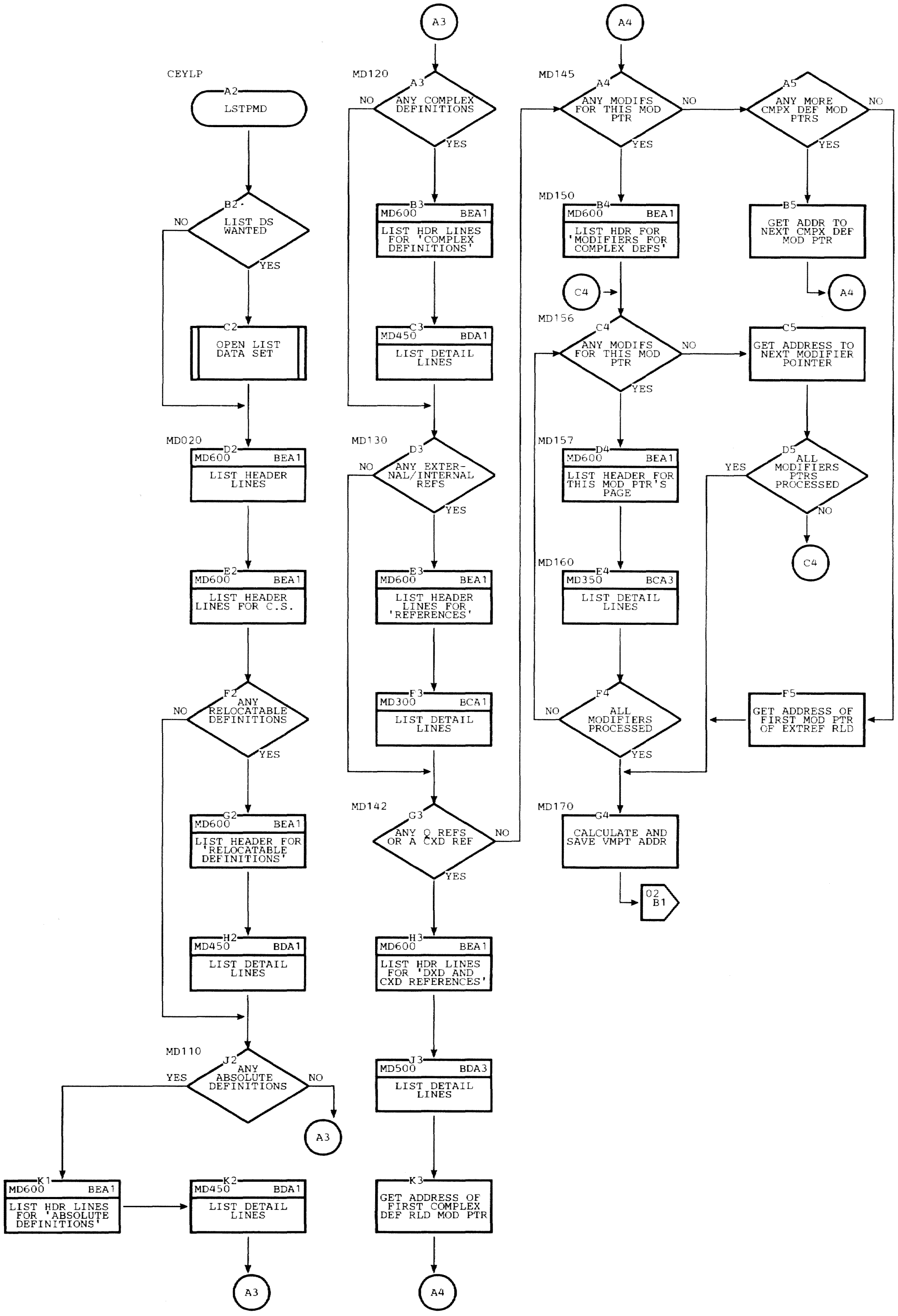

CEYLP
A2
LSTPMD
B2
LIST DS WANTED
NO
YES
C2
OPEN LIST DATA SET
MD020
D2
MD600 BEA1
LIST HEADER LINES
E2
MD600 BEA1
LIST HEADER LINES FOR C.S.
F2
ANY RELOCATABLE DEFINITIONS
NO
YES
G2
MD600 BEA1
LIST HEADER FOR 'RELOCATABLE DEFINITIONS'
H2
MD450 BDA1
LIST DETAIL LINES
MD110
J2
ANY ABSOLUTE DEFINITIONS
YES
NO
A3
K1
MD600 BEA1
LIST HDR LINES FOR 'ABSOLUTE DEFINITIONS'
K2
MD450 BDA1
LIST DETAIL LINES
A3
A3
MD120
A3
ANY COMPLEX DEFINITIONS
NO
YES
B3
MD600 BEA1
LIST HDR LINES FOR 'COMPLEX DEFINITIONS'
C3
MD450 BDA1
LIST DETAIL LINES
MD130
D3
ANY EXTER-NAL/INTERNAL REFS
NO
YES
E3
MD600 BEA1
LIST HEADER LINES FOR 'REFERENCES'
F3
MD300 BCA1
LIST DETAIL LINES
MD142
G3
ANY Q REFS OR A CXD REF
NO
YES
H3
MD600 BEA1
LIST HDR LINES FOR 'DXD AND CXD REFERENCES'
J3
MD500 BDA3
LIST DETAIL LINES
K3
GET ADDRESS OF FIRST COMPLEX DEF RLD MOD PTR
A4
A4
MD145
A4
ANY MODIFS FOR THIS MOD PTR
NO
YES
MD150
B4
MD600 BEA1
LIST HDR FOR 'MODIFIERS FOR COMPLEX DEFS'
C4
MD156
C4
ANY MODIFS FOR THIS MOD PTR
NO
YES
MD157
D4
MD600 BEA1
LIST HEADER FOR THIS MOD PTR'S PAGE
MD160
E4
MD350 BCA3
LIST DETAIL LINES
F4
ALL MODIFIERS PROCESSED
NO
YES
MD170
G4
CALCULATE AND SAVE VMPT ADDR
02 B1
A5
ANY MORE CMPX DEF MOD PTRS
NO
YES
B5
GET ADDR TO NEXT CMPX DEF MOD PTR
A4
C5
GET ADDRESS TO NEXT MODIFIER POINTER
D5
ALL MODIFIERS PTRS PROCESSED
YES
NO
C4
F5
GET ADDRESS OF FIRST MOD PTR OF EXTREF RLD

Chart BB. LSTPMD - Program Module Dictionary Listing subroutine and MD240 subroutine (page 2 of 2)

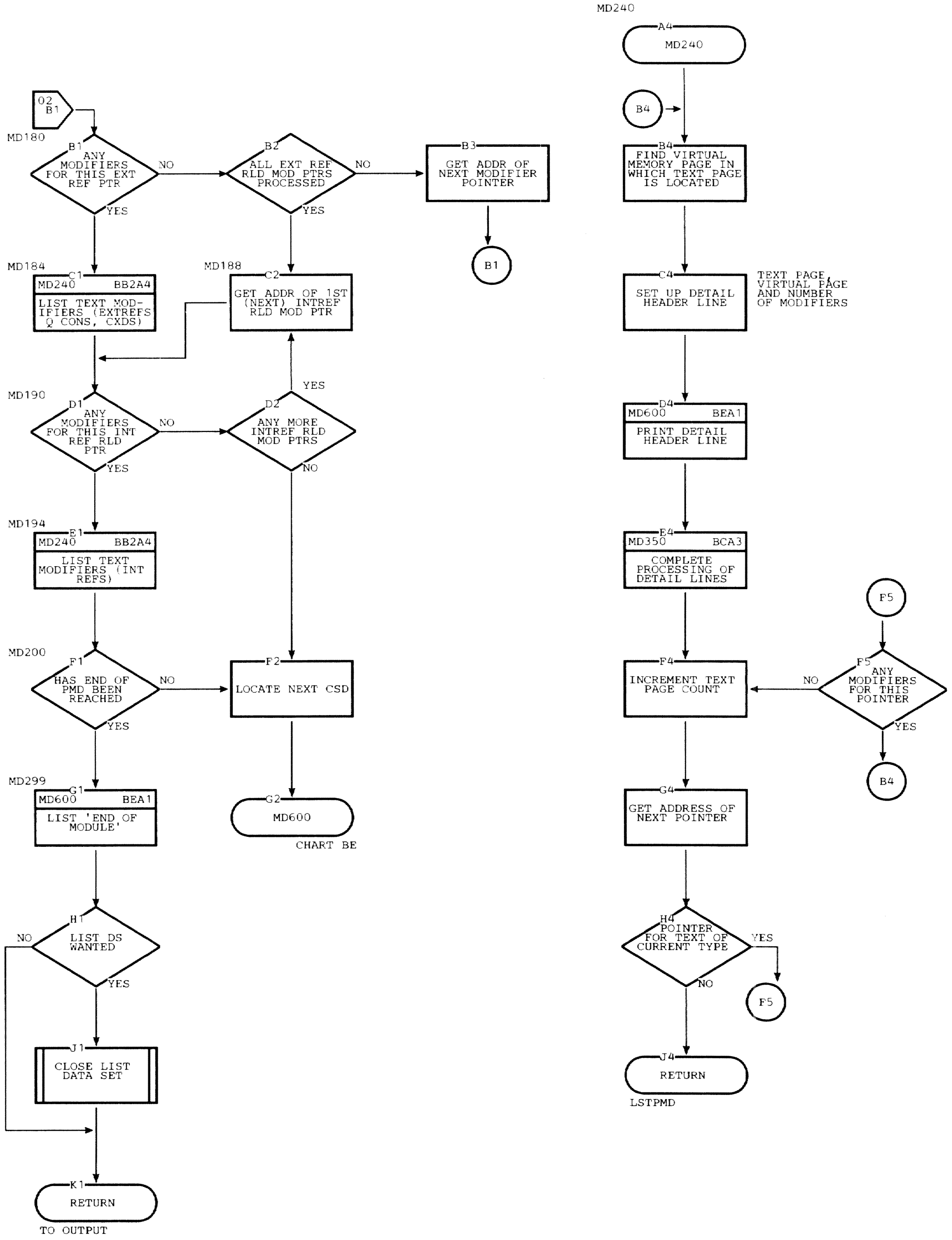

MD240
A4
MD240
B4
B4
FIND VIRTUAL MEMORY PAGE IN WHICH TEXT PAGE IS LOCATED
02 B1
MD180
B1
ANY MODIFIERS FOR THIS EXT REF PTR
NO
YES
B2
ALL EXT REF RLD MOD PTRS PROCESSED
NO
YES
B3
GET ADDR OF NEXT MODIFIER POINTER
B1
MD184
C1
MD240 BB2A4
LIST TEXT MOD- IFIERS (EXTREFS Q CONS, CXDS)
MD188
C2
GET ADDR OF 1ST (NEXT) INTREF RLD MOD PTR
C4
SET UP DETAIL HEADER LINE
TEXT PAGE, VIRTUAL PAGE AND NUMBER OF MODIFIERS
MD190
D1
ANY MODIFIERS FOR THIS INT REF RLD PTR
NO
YES
D2
ANY MORE INTREF RLD MOD PTRS
YES
NO
D4
MD600 BEA1
PRINT DETAIL HEADER LINE
MD194
E1
MD240 BB2A4
LIST TEXT MODIFIERS (INT REFS)
E4
MD350 BCA3
COMPLETE PROCESSING OF DETAIL LINES
F5
MD200
F1
HAS END OF PMD BEEN REACHED
NO
YES
F2
LOCATE NEXT CSD
F4
INCREMENT TEXT PAGE COUNT
NO
F5
ANY MODIFIERS FOR THIS POINTER
YES
B4
MD299
G1
MD600 BEA1
LIST 'END OF MODULE'
G2
MD600
CHART BE
G4
GET ADDRESS OF NEXT POINTER
H1
NO
LIST DS WANTED
YES
H4
POINTER FOR TEXT OF CURRENT TYPE
YES
NO
F5
J1
CLOSE LIST DATA SET
J4
RETURN
LSTPMD
K1
RETURN
TO OUTPUT

