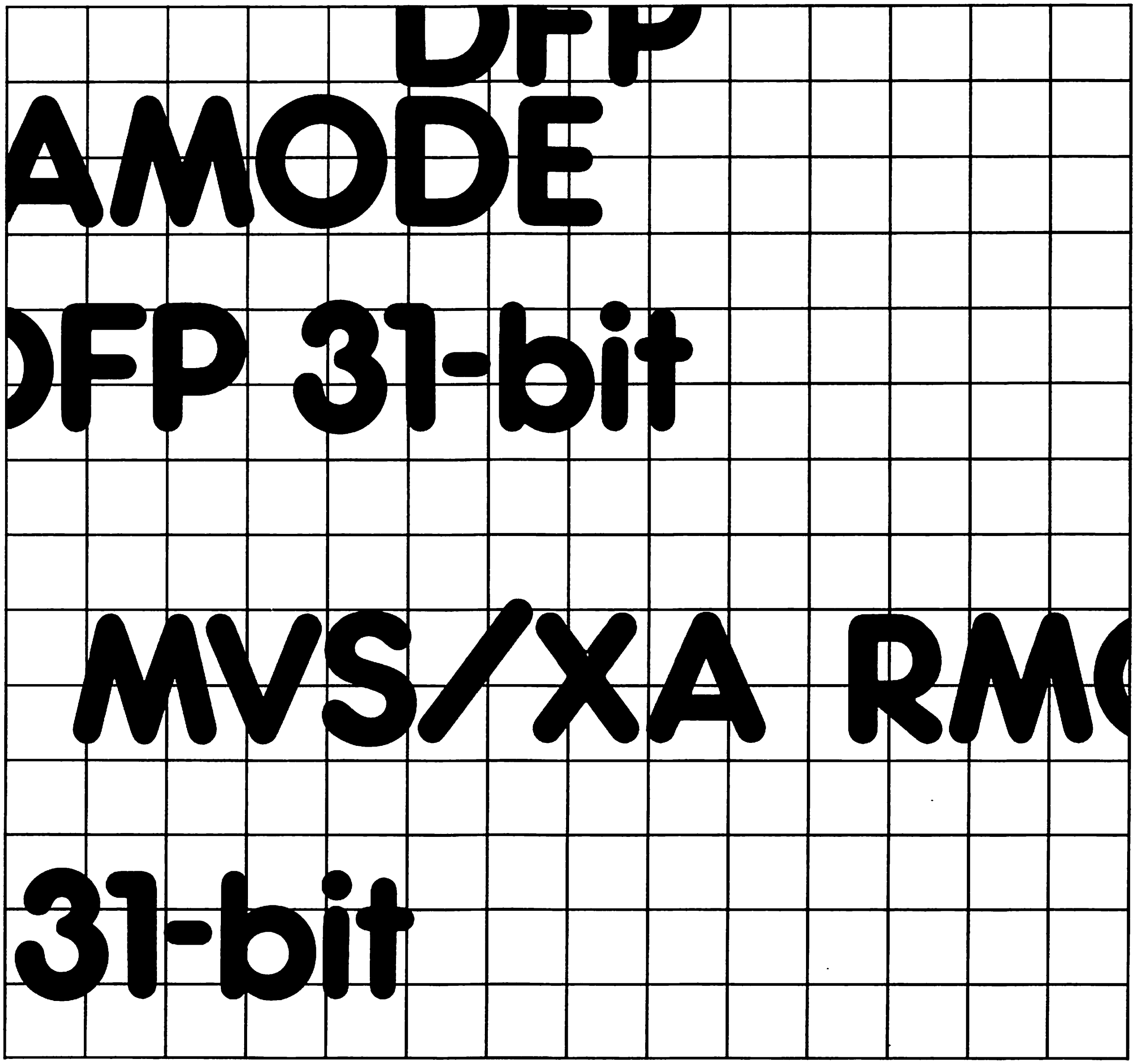
MVS/Extended Architecture Linkage Editor Logic

Program Product



Order No. LY26-3963-0
File No. S370-31

Data Facility Product 5665-XA2
Version 2
Release 1.0

MVS/Extended Architecture
Linkage Editor Logic

Data Facility Product 5665-XA2
Version 2 Release 1.0

LY26-3963-0

First Edition (April 1985)

This edition applies to Version 2 Release 1.0 of MVS/Extended Architecture Data Facility Product, Program Product 5665-XA2, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for Version 2 support are summarized under "Summary of Amendments" following the preface. Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

This book describes the internal organization and logic of the linkage editor. The linkage editor is a processing program that combines and edits modules to produce a load module; this load module can then be loaded into virtual storage by the control program.

ORGANIZATION

This publication contains the following:

- "Introduction" on page 1, describes the linkage editor as a whole, and its relationship to the operating system. This section also discusses the major divisions of the program and how they work together.
- "Method of Operation" on page 15, provides an overview of the logic of the linkage editor, and a detailed description of specific operations.
- "Program Organization" on page 89, describes the organization of the linkage editor. The function and interrelationship of program components (modules, control sections, and routines) are also discussed.
- "Microfiche Directory" on page 146, helps the reader find the named areas of code in the program listing. Microfiche cards contain the program listing.
- "Table Layouts" on page 151, which are used to analyze storage dumps, are illustrated in this section.
- "Diagnostic Aids" on page 183, includes general register contents at major entry points to modules, and an error message—module cross-reference table.
- "Appendix. Conventions/Formats" on page 194, includes input conventions and record formats.

An index is also included.

PREREQUISITE KNOWLEDGE

To use this book efficiently, you should be familiar with the following topics:

- Assembler language functions and specifications available with Assembler H
- Job control language
- General concepts of the linkage editor and loader

REQUIRED PUBLICATIONS

You should be familiar with the information in the following publications:

- Assembler H Version 2 Application Programming: Language Reference, GC26-4037, for a description of assembler language functions

- MVS/Extended Architecture JCL, GC28-1148, for background information on job control language

- MVS/Extended Architecture Linkage Editor and Loader User's Guide, GC26-4143, for a description of the linkage editor and loader

RELATED PUBLICATIONS

Within the text, references are made to the publications listed below.

Short Title	Publication Title	Order Number
Assembler H V2 Application Programming: Language Reference	Assembler H Version 2 Application Programming: Language Reference	GC26-4037
Data Administration Guide	MVS/Extended Architecture Data Administration Guide	GC26-4140
Data Administration: Macro Instruction Reference	MVS/Extended Architecture Data Administration: Macro Instruction Reference	GC26-4141
Debugging Handbook	MVS/Extended Architecture Debugging Handbook, Volumes 1 through 5	LC28-1164[1] LC28-1165 LC28-1166 LC28-1167 LC28-1168
JCL	MVS/Extended Architecture JCL	GC28-1148
Linkage Editor and Loader User's Guide	MVS/Extended Architecture Linkage Editor and Loader User's Guide	GC26-4143

Note:

[1] All five volumes may be ordered under one order number, LBOF-1015.

SUMMARY OF AMENDMENTS

RELEASE 1.0, APRIL 1985

ENHANCEMENTS AND NEW SUPPORT

- Maximum block size has been increased from 18432 to 32760.
- PCI support for overlay modules has been added to IEWFETCH.
- Error messages IEW0664, IEW0801, and IEW0813 have been added to Figure 68 on page 189 under "Diagnostic Aids."
- Figure 67 on page 188 under "Diagnostic Aids" has been updated to reflect changes in table allocation.
- Offsets have been adjusted in Figure 36 on page 152 under "Table Layouts."

VERSION 2 PUBLICATIONS

The Preface includes new order numbers for Version 2.

CONTENTS

Introduction 1
Purpose of Linkage Editor 1
Relationship to the Operating System 2
General Description 2
Module Structure 3
External Symbol Dictionary 4
Relocation Dictionary 4
Composite Dictionaries 4
Linkage Editor Options 5
Module Attributes 6
Linkage Editor Processing for Attributes 8
Input/Output Flow 11

Method of Operation 15
Logic of the Linkage Editor 15
Initialization 15
Input Processing 15
Intermediate Processing 16
Second Pass Processing 17
Final Processing 18
Initialization 19
Preparing the All-Purpose Table (APT) 19
Analyzing Control Information 19
Opening Data Sets 20
Allocating Virtual Storage 21
Buffer Allocation 21
Table Allocation 22
Input Processing 22
Reading Blocked Input 23
Record Lengths for SYSPRINT 23
Record Lengths for SYSTERM 23
Control Statement 24
Control Statement Processors 25
Object Module Processing 31
Load Module Processing 32
ESD Record Types 34
CESD Record Types 34
ESD Processing 35
IDR Processing 39
Processing Object Module END Records Containing IDR Data 40
Processing Load Module IDRs 40
Processing IDENTIFY Control Statement Data 41
TXT Processing 41
Processing Object Module Text 42
Processing Load Module Text 43
Writing Text on SYSUT1 43
RLD Processing 44
END Processing 47
Include Processing 47
Automatic Library Call Processing 49
Intermediate Processing 51
Address Assignment 51
ENTAB Size Determination 54
Entry Processing 56
Intermediate Output Processing 57
MAP/XREF Processing 59
Second Pass Processing 59
Relocation of Address Constants 61
Relocation of Nonbranch-Type (A-Type) Address Constants 61
Relocation of Branch-Type (V-Type) Address Constants 66
ENTAB Creation 68
Relocation Routine 68
Final Processing (HEWLFFNL) 72
Error Logging 73
Cross-Reference Table 74
Diagram 1. Overview of Linkage Editor 75

Diagram 2. Detailed Overview of Linkage Editor Processing 76
Diagram 3. Initialization 77
Diagram 4. Input Processing 78
Diagram 5. Intermediate Processing 79
Diagram 6. Second Pass Processing 80
Diagram 7. Final Processing 81
Diagram 8. Control Statement Processing 82
Diagram 9. ESD Processing 83
Diagram 10. Processing Object Module Text 84
Diagram 11. Processing Load Module Text Records 85
Diagram 12. RLD Processing 86
Diagram 13. Address Assignment 87
Diagram 14. Data Movement During Second Pass Processing 88

Program Organization 89
Initialization and Input Processing 89
    Initial Processor—HEWLFINT (Chart BA) 89
    Attributes and Options Processor—HEWLFOPT 89
    Allocation Processor—ALL001 (Chart BA) 89
    Table Allocation Processor—HEWLFALK (Chart BB) . . 90
    Input Processor—HEWLFINP (Chart CA) 90
    Object Module Processor—HEWLFMDI (Chart CB) 90
    Load Module Processor—INP270 (Chart CC) 91
    SYM Processor—HEWLFSYM (Chart CD) 91
    ESD Processor—HEWLFESD (Chart CE) 92
    Text and RLD Processor—HEWLFRAT (Chart CF) 92
    Text Processor—HEWLFTXT (Chart CG) 92
    RLD Processor—RLD001 (Chart CJ) 92
    End Processor—HEWLFEND (Chart CL) 93
    CSECT Identification Record (IDR) Processor—HEWLFIDR (Chart CQ) 93
    Control Statement Scanner—HEWLFSCN (Chart CS) . . . 93
    Include Processor—HEWLFINC (Chart CU) 93
    Automatic Library Call Processor—HEWLCAUT (Chart CV) 94
Intermediate Processing 94
    Address Assignment Processor—HEWLFADA (Chart DA) . 94
    Intermediate Output Processor—HEWLFOUT (Chart EA) . 94
Second Pass Processing 95
    Second Pass Processor—HEWLFSCD (Chart FA) 95
Final Processing 95
    Final Processor—HEWLFFNL (Chart GA) 95
    SYNAD Routine—HEWLCR01 (Chart GB) 96
Chart AA. Level Major Divisions 99
Chart BA. Initial Processor (HEWLFINT) 100
Chart BB. Table Allocation Processor (HEWLFALK) 101
Chart CA. Input Processor (HEWLFINP) 102
Chart CB. Object Module Processor (HEWLFMDI) 103
Chart CC. Load Module Processor (INP270) 104
Chart CD. SYM Processor (HEWIFSYM) 105
Chart CE. ESD Processor (NEWLFESD)
Chart CF. TXT and RLD Processor (HEWLFRAT) 109
Chart CG. TXT Processor (HEWLFTXT) 110
Chart CH. TXT Write Routine (on SYSUT1) (TXTBUF) 111
Chart CJ. RLD Processor (RLD001)
Chart CK. RLD Write Routine (RLDBUF) 114
Chart CL. END Processor (HEWLFEND) 115
Chart CM. CSECT Identification Record Processor (HEWLFIDR) 116
Chart CN. IDR Translator Data Processor (HEWLFIDR) . . . 117
Chart CP. IDRSPZAP Data Processor (HEWLFIDR) 118
Chart CQ. IDR Identify Data Processor (HEWLFIDR) 119
Chart CR. IDR User Data Processor (HEWLFIDR) 120
Chart CS. Control Statement Scanner (HEWLFSCN)
Chart CT. READ8 Routine 123
Chart CU. Include Processor (HEWLFINC) 124
Chart CV. Automatic Library Call Processor (HEWLCAUT)
Chart DA. Address Assignment Processor (HEWLFADA) . . . 127
Chart DB. ENTAB Size Determination Routine (HEWLFENS) . 128
Chart DC. Entry Processor (HEWLFENT)
Chart EA. Intermediate Output Processor (HEWLFOUT) . . . 131
Chart EB. MAP/REF Processor (HEWLFMAP) 132
Chart EC. IDR WRITE Routine (HEWLFOUT) Chart FA. Second Pass Processor (HEWLFSCD)
Chart FB. GETIDMUL Routine 137

Chart FC. TXT Read Routine (RDTXT), RLD Read Routine (RDRLD)—HEWLFSIO 138
Chart FD. Text Write Routine (on SYSLMOD) (WRTTXT)—HEWLFSIO 139
Chart FE. Relocation Routine (HEWLFREL)
Chart GA. Final Processor (HEWLFFNL) 143
Chart GB. SYNAD Routine (HEWLCR01) 144
Chart GC. Error Logging Routine (HEWLFLOG) 145

Microfiche Directory **146**

Table Layouts **151**

Diagnostic Aids **183**

Appendix. Conventions/Formats **194**
Input Conventions 194
Record Formats 195

Index **207**

FIGURES

1. Linkage Editor Processing—Simple Case 3
2. Combining Control Dictionaries 5
3. Linkage Editor Processing for the Overlay and TEST Attributes 9
4. Linkage Editor Processing for the Scatter Load and TEST Attribute 11
5. Input/Output Flow 13
6. Incompatible Module Attributes and Program Options 20
7. Control Statement Scanner Operation 24
8. INCLUDE Statement Processing for a Sequential Data Set 25
9. INCLUDE Statement Processing With Nested Members . 26
10. Overlay Statement Processing 27
11. Order and Page Processing 29
12. Library Statement Processing 30
13. General Register Information—Object Module Processing 31
14. Input Record Types—Load Module 33
15. General Register Information—Load Module Processing 33
16. RLD Flag Field Processing 46
17. Include Processing 49
18. Automatic Library Call Processing 51
19. ENTAB Size Determination 55
20. Processing of Alias Symbols by the Entry Processor 57
21. Writing Scatter/Translation Records 58
22. Nonbranch-Type Address Constants—Relative Relocation 62
23. Nonbranch-Type Address Constants—Absolute Relocation 63
24. Nonbranch-Type Address Constants—Absolute and Relative Relocation 64
25. Example of Delinking 65
26. Entry List Processing 67
27. Relationship of RLD Flag Field to Relocation . . . 69
28. ENTAB Creation 70
29. Building Error Messages 73
30. Load Module Record Types and Associated Processors 91
31. Linkage Editor Organization 97
32. Sample Flowchart Symbols 98
33. Microfiche Directory 146
34. Module/CSECT Cross-Reference Table 150
35. Table Construction and Usage 151
36. All-Purpose Table (APT) 152
37. Alias Table 161
38. Calls List (As Built by RLD Processor) 162
39. Calls List (As Altered and Used by ENTAB Size Determination Routine) 162
40. Composite External Symbol Dictionary (CESD)—Internal Format 163
41. Normal Combination of Internal CESD Types 164
42. Delink Table 165
43. Downward Calls List 165
44. Entry List 166
45. Entry Table (ENTAB) 166
46. Half External Symbol Symbol Dictionary (HESD) . . . 167
47. High ID Table (HIID) 168
48. Virtual Storage Allocation Table 169
49. Partitioned Organization Directory Record (As Received from BLDL) 170
50. Module Attributes 171
51. Partitioned Organization Directory Record (As Built by Linkage Editor) 173
52. Relocation Constant Table (RCT) 174
53. Renumbering Table (RNT) 174
54. RLD Input Control Block 175
55. RLD Output Control Block 176
56. RLD Note List 177
57. Second Pass Text Control Block 178
58. Segment Length Table (SEGLGTH) 179
59. Segment Table (SEGTAB) 180
60. TABLE and LIST (Referred to by HEWLFBTP) 180
61. Text I/O Table 181

62. Text Note List 181
63. XAD2CESD Table (Built and Referred to by Cross-Reference Table Routine) 182
64. ORDER Table (Built by HEWLFSCN) 182
65. General Register Contents at Major Entry Points . . 183
66. Buffer Allocation 187
67. Table Allocation 188
68. Error Message/Issuer Cross-Reference Table 189
69. SYM Input Record (Card Image) 195
70. ESD Input Record (Card Image) 196
71. Text Input Record (Card Image) 197
72. RLD Input Record (Card Image) 197
73. END Input Record—Type 1 (Card Image) 198
74. END Input Record—Type 2 (Card Image) 198
75. IDR Data in an Object Module End Record 199
76. SYM Record (Load Module) 199
77. CESD Record (Load Module) 200
78. Scatter/Translation Record 201
79. Control Record (Load Module) 202
80. Relocation Dictionary Record (Load Module) 203
81. Control and Relocation Dictionary Record (Load Module) 204
82. Record Format of Load Module IDRs 204

INTRODUCTION

This section describes the purpose, organization, and internal operation of the linkage editor and its relationship to the operating system.

PURPOSE OF LINKAGE EDITOR

The linkage editor is one of the processing programs of the operating system. It is a service program used in conjunction with the language translators to prepare machine-language programs frem symbolic-language programs written in FORTRAN, COBOL, report program generator, assembler language, or PL/I. Linkage editor processing is a necessary step that follows source program assembly or compilation.

Linkage editor processing allows the programmer to divide a program into several parts, each containing one or more control sections. Each part may then be coded in the programming language best suited to it, and may then be separately assembled or compiled by a language translator (under the rules applicable to each language translator).

The primary purpose of the linkage editor is to combine and link object modules (the output of the language translators) into a load module. In that load module, all cross-references between control sections are resolved as though they had been assembled or compiled as one module. The load module produced by the linkage editor consists of executable machine-language code in a format that can be loaded into virtual storage and relocated by program fetch.

In addition to combining and linking object modules, the linkage editor performs the following functions:

- Library Calls. Modules (such as standard subroutines) stored in a library can be placed in the input to the linkage editor, either automatically or upon request. If unresolved external references remain after all input to the linkage editor is processed, an automatic library call routine retrieves the modules required to resolve the references. However, unresolved external references marked "weak call" or "never call" are not resolved by this routine.

- Program Modification. Control sections can be replaced, deleted, or rearranged (in overlay programs) during linkage editor processing, as directed by linkage editor control statements. Common control sections generated by the FORTRAN, PL/I, and assembler language translators are provided locations within the output load module.

- Order and Page Support. The linkage editor can order control sections in the sequence specified on the linkage editor control statements, and can assign control sections to page boundaries according to the control statements.

- Addressing Mode, Residence Mode, and Read-Only Support. The linkage editor assigns an addressing mode for the entry points into a load module, assigns a residence mode for the load module, and indicates which control sections are read-only in a nucleus load module.

- Program Processing History. CSECT identification records built during linkage editor processing contain data describing the language translators and the linkage editor that produced the program, any modifications to that program by AMASPZAP, and, optionally, up to 40 characters of user data for each control section within the program.

- Overlay Module Processing. The linkage editor prepares modules for overlay by assigning relative locations within the module to the overlay segments, and by inserting tables to be used by the overlay supervisor during execution.

- Options and Error Messages. The linkage editor can:

  - Process special options that override automatic library calls or the effect of minor errors

  - Produce a list of linkage editor control statements that were processed

  - Produce coded diagnostic messages and a directory describing those diagnostic messages that were printed out during linkage editor processing

  - Produce a module map or cross-reference table of control sections in the output load module

RELATIONSHIP TO THE OPERATING SYSTEM

The linkage editor has the same relationship to the operating system as any other processing program. Control is passed to the linkage editor in one of three ways:

1. As a job step, when the linkage editor is specified on an EXEC job control statement in the input stream

2. As a subprogram, via the execution of a CALL macro instruction (after execution of a LOAD macro instruction), a LINK macro instruction, or an XCTL macro instruction

3. As a subtask, in multitasking systems, via execution of the ATTACH macro instruction

GENERAL DESCRIPTION

Linkage editor input may consist of a combination of object modules, load modules, and linkage editor control statements. The prime function of the linkage editor is to combine these modules, in accordance with requirements stated on control statements, into a single output load module that can be relocated and loaded into real storage by program fetch for execution. Output load modules are placed into partitioned data sets (libraries).

Each module to be processed by the linkage editor has an origin that was assigned during assembly, during compilation, or during a previous execution of the linkage editor. Each module in the input to the linkage editor may contain symbolic references to control sections in other modules; such references are called external references.

To produce an executable output load module, the linkage editor:

1. Assigns relative virtual storage addresses to the control sections to be included in the output module. Because each input module has an origin that was assigned independently by a language translator, the order of the addresses in the input is unpredictable. (Two input modules, for example, may have the same origin.) The linkage editor assigns an origin to the first control section, and then assigns addresses to all other control sections in the output relative to either this origin or to the last control section aligned on a page boundary.

2. Resolves external references in the input modules. Cross-references between control sections in different modules are symbolic, and must be resolved (translated into relocatable machine addresses) in relation to the contiguous virtual storage addresses assigned to the output load module. These symbolic cross-references are made by means of address constants.

The linkage editor calculates the new address of each relocatable expression in a control section, and determines the assigned origin (value) of the item to which it refers.

Linkage editor processing is affected by specified options, operations requested on control statements, module attributes contained in partitioned data set directories, and control information contained within the modules themselves. The following paragraphs describe the relationship of module structure, linkage editor options, and module attributes to linkage editor processing.

MODULE STRUCTURE

Object modules and load modules have the same basic logical structure (see Figure 1). Each consists of:

- Control dictionaries, containing the information necessary to resolve symbolic cross-references between control sections of different modules and to relocate address constants
- Text, containing the instructions and data of the program
- An end of module (EOM) indicator (END record in object modules; EOM indication in load modules)

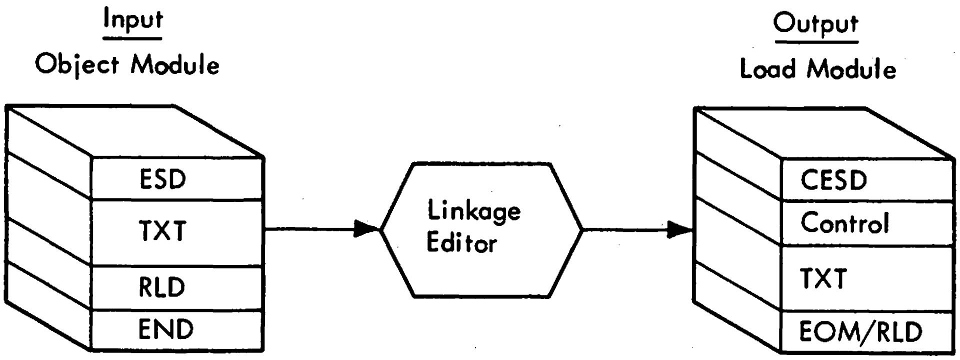


Figure 1. Linkage Editor Processing—Simple Case

Each language translator usually produces two kinds of control dictionaries: an external symbol dictionary (ESD) and a relocation dictionary (RLD). An object module always contains an ESD; a load module contains an ESD unless it is marked with the "not editable" attribute. Object and load modules usually contain an RLD (unless there are no relocatable address constants in the module). Control dictionary entries are generated when external symbols, address constants, or control sections are processed by a language translator.

External Symbol Dictionary

An external symbol dictionary contains entries for all external symbols defined or referred to within a module. (An external symbol is one that is defined in one module and can be referred to in another.) Each entry identifies a symbol, or a symbol reference, and gives its location, if any, within the module. When combining input modules, the linkage editor resolves references between different input modules by matching the referenced symbols to defined symbols; it does this by searching for the external symbol definitions in each input module's ESD. There is an ESD entry for each named control section and each named common area. The ESD also contains entries that identify unnamed control sections and unnamed common areas.

Relocation Dictionary

The relocation dictionary (RLD) lists all relocatable address constants that must be modified when the linkage editor produces an output load module. The linkage editor uses the RLD whenever it processes a module. The RLD is also used to adjust the value of address constants after program fetch reads an output load module from a library and loads it into virtual storage for execution. The RLD contains at least one entry for every relocatable address constant in a module. An RLD entry identifies an address constant by indicating both (1) its location within a control section, and (2) the external symbol (in the ESD) whose value must be used to compute the value of the address constant.

Composite Dictionaries

An output load module is composed of all input object modules and input load modules processed by the linkage editor (except those that are replaced or deleted). The control dictionaries of an output module are therefore a composite of all the control dictionaries in the linkage editor input. The control dictionaries of a load module are called the composite ESD (CESD) and the RLD.

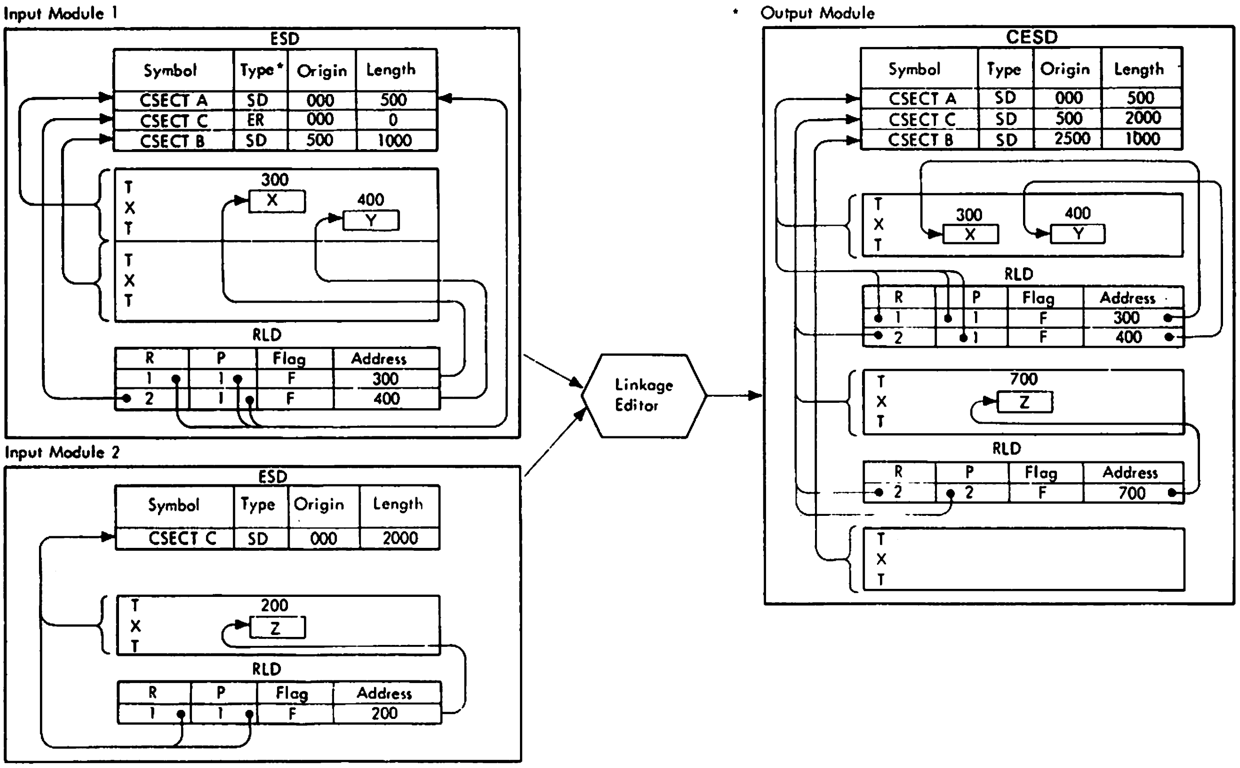
Figure 2 on page 5 shows how the control dictionaries of two input modules are combined into composite dictionaries by the linkage editor. The control dictionaries and their associated text are interrelated through a system of line numbers and pointers. Within an input module, each ESD item on which an address constant may depend has a line number (ESD identifier, or ESD ID); the line number indicates the position of the item relative to the other ESD items associated with the text.[1] Every item of text in an object or load module has associated control information that describes it. This control information includes the ESD ID of the ESD item for the control section that contains the text. (In Figure 2, the ESD ID of the text item that contains X and Y points to line 1 of the ESD for input module 1. The ESD ID of the text item containing Z points to line 1 of the ESD for input module 2.)

Each RLD item must point to two ESD items:

1. The ESD item for the symbol on which the address constant depends. This is referred to by the RLD <u>relocation pointer</u> (R pointer).

2. The ESD item for the control section that contains the address constant. This is referred to by the RLD <u>position pointer</u> (P pointer).

[1] In an object module, one type of ESD item (ID) may have associated text or address constants that depend on it (see "ESD Processing"). Such ESD items are excluded from the numbering system.



Input Module 1 — ESD

Symbol	Type*	Origin	Length
CSECT A	SD	000	500
CSECT C	ER	000	0
CSECT B	SD	500	1000

Input Module 1 — RLD

R	P	Flag	Address
1	1	F	300
2	1	F	400

Input Module 2 — ESD

Symbol	Type	Origin	Length
CSECT C	SD	000	2000

Input Module 2 — RLD

R	P	Flag	Address
1	1	F	200

Output Module — CESD

Symbol	Type	Origin	Length
CSECT A	SD	000	500
CSECT C	SD	500	2000
CSECT B	SD	2500	1000

Output Module — RLD

R	P	Flag	Address
1	1	F	300
2	1	F	400

Output Module — RLD

R	P	Flag	Address
2	2	F	700

*See "ESD Record Types"

Figure 2. Combining Control Dictionaries

In input module 1, X and Y are address constants in the same control section (CSECT A). X refers to a symbol in CSECT A; therefore, both pointers of its associated RLD item refer to the ESD entry for CSECT A (line 1). The value field of Y refers to a symbol in a different control section (CSECT C); therefore, the R pointer of its associated RLD points to the ESD entry for the external reference (line 2), whereas the P pointer refers to the ESD entry for its control section (line 1).

When the linkage editor combines the input modules, it must maintain this system of pointers by renumbering the ESD items to reflect their relative positions in the CESD of the output module. It must also update the RLD pointers and control information for the text so that they refer to the renumbered CESD items; the resulting CESD and RLD items are shown in Figure 2.

LINKAGE EDITOR OPTIONS

Options for error diagnostics, processing, and space allocation may be specified by parameters listed on the EXEC card, or they may be passed internally by a program requesting the linkage editor via LINK, LOAD, ATTACH, or XCTL macro instructions.[2] If the options are passed internally, the user can also provide

[2] For more information, see Data Administration Guide and Data Administration: Macro Instruction Reference.

alternates for the standard ddnames.[3] If the options are not user-specified, the defaults are used. The options that may be specified are as follows:

- LIST. A list of all linkage editor control statements is written on the diagnostic output data set.
- MAP. A module map, which lists external names and their storage addresses, is written on the diagnostic output data set.
- Cross-reference table (XREF). A cross-reference table, which includes a module map and a list of all address constants that refer to other control sections, is written on the diagnostic output data set.
- TERM. Error messages are directed to the terminal data set as well as to the diagnostic output data set.
- LET. The output module is marked as executable even though a severity 2 error condition was found during processing.
- Exclusive call (XCAL). The output module is marked as executable even though valid exclusive references between overlay segments have been made.
- No automatic library call (NCAL). The automatic library call mechanism is not to be invoked to resolve external references.
- SIZE (value 1, value 2). The user can supply two values to specify (1) the maximum amount of storage to be obtained for linkage editor processing and (2) what amount of the gotten storage is to be used as the load module buffer.
- DCBS. The linkage editor initialization routine examines the SYSLMOD DD statement for a DCB BLKSIZE parameter and uses that value, if it is acceptable, for its block size limit. If the DEVTYPE capacity is less than the specified block size, the DEVTYPE value is used.

MODULE ATTRIBUTES

When the linkage editor generates a load module in a library partitioned data set (PDS), it places an entry for the module in the PDS directory. This entry contains "attributes" describing the structure, content, and logical format of the load module. The control program uses these attributes to determine how a module is to be loaded, what it contains, whether or not it is executable, whether it is executable more than once without reloading, and whether it can be executed by concurrent tasks.

Some options for module attributes can be specified by the user; others are specified by the linkage editor as a result of information gathered during processing. In the following list, attributes marked with an asterisk (*) cannot be specified by the user:

- Reenterable (RENT). A reenterable module can be executed by more than one task at a time, and cannot be modified by itself or by any other module during execution; that is, a task may begin executing a reenterable module before a previous task has finished executing it.
- Refreshable (REFR). A refreshable module cannot be modified by itself or by any other module during execution. A refreshable module can be replaced by a new copy during execution by a recovery management routine, without changing either the sequence or results of processing.

[3] For more information, see JCL.

- Serially reusable (REUS). A serially reusable module will be executed by only one task at a time, and will either initialize itself and/or will restore any instructions or any data in the module that it alters during its execution.

- Overlay format (OVLY). A load module structured for overlay includes a segment table (SEGTAB) to enable the overlay supervisor to load the proper segments, and at least one ENTAB to assist in passing control from one segment to another. If a load module has the overlay format attribute, the reenterable, reusable, refreshable, hierarchy, scatter, addressing mode, and residence mode attributes cannot be present.

- Hierarchy format (HIAR). When a HIARCHY statement is detected, the "number" and "name" operand values are used in building the scatter table and the translation table. The high-order byte of each CSECT address entry contains the hierarchy number that is included in the GETMAIN request for main storage for program loading. Hierarchy information is used only when the program is loaded under the OS system.

- Test (TEST). If this module is an assembler language program and testing by the test translator or the TSO TEST command is desired, this attribute can be specified. Test will cause SYM records to be written. Note that modules using TESTRAN should not be marked with the RENT, REUS, or REFR attribute.

- Only loadable (OL). This attribute indicates that the control program may load this module only through the execution of the LOAD macro instruction.

- Scatter format (SCTR). In the OS environment, a load module in scatter format is suitable for block or scatter loading. The scatter table, translation table, and the relocation dictionary maintain logical linkage between scattered control sections when program fetch loads them into storage. In the virtual storage environment, the scatter format is ignored by program fetch. The SCTR attribute is, however, relevant in the link-editing of a nucleus for a virtual storage system, which requires the scatter and translate tables for its proper initialization.

- ALIGN2. If the ALIGN2 attribute is present, all control sections or named common areas specified on the PAGE control statements are placed in storage on 2K-byte page boundaries. The ALIGN2 attribute also aligns, on 2K-byte page boundaries, those control sections or named common areas associated with the "P" operand on the ORDER control statement.

- *Block format. If neither the overlay nor scatter attributes are specified, it is implied that the module can only be block loaded. The control program will load the module only if enough contiguous storage is available for the entire module.

- *Executable. This attribute indicates that linkage editor did not find any errors that would prevent successful execution. If this attribute is not present, the control program will not load the module.

- *Module contains one text record and no relocation dictionary records. This attribute indicates that the control program does not have to allocate storage for relocation dictionary items when loading the module. It also indicates that the first text record is the last one; there is no control record following it. The entire module can be read by program fetch in a single read operation.

- Downward compatible (DC). This attribute indicates that the module can be processed by either the level E or level F linkage editor. The downward-compatible attribute is assumed by the level E linkage editor. Modules processed by the level F linkage editor that are not marked "downward compatible" cannot be processed by the level E linkage editor.

- *Linkage editor assigned origin of first text record is zero. If this attribute is present, the first byte of instruction or data in the first text record is assigned to location zero.

- *Entry point assigned by linkage editor is zero. This attribute indicates that the entry point is at the first byte of the module.

- *No relocation dictionary items present. This attribute indicates to the control program that no allocation of storage is necessary to receive relocation dictionary items when program fetch loads them into virtual storage.

- Not editable (NE). This attribute indicates that the load module cannot be accepted by the linkage editor for subsequent processing. The CESD from an output load module is dropped to conserve space on the library.

- *Symbol statements present. If a module produced by the assembler language translator is to be tested by the test translator (TESTRAN) or the TSO TEST command, it may contain a testing symbol dictionary. In a load module, this dictionary contains the information from the SYM statement images that were in the input to the linkage editor.

- Authorization Code (AC). The output load module is assigned an authorization code that determines whether or not the load module may use restricted system services and resources.

- Addressing Mode (AMODE). The entry points—either main, true alias, or alternate—into the output load module are assigned the addressing mode that is to be in effect when the load module is entered at those entry points.

- Residence Mode (RMODE). The output load module is assigned the residence mode that applies to that load module when it is loaded into virtual storage for execution.

- *Read-Only Control Section. "Read-only" is an attribute of a control section deliberately created as such by specification of the control section as an RSECT to the language translator. The attribute is effective only when the control section is included in the nucleus load module for an MVS/XA system; otherwise it is ignored. The attribute is obtained from the ESD entries for the read-only control sections, and is reflected in the scatter table entries for those control sections in the nucleus load module.

LINKAGE EDITOR PROCESSING FOR ATTRIBUTES

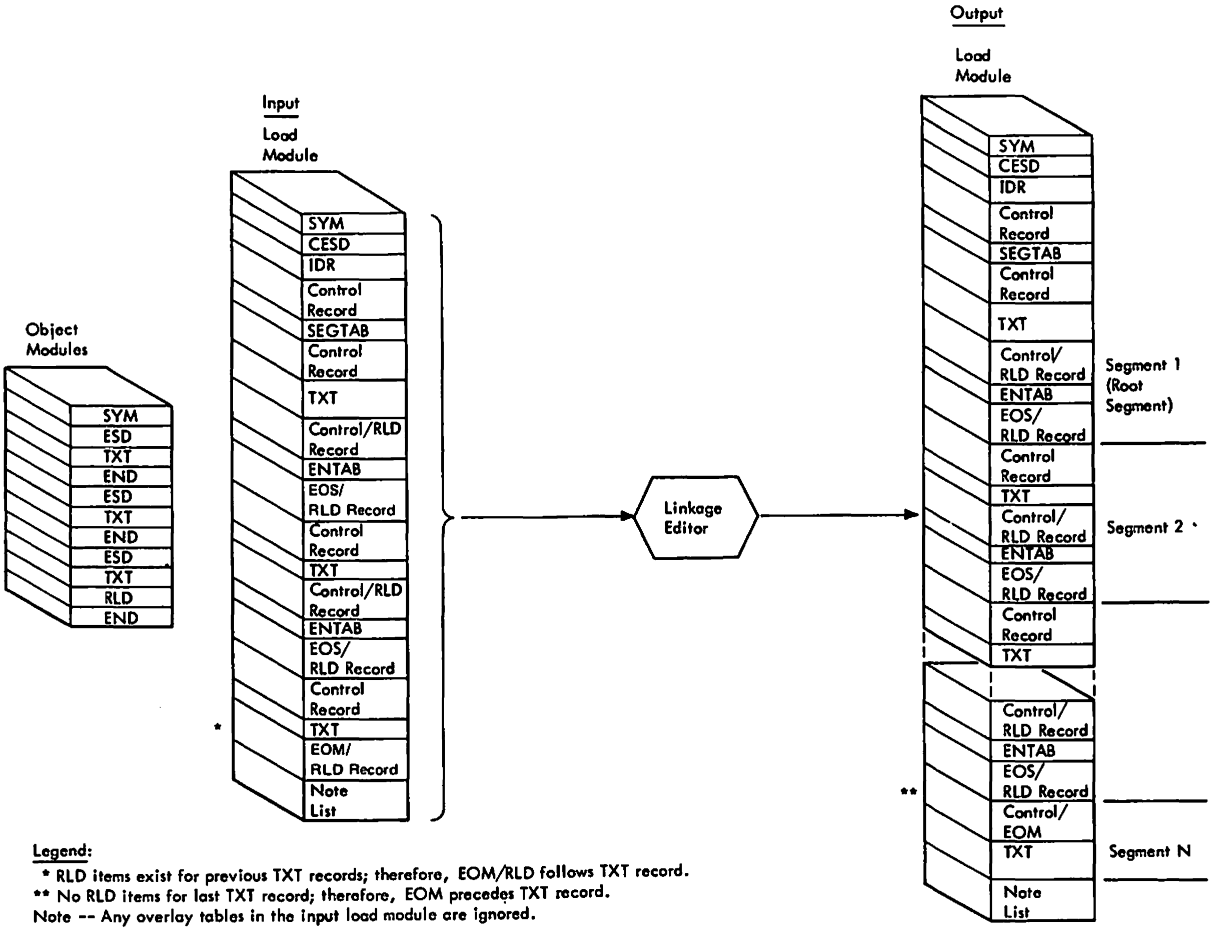
Several examples are given here of how linkage editor processing is affected by attributes specified by the user. Figure 1 on page 3 shows a simple case in which a single object module, containing only one control section, is processed by the linkage editor for block loading.

Figure 3 on page 9 shows the processing of an object module and a load module, each containing several control sections. In this example, test translator macro instructions were included in an assembler language source program, and test symbol (SYM) records were produced by the assembler language translator. The TEST and OVLY attributes were specified in the control information passed to the linkage editor, and overlay control statements were included in the input to the linkage editor.

With these attributes, the output load module produced by the linkage editor contains:

- SYM records to be used by the test translator. (If the TEST attribute is not specified, input SYM records are not included in the output load module.) These records contain blocked SYM and ESD statements created during a previous execution of the linkage editor. SYM records in load modules are passed unmodified through the linkage editor to the output load module.

- A composite ESD. CESD records contain the ESD items for the module. There is a maximum of 15 ESD items per record on the output record. The first 8 bytes of the CESD record contain control information pertaining to the ESD items in the record. This information consists of the ESD ID of the first ESD item and the number of bytes of ESD items in the record.



Legend:
* RLD items exist for previous TXT records; therefore, EOM/RLD follows TXT record.
** No RLD items for last TXT record; therefore, EOM precedes TXT record.
Note -- Any overlay tables in the input load module are ignored.

Figure 3. Linkage Editor Processing for the Overlay and TEST Attributes

- CSECT Identification Records (IDR). The IDRs are input from either an input load module, an END record, or the linkage editor IDENTIFY control statement. IDRs may contain data:

  - Identifying the language translator creating the control section, its level, and the translation date
  - Describing the most recent processing by the linkage editor
  - Describing any modification to the executable code of a control section
  - Supplied by a user and associated with the executable code of a control section

  Note: The user-supplied data is specified on the IDENTIFY control statement.

- A control record, or a composite control/RLD record, preceding each text record. The RLD portion, if present, contains the RLD items used to relocate the previous text.[4] The control portion may contain:

  - An end of segment (EOS) indication, if the following text record is the last text record of an overlay segment[5]
  - An end of module (EOM) indication, if the following text record is the last text record of the module[5]
  - The number of bytes of RLD information that follow, if it is a composite control/RLD record
  - The number of bytes of control information

  The control portion also contains the IDs and lengths (in bytes) of all the control sections in the following text, to a maximum of 60, and a channel command word (CCW). The channel command word contains the address assigned by the linkage editor to the first byte of that record, plus the total length of the record. This information is used by program fetch to read the following text.

  Note: The control portion contains as many IDs and lengths as there are control sections in the following text record.

- Text for each control section. Text records contain the instructions and data for the module. In overlay, the linkage editor produces two special types of text records, the segment table (SEGTAB) and entry table (ENTAB).

  SEGTAB, located in the root segment, is used by the overlay supervisor to keep track of the relationship of segments during execution. ENTAB is a separate control section that may be created by the linkage editor for each overlay segment. ENTAB is used by the overlay supervisor to determine the segment to be loaded when a segment not in the current path is referred to.

- A note list. A note list gives the location of each overlay segment in the output module library.

[4] If there are many RLD items for the previous text, there may be several RLD records preceding the next text record. The last of these is a control/RLD record.

[5] If there are no RLD items for the last text record, the control record that precedes the text contains the EOS or EOM indication. If there are RLD items, the EOS or EOM follows the text record (see Figure 3 on page 9).

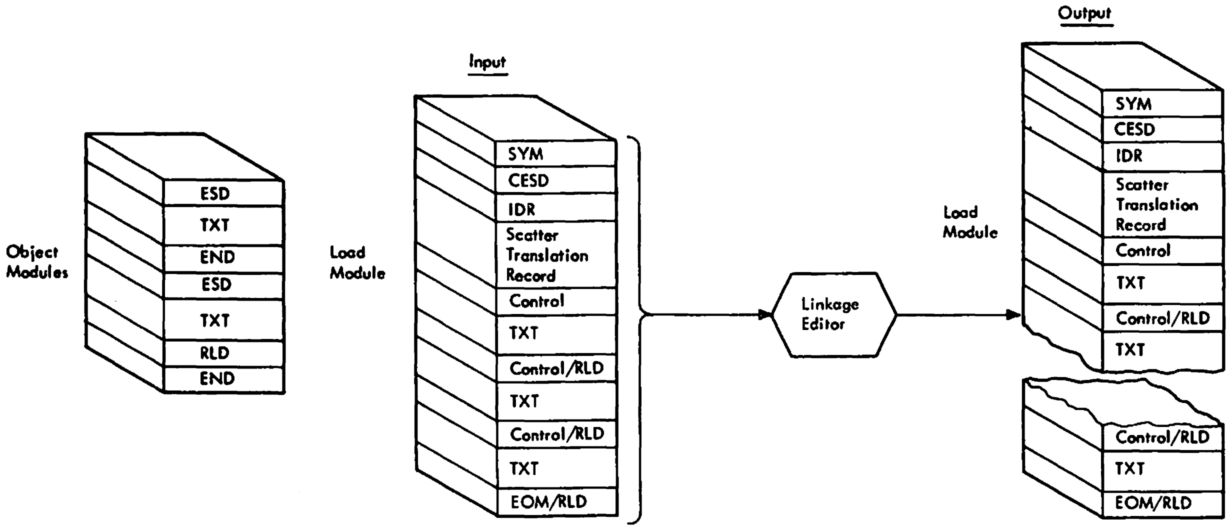


Figure 4. Linkage Editor Processing for the Scatter Load and TEST Attribute

Figure 4 shows the module structure when the scatter load and TEST attributes are requested. With these attributes, the output load module contains:

- SYM records.
- A composite ESD.
- IDR records.
- A scatter/translation record used by program fetch to compute the relocated addresses required for scatter loading the module into storage. The record contains a scatter table and a translation table. The scatter table is a list of control section addresses; the translation table correlates the CESD entry for each control section with the address indicated in the scatter table. (When a load module in scatter format is processed again by the linkage editor, this information is ignored.)
- Text for each control section, preceded by a control or control/RLD record describing it.
- RLD or control/RLD records containing any RLDs pertaining to the preceding text record.
- An EOM indication that marks the end of the module.

  The appendix "Input Conventions and Record Formats" contains the format of each record type.

INPUT/OUTPUT FLOW

Four data sets must be specified for linkage editor processing; their ddnames and functions are:

1. SYSLIN. This is the "primary input data set" containing object modules and control statements. All input from SYSLIN must be in 80-column card image format, unblocked or blocked, from 1 to 40 records per block. The SYSLIN source may be a card reader, magnetic tape, a direct access device, or a concatenation of data sets from different types of input devices.

2. SYSPRINT. This is the "diagnostic output data set." Diagnostic messages as well as any diagnostic options requested, such as a module map or cross-reference table, are written on SYSPRINT. It is a sequential data set and may be partitioned. The SYSPRINT device may be a printer, magnetic tape, terminal, or a direct access device.

3. SYSUT1. This is the "intermediate data set." The linkage editor uses this data set for temporary storage of text and RLD items being processed. SYSUT1 must be on a direct-access volume.

   Note: SYSUT1 is opened only when twopass processing is in effect.

4. SYSLMOD. This is the "output module data set." It is a partitioned data set on a direct-access volume. SYSLMOD contains load modules; their attributes are described in the user's portion of the directory entry for the member.