Chart BC. MD300 and MD350 subroutines

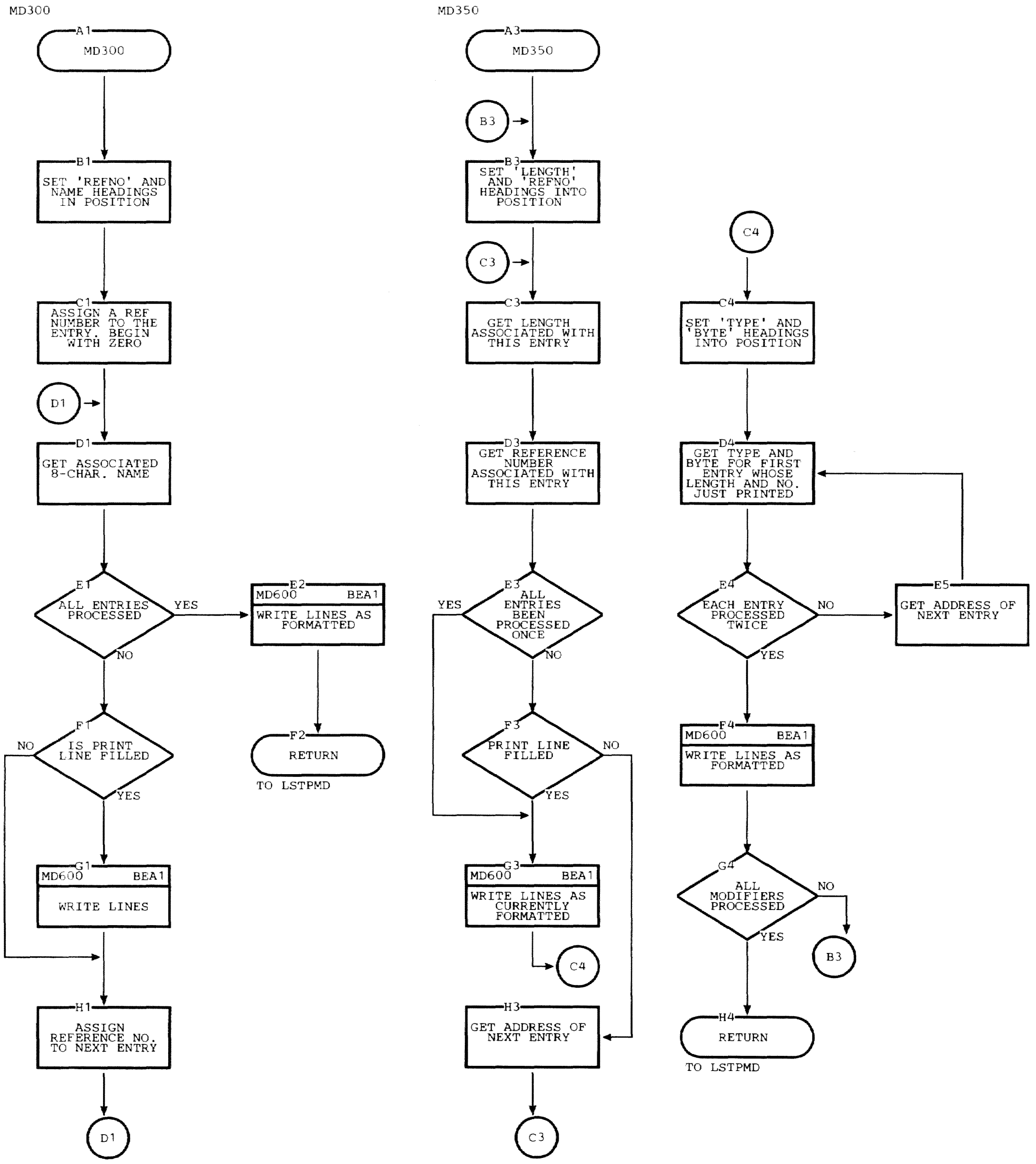

MD300
A1
MD300
B1
SET 'REFNO' AND NAME HEADINGS IN POSITION
C1
ASSIGN A REF NUMBER TO THE ENTRY. BEGIN WITH ZERO
D1
D1
GET ASSOCIATED 8-CHAR. NAME
E1
ALL ENTRIES PROCESSED
YES
NO
E2
MD600
BEA1
WRITE LINES AS FORMATTED
F2
RETURN
TO LSTPMD
F1
IS PRINT LINE FILLED
NO
YES
G1
MD600
BEA1
WRITE LINES
H1
ASSIGN REFERENCE NO. TO NEXT ENTRY
D1
MD350
A3
MD350
B3
B3
SET 'LENGTH' AND 'REFNO' HEADINGS INTO POSITION
C3
C3
GET LENGTH ASSOCIATED WITH THIS ENTRY
D3
GET REFERENCE NUMBER ASSOCIATED WITH THIS ENTRY
E3
ALL ENTRIES BEEN PROCESSED ONCE
YES
NO
F3
PRINT LINE FILLED
NO
YES
G3
MD600
BEA1
WRITE LINES AS CURRENTLY FORMATTED
C4
H3
GET ADDRESS OF NEXT ENTRY
C3
C4
C4
SET 'TYPE' AND 'BYTE' HEADINGS INTO POSITION
D4
GET TYPE AND BYTE FOR FIRST ENTRY WHOSE LENGTH AND NO. JUST PRINTED
E4
EACH ENTRY PROCESSED TWICE
NO
YES
E5
GET ADDRESS OF NEXT ENTRY
F4
MD600
BEA1
WRITE LINES AS FORMATTED
G4
ALL MODIFIERS PROCESSED
NO
YES
B3
H4
RETURN
TO LSTPMD

Chart BD. MD450 and MD500 subroutines

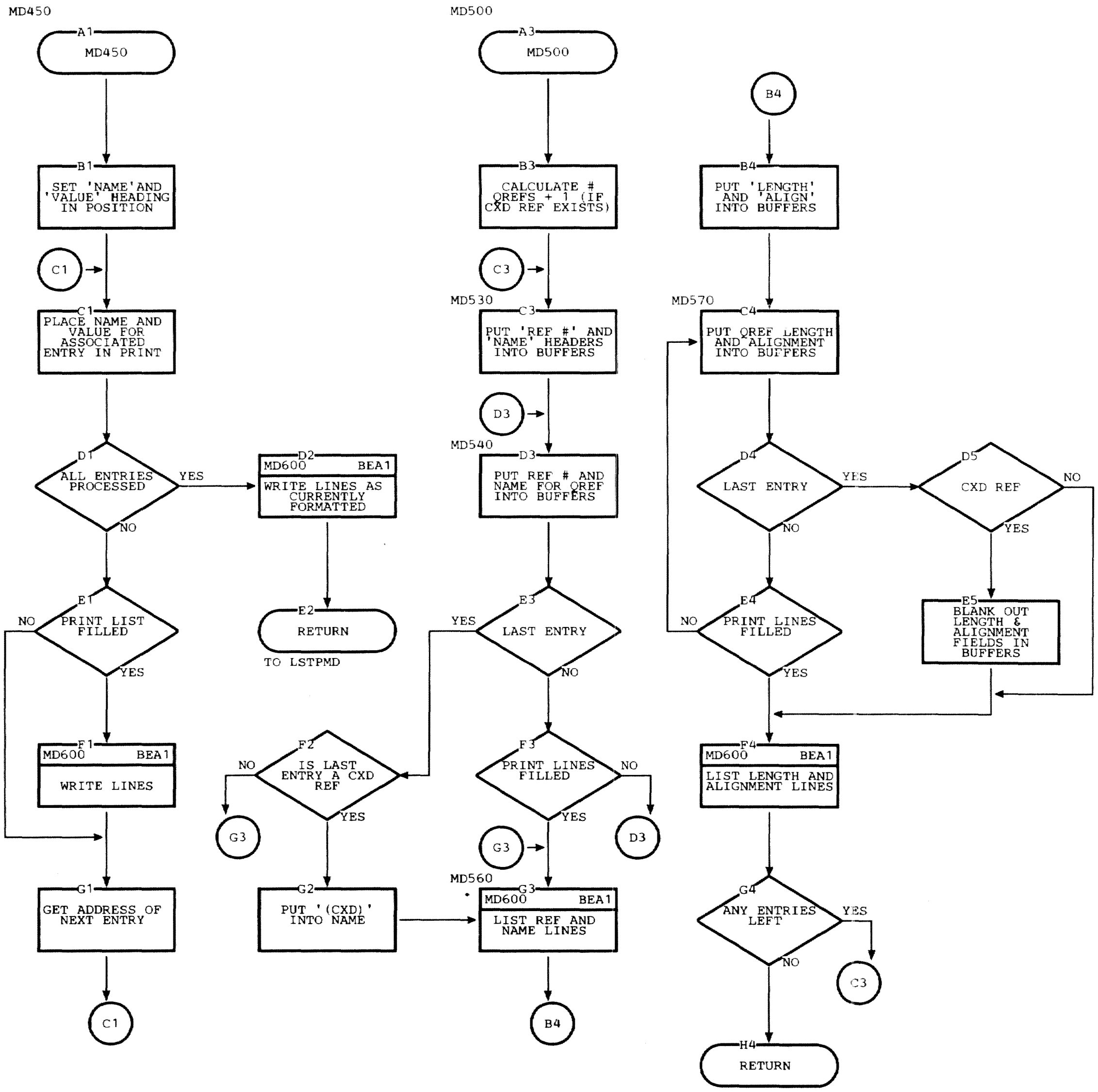

MD450
A1
MD450
B1
SET 'NAME'AND 'VALUE' HEADING IN POSITION
C1
C1
PLACE NAME AND VALUE FOR ASSOCIATED ENTRY IN PRINT
D1
ALL ENTRIES PROCESSED
YES
NO
D2
MD600 BEA1
WRITE LINES AS CURRENTLY FORMATTED
E2
RETURN
TO LSTPMD
E1
PRINT LIST FILLED
NO
YES
F1
MD600 BEA1
WRITE LINES
G1
GET ADDRESS OF NEXT ENTRY
C1
MD500
A3
MD500
B3
CALCULATE # QREFS + 1 (IF CXD REF EXISTS)
C3
MD530
C3
PUT 'REF #' AND 'NAME' HEADERS INTO BUFFERS
D3
MD540
D3
PUT REF # AND NAME FOR QREF INTO BUFFERS
E3
LAST ENTRY
YES
NO
F2
IS LAST ENTRY A CXD REF
NO
YES
G3
G2
PUT '(CXD)' INTO NAME
F3
PRINT LINES FILLED
NO
YES
D3
G3
MD560
G3
MD600 BEA1
LIST REF AND NAME LINES
B4
B4
B4
PUT 'LENGTH' AND 'ALIGN' INTO BUFFERS
MD570
C4
PUT QREF LENGTH AND ALIGNMENT INTO BUFFERS
D4
LAST ENTRY
YES
NO
D5
CXD REF
NO
YES
E5
BLANK OUT LENGTH & ALIGNMENT FIELDS IN BUFFERS
E4
PRINT LINES FILLED
NO
YES
F4
MD600 BEA1
LIST LENGTH AND ALIGNMENT LINES
G4
ANY ENTRIES LEFT
YES
NO
C3
H4
RETURN

Chart BE. MD600 subroutine

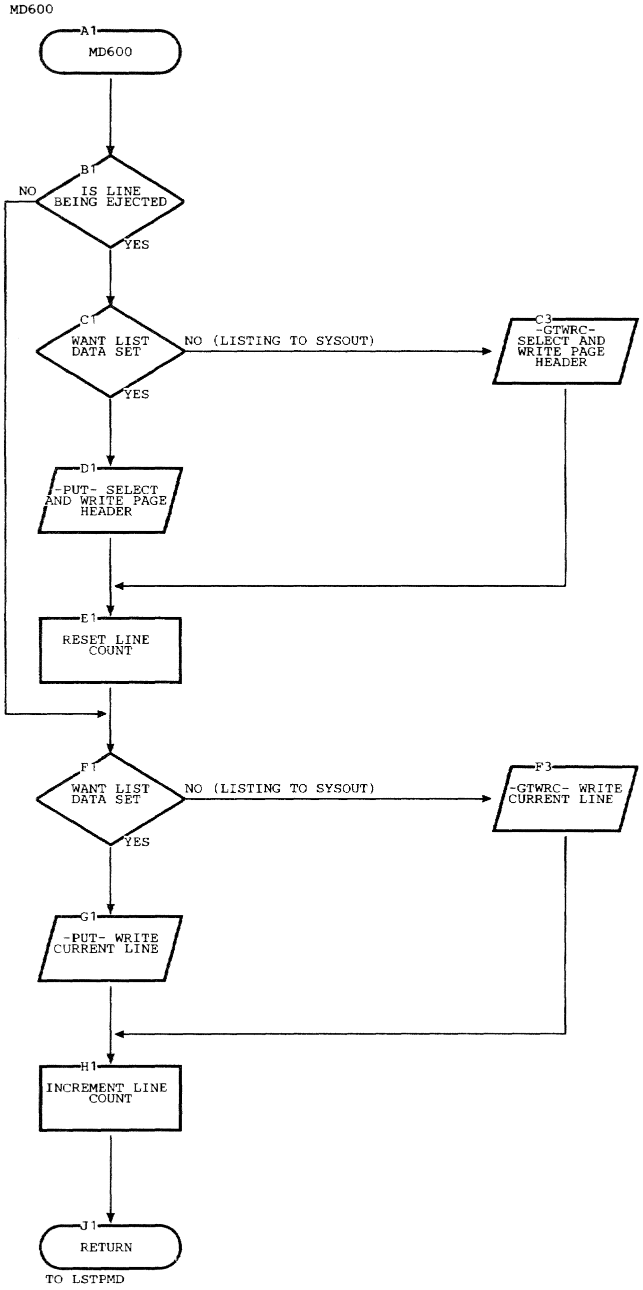
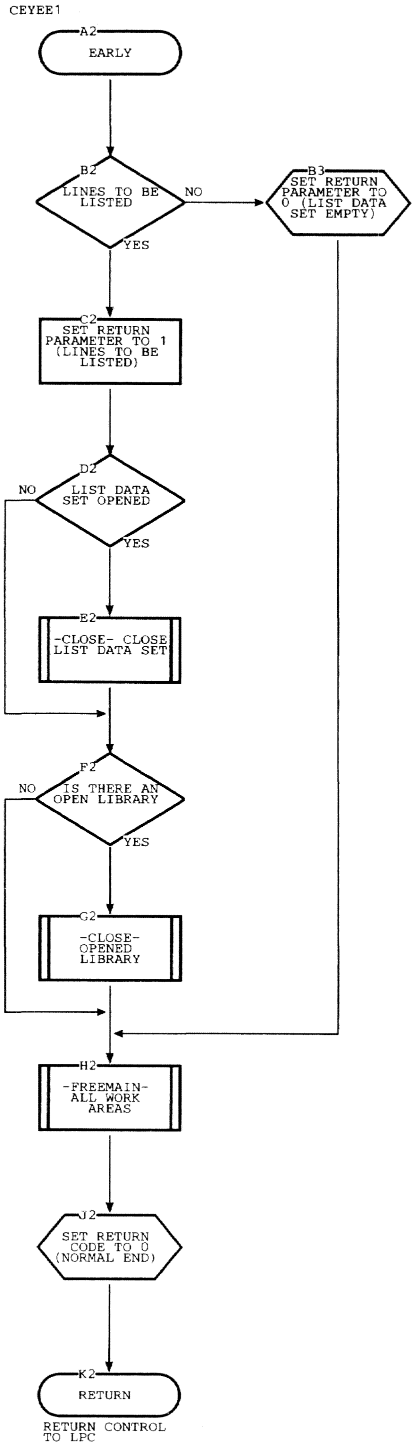


Chart CA. EARLY END - Early-End routine


CEYEE1
A2
EARLY
B2
LINES TO BE LISTED
NO
YES
B3
SET RETURN PARAMETER TO 0 (LIST DATA SET EMPTY)
C2
SET RETURN PARAMETER TO 1 (LINES TO BE LISTED)
D2
LIST DATA SET OPENED
NO
YES
E2
-CLOSE- CLOSE LIST DATA SET
F2
IS THERE AN OPEN LIBRARY
NO
YES
G2
-CLOSE- OPENED LIBRARY
H2
-FREEMAIN- ALL WORK AREAS
J2
SET RETURN CODE TO 0 (NORMAL END)
K2
RETURN
RETURN CONTROL TO LPC

LINKAGE EDITOR ROUTINE DIRECTORY

The TSS/360 linkage editor consists of a single object module, CEYTS. It includes one prototype control section, CEYPSC, and one CSECT, CEYTS1. The routines described in this PLM are all in CEYTS1.