Two additional data sets may be specified for linkage editor processing; their ddnames and functions are:

- SYSTERM. This is the "terminal data set." Diagnostic messages are written on SYSTERM if the TERM option was specified. When the linkage editor is being executed in the time-sharing foreground, the SYSTERM device is always the terminal; when the linkage editor is being executed in the background, the SYSTERM device may be a printer, magnetic tape, or a direct-access device.

- SYSLIB. This data set is used by the linkage editor if there are any automatic library calls to be processed. SYSLIB can be defined only as a partitioned data set (PDS). The members of SYSLIB can be either load modules or object modules (but object modules and load modules cannot be contained in the same PDS, and a data set containing load modules cannot be concatenated with a data set containing object modules).

  When SYSLIB is opened, the linkage editor determines whether the PDS contains object or load modules by checking the record format field (RECEM) in the data control block (DCB). (The format is fixed (F) for object modules and undefined (U) for load modules. Load module records are of variable length.) If SYSLIB contains object modules, the linkage editor ignores the user's portion of the PDS directory entries for the object modules.

Other data sets may be read by linkage editor when it processes INCLUDE or LIBRARY statements specifying ddnames. Data sets referenced with INCLUDE statements may be either sequential or partitioned. SYSLIB, and any data sets specified in LIBRARY statements for use by automatic library call, must be partitioned.

The attributes for the "execute linkage editor" job step are the attributes specified on the EXEC statement. These attributes may be modified if a load module having different attributes is processed.

Figure 5 on page 13 shows the input/output flow. During the initial processing, SYSLIN, SYSPRINT, SYSLMOD, and SYSTERM (if the TERM option was specified) are opened. During input processing, the primary input is read from SYSLIN. If an INCLUDE statement is read in the primary input, the data set whose ddname is specified on the statement is opened and processed. At the end of all SYSLIN input, SYSLIB and any other data sets whose ddnames are specified on LIBRARY statements are processed through automatic library calls.

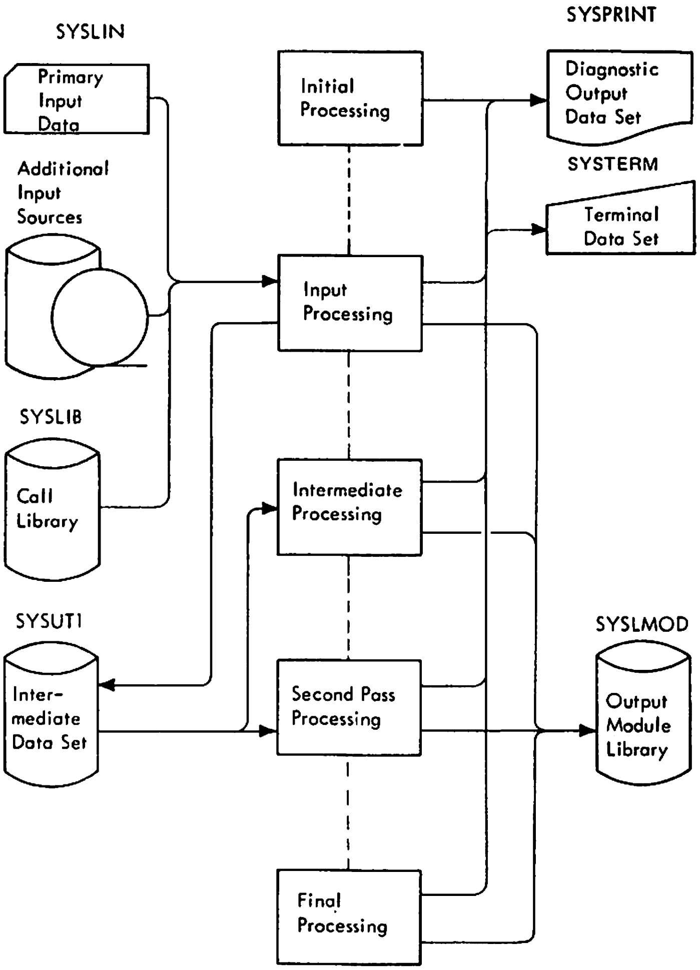


Figure 5. Input/Output Flow

If the TEST attribute has been selected, SYM records are written during input processing; text and RLD items are written sequentially on SYSUT1, except during single pass processing. The location of each text record on SYSUT1 is entered in a text note list. The location of each RLD record on SYSUT1 is entered in a RLD note list. If either note list overflows, a table overflow message (IEW0364) will be issued.

In intermediate processing, the CESD is written on SYSLMOD. If a scatter table, translation table, or SEGTAB is required, it is also written on SYSLMOD. The note list for the text and RLD items on SYSUT1 are read into storage. If a module map was required, the CESD is used in producing the map. If a cross-reference table was requested and all RLDs are in storage, the table is produced during intermediate processing.

During second pass processing, text and RLD records are read into storage from SYSUT1 in the order of assigned addresses within each segment (using the note lists to find the records), and are written out on SYSLMOD.

In final processing, the member name and any alias names are entered into the PDS directory entry for the output load module through the execution of the STOW macro instruction. If any coded diagnostic messages were written on SYSPRINT during linkage editor processing, a diagnostic message directory containing error message text is written out on SYSPRINT. If a cross-reference table was requested and was not produced during intermediate processing, SYSLMOD is opened for input, RLDs are read, and the cross-reference table is produced. At the end of final processing, SYSLMOD is closed (if it was opened for input). All other data sets are then closed and control is returned to the calling program, unless the SYSLIN input during input processing was terminated by a NAME statement. If a NAME statement terminated the primary input, control is returned to initial processing, and SYSLMOD is opened for output if it had been closed during final processing.

When multiple load modules are produced in a single execution of the linkage editor, SYSLIN, SYSPRINT, and SYSUT1 remain open for the entire execution. (A pointer in the DCB for SYSUT1 is repositioned to the beginning of the extent of SYSUT1 after each load module is produced.) If neither a module map nor a cross-reference table is requested, or if a cross-reference table is requested and all RLDs are in storage, SYSLMOD remains open for output for the entire linkage editor execution.

METHOD OF OPERATION

This section contains an introduction to the logic of the linkage editor, emphasizing the flow of primary data and control information through tables and buffers, and providing functional descriptions of its phases.

LOGIC OF THE LINKAGE EDITOR

The linkage editor can be functionally divided into five phases:

- Initialization
- Input (first pass) processing
- Intermediate (first pass) processing
- Second pass processing
- Final processing

Operation diagrams at the end of this section illustrate the functional operation of the linkage editor. The shaded areas of the diagrams correspond to operations described in the text.

Initialization

When the linkage editor receives control from the job scheduler or a calling program, it performs initialization functions in preparation for all subsequent processing (see "Diagram 3. Initialization" on page 77). The operations included in initialization are:

- Build the all-purpose table (APT) and enter addresses and descriptions of all other tables and buffers into it.
- Analyze the attributes and options passed by the calling program (specified by the programmer), and save them in the all-purpose table.
- Initialize DCBs and open data sets to be used during linkage editor processing.
- Allocate storage for all tables, buffers, and work areas to be used by linkage editor processing.

When all initialization functions are completed, the linkage editor is ready to accept input.

Input Processing

All linkage editor input is processed initially during the first pass (see "Diagram 4. Input Processing" on page 78). Object modules from SYSLIN (primary input data set) are read into the SYSLIN buffer. Object modules from SYSLIB or a specified user's library (secondary input data sets) are read into the object module buffer. Text records in load modules from SYSLIB or a user's library are read into the input text buffer; all other load module records are read into the first pass RLD buffer. The various records that constitute these modules are processed as follows.

Control Statements: These records, which may precede or follow object modules, contain information that is later used in symbol resolution and that specify libraries containing secondary input. Depending on the type of control statement, entries are made in either the all-purpose table (APT) or the composite external symbol dictionary (CESD).

ESD Records: These records from object modules, and CESD records from load modules, describe symbols that have been defined for external use. Entries for the symbols are made in the CESD. Entries are made in the renumbering table to allow the translation of the input ESD identifiers (IDs) into new CESD IDs. Entries are made in the delink table for symbols that are to be deleted or replaced.

TXT Records: These records, containing the instructions and data of the program, are moved from the SYSLIN buffer and object module buffer to the input text buffer (text records from load modules are read directly into the input text buffer). They are arranged in the proper sequence and recorded in the text I/O table and the text note list. When the input text buffer is filled, its contents are written onto SYSUT1; if it does not become filled, text records are retained in the buffer and "single pass" processing is in effect. Text note list entries contain the location of text records (SYSUT1 address or buffer address) and other descriptive information. Text I/O table entries contain information identifying text records by ESD ID.

RLD Records: These records, to be used later in relocating address constants, are moved from the SYSLIN buffer and object module buffer to the RLD buffer. The relocation and position pointers (R and P pointers) are updated, using control information from the renumbering table and the delink table. RLD items are examined and marked for future processing. If V-type (branch-type) address constants are found in overlay programs, entries are made in the call list for use during intermediate processing. When the RLD buffer is full, RLD records are written on SYSUT1, and control information identifying RLD records by size (byte count), P pointer, and location on SYSUT1 is entered in the RLD note list. If the RLD buffer does not become filled, RLD records are retained in the buffer and single pass processing is in effect.

SYM Records: These records, which are not involved in linkage editor processing, are gathered in the RLD buffer and are written directly on SYSLMOD if the TEST attribute has been specified. If TEST has not been specified, SYM records are ignored.

IDR Records: These records, which contain data either from an input load module, an END record, or the linkage editor IDENTIFY control statement, supply information concerning the processing history of the modules in which the IDRs occur. If the data is from an input load module, control is passed to the IDR processor HEWLFIDR. If the data is from an END record, the data refers to the compiler that created the object module. The compiler or translator data is passed in a parameter list to the IDR processor. The user data, supplied via the linkage editor IDENTIFY control statement, is converted into a parameter list and passed to the IDR processor.

When all input records have been processed (all external symbols have been entered into the CESD), control is passed to intermediate processing.

Intermediate Processing

The operations included in intermediate processing (see "Diagram 5. Intermediate Processing" on page 79) have two primary objectives: (1) to assign relative storage addresses to symbols in the CESD and (2) to write some of the records to be included in the output load module on the SYSLMOD data set. The MAP and XREF options may also be processed during intermediate processing.

Address Assignment: Entries that require no further processing are deleted from the CESD; all other CESD symbols are assigned temporary linked addresses. Relocation constants are determined for all control sections, and the relocation constant table (RCT) is built.

For all programs in overlay, additional processing is required. The calls list is used to determine ENTAB entries to be placed in the CESD, and the downward calls list is built. The segment length table (SEGLGTH) is built, and segment relocation constants are computed. Temporary linked addresses in the CESD and entries in the relocation constants are computed. Temporary linked addresses in the CESD and entries in the relocation constant table are adjusted for overlay by adding to them the segment relocation constants.

Temporary linked addresses and relocation constants are combined to determine final linked addresses for symbols, and the results are placed in the CESD. The alias table is built from alias symbols in the CESD. At this point, CESD processing is complete.

MAP/XREF Processing: If the MAP option has been specified, and enough table space is available, a module map containing sorted CESD items is built and written on SYSPRINT. If the XREF option has been specified and all RLDs are in storage, a cross-reference table is built from RLDs (in the RLD buffer) and written on SYSPRINT. If all RLDs are not in storage, or the MAP table size is insufficient, the cross-reference table is not built but is deferred until final processing.

Intermediate Output: The principal function of this section of intermediate processing is to write the CESD on the output load module data set (SYSLMOD). The half ESD (HESD), containing control information from CESD entries, is built and held in storage for use during second pass processing. The text I/O table is reorganized according to the sequence in the order table, and scanned to determine the ID of the last control section containing text in the program (or in each segment of an overlay program); this information is placed in the high ID table (HIID), and noted in the HESD for use during second pass processing.

For a program in overlay, the segment table (SEGTAB), which defines the relationships among segments, is built and written (with a control record) on SYSLMOD.

For a program that is to be scatter loaded in the OS environment (MFT or MVT), a scatter table and a translation table are built from information in the CESD, and scatter/translation records are written on SYSLMOD.

The IDRs are written out on the output load module data set (SYSLMOD).

Second Pass Processing

The objectives of second pass processing (see "Diagram 6. Second Pass Processing" on page 80) are (1) relocating address constants in the text and (2) writing on the SYSLMOD data set the remaining records that constitute the output load module.

Text records are read from SYSUT1 (intermediate data set) into the second pass text buffer, using the text I/O table and the text note list to locate the records on SYSUT1. The text I/O table is also used to determine the order in which text records are to be processed. RLD records associated with the text being processed are read into the second pass RLD input buffer, using the RLD note list to locate the required records.

Single Pass Processing: If the linkage editor did not write text or RLD records on SYSUT1, single pass processing is in effect for these records. The records are accessed directly in the input text buffer and the RLD buffer, which are physically the same storage areas as the second pass RLD input buffer. If text records or RLD records were written on SYSUT1, they are read back into the same locations.

Relocation: Address constants described by RLD items are moved from the second pass text buffer to a work area, where relocation is performed. The manner in which each address constant is relocated depends on whether it is a V-type (branch-type) or an A-type (nonbranch-type) address constant, or a pseudo register (type 1 or type 2).

The V-type address constant can refer to a named location in some other control section (branch type address constant). The value field of such a V-type address constant always contains a zero because the address was not known at compilation time. During second pass processing, the linkage editor address (absolute relocation factor) that was assigned to the symbol and saved in the HESD is inserted in the value field. This is called absolute relocation.

If the V-type address constant is in an overlay program, the address of an ENTAB entry for the symbol and the segment number of the current text is inserted in the value field. (ENTABs are created in the second pass RLD buffer from information in the HESD and the entry list, which contains an entry for each V-type address constant in the path of a referred-to symbol.)

The value field of an A-type address constant that refers to a named location in the same input module (nonbranch-type address constant) contains an address assigned by the language translator. During second pass processing, this address is modified by adding or subtracting the relative relocation factor that was determined for the symbol referred to by the address constant. Relative relocation factors are saved in the relocation constant table. This process is called relative relocation.

When each address constant is relocated, it is placed back in the text, and the address field of the associated RLD item is updated. The RLD item is then moved to the second pass RLD output buffer. When all address constants in the text buffer are relocated, the text is written on SYSLMOD, followed by the associated RLD items. A control record pertaining to the next text record is written on SYSLMOD following the RLD records. If the output load module is structured for overlays, a TTR list, containing the address of the first control record of each segment (for the first segment, the list contains the address of the first text record) is also created and retained in virtual storage.

Second pass processing continues until all segments in the output module are processed. The last control record contains end-of-module indicators. Control is then passed to final processing.

Final Processing

The objectives of final processing (see "Diagram 7. Final Processing" on page 81) include writing remaining output to SYSLMOD, producing certain optional output, and "cleanup" functions.

The partitioned data set directory for SYSLMOD is completed, including modifications for ALIAS symbols (found in the ALIAS table), and a STOW macro is issued. The TTR list, containing the address of the first text record in each segment, is written on SYSLMOD for overlay programs.

The error logging map, produced as errors are encountered throughout linkage editor processing, is scanned and an error diagnostic directory is built and written on SYSPRINT. If the TERM option was specified, the error diagnostic directory is also written on SYSTERM. Storage allocated to the linkage editor is released.

If the MAP or XREF option is specified and was not processed during intermediate processing, RLD records are read from SYSLMOD, and a cross-reference table is built and written on SYSPRINT. If the alternate MAP table is too small, warning message IEW0801 will be issued.

At the completion of linkage editor processing, control is returned to the calling program.

INITIALIZATION

The initialization phase comprises modules HEWLFINT, HEWLFOPT, and HEWLFDEF.

When the linkage editor begins processing, it readies the all-purpose table, analyzes control information, opens necessary data sets, and allocates space to buffers and tables.

PREPARING THE ALL-PURPOSE TABLE (APT)

The linkage editor maintains the all-purpose table as the common communication area for all internal functions (see Figure 27 on page 68 for the contents of the all-purpose table). The basic information in the all-purpose table is added to during initialization as operating conditions are learned. This information includes the results of the control information analysis and descriptions of the tables and buffers built by the linkage editor.

ANALYZING CONTROL INFORMATION

When the linkage editor receives control from the job scheduler, or from another program via a CALL macro instruction, control information may be passed to it. This information includes the options that control linkage editor processing, and the attributes to be assigned to the output load module. A calling program may also provide a substitute list of ddnames to be used in place of the standard names, and a PDS directory name for the output load module.

During initialization, the specified attributes and options are interpreted, checked for validity against an attribute-option table, and recorded in the all-purpose table. When options with associated values are recognized, the linkage editor also saves the value in the all-purpose table. (For example, the SIZE option gives user-chosen values to be used instead of the default values.)

Besides being checked for valid specification, the attributes and options are also checked to ensure that they are requested only in allowable combinations. When mutually exclusive attributes or options are noted, the dominant attribute or option is retained; the other is ignored (see Figure 6 on page 20).

IDR Records: These records, which contain data either from an input load module, from an END record, or from the linkage editor IDENTIFY control statement, supply information concerning the processing history of the modules in which they occur. If the data is from an input load module, control is passed to the IDR processor, HEWLFIDR. If the data is from an END record, the data refers to the compiler that created the object module. The compiler or translator data is passed in a parameter list to the IDR processor. The user data supplied via the linkage editor

IDENTIFY control statement is converted into a parameter list and passed to the IDR processor.

	OVLY	TEST	XREF	MAP	LET	RENT	REUS	HIAR	SCTR	NE	XCAL	LIST	NCAL	OL	DC	SIZE	REFR	DCBS	TERM	AC	ALIGN2	AMODE
TEST																						
XREF																						
MAP			X																			
LET																						
RENT	X																					
REUS	X					X																
HIAR	X																					
SCTR	X							X														
NE		X	X	X																		
XCAL																						
LIST																						
NCAL																						
OL																						
DC																						
SIZE																						
REFR	X																					
DCBS																						
TERM																						
AC																						
ALIGN2																						
AMODE	X																					
RMODE	X																					

Note: An X indicates incompatible attributes; the attribute that appears lower on the list is ignored. For example, to check the compatibility of XREF and NE, follow the XREF column down and the NE row across until they intersect. Because an X appears where they intersect, they are incompatible attributes. NE is ignored.

Figure 6. Incompatible Module Attributes and Program Options

OPENING DATA SETS

After the standard ddnames (or passed ddnames) have been entered into the proper DCBs, the data sets always required for linkage editor operation are opened. These are the SYSLIN, SYSPRINT, and SYSLMOD data sets. If the TERM option was specified, the SYSTERM data set is also opened.

Note: SYSLIB is opened during input processing if automatic library calls or INCLUDE statements are recognized. SYSUT1 is opened only for twopass processing.

When SYSLIN is opened, the "unlike attributes" indicator in the associated DCB is set to signify that SYSLIN may be a concatenation of data sets with varying blocking factors.

In preparation for the opening of the SYSLMOD data set, the linkage editor obtains storage for the JFCB for the data set and reads the JFCB into that storage. From the JFCB, the linkage editor obtains the data set disposition and the block size, in case the DCBS option is specified. The block size in the JFCB is zeroed in order to obtain the DSCB block size when the data set is opened.

In addition, a DEVTYPE macro instruction is issued to obtain the maximum block size for the type of device on which the SYSLMOD data set resides. The value obtained will be used subsequently in determining the output block size.

If the SYSLMOD data set resides on a shared device and if the data set is not a temporary data set, the linkage editor reserves the shared device for the duration of the job step. If the SYSLMOD data set does not reside on a shared device but a disposition of SHR was specified for it on the SYSLMOD DD statement, the linkage editor enqueues the SYSLMOD data set for the duration of the job step.

The SYSLMOD data set is opened with an OPENJ macro instruction, specifying the JFCB previously read and modified.

During the opening of the SYSLMOD data set, the block size to be used for output to the data set is determined in the open exit routine. The appropriate block size is selected considering the following factors: The value obtained from the DEVTYPE macro instruction, establishing the absolute maximum block size; the block size in the DSCB for an existing data set, which can be increased but not decreased; the block size from the SYSLMOD DD statement, when the DCBS option is used; the implied maximum block size of 1024, when the DC option is used; the implied minimum block size of 1024, when the SCTR option is used; the absolute minimum block size of 256.

ALLOCATING VIRTUAL STORAGE

To obtain storage for buffers and tables, the linkage editor issues the GETMAIN macro instruction specifying the minimum amount of additional virtual storage required for operation. The minimum provides for overlay or hierarchy tables if these options are selected, and, for an area, a 12K-byte block of storage that is returned by means of a FREEMAIN macro instruction for use by system and data management functions. The minimum also includes the added space required for the primary input buffer when the SYSLIN data set contains blocked records. If the minimum is not available, control is not returned to the linkage editor; instead, a system abnormal termination occurs.

Buffer Allocation

When the supervisor returns virtual storage space, the linkage editor determines whether the area is sufficient to use maximum lengths for the SYSLIN, SYSPRINT, and object module buffers. If the space is not sufficient, intermediate or, when necessary, minimum buffer lengths are used.

The RLD and text buffers are then assigned storage. The default text buffer length (48K bytes) is used unless a specific allocation was requested via the SIZE parameter. The RLD buffer is also assigned the minimum length, unless the SIZE parameter allows it to be given additional space. In this case, the increased length depends on the amount of storage remaining after the text buffer has been allocated.

Note: Space allocated for buffers is not released until linkage editor processing is completed.

Table Allocation

Following buffer allocation, the linkage editor assigns storage to its fixed-length and variable-length tables. In initial allocation, the linkage editor determines the minimum storage required by each table. The size of each table and the number of entries in each table are saved in the all-purpose table.

Storage is then reallocated. The storage in excess of the minimum required for all the tables is determined. The excess is used to expand proportionately the variable-length tables. Then, the size of each table and the number of entries per table are calculated. This information and the newly assigned table addresses are saved in the all-purpose table. When all linkage editor processing is completed, all table space is released.

INPUT PROCESSING

The input processing phase comprises modules HEWLFINP, HEWLFINC, HEWLFSCN, HEWLFESD, HEWLFSYM, HEWLFRAT, HEWLFEND, HEWLFIDR, and HEWLFRCG.

The operations performed during input processing depend on the nature of the input; special processing is required for each input record type. Each input record is read, using one of two read blocks. The first read control block contains the address of the SYSLIN buffer, the address of the SYSLIN DCB, and the block size and logical record length. The second read control block contains the address of the buffer for library records (object module buffer or load module buffers), the address of the library DCB, and the block size and logical record length. A pointer is used to indicate which read control block is to be used for the input record. Initially, the pointer is set to the SYSLIN read control block.

The type of input processing required is determined by the following conditions:

- For all object module records whose first column character is a blank, control statement scanning is required, provided that the record is not encountered "in module." (Control statements encountered within a module cause an error indication.)

- Either object module processing or load module processing is required, depending on the type of input module. Only object modules are read from SYSLIN. Input modules from libraries are identified by record format. (Fixed format (F) indicates object modules; undefined format (U) indicates load modules.)

- When an INCLUDE control statement is detected during normal processing, "include" processing is initiated. At end-of-input from the specified include library, normal processing resumes. If an INCLUDE control statement is detected during "include" processing, "include" processing is reinitiated for the new include library.

- At end-of-input from SYSLIN, automatic library call processing is required if the NCAL option (no automatic library calls) was not selected. If the NCAL option was selected, input processing is complete.

- At end-of-input from SYSLIB during automatic library call processing, automatic library call processing is reinitiated.

- If a NAME statement, which may indicate a multiple execution of the linkage editor, is detected during control statement scanning, processing proceeds as if an end-of-input has occurred on SYSLIN (automatic library call processing is performed).

- If an end-of-input occurs on SYSLIN but no valid input was received, linkage editor processing is terminated.

Reading Blocked Input

The linkage editor can accept blocked card image input from the SYSLIN data set and blocked object module records from the SYSLIB data set (or from a user's library). Generally, the record format, block size, and logical record length are established either when the data set is created, or when they are specified on the DD statement for the data set in an execution of the linkage editor. If the BLKSIZE field is not specified, the linkage editor assumes a block size of 80. The logical record length (LRECL) is fixed at 80.

If the block size specified on primary input exceeds the allowable maximum or is not a multiple of the logical record length, an error message (IEW0594) is issued and linkage editor processing is terminated; if the invalid block size is specified on input from a library, the data set is ignored, but processing is not terminated. The block size specified by the user is used as the read count; if a short block is read, the linkage editor determines (via an exit at SYNAD) whether the length of the short block is valid (a multiple of the logical record length), and the number of the logical records it contains.

If SYSLIN is a concatenation of data sets, the input processor reexamines the block size fields whenever a data set boundary is crossed to determine whether their values have changed.

Record Lengths for SYSPRINT

In determining record lengths for SYSPRINT, the linkage editor first checks the block size unless time sharing is in effect. If the BLKSIZE is not specified by the user, it is set equal to 121. If the block size exceeds the allowable maximum or is not an integral multiple of 121, linkage editor processing is terminated and a condition code of 16 is returned. If the block size is a multiple of 121, it is not changed and the logical record length for output to SYSPRINT is set equal to 121.

If time sharing is in effect, both the block size and the logical record length for SYSPRINT are set equal to 81. When the linkage editor is being executed in the time-sharing foreground, header messages are printed only once and all line-counting functions are ignored.

Note: If SYSPRINT is a member of a partitioned data set and the DSCB is changed (by setting the logical record length or changing the block size), it may be impossible to use the information in the other members of the PDS.

Record Lengths for SYSTERM

The block size and the logical record length for output to SYSTERM are set equal to 121 unless time sharing is in effect. If time sharing is in effect, the block size and the logical record length are set equal to 81.

Note: If SYSTERM is a member of a partitioned data set and the DSCB is changed (by setting the logical record length or changing the block size), it may be impossible to use the information in the other members of the PDS.

Control Statement

When an input record is found to be a control statement (a blank in column 1), it is scanned to detect format errors and continuation of comments or operands. A vector table is scanned to determine the appropriate processor; separate processing is required for each type of control statement (INCLUDE, REPLACE, LIBRARY, CHANGE, INSERT, OVERLAY, ENTRY, ALIAS, NAME, SETSSI, IDENTIFY, HIARCHY, ORDER, PAGE, SETCODE, EXPAND, and MODE). "Diagram 8. Control Statement Processing" on page 82 illustrates general processing of each control statement type.

The general format for linkage editor control statements is shown in Figure 7. The control statement scanner interprets symbols enclosed in parentheses as "level 1" symbols; symbols not enclosed within parentheses are "level 0.." ENTRY, ALIAS, INSERT, HIARCHY, SETSSI, and PAGE control statement operands contain only level 0 symbols. CHANGE, IDENTIFY, SETCODE, EXPAND, and MODE statement operands always contain both a level 0 symbol and a level 1 symbol.

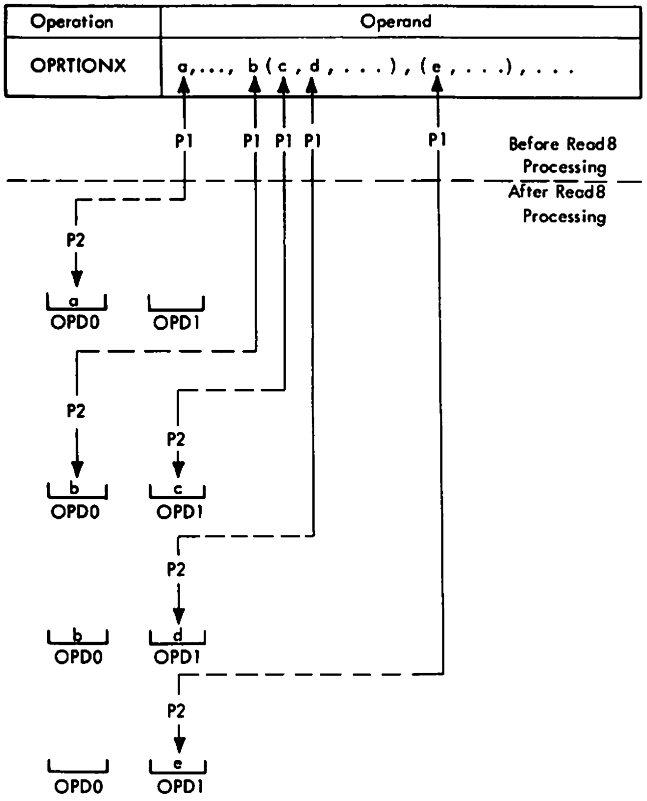


Figure 7. Control Statement Scanner Operation

The operands of REPLACE, INCLUDE, OVERLAY, NAME, and ORDER control statements contain level 0 symbols, or both level 0 and level 1 symbols. LIBRARY statement operands may contain level 1, or both level 0 and level 1 symbols. The operation to be performed depends on the operand format.

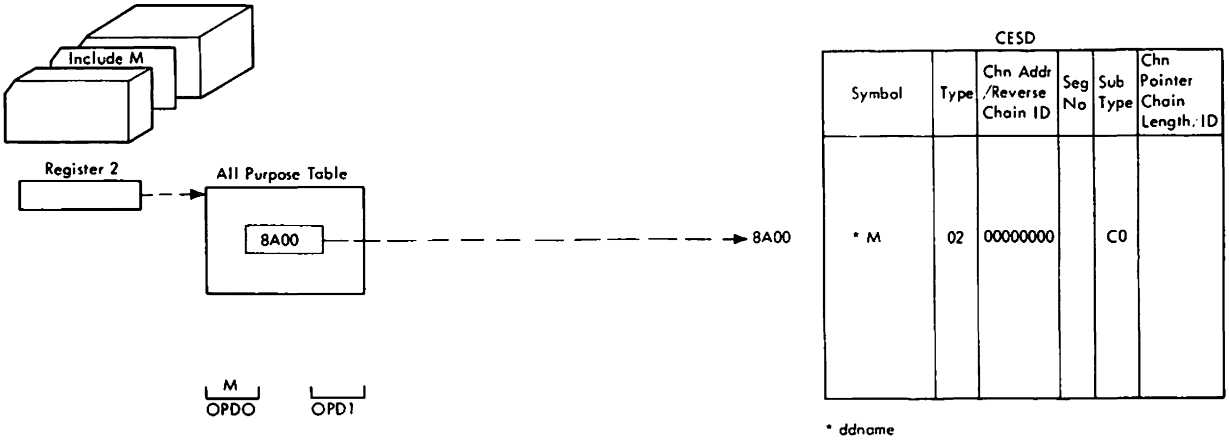
The control statement scanner searches a vector table for the operation symbol to determine the associated control statement processor. It then analyzes the operands using two work areas, "OPD1" and "OPD0", and two pointers, "P1" and "P2". OPD1 is used for level 1 operand symbols; OPD0 is for level 0 operand symbols. P1 points to the operand symbol being analyzed; P2 points to either OPD0 or OPD1, depending on the level of the operand symbol referred to by P1.

An operand symbol referred to by P1 is placed by the READ8 routine into the work area referred to by P2. Parentheses and commas control the switching of pointer P2 between the work areas. For example, when a left parenthesis is encountered, P2 moves to OPD1 because a level 1 operand symbol will follow. When a comma, blank, or right parenthesis is detected, the PROCENTY routine passes control to the control statement processor that was previously found during the search of the vector table.

When an IDENTIFY control statement is read by the control statement processor, a switch is set on for the special string option utilized by IDENTIFY. When the switch in the control statement processor is picked up by the READ8 routine, it sets another switch, permitting up to 40 characters to appear in the IDENTIFY operand. This allows any character, including embedded blanks, to appear between single quotation marks.

Control Statement Processors

When the operand symbols have been read into work areas OPD0 and OPD1, control is passed to the control statement processor at the saved entry point. Scanning of the control statement resumes when the control statement processor returns control. The individual control statement processors are described in the following paragraphs.



Symbol	Type	Chn Addr /Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chain Length/ID
* M	02	00000000		C0	

* ddname

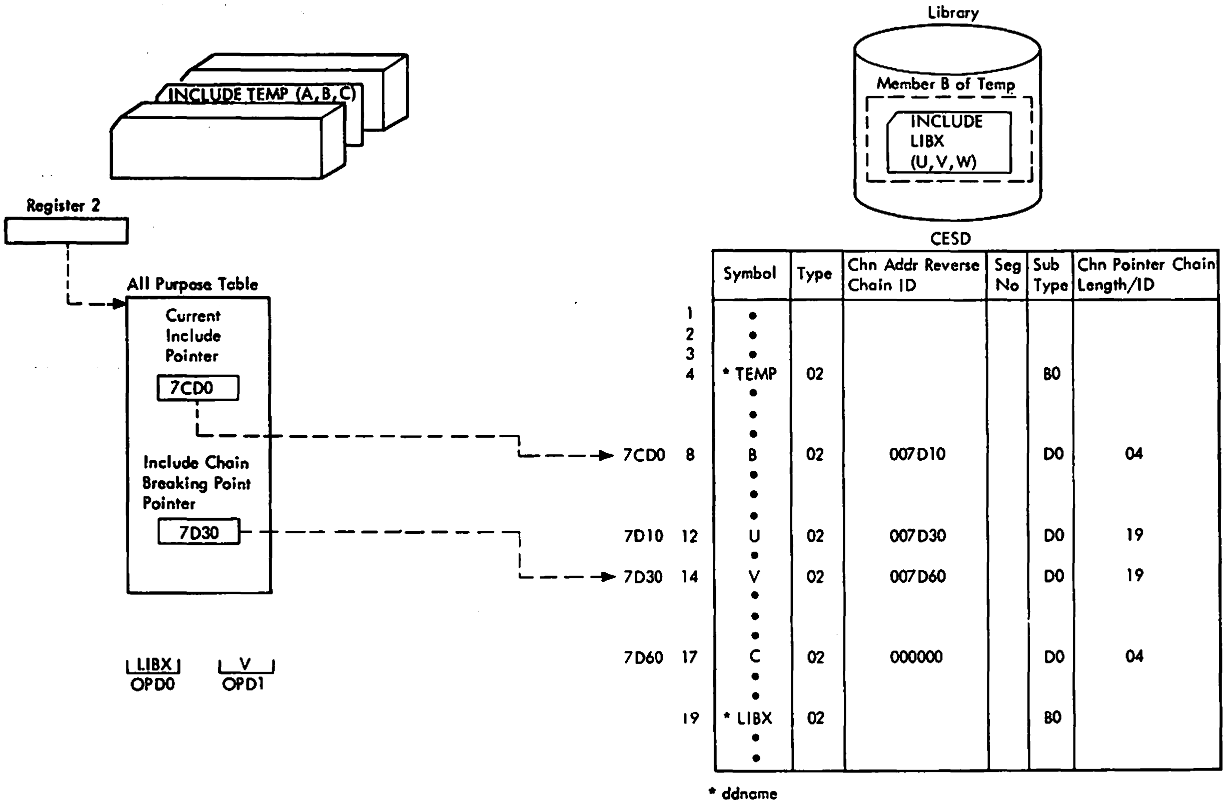
Figure 8. INCLUDE Statement Processing for a Sequential Data Set

INCLUDE STATEMENT PROCESSOR: The INCLUDE statement processor builds a chain in the CESD of items to be included. Each item in the chain contains the address of the next item in the chain (in the chain/address field—bytes 9, 10, and 11). The last item in the chain contains zeros in this field.

Chained include items have two kinds of subtypes: "include with pointer" and "include without pointer." In Figure 8, the statement INCLUDE M defines M as a sequential data set. The INCLUDE statement processor creates an entry for the ddname M in the CESD, with the subtype "include without pointer."

In the statement INCLUDE LIBX (A), A is defined as a member of a PDS. The INCLUDE statement processor creates an entry for A in the CESD, with the subtype "include with pointer." The pointer is in the chain pointer/chain ID field (bytes 14 and 15); it contains the CESD line number of the ddname LIBX. A single ddname, such as LIBX, may be referred to by several pointers.

Figure 9 describes INCLUDE statement processing with nested members.



		Symbol	Type	Chn Addr Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chain Length/ID
	1	•					
	2	•					
	3	•					
	4	* TEMP	02			B0	
		•					
		•					
		•					
7CD0	8	B	02	007D10		D0	04
		•					
		•					
		•					
7D10	12	U	02	007D30		D0	19
		•					
7D30	14	V	02	007D60		D0	19
		•					
		•					
		•					
7D60	17	C	02	000000		D0	04
		•					
		•					
	19	* LIBX	02			B0	
		•					
		•					

* ddname

Figure 9. INCLUDE Statement Processing With Nested Members

- The statement INCLUDE TEMP (A, B, C) indicates that A, B, and C are members to be included from library TEMP.

- Member B contains the nested statement INCLUDE LIBX (U, V, W); this is the last statement processed in member B.

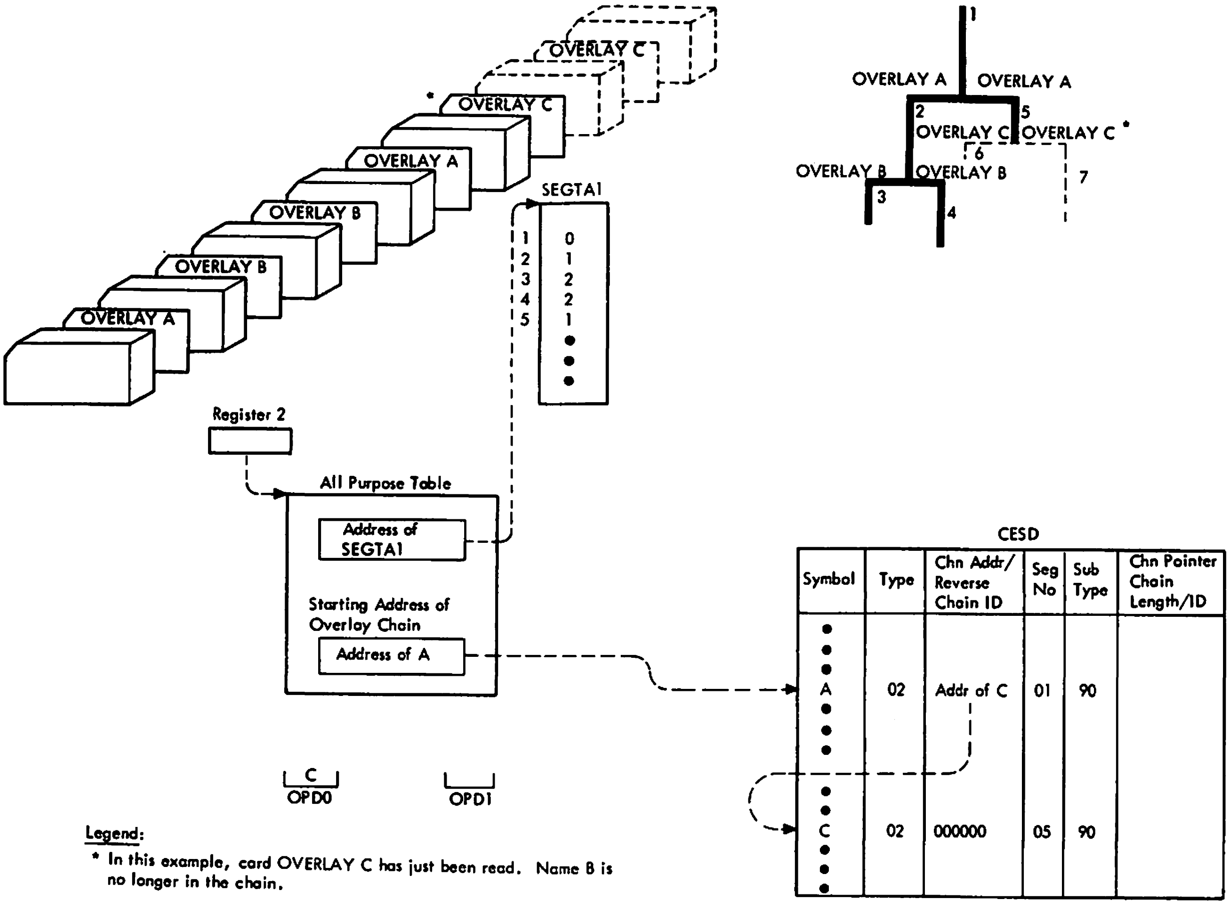
- The CESD is shown at the time when the control statement scanner has read operand V, but not W. The INCLUDE statement processor has created a CESD line for operand V in the LIBX include chain. C is currently the last item in the TEMP include chain. When the control statement scanner reads operand W, the INCLUDE statement processor enters a CESD line for W between V and C; this process is distinct from the one that actually searches the members U, V, and C on the library (see "INCLUDE Processing").

At the time chosen for this example, the data set member B is being read; data set member A has been read and therefore is no longer in the CESD as a member name, but data set members U, V, and C have not yet been read.

The chained CESD entries created by the INCLUDE statement processor are later processed by the include processor.

OVERLAY STATEMENT PROCESSOR: The OVERLAY statement processor maintains a record of the current segment number and updates it by one each time a new OVERLAY statement is encountered. The relationship of segments in an overlay tree structure is kept in the segment path table (SEGTA1) (see Figure 10). Entry n in SEGTA1 contains the number of the segment that precedes the nth segment of the overlay tree structure (the next higher segment in its path). The OVERLAY statement processor creates a chain of overlay items in the CESD and updates SEGTA1. If the level 1 operand (REGION) is detected, the current region number is incremented by one, and a zero is entered as the previous segment number in SEGTA1.

If an OVERLAY statement is encountered that refers to a node point higher in the overlay tree structure, all symbols identifying node points higher in the path are removed from the chain; their CESD lines are marked "null." For example, in Figure 10, when the statement OVERLAY A is encountered after segment 4, the CESD entry for symbol B is marked null and is no longer in the chain. If an OVERLAY B statement was encountered at the end of segment 5, a new node point would be established for B, and symbol B would again be entered in the CESD.



Symbol	Type	Chn Addr/ Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chain Length/ID
•					
•					
•					
A	02	Addr of C	01	90	
•					
•					
•					
•					
•					
C	02	000000	05	90	
•					
•					
•					

Legend:
* In this example, card OVERLAY C has just been read. Name B is no longer in the chain.

Figure 10. Overlay Statement Processing

HIARCHY STATEMENT PROCESSOR: The HIARCHY routine first determines if the hierarchy number is valid. If it is invalid, the statement is printed; an error message is written and the remainder of the statement is ignored. If the number is valid, it is converted to binary and saved for the SCAN routine.

Processing of the statement continues with the collection of the next symbol, up to a comma or a blank. The CESD is searched for this symbol; the location in the hierarchy table corresponding to this CESD item is set to the hierarchy number specified. (The hierarchy table is built during initialization if HIAR was specified. The hierarchy table consists of 1 byte per entry in a one-to-one correspondence with the number of items allocated to the CESD. The address of this table is kept in a fullword in the all-purpose table.)

If the symbol does not appear in the CESD, the symbol is entered in an unused entry in the CESD, marked external reference, and the hierarchy number is stored in the corresponding entry in the hierarchy table. This procedure is repeated for each additional symbol in the HIARCHY statement.

The intermediate output routine uses the hierarchy table to place the hierarchy number associated with each CESD item in the scatter/translation table.

INSERT STATEMENT PROCESSOR: The insert statement processor scans the CESD for the symbol indicated in the INSERT statement. If the symbol is found, the segment number field is changed to the number of the segment that contains the INSERT statement. If the symbol is not found in the CESD, a new ER entry in the CESD is created. In either case, the CESD entry is marked "insert" in the subtype field, and the segment number of the INSERT statement is placed in the segment number field.

REPLACE AND CHANGE STATEMENT PROCESSORS: The REPLACE and CHANGE statement processors build a chain of CESD entries. Each entry to be replaced, changed, or deleted is so marked in the subtype field. The ESD processor examines the replace/change chain before processing any ESD item. Because a REPLACE or CHANGE statement applies only to the module that immediately follows it in the input, the replace/change chain is removed from the CESD at the end of the module.

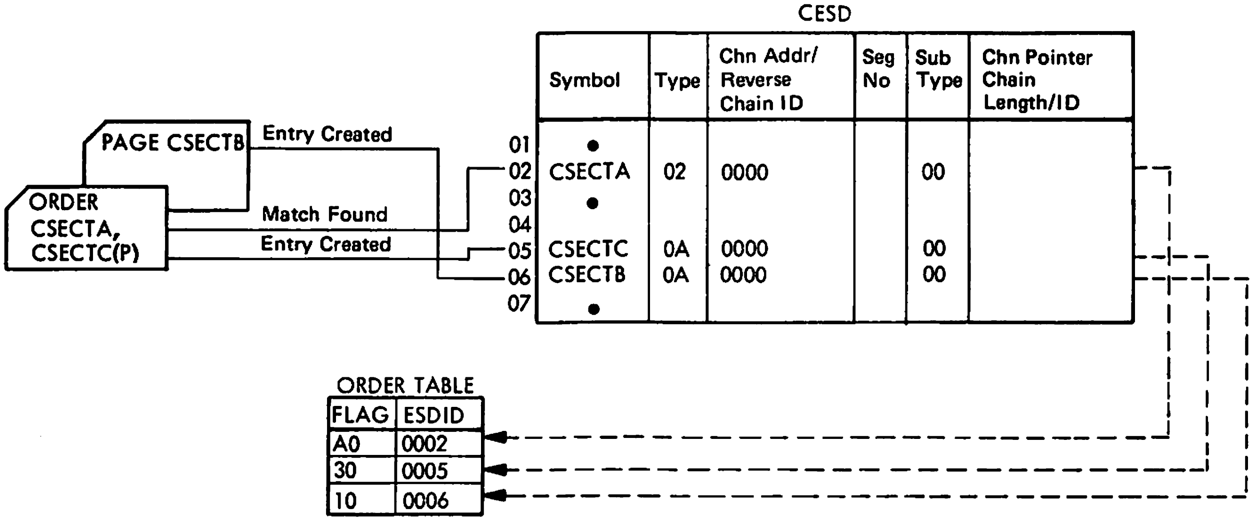
When a REPLACE statement or a CHANGE statement operand contains two symbols, such as CHANGE A (B), A and B are entered in consecutive lines of the CESD. Only the first line of the pair (the line for A) contains the address (in the chain address field) of the next item in the replace/change chain.

NAME STATEMENT PROCESSOR: The NAME statement processor places an entry in the all-purpose table containing the name under which the output load module is to be stowed in the PDS directory. If the operand contains the level 1 symbol (R), a bit is set to indicate that the module is to be stowed as a replacement for a module of the same name. Another bit is set to indicate that a NAME statement was encountered; the input processor tests this indicator and terminates input operations if it is set. If a NAME statement is received from any input source other than SYSLIN, the error routine is entered; NAME statements are accepted only if they are in the primary input.

SETSSI STATEMENT PROCESSOR: The SETSSI statement processor converts the 8 bytes of hexadecimal information specified on a SETSSI statement to a 4-byte field, and enters it into the all-purpose table. During final processing, this information is entered into the system status index, a 4-byte extension of the user data area in the PDS directory. The index contains information describing the status of members in the library and is used for maintenance purposes.

ORDER AND PAGE STATEMENT PROCESSOR: The ORDER and PAGE statement processor builds the ORDER table. First, the CESD is searched for a match to the symbol specified in the ORDER or PAGE statement. If the symbol is not found in the CESD, the symbol is entered into the CESD as a "weak external" reference (WX). The ESD identifier of the CESD line is entered into the ORDER table. When a matching symbol is found in the CESD, and the entry is not a control type ER, the ID of the CESD line is entered into the ORDER table.

The appropriate flags are set in the ORDER table entry to indicate if the specified request is either an ORDER or a PAGE statement. The ORDER flags are set when text ordering during output processing is requested. The PAGE flags specify that the linkage editor is to perform page alignment during address assignment. The ORDER flags can be set only when an ORDER control statement is present. The PAGE flags can be set in one of two ways: (1) if P is specified on an ORDER control statement or (2) if the PAGE control statement is present. See Figure 11 for an example of order and page processing.



CESD

	Symbol	Type	Chn Addr/ Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chain Length/ID
01	•					
02	CSECTA	02	0000		00	
03	•					
04						
05	CSECTC	0A	0000		00	
06	CSECTB	0A	0000		00	
07	•					

ORDER TABLE

FLAG	ESDID
A0	0002
30	0005
10	0006

Figure 11. Order and Page Processing

Note: In this example, CSECTB must follow CSECTA and CSECTC in the order table. Ordering was not specified for CSECTB.

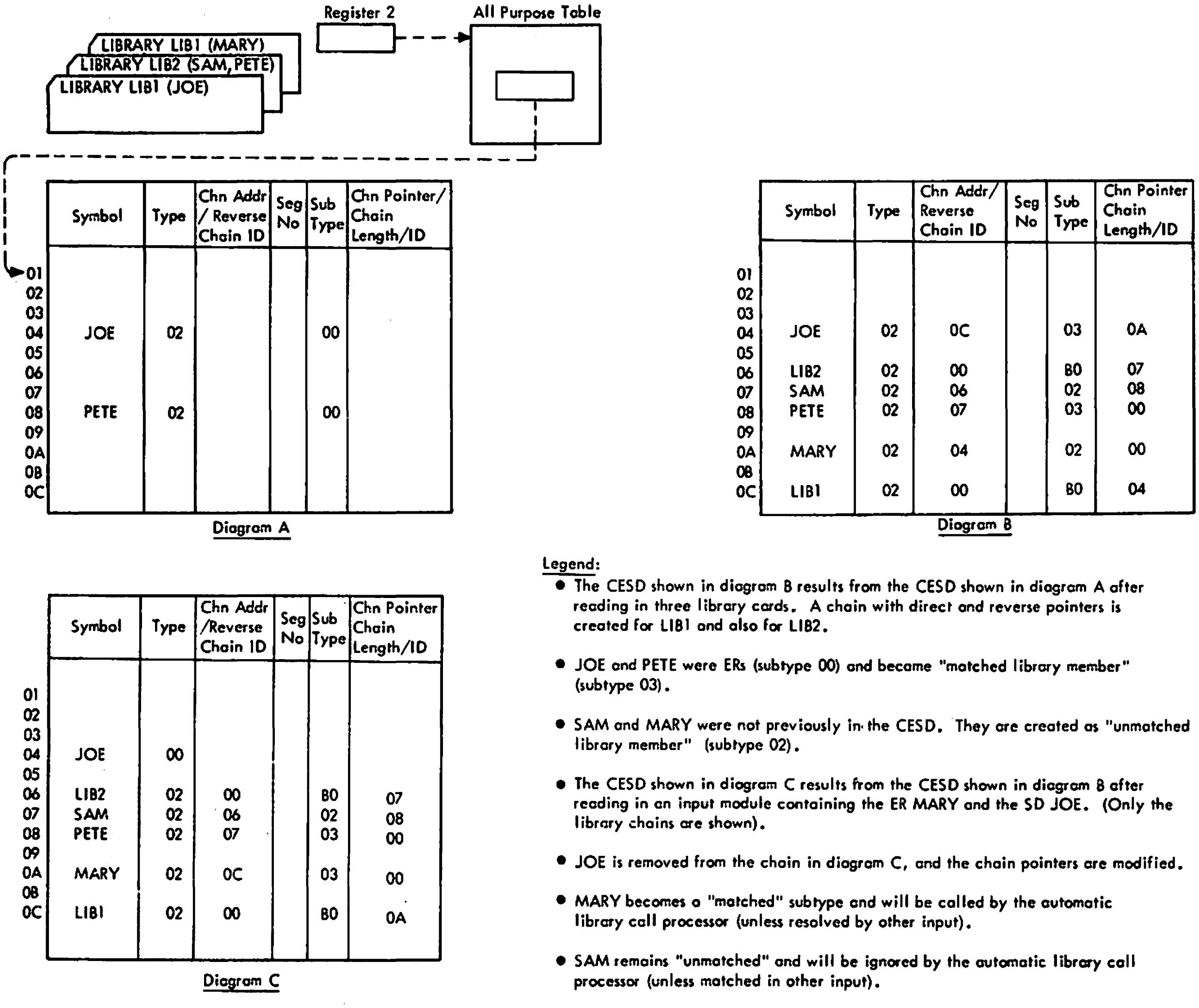
IDENTIFY STATEMENT PROCESSOR: The IDENTIFY statement processor picks up the CSECT name from OPDO, the length of the special string SPECSTR extracted by the READ8 routine, and the identify data placed in SPECSTR by the READ8 routine. This information is placed in storage, and parameters and control are passed to the IDR processor HEWLMIDR at the entry point HEWLCIDR.

ENTRY STATEMENT PROCESSOR: The ENTRY statement processor places the symbol specified in an ENTRY statement in the all-purpose table. The symbol will override any symbol specified in an END statement as the entry point for the module.

ALIAS STATEMENT PROCESSOR: The ALIAS statement processor creates chained CESD entries for a maximum of 16 alias names specified in ALIAS statements. During address assignment, these entries are used to build the alias table.

LIBRARY STATEMENT PROCESSOR: The LIBRARY statement processor creates chained CESD entries for the operands specified in LIBRARY statements; a chain is created for each distinct library. Each chain begins with a library ddname and contains all member names specified for the library (see Figure 12).

A member name specified in a LIBRARY statement can result in one of two kinds of ER subtypes: "matched library member" or "unmatched library member." If a CESD entry is created for a member name specified in an input ER and also specified in a LIBRARY statement, it is called a "matched library member." However, if the member name was specified only in a LIBRARY statement, the entry subtype is "unmatched library member."



	Symbol	Type	Chn Addr / Reverse Chain ID	Seg No	Sub Type	Chn Pointer/ Chain Length/ID
01						
02						
03						
04	JOE	02			00	
05						
06						
07						
08	PETE	02			00	
09						
0A						
0B						
0C						

Diagram A

	Symbol	Type	Chn Addr/ Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chain Length/ID
01						
02						
03						
04	JOE	02	0C		03	0A
05						
06	LIB2	02	00		80	07
07	SAM	02	06		02	08
08	PETE	02	07		03	00
09						
0A	MARY	02	04		02	00
0B						
0C	LIB1	02	00		80	04

Diagram B

	Symbol	Type	Chn Addr /Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chain Length/ID
01						
02						
03						
04	JOE	00				
05						
06	LIB2	02	00		80	07
07	SAM	02	06		02	08
08	PETE	02	07		03	00
09						
0A	MARY	02	0C		03	00
0B						
0C	LIB1	02	00		80	0A

Diagram C

Legend:

- The CESD shown in diagram B results from the CESD shown in diagram A after reading in three library cards. A chain with direct and reverse pointers is created for LIB1 and also for LIB2.
- JOE and PETE were ERs (subtype 00) and became "matched library member" (subtype 03).
- SAM and MARY were not previously in the CESD. They are created as "unmatched library member" (subtype 02).
- The CESD shown in diagram C results from the CESD shown in diagram B after reading in an input module containing the ER MARY and the SD JOE. (Only the library chains are shown).
- JOE is removed from the chain in diagram C, and the chain pointers are modified.
- MARY becomes a "matched" subtype and will be called by the automatic library call processor (unless resolved by other input).
- SAM remains "unmatched" and will be ignored by the automatic library call processor (unless matched in other input).

Figure 12. Library Statement Processing

EXPAND STATEMENT PROCESSOR: The EXPAND statement processor accumulates the expansion length specified and, if necessary, limits that length to 4095 bytes. If the name specified matches the name of a named control section or common section, the length of that control section or common section is updated by the expansion length in the matching CESD entry. A special entry is then made to the text processor to create and save text of the expansion length to be added to the control section or common section.

MODE STATEMENT PROCESSOR: The MODE statement processor verifies that the valid mode keywords, AMODE and RMODE, are specified and that the valid mode specifications are used—24, 31, or ANY for AMODE; 24 or ANY for RMODE. Appropriate values are set in the all-purpose table to indicate the mode(s) specified.

SETCODE STATEMENT PROCESSOR: The SETCODE statement processor accumulates the authorization code specified. A limit of 8 digits specifying a value of 0 to 255 is imposed. Once verified, the authorization code is saved in the all-purpose table.

Object Module Processing

If input to be read by the linkage editor consists of object modules (fixed (F) record format indicates object modules), the following operations are performed:

- Determine record type
- Set up general registers
- Perform special event processing

The record type is determined by examining columns 2 through 4 of each logical input record. For each record type (SYM, ESD, TXT, RLD, IDR, END), special processing is required.

The general registers are loaded with input record information to be used in the required processing, as described in Figure 13.

Input Record Type	3	4	5	6
SYM		SYM record byte count		Address of SYM record in buffer
ESD		Number of bytes of ESD information	ESD ID of first ESD item on record	Address of first byte of ESD in buffer
TXT	Assigned address of first byte of text	Number of bytes of text information	ESD ID of CSECT to which text belongs	Address of first byte of text in buffer
RLD		Number of bytes of RLD information		Address of first byte of RLD in buffer
END	Absolute address of entry point on END record	Length of CSECT for which no length was given in ESD item	ESD ID CSECT containing entry point	

Figure 13. General Register Information—Object Module Processing

Following is a description of special event processing:

- When end-of-input is detected, any data still contained in the input RLD buffer or the input text buffer is written out on SYSUT1, if necessary.

  See the Appendix, "Input Convention and Record Formats."

- If the TEST attribute is selected, the SYM records from the object module are blocked 3-to-1 in the input RLD buffer and written out on SYSLMOD. When the first TXT record in a module is encountered (or, if no text record has been encountered when the END record is detected), remaining SYM records in the input RLD buffer are written out on SYSLMOD.

- When processing of an ESD record is completed, indicators in the all-purpose table are examined to determine if:

  - A control section (SD, PC, or common) was indicated on the ESD record

  - The TEST attribute was specified

  If both conditions are met, the ESD record is blocked 3-to-1 in the input RLD buffer and written out on SYSLMOD.

- If a control statement continuation is expected and an object module record is read, an error condition occurs and a coded diagnostic message is produced. Normal object module processing is then performed on the record.

- If, during object module processing, a record is encountered that is not one of the six acceptable types (SYM, ESD, TXT, RLD, IDR, or END), an error condition occurs and a diagnostic message is produced. The input record is then ignored.

Load Module Processing

Load modules included as input to the linkage editor are processed in the following manner:

- The input record type is determined by an identification field (byte 1 of the record), as shown in Figure 14 on page 33. Special processing is performed for each record type.

- The parameter registers are loaded with input record information to be used in the required processing, as described in Figure 15 on page 33.

- If the record is not identified as a TXT, CESD, IDR, scatter/translation, SYM, or CTL/RLD record, an error condition occurs and a diagnostic message is printed out. The input record is otherwise ignored.

- If the TEST attribute was not specified, all SYM records are ignored.

- If an end-of-module indication is found in a CTL or RLD record, cleanup functions are performed.

- When a CTL record is detected, the following TXT record is immediately read into the input text buffer if it is not to be deleted.

- If the TEST attribute was specified and a SYM record is received, the record is written out as text translation data from the RLD input buffer.

Record Type	Identifier (in Hexadecimal)
TXT	Identified by preceding control record
CESD	'20'
IDR	'80'
Scatter/Translation	'10'
SYM	'40'
CTL	'01'
CTL/RLD	'03
RLD	'02'

If end-of-segment indicator is on:

CTL	'05'
CTL/RLD	'07'
RLD	'06'

If end-of-module indicator is on:

CTL	'0D'
CTL/RLD	'0F'
RLD	'0E'

Figure 14. Input Record Types—Load Module

Load Module Record Type	3	4	5	6
SYM		Zero		
CESD		Byte count of ESD items in record	ESD ID of first CESD item on record	Address of first byte of CESD item in buffer
CTL (TXT)	Assigned address of first byte of text	Number of entries in ID-length list	ESD ID CSECT to which text belongs	
RLD		Byte count of RLD items in record		Address of first RLD item in buffer

Figure 15. General Register Information—Load Module Processing

The following describes the special processing performed during object and load module processing for the ESD, IDR, TXT, RLD, and END records.

ESD Record Types

Every object module in the input to the linkage editor must contain at least one ESD item. An ESD item is created by a language translator whenever it finds a symbol that is defined for external use. An ESD item is created to define the beginning of each control section, common areas, entry point names, and external references. Each ESD item has a type assigned to it that indicates its function. The ESD types are:

- Section Definition (SD). Defines the beginning of a named control section.
- Private Code (PC). Defines the beginning of an unnamed control section.
- Label Definition (LD). Defines a label (symbol) whose location is defined relative to the location of the control section in which it is contained. An LD type ESD item contains the ESD ID of the control section that contains the label.
- Common (CM). Defines a common area for which a virtual storage address is assigned during linkage editor processing. The area may be named or unnamed; an unnamed area is referred to as a "blank common" area.
- Pseudo Register (PR). Defines an area external to the output module but referred to by it, for which virtual storage space is allocated at execution time. The linkage editor treats PR symbols as a block that is external to the program. The value assigned to each symbol is a displacement within this block.
- External Reference (ER). Specifies a symbol that is referenced but not defined within an input module.
- Weak External Reference (WX). Specifies an external reference that is not to be resolved by automatic library call. A WX entry is processed as an ER entry with a "weak call" flag.

| CESD Record Types

A load module in the input to the linkage editor contains at least one CESD record unless the module is marked not editable (NE). The CESD record types are the same as for ESD records, with the following additions:

- Null Type. This indicates that the item is to be ignored in any reprocessing of the module by the linkage editor.
- Label Reference (LR). This defines a label (symbol) within a control section. An LR type CESD entry is numbered; it contains the ESD ID of the control section entry in the ID/length field. An LR may be referenced directly by an RLD item in the same module, whereas an LD may not. All LD items are changed to LR items during linkage editor processing (LDs are contained only in object modules, never in load modules).
- Private Code (PC) Marked Delete. This is a CESD item created only for ENTABs and SEGTABs. PC-delete entries are placed in the renumbering table, indicating that associated TXT and RLD information is to be deleted.

LY26-3963-0

ESD Processing

The main function of ESD processing is symbol resolution. Individual ESDs in the input to the linkage editor are combined into a composite ESD, which contains all symbols in the input that were not changed, deleted, or replaced. A chained replace/change list (produced by the control card scanner) specifies which ESD items are to be changed, deleted, or replaced. A renumbering table (RNT) is also produced during ESD processing; it is used during TXT, RLD, and END processing to translate the ESD IDs of the input ESD items to CESD IDs. "Diagram 9. ESD Processing" on page 83 provides a general illustration of several types of ESD processing.

At the beginning of ESD processing, control information from the ESD record is saved: the ESD ID of the first ESD item in the record (other than an LD); the number of bytes of ESD information; and the type field of the first ESD item.

If the OVLY option has been specified for the output load module, the following occur:

- The current segment number is placed in the ESD, unless the entry type is PR (PRs have an alignment value in the segment number field).
- The RMODE for the load module is forced to 24.
- If automatic library call processing is being performed, the segment number is forced to 1 (all automatically called modules are placed in the root segment of the overlay structure).

If the OVLY option has not been specified for the output load module:

- If the current ESD item is from a load module in overlay format, the AMODE/RMODE data is forced to 24/24.
- Otherwise, the content of byte 12 is interpreted as AMODE/RMODE data.

The ESD item is then processed according to its type, in the following manner:

- If the ESD item is an ER, bytes 10, 11, and 12 are set to zero in the input buffer (either the object module buffer, the SYSLIN buffer, or the first pass RLD input buffer). Byte 10 must be cleared because automatic library call processing uses it to indicate whether automatic library calls have been processed. Bytes 11 and 12 must be cleared because any nonzero data (including blanks) will be entered in the delink table if delinking is required for the symbol. If the input item is an ER item from an object module, the CESD subtype field is also reset to zero to indicate that there are no modifiers in the subtype field.
- If a REPLACE/CHANGE function has been requested for the input module, the replace/change chain that was built in the CESD by the control statement scanner is examined and the appropriate modifications are made. For example, if the scanner received the statement CHANGE A(B), the CESD contains a line for A, marked as a change statement item in the subtype field; the next line contains the symbol B. The input ESD item symbol is changed from A to B during ESD processing.
- If the ESD item is a PC, the CESD is not searched because each PC entry is treated as a unique entry. The PC is placed in the next available CESD line and is processed in the same manner as an SD.

- If the ESD item is a null item, the renumber routine is entered. (This routine is described under "Nonresolution Processing.")
- If the ESD item is an LD, it is changed to an LR. The item is then processed as an LR. (There are some minor differences in processing LDs that have been changed to LRs; for this reason, an internal indicator is set when the type is changed to LR.)
- If the ESD item is a PC or SD, the AMODE/RMODE data is checked for a valid combination. If an invalid combination is found, an error message is issued and the RMODE for the output load module is forced to 24.

After the ESD type is determined, the CESD is scanned for a matching symbol. If no match is found, nonresolution processing is performed. If the input ESD symbol matches a symbol in the CESD, resolution processing is performed. Resolution processing results in only one CESD entry for each unique input ESD symbol; multiple occurrences of the same input ESD symbol are listed in the renumbering table (RNT), with pointers to the single CESD entry.

NONRESOLUTION PROCESSING: If no matching symbol is found in the CESD, the input ESD item is processed as described below.

<u>SD Items</u>: If the input ESD item is an SD (see "Diagram 9. ESD Processing" on page 83, area A):

- The freeline routine selects an empty line in the CESD. The line following the current line is chosen unless a previous CESD line is marked null. (Whenever possible, null lines are used to save space.)
- If automatic library calls are being processed, an indicator is set in the type field of the selected CESD line. (If a module map was requested, this indicator is checked during module map processing. If the indicator is set, the control section is marked with an asterisk in the module map or cross-reference table to indicate that it was obtained from a library during automatic library call processing.)
- If the load module is in overlay structure, then those routines brought into the load module via the automatic library call are placed in the root segment of the load module.
- A "write" indicator is set in the all-purpose table to note that SDs, PCs, or CMs were encountered in the input record. When ESD processing is completed, the write indicator is tested. If it is on and the TEST attribute was specified, ESD records containing SDs, PCs, or CMs are saved, blocked 3-to-1 in the input RLD buffer, and written out on SYSLMOD.
- In any input object module, the CESD line number of the first SD entry whose length is zero is saved. END processing uses this CESD line number to enter the length specified on the END card.
- The enter routine creates a CESD entry for the input ESD item; it moves the symbol length, segment number, ID, and type into the selected CESD line. In addition, the enter routine accumulates the residence mode for the output load module. Initially, the residence mode is ANY. As each control section (SD or PC) is allocated in the output load module, its residence mode is included in the accumulation. If all control sections allocated in the output load module have a residence mode of ANY, the output load module has a residence mode of ANY; if any control section allocated in the output load module has a residence mode of 24, the output load module has a residence mode of 24. (The residence mode accumulated from the ESD data may be

overridden by the residence mode specifications in the PARM field or the MODE control statement.)

- The renumber routine places the line number of the new CESD entry into the renumbering table to provide a means of translating the input IDs to the new CESD IDs. For example, if the input ESD item has a line number (ESD ID) of 3, but the item is placed into the CESD at line 5, a 5 is placed in the third line of the renumbering table. (For each input ESD line, except LD lines, there is a corresponding RNT line. The RNT contains information for the current module; it is set to zero at the end of each input module.)

<u>ER Items</u>: If the input ESD item is an ER, it is entered in the CESD and renumbered as described above; no special processing is required.

<u>WX Items</u>: If the input ESD item is a WX, it is entered in the CESD and renumbered as described above; no special processing is required.

<u>CM Items</u>: If the input ESD item is CM (see "Diagram 9. ESD Processing" on page 83, area E), a "common" indicator is set and the item is treated as a delete item. If the address that was assigned to the CM item by the language translator is not zero, it is saved in the delink table for later use. (Two CM items with the same identifying symbol may have different assigned addresses; therefore, the assigned address in the input must be subtracted from all address constants that refer to the CM items, so that they are returned to their displacement value before relocation.) The CM item is then renumbered and entered into the CESD.

<u>LR (or LD) Items</u>: If the input ESD item is an LR or LD (see "Diagram 9. ESD Processing" on page 83, area C):

- When processing an LR, the label routine determines whether the SD for the control section has been processed. If the SD has not been received, any LRs that refer to that SD are chained together in the CESD until the SD is received. (The SD might be marked replace; therefore, the LR cannot be processed until the SD is received.) When the SD is received, all dependent LRs are processed. Each LR ID field is renumbered, using the renumbering table, so that it refers to the CESD ID of the SD.

- LDs are not renumbered because they are not referred to by RLDs and are not numbered in language translator output. The enter routine places them directly in the CESD. If an LD is received before the SD to which it belongs, it is handled as an LR.

<u>PR Items</u>: If the input ESD item is a pseudo register, the current segment number is not entered in column 12 of the ESD item. Column 12 of a PR item contains an alignment value, which indicates that the PR must be aligned to a halfword, fullword, or doubleword boundary. The PR is then processed by the freeline, enter, and renumber routines as described above.

RESOLUTION PROCESSING: If a matching symbol is found in the CESD, the type fields of the input item and the matching CESD item are compared and resolution processing is then performed. The following conventions are observed during resolution processing:

- Input PR items may match only PR entries in the CESD. If an input PR item matches a non-PR item in the CESD, it is not treated as a match; the CESD search for a matching PR item continues.

- If the matching CESD item is marked "chained," resolution is performed on the item to which it is chained.

- If the CESD line is marked null, the match is ignored and the search continues.

- If the CESD item is an ER produced from a REPLACE, CHANGE, OVERLAY, or ALIAS statement, or from the ddname field of an INCLUDE or LIBRARY statement, the match is ignored and the search continues.

Matching items are processed in the following manner:

- If the input ESD item is CM, SD, or LR and it matches an ER in the CESD, the input type replaces the type indicated in the CESD item (see "Diagram 9. ESD Processing" on page 83, area B). Nonresolution processing is then performed on the input item.

- If the input ESD item is an LR and it matches a CM, SD, or LR in the CESD, a "match" bit is set indicating that a double symbol definition is possible. If the SD for the control section has been entered in the CESD and is marked for deletion, the Label routine deletes the label; if it is not marked for deletion, a "double symbol definition" message is produced. If the SD for the control section is not in the CESD, the LR is chained to the matching LR; when the SD is received, the LR is deleted or a double symbol definition message is produced, depending on whether or not the SD is being deleted.

- If an input PR matches a PR in the CESD ("Diagram 9. ESD Processing" on page 83, area D), the greater length and the most "constrictive" boundary alignment are placed in the CESD entry. (A doubleword alignment is more constrictive than fullword alignment; fullword is more constrictive than halfword; etc.) The input PR entry is then renumbered to the updated PR entry in the CESD.

- If an input SD item matches an SD entry in the CESD, automatic replacement of the control section occurs. The input SD item is entered in the CESD as a delete type and is chained to the matching SD entry. (During second pass processing, the assigned address of the control section being replaced will be subtracted ("delinked") from the addresses of any nonbranch-type address constants that refer to the SD-delete entry.) The SD-delete item remains chained only while the module is being processed; END processing will change the chained items to null entries (see "Delinking Nonbranch-Type Address Constants").

- If an input SD item matches a CM entry in the CESD, and the length of the SD item is greater than or equal to the length of the CM item, the length of the SD item is entered in the CESD. If the program is in overlay, the common path routine scans the <u>segment path table</u> (SEGTAl) to find the segment in the overlay structure that is common to both items, and places the segment number in the SD entry. The SD item is then written over the CM line and renumbered. (This is referred to as "automatic promotion of common.")

- If an input SD or CM item matches an LR in the CESD, a "double symbol definition" message is produced, and the SD or CM item is entered in the CESD as a delete type and is chained to the matching LR entry, causing the SD or CM to be replaced.

- If the input item is CM, it may be "blank common." Blank common may match a PC item in the CESD because both contain blanks in the symbol field. In such a case, the match is ignored and the search continues.

- If an input CM item matches a CM item in the CESD ("Diagram 9. ESD Processing" on page 83, area F), the greater of the two lengths is entered in the CESD. If the module is being processed for overlay, the segment number of the segment common to both the input item and the CESD item is also entered in the CESD item (automatic promotion of common).

- If an input CM item matches an SD item in the CESD, and the length of the SD item is greater than or equal to the length of the CM item, the length of the SD item is entered in the CESD. The CESD type is not changed. If the module is being processed for overlay, the segment number of the segment common to both the input item and the CESD item is also entered in the CESD item (automatic promotion of common).

- Whenever an input ER item matches an ER in the CESD, both the type and subtype fields are examined; the ER items are then resolved in the following manner:

  - If the subtype fields of both ER items are not marked, the input item is not entered in the CESD; the matching ER remains in the CESD and a pointer to it is placed in the renumbering table entry for the input item.

  - If both items are marked "delete," the new ER is entered in the CESD and the old item remains there so that they can be delinked individually (in this case, the CESD may contain two ER items for the same symbol). Delinking is described in "Second Pass Processing."

  - If the input ER item is marked for deletion but the ER item in the CESD is not marked delete, the input ER is chained to the matching ER in the CESD. The chained ER item remains in the CESD until end-of-module is detected, so that the delink value can be saved.

  - If the input ER item is not marked for deletion and the ER item in the CESD is marked "delete" or "replace," the delete bit in the subtype field is cleared (delete is changed to replace) and the item is renumbered. If the matching ER item in the CESD is marked "no call" or "library member," it is marked "matched" before renumbering.

  - If the input ER item is marked in the subtype field but is not marked "delete" or "replace," it is assumed to be "never call"; if the matching ER item in the CESD is "library member," the CESD item is removed from the chain of library members and the input ER item is entered in the CESD and renumbered.

- If an input WX matches a WX in the CESD, no change is made to the CESD. If the matching entry in the CESD is not a WX or a control card entry, the input WX is changed to an ER.

- If an input ESD item that is not a WX matches a WX in the CESD, the CESD item is changed to an ER. In all cases, processing continues normally.

IDR Processing

The manner in which CSECT identification records (IDR) are processed depends on the type of IDR input records or control statements being processed. The input records or control statements are:

- Object module END records

- Load module IDRs

- IDENTIFY control statements

An object module END record contains only translation data; however, load module IDRs may contain four different types of IDR data:

- HMASPZAP-supplied data

- Linkage editor data
- Translator-supplied data
- User-supplied data

Load module IDR processing is dependent upon the type of data present in the IDR record. The IDENTIFY control statement contains only user-supplied data.

Before any IDR data processing begins, the type of IDR input is determined either by the input processor HEWLFINP or, if the data is from the IDENTIFY control statement, by the control statement processor HEWLFSCN. HEWLFINP passes control to HEWLFIDR at the entry point HEWLFIDR. HEWLFSCN passes control to HEWLFIDR at the entry point HEWLCIDR.

Processing Object Module END Records Containing IDR Data

When byte 33 of an object module END record has an EBCDIC 1 in it, one IDR item follows in bytes 34 through 52. If an EBCDIC 2 appears in byte 33, two IDR items follow in bytes 34 through 71. A blank in byte 33 indicates that the record contains no IDR data. IDR data is present only on an object module END record if the translator that produced the object module contains IDR support.

The renumbering table is scanned to determine the correct ESD identifiers of the CSECTs to which the translator data applies. If any of the CSECTs are marked delete in the renumbering table, they are not identified in the IDR output. If the input object module contains at least one CSECT that is not marked delete, the translator data is removed from the END record and placed in the IDR translator data table (IDRTRTAB) and the IDR translator ID table (IDRTITAB). These two tables contain the ESD identifier of the CSECT to which the translator data applies, the translator identification, the version and modification level of the translator, and the date of translation.

A comparison is made with the other entries in the IDRTRTAB for a duplicate entry. If a duplicate entry is found, the incoming data is combined with that of the previous entry to avoid repetition of data.

Processing Load Module IDRs

The subtype of the load module IDR is scanned for the type of IDR data. If the subtype is 02, the data is from the linkage editor; these records are ignored by IDR input processing. When the input data is from HMASPZAP (subtype 01), bits 2 through 7 of the flags and count field are scanned to determine the number of HMASPZAP entries in the record (from 1 to 19 entries are possible).

The entry in the renumbering table corresponding to the ESD identifier of the CSECT processed by HMASPZAP is examined. If the entry in the renumbering table is marked delete, then the IDR data associated with that CSECT is deleted. However, the data that is not deleted is placed at the end of the IDR HMASPZAP data table (IDRZPTAB). IDRZPTAB contains the ESD identifier of the CSECT processed by HMASPZAP, the date of the HMASPZAP processing, the data specified during HMASPZAP processing (this may be a PTF number or up to 8 bytes of variable user data specified on an HMASPZAP control statement). If the IDRZPTAB overflows, an error message is written and processing is terminated.

When the input data is translator-supplied data (subtype 04), the renumbering table entry corresponding to each ESD identifier in the string preceding a translator description is examined. If the entry is marked delete, the corresponding ESD identifier is deleted from the string; otherwise, the input ESD identifier is replaced by the renumbered identifier. If at least one ESD identifier remains on the string, a check is made among the table entries in the IDR translator data table (IDRTRTAB) to see whether an identical description has already been entered into IDRTRTAB. If an identical description does exist in IDRTRTAB, the CSECTs associated with the incoming translator description are combined with the existing translator data item to form a single table entry in order to avoid needless repetition of data. If it does not exist, a new entry is added for the input data.

When the input data is user data (subtype 08), the renumbering table entry corresponding to the ESD identifier of each input user data item is examined. If it is marked delete, the user data is ignored. If not, the input ESD identifier is replaced by the renumbered identifier and the user data is entered at the end of the IDR user data table (IDRUDTAB). The IDRUDTAB entries contain the ESD identifier of the CSECT to which the user data applies, the date the data was supplied to the module via the linkage editor IDENTIFY function, the number of characters in the user data field, and the user data.

In the case of input load module IDRs containing translator or user-supplied data, an individual data item may span more than one record. When this occurs, the incomplete portion of the item is saved in either IDRTRTAB or IDRUDTAB. The item is processed after the next input record has been read, and the continued portion of the item is combined with the saved portion to form a complete data item.

Processing IDENTIFY Control Statement Data

The control statement processor, HEWLFSCN, passes control to the IDR processor at the entry point, HEWLCIDR. The CESD is searched for an SD type entry matching the CSECT name to be identified. If the name is not an SD, an error is logged and processing is terminated. If the CESD line is an SD marked delete, the data is ignored. If the CESD line is an SD not marked delete, the ESD identifier of the matched SD name is saved. A check is made to see whether there was any user-supplied data previously associated with the CSECT. If there was, the old data is replaced with the new incoming data. If no earlier data exists, the incoming data is added to the end of the table IDRUDTAB.

TXT Processing

The manner in which TXT records are processed depends on whether they are part of either a load module or an object module, or are added using the EXPAND control statement. A load module contains records in a specified order. However, in an object module, the records may not be in the proper sequence because the language translator may have created them out of order (EXPAND data is always identified as out of order text). (The restrictions on linkage editor input are described in Appendix, "Input Conventions and Record Formats.") "Diagram 10. Processing Object Module Text" on page 84 and "Diagram 11. Processing Load Module Text Records" on page 85 illustrate processing of TXT records from object and load modules, respectively.

Before any address constants can be relocated within a control section of an object module, all TXT records must be placed in the proper order. This is done in the <u>input text buffer</u> (TXTBFBEG), which is variable in length, allowing grouping of data within the buffer.

Each "multiplicity" of text is assigned a number as it is moved (or read) into TXTBFBEG. A multiplicity is a portion of text equal in length to the maximum size of a SYSLMOD output record. Within each control section, multiplicity numbers are assigned consecutively, starting at 0.

Text records from object modules contain both text data and the control information needed for processing. Text records from load modules contain only text, so the associated control record must also be examined to obtain the required control information. During object module processing, control information is placed in registers; this information allows the object module text to be moved from the object module buffer into TXTBFBEG. For load module text, the assigned address of the first byte of text and a pointer to the ID-length list (in the control record) is determined during load module processing. This information allows the text record to be read directly into TXTBFBEG.

Processing Object Module Text

When text is received from an object module, the text record ID is renumbered using the renumbering table, so that it refers to the CESD entry for the control section that contains the text. The size of the control section is obtained from the CESD, and a test is made to determine if the whole control section or a multiplicity (whichever is smaller) will fit into the space available in TXTBFBEG. If the control section length was not specified in the CESD entry, only text for the current ID is accepted; see "No-length Control Section" below.

If there is sufficient space in TXTBFBEG to accommodate the tabletext I/O table control section or multiplicity, the text is moved into the buffer and an entry (containing the ID and multiplicity number of the text) is made in the text I/O table. A corresponding entry containing the location of the multiplicity and the length of the text is made in the text note list. The text note list entry also contains a displacement field. When text is in order, or on the first occurrence of text for a multiplicity, the displacement field is set to 0; for out-of-order text, the displacement field contains the displacement from the beginning of the multiplicity of the first byte of contiguous text.

If the SYSUT1 record size is smaller than the multiplicity size, each multiplicity is divided into pieces, each piece having a length equal to the SYSUT1 record size. New text I/O table and text note list entries are made for each piece; the displacement field will contain the displacement of each piece from the beginning of the multiplicity.

NO-LENGTH CONTROL SECTION: When text is received for a no-length control section (a control section for which no length is specified in its CESD item), space for one multiplicity is allocated in TXTBFBEG. Entries are made in the text I/O table and the text note list for the multiplicity, and the text is moved into TXTBFBEG. This procedure is repeated for each subsequent multiplicity of text for the no-length control section. If TXTBFBEG becomes full, its contents are written on SYSUT1 as described in "Writing Text on SYSUT1." When the length is received, it is entered in the text note list.

PROCESSING OUT-OF-ORDER TEXT: A load module contains records in a definite order. However, records in an object module may not be in the proper sequence because the language translator may have created them out of order (records resulting from the EXPAND control statement are marked out of order). Such records may contain discontinuities in addresses (because of a reorigin or a disjointed control section), or they may not be contiguous (that is, text of a given ID and multiplicity may be interspersed with text of other IDs or multiplicities). Records of contiguous text must be built on SYSUT1, so that during second pass processing the text can be placed into its proper position, within its ID and multiplicity, in the second pass text buffer.

The first occurrence of a given ID and multiplicity is read into the input text buffer as it is received. Discontinuities and noncontiguous text are of no consequence at the first occurrence of an ID and multiplicity. However, once text of a given ID and multiplicity has been written out on SYSUT1, any subsequent text of that ID and multiplicity must be contiguous to be written out on SYSUT1 within each text record.

Text of a previously written ID and multiplicity is read into the input text buffer until a discontinuity, or text of a different ID or multiplicity, is encountered. The contiguous text in the buffer is then written out on SYSUT1. The discontinuous (or noncontiguous) text is then placed in the buffer. If this text represents the first occurrence of an ID and multiplicity, the buffer is loaded without regard for discontinuities or noncontiguous text. If the text belongs to a previously written ID and multiplicity, the text processor will again place only continuous text of that ID and multiplicity in the buffer.

A record that contains noncontiguous text is called a loose record; a record that contains contiguous text is called dense. The text note list entry for a dense record usually has a nonzero value in the displacement field. When the text is read back from SYSUT1 into the second pass text buffer during second pass processing, this displacement is used to place the text in its proper position within its ID and multiplicity.

Processing Load Module Text

Because text records from load modules are ordered and well-defined, they require little further processing by the text processor. The information in the ID-length list (in the control record) is scanned, and each ID is renumbered and checked to determine whether it is to be deleted. If all IDs are to be deleted, the record is ignored and control is returned to the input processor.

When an ID that is to be processed is found, the text record containing the ID must be read into TXTBFBEG. The text record length is obtained from the associated control record and compared against the free space available in TXTBFBEG. If sufficient space is available, the text record is read into the buffer; otherwise, the contents of the buffer are written on SYSUT1 to ensure sufficient space, and the record is read.

Text is processed in the buffer in the order specified by the ID-length list. IDs that are to be deleted are overlaid by IDs that are to be processed. The text is divided into multiplicities, and entries are made in the text I/O table and the text note list. When all text identified by the ID-length list is processed, text processing is completed.

Writing Text on SYSUT1

When no more control sections can be accommodated in TXTBFBEG, the contents of the buffer must be written on the intermediate data set (SYSUT1). The text I/O table is scanned to determine the order in which control sections are to be written. The length of the first control section (that is, corresponding to the first text I/O table entry) is obtained from its corresponding ESD ID; if the length is less than the size of the SYSUT1 record, the text I/O table entry for the control section is marked "written." Each subsequent control section is similarly processed, and its length added to the sum of the lengths of previously processed control sections.

When the sum of control section lengths reach the limit of a SYSUT1 record, the entire group of control sections is written on SYSUT1. The relative track address (TTR) is placed in the text note list entry corresponding to the last text I/O table entry that was processed.

When a single control section is larger than a SYSUT1 record, the multiplicities of the control section are grouped up to the limit of the SYSUT1 record size, and written.[6] When control sections or multiplicities are grouped on SYSUT1, the multiplicities must be in ascending, consecutive order. If the overlay attribute has been specified, no grouped control sections are permitted on SYSUT1.

Note: Each time an entry is made in the text note list during text processing, a check is made to determine whether the list is full. If it is full, an error message (IEW0364) will be issued.

If neither TXTBFBEG nor the text note list becomes full during text processing, no text is written on SYSUT1. The text is retained in the buffer, and single pass processing is in effect for text records.

RLD Processing

RLD processing basically consists of:

- Updating each set of relocation and position pointers (R and P Pointers)
- Processing each flag and address (FA) in the input item until the end of the record or the next item with an R and P pointer is detected

RLD records from object modules and load modules are processed in the same manner.

RLD information is grouped in the RLD buffer by P pointer. Each P pointer of an input RLD record refers to the ESD entry in the input module for the control section that contains the address constant. Each time a new P pointer (one referring to a different ESD ID) is detected, an entry is made in the RLD note list for the RLD set (a set being an unbroken sequence of RLD items having the same P pointer). The RLD note list entry contains the following information for each set:

- The renumbered P pointer to which these RLDs refer.
- The lowest multiplicity of text to which these RLDs refer.
- The number of bytes of RLDs.
- The storage address of the first byte of RLD data if all RLDs remain in virtual storage. If RLDs are written on SYSUT1, this field contains the accumulated byte count for intermediate chains or the TTR of the record on SYSUT1.

All adjacent RLD items containing the same P pointer are referred to by only one RLD note list entry. Adjacent RLD items containing the same R and P pointers are chained, with the R and P pointers appearing only once, at the beginning of the chain. The remaining RLDs in the chain are compressed by setting the flag indicating continuation, and discarding the 4 bytes containing the R and P pointers.

Each R pointer of an input RLD record refers to the ESD entry in the input module upon whose value the address constant depends. The R and P pointers are updated using the renumbering table. Before renumbering, the R and P pointers refer to ESD entries of the input module that contains the RLD items. The pointers are renumbered so that they point to the proper entries in the CESD being created for the output load module. If the R pointer refers to a deleted ESD entry, delinking may be performed. If the assigned address of the symbol referred to by the address constant is zero, the address constant is not delinked. (Normal

[6] If the SYSUT1 record size is smaller than the SYSLMOD record size, no grouping is permitted.

relocation is performed.) When delinking is necessary, an entry is placed in the delink table (a function of ESD processing). The delink table entry contains the address (delink value) of the symbol being deleted, and the CESD entry number of the identically named symbol that is to replace the deleted symbol.

The ID of the delink table entry for the deleted symbol is saved in the renumbering table, and a "delink value saved" indicator is set. The ID of the identically named symbol and the ID of the new delink table entry are saved because they are later used to complete the delinking operation. The R pointer of the RLD item must be modified to refer to the delink table entry for the deleted symbol, but the original R pointer is needed to process any V-type address constants referred to in the RDL item. (V-type address constants do not require delinking, but may be in a FA string with A-type address constants that do require delinking.) Therefore, the R pointer is not modified until the string of flag-address (FA) fields following the R and P pointers has been processed as described below. At that time, if the module is to be structured for overlay and it contains V-type address constants that refer to the deleted symbol, the ID of the identically named symbol is inserted into the calls list.

Each FA field of the RLD record is processed as follows:

- The high-order bit of the flag field is set to zero.

- If the address constant is an A-type, the renumbering table entry referred to by the R pointer is checked to determine whether it is marked as a PR type. If it is a PR, the RLD flag field is also marked PR (because second pass processing must handle PRs in a special manner). If the renumbering table entry is not an ER or marked delete, the RLD flag field is marked for relative relocation. This indicates to second pass processing that the difference between the origin of the control section in the input and the origin assigned by the linkage editor is to be used as a relocation factor for the value of the address constant. If the RNT entry is an ER or marked delete, the RLD flag field is not marked. This indicates to second pass processing that the address constant is to be relocated by absolute relocation; second pass processing uses the linkage editor assigned address of the symbol in the output module as a relocation factor for the value of the address constant. (This procedure is described under "Second Pass Processing.")

- If the address constant is a 4-byte V-type ("branch-type") and the program is in overlay, an entry is placed in the calls list, provided that the address constant refers across control sections (R not equal to P). The calls list is used during address assignment processing to determine which segments require ENTABs, and the number of entries each ENTAB must contain.

- For both A-type and V-type address constants, the text multiplicity of the address field is determined and is saved in the RLD note list if it is lower than any previous multiplicity in the RLD record. If two pass processing is in effect, the RLD note list is used during second pass processing to read back RLD data from SYSUT1 (each RLD note list entry contains the relative track location (TTR) of an RLD record on SYSUT1). The second pass processor uses the multiplicity field of the RLD note list entry to determine whether the associated RLD record should be read back from SYSUT1 for a given multiplicity of text.

- When the last FA field in the string has been processed, all items in the string have been checked to determine whether they require delinking. If any A-type address constants in the string required delinking, the R pointer for the string is modified to refer to the associated delink table entry.

Figure 16 shows the actions performed during RLD processing for each input flag format, and the format of the flags after RLD processing. (The "output" column shows the flag formats that are passed as input to the Relocation routine of second pass processing; see Figure 27 on page 68.) After all FA fields have been processed, the next RLD record is processed.

If the RLD buffer becomes full, its contents must be written on the intermediate data set (SYSUT1). The RLD buffer is allocated with a maximum length less than or equal to the size of a SYSUT1 record, so the entire buffer may always be written. As many consecutive RLD sets as possible are grouped in a SYSUT1 record.

Input			Output	
Flag[1]	Type	Action Performed	Flag	Type
0000LIST	Not PR, ER, WX, CM, or delete	Marked for relative relocation	1000LIST	Relative
0000LIST	ER ('02' in renumbering table)	Marked for absolute relocation	0000LIST	Absolute
0000LIST	Delete or CM ('05')	Marked for absolute relocation if assigned address of input item is zero	0000LIST	Absolute
0000LIST	PR ('06')	Marked as PR (displacement value)	0010LIST	Pseudo Register Type 1
0000LIST	Delete or CM	Marked "delink value saved" if assigned address of input item is not zero	High-order bit of P pointer	Delink
0001LIST	Type is not checked	RLD is marked branch-type	0001LIST	Branch
0001LIST or 1001LIST[2]	Delete	Marked "delink value saved and other FA items in string exist that are nonbranch-type" and are being delinked	High-order bit of P pointer.	Delink
0010LIST	Pseudo Register Type 1	None. Remains as a PR (displacement value)	0010LIST	Pseudo Register Type 1
0011LIST	Type is not checked	Marked as PR (cumulative length)	0011LIST	Pseudo Register Type 2

Figure 16. RLD Flag Field Processing

Notes to Figure 16:

[1] Refer to "RLD Input Record (Card Image)" and "Relocation Dictionary Record (Load Module)" in Appendix, "Input Conventions and Record Formats."

[2] Internal types processed during second pass.

The RLD note list entry for each RLD set in the group contains a "grouped" indicator; the note list entry for the last RLD set in the group also contains the relative track address (TTR) of the group.

RLD sets whose lengths exceed that of a SYSUT1 record (requiring more than one output record) are not grouped. RLD note list entries for RLD sets that are not grouped contain the relative track location (TTR) of the SYSUT1 record and a "nongrouped" indicator.

Each time an entry is made in the RLD note list, a check is made to determine whether the list is full. If it is full, error message IEW0364 will be issued.

Note: If neither the RLD buffer nor the RLD note list becomes full during RLD processing, no RLDs are written on SYSUT1. The RLD information is retained in the RLD buffer, and single pass processing is in effect for RLDs.

END Processing

When an END record or the end of an input load module is detected, END processing is required. The functions of END processing include:

- Resetting tables (such as the renumbering table) that were involved in the processing of the input module
- Processing entry point information
- Deleting any CESD lines marked CHAIN or DELETE, and keeping track of deleted lines
- Entering in the CESD the length of a control section for which no length was specified in the ESD item (if the length is contained on the END record)
- Setting flags in the ORDER table for each entry matched by an entry in the CESD, and resetting the flag for formerly matched entries
- Placing the data from END records in object modules created by a translator that supports IDR into the IDR translator ID and data tables, IDRTRTAB and IDRTITAB

Include Processing

Include processing is required when:

- The control statement scanner has detected an INCLUDE statement and the INCLUDE statement processor has built an include chain
- End-of-input has been detected, and the "more includes" indicator in the all-purpose table (APT) is on

Include processing consists of preparatory functions (OPEN, BLDL, FIND) required before the module to be included can be read. These functions include:

- An input pointer to the library read block is set.
- The SYSLIB DCB is closed (unless it is open for a partitioned data set currently being used).
- Each entry in the include chain is examined sequentially.

SEQUENTIAL DATA SETS: If an include chain entry specifies a sequential data set, the data set organization field of the DCB is changed from partitioned to physical sequential, and the ddname field is updated. The DCB is then opened, and the module is read in.

PARTITIONED DATA SETS: If an include chain entry specifies a member of a partitioned data set the member name is entered into the BLDL list, and the next entry is examined. If the next entry specifies a different data set name, the partitioned data set is opened and a BLDL macro instruction is executed for the single member name.

If the next entry specifies another member of the same partitioned data set, the member name is added to the BLDL list and the next entry in the include chain is examined. Member names are added to the BLDL list until a different data set name is encountered, the BLDL list becomes full, or the end of the include chain is reached. Because the BLDL list must be in collating sequence, each member name is inserted into its proper position, moving other entries as necessary. Because included modules must be read in the order in which they appear in the INCLUDE statement (without regard to the collating sequence), a separate table indicating the order of processing BLDL list entries is maintained.

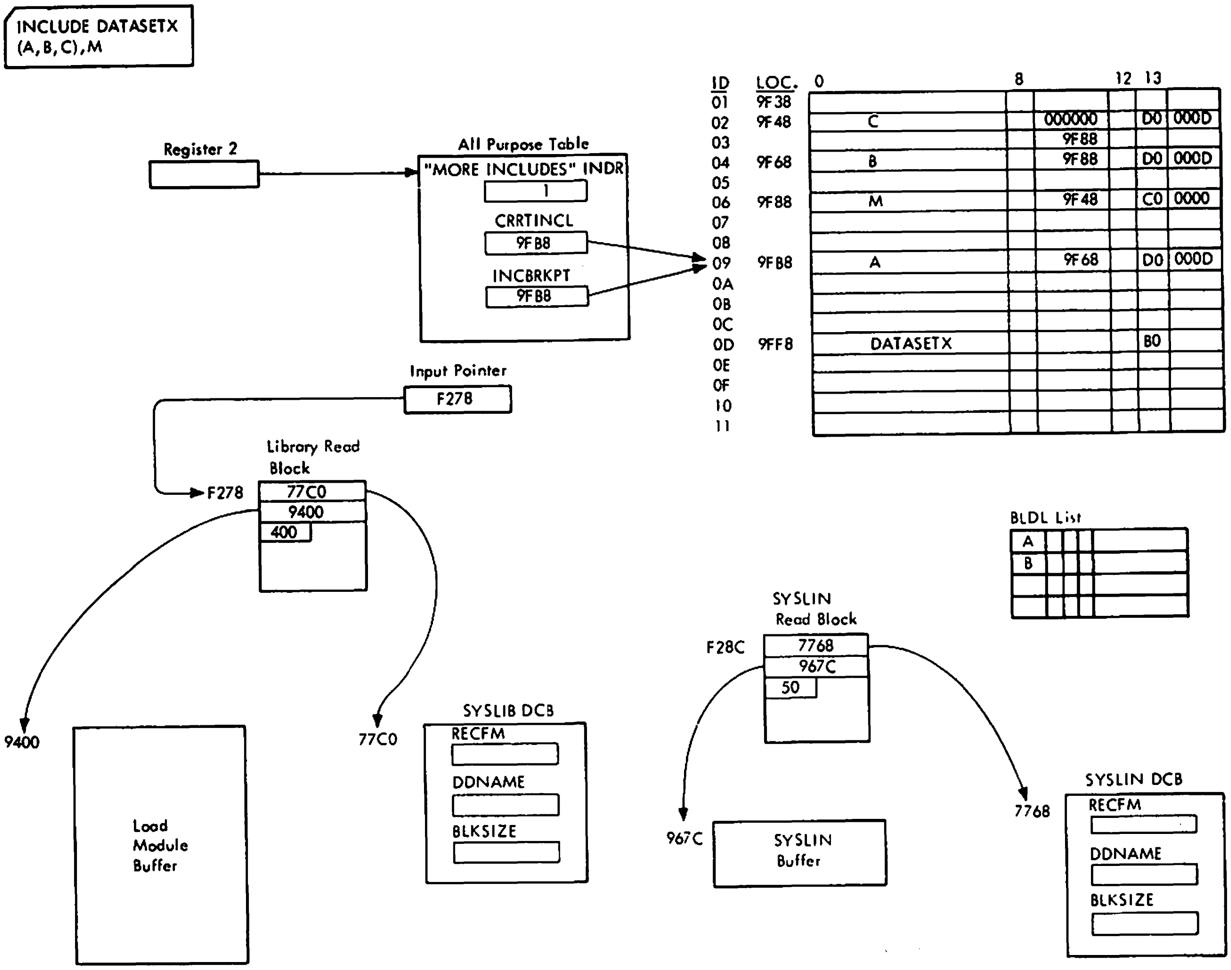
When the BLDL list is completed, the partitioned data set is opened, and the record format field (RECFM) in the DCB is tested to determine whether the included modules are load modules (undefined format) or object modules (fixed format). If they are load modules, the "load module" indicator is set in the APT. This indicator is tested when each module is read in. A BLDL macro instruction is then executed for the member names in the list. The list is then examined in the order specified in the INCLUDE statement to obtain the attributes of each included module (if it is a load module); the attributes of the output load module may be "downgraded" accordingly in the APT.