Table 5 provides a cross-reference between entry points to routines, routine names, and flowcharts.

Each routine is assigned an entry point of the form CEYxx, where xx are alphabetic characters identifying the routine. Within a routine, names are of the form xxnnn, where n is numeric. For example, the INCLUDE statement routine is entered at CEYIC. A typical instruction is labeled IC200.

Except for the external entry points CEYIA1, CEYOP1, and CEYEE1, all entry points in Table 5 are internal entry points.

Table 5. Linkage editor routine directory

Routine Label or Entry Point Name	Name of Routine (Short name; long name)	Flowchart
CEYBR	BRING; Bring PMD, Text and ISD from Library	AN
CEYCL	CLEANUP; Cleanup Final Module	AH
CEYCO	COMBINE; COMBINE Statement Processor	AC
CEYCT	COLLECT; Collect Common Requirements	AI
CEYCX	APENCX; Append Complex RLD	AM
CEYDN	DELNAME; Delete Entry Point Name	AE
CEYEE CEYEE1	EARLY; Early End Processor	CA
CEYEN	END; END Statement Processor	AG
CEYER	ERROR; Error Message Processor	AO
CEYEX	APENEX; Append External RLD	AM
CEYGA	GTCSAD; Get CSD Table Addresses	AQ
CEYGC	GETCSD; Locate Control Section Dictionary	AP
CEYIA CEYIA1	INANAL; Input/Analyze Routine	AA
CEYIC	INCLUDE; INCLUDE Statement Processor	AB
CEYIN	APENIN; Append Internal RLD	AM
CEYLK	LINK; Link Modules Subroutine	AJ
CEYLP	LSTPMD; List PMD	BB
CEYOP CEYOP1	OUTPUT; Output Processor	BA
CEYRN	RENAME; RENAME Statement Proceeesor	AE
CEYSC	SCAN; Scan Subroutine	AQ
CEYTR	TRAITS; TRAITS Statement Processor	AF
CEYXR	EXTREF; External Reference Search Subroutine	AP

COMMON AREAS (STORAGE AREAS, TABLES, AND LISTS)

The linkage editor contains internal storage areas, tables, and lists that are used by more than one processor. The linkage editor also makes repeated references to a module's PMD, text, and ISD. The PMD and ISD formats are described in detail in Appendixes A and B.

COMMON INTERNAL STORAGE AREAS

Work Areas

Three work areas (WORKC1, WORKC2, and WORKT) are used by the COMBINE statement processor and are obtained via GETMAIN. WORKA is a 2000-word work area which resides in the PSECT. It is used by the COMBINE routine and the Append RLD subroutine to hold a combined RLD (Relocation Dictionary) or a complex RLD to be linked.

TEMP Storage

TEMP is a temporary storage area that is used for passing parameters to and from subroutines. TEMP is in the PSECT. (The PSECT is described at the end of this section.)

COMMON INTERNAL TABLES AND LISTS

The tables and lists in this subsection are presented in alphabetic order.

Exclusion Table (EXCLUD)

The EXCLUD table (Figure 8) is generated and used by the form-3 INCLUDE statement. It is also used by the END statement processor. It is a variable-length table, which consists of 2-word entries representing the alphameric names of user-supplied external references that are not to be resolved by linking of modules. Two pointers preface the table: the first pointer is the displacement in bytes to the first name in the table; the second pointer is the displacement from the head of the table to the first vacant position in the table.

While an INCLUDE statement is being processed, only those entries in the table that were mentioned in the statement are used.

When the END statement processor is in operation, the entire EXCLUD table is accessed. The table is in the PSECT.

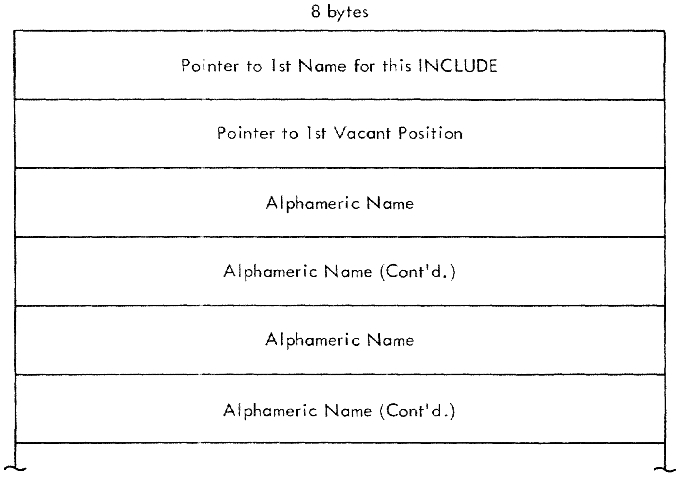

8 bytes
Pointer to 1st Name for this INCLUDE
Pointer to 1st Vacant Position
Alphameric Name
Alphameric Name (Cont'd.)
Alphameric Name
Alphameric Name (Cont'd.)

Figure 8. Exclusion table (EXCLUD) format

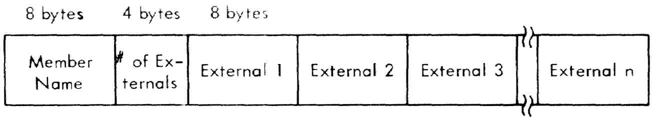


Figure 9. External name list format

External Name List (NAMES)

The External Name List (NAMES) is generated by the APENDF (Append Definition Table) subroutine as each module is linked. It contains the alphameric names of all external definitions appearing in the output PMD. It is used during output processing, at which time the list location is delivered to LPC as part of the output parameters.

The format (Figure 9) consists of the 8-byte alphameric member name assigned to the module, followed by a 4-byte field containing the number of external definitions. This is followed by the list of 8-byte external names. All 8-byte fields are left-justified and filled with blanks. This area is obtained via GETMAIN.

Hash Table (HASHTB)

HASHTB is used to direct the search in the definition tables of the output module whenever references are chained to a definition, or when names are checked for existence in a definition table. HASHTB is updated each time a module is linked by the LINK subroutine. The hashing algorithm produces a value representing the relative position in the hash table of a pointer into the module. Thus, HASHTB consists of

1-word pointers into the module. HASHTB is 127 words long and is contained in the PSECT.

The hash value is derived in the following manner:

1. The first four characters of the name are "exclusively ORed" with the last four characters of the name.

2. The result is divided by 127 and the remainder is multiplied by 4 to get the displacement in bytes from the base of the hash table to the first link in the proper hash chain. Figure 10 shows the format of HASHTB.

Rename/Combine Table (RCTBL)

An entry in the RCTBL table (Figure 11) is generated by the RENAME statement processor for each renamed control section and by the COMBINE statement processor for each combined control section. The table varies in length and contains entries that are to be placed in the ISD when the control section is linked to the output module.

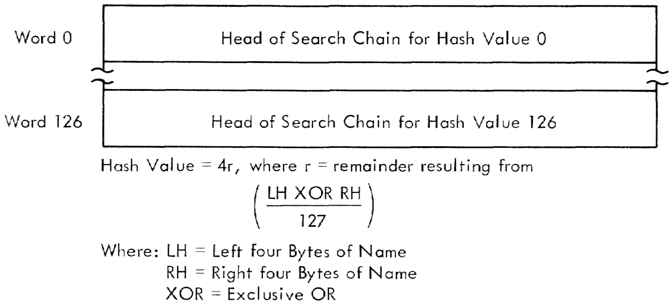
The table is used by the UPISD (Update ISD) subroutine, which removes entries from the table and places them in the composite ISD directory. The table is reinitialized for each module linked.

Stack Table (STACK)

The STACK table (Figure 12) holds COMBINE, RENAME, or TRAITS statements, as received from LPC, until receipt of a form-1 INCLUDE statement. The COMBINE, RENAME, and TRAITS statement processors place entries in STACK. STACK varies in length and is in the PSECT. The following example shows statements entered in the STACK table.

Example Statements:

```
RENAME    NAME1(NAME2)
TRAITS    CSECT1(PUBLIC)
```


Word 0
Head of Search Chain for Hash Value 0
Word 126
Head of Search Chain for Hash Value 126

Hash Value = 4r, where r = remainder resulting from $\left(\frac{\text{LH XOR RH}}{127}\right)$

Where: LH = Left four Bytes of Name
RH = Right four Bytes of Name
XOR = Exclusive OR

Figure 10. Hash table (HASHTB) format

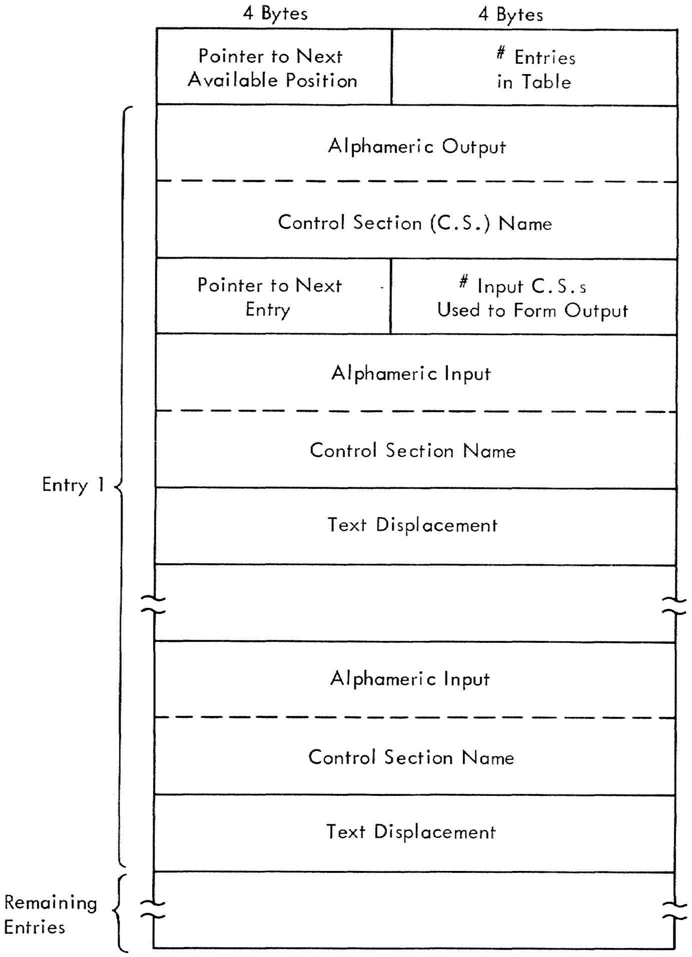


Figure 11. Rename/combine table (RCTBL) format

Stack Table Format

1 Byte	1 Byte	1 Byte	1 Byte
R	F	N	A
M	E	b	N
A	M	E	1
(	N	A	M
E	2	)	b
T	R	A	I
T	S	b	C
S	E	C	T
1	(	P	U
B	L	I	C
)	b		

Figure 12. Stack table (STACK) format

Table 6. Data references by routines

Routine Name	RCTBL	HASHTB	STACK	EXCLUD	NAMES	PMD		Text		ISD	
						Input	Output	Input	Output	Input	Output
INANAL	X	X	X	X	X						
INCLUDE			X	X		X	X	X	X	X	X
COMBINE	X		X			X		X			
RENAME	X	X	X			X					
TRAITS			X			X					
END				X			X			X	X
LINK	X	X			X	X	X	X	X		X
APENCX, APENEX, APENIN						X	X				
BRING						X		X		X	
ERROR											
EXTREF							X				
GETCSD						X	X				
GTCSAD						X	X				
SCAN											
OUTPUT							X				
EARLY											

REFERENCE TO TABLES AND LISTS

Table 6 shows the tables and lists referred to by linkage editor routines.

PSECT ORGANIZATION

A general description of the contents of the linkage editor PSECT is outlined in Table 7. The following breakdown will help in understanding the organization.

1. PSECT Storage

   This area is always covered by register 13 and contains the following items, grouped according to their particular function:

   a. Register save area.

   b. Address constants (adcons) for the entry points to all routines internal to the linkage editor. The name fields are prefixed by the letter A.

   c. Adcons for the PSECT tables. The name fields are prefixed by the letter A.

   d. Constants that are used by more than one routine.

   e. Register save areas; in general, the name field is prefixed by the letter Z.

   f. Switches; in general the name field is suffixed by the letters SW.

   g. Pointers (other than those that are maintained in general registers).

h. Counters.

i. Miscellaneous storage, including various save areas, parameter lists for the GETLINE and PUTDIAG routines (PARGET and PUTPAR), parameter list for the PMD list routine (LSPAR), storage area for diagnostic messages (ERBUFF), initial values for GETMAIN macro instructions, and the skeleton DCB for accessing VPAM modules.

2. PSECT Tables

This area contains the various tables used by the linkage editor. Adcons for these tables are contained in the PSECT storage area. The tables are described earlier in this section, under the heading "Common Internal Tables and Lists."