If the BLDL macro instruction was successful for a particular member, the member is read in. The FIND macro instruction and the directory entry obtained from BLDL are used to set a pointer in the DCB to the first record of the member. If the BLDL was not successful for a particular member, a diagnostic message is printed.

The INCLUDE processor checks the PDS directory information returned by BLDL for an included load module to determine whether the load module is in overlay format. If it is, an indicator is set in the all-purpose table, so that the ESD processor can interpret byte 12 of each ESD item as a segment number rather than as AMODE/RMODE data.

An example of INCLUDE processing is given in Figure 17 on page 49. The input pointer is set to the address of the library read block. The address of the current include item is contained in the APT.

Assuming that no includes have yet been processed, A will be the first item examined. The subtype 'D0' indicates that A is a member of a partitioned data set, so A will be entered into the BLDL list. The pointer 000D refers to the data set DATASETX. The next item in the include chain, B, is also a member of DATASETX, so it is added to the BLDL list. The next item in the chain, M, is a sequential data set (subtype C0), so the BLDL list is completed with two entries (A and B). Assuming that DATASETX is not currently open and the SYSLIB DCB is not opened for another data set, the SYSLIB DCB is opened for DATASETX. (The RECFM field of the data set DSCB is merged into the DCB.) Assuming that the RECFM field indicates undefined (U) format, a load module indicator is set in the APT, and a pointer to the load module buffer is placed in the library read block. The attributes of A and B are obtained using the BLDL macro instruction, and the attributes previously specified are updated accordingly. (The attributes of the output load module may be downgraded as a result.) A pointer in the DCB is then set to the first record of member A, using the FIND macro instruction, and the "include initiated" indicator is set in the APT.



ID	LOC.	0	8		12	13	
01	9F38						
02	9F48	C		000000		D0	000D
03				9F88			
04	9F68	B		9F88		D0	000D
05							
06	9F88	M		9F48		C0	0000
07							
08							
09	9FB8	A		9F68		D0	000D
0A							
0B							
0C							
0D	9FF8	DATASETX				B0	
0E							
0F							
10							
11							

Figure 17. Include Processing

Member A is read using the input pointer and library read block. Module A is then processed. When the end of module A is reached, item A is deleted from the chain and the CESD line is marked null. Member B is then read and processed.

When the end of module B is reached, item B is deleted from the chain, the CESD line is marked null, and the remainder of the chain is processed.

Automatic Library Call Processing

Automatic library call processing is required:

- At the end of SYSLIN input when unresolved ERs still exist and the NCAL option was not specified

- When a NAME statement has been detected (provided that the NCAL option was not specified and no more entries in the include chain are to be processed)

Automatic library call processing consists of two series of CESD scans. The first series of scans operates on unresolved ERs specified on LIBRARY statements. It finds the first ddname that contains a pointer in the chain pointer field (bytes 14 and 15). Such an entry is the first item in a chain of members associated with this ddname; there is a distinct chain for each ddname that was specified on a LIBRARY statement. Chained member names for a particular ddname are entered into a BLDL list, which is processed as previously described under "INCLUDE Processing."

The scan of the CESD continues until all ddname chains have been processed. A second scan of the CESD then searches for ERs not specified on LIBRARY statements and attempts to resolve them by calling members of the same name from SYSLIB.[7]

An example of automatic library call processing is given in Figure 18 on page 51. Diagram A shows two library chains that were built in the CESD by the library statement processor. In Figure 17 on page 49, Diagram B, an SD item for JOE has been entered in the CESD, resolving the reference to JOE. (JOE was removed from the chain by ESD processing, and the LIB1 chain ID now points to the line containing TOM.) Automatic library call processing operates on the library chains, as modified by ESD processing (Diagram B).

In the first series of scans, the CESD is searched for a ddname (type 02, subtype B0) with a chain pointer. The ddname item LIB1 is found; its chain ID points to TOM. Because TOM is unmatched (subtype 02), it is not called and because TOM is the last item in the chain (0 in the chain ID field), the scan is resumed for another ddname with a chain pointer. LIB2 is found; its chain ID points to SAM. No call is issued for SAM, because it is unmatched. The chain ID of SAM points to PETE, which is matched (indicating that PETE is an external reference, and not just an operand of a LIBRARY statement). PETE is entered in the BLDL list; because PETE is the last item in the chain, the list is completed with one entry.

LIB2 is opened and the BLDL macro instruction is used to obtain the attributes of PETE (the attributes of PETE are not obtained if the format is fixed (F)). A "BLDL attempted" indicator is set for the CESD entry for PETE, so that no other search for PETE will be made in the event of an unsuccessful BLDL, or nonresolution of the ER for PETE by the member PETE. The FIND macro instruction is used to set a pointer in the SYSLIB DCB to the member PETE. PETE is then read in.

When processing for PETE is completed, the scan for ddnames resumes at the beginning of the CESD rather than at the CESD line where the scan was interrupted, because additional ddname items may have been entered at any available line in the CESD. (Object modules with additional LIBRARY statements may have been read in.) When the last line of the CESD is reached, the second series of scans is begun.

[7] SYSLIB is the standard library whenever the linkage editor is executed as a job step. If another program links to the linkage editor, the ddname of the standard library is passed in a parameter list.

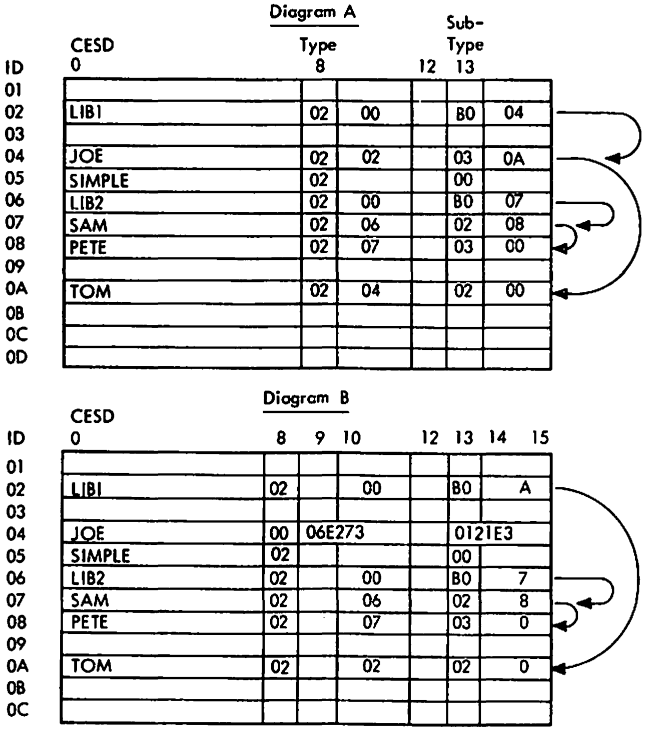


Diagram A

ID	CESD 0	Type 8		12	Sub-Type 13	
01						
02	LIB1	02	00		B0	04
03						
04	JOE	02	02		03	0A
05	SIMPLE	02			00	
06	LIB2	02	00		B0	07
07	SAM	02	06		02	08
08	PETE	02	07		03	00
09						
0A	TOM	02	04		02	00
0B						
0C						
0D						

Diagram B

ID	CESD 0	8	9	10	12	13	14 15
01							
02	LIBI	02		00		B0	A
03							
04	JOE	00	06E273			0121E3	
05	SIMPLE	02				00	
06	LIB2	02		00		B0	7
07	SAM	02		06		02	8
08	PETE	02		07		03	0
09							
0A	TOM	02		02		02	0
0B							
0C							

Figure 18. Automatic Library Call Processing

During the second series of scans, the CEDS is searched for "unmarked" external references (type 02, subtype 00). These are ER items not specified on LIBRARY statements. In Diagram B, the scan finds SIMPLE. Assuming that SYSLIB is the ddname for the standard library, SIMPLE is called from SYSLIB in the same way that PETE was called from LIB2. Every time automatic library call processing is resumed after a module is read, the second series of scans resumes at the beginning of the CESD (because ER items from a library member may have been entered in any available CESD line).

When the second series of scans is finished, input processing is complete.

INTERMEDIATE PROCESSING

The intermediate processing comprises modules HEWLFADA, HEWLFOUT, HEWLFENS, HEWLFENT, and (optionally) HEWLFMAP.

When all input processing is completed, the second phase of the linkage editor (intermediate processing) begins operation. The two major functions of the second phase are address assignment and intermediate output.

ADDRESS ASSIGNMENT

At the conclusion of input processing, address assignment processing is required (see "Diagram 13. Address Assignment" on page 87). Address assignment includes the following operations:

- Deletes CESD entries for ER items marked included, called, ddname, or overlay in the subtype field. These lines are marked null and are deleted if the module is processed again in a subsequent execution of the linkage editor.

- Computes, for programs in overlay, the size of SEGTAB,[8] enters the size in the all-purpose table, and places a private code delete entry for the SEGTAB in the CESD. The PC-delete entry is deleted from the module if it is processed again by the linkage editor (see "Diagram 13. Address Assignment" on page 87, area A).

- Determines whether the first text record of a load module is not assigned to address 0. If it is not assigned to address 0, a private code delete entry of 1 byte is created in the CESD. The PC-delete entry is deleted from the module if the module is later reprocessed by the linkage editor.

- Enters segment numbers for label references in the CESD. If the program is in an overlay structure, the calls list (built during RLD processing) is also scanned, and pointers from one chain of calls to the next chain are entered (area B); the number of ENTAB bytes[9] for each segment is determined; and a PC-delete entry is placed in the CESD for each ENTAB (see "ENTAB Size Determination").

- Assigns temporary linked addresses to SD, PC, and CM entries in the CESD (area C). SDs and CMs that have entries in the ORDER table are addressed first in the order of their appearance in the table. The remaining control sections are then assigned addresses to the SDs, PCs, and CMs that have no entries either in the ORDER table or the text I/O table. To avoid assigning addresses to any SD or CM more than once, a "processed" bit (bit 4 of the 'type' byte in the CESD) is set in each CESD entry when it is first processed. The bit is reset to zero in the final scan of the CESD.

- Considers each segment to be at zero origin. The temporary starting address of each control section is computed with consideration for its location in the segment relative to the zero origin (plus any adjustments for boundary alignment). These addresses are temporary because the starting addresses of the segments must later be relocated with respect to their positions in the overlay tree. If the program is not in overlay (consists of a single segment), the addresses are final because no further relocation by address assignment is necessary.

- Performs page alignment while assigning temporary linked addresses if the program is not in overlay. The ORDER table is searched for a match of the CESD ID of the SD or CM being processed. When a match is found, and page alignment is specified, the assigned address is forced to a 4K-byte boundary. If the ALIGN2 option is taken, the address is forced to a 2K-byte boundary.

- Computes the temporary relocation constant for each control section (the difference between the temporary linked address and the assigned address in the relocation constant table (RCT) (area D). If the program is not in overlay, these are the final relocation constants (relative relocation factors).

- Accumulates the length of each segment in the leftmost 3 bytes of an entry in the segment length table (SEGLGTH). The boundary alignment factor of the first control section in the segment is placed in the fourth byte of the entry.

[8] SEGTAB size = 24 + (4 x number of segments).

[9] ENTAB size = 12 + (12 x number of unique downward calls per segment).

- Determines the address of each PR entry in the CESD, using the total length of all PRs previously encountered plus the boundary alignment factor. This address is placed in the CESD entry for the PR. The length of this PR is then added to the cumulative PR length.

- Processes the SEGLGTH table (if the program is in overlay) to determine the starting address of each segment relative to the beginning of the program (area E). SEGTAl is checked to find the proper location of each segment in the tree. SEGLGTH at this time contains the length of each segment. To determine the starting address of a segment, the length of all previous segments in the same path are added, together with any adjustments for boundary alignment. (Boundary alignment adjustment is determined by the last 3 bits of the address of the first control section in a segment.) This sum, minus the boundary alignment factor for the segment, is the <u>segment relocation constant</u> (SRC). The SRC is then placed in the rightmost 3 bytes of the SEGLGTH table. The sum of the SRC, the boundary alignment factor, and the segment length is placed in the leftmost 3 bytes of the SEGLGTH table entry for the segment. It is the length of the path of the segment (including the segment itself). At the completion of this process, the entry in SEGLGTH for each segment contains the cumulative length of its path; the longest of these lengths is the program length.

- Performs a second scan of the CESD if the program is in an overlay structure. The segment relocation constant in the SEGLGTH table is added to the temporary linked address in the CESD entry for the control section; this sum is the <u>final linked address</u>. The SRC is also added to the temporary relocation constant table; this sum is the <u>final relocation constant</u> for the control section.

- Assigns final linked addresses in ascending order of segments if page alignment is specified for any SD or CM type symbol. For each segment, three cycles of scanning are performed. First, SDs and CMs having entries in the ORDER table are processed. The final address is calculated by adding the SRC and the temporary linked address, and is aligned on a page boundary, if required. A cumulative count of any increment within a segment caused by page alignment is kept, in order to assign correct addresses to the unprocessed SDs, PCs, and CMs. Next, the text I/O table is scanned for the remaining SDs and PCs in the segment. These SDs and PCs are assigned final addresses. Finally, a scan of the CESD gives addresses to all unprocessed SDs, PCs, and CMs in the segment. For every processed SD, its entry in the relocation constant table is calculated. Before going on to the next segment, the length of the segment just processed and the SRC of the next segment are updated.

- Makes a final scan of the CESD to assign a final linked address to each label reference.

  The CESD entry for each LR contains a reference to the control section in which it resides. The relocation constant for that control section is located in the RCT and is added to the temporary linked address in the CESD entry for the LR. This sum, the final linked address for the LR, is placed in the CESD.

- Marks the program as not executable if there are still unresolved ERs and if neither the NCAL option nor the LET option has been specified. Unresolved WXs do not inhibit program execution.

- Builds the alias table and compute an entry point for the program (see "Entry Processing").

ENTAB Size Determination

ENTAB size determination consists of computing the size of ENTABs so that the size of each segment in an overlay program can be determined, and relative relocation factors can be computed for use by second pass processing. The size is determined by the number of downward calls, or calls across regions, to symbols that are not referred to by segments higher in the path of the calling segments.

An example of ENTAB size determination is given in Figure 19 on page 55. The overlay tree structure shown in the illustration consists of nine segments residing in two regions; all references between segments are made using V-type address constants. Functions of ENTAB size determination are:

- Scanning the CESD for LR entries and entering their segment numbers. In Figure 19, item 6 is an LR item; its ID/length field points to the CESD entry for the control section in which it resides (line 3). The segment number contained in line 3 (segment number 3) is entered in the segment number field of the LR item.

- Scanning the calls list, inserting chaining values that point from one group of R and P pointers to the next.

- Scanning the calls list for each segment (starting with segment 1), and finding symbols referred to by that segment. For each reference found, the type of call (upward, downward, or exclusive) is determined. If an ENTAB is required for the segment, its size is determined and a PC-delete entry for the ENTAB is made in the CESD. Referring to Figure 19, the segments are processed in the following manner:

  1. The calls list is scanned for P pointers that refer to control sections in segment 1. If one is found, the associated R pointers (which refer to referenced symbols) are examined to determine the segment in which each referenced symbol resides. In Figure 19, the fifth P pointer refers to line 7 of the CESD, which contains an SD entry for a control section in segment 1. The associated R pointers refer to line 6 (symbol B in segment 3) and line 4 (symbol C in segment 5). For each reference, the type of call (upward, downward, or exclusive) is determined, using SEGTA1 and the segment numbers of the calling and called segments. In Figure 19, SEGTA1 indicates that segment 1 is in the path of segments 3 and 5; therefore, the calls from segment 1 to B and C are downward calls. This is noted in the downward calls list by entering segment number 1 in the lines referred to by the R pointer (lines 6 and 4). Since segment 1 is the root segment, it must have an ENTAB; the size of the ENTAB is determined and a PC-delete entry for it is created in the CESD.

  2. When the scan for segment 1 is completed, the calls list is scanned for P pointers that refer to segment 2. In Figure 19, the third P pointer in the calls list refers to CESD line 6, which contains segment number 3. In this case, however, no entry is made in the downward calls list because it indicates a call to B in segment 3 from segment 1, which is higher in the path of the calling segment (segment 2). No ENTAB is required for segment 2 because the reference to symbol B in segment 2 can be resolved through the ENTAB entry in segment 1.

  3. The calls list is scanned for P pointers that refer to segment 3. In Figure 19, the fourth P pointer in the calls list refers to CESD line 3 (segment 3). The R pointer refers to CESD line 8 (segment 8). SEGTA1 indicates that the call from 3 to 8 is downward, across regions, and the call is noted in the downward calls list. Segment 3 requires an ENTAB, because it contains

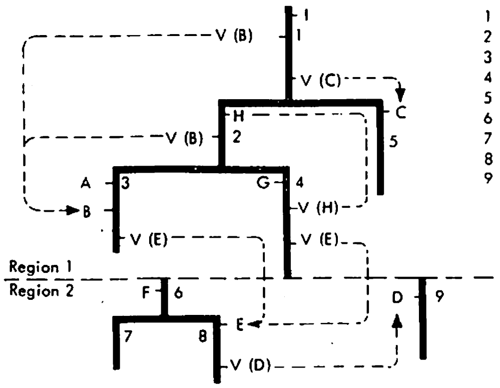
a downward call to a symbol not referred to by a segment in the path of the calling segment; the ENTAB size is determined, and a PC-delete entry for the ENTAB is created in the CESD.

4. The calls list is scanned for P pointers that refer to segment 4. In Figure 19, the first P pointer in the calls list refers to CESD line 9 (segment 4). The R pointers refer to line 2 (segment 2) and line 8 (segment 8). SEGTAl indicates that the call from 4 to 2 is upward, while the call from 4 to 8 is downward, across regions. The upward call is ignored, because the address constant can be resolved directly to the referenced symbol. The downward call from 4 to 8 is noted in the downward calls list, replacing the previous entry for segment 3 (because no segment with a segment number greater than 4 can have segment 3 in its path). Because an ENTAB is required, the size is determined and a PC-delete entry is created in the CESD.

This process continues until all segments have been processed. The required ENTABs are built during second pass processing (see "ENTAB Creation" and "Relocation of V-Type Address Constants in Overlay").

CESD

	Symbol	Type	Chain Address	Seg No	Sub-Type	Length / ID
1	D	SD		9		
2	H	SD		2		
3	A	SD		3		
4	C	SD		5		
5						
6	B	LR		3		3
7	I	SD		1		
8	E	SD		8		
9	G	SD		4		
10	F	CM		6		
11		PC		7		
	*	PC(d)		1		60
	†	PC(d)		1		36
	†	PC(d)		3		24
	†	PC(d)		4		24



SEGTAl

1	0
2	1
3	2
4	2
5	1
6	0
7	6
8	6
9	0

Downward Calls List

1	
2	
3	
4	1
5	
6	1
7	
8	~~3~~ 4
9	
10	
11	

* PC - delete type entry for SEGTAB
† PC - delete type entries for ENTABs

CALLS LIST

8	9	2	8	6	8	1	6	2	6	6	3	8	8	7	6	4	0
* CV	P	R	R	CV	P	R	CV	P	R	CV	P	R	CV	P	R	R	End of Calls List

* CV = Chaining Value (gives number of bytes to next CV)

Figure 19. ENTAB Size Determination

Entry Processing

Entry processing includes the following operations:

- Entering in the alias table any alias symbols that were chained together and saved in the CESD by the ALIAS statement processor. Each entry in this table consists of an 8-byte symbol field and a 2-byte ESDID field. For each saved alias symbol, the entry processor scans the CESD for a matching SD or LR entry. If no match is found, a zero is placed in the ESDID field of the alias table entry for the symbol. If a matching SD or LR entry is found, the ESD ID of the alias entry in the chain is placed in the ESDID field of the alias table entry for the symbol (see Figure 20 on page 57). The address assigned by linkage editor to the matching SD or LR and the ESD ID of its control section are placed in the CESD entry for the chained symbol, and the type of the chained symbol is changed to null.

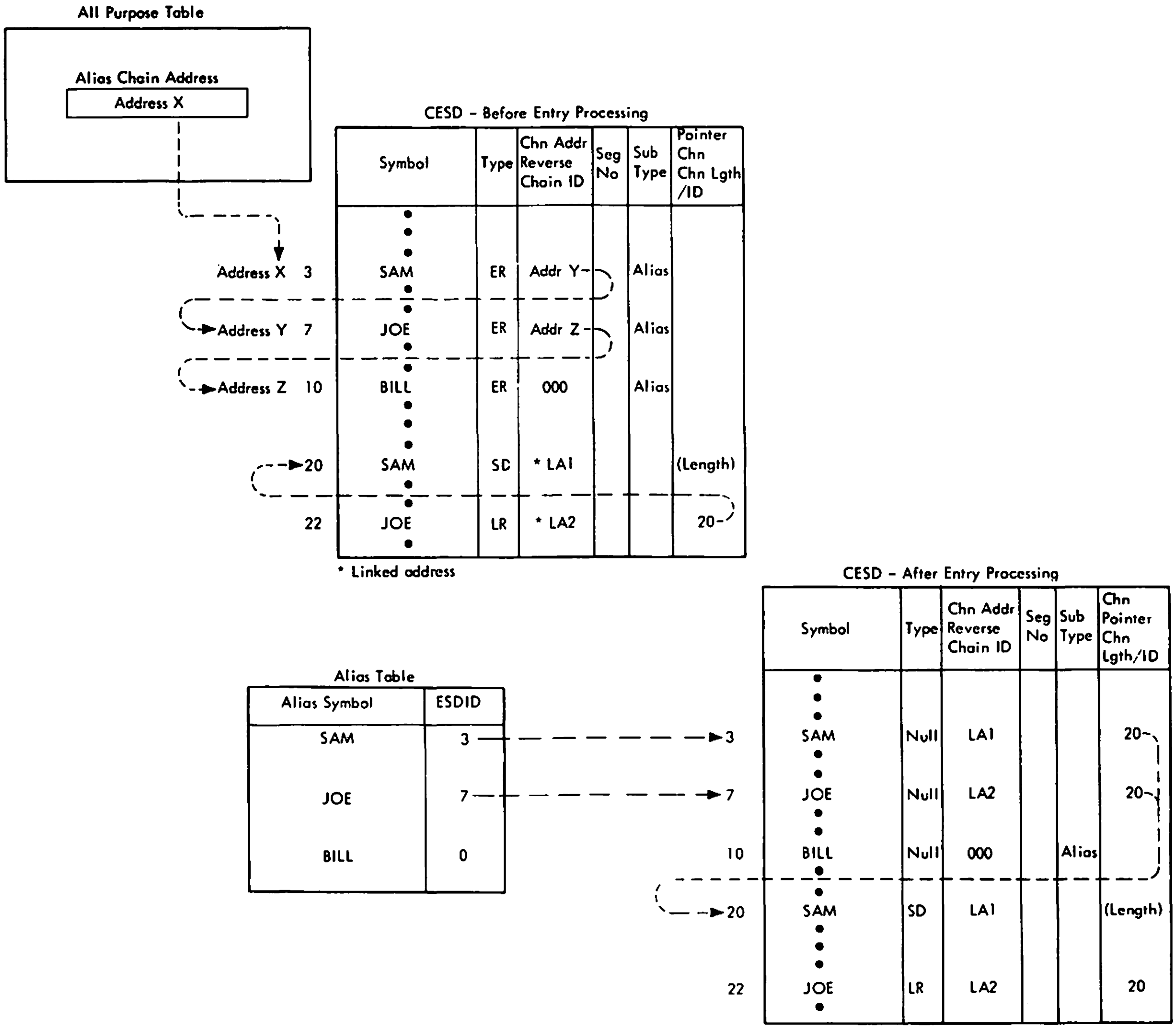
- Determining whether the entry point was specified as an address on an END record, or as a symbol on an ENTRY statement or END record:

  1. If the entry point was specified as an address on an END record, the assigned address is determined by either absolute or relative relocation. If the ID on the END record referred to an ER which was resolved with an SD or LR, the address assigned by the linkage editor to the SD or LR is added to the address from the END record (absolute relocation). If the ID on the END record referred directly to an SD or PC, the relocation constant for the SD or PC is added to the address from the END record (relative relocation).

  2. If a symbolic entry point was specified on an ENTRY statement or END record, the CESD is scanned for a matching SD or LR symbol. The address of the matching symbol is used as the entry point.

  3. If no entry point was specified, the starting address of the SD or PC control section (not marked delete) with the lowest assigned address is chosen as the entry point. The entry point associated with the main name (not an alias) and all alias entry points must be in segment 1 if the program is in overlay.

- Assigning the addressing mode for the main entry point into the output load module. If the load module is in overlay format, the addressing mode is 24; otherwise, the addressing mode is obtained from the CESD entry that defines the SD or PC that contains the entry address. The addressing mode, along with the entry address and the ESDID of the SD or PC, is saved in the all-purpose table.



CESD - Before Entry Processing

	Symbol	Type	Chn Addr Reverse Chain ID	Seg No	Sub Type	Pointer Chn Chn Lgth /ID
Address X 3	SAM	ER	Addr Y		Alias	
Address Y 7	JOE	ER	Addr Z		Alias	
Address Z 10	BILL	ER	000		Alias	
20	SAM	SD	* LA1			(Length)
22	JOE	LR	* LA2			20

Alias Table

Alias Symbol	ESDID
SAM	3
JOE	7
BILL	0

CESD - After Entry Processing

	Symbol	Type	Chn Addr Reverse Chain ID	Seg No	Sub Type	Chn Pointer Chn Lgth/ID
3	SAM	Null	LA1			20
7	JOE	Null	LA2			20
10	BILL	Null	000		Alias	
20	SAM	SD	LA1			(Length)
22	JOE	LR	LA2			20

Figure 20. Processing of Alias Symbols by the Entry Processor

INTERMEDIATE OUTPUT PROCESSING

Intermediate output processing includes the following operations:

- Writing out the CESD on SYSLMOD in groups of 15 entries per record. The last record may consist of less than 15 entries. In writing CESD records on to SYSLMOD, the intermediate output processor sets a flag in the control information indicating the content of byte 12 in the CESD entries in the record. If the CESD entries contain segment numbers (that is, the load module is in overlay format), the flag is off; if the CESD entries contain AMODE/RMODE data, the flag is on.

- Building and writing out the IDRs from the IDR tables (IDRTRTAB, IDRTITAB, IDRUDTAB, and IDRZPTAB) onto SYSLMOD. The linkage editor IDR is also built and written on to SYSLMOD.

- Building a half ESD (HESD), consisting of the last 8 bytes of each CESD entry. (The symbol is deleted from each CESD entry to conserve virtual storage space during second pass processing.) The HESD is not complete at this time. (The ID of each label reference is used in building the scatter and translation tables.)

- Building and writing out the segment table (SEGTAB), preceded by a control record describing it if the program is in overlay. SEGTAB contains information required by the overlay supervisor.

- Building and writing a 1-byte text record if the first load module text record does not begin at address 0. The 1-byte text record is preceded by a control record describing it.

- Building a scatter table and a translation table for a program that is to be scatter loaded, and writing out scatter/translation records in a form acceptable to program fetch at execution time. The scatter/translation information is written out on SYSLMOD in 1024-byte records. The first 4 bytes of each record are used to identify the records as scatter/translation information. If the length of scatter/translation information is greater than 1020 bytes, the last 1020 bytes (plus 4 bytes of header information) are written out as the first scatter/translation record. The data in the last record may be 1020 bytes or less (see Figure 21). In creating the scatter table entries, the RMODE/RSECT data is obtained from the HESD entries (byte 4) and inserted into the flag byte.

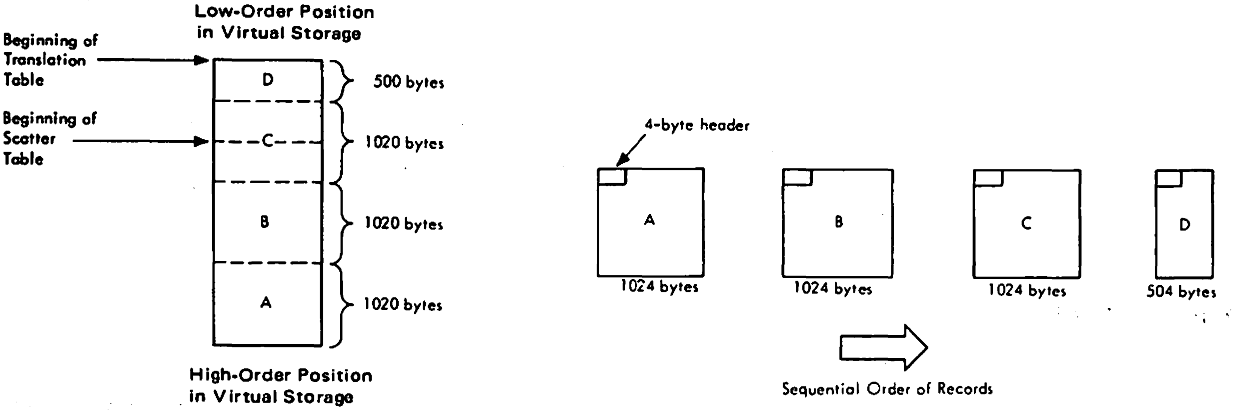

Low-Order Position in Virtual Storage
Beginning of Translation Table
Beginning of Scatter Table
D
C
B
A
500 bytes
1020 bytes
1020 bytes
1020 bytes
High-Order Position in Virtual Storage
4-byte header
A
B
C
D
1024 bytes
1024 bytes
1024 bytes
504 bytes
Sequential Order of Records

Figure 21. Writing Scatter/Translation Records

- Determining the control section containing the last text in the program (or in each segment if the program is structured for overlay) and the highest segment number of the segments that contain text. (This information is necessary so that second pass processing can determine when to set the end-of-segment or end-of-module indicator.) The highest ESD ID is determined by scanning the text I/O table for the ESD IDs of control sections that contain text. This ESD ID is entered into the high ID (HIID) table along with its associated segment number.

- Determining, via bits in the all-purpose table (APT), if the MAP option has been specified, or if the XREF option has been specified and all RLDs are in storage. If either of these conditions exists, the module map and/or the cross-reference table are produced. If the XREF option is specified and all RLDs are not in storage, XREF processing will be done as part of final processing.

- If the ORDER option has been invoked during input processing, the text I/O table and the text note list II (formed after merging all text note lists from SYSUT1) are sorted according to the ORDER table. The sorting, however, preserves the original order for those control sections that do not have entries in the ORDER table.

MAP/XREF Processing

When MAP/XREF processing is performed as part of intermediate output processing, a table address is obtained from the APT, and a table of 2-byte entries pointing directly to the CESD is constructed. The CESD records for the current segment are gathered and sorted by address. The module map is then printed out; the map lists, in ascending order according to their assigned origins, all control sections contained in the output module and the entry points within the control sections. Control sections in an overlay output module are grouped by segment.

If XREF processing is done during intermediate output processing, RLD items are incompletely relocated; their addresses are relative to the origins of their respective CSECTs rather than the origin of the load module, and the address of each RLD must be added to the linkage editor assigned address of its corresponding control section before the cross-reference table is produced. The cross-reference table includes a module map and a list of all references within a given segment that refer across control section boundaries. Each entry in the list contains the address of the reference, the symbol to which it refers, and the name of the control section in which the symbol is defined. For overlay programs, each item in the list also contains the number of the segment in which the symbol is defined.

If the MAP and XREF options are processed during intermediate output processing, disposition messages and the diagnostic message directory are printed after the module map and cross-reference table. If the cross-reference table is produced during final processing, the disposition messages and the diagnostic message directory are printed before the cross-reference table.

SECOND PASS PROCESSING

Second pass processing comprises modules HEWLFSCD and HEWLFREL.

After intermediate processing is completed, the third phase of the linkage editor (second pass processing) begins operation (see "Diagram 14. Data Movement During Second Pass Processing" on page 88). The major functions of second pass processing include:

- Relocating address constants contained in the text.
- Creating control/RLD records.
- Writing TXT and control/RLD records on SYSLMOD in a format that can be loaded by program fetch. Included in the control information of the control or control/RLD record that precedes each text record is a count of the RLD and control/RLD records that follow the text record. This count is used by program fetch to build optional channel programs when loading the load module.
- Creating ENTABs and associated RLD items for overlay modules.

SINGLE PASS PROCESSING: Indicators residing in virtual storage in the text I/O table and the RLD note list are checked to determine whether text and RLD records have been written on SYSUT1, or have been retained in the input text buffer and the RLD buffer. If either text or RLD records have been retained in storage, single pass processing is in effect for that record type. If two pass processing is in effect, the records are read into the buffers from SYSUT1.

ORDERING OF TEXT: In two pass processing, the ID sequence in the text I/O table is used to determine the order in which CSECTs are to be read into the second pass text buffer (which is physically the same storage area as the input text buffer). The text I/O table entry for each ID and the corresponding text note list entry are used to locate text on SYSUT1 (see "Diagram 14. Data Movement During Second Pass Processing" on page 88, area A). Text is read into the buffer one multiplicity at a time, using the displacement field in the text note list to determine where within the buffer the text must be placed. Information about the text is entered into the second pass text control table, which is used to control subsequent processing of the text (area B).

SECOND PASS RLD BUFFERS: When the required text is in the text buffer, the corresponding RLDs are read into the RLD input buffer, using the RLD note list to locate the RLD records (area C). The RLD input buffer can contain two RLD records from SYSUT1; for each RLD input buffer area, an RLD input control block is maintained (area D). The RLD output buffer is 768 bytes long and is divided into three buffer areas (the maximum RLD output record is 256 bytes long); for each RLD output buffer area, an RLD output control block is maintained (area F). While text is being relocated, the control record for that portion of text occupies one of the output buffers; the other two output buffers contain the relocated RLDs for the text being processed (area E). If the relocated RLDs exceed two buffers, the control record is written on SYSLMOD; relocated RLDs may then be moved into the third output buffer.

When all three RLD output buffers and the RLD input buffers are filled and additional RLDs are required to relocate the text currently being processed, the contents of the output buffer must be written out. However, to maintain the required TXT/RLD sequence in the output module (area G), the associated text must precede the RLD record. Space for the text is reserved in the output module by writing the incompletely relocated text; the contents of the RLD output buffer may then be written, and processing can continue. When the text is completely relocated, it is written over the space reserved for it using the XDAP ("execute direct access program") macro instruction.

GROUPING SYSLMOD OUTPUT: As many CSECTs as will completely fit in one SYSLMOD record (up to a maximum of 60) are grouped and written as one record. RLDs are grouped to correspond to the grouping of their associated text. If the overlay attribute is specified, only CSECTs belonging to the same segment will be grouped.

If a CSECT is larger than the SYSLMOD record size, the CSECT is divided into multiplicities, each multiplicity being equal to the SYSLMOD record size. The length of the last multiplicity may be less than the SYSLMOD record size. Each multiplicity is written as a record, followed by RLDs associated with only that multiplicity.

Note: If the downward compatible attribute (DC) or the scatter format attribute (SCTR) is specified, CSECTs will not be grouped.

END-OF-MODULE: When control sections for all segments of the output load module have been processed (determined via the "high ID" indicator in the HESD type field and the "last segment with text" field in the all-purpose table), indicators are set in the last control/RLD record to mark it as the end of the module. The control/RLD record is written out on SYSLMOD, and second pass processing is completed.

Note: If the output load module is to be structured for overlay, a list of relative track addresses (TTR list) is created to be used by program fetch when it loads the segments into virtual storage for execution. The TTR list contains one entry for each segment in the overlay load module. Each entry contains the relative track address of the first record (control record) of a segment, except for the first segment, which contains the relative track address of the first text record. A PC-delete control section that contains ENTAB entries in each segment where the text requires them and the RLD records required by program fetch to relocate address constants contained in the ENTAB entries are also created. The fourth byte of each entry contains the number of blocks of the text in the corresponding segment, or 255 (X'FF') if there are more than 255 blocks of text.

Relocation of Address Constants

There are two types of relocatable address constants:

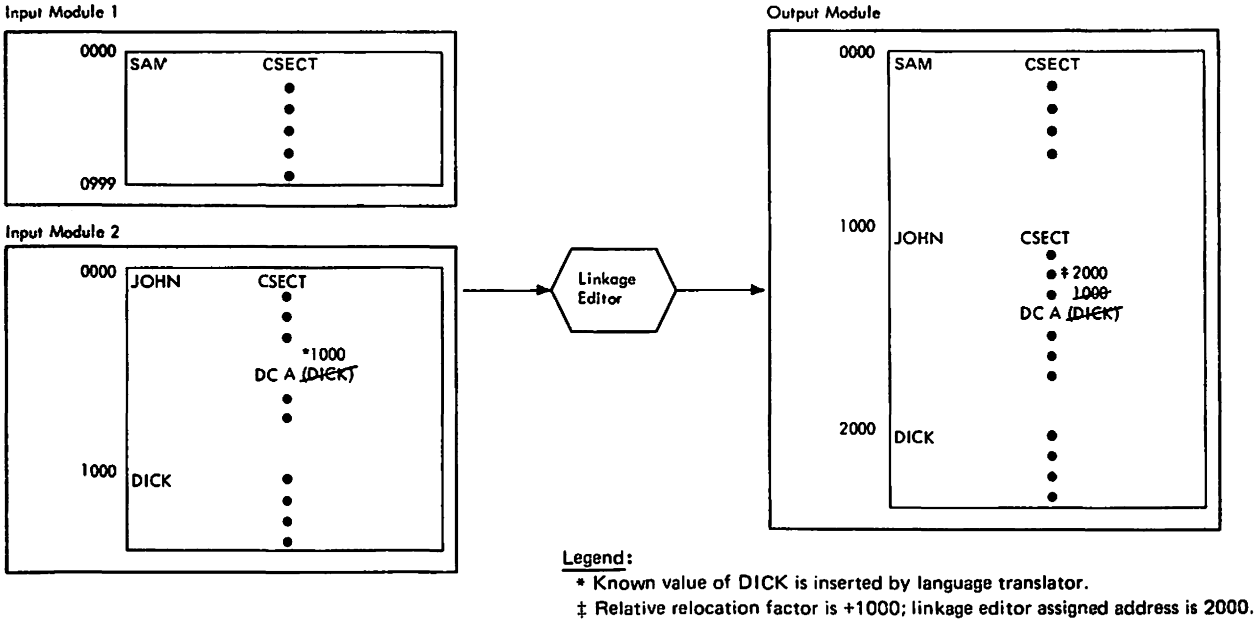
- Branch-type, such as DC V(X)
- Nonbranch-type, such as DC A(X)

The value of a branch-type or nonbranch-type address constant depends on a symbol in the CESD. To adjust an address constant to its proper value in the output load module, the linkage editor uses an absolute or relative relocation factor. The absolute relocation factor is the address assigned by the linkage editor to the symbol on which the value of the address constant depends. The relative relocation factor is the difference between the address assigned to the symbol by the linkage editor and the address of the symbol in the input module. The relative relocation factor may be positive or negative.[10] The absolute and relative relocation factors of each symbol in the CESD are computed during address assignment and are saved in the half ESD (HESD).

Relocation of Nonbranch-Type (A-Type) Address Constants

A <u>relative relocation factor</u> is used for a nonbranch-type address constant if the symbol on which its value depends is in the same input module as the control section that contains the address constant. (The address constant and the symbol it refers to were assembled or compiled together, or were previously processed together by the linkage editor.) An example of relative relocation of nonbranch-type address constants is shown in Figure 22 on page 62. Because the address of DICK is known, the language translator places it in the value of the address constant. DICK is a known value prior to linkage editor processing (not an external reference in the input); therefore, a relative relocation factor (+1000) is used to relocate DICK during linkage editor processing.

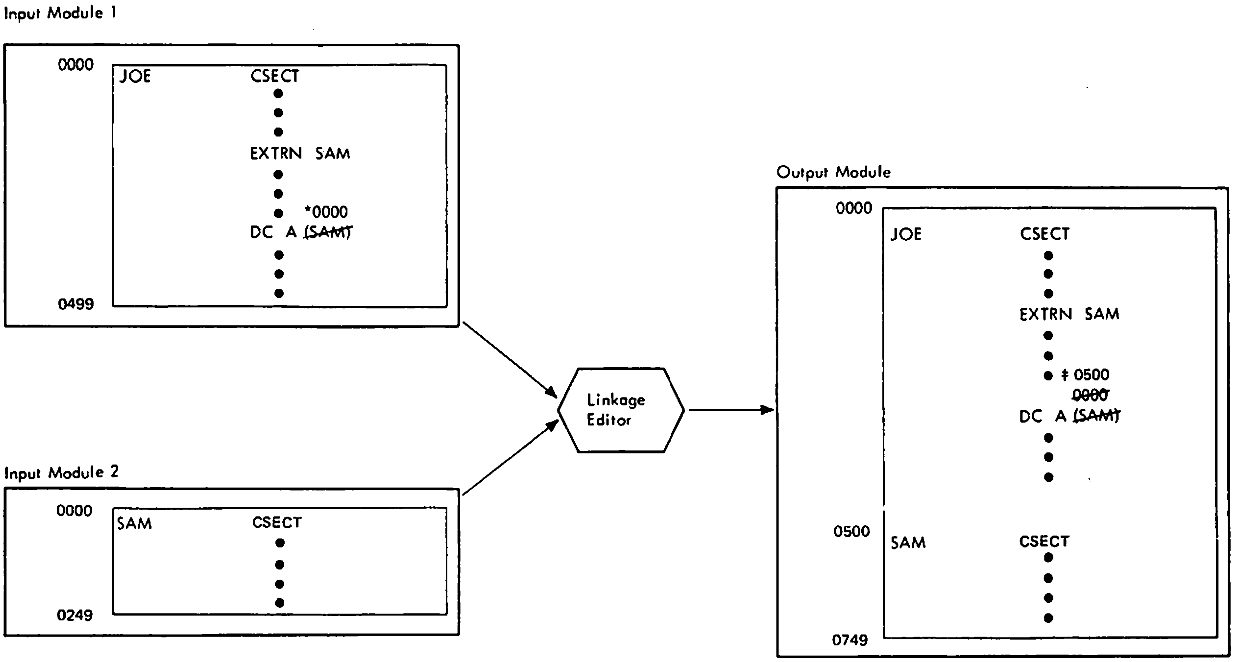
[10] If it is negative, an indicator is set in the HESD to note that it is in complement form.



Legend:
* Known value of DICK is inserted by language translator.
‡ Relative relocation factor is +1000; linkage editor assigned address is 2000.

Figure 22. Nonbranch-Type Address Constants—Relative Relocation

An absolute relocation factor is used for a nonbranch-type address constant if the symbol referred to by the address constant does not have a defined value within the same input module. (The R pointer of the RLD item refers to an external reference.) An example of absolute relocation of a nonbranch-type address constant is shown in Figure 23 on page 63. In this example, the value of SAM is unknown when input module 1 is processed by the language translator; therefore, zeros are placed in the value of the address constant. During second pass processing, the absolute relocation factor (the linkage editor assigned address) is used to relocate the address constant.



Legend:
* Language translator inserts zeros because value of SAM is unknown.
‡ Actual address of SAM in the output module (0500) is added to value of address constant.

Figure 23. Nonbranch-Type Address Constants—Absolute Relocation

Figure 24 on page 64 shows the use of both a relative relocation factor and an absolute relocation factor in relocating a symbol. Two input modules are to be processed by the linkage editor. Input module 1 contains a nonbranch-type address constant whose value depends on the symbol PETE; PETE is an external reference in the same module. The language translator has assigned a value of +10 to the address constant. The R pointer of the RLD item refers to the ER entry for PETE in the ESD; this entry contains zeros in the origin and length fields. The P pointer refers to the SD entry for the control section that contains the address constant.

Input module 2 contains two control sections, BOB and PETE. BOB contains a nonbranch-type address constant whose value depends on PETE; because PETE has a defined value of (300) in the same module, the language translator has used that value to compute the value of the address constant (PETE + 10 = 310). The R pointer of the RLD item refers to the SD entry for PETE in the ESD; the P pointer refers to the SD entry for BOB (the control section that contains the address constant).

During linkage editor processing, the ER and SD entries for PETE are merged into one CESD entry; the R pointers of both RLD items in the output module will refer to that entry. The RLD P pointer for the address constant in control section BILL will refer to the SD entry for BILL; the P pointer for the other address constant will refer to the SD entry for BOB. In the output load module, both address constants will contain the same value. Because the R pointer of the RLD item in input module 1 refers to an ER entry in the ESD in that module, it is marked for absolute relocation; the absolute relocation factor for PETE (+500) is added to the value (+10) assigned by the language translator. Because the R pointer of the RLD item in input module 2 refers to an SD entry in the ESD in module 2, it is marked for relative relocation; therefore, the relative relocation factor for PETE (+200) is added to the value (+300) assigned by the language translator. The relocated value for both address constants is 510.

Relocation of all nonbranch-type address constants requires an addition or subtraction of the relocation factor to or from the value of the address constant in the text of the input module. (Addition or subtraction is specified in the flag field of the RLD item for the address constant.)