Table 7. PSECT organization

Label at Beginning of Area	Length in bytes	Description
SAVE	76	Save area.
PATCH	200	Area for patches.
ACEYER	80	Address constants for routines.
APMD	100	Address constants for storage areas.
ADB1	16	Address constants for GETLINE and PUTDIAG.
LEBLNK	36	Miscellaneous constants.
ZLE1	1,188	Temporary register save areas.
FTNMAIN	60	Various switches.
CONSWA	148	Pointers.
TOPGCT	60	Counters.
CSIZE	168	Miscellaneous storage.
PARGET	40	Parameter list for GETLINE.
PUTPAR	20	Parameter list for PUTDIAG.
TEMP	32	Temporary save areas for inter-routine parameters.
LBSLST	36	Parameter list for LIBESRCH macro.
USER	28	User information from FIND or VPAM library POD (describes modules' PMD, text, ISD location).
COTBL	24	Table of addresses of CSD tables for control sections being combined.
LSPAR	20	Parameter list for LSTPMD.
ERBUFF	132	Diagnostic message area.
CINPMD	76	Constants and page number for GETMAIN and FREEMAIN requests.
LBDCB	(depends on DCB expansion)	DCB macro, used for getting modules from libraries.
TABLE	24	Table for GTCSAD routine (addresses of CSD tables).
STMENT	257	Area for current linkage editor control statement.
MDWK	544	Storage for PMD listing routine.
(A macro-generated label)	(depends on macro used)	Macro expansions into PSECT.
FTNSAVE	512	Save area for FORTRAN header.
MSGDIA	3,400	Table of diagnostic messages.
ERTBL	400	Table of addresses for diagnostic messages.
WKPGT	1,024	Work area to build VMPT.
COWKEMPT	512	Work area for external reference RLD modifier pointers.
COWKIMPT	512	Work area for internal reference RLD modifier pointers.
STACK	24,000	Stack area for control statements.
EXCLUD	8,008	External reference exclusion table built by form-3 INCLUDE processor.

GENERAL DEBUGGING AIDS

The system programmer (authority code O or P) may obtain dynamic dumps of the linkage editor through use of the TSSS facilities. See Time Sharing Support System, GC28-2006. TSS/360's program control system (PCS) is available to all users; to use PCS, however, the user must have a copy of the linkage editor with the PUBLIC attribute removed. See Command System User's Guide, GC28-2001, for how to use PCS commands.

In debugging and checkout, the following points may be useful to consider:

1. Dynamic dumps of each input PMD, text, and ISD may be taken in the BRING subroutine (CEYBR) following the GET macro instruction.

2. Dynamic dumps of the output module as it develops following each INCLUDE statement may be taken in the INCLUDE statement processor (CEYIC) upon return from the LINK subroutine; that is, after the INVOKE ACEYLK instructions.

3. Register 8 is a pointer to the current character position in the statement operand.

4. Register 11 is used for local cover.

5. Register 13 is the PSECT storage cover register.

6. Register 14 contains the subroutine return address.

7. Register 15 contains the subroutine entry point address (uses for the other registers are shown in Table 8).

8. Other clues can be gained from the counters, switches, pointers, etc. in the PSECT (see Table 7 in Section 5).

9. Limitations exist for object module size and number of linkage editor statements (see Table 9).

Table 8. Register usage

Category of Registers	Label	Register	Use
Parameter Registers			
	P0	0	Parameters, General Use
	P1	1	Parameters, General Use
	P2	2	Parameters, General Use
	P3	3	Parameters, General Use
	P4	4	Parameters, General Use
	P5	5	Parameters, General Use
Volatile Registers			
	V6	6	General Use
	V7	7	General Use
Common Registers			
	C8,RSTA	8	Statement Pointer
	C9,RPMD	9	Input PMD Pointer
	C10,RTSD	10	Output PMD Pointer
	C11	11	Local Cover Register
	C12	12	General Use
Linkage Registers			
	L13,RPSC	13	PSECT COVER
	L14	14	Subroutine Return Register
	L15	15	Branch Register

Table 9. Size limitations and requirements

Limitation on	Maximum
Input or output object module	
PMD	256 pages
Text	256 pages
ISD (Combined length of all input ISDs and the generated output ISD)	512 pages
Number of statements	
INCLUDE	No limit
RENAME, COMBINE, TRAITS	64 stacked
END	1
Statement length (excluding extraneous blanks and continuation characters)	256 characters
Virtual storage requirements for linkage editor processing (approximate): 28 + p(m + 3) pages where: p = the average number of pages per object module, including PMD, text, and ISD, input to linkage editor. m = the total number of input object modules.	

The output from an assembler, compiler, or the linkage editor is known as an object module. The object module is composed of a program module dictionary (PMD), text, and possibly an internal symbol dictionary (ISD).

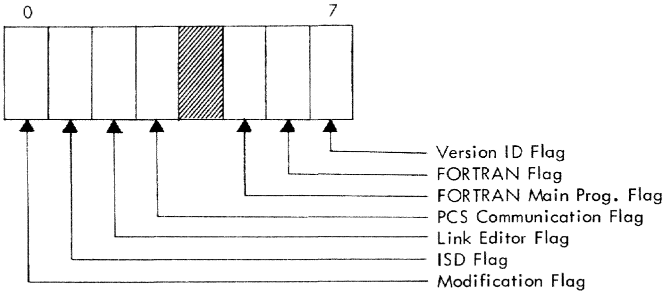
Each PMD consists of one PMD heading plus as many control section dictionaries (CSD) as there are control sections in the module. Address pointers in the PMD are relative to the beginning of the PMD, except where otherwise specified. Some fields in the PMD are filled in by the dynamic loader. These are left zero by the language processor or linkage editor. The PMD format is shown in Figure 13.

PMD HEADING

1. Length of PMD in bytes - This length does not include the PMD preface.

2. Diagnostic code (1 byte) - The diagnostic code indicates the highest level diagnostic message encountered during generation of the module by the language processor.

3. Flags (1 byte) - The flag bits are numbered from left to right starting with zero and are defined as follows:



Bit 0 - System module; was modified by other than a language processor.

Bit 1 - Module has an ISD. This bit is set by the processor that creates the PMD.

Bit 2 - Module was produced by linkage editing. This bit is set by the linkage editor.

Bit 3 - PCS is to be called before module is dynamically unlinked. This bit is set by PCS.

Bit 4 - Not used.

Bit 5 - Module was produced by the FORTRAN compiler.

Bit 6 - FORTRAN module is a main program, not a SUBROUTINE FUNCTION or BLOCK DATA subprogram.

Bit 7 - Version ID indicator. If this bit is set, the module version ID is to be interpreted as a 64-bit binary number which is the creation date of the module. If this bit is not set, the version ID is eight alphameric EBCDIC characters.

4. Length of PMD heading - This is the length in bytes of the PMD heading.

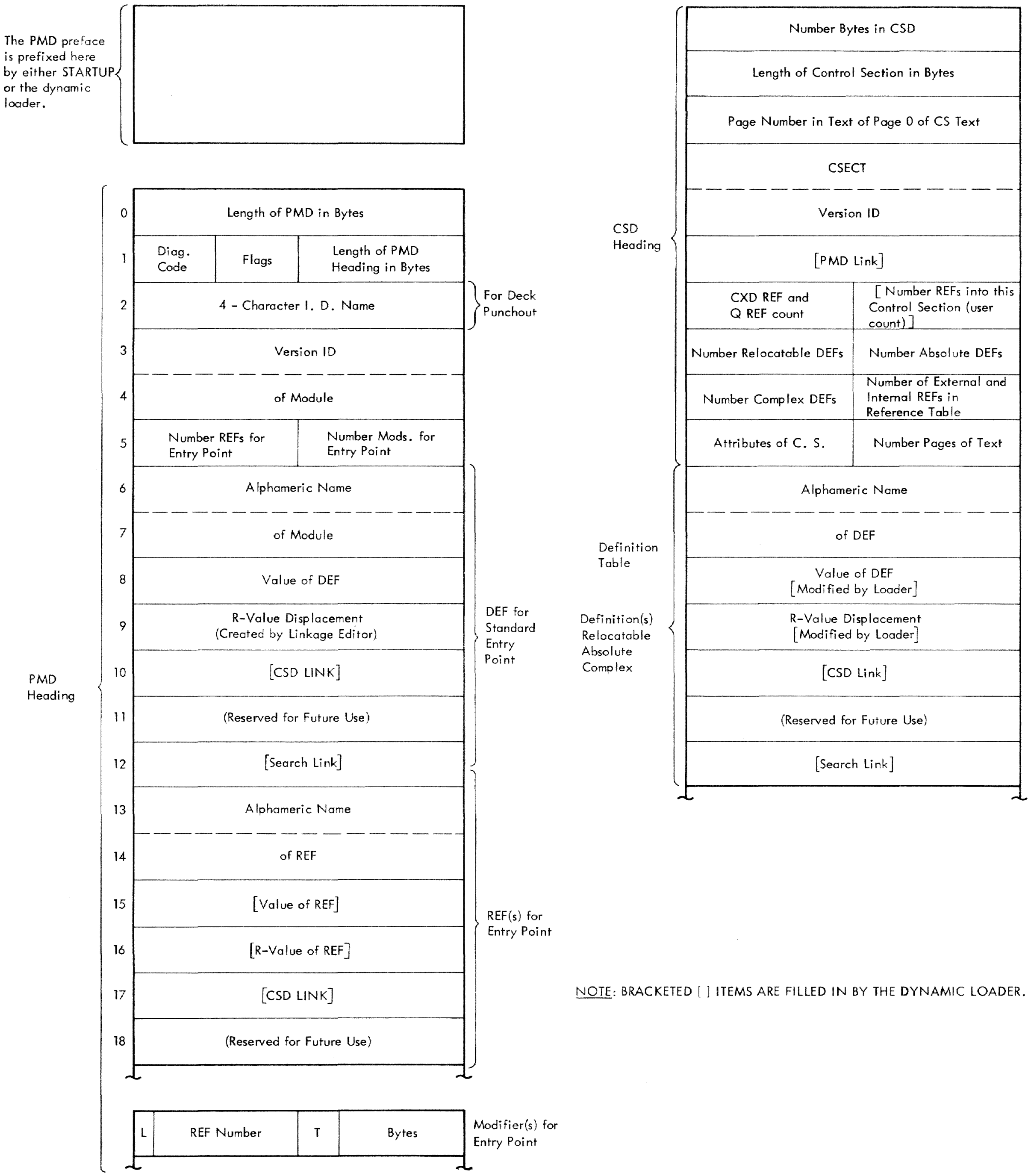
5. 4-Character I.D. name - The 4-character I.D. name is supplied by the user to serve as deck identification if the module is punched into cards. This field is currently unused.

6. Version I.D. - See item 3 (Bit 7 discussion) for interpretation of version I.D.

7. Number of REFs for the standard entry point - The DEF for the standard entry point is always treated as a complex DEF. This field contains the number of REFs. It may be zero.

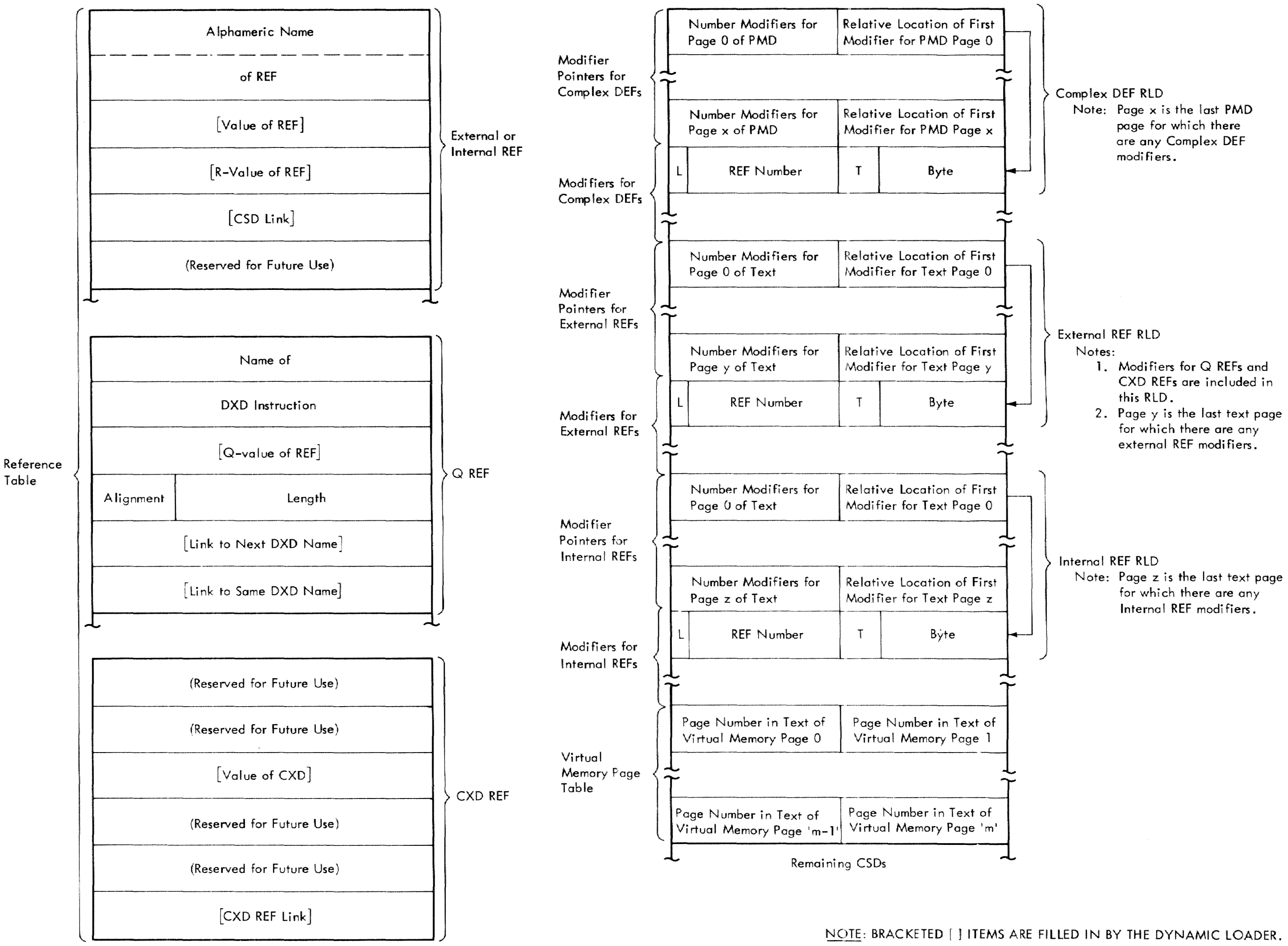
8. Number of modifiers for the standard entry point - This field contains the number of modifiers that are to be used to compute the DEF for the standard entry point.

9. DEF for standard entry point - This 7-word entry describes the DEF for the standard entry point of the module. It has the same form as a DEF entry within a CSD. The standard entry point DEF for the module is considered to belong to the first PSECT of the module and is treated the same as a complex DEF whose ENTRY statement appears within that PSECT. If no PSECT is declared, the standard entry point is associated with the first CSECT instead. This DEF entry contains the following subfields which are described under the heading "Definition Table" in this appendix.


The PMD preface is prefixed here by either STARTUP or the dynamic loader.
PMD Heading
0
Length of PMD in Bytes
1
Diag. Code
Flags
Length of PMD Heading in Bytes
2
4 - Character I. D. Name
For Deck Punchout
3
Version ID
4
of Module
5
Number REFs for Entry Point
Number Mods. for Entry Point
6
Alphameric Name
7
of Module
8
Value of DEF
9
R-Value Displacement (Created by Linkage Editor)
10
[CSD LINK]
11
(Reserved for Future Use)
12
[Search Link]
DEF for Standard Entry Point
13
Alphameric Name
14
of REF
15
[Value of REF]
16
[R-Value of REF]
17
[CSD LINK]
18
(Reserved for Future Use)
REF(s) for Entry Point
L
REF Number
T
Bytes
Modifier(s) for Entry Point
CSD Heading
Number Bytes in CSD
Length of Control Section in Bytes
Page Number in Text of Page 0 of CS Text
CSECT
Version ID
[PMD Link]
CXD REF and Q REF count
[Number REFs into this Control Section (user count)]
Number Relocatable DEFs
Number Absolute DEFs
Number Complex DEFs
Number of External and Internal REFs in Reference Table
Attributes of C. S.
Number Pages of Text
Definition Table
Definition(s) Relocatable Absolute Complex
Alphameric Name
of DEF
Value of DEF [Modified by Loader]
R-Value Displacement [Modified by Loader]
[CSD Link]
(Reserved for Future Use)
[Search Link]

NOTE: BRACKETED [ ] ITEMS ARE FILLED IN BY THE DYNAMIC LOADER.

Figure 13. Program module dictionary format (part 1 of 2)


Alphameric Name
of REF
[Value of REF]
[R-Value of REF]
[CSD Link]
(Reserved for Future Use)
External or Internal REF
Name of
DXD Instruction
[Q-value of REF]
Alignment
Length
[Link to Next DXD Name]
[Link to Same DXD Name]
Q REF
Reference Table
(Reserved for Future Use)
(Reserved for Future Use)
[Value of CXD]
(Reserved for Future Use)
(Reserved for Future Use)
[CXD REF Link]
CXD REF
Modifier Pointers for Complex DEFs
Number Modifiers for Page 0 of PMD
Relative Location of First Modifier for PMD Page 0
Number Modifiers for Page x of PMD
Relative Location of First Modifier for PMD Page x
Modifiers for Complex DEFs
L
REF Number
T
Byte
Complex DEF RLD
Note: Page x is the last PMD page for which there are any Complex DEF modifiers.
Modifier Pointers for External REFs
Number Modifiers for Page 0 of Text
Relative Location of First Modifier for Text Page 0
Number Modifiers for Page y of Text
Relative Location of First Modifier for Text Page y
Modifiers for External REFs
L
REF Number
T
Byte
External REF RLD
Notes:
1. Modifiers for Q REFs and CXD REFs are included in this RLD.
2. Page y is the last text page for which there are any external REF modifiers.
Modifier Pointers for Internal REFs
Number Modifiers for Page 0 of Text
Relative Location of First Modifier for Text Page 0
Number Modifiers for Page z of Text
Relative Location of First Modifier for Text Page z
Modifiers for Internal REFs
L
REF Number
T
Byte
Internal REF RLD
Note: Page z is the last text page for which there are any Internal REF modifiers.
Virtual Memory Page Table
Page Number in Text of Virtual Memory Page 0
Page Number in Text of Virtual Memory Page 1
Page Number in Text of Virtual Memory Page 'm-1'
Page Number in Text of Virtual Memory Page 'm'
Remaining CSDs

NOTE: BRACKETED [ ] ITEMS ARE FILLED IN BY THE DYNAMIC LOADER.

Figure 13. Program module dictionary format (part 2 of 2)

a. Alphameric name of module

b. Value of DEF

c. R-Value displacement

d. CSD link

e. Reserved word

f. Search link

The alphameric name is also the name of the module.

10. REF(s) for entry point - These have the same form and function as the REFs described in the CSD discussion below.

11. Modifier(s) for entry point - These have the same form and function as the modifiers for the RLD for complex definitions described in the CSD discussion below, except that they apply to the standard entry point DEF.

CONTROL SECTION DICTIONARY (CSD)

The control section dictionary has the following components:

1. CSD heading

2. Definition table

3. Reference table

4. Relocation dictionaries (RLDs)

5. Virtual memory page table (VMPT)

CSD HEADING

1. Number of bytes in CSD - This field specifies the length of the control section dictionary in bytes.

2. Length of control section in bytes - This specifies the virtual memory span of the control section. The length of the virtual memory page table is derived from this length. For example, if the length of the control section is 8192, the virtual memory page table will contain two pages; but if the length is 8193 bytes, the virtual memory page table will contain three pages. This value will be equal to the highest location counter value assigned by the language processor, plus one.

3. Page number in text of page 0 of CSECT text - The text for each control section in the module occupies an integral number of pages in its resident data set. The text pages for all control sections in a module are contiguous. This number is the page number, relative to the first page of text for this module, of the first page of text for this CSECT. (Numbering begins with 0.)

4. Version I.D. - This is a 64-bit binary number which is the creation date of the control section expressed as the number of microseconds that have elapsed from March 1, 1900, until the time of CSECT creation. This number is changed by the linkage editor when CSECT combining occurs.

5. PMD link - The PMD link is filled in by STARTUP or the dynamic loader. It points to the beginning of the PMD preface.

6. Whether CXD REF exists and number of Q REFs. Bits from left to right contain:

   Bit 0 - Set to 0 if no CXD REF exists; set to 1 if a CXD REF exists. (Only one CXD REF is possible.)

   Bit 1 - Not used.

   Bits 2 through 14 - Number of Q REFs (contains all zeros if none).

7. Number of implicit references to this control section (user count) - This is a count of the number of REF entries that refer to this control section and are linked to this CSD through their CSD link. It is computed by the loader. It includes both external and internal references. This number is arbitrarily set to X'7FFF' by STARTUP for each CSECT in initial virtual memory to prevent unloading of IVM modules.

8. Number of relocatable definitions - This is the number of relocatable definitions in the definition table. It is always at least one, namely, the control section name DEF.

9. Number of absolute definitions - This is the number of absolute definitions in the definition table. It may be zero.

10. Number of complex definitions - This is the number of complex definitions in the definition table. It may be zero.

11. Number of references from this CSD - This is the sum of external and internal references in the reference table. It may be zero.

12. Attributes - This halfword has one bit set for each attribute possessed by the control section. Currently defined attributes are shown below. Bits are numbered from left to right starting with 0.

    a. Public name (Bit 0 on) - This is used only by the dynamic loader to specify nonblank control sections whose names appear in the SDST (Shared Data Set Table). The first such control section will appear in the SDST under the module name. A section may be indicated as both having a public name and rejected.

    b. CSD has been allocated storage (Bit 1 on) - Set by the dynamic loader, if it applies.

    c. PCSA (CGCCT) called for this CSD (Bit 2 on) - Set by the dynamic loader, if it applies.

    d. Public storage assigned by CONNECT (CZCGA7) (Bit 3 on) - Set by the dynamic loader, if it applies.

    e. Bits 4 and 5 are not used.

    f. Common CSECT rejected (Bit 6 on) - The dynamic loader sets this flag to indicate to the program control system that the CSECT was rejected as a common CSECT that was already loaded in another module.

    g. Q REF count validity (Bit 7 on) - The assembler, PL/I compiler, and the linkage editor set this flag to indicate that the count of Q REFs in field TDYCQR is valid. If bit 7 is off, the count is not valid.

h. System (Bit 8 on) - Any external symbol that appears in a CSECT with the system attribute cannot be referenced by a user program unless the symbol begins with "SYS." Conversely, no reference from a control section with a system attribute may be to a "user" symbol.

i. Privileged (Bit 9 on) - A CSECT with a privileged attribute is assigned storage key C which provides fetch as well as store protect. This attribute overrides the read-only attribute. Anything in a privileged CSECT may be referenced only when the PSW key is zero.

j. Common (Bit 10) - A common section is a control section common to all modules in which it is declared. Common sections are more fully discussed in the Linkage Editor and Assembler Language SRLs.

   Common sections are of two types:

   (1) Named common sections (those with a name not all blanks). These are treated as fixed-length sections.

   (2) Blank common sections, whose name consists of eight blanks. FORTRAN blank common is assigned the variable and common attributes by the FORTRAN compiler.

   The treatment of blank common sections differs from that of blank non-common sections. Control section rejection is instituted between blank common sections of different modules whereas blank non-common sections of different modules are treated as independent control sections. The latter are called unnamed control sections.

k. PSECT (Bit 11 on) - If this bit is set, it causes the dynamic loader to override the system packing indicator and insert this control section as packed.

l. Public (Bit 12 on) - Control sections are not shared by CSECT name alone. A public control section of a module residing in a given data set (library) is shared if another user has access to the same data set and module. CSECTs of a given module need not all be public or non-public. Fixed-length public CSECTs with the same attributes are assigned storage in the same assignment. A public CSECT must not contain relocatable adcons (A-, V-, or R-type).

m. Read-only (Bit 13 on) - Read-only specifies that no data can be stored in the control section. Causes memory protection by means of a storage class B assignment to all pages of the control section. Nonread-only and nonprivileged CSECTs are assigned storage class A.

n. Variable-length (Bit 14 on) - A variable-length control section is of indeterminate length. It will be allocated pages in excess of the length stated in the CSD headers.

o. Fixed-length (Bit 14 off) - A fixed-length control section is a section of fixed length. It will be allocated a fixed number of pages at load time.

13. Number of pages of text - This specifies the number of pages of text for this control section in the data set. It should be noted that this generally does not correspond to the number of pages in the virtual memory page table. It cannot be larger.

DEFINITION TABLE

The definition table contains 7-word entries, one for each external definition in the current control section. Definitions are grouped as relocatable, absolute, and complex in that order. The first definition in the table is the name of the current control section.

A relocatable definition is an external definition whose value may be computed as the sum of the origin of the control section wherein it appears, and a constant that is the symbol's displacement from the section origin.

An absolute definition is an EQU item with an absolute value whose name has been declared an entry point in the CSECT in which the name is defined.

A complex definition is either an EQU item with a complex relocatable value (i.e., containing external symbols) or a simple relocatable definition whose ENTRY statement appeared within a control section other than the section in which it is defined. The definition entry appears within the CSD of the control section that contains the ENTRY statement. (Note that the origin of the same control section is the R-value for the DEF.) The complex DEF is required in this case, with one REF entry that names the control section in which the DEF symbol is actually defined.

Each DEF in the definition table contains the following entries:

1. Alphameric name of DEF - This field contains the 8-character alphameric name of the DEF.

2. Value of DEF - The value of DEF is set by the language processor and is modified by STARTUP or the loader in the case of complex and relocatable definitions. For relocatable DEFs, the value portion of the definition entry contains the displacement value of the symbol relative to the base of its control section. For absolute DEFs, this entry contains the absolute value; for complex DEFs it contains the absolute portion of the DEF value, which may be zero.

3. R-value displacement - The "displacement for R-value" word contains the displacement of the original defining control section origin with respect to the head of the control section within which the definition now appears. This is required to compute valid R-values for control sections which have been combined by linkage editing. In creating the PMD, only the linkage editor will ever produce a nonzero value in this word.

4. CSD link - The CSD link is initially zero. It is filled in by STARTUP or the dynamic loader when the control section is loaded. It is a pointer to the beginning of the CSD in which this DEF appears, provided that neither the DEF nor the control section has been rejected.

5. Reserved for future use.

6. Search link - This field is filled by the HASH SEARCH routine of either the loader or STARTUP. It contains the address of the beginning of the next DEF entry, which hashes to the same value. It contains zero if there are no more DEFs with the same hash value in this chain.

REFERENCE TABLE

The reference table is made up of 6-word entries, one for each external symbol referenced within the control section. Each entry for an external or internal REF contains the following:

1. Alphameric name of REF - This field contains the 8-character alphameric name of the REF.

2. Value of REF - This is filled in by STARTUP or the dynamic loader. It contains the value of the DEF to which the REF refers. If the DEF is undefined, it contains the address of a portion of virtual storage wherein reference is illegal.

3. R-value of REF - This is filled in by STARTUP or the dynamic loader. It contains the virtual storage address of the beginning of the control section wherein the DEF appears. This value is obtained from the "R-value displacement" word of the satisfying DEF entry.

   If the DEF is undefined, this word contains the address of a portion of virtual storage wherein reference is illegal.

4. CSD link - This pointer, initially zero, is filled by STARTUP or the dynamic loader. It points to the beginning of the CSD wherein the DEF that defines this REF appears. If a corresponding DEF could not be found upon the appearance of a REF, the CSD link is to the beginning of the CSD wherein the REF itself appears.

5. Reserved for future use.

Each entry for a Q REF contains:

1. Name of Q REF - This is the 8-character alphameric name of a DXD instruction or of a DSECT instruction referred to in a Q-type address constant.

2. Q-value of REF - This is filled in by the RESOLVE Q-REF routine of the dynamic loader. It contains the displacement of the dummy section defined by the DXD instruction from the beginning of the combined external dummy sections.

3. Alignment, Length - The alignment and length specified by the assembler or another language processor.

4. Link to next DXD name - This is filled in by the Q-CHAIN routine of the

dynamic loader when Q-CHAIN posts the REF on one of the 11 hash chains for Q REFs.

5. Link to same DXD name - This is filled in by the Q-CHAIN routine of the dynamic loader when Q-CHAIN posts the REF on one of the secondary Q REF chains for duplicate name DXDs.

Each entry for a CXD REF contains:

1. For future use.

2. Value of CXD - This is filled in by the EXPLICIT LINK routine of the dynamic loader. It contains the length of the combined external dummy sections for modules currently loaded in the present task.

3. For future use.

4. CXD REF link - This is filled in by the ALLOCATE MODULE routine of the dynamic loader as CXD REFs are chained together.