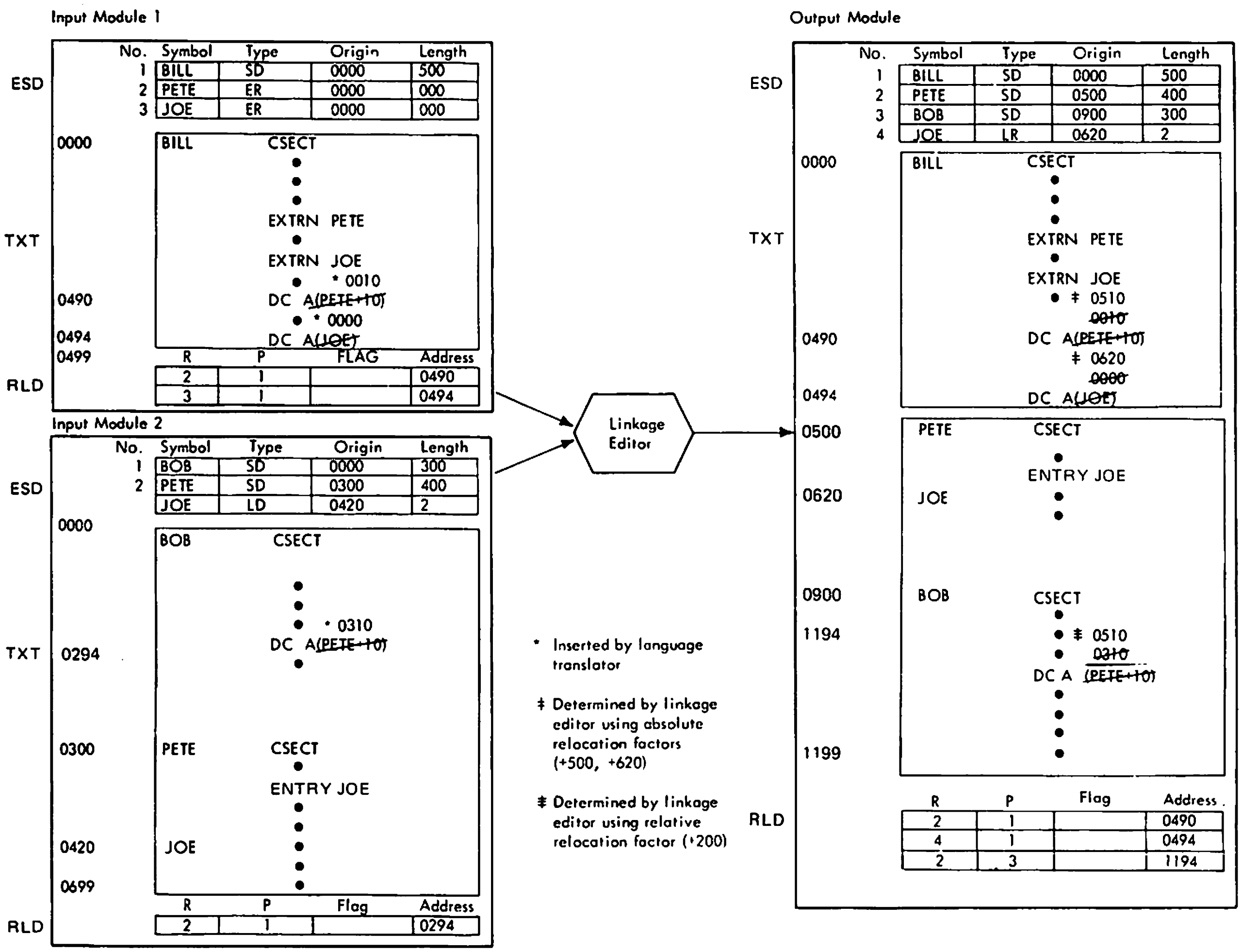
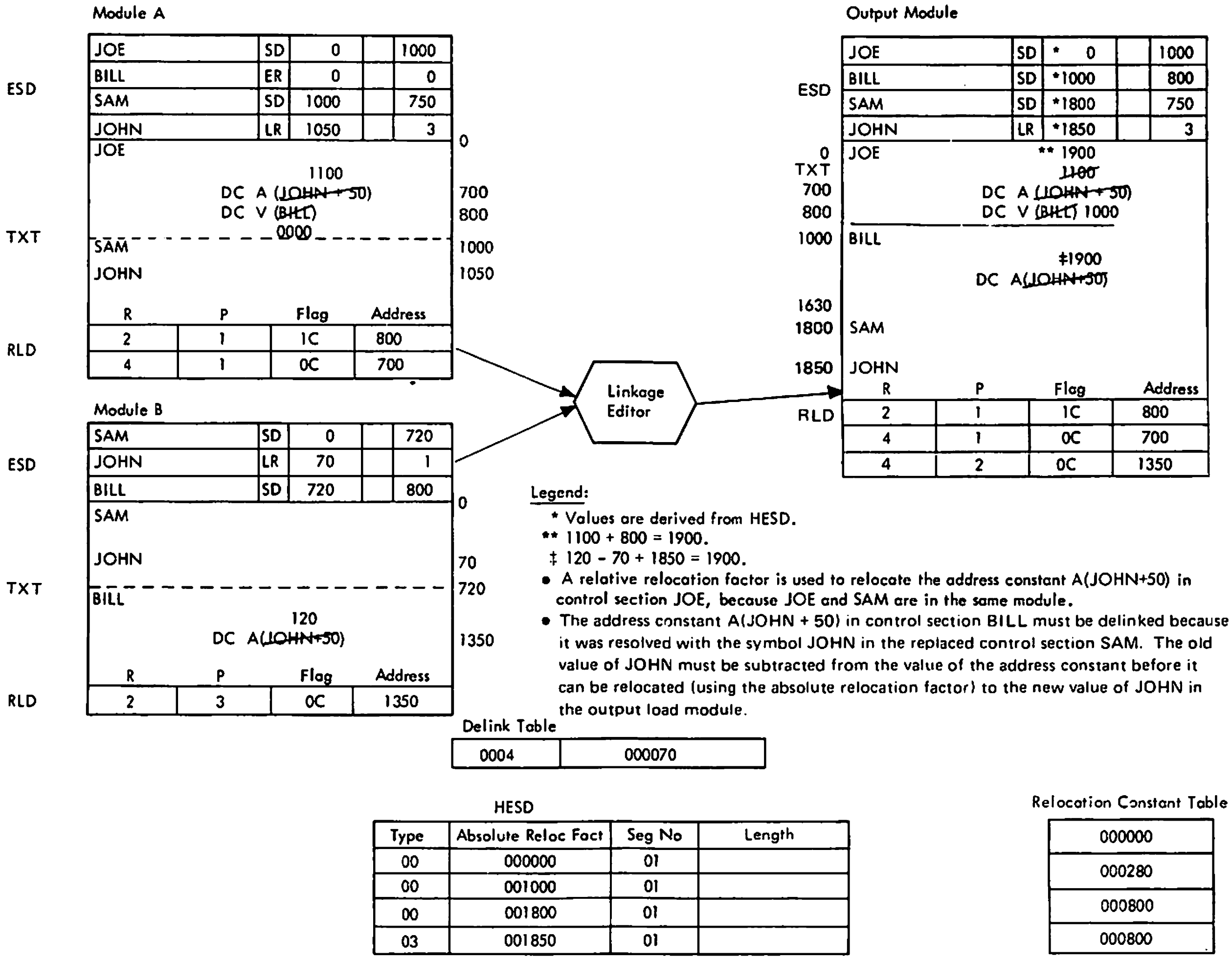


Figure 24. Nonbranch-Type Address Constants—Absolute and Relative Relocation

DELINKING NONBRANCH-TYPE ADDRESS CONSTANTS: A relative relocation factor cannot be used to relocate an A-type address constant that refers to a symbol in a control section being replaced. Because the address constant has been previously relocated (by a language translator or by the linkage editor), it contains the value of a symbol being replaced; therefore, the value of that symbol must be subtracted from the value of the address constant. This process is called delinking. In delinking, an address constant is reduced to the value it would have contained if it referred to an external reference in the input module. After delinking, the address constant contains the value required for proper relocation, should the replaced symbol appear later in the input in another control section. Delinked address constants are treated as address constants whose values depend on external references. (Absolute relocation factors are used in relocating them.)

Delinking of an A-type address constant is shown in Figure 25. Input load modules A and B both contain control section SAM. During linkage editor processing, the first occurrence of control section SAM is accepted, while the second occurrence is deleted through automatic control section replacement.



Module A — ESD

JOE	SD	0		1000
BILL	ER	0		0
SAM	SD	1000		750
JOHN	LR	1050		3

Module A — RLD

R	P	Flag	Address
2	1	1C	800
4	1	0C	700

Module B — ESD

SAM	SD	0		720
JOHN	LR	70		1
BILL	SD	720		800

Module B — RLD

R	P	Flag	Address
2	3	0C	1350

Output Module — ESD

JOE	SD	* 0		1000
BILL	SD	*1000		800
SAM	SD	*1800		750
JOHN	LR	*1850		3

Output Module — RLD

R	P	Flag	Address
2	1	1C	800
4	1	0C	700
4	2	0C	1350

Legend:

* Values are derived from HESD.
** 1100 + 800 = 1900.
‡ 120 - 70 + 1850 = 1900.

- A relative relocation factor is used to relocate the address constant A(JOHN+50) in control section JOE, because JOE and SAM are in the same module.
- The address constant A(JOHN + 50) in control section BILL must be delinked because it was resolved with the symbol JOHN in the replaced control section SAM. The old value of JOHN must be subtracted from the value of the address constant before it can be relocated (using the absolute relocation factor) to the new value of JOHN in the output load module.

Delink Table

0004	000070

HESD

Type	Absolute Reloc Fact	Seg No	Length
00	000000	01	
00	001000	01	
00	001800	01	
03	001850	01	

Relocation Constant Table

000000
000280
000800
000800

Figure 25. Example of Delinking

Control section BILL in module B contains a reference to symbol JOHN in control section SAM. Because SAM in module B will be deleted, the address constant A (JOHN+50) in module B must be delinked so that it may be properly resolved with the symbol JOHN in module A. In delinking, the old value of JOHN is subtracted from the value of the address constant in BILL (120-70=50). The absolute relocation factor for JOHN (1850) is then added to the delinked value of JOHN (50+1850=1900).

DELINKING COMMON CONTROL SECTIONS: Common control sections (either blank common or named common) must be "delinked" by the linkage editor. All references to common control sections are made by means of nonbranch-type address constants.

If the assigned address of a common control section in the input to the linkage editor is not zero, all such references must be delinked. Delinking is necessary, because during linkage editor processing all blank common control sections are collected into a single control section. All identically named common control sections are gathered into individual control sections; references to them from different input modules must be delinked so that they can be properly relocated with respect to the locations of the common control sections in the output module.

Delinking adjusts the value of each address constant in a common control section so that it contains its correct displacement from the control section origin. The values of such address constants are then relocated so that they refer to the linkage editor-assigned addresses, using absolute relocation factors.

Relocation of Branch-Type (V-Type) Address Constants

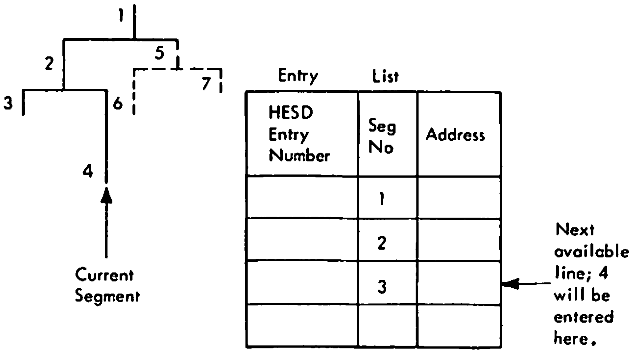
Only absolute relocation factors are used to relocate branch-type address constants. Because a displacement is not allowed in the value of a V-type address constant, the absolute relocation factor is inserted in the value field during relocation. (It is not added to or subtracted from in value assigned by the language translator, as described for A-type address constants.) Because the value of a V-type address constant is inserted, delinking is never necessary for such address constants. Relocation of V-type address constants in an overlay structure is discussed in the following paragraph.

RELOCATION OF V-TYPE ADDRESS CONSTANTS IN OVERLAY: If the output of the linkage editor is to be overlay load module, a 4-byte[11] branch-type address constant in the path of the symbol it refers to (but in a different segment), or in a different region, will be relocated in a special manner. The value field of the address constant will contain the address of an ENTAB entry. The ENTAB entry will contain the address assigned by the linkage editor to the symbol referred to by the value of the address constant. An ENTAB entry is created for each V-type address constant that is in the path of the symbol it refers to (but is not in the same segment), or located in a different region, provided that the symbol is not referred to in a segment higher in the path of the calling segment. (Such address constants are resolved so that they refer to the ENTAB entry previously created for the symbol in the higher segment.) ENTAB entries are not created for address constants that refer to symbols higher in the path. Whenever an ENTAB entry is created, it is noted in an entry list; each item in the entry list contains the entry number of the referenced symbol in the HESD, the segment number of the calling segment, and the address assigned to the ENTAB entry by the linkage editor. The ENTAB

[11] Any address constant must be 4 bytes, because the high-order byte is used by the overlay supervisor during execution. The number of the segment containing the address constant will be placed in the high-order byte of any V-type address constant resolved to an ENTAB entry. (The high-order byte must be zero if it is not resolved to an ENTAB entry.)

creation routine uses the entry list to build ENTAB entries (see "ENTAB Creation").

When second pass processing begins to process a segment, the entry list is modified so that is contains only entries for segments higher in the path of the current segment. (In Figure 26, segment 4 is being processed; the entry for segment 3 is removed because it is not higher in the path of segment 4.)



Entry List

HESD Entry Number	Seg No	Address
	1	
	2	
	3	

Figure 26. Entry List Processing

During relocation, each V-type address constant is examined to determine if an ENTAB entry must be created for it. The R pointer of the RLD item for the address constant is used to find the associated HESD entry; this entry contains the segment number of the symbol referred to by the address constant. The relationship of this segment to the current segment is then determined, using SEGTA1. Depending on the relationship in SEGTA1, the address constant is relocated in one of three ways:

1. If the segment that contains the symbol is higher in the path of the current segment, the call is upward and the address constant is resolved directly. (The absolute relocation factor of the symbol is inserted in the value of the address constant.)

2. If the current segment is higher in the path of the segment that contains the symbol, the call is downward. The entry list is checked to determine if an ENTAB entry was previously created for the symbol in this segment, or in a segment higher in the path of this segment. If an ENTAB entry for the symbol exists, its address (contained in the entry list) is placed in the value field of the address constant. If no ENTAB entry exists for the symbol, a new entry is placed in the entry list, and an ENTAB entry will be created by the ENTAB creation routine (see "ENTAB Creation"). The ENTAB entry will contain the address assigned to the symbol by the linkage editor, and the address of the ENTAB entry will be placed in the value of the address constant and in the entry list item.

3. If neither of the two segments is higher in the path of the other, the call is either exclusive or across regions. If the two segments are in different regions, and no ENTAB entry already exists for the symbol in the entry list, an ENTAB entry will be created and an entry is made in the entry list; the value field of the address constant is relocated to the address of the ENTAB entry, which in turn contains the relocated address of the symbol. If the two segments are in the same region, the call is exclusive. If there is an entry in the entry list for the symbol, the address constant is resolved through its ENTAB entry; if there is no entry for the symbol in the entry list, the call is an invalid exclusive call and the address constant is

resolved directly to the symbol. (This usually leads to incorrect results during execution of the module.)

ENTAB Creation

The ENTAB creation routine uses the size field in the HESD to determine the number of ENTAB entries to be created for a given segment. The entry list is scanned for all entries that were created for the current segment; each of these entries contains the HESD entry number for the corresponding symbol. The value and segment number of the symbol are obtained from the HESD and are entered in the ENTAB entry, along with standard information shown in the table format (see "Table Layouts").

ENTAB creation is shown in Figure 28 on page 70. The V-type address constants referring to SAM and BILL in segment 1 meet the requirements for building ENTAB entries. The ESD and RLD input to the second pass processor, and the overlay tree structure are shown in Diagram A. During relocation, entries are created for SAM and BILL in the entry list (see Diagram B); each entry contains the address of the ENTAB entry created for the address constant.

In segment 1, location 136 of control section JOE contained a call to control section SAM before relocation. After relocation, location 136 contains the address of the ENTAB entry for SAM, and the high-order byte of the address constant contains the segment number of the calling segment. An ENTAB entry is created, in like manner, for BILL in segment 1.

In segment 2, the address constant referring to BILL does not meet the requirements for building an ENTAB entry. (It is not in the path of the segment containing the symbol.) Therefore, no ENTAB is created in segment 2. The call for segment 2 to BILL in segment 3 is an exclusive call. Because a call to the same symbol appears in a higher segment common to 2 and 3 (segment 1), the address constant may refer to the ENTAB entry for BILL in segment 1. (This is determined by scanning the entry list for the HESD entry corresponding to the symbol BILL.)

If a call to BILL was not contained in a common segment, the address constant DC V (BILL) in segment 2 would be resolved using the value assigned by the linkage editor to the symbol BILL, which results in an error.

In segment 3, the address constant is an upward call and is resolved directly.

Relocation Routine

The relocation of address constants is performed by the relocation routine; the routine operates on the following input data:

- The address of the RLD input buffers that contain RLD records
- The address of the RLD note list entry for the RLDs being processed
- The address of the next available entry in the RLD output buffer
- The buffer relocation constant (BRC) where:

  BRC = starting buffer address of current text + relative relocation constant of current control section - address assigned to current control section by the linkage editor - multiplicity size x current multiplicity number

Input			Output	
Flag	Type	Action Performed	Flag	Type
0000LLST	Absolute	Absolute relocation factor is added to value of address constant	0000LLST	A-type
0001LLST	Branch	Absolute relocation factor is inserted into value of address constant	0001LLST	V-type
0010LLST	PR displacement value (PR type 1)	Absolute relocation factor is inserted into value of address constant	0010LLST	PR displacement value
0010LLST	PR cumulative displacement value (PR type 2)	PR length from all purpose table is inserted into value of address constant	0011LLST	PR cumulative displacement value
1000LLST	Relative	Relative relocation factor is added to value of address constant	0000LLST	A-type

Figure 27. Relationship of RLD Flag Field to Relocation

Notes to Figure 27 on page 68:

1. If S (sign) in LIST is 1, subtraction is performed, rather than addition.

2. In delink type, the delink value is added or subtracted according to the opposite of the sign; the absolute relocation factor is added to or subtracted from the address constant according to the indicated sign.

3. If an RLD item refers to an undefined symbol, the associated address constant is not relocated. (It may have been delinked.) The high-order bit of the RLD item flag field is set to one (1000LLST for an A-type constant, 1001LLST for a V-type constant), and no relocation will be performed when the module is loaded into virtual storage for execution.

4. Delinking is noted in the high-order bit of the P pointer.

Diagram A.

HESD

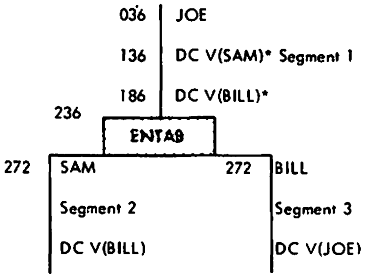
	Type	L.E. Assigned Address	Seg	Length
JOE	SD	36	1	
SAM	SD	272	2	
BILL	SD	272	3	
SEGTAB	PC	0	1	
ENTAB	PC	236	1	

Relocation Constant Table
200
500
500
36
36

RLD

R	P	Flag	Address
2	1	1C	100
3	1	1C	150

Input RLDs – Segment 1



Structure with V-type address Constants.

* Zero value assigned by the assembler.

Diagram B.

Output RLD Buffer

2	1	1C	136
3	1	1C	186

Entry List

2	1	236
3	1	248

Entab RLD Items

0	1	1D	240
		1D	252

RLDs and Entry List after relocation for control section JOE.

Diagram C.

Segment 1 after processing by Second Pass Processor.

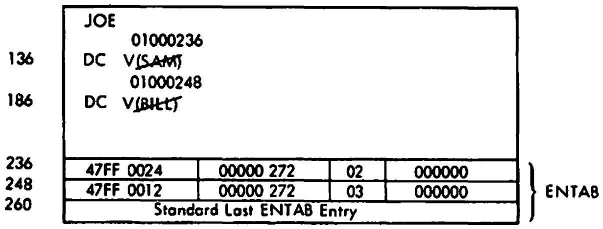
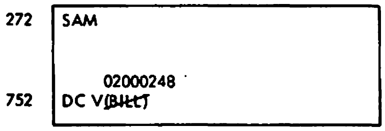


Diagram D.

Segment 2 after processing by Second Pass Processor.



Input RLD Buffer

3	2	1C	680

Output RLD Buffer

3	2	1C	752

ENTAB RLD Items

None

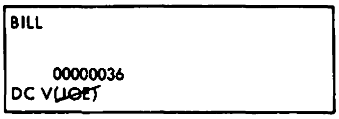
Entry List

*

* Same as after processing segment 1.

Diagram E.

Segment 3 after Second Pass Processing



Input RLD Buffer

1	3	1C	690

Output RLD Buffer

1	3	1C	762

ENTAB RLD Items

None

Entry List

*

* Same as after processing segment 1

Figure 28. ENTAB Creation

The relocation routine operates in the following manner:

1. The size of the RLD set[12] and the displacement from the beginning of the buffer are determined from the RLD note list.

2. Each RLD item in the current RLD set is scanned to determine whether:

   a. It describes an address constant for the current text being processed (BRC + address contained in the RLD address field falls within the text buffer boundaries of the current text.)

   b. The address constant is either a valid 2-, 3-, or 4-byte address constant. (The only valid 2-byte address constants are defined by pseudo register symbols.)

3. Each address constant whose RLD meets the above requirements is moved from the text into a computation area. The address constant associated with the RLD item is then relocated according to the information in the flag field of the RLD item (see Figure 27 on page 68). In relocating 4-byte address constants (VCONs), the high-order bit in the address constant before relocation is reproduced in the address constant after relocation. The relocated address constant is then placed back into the text.

4. The RLD address field is updated using the relative relocation factor for the control section being processed. (The control section referred to by the P pointer of the RLD item).

5. The RLD is moved into the RLD output buffer if space is available. If space is not available, the contents of the RLD output buffer are written out on SYSLMOD. See "Second Pass RLD Buffers" under "Second Pass Processing."

6. Steps 2 through 5 are repeated until all RLD items have been scanned in the RLD set being processed. The multiplicity number in the RLD note list is updated if unprocessed RLDs remain in the set.

7. If there are more RLD sets in the input buffer to be processed, the address of the next record is determined and steps 1 through 6 are performed.

Note: To minimize the number of times RLD records are read from SYSUT1, RLD records for a control section are held in the input RLD buffer, when possible, until all RLD records in the buffer have been processed (because each RLD record may pertain to many multiplicities of text). After each set of RLDs is scanned, the multiplicity number in the RLD note list is updated to reflect the multiplicity of the remaining unprocessed RLD records in the set. An RLD record is removed from the buffer when:

- All RLD items in the record have been processed. (Their associated address constants have been relocated.)

- Another RLD record must be read into the buffer and space is not available.

When all records in the input RLD buffer have been scanned, the relocation routine determines if more RLD records for the current multiplicity of text are to be read in. (The RLD read routine sets an indicator when it encounters such a record but cannot read it into the buffer because the buffer is full.) When both buffers are full, the second buffer is freed, and a bit is set in the corresponding RLD note list entries which

[12] An RLD set is a group of RLDs referred to by a particular RLD note list entry.

indicates that the RLDs are not in virtual storage. The records to be read in are then placed in the second RLD buffer; these records are processed in the same manner as those already residing in the first buffer. This process is repeated until all records that contain RLD items pertaining to the current multiplicity of text have been scanned and processed.

When all RLDs in a buffer are processed, the buffer is marked "free" in the RLD control block. When a new multiplicity of text is to be relocated, the RLD note list is scanned sequentially (on ID and multiplicity number) from the first entry. If an entry indicates that the record is "in virtual storage" and the record contains RLD items pertaining to the new multiplicity of text, it is processed.

FINAL PROCESSING (HEWLFFNL)

Final processing comprises modules HEWLFFNL, HEWLFBTP, and (optionally) HEWLFMAP.

The fourth phase of the linkage editor (final processing) performs "cleanup" functions, and is the last operation of the linkage editor processing. Functions of final processing include:

- Writing the TTR note list, created during second pass processing, on SYSLMOD if the output load module is to be used in overlay. The TTR list contains the relative track address of the first record of each segment of the overlay load module. It is used by the overlay supervisor to find the segments when it loads them into virtual storage for execution.
- Placing each entry in the proper format for the partitioned data set directory, modifying it if there are alias symbols, and issuing a STOW macro instruction[13] for the member name and each alias.
- Checking attributes (reusable, reenterable, and refreshable). If the attributes have become more restrictive, a message describing the change in attributes is printed out. (For example, the input module was specified as "reusable" and is now "not reusable.")
- Printing out a directory of logged errors.
- Producing a MAP or a cross-reference table if the MAP or XREF option is specified, and the MAP or cross-reference table was not produced during intermediate output processing. If the MAP alternate 2-byte table is too small, warning message IEW0801 is issued.
- Printing a diagnostic message if the module has been marked "not executable."
- Reinitiating linkage editor processing, beginning with initialization, if a NAME statement terminated SYSLIN input.
- Completing linkage editor processing if end-of-file terminated SYSLIN input; releasing virtual storage and returning control to the caller.

[13] The STOW macro instruction is not issued if there was no valid input, if there were no ESDs, if nothing was written out on SYSLMOD, or if the run was terminated by a severity 4 error.

Error Logging

Whenever an error condition is detected during linkage editor processing, an indicator is set in an error logging map and a coded diagnostic message is printed out. During final processing, the error logging map is scanned. When an indicator is found "on" in the map, an associated list is used to build a diagnostic message.

Note: An example of error logging is given in Figure 29. Each entry in the list contains a length indicator and a pointer to a phrase to be assembled into the message. (Phrases are stored to save virtual storage space; complete messages would require additional space because of repetition of identical phrases.) The diagnostic directory is then printed out, one or two lines to a message. This directory is normally directed to the SYSPRINT data set. However, if the TERM option was specified, diagnostic messages are directed to both the SYSPRINT and SYSTERM data sets.

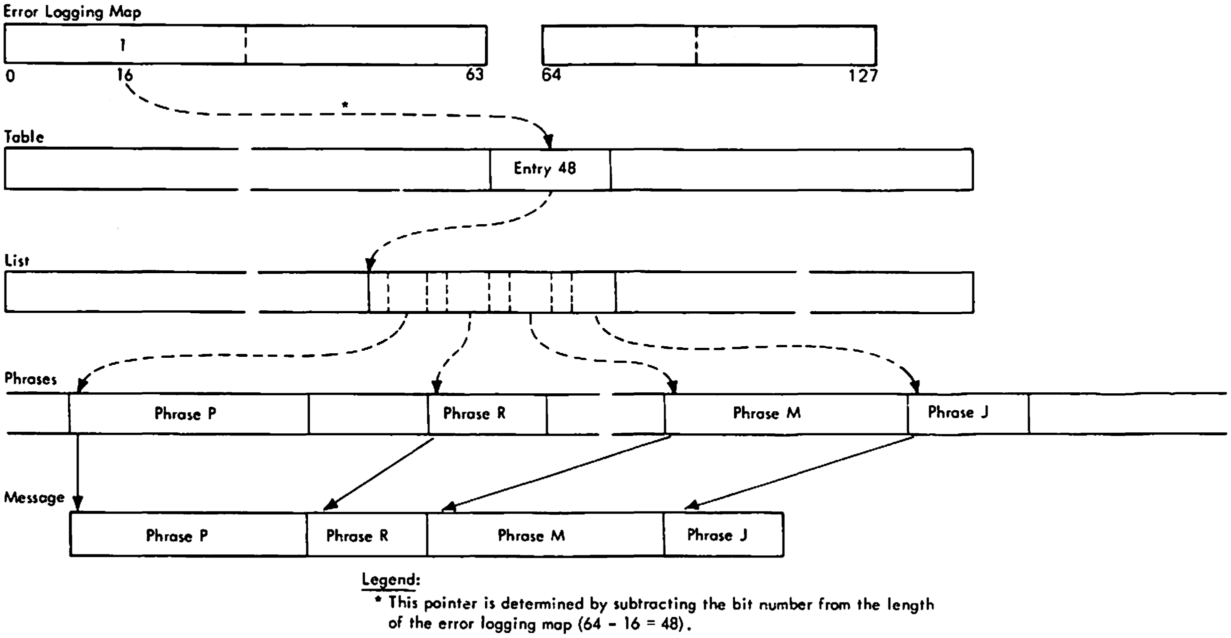


Figure 29. Building Error Messages

All error messages produced by the linkage editor are identified by a message ID having the format:

IEWDMMS

where:

IEW identifies the message as a linkage editor error message.

D contains a zero.

MM is the message number.

S is the severity code.

The module in which an error message occurred is identified by the message number (MM; see Figure 68 on page 189).

Cross-Reference Table

If the XREF option is specified and the cross-reference table was not produced during intermediate output processing, the RLD records are read back from SYSLMOD and the cross-reference table is built, as described in the discussion of intermediate processing.

DIAGRAM 1. OVERVIEW OF LINKAGE EDITOR

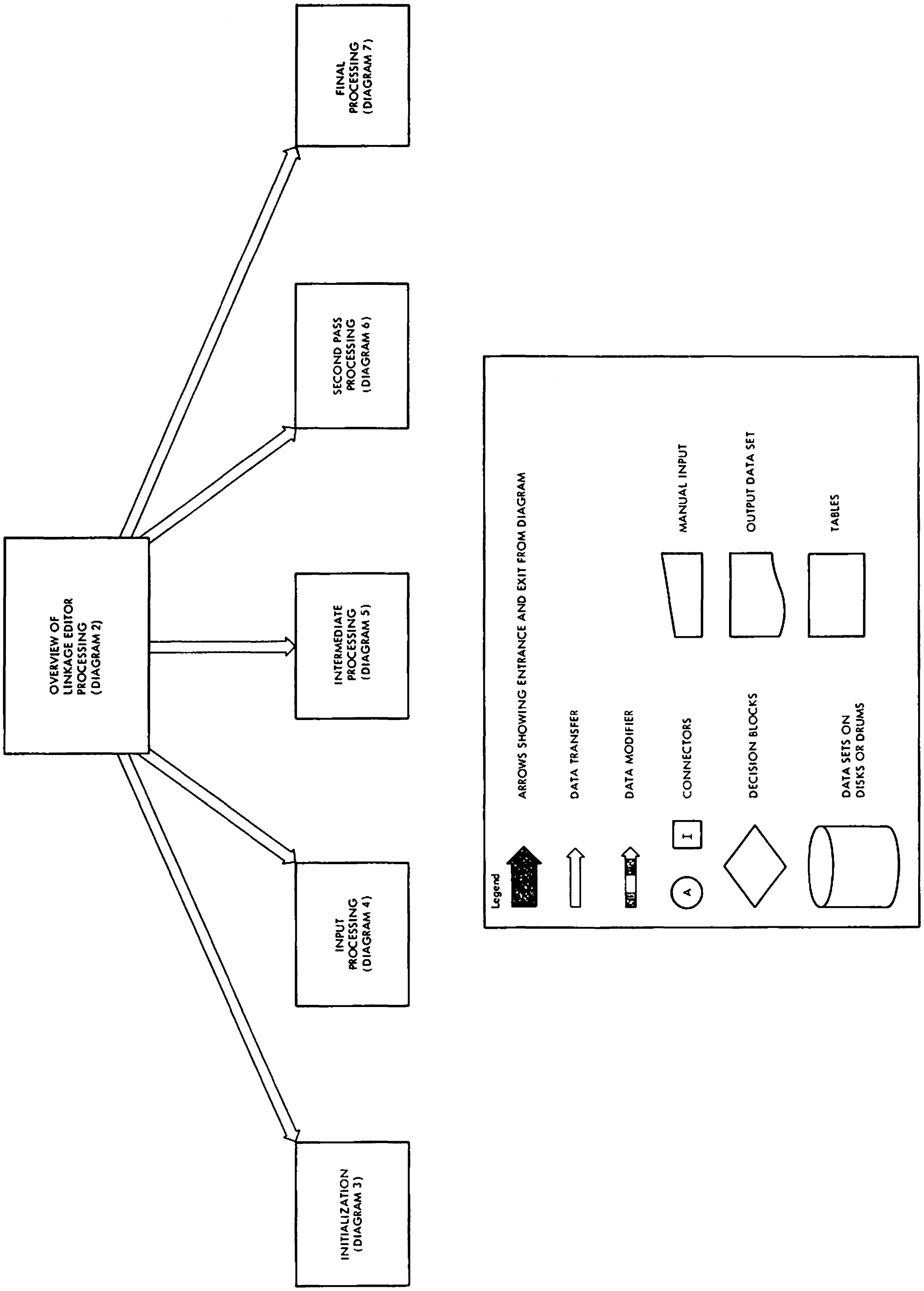
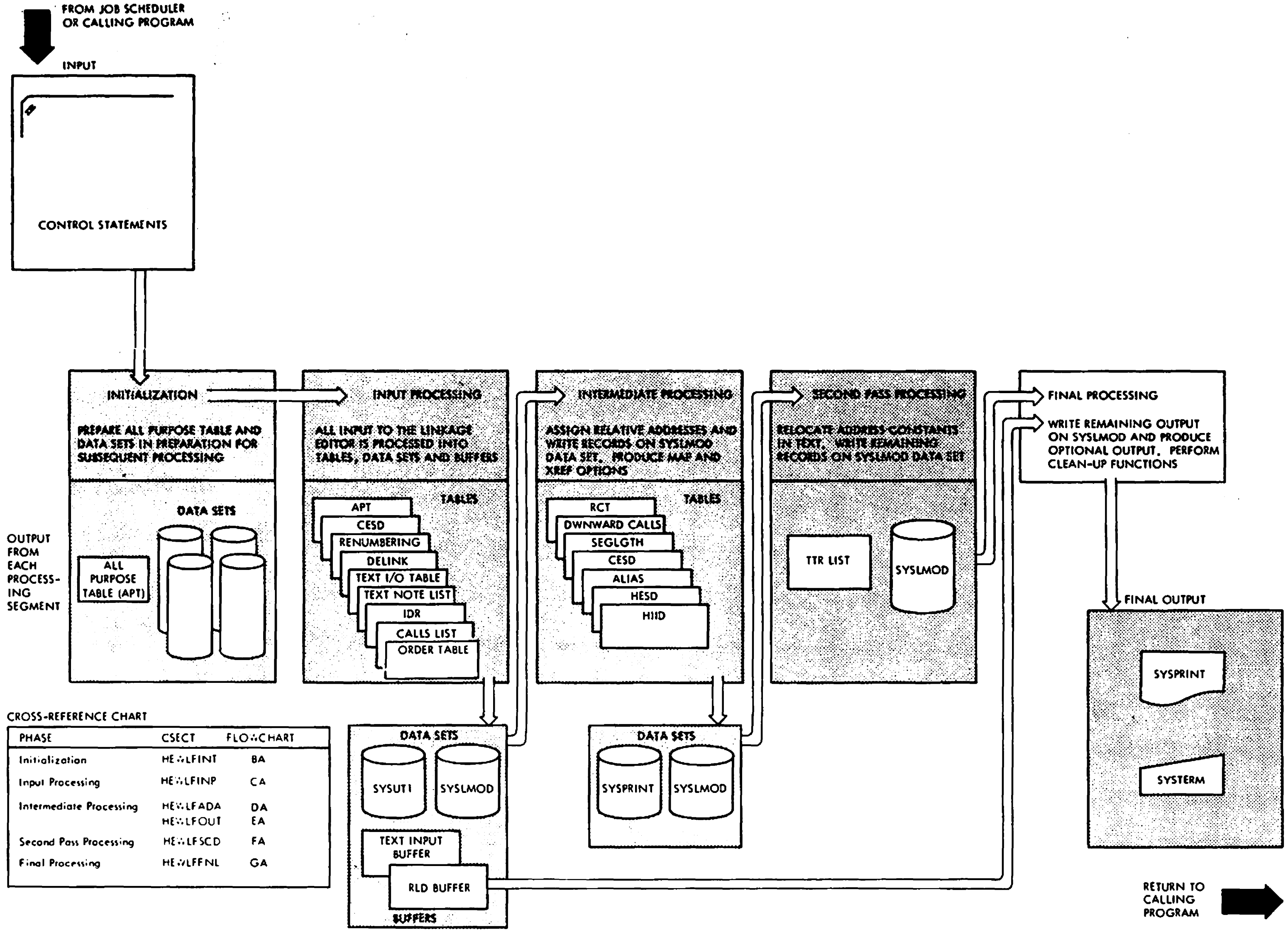

OVERVIEW OF LINKAGE EDITOR PROCESSING (DIAGRAM 2)
INITIALIZATION (DIAGRAM 3)
INPUT PROCESSING (DIAGRAM 4)
INTERMEDIATE PROCESSING (DIAGRAM 5)
SECOND PASS PROCESSING (DIAGRAM 6)
FINAL PROCESSING (DIAGRAM 7)
Legend
ARROWS SHOWING ENTRANCE AND EXIT FROM DIAGRAM
DATA TRANSFER
DATA MODIFIER
A
I
CONNECTORS
MANUAL INPUT
DECISION BLOCKS
OUTPUT DATA SET
DATA SETS ON DISKS OR DRUMS
TABLES

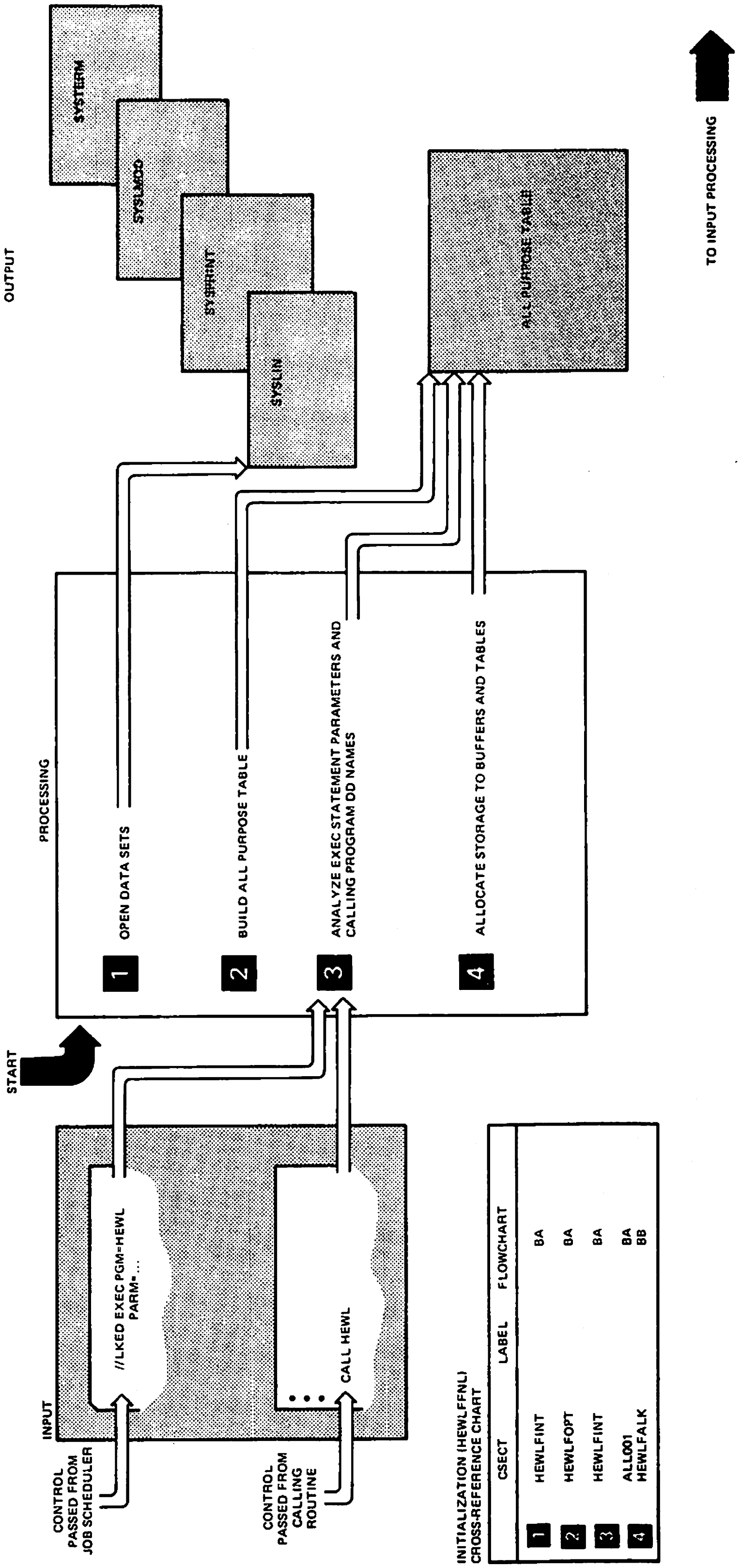
DIAGRAM 2. DETAILED OVERVIEW OF LINKAGE EDITOR PROCESSING


FROM JOB SCHEDULER OR CALLING PROGRAM
INPUT
CONTROL STATEMENTS
OUTPUT FROM EACH PROCESSING SEGMENT
INITIALIZATION
PREPARE ALL PURPOSE TABLE AND DATA SETS IN PREPARATION FOR SUBSEQUENT PROCESSING
DATA SETS
ALL PURPOSE TABLE (APT)
INPUT PROCESSING
ALL INPUT TO THE LINKAGE EDITOR IS PROCESSED INTO TABLES, DATA SETS AND BUFFERS
TABLES
APT
CESD
RENUMBERING
DELINK
TEXT I/O TABLE
TEXT NOTE LIST
IDR
CALLS LIST
ORDER TABLE
DATA SETS
SYSUT1
SYSLMOD
TEXT INPUT BUFFER
RLD BUFFER
BUFFERS
INTERMEDIATE PROCESSING
ASSIGN RELATIVE ADDRESSES AND WRITE RECORDS ON SYSLMOD DATA SET. PRODUCE MAP AND XREF OPTIONS
TABLES
RCT
DWNWARD CALLS
SEGLGTH
CESD
ALIAS
HESD
HIID
DATA SETS
SYSPRINT
SYSLMOD
SECOND PASS PROCESSING
RELOCATE ADDRESS CONSTANTS IN TEXT. WRITE REMAINING RECORDS ON SYSLMOD DATA SET
TTR LIST
SYSLMOD
FINAL PROCESSING
WRITE REMAINING OUTPUT ON SYSLMOD AND PRODUCE OPTIONAL OUTPUT. PERFORM CLEAN-UP FUNCTIONS
FINAL OUTPUT
SYSPRINT
SYSTERM
RETURN TO CALLING PROGRAM

CROSS-REFERENCE CHART

PHASE	CSECT	FLOWCHART
Initialization	HEWLFINT	BA
Input Processing	HEWLFINP	CA
Intermediate Processing	HEWLFADA HEWLFOUT	DA EA
Second Pass Processing	HEWLFSCD	FA
Final Processing	HEWLFFNL	GA

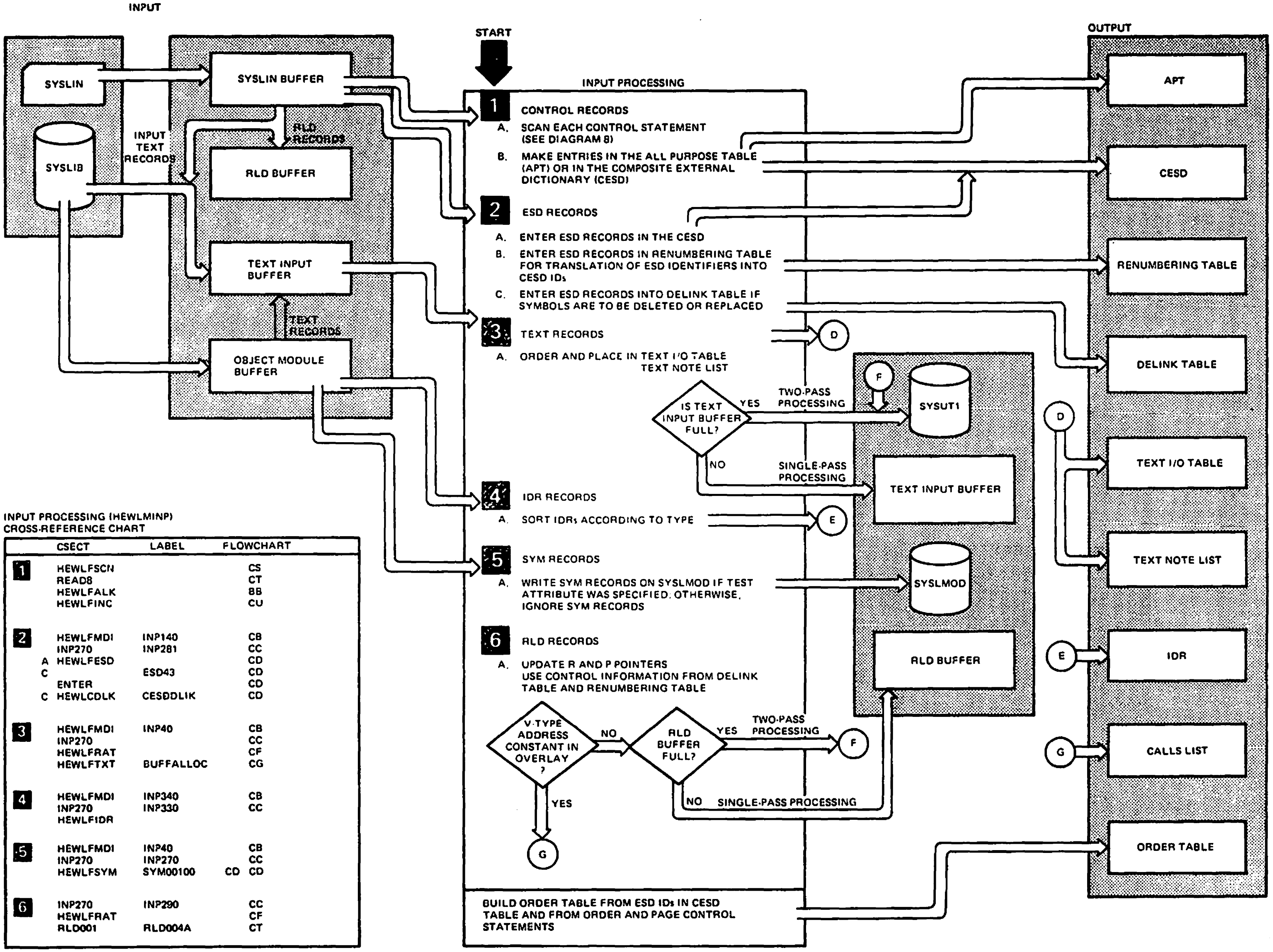
DIAGRAM 3. INITIALIZATION


INPUT
CONTROL PASSED FROM JOB SCHEDULER
//LKED EXEC PGM=HEWL PARM=...
CONTROL PASSED FROM CALLING ROUTINE
CALL HEWL
START
PROCESSING
1 OPEN DATA SETS
2 BUILD ALL PURPOSE TABLE
3 ANALYZE EXEC STATEMENT PARAMETERS AND CALLING PROGRAM DD NAMES
4 ALLOCATE STORAGE TO BUFFERS AND TABLES
OUTPUT
SYSTERM
SYSLMOD
SYSPRINT
SYSLIN
ALL PURPOSE TABLE
TO INPUT PROCESSING

INITIALIZATION (HEWLFFNL)
CROSS-REFERENCE CHART

	CSECT	LABEL	FLOWCHART
1	HEWLFINT		BA
2	HEWLFOPT		BA
3	HEWLFINT		BA
4	ALL001 HEWLFALK		BA BB

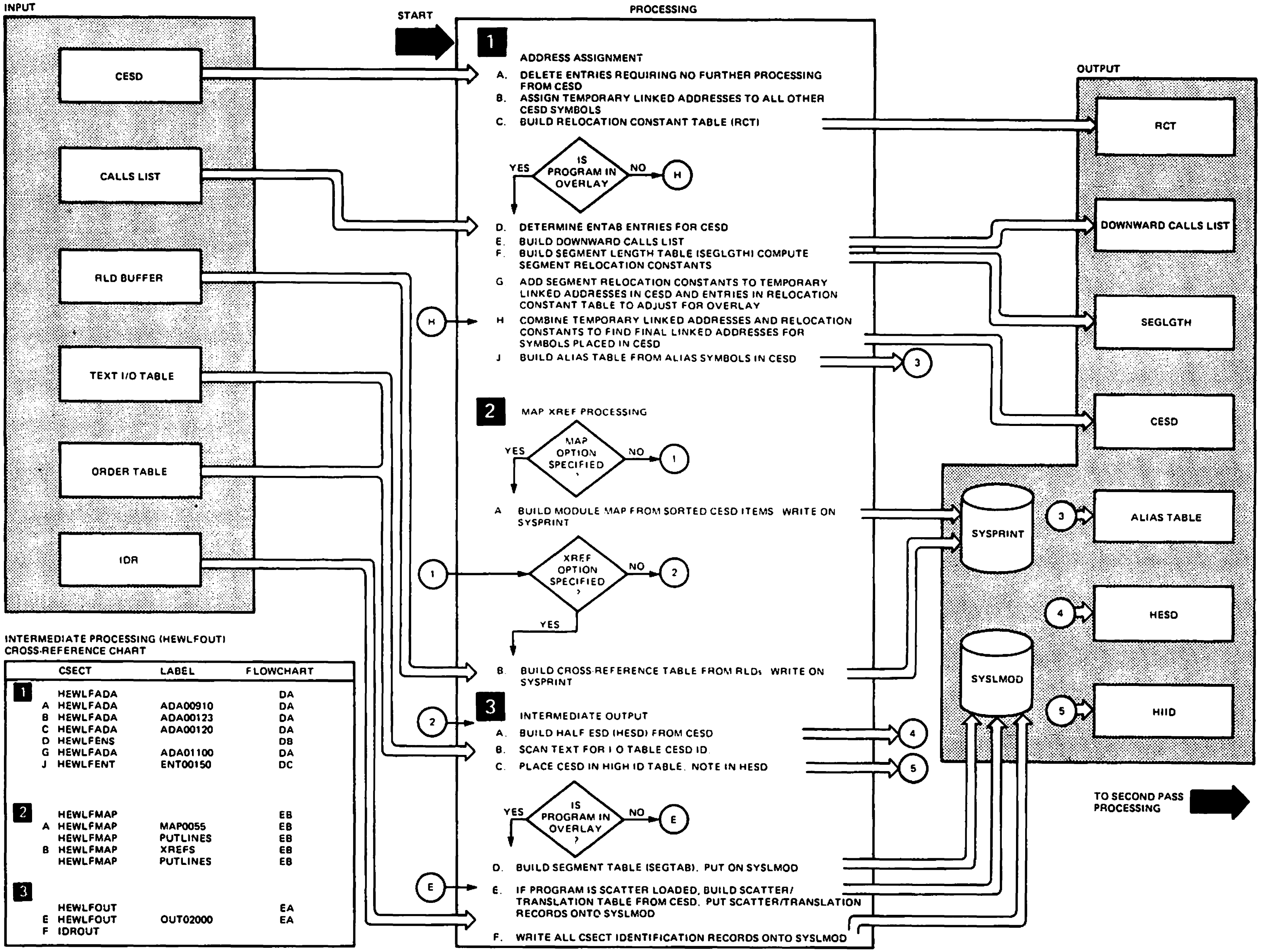
DIAGRAM 4. INPUT PROCESSING


INPUT
SYSLIN
SYSLIB
SYSLIN BUFFER
INPUT TEXT RECORDS
RLD RECORDS
RLD BUFFER
TEXT INPUT BUFFER
TEXT RECORDS
OBJECT MODULE BUFFER
START
INPUT PROCESSING
1 CONTROL RECORDS
A. SCAN EACH CONTROL STATEMENT (SEE DIAGRAM 8)
B. MAKE ENTRIES IN THE ALL PURPOSE TABLE (APT) OR IN THE COMPOSITE EXTERNAL DICTIONARY (CESD)
2 ESD RECORDS
A. ENTER ESD RECORDS IN THE CESD
B. ENTER ESD RECORDS IN RENUMBERING TABLE FOR TRANSLATION OF ESD IDENTIFIERS INTO CESD IDs
C. ENTER ESD RECORDS INTO DELINK TABLE IF SYMBOLS ARE TO BE DELETED OR REPLACED
3 TEXT RECORDS
A. ORDER AND PLACE IN TEXT I/O TABLE TEXT NOTE LIST
D
IS TEXT INPUT BUFFER FULL?
YES
TWO-PASS PROCESSING
NO
SINGLE-PASS PROCESSING
F
SYSUT1
TEXT INPUT BUFFER
4 IDR RECORDS
A. SORT IDRs ACCORDING TO TYPE
E
5 SYM RECORDS
A. WRITE SYM RECORDS ON SYSLMOD IF TEST ATTRIBUTE WAS SPECIFIED. OTHERWISE, IGNORE SYM RECORDS
SYSLMOD
6 RLD RECORDS
A. UPDATE R AND P POINTERS USE CONTROL INFORMATION FROM DELINK TABLE AND RENUMBERING TABLE
RLD BUFFER
V-TYPE ADDRESS CONSTANT IN OVERLAY ?
NO
RLD BUFFER FULL?
YES
TWO-PASS PROCESSING
F
NO
SINGLE-PASS PROCESSING
YES
G
BUILD ORDER TABLE FROM ESD IDs IN CESD TABLE AND FROM ORDER AND PAGE CONTROL STATEMENTS
OUTPUT
APT
CESD
RENUMBERING TABLE
DELINK TABLE
D
TEXT I/O TABLE
TEXT NOTE LIST
E
IDR
G
CALLS LIST
ORDER TABLE

INPUT PROCESSING (HEWLMINP)
CROSS-REFERENCE CHART

		CSECT	LABEL		FLOWCHART
1		HEWLFSCN			CS
		READ8			CT
		HEWLFALK			BB
		HEWLFINC			CU
2		HEWLFMDI	INP140		CB
		INP270	INP281		CC
	A	HEWLFESD			CD
	C		ESD43		CD
		ENTER			CD
	C	HEWLCDLK	CESDDLIK		CD
3		HEWLFMDI	INP40		CB
		INP270			CC
		HEWLFRAT			CF
		HEWLFTXT	BUFFALLOC		CG
4		HEWLFMDI	INP340		CB
		INP270	INP330		CC
		HEWLFIDR			
5		HEWLFMDI	INP40		CB
		INP270	INP270		CC
		HEWLFSYM	SYM00100	CD	CD
6		INP270	INP290		CC
		HEWLFRAT			CF
		RLD001	RLD004A		CT

DIAGRAM 5. INTERMEDIATE PROCESSING



1 ADDRESS ASSIGNMENT

A. DELETE ENTRIES REQUIRING NO FURTHER PROCESSING FROM CESD
B. ASSIGN TEMPORARY LINKED ADDRESSES TO ALL OTHER CESD SYMBOLS
C. BUILD RELOCATION CONSTANT TABLE (RCT)

IS PROGRAM IN OVERLAY

D. DETERMINE ENTAB ENTRIES FOR CESD
E. BUILD DOWNWARD CALLS LIST
F. BUILD SEGMENT LENGTH TABLE (SEGLGTH) COMPUTE SEGMENT RELOCATION CONSTANTS
G ADD SEGMENT RELOCATION CONSTANTS TO TEMPORARY LINKED ADDRESSES IN CESD AND ENTRIES IN RELOCATION CONSTANT TABLE TO ADJUST FOR OVERLAY
H COMBINE TEMPORARY LINKED ADDRESSES AND RELOCATION CONSTANTS TO FIND FINAL LINKED ADDRESSES FOR SYMBOLS PLACED IN CESD
J BUILD ALIAS TABLE FROM ALIAS SYMBOLS IN CESD

2 MAP XREF PROCESSING

MAP OPTION SPECIFIED ?