RELOCATION DICTIONARY (RLD)

Three RLDs appear in each control section dictionary. The three RLDs are:

1. RLD for complex definitions

2. RLD for internal references

3. RLD for external references

Each RLD has the same format consisting of modifier pointers and modifiers. The RLD for complex definitions differs in that pages mentioned in this table are pages of the PMD rather than the text.

Modifier Pointer

Modifier pointers are used to designate the application of modifiers to adcons on appropriate pages of text (or of the PMD for complex DEFs). The first modifier pointer applies to the first page; the second modifier pointer, the second page; etc. For an RLD there always exists at least one modifier pointer. However, there need not necessarily be a modifier pointer for each page of text; the modifier pointers may be ended at the last text page for which there exists any modifier.

The modifier pointers consist of two fields, in the left and right halfwords.

Left half - Number of modifiers of page. This field contains the number of modifiers that apply in this page.

Right half - Location of first modifier for this page. This contains the location in bytes relative to the right half of the pointer itself for the first modifier for this page. If there are none, it points to the location where one would have appeared if there had been one.

A special note should be made of the technique for determining the length of an RLD. If one looks in the right half of the first pointer for an RLD, one finds the location of the first modifier for this page. In the word preceding the first modifier word is the last modifier pointer for the RLD. By adding the location of the right half (of the last pointer) to the contents of the right half (of the last pointer), one gets the beginning of the last set of modifiers. Adding to this four times the number of modifiers in the last set, one gets the end of the RLD.

Modifier

1. L - L (2 bits) is the length in bytes of the adcon to be modified. A value of zero indicates a fullword (4 bytes).

2. Ref number - Reference number (14 bits) is the ordinal number in this CSD's reference table of the reference whose definition value is to be used in modifying the adcon. References are numbered starting with zero.

3. T - T (4 bits) is the operation to be performed in modifying the adcon by the reference value. The values of T currently defined are as follows:

   a. Addition (T = 1) - The definition value of the reference at "Reference Number" is added to the field of L bytes at the location specified by "Byte."

   b. Subtraction (T = 2) - Same as addition, except read "subtracted from" for "added to."

   c. R-value (T = 3) - The "R-value" of the REF is stored into the field of length L at the location specified by "Byte."

   d. Q-value (T = 4) - The "Q-value" of the REF is stored into the field of length L at the location specified by "Byte."

   e. Value of CXD (T = 5) - The value of the CXD instruction is stored into the field of length L at the location specified by "Byte."

4. Byte - Byte (12 bits) is the displacement in bytes (from the origin of its original containing page) of the adcon to be modified. It should be noted that since PMDs are packed to word boundaries, this displacement will be added to an address for complex DEFs which generally is not a page boundary.

RLD for Complex Definitions

The format of these modifiers is as described above under "Modifier." These modifiers apply to the values of complex definitions; that is, the byte addresses in the modifier will be added to the value words of complex DEF entries in the definition table, and the page numbers in the modifier pointers are for pages of the program module dictionary itself.

RLD for Text External Reference

This relocation dictionary is in the same form as described above. It has one pointer for each page of program text up to that text page, which is the last to contain an adcon, and appropriate modifiers for each adcon in the text, which refers to a symbol defined externally to this module, to a DXD symbol, or to a CXD value. The page numbers are based on the first page for this control section, beginning with 0.

RLD for Text Internal Reference

This is identical to RLD for text external reference above, except that the modifiers are to adcons in the text that reference symbols defined within this module, such as control section names. This permits communication between control sections of the same module that may be allocated noncontiguous virtual storage.

VIRTUAL MEMORY PAGE TABLE (VMPT)

This table has a halfword for each page of virtual storage that the CSECT occupies, beginning with page 0 and continuing upward in order.

The contents of each entry will be either:

1. All 1-bits if the corresponding page is empty as a result of a DS or ORG statement.

2. The number of the page in the text relative to the beginning of text for this CS if the page contains code or data.

This table is the means by which the text of the control section is related to the virtual memory (virtual storage) assigned the control section. This is because language processors do not necessarily output a byte of text for each byte of virtual storage assigned; that is, large ORG and DS statements may result in pages of text being skipped.

If, for example, a source program were to begin with

```
ORG  10000
```

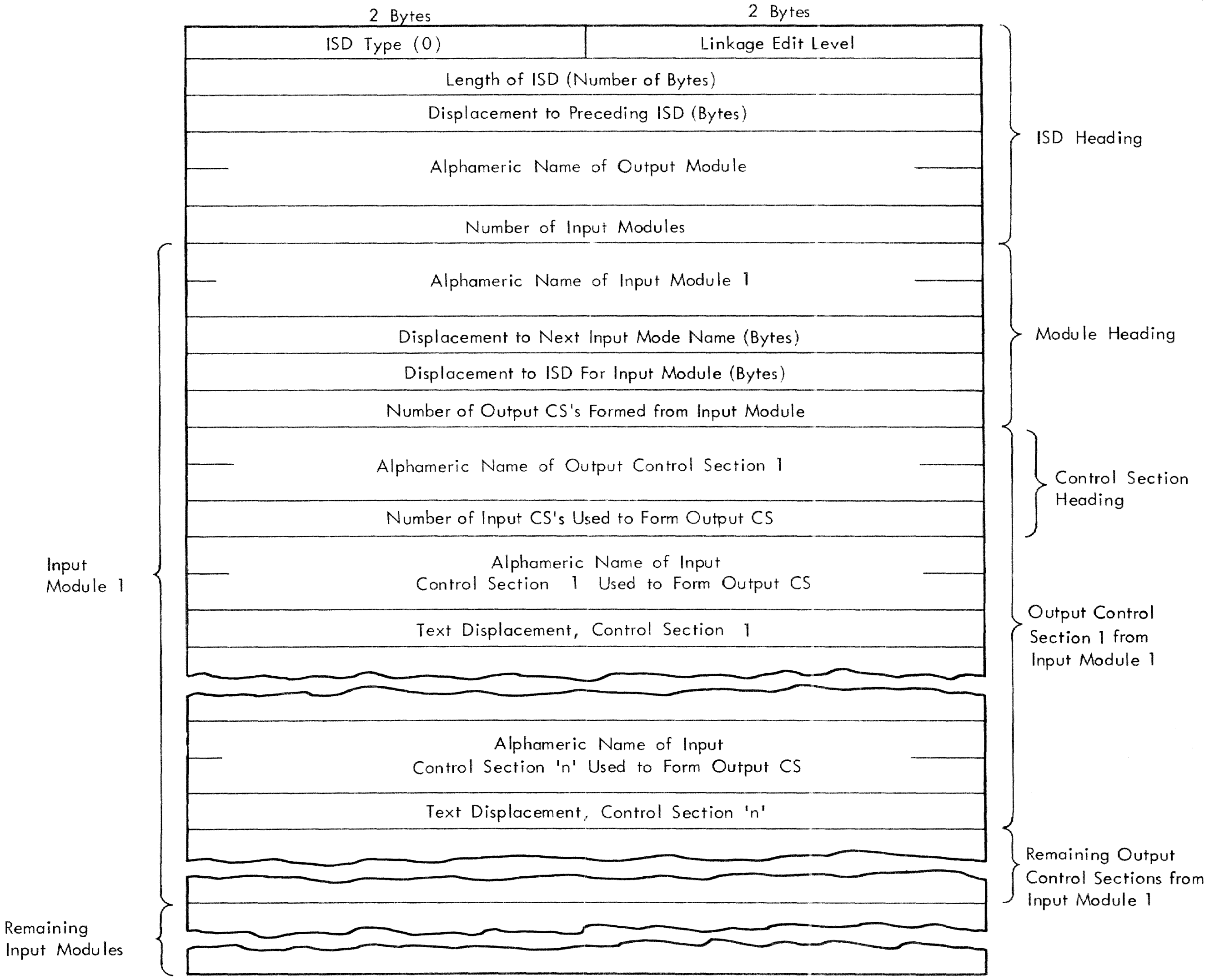
there would be no text output for the first two pages of virtual storage, and the first page of text would correspond to the third page of the user's virtual storage. The first two VMPT entries would be all bits, and the third would contain zero. Within a page, however, the bytes of text correspond directly to the bytes of virtual storage. Thus, in the example above, the first page of text would represent virtual storage locations 8192-12,287, and the first 1808 bytes of the page of text would be vacant (10,000-8192 = 1808). The pages of text always begin on page boundaries within the text module.

In TSS/360, at user option, an internal symbol dictionary (ISD) is built for an object module. The ISD enables the user to later use the commands of the program control system (PCS) for debugging; PCS uses the ISD to find internal symbols.

The ISDs produced by the assembler and FORTRAN compiler differ slightly; the format of each is shown, respectively, in IBM System/360 Time Sharing System: Assembler Program Logic Manual, GY28-2021, and FORTRAN Compiler Program Logic Manual, GY28-2019. (The PL/I compiler does not produce an ISD.)

THE COMPOSITE ISD

The linkage editor produces an object module containing a composite ISD. This composite ISD contains all ISDs from input modules, retained just as they were upon input and chained in the order in which the input modules were included. In addition, the linkage editor produces a directory pointing to the retained ISDs. The format of this directory is shown in Figure 14. The directory and the retained input ISDs constitute the composite ISD.



2 Bytes | 2 Bytes
ISD Type (0) | Linkage Edit Level
Length of ISD (Number of Bytes)
Displacement to Preceding ISD (Bytes)
Alphameric Name of Output Module
Number of Input Modules
Alphameric Name of Input Module 1
Displacement to Next Input Mode Name (Bytes)
Displacement to ISD For Input Module (Bytes)
Number of Output CS's Formed from Input Module
Alphameric Name of Output Control Section 1
Number of Input CS's Used to Form Output CS
Alphameric Name of Input Control Section 1 Used to Form Output CS
Text Displacement, Control Section 1
Alphameric Name of Input Control Section 'n' Used to Form Output CS
Text Displacement, Control Section 'n'

ISD Heading
Module Heading
Control Section Heading
Output Control Section 1 from Input Module 1
Remaining Output Control Sections from Input Module 1
Input Module 1
Remaining Input Modules

Figure 14. Composite ISD directory format

THE COMPOSITE ISD DIRECTORY

The directory consists of a heading and a list of entries for each input ISD.

DIRECTORY HEADING

1. ISD type (2 bytes) - Contains 0 to indicate this is a linkage-editor produced ISD.

2. Link edit level (2 bytes) - A counter equal to 1 plus the highest linkage edit level value present in any previously generated composite ISD. If there are no previous composite ISDs present, the value is 1.

3. Length of ISD (4 bytes) - The number of bytes in the composite ISD.

4. Pointer to the first composite ISD included (4 bytes) - A displacement in bytes from an input ISD (if present) which was itself the product of a previous linkage edit.

5. Alphameric name of output module (8 bytes).

6. Number of input modules (4 bytes) - The number of object modules used to produce the output module.

ENTRIES FOR EACH INPUT MODULE

The composite ISD directory contains an entry for each input module. The entry, of variable length, relates the control sections produced in the output module to the control sections from input modules. The entries described below are each fullword aligned.

1. Alphameric name of input module (8 bytes).

2. Displacement to next input module name (4 bytes) - The number of bytes from the beginning of this entry to the beginning of the next input module entry. It will be zero if this is the last entry.

3. Displacement to ISD for input module (4 bytes) - The number of bytes from the beginning of this entry to the beginning of the input module's ISD. It will be zero if the module has no ISD.

4. Number of output control sections produced from input modules (4 bytes) - May be less than the number of control sections in the input module. This is the case if any control sections are renamed or combined.

5. Entries for output control sections in this form:

   a. Alphameric name of output control section (8 bytes).

   b. Number of input control sections used to produce the output control section (4 bytes) - Zero if the output control section is merely one of the input control sections. One if the output control section was produced by renaming an input control section. Two or more if the control section was produced by combining two or more input control sections.

   c. Entries for each input control section used to produce the output control section in this form:

      (1) Alphameric name of input control section (8 bytes).

      (2) Displacement to text for input control section (4 bytes) - The number of bytes to the first byte of the input control section's text from the beginning of the output control section's text as produced by the linkage editor. A non-zero value will appear here only for second and subsequent input control sections combined to make the output control section.

COMPOSITE ISDS AS INPUT

Object modules containing composite ISDs (that is, modules that were produced by the linkage editor) may be specified as input to the linkage editor. The resultant composite ISD will note that the particular input ISD was itself a composite ISD.

The message delivered by the ERROR routine is chosen from the following list according to the error code in register 1. The number preceding each message is not part of the message, but is used to relate the message to the error code. Each message will, at the time it is delivered to PUDIAG, be prefaced by:

nnnnnnnbsb***

where nnnnnnn is the line number of the linkage editor control statement, b is blank, and s is either E or W: E denotes a major error (severity level 2) and W denotes a warning message (severity level 0).

Following is a list of the diagnostic messages, with the message numbers used on the flowcharts and appearing in the text.