A BUILD MODULE MAP FROM SORTED CESD ITEMS WRITE ON SYSPRINT

XREF OPTION SPECIFIED ?

B. BUILD CROSS-REFERENCE TABLE FROM RLDs WRITE ON SYSPRINT

3 INTERMEDIATE OUTPUT

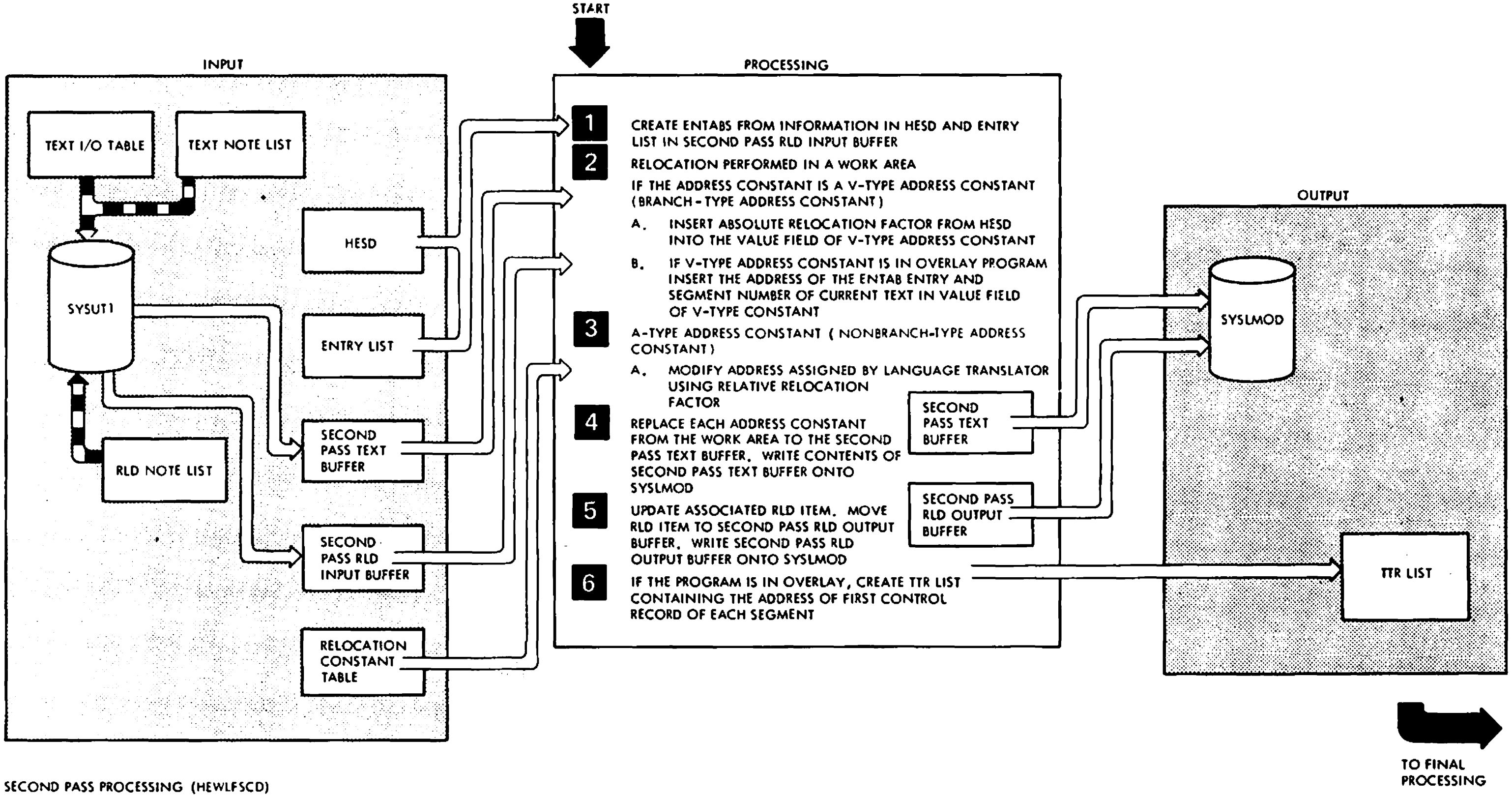
A. BUILD HALF ESD (HESD) FROM CESD
B. SCAN TEXT FOR I O TABLE CESD ID
C. PLACE CESD IN HIGH ID TABLE. NOTE IN HESD

IS PROGRAM IN OVERLAY ?

D. BUILD SEGMENT TABLE (SEGTAB). PUT ON SYSLMOD
E. IF PROGRAM IS SCATTER LOADED, BUILD SCATTER/TRANSLATION TABLE FROM CESD. PUT SCATTER/TRANSLATION RECORDS ONTO SYSLMOD
F. WRITE ALL CSECT IDENTIFICATION RECORDS ONTO SYSLMOD

INTERMEDIATE PROCESSING (HEWLFOUT) CROSS-REFERENCE CHART

		CSECT	LABEL	FLOWCHART
1		HEWLFADA		DA
	A	HEWLFADA	ADA00910	DA
	B	HEWLFADA	ADA00123	DA
	C	HEWLFADA	ADA00120	DA
	D	HEWLFENS		DB
	G	HEWLFADA	ADA01100	DA
	J	HEWLFENT	ENT00150	DC
2		HEWLFMAP		EB
	A	HEWLFMAP	MAP0055	EB
		HEWLFMAP	PUTLINES	EB
	B	HEWLFMAP	XREFS	EB
		HEWLFMAP	PUTLINES	EB
3		HEWLFOUT		EA
	E	HEWLFOUT	OUT02000	EA
	F	IDROUT		



1 CREATE ENTABS FROM INFORMATION IN HESD AND ENTRY LIST IN SECOND PASS RLD INPUT BUFFER

2 RELOCATION PERFORMED IN A WORK AREA

IF THE ADDRESS CONSTANT IS A V-TYPE ADDRESS CONSTANT (BRANCH-TYPE ADDRESS CONSTANT)

A. INSERT ABSOLUTE RELOCATION FACTOR FROM HESD INTO THE VALUE FIELD OF V-TYPE ADDRESS CONSTANT

B. IF V-TYPE ADDRESS CONSTANT IS IN OVERLAY PROGRAM INSERT THE ADDRESS OF THE ENTAB ENTRY AND SEGMENT NUMBER OF CURRENT TEXT IN VALUE FIELD OF V-TYPE CONSTANT

3 A-TYPE ADDRESS CONSTANT (NONBRANCH-TYPE ADDRESS CONSTANT)

A. MODIFY ADDRESS ASSIGNED BY LANGUAGE TRANSLATOR USING RELATIVE RELOCATION FACTOR

4 REPLACE EACH ADDRESS CONSTANT FROM THE WORK AREA TO THE SECOND PASS TEXT BUFFER. WRITE CONTENTS OF SECOND PASS TEXT BUFFER ONTO SYSLMOD

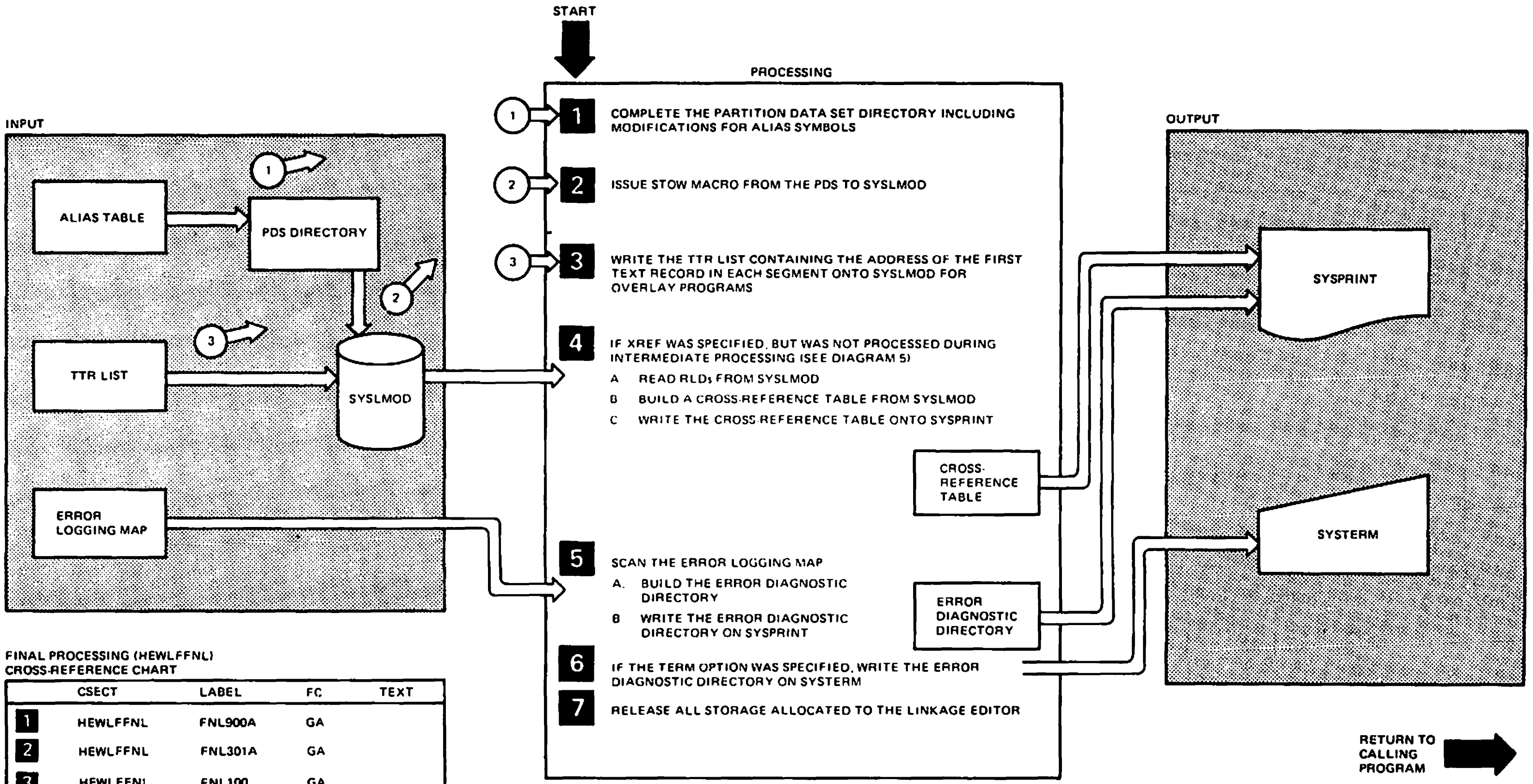
5 UPDATE ASSOCIATED RLD ITEM. MOVE RLD ITEM TO SECOND PASS RLD OUTPUT BUFFER. WRITE SECOND PASS RLD OUTPUT BUFFER ONTO SYSLMOD

6 IF THE PROGRAM IS IN OVERLAY, CREATE TTR LIST CONTAINING THE ADDRESS OF FIRST CONTROL RECORD OF EACH SEGMENT

SECOND PASS PROCESSING (HEWLFSCD)
CROSS-REFERENCE CHART

	CSECT	LABEL	FLOWCHART
1	SCDENTAB	SGEND1	FA
2	HEWLFREL	SCDOVLY	FE
3	HEWLFREL	RELOC20	FE
A	HEWLFREL	RELOC100	FE
4	WRTTXT		FD
5	WRTCRRLD		FA
6	HEWLCPTH		FE

DIAGRAM 7. FINAL PROCESSING



1 COMPLETE THE PARTITION DATA SET DIRECTORY INCLUDING MODIFICATIONS FOR ALIAS SYMBOLS

2 ISSUE STOW MACRO FROM THE PDS TO SYSLMOD

3 WRITE THE TTR LIST CONTAINING THE ADDRESS OF THE FIRST TEXT RECORD IN EACH SEGMENT ONTO SYSLMOD FOR OVERLAY PROGRAMS

4 IF XREF WAS SPECIFIED, BUT WAS NOT PROCESSED DURING INTERMEDIATE PROCESSING (SEE DIAGRAM 5)
- A READ RLDs FROM SYSLMOD
- B BUILD A CROSS-REFERENCE TABLE FROM SYSLMOD
- C WRITE THE CROSS-REFERENCE TABLE ONTO SYSPRINT

5 SCAN THE ERROR LOGGING MAP
- A. BUILD THE ERROR DIAGNOSTIC DIRECTORY
- B WRITE THE ERROR DIAGNOSTIC DIRECTORY ON SYSPRINT

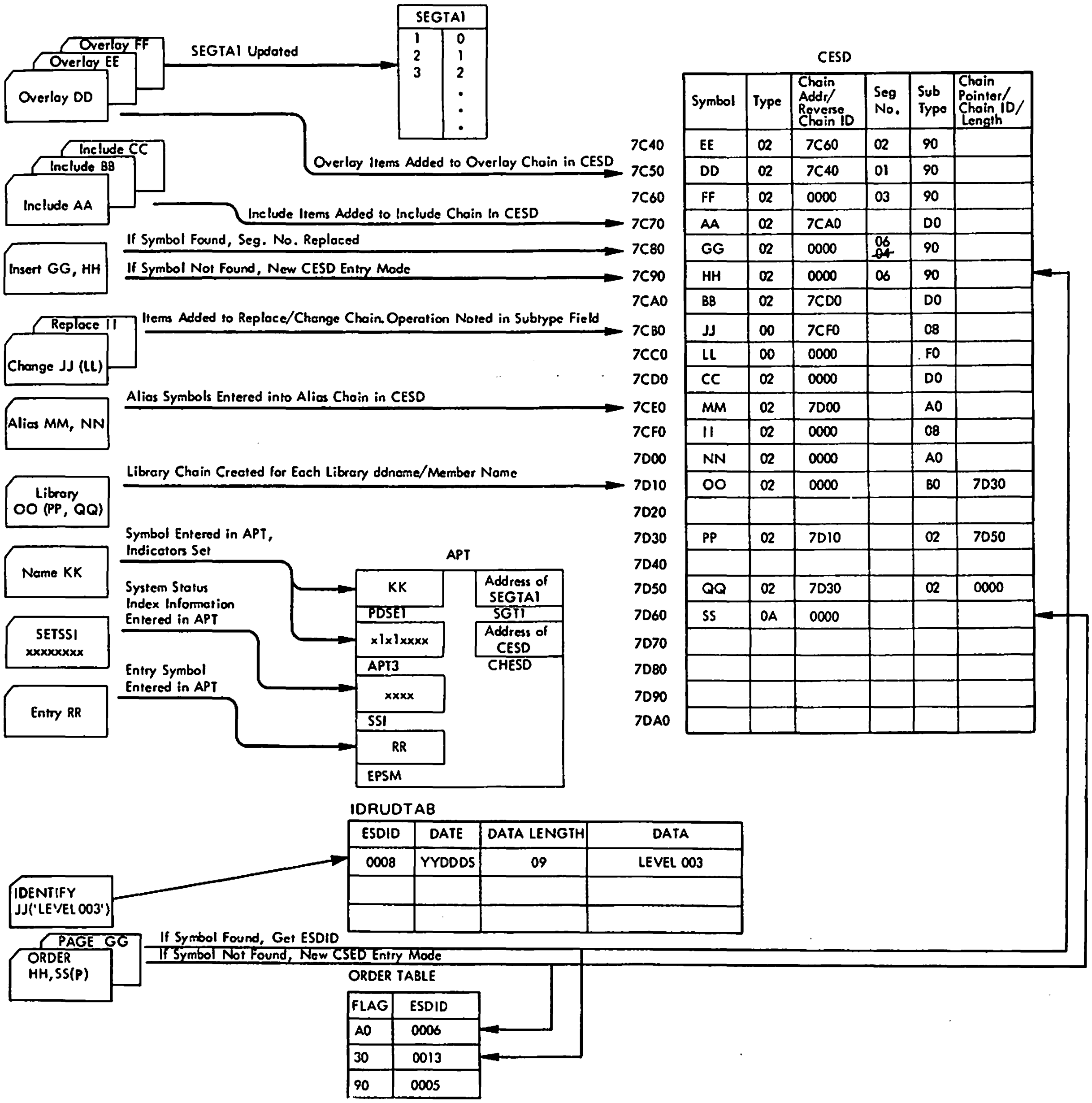
6 IF THE TERM OPTION WAS SPECIFIED, WRITE THE ERROR DIAGNOSTIC DIRECTORY ON SYSTERM

7 RELEASE ALL STORAGE ALLOCATED TO THE LINKAGE EDITOR

FINAL PROCESSING (HEWLFFNL)
CROSS-REFERENCE CHART

	CSECT	LABEL	FC	TEXT
1	HEWLFFNL	FNL900A	GA	
2	HEWLFFNL	FNL301A	GA	
3	HEWLFFNL	FNL100	GA	
4	HEWLFMAP		EB	
B	HEWLFMAP	RLDOUTA	EB	
C	HEWLFMAP	PUTLINES	EB	
5	HEWLFLOG		GC	
A	HEWLFBTP			
B	HEWLFLOG	LOG10	GC	
6	HEWFLOG		GC	
7	HEWFFNL	IEWLCEOI	GA	

DIAGRAM 8. CONTROL STATEMENT PROCESSING


Overlay FF
Overlay EE
Overlay DD
SEGTA1 Updated
SEGTA1
1 0
2 1
3 2
Overlay Items Added to Overlay Chain in CESD
Include CC
Include BB
Include AA
Include Items Added to Include Chain In CESD
If Symbol Found, Seg. No. Replaced
Insert GG, HH
If Symbol Not Found, New CESD Entry Made
Replace II
Change JJ (LL)
Items Added to Replace/Change Chain. Operation Noted in Subtype Field
Alias MM, NN
Alias Symbols Entered into Alias Chain in CESD
Library OO (PP, QQ)
Library Chain Created for Each Library ddname/Member Name
Name KK
Symbol Entered in APT, Indicators Set
SETSSI xxxxxxxx
System Status Index Information Entered in APT
Entry RR
Entry Symbol Entered in APT
APT
IDENTIFY JJ('LEVEL 003')
PAGE GG
ORDER HH,SS(P)
If Symbol Found, Get ESDID
If Symbol Not Found, New CSED Entry Made

CESD

	Symbol	Type	Chain Addr/ Reverse Chain ID	Seg No.	Sub Type	Chain Pointer/ Chain ID/ Length
7C40	EE	02	7C60	02	90	
7C50	DD	02	7C40	01	90	
7C60	FF	02	0000	03	90	
7C70	AA	02	7CA0		D0	
7C80	GG	02	0000	06 ~~04~~	90	
7C90	HH	02	0000	06	90	
7CA0	BB	02	7CD0		D0	
7CB0	JJ	00	7CF0		08	
7CC0	LL	00	0000		F0	
7CD0	CC	02	0000		D0	
7CE0	MM	02	7D00		A0	
7CF0	II	02	0000		08	
7D00	NN	02	0000		A0	
7D10	OO	02	0000		B0	7D30
7D20						
7D30	PP	02	7D10		02	7D50
7D40						
7D50	QQ	02	7D30		02	0000
7D60	SS	0A	0000			
7D70						
7D80						
7D90						
7DA0						

APT

KK	Address of SEGTA1
PDSE1	SGT1
x1x1xxxx	Address of CESD
APT3	CHESD
xxxx	
SSI	
RR	
EPSM	

IDRUDTAB

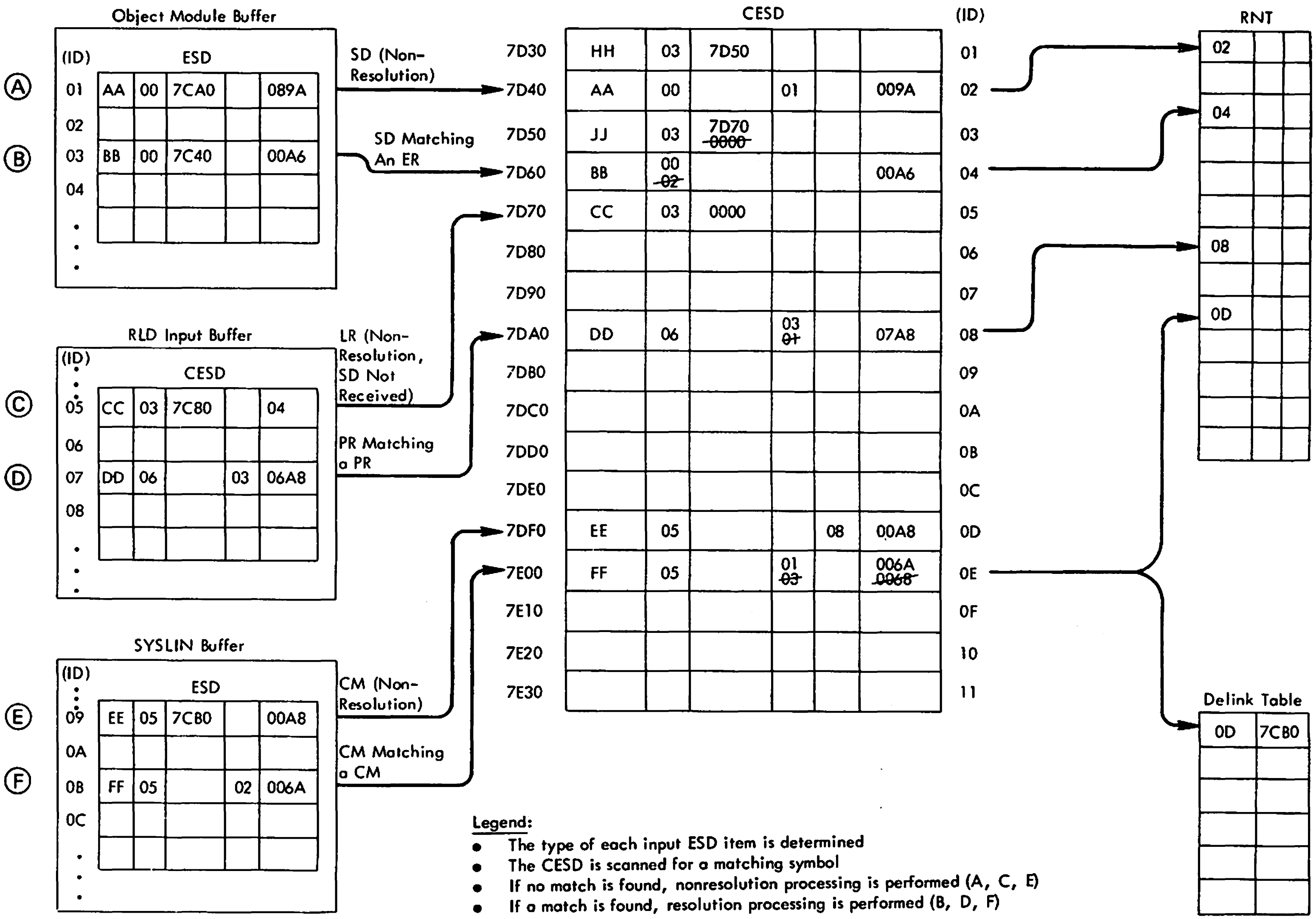
ESDID	DATE	DATA LENGTH	DATA
0008	YYDDDS	09	LEVEL 003

ORDER TABLE

FLAG	ESDID
A0	0006
30	0013
90	0005

Contains Restricted Materials of IBM
Licensed Materials — Property of IBM

DIAGRAM 9. ESD PROCESSING



Object Module Buffer

	(ID)	ESD				
Ⓐ	01	AA	00	7CA0		089A
	02					
Ⓑ	03	BB	00	7C40		00A6
	04					
	⋮					

RLD Input Buffer

	(ID)	CESD				
	⋮					
Ⓒ	05	CC	03	7C80		04
	06					
Ⓓ	07	DD	06		03	06A8
	08					
	⋮					

SYSLIN Buffer

	(ID)	ESD				
	⋮					
Ⓔ	09	EE	05	7CB0		00A8
	0A					
Ⓕ	0B	FF	05		02	006A
	0C					
	⋮					

CESD

							(ID)
7D30	HH	03	7D50				01
7D40	AA	00		01		009A	02
7D50	JJ	03	7D70 ~~0000~~				03
7D60	BB	00 ~~02~~				00A6	04
7D70	CC	03	0000				05
7D80							06
7D90							07
7DA0	DD	06		03 ~~01~~		07A8	08
7DB0							09
7DC0							0A
7DD0							0B
7DE0							0C
7DF0	EE	05			08	00A8	0D
7E00	FF	05		01 ~~03~~		006A ~~0068~~	0E
7E10							0F
7E20							10
7E30							11

RNT

02		
04		
08		
0D		

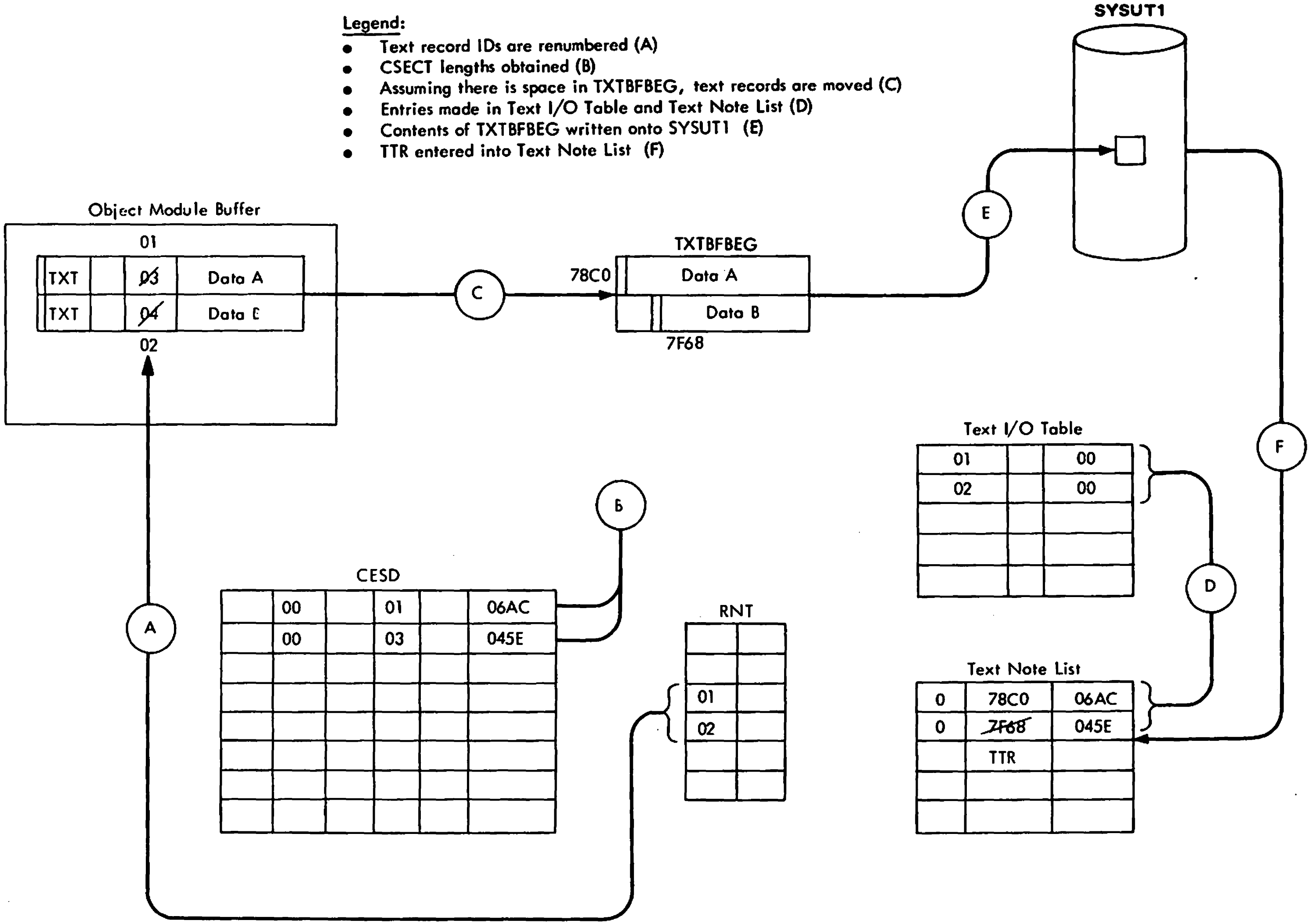
Delink Table

0D	7CB0

Legend:

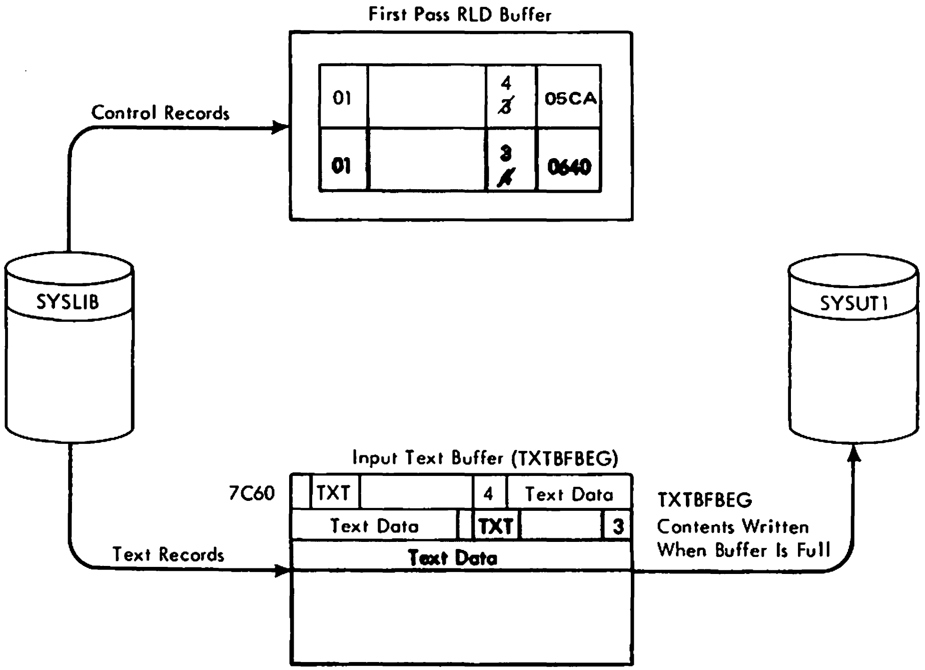
- The type of each input ESD item is determined
- The CESD is scanned for a matching symbol
- If no match is found, nonresolution processing is performed (A, C, E)
- If a match is found, resolution processing is performed (B, D, F)

DIAGRAM 10. PROCESSING OBJECT MODULE TEXT



Legend:
- Text record IDs are renumbered (A)
- CSECT lengths obtained (B)
- Assuming there is space in TXTBFBEG, text records are moved (C)
- Entries made in Text I/O Table and Text Note List (D)
- Contents of TXTBFBEG written onto SYSUT1 (E)
- TTR entered into Text Note List (F)

DIAGRAM 11. PROCESSING LOAD MODULE TEXT RECORDS



Renumbering Table (RNT)

2	
1	
4	
3	
5	

Text I/O Table

ID	Mult
4	0
3	0

Text Note List

Disp	Addr	Length
0	7C60	05CA
0	822A	0640

CESD

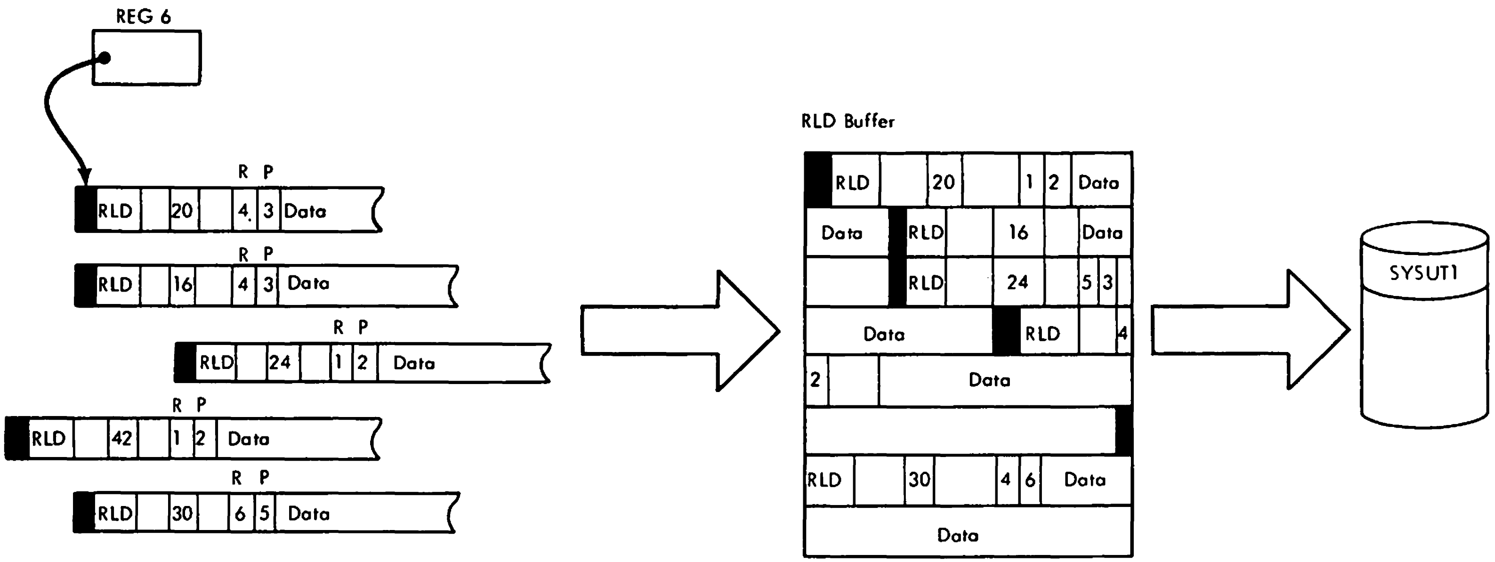
AA	02				
BB	03				
CC	00				0640
DD	00				05CA
EE	06				

Legend:

- The ID in the first control record is renumbered. The third line of the RNT contains a 4, so the ID is renumbered to refer to the fourth line of the CESD (CSECT DD).
- Assuming CSECT DD (CESD ID = 4) is not to be deleted, its length (in the control record) is checked.
- If the entire CSECT or a complete multiplicity will fit in TXTBFBEG, the record containing text for DD is read into TXTBFBEG, and entries are made in the text I/O table and the text note list*.
- Each subsequent control record is processed. Text records are read into TXTBFBEG until it becomes full, at which time its contents are written onto SYSUT1.

* In the two text records in this example, the multiplicity number is 0, because they are the first text records for their respective control sections.

DIAGRAM 12. RLD PROCESSING



RNT

5	
3	
2	
1	
6	
4	

CESD

CSECTA	00					
RLDA	02					
RLDB	02					
CSECTC	00					
CSECTB	00					
RLDC	02					

RLD Note List

ID	Mult	Length	Addr/ Displ
2		20	16
3		24	82
6		30	~~138~~ TTR

Legend:

- Register 6 initially points to the first RLD input record.
- RLD records are grouped in the RLD buffer by P pointer. In this example, the first and second, and third and fourth RLD records are grouped.
- R and P pointers are renumbered, using the renumbering table, as RLD records are moved into the buffer.
- Entries for each RLD set are made in the RLD note list. Length and displacement fields refer to the first record of the set.
- When the contents of the RLD buffer are written, the displacement field of the RLD note list entry for the last set included in the output record is replaced by the relative track address (TTR) of the SYSUT1 record.

DIAGRAM 13. ADDRESS ASSIGNMENT

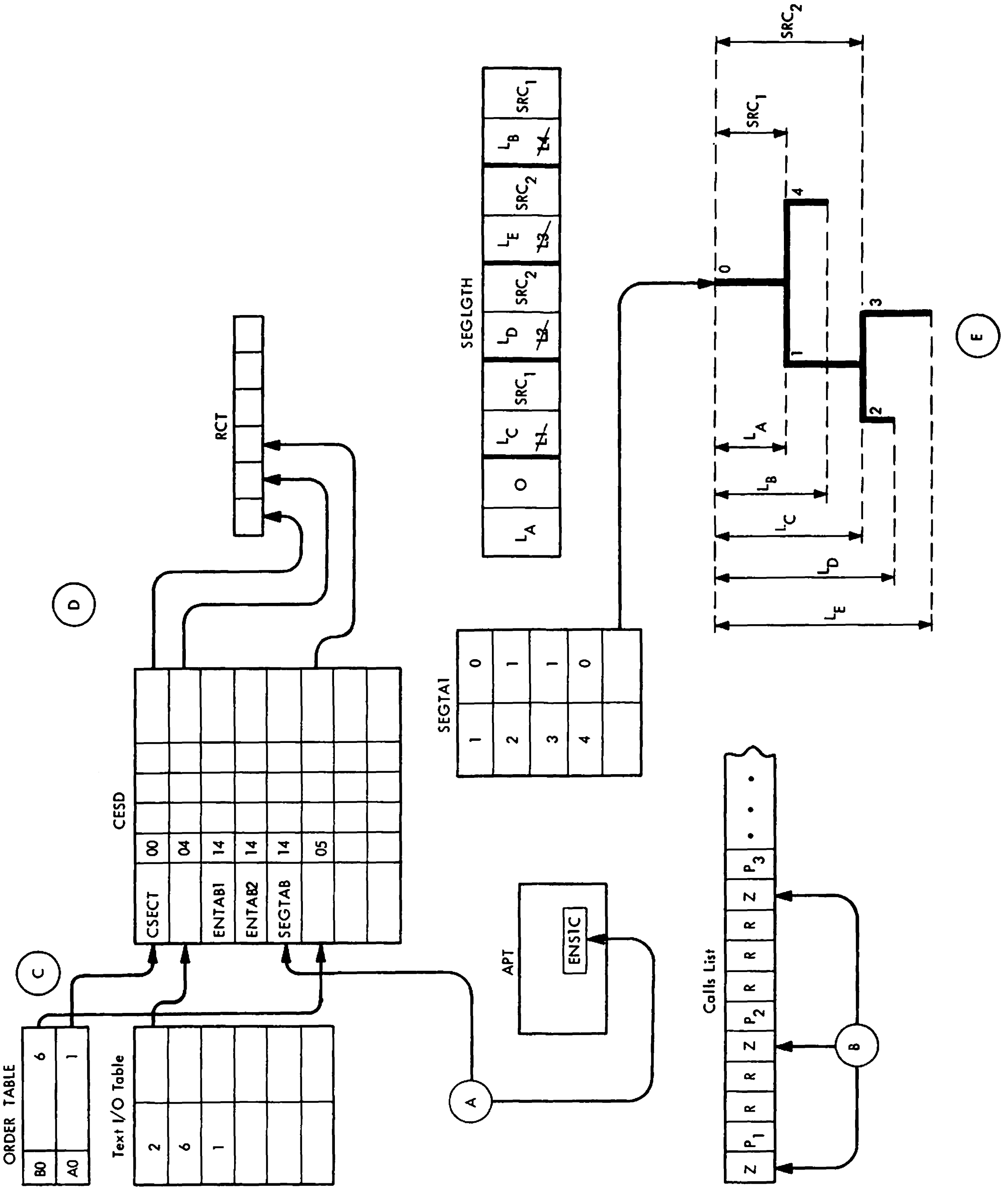
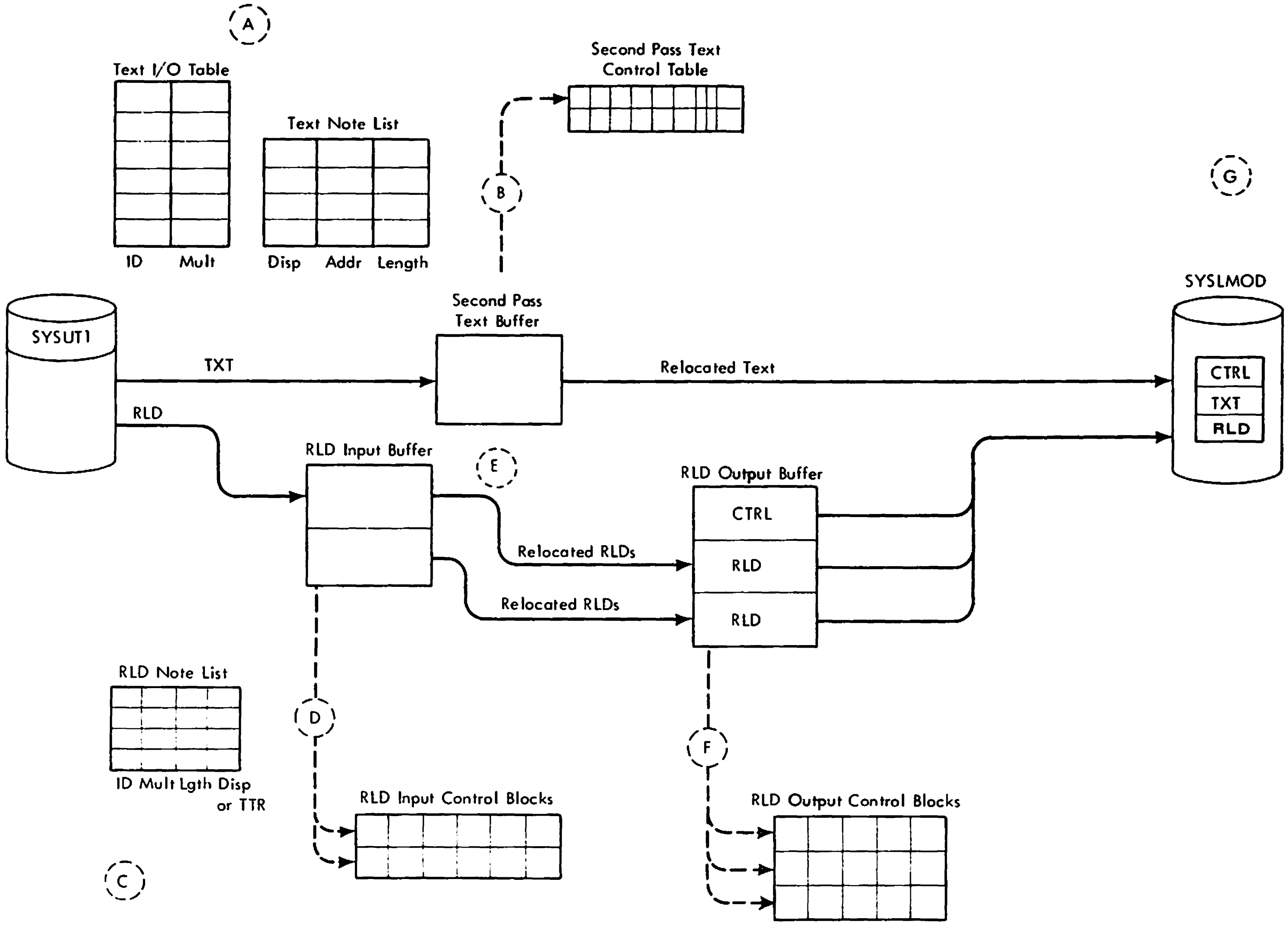


DIAGRAM 14. DATA MOVEMENT DURING SECOND PASS PROCESSING



PROGRAM ORGANIZATION

The following text and the flowcharts at the end of this section describe the processors (code modules, control sections, and routines) that accomplish the functions of the linkage editor. The organization of this section corresponds to the organization of the linkage editor; descriptions of all processors that constitute a phase of the linkage editor are grouped together. For each processor, the symbolic name is given to facilitate use of program listing (see "Microfiche Directory"), and the descriptive name is given to facilitate reference to "Method of Operation."