Message Number	Severity Level	Message Text
1	2	MODULE name IS IN SYSLIB
2	2	MODULE name DOES NOT EXIST
3	2	ILLEGAL OPERATION SYMBOL
4	2	ENTRY NAME name APPEARS IN BOTH CSECT name OF OUTPUT MODULE AND CSECT name OF MODULE name
5	2	ENTRY NAME name ALREADY EXISTS IN CONTROL SECTION name
6	2	EXTERNAL SYMBOL name DOES NOT EXIST
7	0	THE FOLLOWING EXTERNAL REFERENCES ARE UNRESOLVED (followed by list of names)
8	2	CONTROL SECTION name AND name TO BE COMBINED DO NOT HAVE IDENTICAL ATTRIBUTES
9	2	FORM-1 INCLUDE STATEMENT NOT YET GIVEN
10	2	ILLEGAL USE OF NAME name
11	2	ILLEGAL DELIMITER
12	2	$\left\{\begin{array}{l}\text{TEXT}\\ \text{PMD}\\ \text{ISD}\end{array}\right\}$ OF MODULE name EXCEEDS AVAILABLE VIRTUAL MEMORY. CANNOT CONTINUE
13	2	$\left\{\begin{array}{l}\text{TEXT}\\ \text{PMD}\\ \text{ISD}\end{array}\right\}$ OF OUTPUT MODULE EXCEEDS AVAILABLE VIRTUAL MEMORY. CANNOT CONTINUE
14	2	CONTROL SECTION name OF MODULE name DUPLICATES PREVIOUSLY NAMED ENTRY POINT
15	0	WARNING - CONTROL SECTION name OF MODULE name REJECTED. COMMON ATTRIBUTE CONFLICT
16	0	WARNING - CONTROL SECTION name OF MODULE name REJECTED. STORAGE PROTECTION ERROR POSSIBLE
17	0	WARNING - PRIVILEGED CSECT name OF MODULE name REJECTED BY NONPRIVILEGED CSECT

18	0	WARNING - REJECTED CSECT name OF MODULE name EXCEEDS LENGTH OF PREVIOUS CSECT. STORAGE PROTECT ERROR POSSIBLE
19	0	THE FOLLOWING REFERENCES ARE RESOLVABLE FROM SYSLIB - (followed by list of names)
20	2	EXTERNAL SYMBOL name DUPLICATES OUTPUT MODULE NAME
21	2	CONTROL SECTION name HAS ALREADY BEEN COMBINED
22	2	FORTRAN MAIN PROGRAM name CANNOT BE LINKED. THE OUTPUT MODULE CANNOT CONTAIN 2 FORTRAN MAIN PROGRAMS.
23	2	STATEMENT IS MORE THAN 256 CHARACTERS IN LENGTH.
24	0	THE STANDARD ENTRY POINT OF INPUT MODULE name CANNOT BE SAVED AS AN AUXILIARY ENTRY POINT.
25	0	MODULE (name) PRODUCED WITH LEVEL number ERRORS.
26	0	ISD OF OUTPUT MODULE EXCEEDS 256 PAGES IN LENGTH. ISD NOT PRODUCED.

The meanings of the words defined in this glossary apply only to their use in this book; these words may have slightly different meanings in other TSS/360 publications. General TSS/360 definitions are provided in IBM System/360 Time Sharing System: Concepts and Facilities, GC28-2003.

absolute DEF: A DEF (external definition) established by an assembler EQU statement whose operand is an absolute value. For instance, this example would produce in the control section dictionary an absolute DEF entry for symbol A101 whose value would be 100:

```
          ENTRY A101
     A101 EQU 100
```

adcon: See address constant.

address constant: Space reserved in a program for the address of a symbol; program text that changes as the result of relocating the program in storage. The address constant reserves storage in a program for an address that cannot be known when the program is written and ensures that the address value will be filled in before the code containing the address constant is brought into main storage. In the following assembler statement, NAME1 contains an address constant and SUBPROG is the symbol whose address is furnished:

```
     NAME1 DC A(SUBPROG)
```

In processing address constants, the language processors and linkage editor create external reference (REF) entries in the control section dictionary. These entries enable the dynamic loader to resolve the address constant (that is, compute the virtual storage address and insert it in the reserved text word) when the page containing the address constant is referred to during program execution.

alias:

1. An alternate name that may be used to refer to a member of a partitioned data set.

2. An alternate entry point by which a program (that is, a stored member of a partitioned data set) may be called.

The linkage editor and language processors all produce an external name list which is used by the VPAM STOW system routine to compile a list of aliases by which a program (that is, object module) may be called.

COMBINE:

1. A linkage editor control statement that combines two or more control sections from an input object module into one control section in the object module being built by the linkage editor.

2. The name of the linkage editor routine that processes this statement.

Since each control section must start on a page boundary, combining several short control sections may reduce the total number of pages required. Page compaction in terms of virtual or main storage may also be achieved in TSS/360 through CSECT packing (specified as a LOGON command parameter); an advantage of combining with the linkage editor is that space is saved on external storage as well.

common control section: A type of control section (created with the COM assembler language instruction or the FORTRAN COMMON statement) which may contain areas and constants referred to by independent assemblies or compilations (separate object modules) that are to be loaded for execution as one overall program. (See also control section.)

complex DEF: Either of two types of external definition (DEF): A type-1 complex DEF results from a symbol being named as the operand of an ENTRY statement in a control section other than the one in which the symbol occurs as the name of a statement. This DEF is an entry in the CSD of the control section containing the ENTRY statement. A related REF (external reference) is created in that CSD to refer to the control section in which the symbol names a statement.

A type-2 complex DEF results from an EQU statement whose name is the operand of an ENTRY statement and whose operands are one or more symbols defined as external in an EXTRN statement.

complex RLD: The part of the RLD (relocation dictionary) that contains modification values for complex DEFs.

composite ISD: The ISD (internal symbol dictionary) produced by the linkage editor. The linkage editor does not recompile a list of internal symbols, but simply includes in its output module each ISD existing in input modules. The composite ISD thus consists of:

1. Each ISD just as it appeared in its input module, and
2. A directory which heads the composite ISD and relates the external definitions and references of each input module to those in the output module.

control section: The smallest unit of a program that is relocatable to virtual storage; that portion of text specified by the programmer to be an entity, all elements of which are to be allocated contiguous virtual storage locations. A control section begins on a page boundary and consists of an integral number of pages; the page (4096 bytes) is the smallest unit of a program that can be placed in main storage. Control section may refer to any section created by the assembler language START, CSECT, DSECT, COM, PSECT, or DXD instructions (whether directly by an assembler programmer or indirectly by the FORTRAN or PL/I compilers or the linkage editor) or to the type of section created by the START or CSECT instruction as distinguished from the other instructions.

control section dictionary (CSD): A table within the program module dictionary (PMD) which contains information on the external definitions and external references within a particular control section. This table makes possible communication between control sections in the same or different object modules. There is one CSD for each control section; the program module dictionary is essentially a collection of control section dictionaries. The CSD is divided into: a heading, a definition table, a reference table, a relocation dictionary containing modification values, and a virtual memory page table which relates virtual storage assigned to the object module to the text pages it contains.

control statement: A source statement for the linkage editor. Control statements in the TSS/360 linkage editor are: INCLUDE (three forms), RENAME, COMBINE, TRAITS, and END.

CSD: See control section dictionary.

CSECT:

1. The assembler language instruction that creates and names a control section.
2. The type of control section that is created by a START or CSECT instruction.

CXD REF: A REF (external reference) entry created in the reference table of the control section dictionary by a compiler or as the result of a CXD assembler instruction. The value of the CXD REF (which is the length of combined external dummy sections) is calculated and filled in by the dynamic loader. There can be no more than one CXD REF in any CSD.

CXD-type reference: See CXD REF.

DEF: See external definition.

definition: See external definition.

definition table: A component of the control section dictionary which contains an entry for each external definition appearing in the control section. (See also control section dictionary.)

delimiter: An indicator that separates and organizes items of data. This indicator is often a character (such as a blank or a parenthesis).

dynamic loader: A TSS/360 system component which has two main functions:

1. As the result of some demand (such as a CALL command), to allocate virtual storage to object modules residing in external storage, and
2. To resolve address constants when a page of text within a module is actually referred to during program execution.

The dynamic loader does not load anything into main storage (the resident supervisor does this); the dynamic loader merely relates an object module's external location on an I/O device to a logical (virtual) address within a user's task by changing relative addresses within a module to virtual addresses within a task. The second function, resolution of address constants, is dynamic in that it does not occur until a page containing address constants is referred to by a page executing in main storage. Resolution consists of computing the virtual storage address value and inserting it into the space reserved for it in the text.

END: The linkage editor control statement which terminates control statement processing.

entry point: Generally, any location in a program or routine to which control can be passed by another program or routine. (See also standard entry point.)

entry point name:

1. A symbol whose value locates an entry point.

2. An operand in the RENAME control statement which must be an external entry point (one defined by an ENTRY assembler instruction or the name of a control section statement such as CSECT), not an internal entry point (accessible only from some other place within the same control section).

exclude: Pertaining to linkage editor output, not to include in the output module those object modules containing definitions that would resolve specified external references. The user specifies external references he does not want resolved in a form-3 INCLUDE statement. Presumably the unresolved references will be resolved by subsequent INCLUDE statements or by the dynamic loader.

external definition (DEF): Synonymous with external symbol definition. A type of entry in a control section dictionary for each external symbol in the control section. A DEF resolves a corresponding REF in some other control section. A DEF is created in the control section dictionary as the result of:

1. An object module being created (its name is made the standard entry point DEF and placed in the PMD header).

2. A control section being declared (its name is made a DEF), or

3. A symbol occurring as the operand of an ENTRY instruction. (See also absolute DEF, relocatable DEF, and complex DEF.)

external dummy control section: A dummy section (displacement map) known externally to the module in which it is defined. Each of different object modules forming a common program may contain one or more external dummy sections; the storage may be secured for all of them as one block. Each module will be able to refer to any displacement represented by a dummy section within that block. The external dummy section is created as the result of an assembler DXD instruction or a DSECT instruction in association with a Q-type address constant. (The total length of all external dummy sections defined in object modules loaded together must be provided for by a CXD instruction in one of the modules.) The external dummy section is used mainly by the PL/I compiler and assembler language programs that interface with PL/I programs.

external name list: A list of control section and entry point names produced from external definitions by language processors and the linkage editor and passed to language processor control (in the case of the PL/I compiler, to program language control). The VPAM STOW routine places these names in the partitioned organization directory (POD) of the library in which the module is placed. These names become aliases, or alternate names, by which the module can be referred to and retrieved.

external reference (REF): Synonymous with external symbol reference. A type of entry in the control section dictionary for each external symbol referred to but not necessarily defined (by an ENTRY statement) in the control section. The assembler user creates a REF as the result of an EXTRN instruction or by setting up a V-type, R-type, or Q-type address constant. A REF may also be created as the result of a complex DEF (external definition). If no corresponding external definition (DEF) exists or is found, the REF is unresolved.

external symbol: A symbol used by more than one control section within the same or different object modules. (See also external definition and external reference.)

external symbol definition: See external definition.

external symbol reference: See external reference.

INCLUDE:

1. A linkage editor control statement which has three forms:

   - Form-1 -- includes into the object module being developed by the linkage editor one or more input object modules from a specified library, and defines the input module to which any preceding TRAITS, COMBINE, or RENAME statements apply.

   - Form-2 -- includes from a specified library all object modules whose external definitions resolve external references in the module being developed by the linkage editor.

   - Form-3 -- includes from a specified library all object modules whose external definitions resolve external references in the module being developed by the linkage editor, except those external references specified.

2. The name of the linkage editor routine that processes this statement.

internal reference: A type of external reference (REF) for a symbol which is internal to the object module (that is, it can be resolved by an external definition in some control section within the same object module).

internal symbol dictionary (ISD): A table containing the location, length, and type of all symbols that name program elements (the module, control sections, instruction labels, and data areas) within an object module. The assembler, FORTRAN compiler, and linkage editor produce an ISD unless the user suppresses it; the TSS/360 PL/I compiler does not produce an ISD. The linkage editor produces a composite ISD, containing all ISDs present in input modules and an initial directory pointing to these retained ISDs. The ISD makes possible program analysis using the TSS/360 program control system (PCS) commands.

ISD: See internal symbol dictionary.

linkage editor: A system-provided program, in some respects similar to a language processor, which may be optionally used to:

1. Join, or link two or more object modules into a new, comprehensive object module, and

2. Change, or edit, control section attributes or names, entry point names, or external references in an object module by producing a new module that includes the desired changes.

Using the linkage editor eliminates the need to reassemble or recompile, may save external storage and dynamic loader processing time, and may reduce paging activity when the program is executed.

load:

1. Generally, to place data into main storage or registers.

2. Also, in TSS/360, to place programs (one or more related object modules) into virtual storage.

The dynamic loader loads an object module (that is, allocates virtual storage addresses to it within a task) as a consequence of some user or system invocation; the program, or module, is not yet moved into main storage. Physical transfer of the program, or module, into main storage is performed in page units by the resident supervisor. When a page is physically loaded into main storage, hardware-implemented dynamic address translation converts the virtual address of the page into a real main storage address.

loader: See dynamic loader.

object module: Also called a program module or an object program module, an object module in TSS/360 is the primary output of a language processor or the linkage editor. The object module is made up of a program module dictionary (PMD) containing control information, the text (that is, the program itself), and, at the user's option, an internal symbol dictionary (ISD), used for program analysis. When invoked by a user, an object module becomes input to the dynamic loader (unless it is already loaded).

object program module: Synonymous with object module.

PMD: See program module dictionary.

program module: Synonymous with object module.

program module dictionary (PMD): A table at the logical beginning of an object module containing control and descriptive information required by routines that must process the module. A PMD consists of a header and one or more control section dictionaries (CSDs).

pseudo-register: Synonymous with external dummy section.

Q REF: A reference to an external symbol that defines an external dummy section (for instance, the name of a DXD statement).

Q-type address constant: A constant that reserves storage for the value of the displacement of a symbol into an external dummy section into an area described by the dynamic loader. The symbol in the Q-type address constant must have been previously used as the name of a DXD or DSECT instruction. (See also address constant.)

Q-type reference: See Q REF.

Q-value: A value that represents the displacement of an external dummy section into the storage area reserved for external dummy sections. The dynamic loader supplies the Q-value. A program using the Q-value must get or reserve the storage required for the combined external dummy sections.

REF: See external reference.

reference: See external reference.

relocatable DEF: A DEF (external definition) whose value during execution is storage-location dependent. The value of a relocatable DEF as the result of language processing or linkage editing will be some displacement from the beginning of the control section in which the definition occurs. For example, if some statement at byte location 1000, relative to the origin of its control section, is named CHXAAA, then

```
   ENTRY CHXAAA
```

will produce a relocatable DEF entry for the symbol CHXAAA whose value will be 1000. The dynamic loader processes relocatable DEFs by adding, to the value assigned by the language processor or linkage editor, the virtual storage address of the defining control section.

relocation dictionary (RLD): A table within each control section dictionary which contains modifier pointers and modifiers for address constants (adcons). Each modifier pointer indicates a text page within the control section that contains address constants; each modifier contains information which the dynamic loader uses to determine the final value of the address constant. There are three RLDs in each control section dictionary: one for complex DEFs, one for external references, and one for internal references.

RLD: See relocation dictionary.