Figure 31 on page 97 shows the overall organization of the linkage editor; this illustration is designed to help determine relationships among the processors described in this section.

INITIALIZATION AND INPUT PROCESSING

Initial Processor—HEWLFINT (Chart BA)

Entrance: HEWLFINT is entered from HEWLFROU at the beginning of linkage editor processing.

Operation: HEWLFINT performs initialization functions, including building the all-purpose table (APT), analyzing attributes and options passed by the calling program, opening data sets, and allocating virtual storage for buffers and work areas.

Routines Called: HEWLFINT calls the attributes and options processor (HEWLFOPT) and the allocation routine (ALL001). The HEWLFINT routine is recalled immediately upon returning from the first call of the allocation routine (ALL001).

Exits: When initialization is completed, HEWLFINT passes control to the input processor (HEWLFINP).

Attributes and Options Processor—HEWLFOPT

Entrance: HEWLFOPT is entered from the initial processor.

Operation: HEWLFOPT analyzes the options requested and the attributes specified by the calling program, and notes this information in the APT. If a valid authorization code is found, it is converted to binary and stored in both the default field and the PDS entry field of the APT.

Routines Called: None.

Exits: When attribute and option processing is completed, HEWLFOPT returns control to the initial processor (HEWLFINT).

Allocation Processor—ALL001 (Chart BA)

Entrance: HEWLFOPT is entered from the initial processor.

Operation: ALL001 issues the GETMAIN macro instruction and assigns storage to buffers.

Routines Called: ALL001 calls the table allocation processor (HEWLFALK) to allocate storage for fixed-length and variable-length tables.

Exits: When allocation processing is completed, ALL001 returns control to the initial processor (HEWLFINT).

Table Allocation Processor—HEWLFALK (Chart BB)

Entrance: HEWLFALK is entered from ALL001 after storage has been allocated for the buffers.

Operation: HEWLFALK assigns storage to the internal tables. HEWLFALK checks minimum required to available storage. If insufficient, an error message (IEW0664) will be issued. HEWLFALK determines the amount of storage in excess of the minimum required. This excess is used to expand proportionately the variable-length tables over the minimum required.

Routines Called: None.

Exits: When table allocation processing is completed, HEWLFALK returns to the calling routine.

Input Processor—HEWLFINP (Chart CA)

Entrance: HEWLFINP receives control from the initial processor when all initialization functions are completed.

Operation: HEWLFINP reads and initially processes all linkage editor input. Input type (object module or load module) and input conditions are determined, and control is passed to appropriate processors.

Routines Called: HEWLFINP calls the following processors:

- Control statement scanner (HEWLFSCN) when a control statement is detected (blank in column 1)
- Object module processor (HEWLFMDI) when object module input is detected (SYSLIN input or fixed (F) format input from SYSLIB)
- Load module processor (INP270) when load module input is detected (undefined (U) format input from SYSLIB)
- Include processor (HEWLFINC) at end-of-input if more modules must be included
- Automatic library call processor (HEWLCAUT) at end-of-input on SYSLIN if the NCAL option is not specified

Exits: When input processing is completed, HEWLFINP passes control to the address assignment processor (HEWLFADA), if valid input was received. If no valid input was received, control is passed to the final processor (HEWLFENL) to terminate linkage editor processing.

Object Module Processor—HEWLFMDI (Chart CB)

Entrance: HEWLFMDI is entered from the input processor when object module input is detected.

Operation: HEWLFMDI determines the input record type (SYM, TXT, RLD, ESD, END), loads input record information into general registers, and passes control to the appropriate processors.

Routines Called: Depending on input record type, HEWLFMDI calls the following processors:

- SYM processor (HEWLFSYM)
- ESD processor (HEWLFESD)
- END processor (HEWLFEND)
- Text and RLD processor (HEWLFRAT)
- IDR processor (HEWLFIDR)

Exits: When object module processing is completed, HEWLFMDI returns control to the input processor.

Load Module Processor—INP270 (Chart CC)

Entrance: INP270 is entered from the input processor when load module input is detected.

Operation: INP270 determines the input record type (TXT, CESD, scatter/translation, SYM, CCW, CCW/RLD, RLD, IDR), loads input record information into general registers, and passes control to the appropriate processors.

Routines Called: Depending on input record type, INP270 calls an associated processor, as shown in Figure 30.

Exits: When load module processing is completed, INP270 returns control to the input processor.

Record Type	Processor
TXT	HEWLFRAT
CESD	HEWLFESD
Scatter/translation	(Ignored)
SYM	HEWLFSYM
CCW	HEWLFRAT
CCW/RLD	HEWLFRAT
RLD	HEWLFRAT
IDR	HEWLFIDR

If end-of-module indicator is on:

CCW	HEWLFEND
CCW/RLD	HEWLFEND
RLD	HEWLFEND

Figure 30. Load Module Record Types and Associated Processors

SYM Processor—HEWLFSYM (Chart CD)

Entrance: HEWLFSYM is entered from the object module processor when SYM records have been detected and the TEST attribute has been specified. If TEST is not specified, SYM records are ignored.

Operations: HEWLFSYM gathers SYM records in the RLD input buffer, and writes the buffer contents on SYSLMOD when the first TXT record of a module is detected.

Routines Called: None.

Exits: When SYM processing is completed, HEWLFSYM returns control to the object module processor.

ESD Processor—HEWLFESD (Chart CE)

Entrance: HEWLFESD is entered from the object module processor when an ESD record is detected, and from the load module processor when a CESD record is detected.

Operation: HEWLFESD combines ESDs in the linkage editor input into a composite ESD. Matching input symbols are resolved, and specified operations (replace, change, delete) are performed on the symbols. A renumbering table (RNT) is produced to allow input ESD IDs to be translated into CESD IDs.

Exits: When ESD processing is completed, HEWLFESD returns control to the routine from which it was entered (object module processor or load module processor).

Text and RLD Processor—HEWLFRAT (Chart CF)

Entrance: HEWLFRAT is entered from the object or load module processors when a text or RLD record is detected.

Operation: HEWLFRAT determines record type (TXT or RLD), checks for error conditions (input record larger than buffer), and passes control to the appropriate processor.

Routines Called: Depending on the record type, HEWLFRAT passes control to either the text processor (HEWLFTXT) or the RLD processor (RLD001).

Exits: When text and RLD processing is completed, HEWLFRAT returns control to the object or load module processor.

Text Processor—HEWLFTXT (Chart CG)

Entrance: HEWLFTXT is entered from the text and RLD processor when a text record is detected.

Operations: HEWLFTXT operation depends on whether text input is from object or load modules. Object module text is moved from the object module buffer to the input text buffer, and must be arranged in the proper order. Load module text input is already ordered, so HEWLFTXT reads it directly into the input text buffer. In either case, the input text ID is renumbered to refer to the CESD ID of the appropriate control section. When the input text buffer becomes full, its contents are written on SYSUT1.

Routines Called: When the input text buffer is full, HEWLFTXT calls the text write routine (TXTBUF—Chart CH) to write the buffer contents on SYSUT1.

Exits: When text processing is completed, HEWLFTXT returns control to the text and RLD processor.

RLD Processor—RLD001 (Chart CJ)

Entrance: RLD001 is entered from the text and RLD processor when an RLD record is detected.

Operation: RLD001 groups RLD items in the RLD buffer and renumbers the R and P pointers to refer to appropriate CESD entries. Each RLD item is processed according to its flag and address (FA) field. RLD001 also creates an RLD note list, with entries for each set of RLDs (a set being all RLDs having the same P pointer). If the RLD buffer becomes full, the contents of the buffer are written on SYSUT1 and noted in the RLD note list.

Routines Called: When the RLD buffer is full, RLD001 calls the RLD write routine (RLDBUF—Chart CK) to write the buffer contents on SYSUT1, and an entry is made in the RLD note list.

Exits: When RLD processing is completed, RLD001 returns control to the text and RLD processor.

End Processor—HEWLFEND (Chart CL)

Entrance: HEWLFEND is entered from the object or load module processor when an END record or the end of a load module is detected.

Operation: HEWLFEND resets tables involved in input processing, processes entry point information, deletes CESD lines marked "chain" or "delete," and enters in the CESD the length of control sections for which no length was previously indicated.

Routines Called: None.

Exits: When end processing is completed, HEWLFEND returns control to the object or load module processor.

CSECT Identification Record (IDR) Processor—HEWLFIDR (Chart CQ)

Entrance: HEWLFIDR is entered from the input processor, HEWLFINP, to process object module END records and load module identification records. It is also entered from HEWLFSCN for processing IDENTIFY control statements.

Operation: HEWLFIDR takes IDR information from the input records and enters this data in the appropriate IDR table.

Routines Called: Error and informative messages are processed by calling HEWLFLOG.

Exits: When IDR processing ends, HEWLFIDR returns to the calling program.

Control Statement Scanner—HEWLFSCN (Chart CS)

Entrance: HEWLFSCN is entered from the input processor when a control statement is detected.

Operation: Depending on the type of control statement being processed, the control statement scanner makes entries in the APT, SEGTA1, and/or the CESD. This information is used to control subsequent linkage editor processing.

Routines Called: HEWLFSCN calls the READ8 routine (Chart CT) to process control statement operands.

Exits: When control statement processing is completed, HEWLFSCN passes control to the include processor (HEWLFINC) if an INCLUDE control statement was processed (include chain built in the CESD). Otherwise, HEWLFSCN returns control to the input processor.

Include Processor—HEWLFINC (Chart CU)

Entrance: HEWLFINC is entered from the input processor when "more includes" are indicated at end-of-input, and from the control statement scanner when an INCLUDE statement has been processed.

Operation: HEWLFINC examines the include chain in the CESD and selects the next module to be included. It opens the data set, determines the attributes of the module to be included, and initializes the DCB to allow the module to be read.

Routines Called: If a module for which the REPLACE/CHANGE function has been requested is not contained in the specified library, HEWLFINC calls HEWLFEND to delete the corresponding CESD lines.

Exits: When include processing is completed, control is returned to the input processor.

Automatic Library Call Processor—HEWLCAUT (Chart CV)

Entrance: HEWLCAUT is entered from the input processor at the end of SYSLIN input, or when a NAME statement has been detected (provided that the NCAL option was not specified).

Operation: HEWLCAUT first scans the CESD for unresolved ERs specified on LIBRARY statements. It attempts to resolve these ERs by searching the PDS directories of ddnames included in library chains, allowing the members found to be read. A second CESD scan attempts to resolve ERs not specified on LIBRARY statements by attempting to call them from SYSLIB.

Routines Called: After the first series of CESD scans, HEWLCAUT returns control to the input processor to read the members.

Exits: After the second series of CESD scans, HEWLCAUT passes control to the address assignment processor (HEWLFADA).

INTERMEDIATE PROCESSING

Address Assignment Processor—HEWLFADA (Chart DA)

Entrance: HEWLFADA is entered from the input processor when input processing is completed

Operation: HEWLFADA assigns linked addresses to all CESD entries, determines the size of SEGTAB if the program is in overlay, determines if the first text record does not begin at address 0, determines the number of ENTAB bytes required for each segment, builds the alias table, and determines an entry point for the program.

Routines Called: HEWLFADA calls the ENTAB size determination routine (HEWLFENS—Chart DB) to compute the size of ENTAB, and calls the entry processor (HEWLFENT—Chart DC) to build the alias table and determine an entry point.

Exits: When address assignment processing is completed, HEWLFADA passes control to the intermediate output processor (HEWLFOUT).

Intermediate Output Processor—HEWLFOUT (Chart EA)

Entrance: HEWLFOUT is entered from HEWLFADA when address assignment processing is complete.

Operation: HEWLFOUT writes the following on SYSLMOD; CESD, SEGTAB (for programs in overlay), and scatter/translation records (for programs to be scatter loaded). If a HIARCHY statement is specified, storage hierarchy designations are included in the scatter/translation records. If the MAP option has been specified and the 2-byte table can contain all necessary entries, a module map is produced and written on SYSPRINT; if the XREF option is specified, and the 2-byte table can contain all necessary entries, and all RLDs are in storage, a cross-reference table and a module map are produced and written on SYSPRINT.

HEWLFOUT builds the high ID table (HIID). The half ESD (HESD) is also built, after the CESD has been written.

Routine Called: HEWLFOUT calls the MAP/XREF processor (HEWLFMAP) to produce and write the module map and cross-reference table, if requested.

Exits: When intermediate output processing is completed, control is passed to the second pass processor (HEWLFSCD).

SECOND PASS PROCESSING

Second Pass Processor—HEWLFSCD (Chart FA)

Entrance: HEWLFSCD is entered from HEWLFOUT when intermediate output processing is completed.

Operation: HEWLFSCD performs the following functions:

- Reads text from SYSUT1
- Relocates address constants contained in the text
- Creates control/RLD records
- Writes text and control/RLD records on SYSLMOD in a format that can be loaded by program fetch
- Creates ENTABs and associated RLD items for overlay modules

Routines Called: During second pass processing, HEWLFSCD calls the following routines:

- Control section search routine (GETIDMUL—Chart FB) to determine the next ID and multiplicity to be processed
- Text and RLD read routines (RDTXT, RDRLD—Chart FC) to read required text and RLDs from SYSUT1
- Text write routine (WRTTXT—Chart RD) to write text on SYSLMOD (HEWLMSIO)
- Control/RLD record write routine (WRTCRRLD) to write RLDs and control records on SYSLMOD (HEWLFSIO)
- Second pass initialization routine (HEWLFREL—Chart FE) to initialize text and RLD control blocks
- Relocation routine (RELOCATE—Chart FE) to relocate address constants (branch-type and nonbranch-type) in the text
- Common path routine (HEWLCPTH) to determine common segments in an overlay path
- ENTAB creation routine (SCDENTAB) to create ENTAB items for each segment

Exits: When second pass processing is completed, control is passed to the final processor (HEWLFFNL).

FINAL PROCESSING

Final Processor—HEWLFFNL (Chart GA)

Entrance: HEWLFFNL is entered from HEWLFSCD when second pass processing is completed.

Operation: HEWLFFNL performs the following "cleanup" functions:

- Writes the TTR list for overlay modules on SYSLMOD
- Places entries in the partitioned data set directory and issues a STOW macro instruction
- Prints a directory of logged errors

- Checks for more restrictive module attributes
- Produces a MAP and a cross-reference table if it was requested and not produced during intermediate processing, and if the primary or alternate 2-byte table will contain the needed entries. Warning message IEW0801 is issued if the alternate 2-byte table is too small.

Routines Called: During final processing, HEWLFFNL calls the following routines:

- Diagnostic message directory print routine (HEWLFBTP), which scans the error logging map produced throughout linkage editor processing by the error logging routine (HEWLFLOG—Chart GC); HEWLFBTP builds and prints a directory of error messages.
- MAP/XREF processor (HEWLFMAP—Chart EB), which produces a cross-reference table if it was not produced during intermediate processing.

Exits: If end-of-file was not detected on a SYSLIN input, HEWLFFNL returns control to the initial processor (HEWLFINT), and linkage editor processing is repeated. Otherwise, linkage editor processing is terminated and control is returned to the control program.

SYNAD Routine—HEWLCR01 (Chart GB)

Entrance: The SYNAD routine may be entered from the following routines:

- From the control program when any input/output error has been detected
- From the second pass processor if an error is found after executing the XDAP macro instruction

Operation: Following are SYNAD considerations for the linkage editor:

- The SYNAD fields of the DCBs in HEWLFROU contain the address of the appropriate SYNAD entry point for the access method used with the data set.
- If the SYNAD routine is entered from the input processor because of incorrect length, the length of the incorrect input block is checked. If a valid short block (integral multiple of (LRECL) is found, control is returned to the supervisor to continue processing; if not, processing is terminated with an error message and completion code of 16.
- If the SYNAD routine is entered while writing to the SYSPRINT data set, control is passed to the final processor, and execution is abnormally terminated with a condition code of 16.
- When the include processor opens the DCB for SYSLIB, the address of the appropriate SYNAD entry (for either BSAM or BPAM access methods) is moved into the SYNAD field.
- If the second pass processor finds an error after executing the XDAP macro instruction, it loads register 1 with the IOB address, loads register 15 with the SYNAD entry point for the EXCP macro instruction, and branches on register 15.

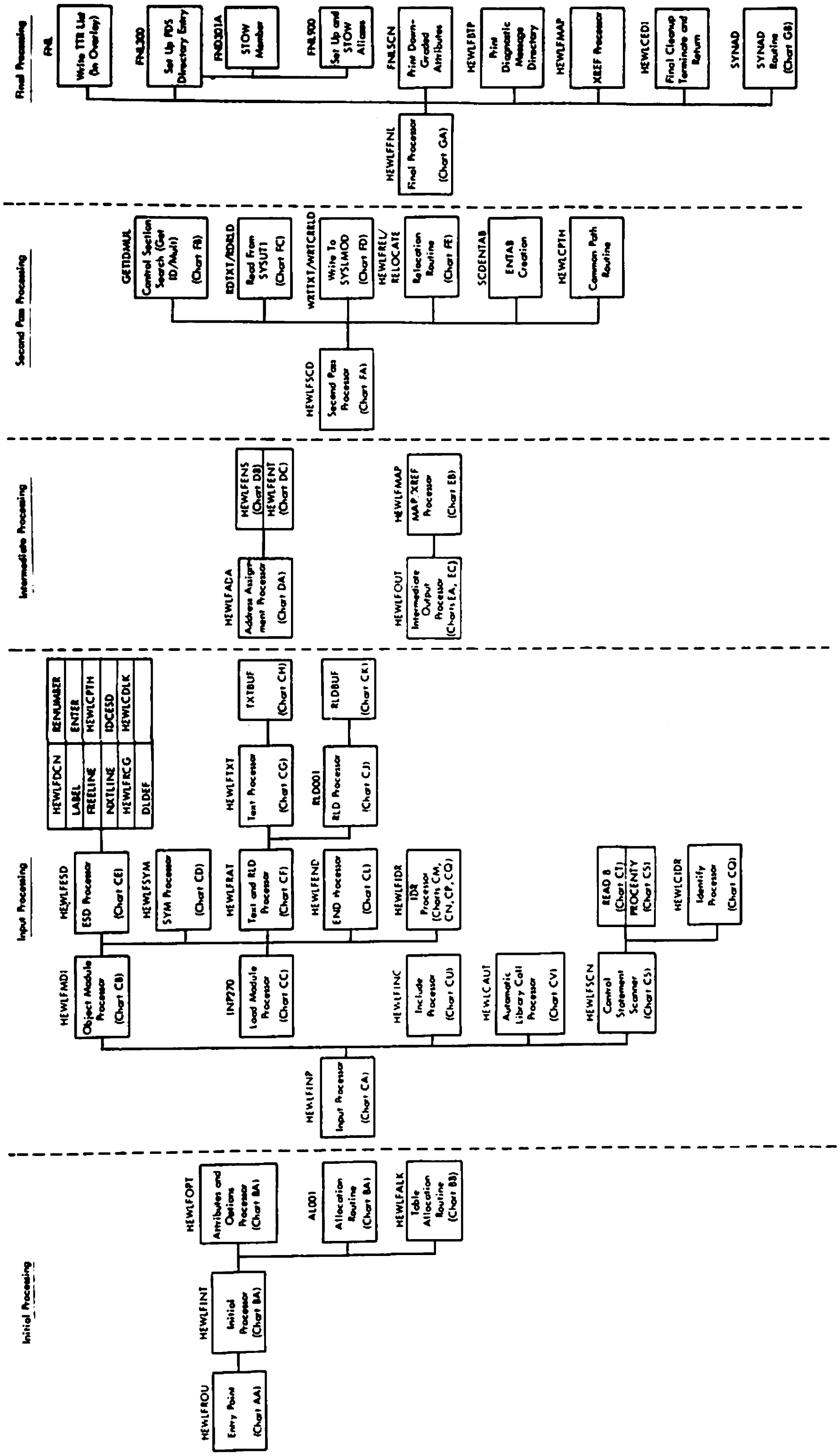
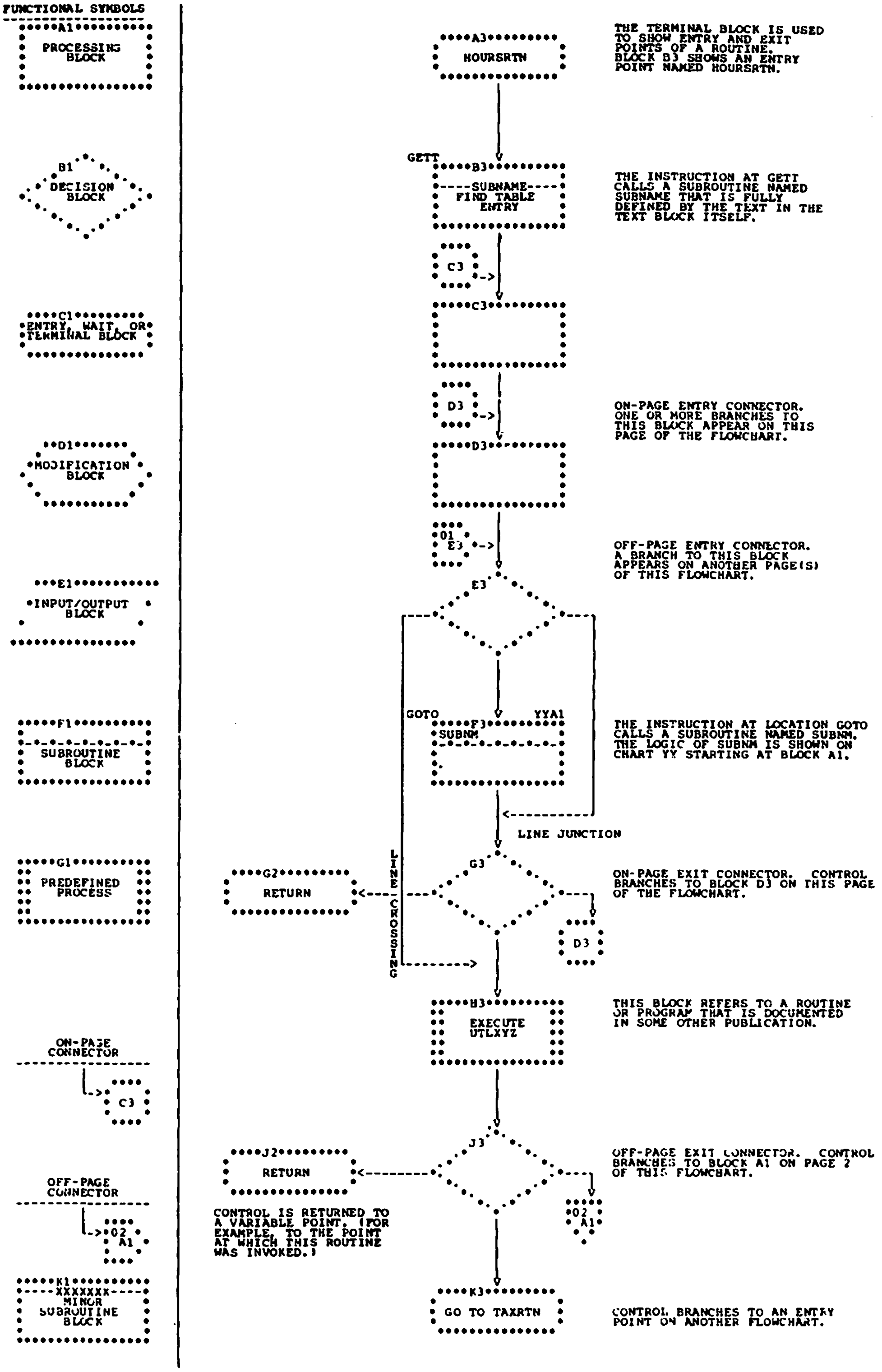

Initial Processing
HEWLFROU
Entry Point (Chart AA)
HEWLFINT
Initial Processor (Chart BA)
HEWLFOPT
Attributes and Options Processor (Chart BA)
AL001
Allocation Routine (Chart BA)
HEWLFALK
Table Allocation Routine (Chart BB)
Input Processing
HEWLFINP
Input Processor (Chart CA)
HEWLFMDI
Object Module Processor (Chart CB)
INP270
Load Module Processor (Chart CC)
HEWLFINC
Include Processor (Chart CU)
HEWLCAUT
Automatic Library Call Processor (Chart CV)
HEWLFSCN
Control Statement Scanner (Chart CS)
HEWLFESD
ESD Processor (Chart CE)
HEWLFSYM
SYM Processor (Chart CD)
HEWLFRAT
Text and RLD Processor (Chart CF)
HEWLFEND
END Processor (Chart CL)
HEWLFIDR
IDR Processor (Charts CM, CN, CP, CQ)
READ B (Chart CT)
PROCENTY (Chart CS)
HEWLCIDR
Identify Processor (Chart CQ)
HEWLFDCN
LABEL
FREELINE
NXTLINE
HEWLFRCG
DLDEF
RENUMBER
ENTER
HEWLCPTH
IDCESD
HEWLCDLK
HEWLFTXT
Text Processor (Chart CG)
TXTBUF (Chart CH)
RLD001
RLD Processor (Chart CJ)
RLDBUF (Chart CK)
Intermediate Processing
HEWLFADA
Address Assignment Processor (Chart DA)
HEWLFENS (Chart DB)
HEWLFENT (Chart DC)
HEWLFOUT
Intermediate Output Processor (Charts EA, EC)
HEWLFMAP
MAP XREF Processor (Chart EB)
Second Pass Processing
HEWLFSCD
Second Pass Processor (Chart FA)
GETIDMUL
Control Section Search (Get ID/Mult) (Chart FB)
RDTXT/RDRLD
Read From SYSUT1 (Chart FC)
WRTTXT/WRTCRRLD
Write To SYSLMOD (Chart FD)
HEWLFREL/RELOCATE
Relocation Routine (Chart FE)
SCDENTAB
ENTAB Creation
HEWLCPTH
Common Path Routine
Final Processing
HEWLFFNL
Final Processor (Chart GA)
FNL
Write TTR List (In Overlay)
FNL300
Set Up PDS Directory Entry
FND301A
STOW Member
FNL900
Set Up and STOW Aliases
FNLSCN
Print Down-Graded Attributes
HEWLFBTP
Print Diagnostic Message Directory
HEWLFMAP
XREF Processor
HEWLCEDI
Final Cleanup Terminate and Return
SYNAD
SYNAD Routine (Chart GB)

Figure 31. Linkage Editor Organization



THE TERMINAL BLOCK IS USED TO SHOW ENTRY AND EXIT POINTS OF A ROUTINE. BLOCK B3 SHOWS AN ENTRY POINT NAMED HOURSRTN.

THE INSTRUCTION AT GETT CALLS A SUBROUTINE NAMED SUBNAME THAT IS FULLY DEFINED BY THE TEXT IN THE TEXT BLOCK ITSELF.

ON-PAGE ENTRY CONNECTOR. ONE OR MORE BRANCHES TO THIS BLOCK APPEAR ON THIS PAGE OF THE FLOWCHART.

OFF-PAGE ENTRY CONNECTOR. A BRANCH TO THIS BLOCK APPEARS ON ANOTHER PAGE(S) OF THIS FLOWCHART.

THE INSTRUCTION AT LOCATION GOTO CALLS A SUBROUTINE NAMED SUBNM. THE LOGIC OF SUBNM IS SHOWN ON CHART YY STARTING AT BLOCK A1.

ON-PAGE EXIT CONNECTOR. CONTROL BRANCHES TO BLOCK D3 ON THIS PAGE OF THE FLOWCHART.

THIS BLOCK REFERS TO A ROUTINE OR PROGRAM THAT IS DOCUMENTED IN SOME OTHER PUBLICATION.

OFF-PAGE EXIT CONNECTOR. CONTROL BRANCHES TO BLOCK A1 ON PAGE 2 OF THIS FLOWCHART.

CONTROL IS RETURNED TO A VARIABLE POINT. (FOR EXAMPLE, TO THE POINT AT WHICH THIS ROUTINE WAS INVOKED.)

CONTROL BRANCHES TO AN ENTRY POINT ON ANOTHER FLOWCHART.

Figure 32. Sample Flowchart Symbols

CHART AA. LEVEL MAJOR DIVISIONS

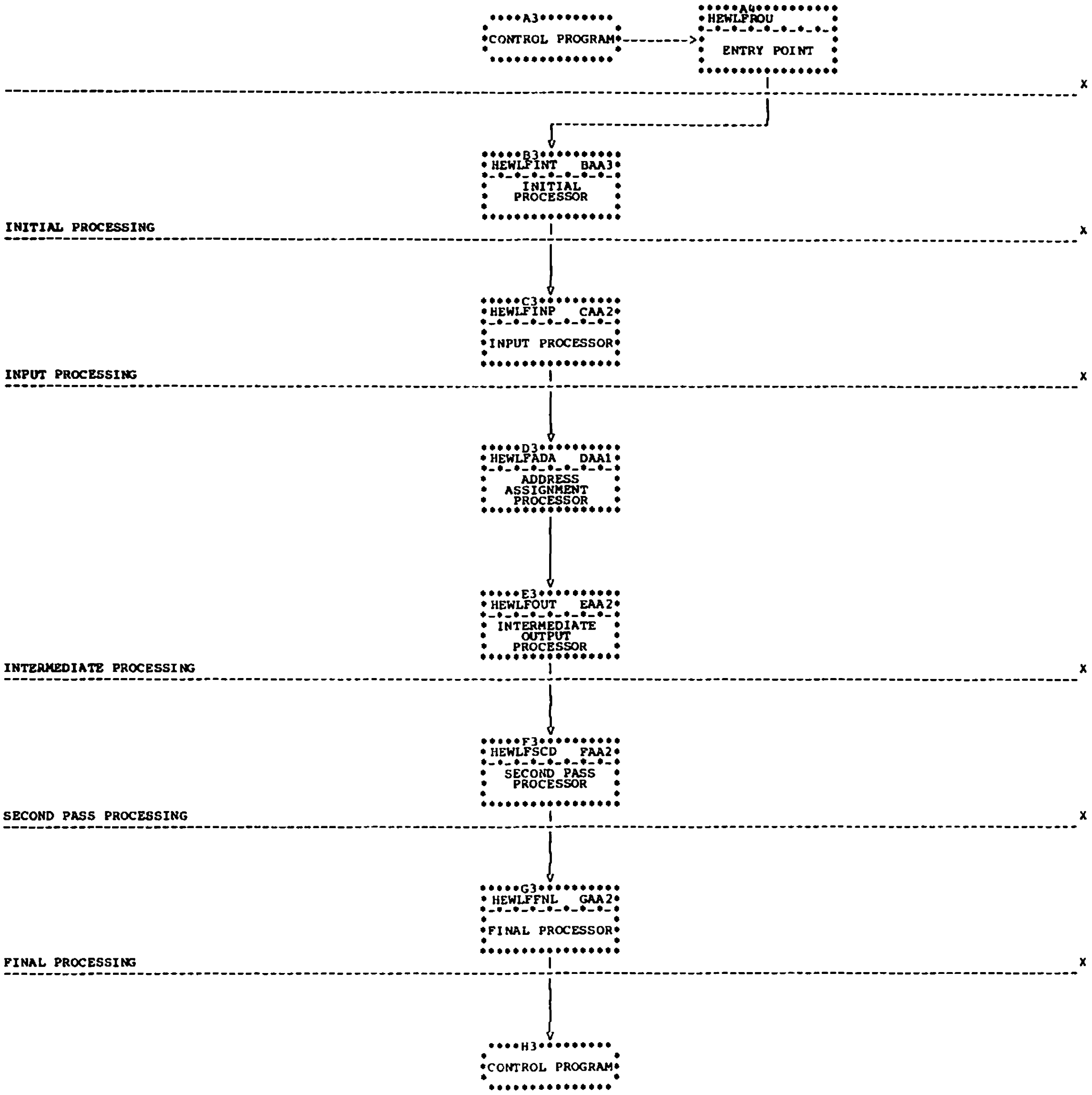


CHART BA. INITIAL PROCESSOR (HEWLFINT)

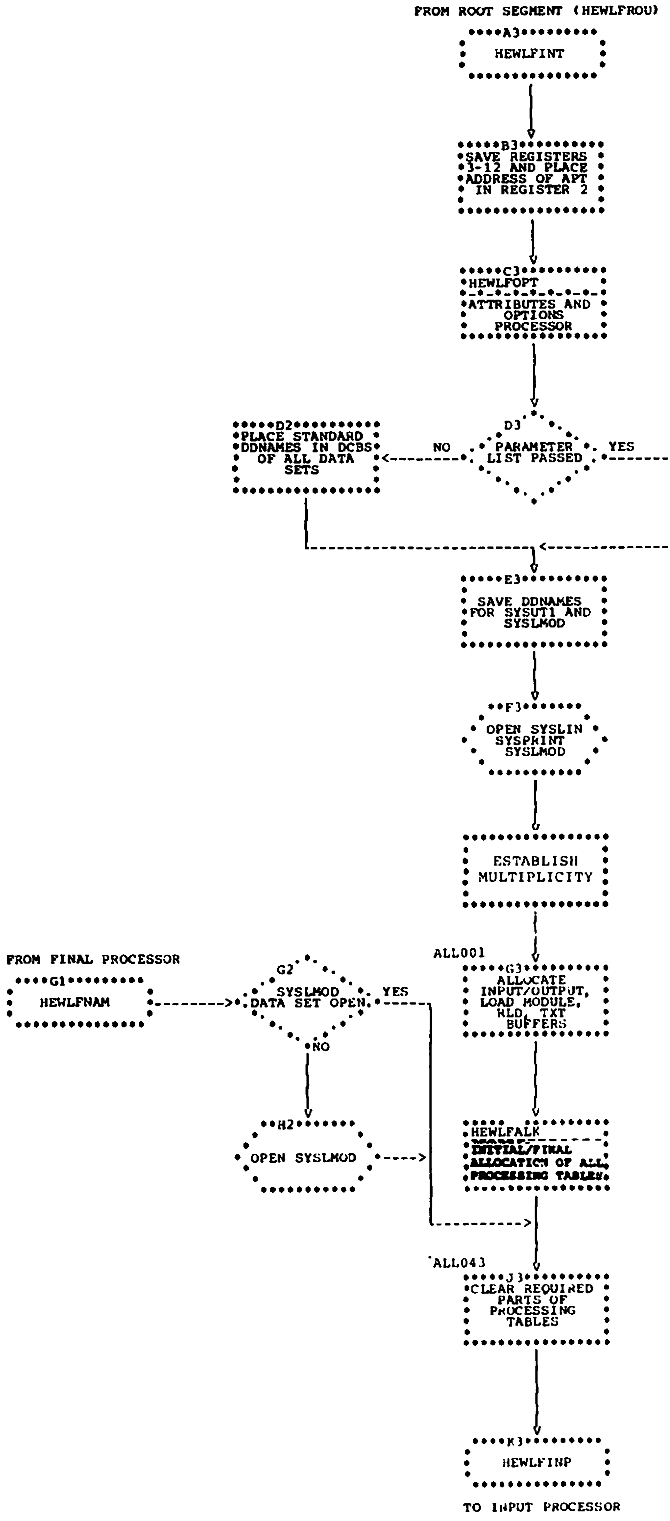
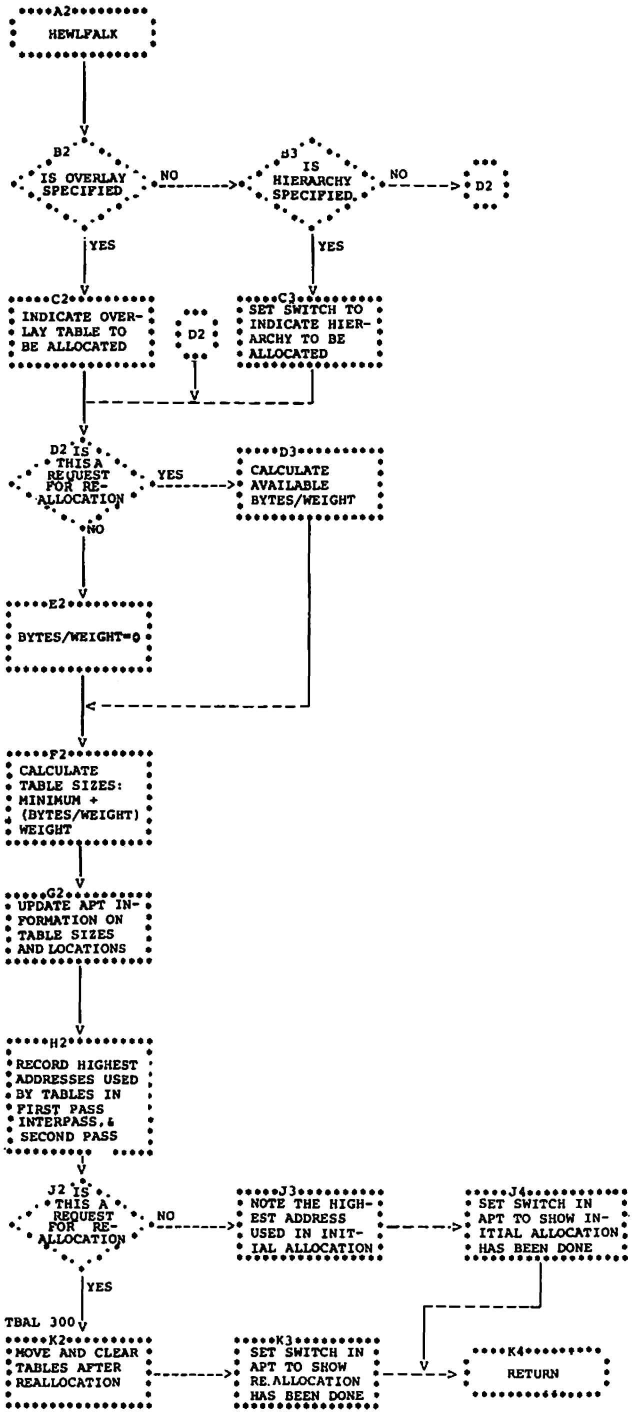

FROM ROOT SEGMENT (HEWLFROU)
A3
HEWLFINT
B3
SAVE REGISTERS 3-12 AND PLACE ADDRESS OF APT IN REGISTER 2
C3
HEWLFOPT
ATTRIBUTES AND OPTIONS PROCESSOR
D2
PLACE STANDARD DDNAMES IN DCBS OF ALL DATA SETS
NO
D3
PARAMETER LIST PASSED
YES
D4
PLACE PASSED DDNAMES IN DCBS OF ALL DATA SETS
E3
SAVE DDNAMES FOR SYSUT1 AND SYSLMOD
F3
OPEN SYSLIN SYSPRINT SYSLMOD
ESTABLISH MULTIPLICITY
FROM FINAL PROCESSOR
G1
HEWLFNAM
G2
SYSLMOD DATA SET OPEN
YES
NO
ALL001
G3
ALLOCATE INPUT/OUTPUT, LOAD MODULE, RLD, TXT BUFFERS
H2
OPEN SYSLMOD
HEWLFALK
INITIAL/FINAL ALLOCATION OF ALL PROCESSING TABLES
ALL043
J3
CLEAR REQUIRED PARTS OF PROCESSING TABLES
K3
HEWLFINP
TO INPUT PROCESSOR

CHART BB. TABLE ALLOCATION PROCESSOR (HEWLFALK)


A2
HEWLFALK
B2
IS OVERLAY SPECIFIED
NO
YES
B3
IS HIERARCHY SPECIFIED
NO
D2
YES
C2
INDICATE OVER-LAY TABLE TO BE ALLOCATED
D2
C3
SET SWITCH TO INDICATE HIER-ARCHY TO BE ALLOCATED
D2
IS THIS A REQUEST FOR RE-ALLOCATION
YES
NO
D3
CALCULATE AVAILABLE BYTES/WEIGHT
E2
BYTES/WEIGHT=0
F2
CALCULATE TABLE SIZES: MINIMUM + (BYTES/WEIGHT) WEIGHT
G2
UPDATE APT IN-FORMATION ON TABLE SIZES AND LOCATIONS
H2
RECORD HIGHEST ADDRESSES USED BY TABLES IN FIRST PASS INTERPASS, & SECOND PASS
J2
IS THIS A REQUEST FOR RE-ALLOCATION
NO
YES
J3
NOTE THE HIGH-EST ADDRESS USED IN INIT-IAL ALLOCATION
J4
SET SWITCH IN APT TO SHOW IN-ITIAL ALLOCATION HAS BEEN DONE
TBAL 300
K2
MOVE AND CLEAR TABLES AFTER REALLOCATION
K3
SET SWITCH IN APT TO SHOW RE.ALLOCATION HAS BEEN DONE
K4
RETURN

RT CA. INPUT PROCESSOR (HEWLFINP)

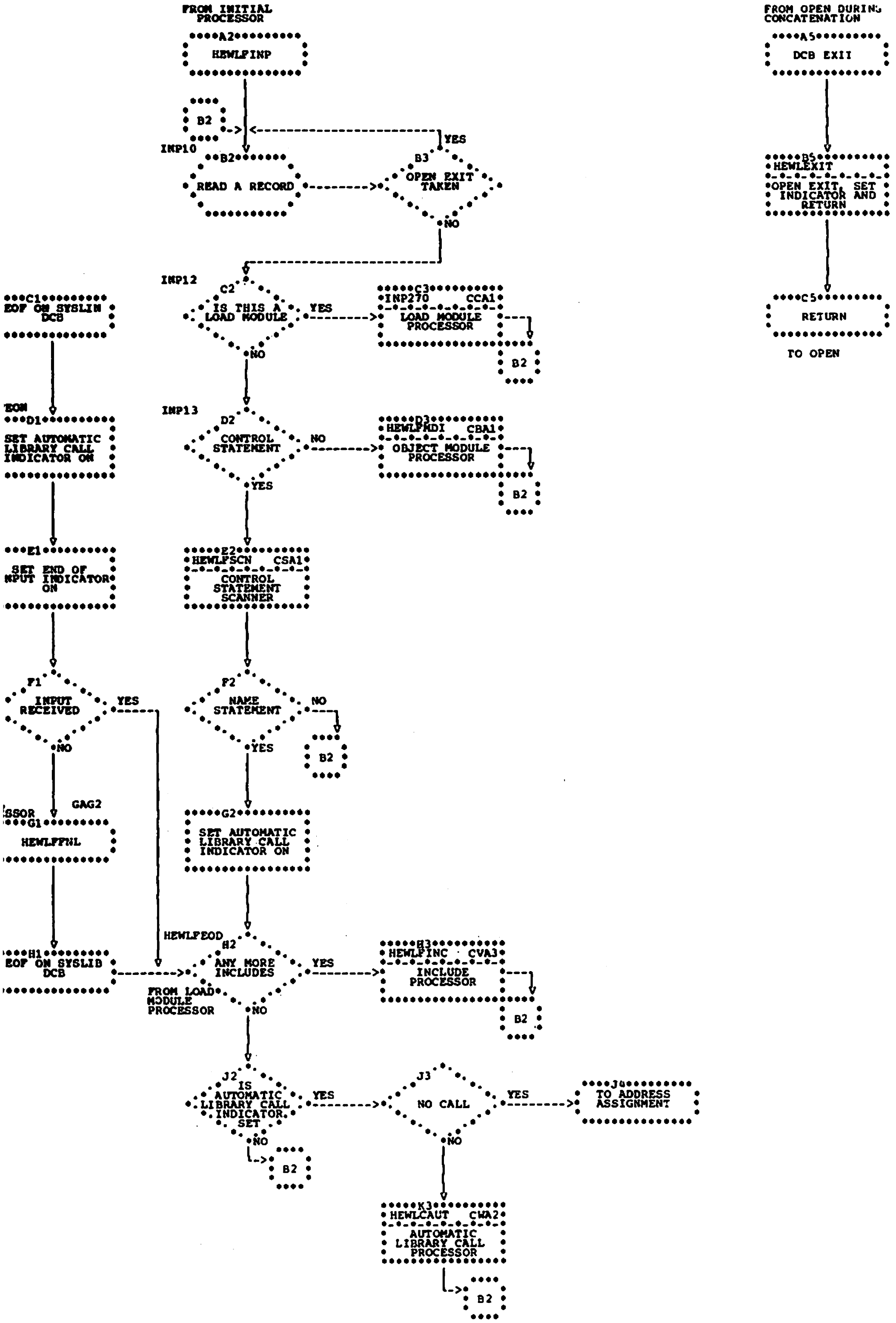


CHART CB. OBJECT MODULE PROCESSOR (HEWLFMDI)

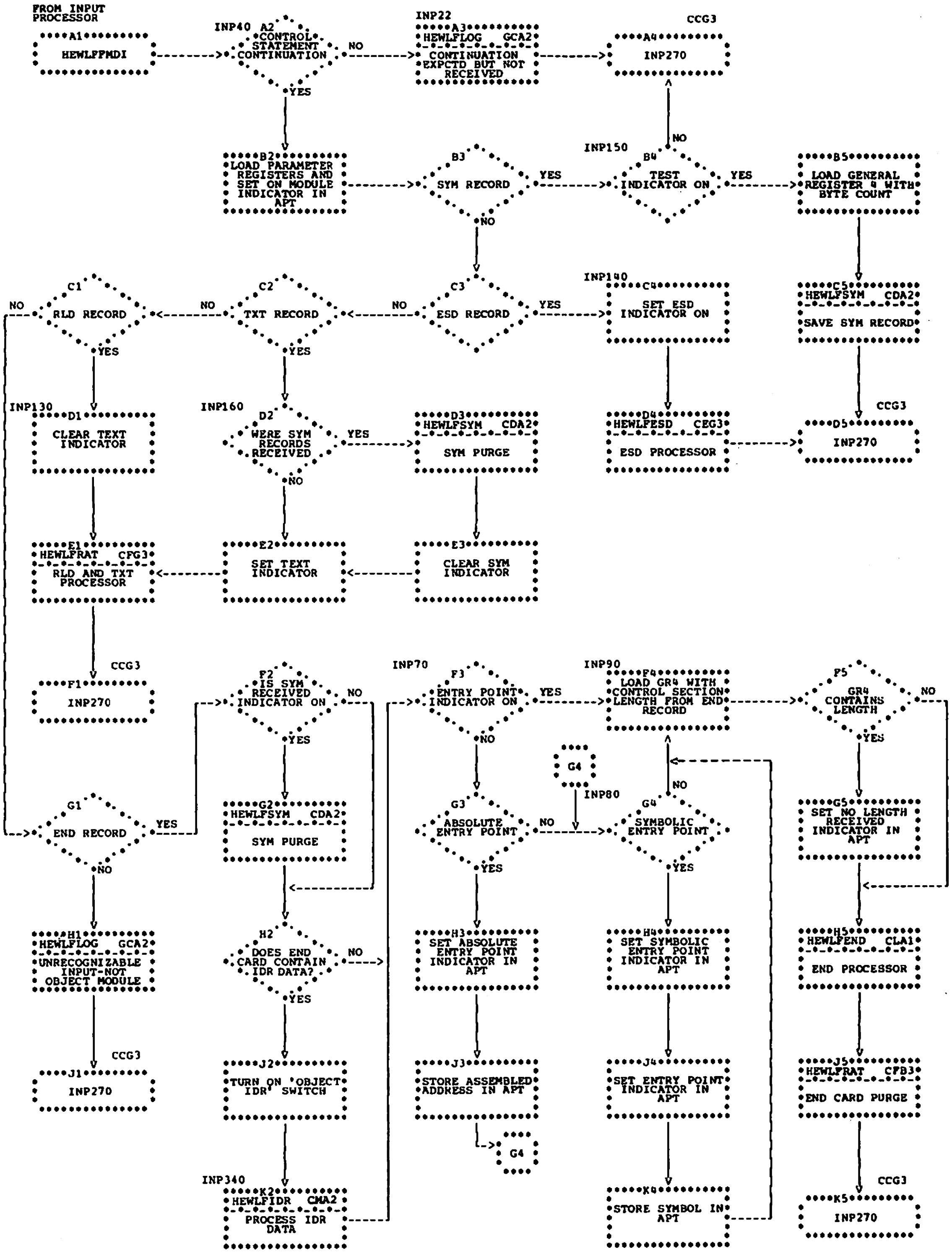

FROM INPUT PROCESSOR
A1 HEWLFFMDI
INP40 A2 CONTROL STATEMENT CONTINUATION
NO
YES
INP22 A3 HEWLFLOG GCA2 CONTINUATION EXPCTD BUT NOT RECEIVED
CCG3
A4 INP270
B2 LOAD PARAMETER REGISTERS AND SET ON MODULE INDICATOR IN APT
B3 SYM RECORD
YES
NO
INP150 B4 TEST INDICATOR ON
YES
NO
B5 LOAD GENERAL REGISTER 4 WITH BYTE COUNT
C1 RLD RECORD
NO
YES
C2 TXT RECORD
NO
YES
C3 ESD RECORD
NO
YES
INP140 C4 SET ESD INDICATOR ON
C5 HEWLFSYM CDA2 SAVE SYM RECORD
INP130 D1 CLEAR TEXT INDICATOR
INP160 D2 WERE SYM RECORDS RECEIVED
YES
NO
D3 HEWLFSYM CDA2 SYM PURGE
D4 HEWLFESD CEG3 ESD PROCESSOR
CCG3
D5 INP270
E1 HEWLFRAT CFG3 RLD AND TXT PROCESSOR
E2 SET TEXT INDICATOR
E3 CLEAR SYM INDICATOR
CCG3
F1 INP270
F2 IS SYM RECEIVED INDICATOR ON
NO
YES
INP70 F3 ENTRY POINT INDICATOR ON
YES
NO
INP90 F4 LOAD GR4 WITH CONTROL SECTION LENGTH FROM END RECORD
F5 GR4 CONTAINS LENGTH
NO
YES
G4
INP80
G1 END RECORD
YES
NO
G2 HEWLFSYM CDA2 SYM PURGE
G3 ABSOLUTE ENTRY POINT
NO
YES
G4 SYMBOLIC ENTRY POINT
NO
YES
G5 SET NO LENGTH RECEIVED INDICATOR IN APT
H1 HEWLFLOG GCA2 UNRECOGNIZABLE INPUT-NOT OBJECT MODULE
H2 DOES END CARD CONTAIN IDR DATA?
NO
YES
H3 SET ABSOLUTE ENTRY POINT INDICATOR IN APT
H4 SET SYMBOLIC ENTRY POINT INDICATOR IN APT
H5 HEWLFEND CLA1 END PROCESSOR
CCG3
J1 INP270
J2 TURN ON 'OBJECT IDR' SWITCH
J3 STORE ASSEMBLED ADDRESS IN APT
G4
J4 SET ENTRY POINT INDICATOR IN APT
J5 HEWLFRAT CFB3 END CARD PURGE
INP340 K2 HEWLFIDR CMA2 PROCESS IDR DATA
K4 STORE SYMBOL IN APT
CCG3
K5 INP270