RENAME:

1. A linkage editor control statement that changes entry point names, control section names, or external references, or deletes entry point or control section names;

2. The name of the linkage editor routine that processes this statement.

resolved: Applied to external reference for which the linkage editor or dynamic loader is able to find a corresponding external definition.

R-type address constant: An address constant whose value is the address of the control section in which a specified symbol was defined. For example, in

```
   A DC R(ENTRY1)
```

the value inserted in location A by the dynamic loader will be the address of the control section in which ENTRY1 was defined (in which an ENTRY statement occurred with ENTRY1 as the operand). (See also address constant.)

R-value: The virtual storage location of the origin of the control section in which an ENTRY statement for a symbol appeared. Conventionally, when linking to reenterable (nonmodifiable) code in TSS/360, the V-value locates an executable instruction to which control is passed; the R-value of a symbol locates the beginning of a control section (usually a PSECT) which may be used for modifiable storage. An R-value is also assigned to symbols that are the names of object modules and control sections. The R-value of the control section is simply the value of the control section name. The R-value of a module is either the address of the first PSECT in the module, or, if no PSECT exists, of the first CSECT.

stack: To collect and hold language statements pending the occurrence of some unifying or clarifying statement. In the TSS/360 linkage editor, RENAME, COMBINE, and TRAITS control statements are stacked until a form-1 INCLUDE statement occurs; then they are processed.

standard entry point: The location in an object module at which program execution will begin if the module is invoked by its name. A user may call a program to run (via the CALL command or by direct call) by specifying the object module name; execution will begin at the standard entry point. An object module may have several entry points to which other programs can pass control; it can have only one standard entry point. The FORTRAN and PL/I compilers generate a value which is the location of the beginning of the main procedure in the module. The assembler uses the address of the first control section (CSECT) in the module as the standard entry point unless the user has specified another location as the operand of an END statement. The standard entry point name is contained as a DEF in the header of the PMD. The linkage editor produces a module whose standard entry point is that of the first input module; it also retains the standard entry point of each input module, enabling the user to run by name not only the linkage edited module but any of its component input modules.

symbol: A character or combination of characters that represents addresses or specified absolute values. Through their use as names and in operands, symbols provide the programmer with a way to name and refer to elements (control sections, instructions, and data areas) of a program.

text: The instructions, constants, and reserved data areas of an object module; the program itself.

TRAITS:

1. A linkage editor control statement that specifies new attributes for a designated control section.

2. The name of the linkage editor routine that processes this statement.

type-1 complex DEF: See complex DEF.

type-2 complex DEF: See complex DEF.

unresolved: Applied to external references for which the linkage editor or dynamic loader is unable to find a corresponding external definition in another object module or control section. The linkage editor provides a list of unresolved references at the termination of its processing (as well as those unresolved but resolvable by definitions in programs in SYSLIB).

version identifier: A character string that identifies a particular assembly, compilation, or linkage editor run. The character string can be one-to-eight alphameric characters specified by the user in his command, or, if defaulted, will be the data and time of the run, supplied by the system. The version identifier is placed in the program module dictionary and appears in the PMD listing section of the list data set.

virtual address: Also called logical address, and address generated by a program that references virtual storage and must, therefore, be translated into a main storage address as it is used.

virtual memory page table (VMPT): A table in each control section dictionary which relates pages of text within the control section to virtual storage assigned the control section. A control section may occupy more space in virtual storage than its text pages require; ORG instructions will cause virtual storage to be allocated which does not contain text pages. The VMPT tells whether a page is empty (reserved) or, if it contains text, which page it is relative to the first page in the control section containing text.

VMPT: See virtual memory page table.

V-type address constant: A type of address constant that reserves storage for and whose value during program execution is the address of an external symbol. By specifying a symbol in a V-type address constant, the assembler language EXTRN instruction need not be used. Conventionally, when linking to a reenterable (nonmodifiable) program in TSS/360, the V-type address constant loaded into a register provides the address to which control is to be passed; the R-type address constant loaded in another register provides the location of a modifiable control section. For each V-type address constant, an external reference (REF) is created in the control section dictionary. (See also address constant.)

V-value: A virtual storage location that an external symbol labels. By convention in TSS/360, when linking to reenterable (nonmodifiable) code, the V-value of a symbol locates the symbol itself (provides its address); the R-value of a symbol locates the beginning of a PSECT which the executable code may use to obtain and modify data. V-values are provided by the dynamic loader.

Where more than one page reference is given, the major reference is first.

APENCX - append complex RLD
   external table references 71
   flowchart (Chart AM) 55
   subroutine description 10,16
APENDF - append definition table subroutine
   flowchart (Chart AL) 54
   subroutine description 16
APENEX - append external RLD
   external table references 71
   flowchart (Chart AM) 55
   subroutine description 10,16
APENIN - append internal RLD
   external table references 71
   flowchart (Chart AM) 55
   subroutine description 10,16
append complex RLD
   (see APENCX)
append definition table
   (see APENDF)
append external RLD
   (see APENX)
append internal RLD
   (see APENIN)
attributes of control sections 5

blank control sections 5
BRING - Bring PMD, Text, and ISD from library
   external table references 71
   flowchart (Chart AN) 56
   subroutine description 16
Bring PMD, Text, and ISD from library
   (see BRING)

CFADA
   (see CPC MAIN)
CFADB
   (see GETLINE)
CFADC
   (see PUTDIAG)
CEYBR
   (see BRING)
CEYCO
   (see COMBINE)
CEYCX
   (see APENCX)
CEYEE
   (see EARLY)
CEYEN
   (see END)
CEYER
   (see ERROR)
CEYEX
   (see APENEX)
CEYGA
   (see GTCSAD)
CEYGC
   (see GETCSD)
CEYIA
   (see INANAL)
CEYIC
   (see INCLUDE)
CEYIN
   (see APENIN)
CEYLK
   (see LINK)
CEYOP
   (see OUTPUT)
CEYRN
   (see RENAME)
CEYSC
   (see SCAN)
CEYTR
   (see TRAITS)
CEYTS
   (see linkage editor)
CEYXR
   (see EXTREF)
CLEANUP - cleanup final module subroutine
   flowchart (Chart AH) 46
   subroutine description 17
Cleanup final module
   (see CLEANUP)
COLLECT - collect common requirements subroutine
   flowchart (Chart AI) 47
   subroutine description 17
Collect common requirements
   (see COLLECT)
COMBINE - COMBINE statement processor
   definition 87
   external table references 71
   flowchart (Chart AA) 40
   subroutine description 17
Combine control sections
   (see COMSUB)
COMBINE statement processor
   (see COMBINE)
Command Analyzer & Executor (CA&E)
   GETLINE subroutine (CFADB) 9,21
   LPC MAIN routine (CFADA) 9
   PUTDIAG subroutine (CFADC) 9,26
common areas
   exclusion table (EXCLUD) 69
   external name list (NAMES) 69
   hash table (HASHTB) 69
   rename/combine table (RCTBL) 70
   statement stack list (STACK) 70
   TEMP storage 69
   work areas 69
common attribute 5
common internal storage areas
   TEMP storage 69
   work areas 69
common internal lists
   external name list (NAMES) 69
   statement stack list (STACK) 70

- common internal tables
  - exclusion table (EXCLUD) 69
  - hash table (HASHTB) 69
  - rename/combine table (RCTBL) 70
- COMSUB - combine control sections subroutine
  - flowchart (Chart AD) 45
  - subroutine description 18
- confirmation messages 85
- control section dictionary (CSD) 77-79,4
  - definition 88
- control sections (CSECT)
  - attributes of 5
  - common 5
  - definition 88
- control statement input/analyze
  - (see INANAL)
- control statement processing 8
- conversational mode
  - (see mode of operation)
- CSD
  - (control section dictionary) 77-79,4

- data areas 69
- debugging aids 73
- decision tables
  - control statement processing 12
  - early-end processing 34
  - output processing 30
- DEF 89
- definition table 79,88
- delete entry name
  - (see DELNAME)
- DELNAME - delete entry name subroutine
  - flowchart (Chart AA) 41
  - subroutine description 18
- diagnostic aids 73
- diagnostic codes 85

- EARLY - early-end processor
  - external table references 71
  - flowchart (Chart CA) 67
  - subroutine description 34
- early-end processing 34,8
- Early-End processor
  - (see EARLY)
- END - END statement processor
  - definition 88
  - external table references 71
  - flowchart (Chart AA) 44
  - subroutine description 18
- END statement processor
  - (see END)
- ERROR - error message processor
  - external table references 71
  - flowchart (Chart AO) 58
  - subroutine description 20
- error message processor
  - (see ERROR)
- error messages 85
- EXCLUD - exclusion table 69
- Exclusion table
  - (see EXCLUD)
- External name list
  - (see NAMES)
- external references 71

- External reference search
  - (see EXTREF)
- EXTREF - external reference search
  - external table references 71
  - flowchart (Chart AP) 59
  - subroutine description 20

- Fix ISD
  - (see FIXISD)
- FIXISD - FIX ISD subroutine
  - flowchart (Chart AI) 47
  - subroutine description 21

- general register assignments 73
- GETCSD - locate control section dictionary
  - flowchart (Chart AP) 59
  - subroutine description 21
- Get CSD table addresses
  - (see GTCSAD)
- GETLINE routine (CFADB) 9,21
- glossary 87
- GTCSAD - get CSD table addresses
  - external table references 71
  - flowchart (Chart AQ) 60
  - subroutine description 21

- hash table (HASHTB) 69
- hashing algorithm 70
- HASHTB - hash table 69

- INANAL - control statement input/analyze processor
  - external table references 71
  - flowchart (Chart AA) 36
  - subroutine description 22
- INCLUDE - INCLUDE statement processor
  - definition 89
  - external table references 71
  - flowchart (Chart AA) 37
  - subroutine description 23
    - form-1 INCLUDE processing 23
    - form-2 INCLUDE processing 24
    - form-3 INCLUDE processing 24
- INCLUDE statement processor
  - (see INCLUDE)
- input modules (maximum number) 74
- internal symbol dictionary (ISD) 83-84,4
  - composite ISD 84
  - composition of ISD 84
  - definition 90
  - external table references 71
- ISD (internal symbol dictionary) 83,4

- job library (JOBLIB) 5

- language processor control (LPC) 9
- LIBE SEARCH - (part of dynamic loader)
  - subroutine description 24
- limitations 74
- linkage editor
  - characteristics of routines 8
  - control sections 5

definition 90
error detection and messages 85
functions 3
input 3
interface with language processor 9
library calls 5
limitations 74
major divisions 8
object module structure 3
output 3,29
programming aids 73,85
program module dictionary 75-82,4
relationship to language processors 9,7
relationship to TSS/360 6
size considerations 74
LINK - link modules subroutine
external table references 71
flowchart (Chart AJ) 48
subroutine description 24
link modules
(see LINK)
LNK command 2
locate control section dictionary
(see GETCSD)
LPC MAIN routine (CFADA) 9
LSTPMD - program module dictionary listing subroutine
flowchart (Chart BB) 62
subroutine description 32

Messages, diagnostic 85
MD240 subroutine
flowchart (Chart BB) 62
subroutine description 33
MD300 subroutine
flowchart (Chart BC) 64
subroutine description 33
MD350 subroutine
flowchart (Chart BC) 64
subroutine description 33
MD450 subroutine
flowchart (Chart BD) 65
subroutine description 33
MD500 subroutine
flowchart(Chart BD) 65
subroutine description 33
MD600 subroutine
flowchart (Chart BE) 66
subroutine description 33
Mode of operation
conversational mode 1,3
error handling 9,20
nonconversational mode 1,3
modules (input) 74
maximum number
modules (output) 74
size

name designations in program listings 68
NAMES - external name list 69
nonconversational mode
(see mode of operation)

object modules
definition 3,90
structure 3
(see also output modules)
output modules 3,74
OUTPUT - output processor 29
external table references 71
flowchart (Chart BA) 61
subroutine description 31
output processing 29,3
output processor
(see OUTPUT)

partitioned data sets 3
partitioned organization dictionary (POD) 5
PMD (program module dictionary) 75-82,4
privileged attribute 5
program control system (PCS) 4
program module dictionary (PMD) 75-82,4
external table references 71
PMD listing 3
program module dictionary listing subroutine
(see LSTPMD)
prototype attribute 5
PSECT
organization 72
storage 71
public attribute 5
PUTDIAG routine (CFADC) 9,26

RCTBL - rename/combine table 70
read-only attribute 5
REF 89
reference table 80
register assignment and usage 73
relocation dictionary (RLD) 81
rename/combine table (RCTBL) 70
RENAME - RENAME statement processor
external table references 71
flowchart (Chart AA) 41
subroutine description 26
RENAME statement processor
(see RENAME)
RLD 81,91

SCAN - scan subroutine
external table references 71
flowchart (Chart AQ) 60
subroutine description 27
scan subroutine
(see SCAN)
size requirements and limitations 74
STACK - statement stack list
statement stack list (STACK) 70
statement store area (SAVLN1) 23
storage areas (PSECT) 71
system attribute 5
system library (SYSLIB) 5

TEMP storage 69
TRAITS statement processor
(see TRAITS)
TRAITS - traits statement processor
definition 92

external table references 71
flowchart (Chart AA) 43
subroutine description 28

update ISD subroutine
(see UPISD)
UPISD - update ISD subroutine
flowchart (Chart AK) 53
subroutine description 28
user library (USERLIB) 5

variable length attribute 5
virtual memory page table (VMPT) 82,92
virtual storage requirements 74

WORKC1, WORKC2, WORKT
(see common areas)

IBM Technical Newsletter

File Number	S360-31
Base Publication No.	GY28-2030-2
This Newsletter No.	GN28-3216
Date	February 1, 1972
Previous Newsletters	None

IBM System/360 Time Sharing System:
Linkage Editor

This Technical Newsletter provides replacement pages for the subject publication. Pages to be inserted and/or removed are:

73-74

A change to the text is indicated by a vertical line to the left of the change.

Summary of Amendments

Errors concerning the size limitations imposed by the linkage editor have been corrected.

IBM Corporation, Dept. 643, Neighborhood Road, Kingston, N. Y. 12401

PRINTED IN U S A

GY28-2030-2

TSS LINKAGE EDITOR Printed in U.S.A. GY28-2030-2

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]