CHART CC. LOAD MODULE PROCESSOR (INP270)

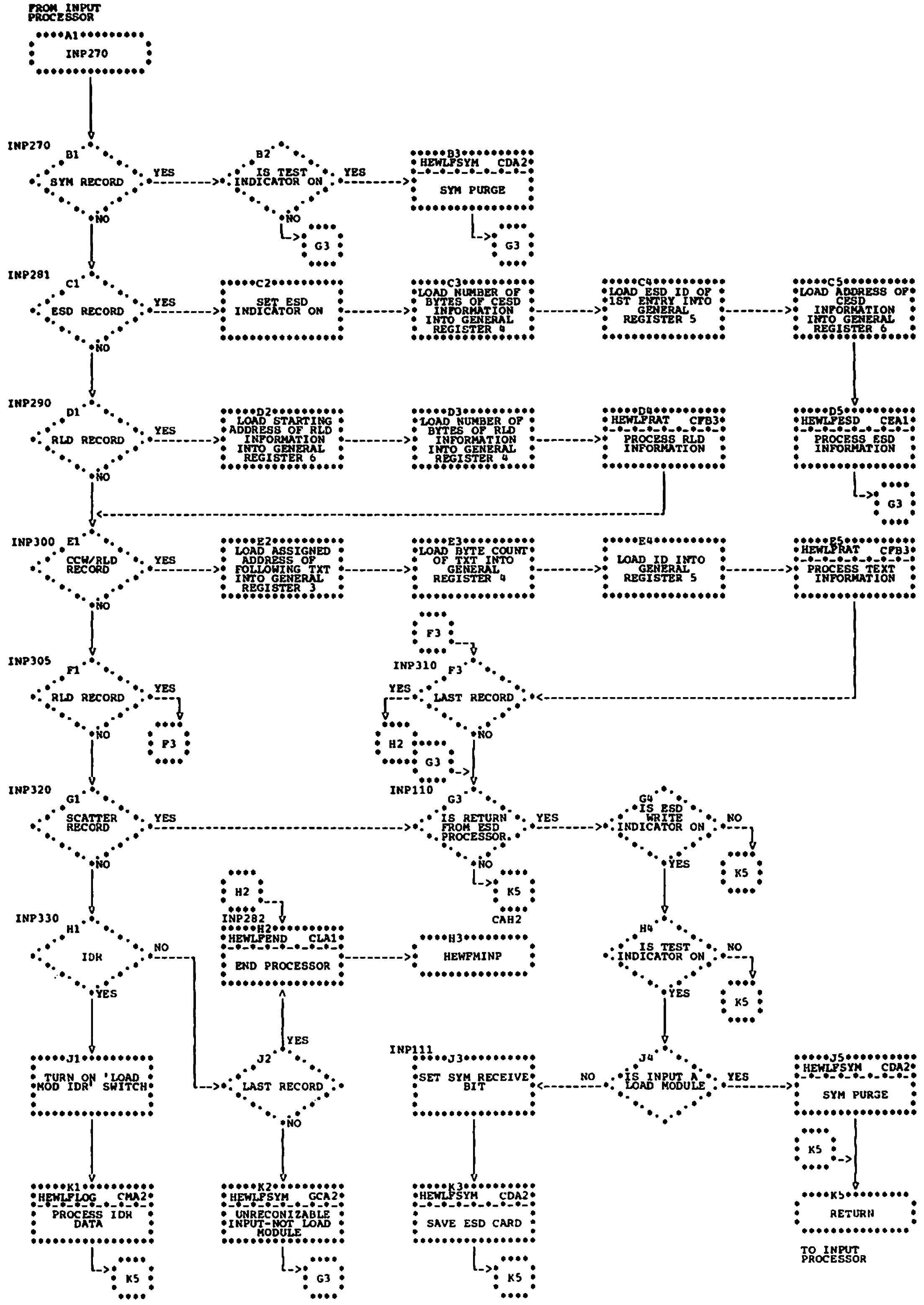


CHART CD. SYM PROCESSOR (HEWIFSYM)

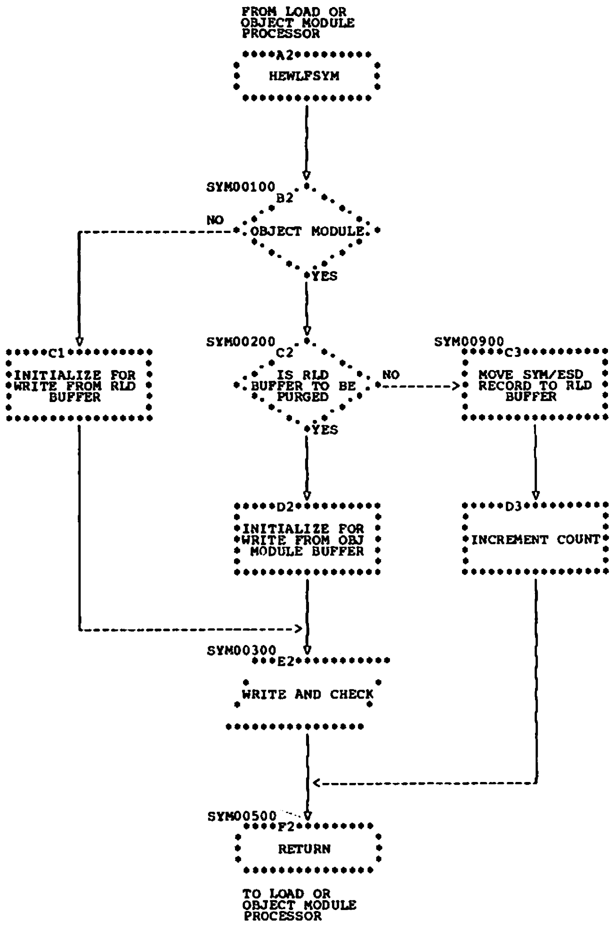

FROM LOAD OR OBJECT MODULE PROCESSOR
A2
HEWLFSYM
SYM00100
B2
OBJECT MODULE
NO
YES
C1
INITIALIZE FOR WRITE FROM RLD BUFFER
SYM00200
C2
IS RLD BUFFER TO BE PURGED
NO
YES
SYM00900
C3
MOVE SYM/ESD RECORD TO RLD BUFFER
D2
INITIALIZE FOR WRITE FROM OBJ MODULE BUFFER
D3
INCREMENT COUNT
SYM00300
E2
WRITE AND CHECK
SYM00500
F2
RETURN
TO LOAD OR OBJECT MODULE PROCESSOR

CHART CE. ESD PROCESSOR (NEWLFESD) (PART 1 OF 3)

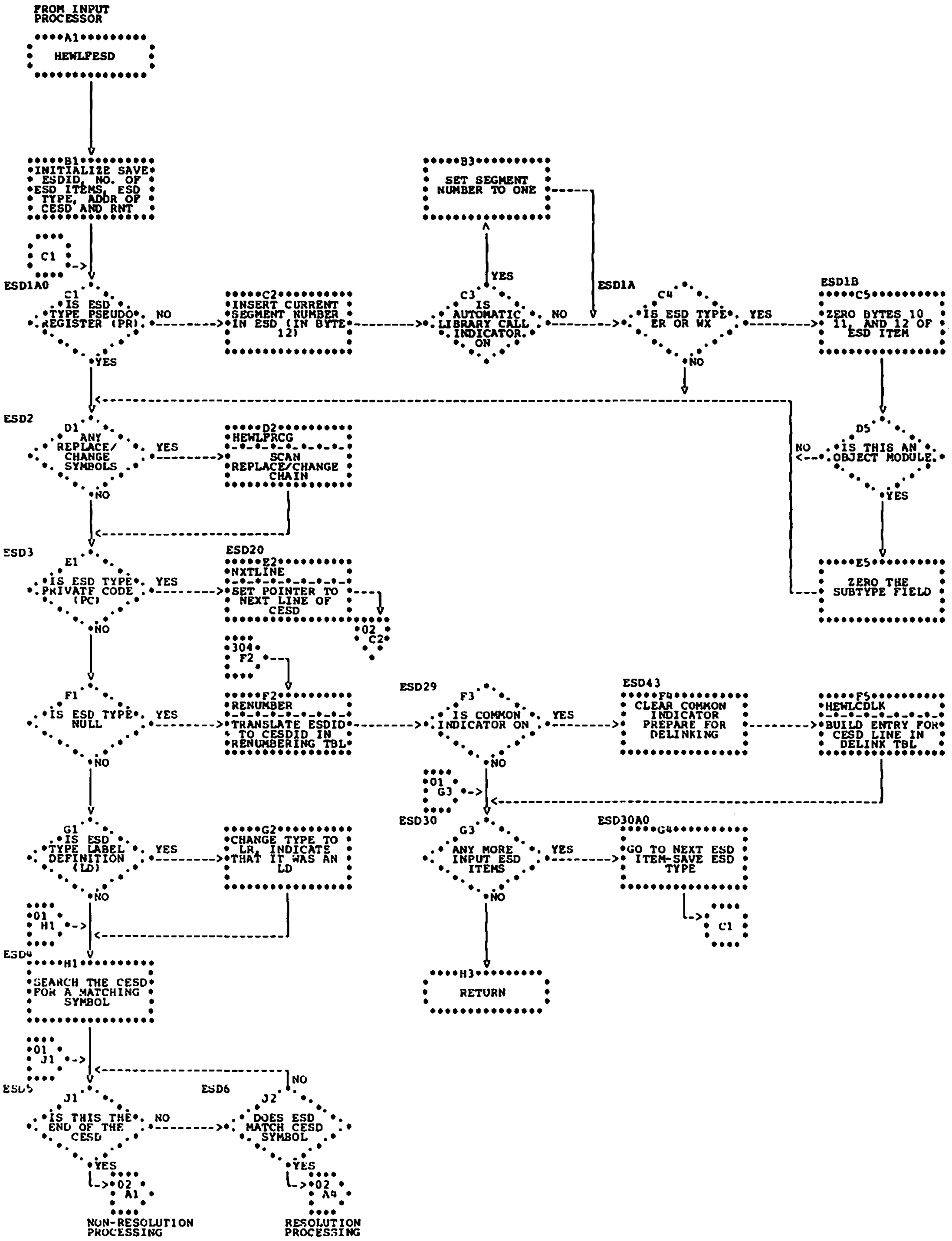

FROM INPUT PROCESSOR
A1
HEWLFESD
B1
INITIALIZE SAVE ESDID, NO. OF ESD ITEMS, ESD TYPE, ADDR OF CESD AND RNT
B3
SET SEGMENT NUMBER TO ONE
C1
ESD1A0
C1
IS ESD TYPE PSEUDO REGISTER (PR)
NO
YES
C2
INSERT CURRENT SEGMENT NUMBER IN ESD (IN BYTE 12)
C3
IS AUTOMATIC LIBRARY CALL INDICATOR ON
YES
NO
ESD1A
C4
IS ESD TYPE ER OR WX
YES
NO
ESD1B
C5
ZERO BYTES 10 11, AND 12 OF ESD ITEM
ESD2
D1
ANY REPLACE/ CHANGE SYMBOLS
YES
NO
D2
HEWLPRCG
SCAN REPLACE/CHANGE CHAIN
D5
IS THIS AN OBJECT MODULE
NO
YES
ESD3
E1
IS ESD TYPE PRIVATE CODE (PC)
YES
NO
ESD20
E2
NXTLINE
SET POINTER TO NEXT LINE OF CESD
02
C2
E5
ZERO THE SUBTYPE FIELD
304
F2
F1
IS ESD TYPE NULL
YES
NO
F2
RENUMBER
TRANSLATE ESDID TO CESDID IN RENUMBERING TBL
ESD29
F3
IS COMMON INDICATOR ON
YES
NO
ESD43
F4
CLEAR COMMON INDICATOR PREPARE FOR DELINKING
F5
HEWLCDLK
BUILD ENTRY FOR CESD LINE IN DELINK TBL
01
G3
G1
IS ESD TYPE LABEL DEFINITION (LD)
YES
NO
G2
CHANGE TYPE TO LR, INDICATE THAT IT WAS AN LD
ESD30
G3
ANY MORE INPUT ESD ITEMS
YES
NO
ESD30A0
G4
GO TO NEXT ESD ITEM-SAVE ESD TYPE
C1
01
H1
ESD4
H1
SEARCH THE CESD FOR A MATCHING SYMBOL
H3
RETURN
01
J1
ESD5
J1
IS THIS THE END OF THE CESD
NO
YES
02
A1
NON-RESOLUTION PROCESSING
ESD6
J2
DOES ESD MATCH CESD SYMBOL
NO
YES
02
A4
RESOLUTION PROCESSING

CHART CE. ESD PROCESSOR (NEWLFESD) (PART 2 OF 3)

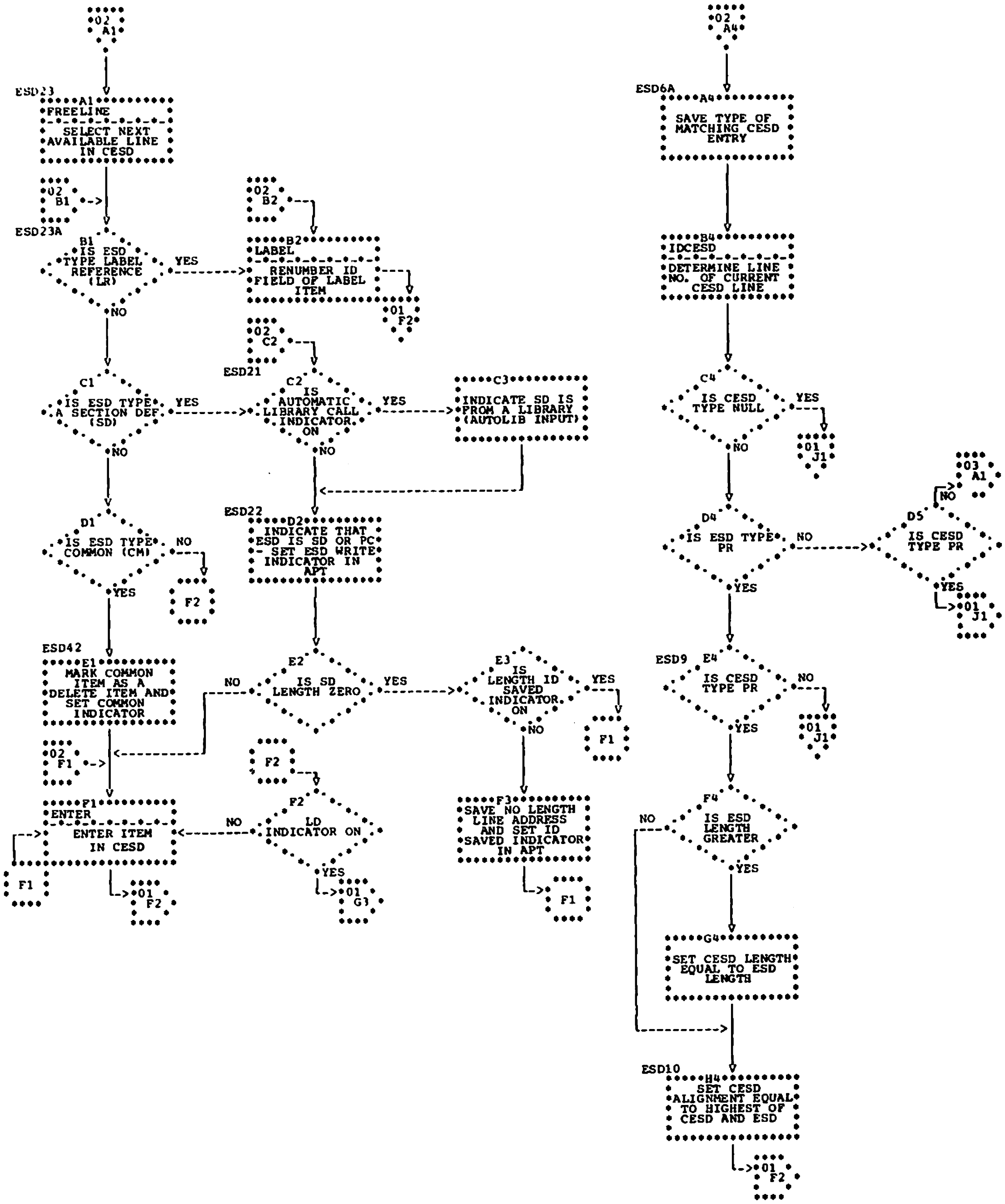

02 A1
ESD23
A1
FREELINE
SELECT NEXT AVAILABLE LINE IN CESD
02 B1
02 B2
ESD23A
B1
IS ESD TYPE LABEL REFERENCE (LR)
YES
NO
B2
LABEL
RENUMBER ID FIELD OF LABEL ITEM
01 F2
02 C2
ESD21
C1
IS ESD TYPE A SECTION DEF. (SD)
YES
NO
C2
IS AUTOMATIC LIBRARY CALL INDICATOR ON
YES
NO
C3
INDICATE SD IS FROM A LIBRARY (AUTOLIB INPUT)
D1
IS ESD TYPE COMMON (CM)
NO
YES
F2
ESD22
D2
INDICATE THAT ESD IS SD OR PC - SET ESD WRITE INDICATOR IN APT
ESD42
E1
MARK COMMON ITEM AS A DELETE ITEM AND SET COMMON INDICATOR
E2
IS SD LENGTH ZERO
NO
YES
E3
IS LENGTH ID SAVED INDICATOR ON
YES
NO
F1
02 F1
F2
F1
ENTER
ENTER ITEM IN CESD
F2
LD INDICATOR ON
NO
YES
01 G3
F3
SAVE NO LENGTH LINE ADDRESS AND SET ID SAVED INDICATOR IN APT
F1
F1
01 F2
02 A4
ESD6A
A4
SAVE TYPE OF MATCHING CESD ENTRY
B4
IDCESD
DETERMINE LINE NO. OF CURRENT CESD LINE
C4
IS CESD TYPE NULL
YES
NO
01 J1
03 A1
NO
D4
IS ESD TYPE PR
NO
YES
D5
IS CESD TYPE PR
YES
01 J1
ESD9
E4
IS CESD TYPE PR
NO
YES
01 J1
F4
IS ESD LENGTH GREATER
NO
YES
G4
SET CESD LENGTH EQUAL TO ESD LENGTH
ESD10
H4
SET CESD ALIGNMENT EQUAL TO HIGHEST OF CESD AND ESD
01 F2

CHART CE. ESD PROCESSOR (NEWLFESD) (PART 3 OF 3)

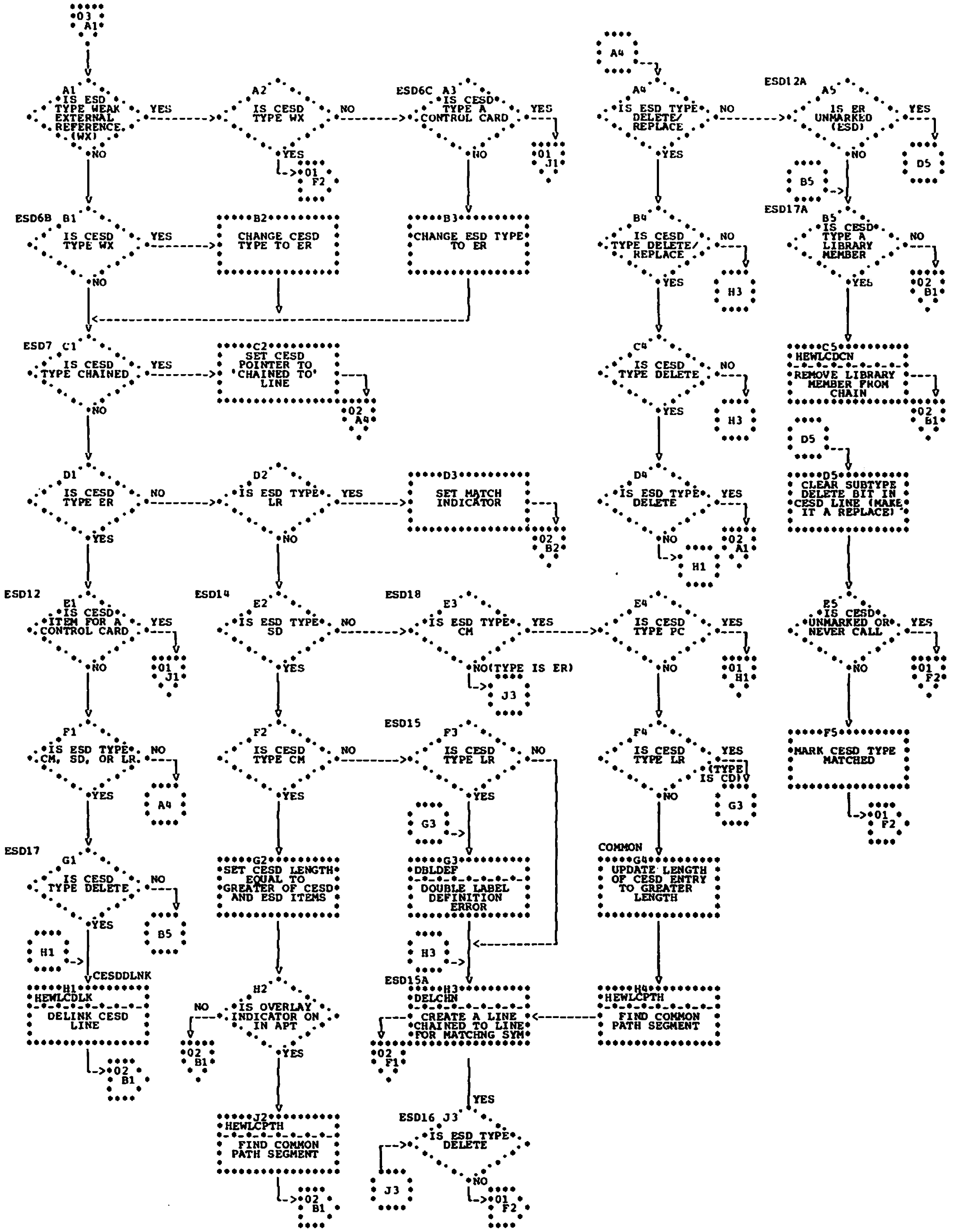

03 A1
A1 IS ESD TYPE WEAK EXTERNAL REFERENCE (WX)
YES
NO
A2 IS CESD TYPE WX
NO
YES
01 F2
ESD6C A3 IS CESD TYPE A CONTROL CARD
YES
NO
01 J1
A4
A4 IS ESD TYPE DELETE/ REPLACE
NO
YES
ESD12A A5 IS ER UNMARKED (ESD)
YES
NO
D5
B5
ESD6B B1 IS CESD TYPE WX
YES
NO
B2 CHANGE CESD TYPE TO ER
B3 CHANGE ESD TYPE TO ER
B4 IS CESD TYPE DELETE/ REPLACE
NO
YES
H3
ESD17A B5 IS CESD TYPE A LIBRARY MEMBER
NO
YES
02 B1
ESD7 C1 IS CESD TYPE CHAINED
YES
NO
C2 SET CESD POINTER TO 'CHAINED TO' LINE
02 A4
C4 IS CESD TYPE DELETE
NO
YES
H3
C5 HEWLCDCN REMOVE LIBRARY MEMBER FROM CHAIN
02 B1
D5
D1 IS CESD TYPE ER
NO
YES
D2 IS ESD TYPE LR
YES
NO
D3 SET MATCH INDICATOR
02 B2
D4 IS ESD TYPE DELETE
YES
NO
02 A1
H1
D5 CLEAR SUBTYPE DELETE BIT IN CESD LINE (MAKE IT A REPLACE)
ESD12 E1 IS CESD ITEM FOR A CONTROL CARD
YES
NO
01 J1
ESD14 E2 IS ESD TYPE SD
NO
YES
ESD18 E3 IS ESD TYPE CM
YES
NO(TYPE IS ER)
J3
E4 IS CESD TYPE PC
YES
NO
01 H1
E5 IS CESD UNMARKED OR NEVER CALL
YES
NO
01 F2
F1 IS ESD TYPE CM, SD, OR LR
NO
YES
A4
F2 IS CESD TYPE CM
NO
YES
ESD15 F3 IS CESD TYPE LR
NO
YES
G3
F4 IS CESD TYPE LR
YES (TYPE IS CD)
NO
G3
F5 MARK CESD TYPE MATCHED
01 F2
ESD17 G1 IS CESD TYPE DELETE
NO
YES
B5
H1
G2 SET CESD LENGTH EQUAL TO GREATER OF CESD AND ESD ITEMS
G3 DBLDEF DOUBLE LABEL DEFINITION ERROR
COMMON G4 UPDATE LENGTH OF CESD ENTRY TO GREATER LENGTH
CESDDLNK
H1 HEWLCDLK DELINK CESD LINE
02 B1
H2 IS OVERLAY INDICATOR ON IN APT
NO
YES
02 B1
H3
ESD15A H3 DELCHN CREATE A LINE CHAINED TO LINE FOR MATCHING SYM
02 F1
H4 HEWLCPTH FIND COMMON PATH SEGMENT
J2 HEWLCPTH FIND COMMON PATH SEGMENT
02 B1
YES
ESD16 J3 IS ESD TYPE DELETE
J3
NO
01 F2

CHART CF. TXT AND RLD PROCESSOR (HEWLFRAT)

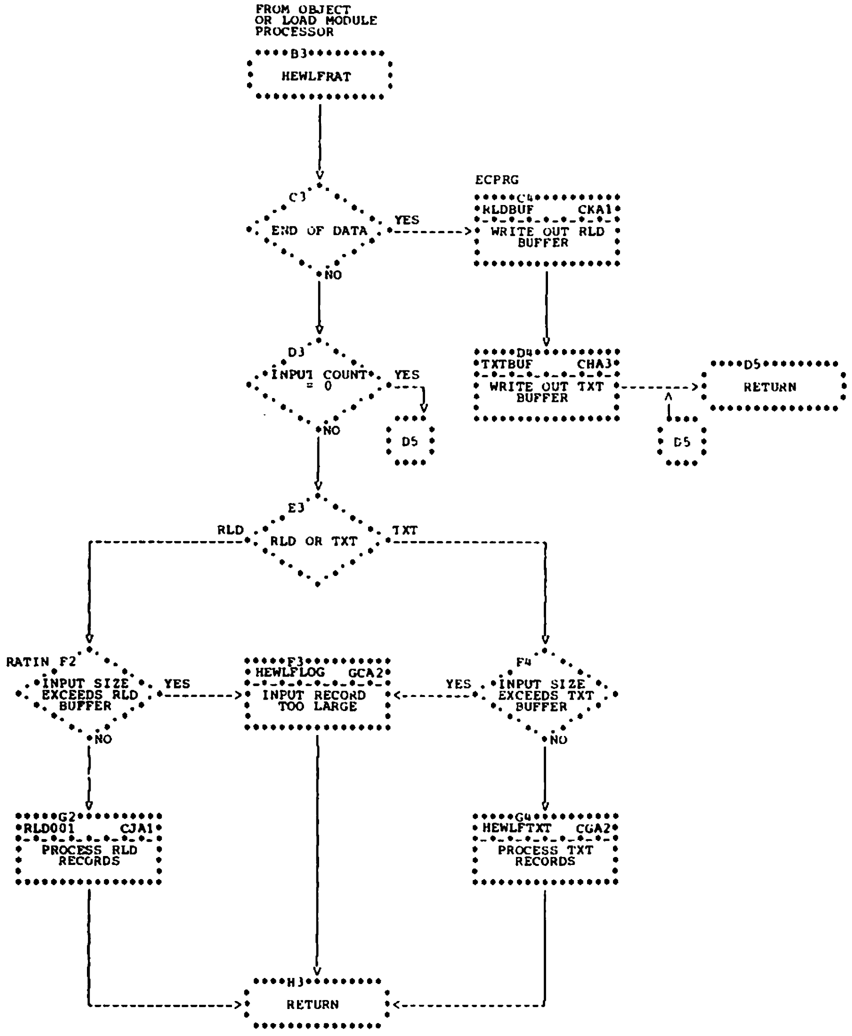


CHART CG. TXT PROCESSOR (HEWLFTXT)

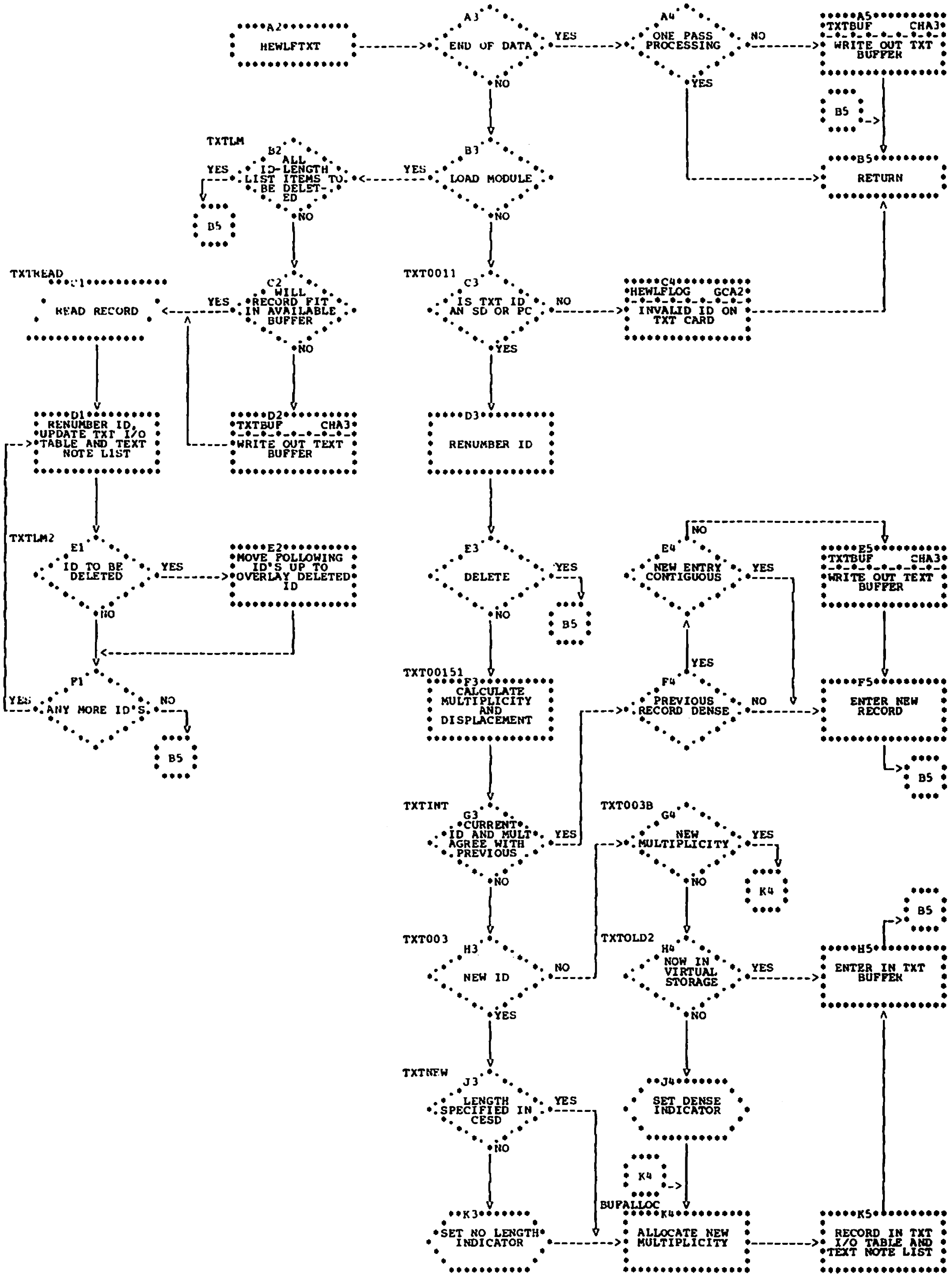


CHART CH. TXT WRITE ROUTINE (ON SYSUT1) (TXTBUF)

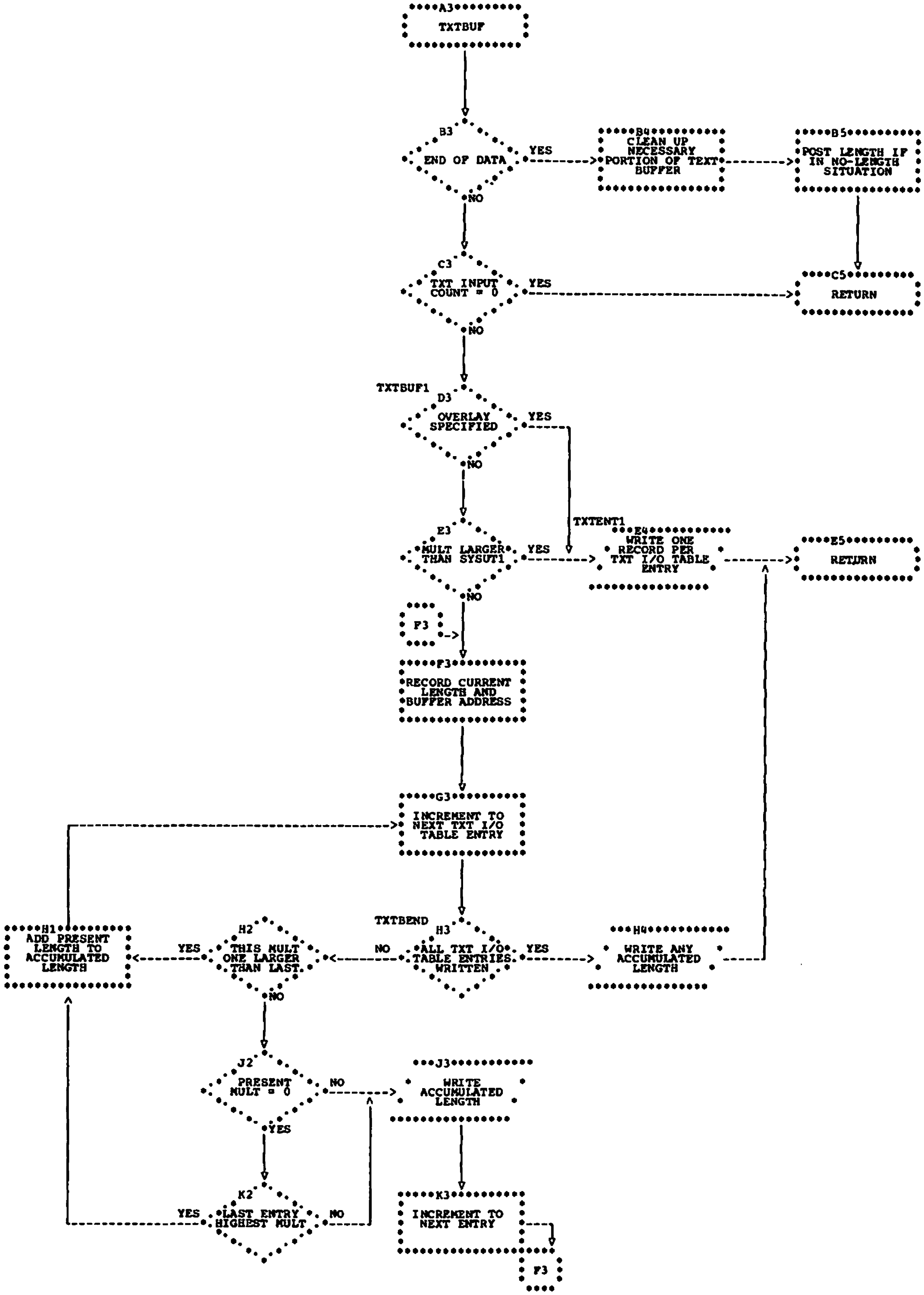

A3
TXTBUF
B3
END OF DATA
YES
NO
B4
CLEAN UP NECESSARY PORTION OF TEXT BUFFER
B5
POST LENGTH IF IN NO-LENGTH SITUATION
C3
TXT INPUT COUNT = 0
YES
NO
C5
RETURN
TXTBUF1
D3
OVERLAY SPECIFIED
YES
NO
TXTENT1
E4
WRITE ONE RECORD PER TXT I/O TABLE ENTRY
E3
MULT LARGER THAN SYSUT1
YES
NO
E5
RETURN
F3
F3
RECORD CURRENT LENGTH AND BUFFER ADDRESS
G3
INCREMENT TO NEXT TXT I/O TABLE ENTRY
TXTBEND
H3
ALL TXT I/O TABLE ENTRIES WRITTEN
NO
YES
H4
WRITE ANY ACCUMULATED LENGTH
H2
THIS MULT ONE LARGER THAN LAST.
YES
NO
H1
ADD PRESENT LENGTH TO ACCUMULATED LENGTH
J2
PRESENT MULT = 0
NO
YES
J3
WRITE ACCUMULATED LENGTH
K2
LAST ENTRY HIGHEST MULT
YES
NO
K3
INCREMENT TO NEXT ENTRY
F3

CHART CJ. RLD PROCESSOR (RLD001) (PART 1 OF 2)

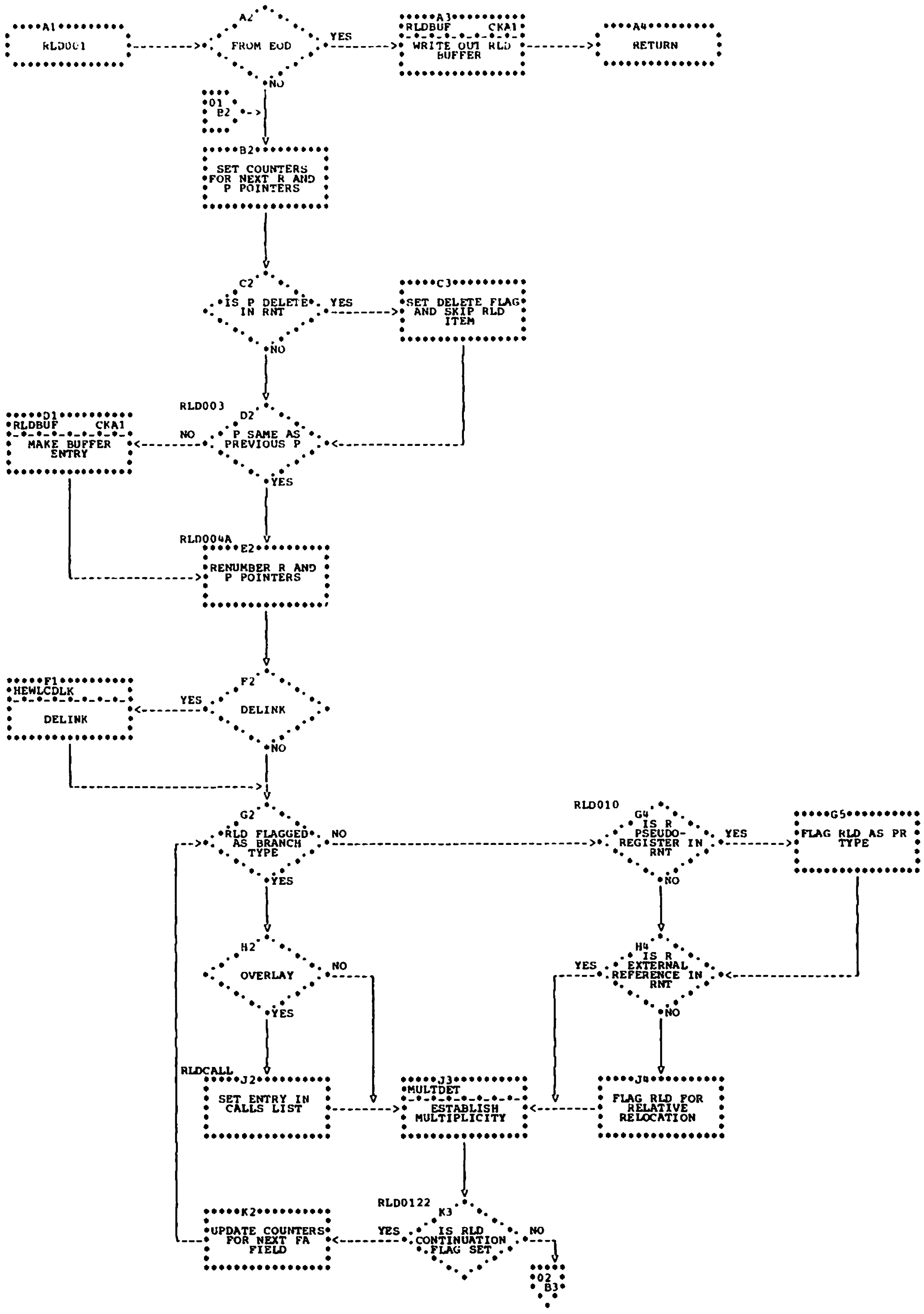


CHART CJ. RLD PROCESSOR (RLD001) (PART 2 OF 2)

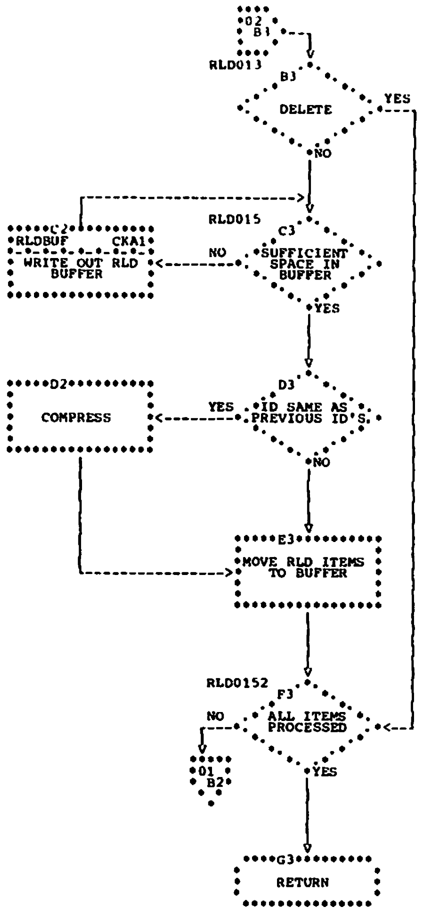


CHART CK. RLD WRITE ROUTINE (RLDBUF)

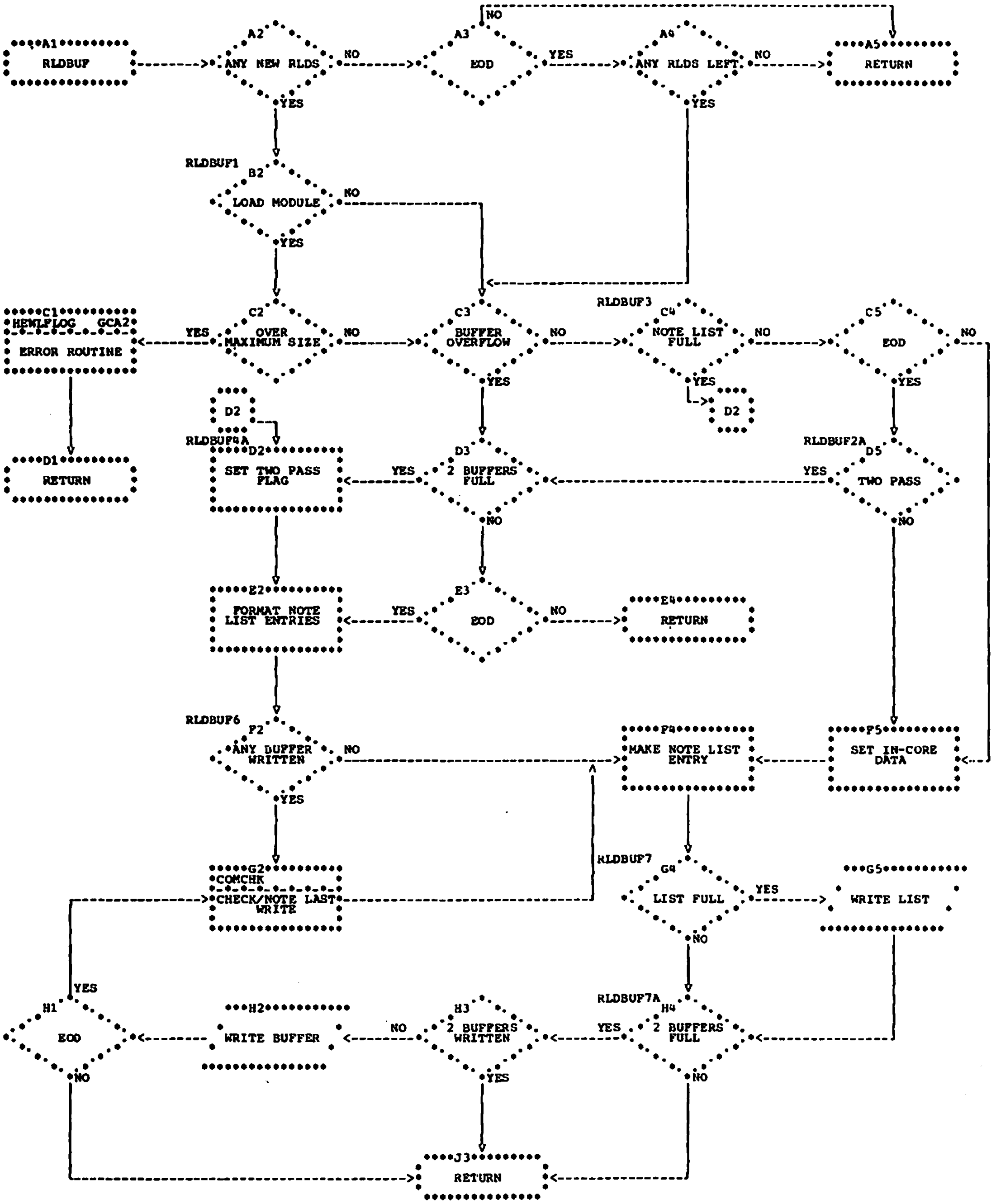


CHART CL. END PROCESSOR (HEWLFEND)

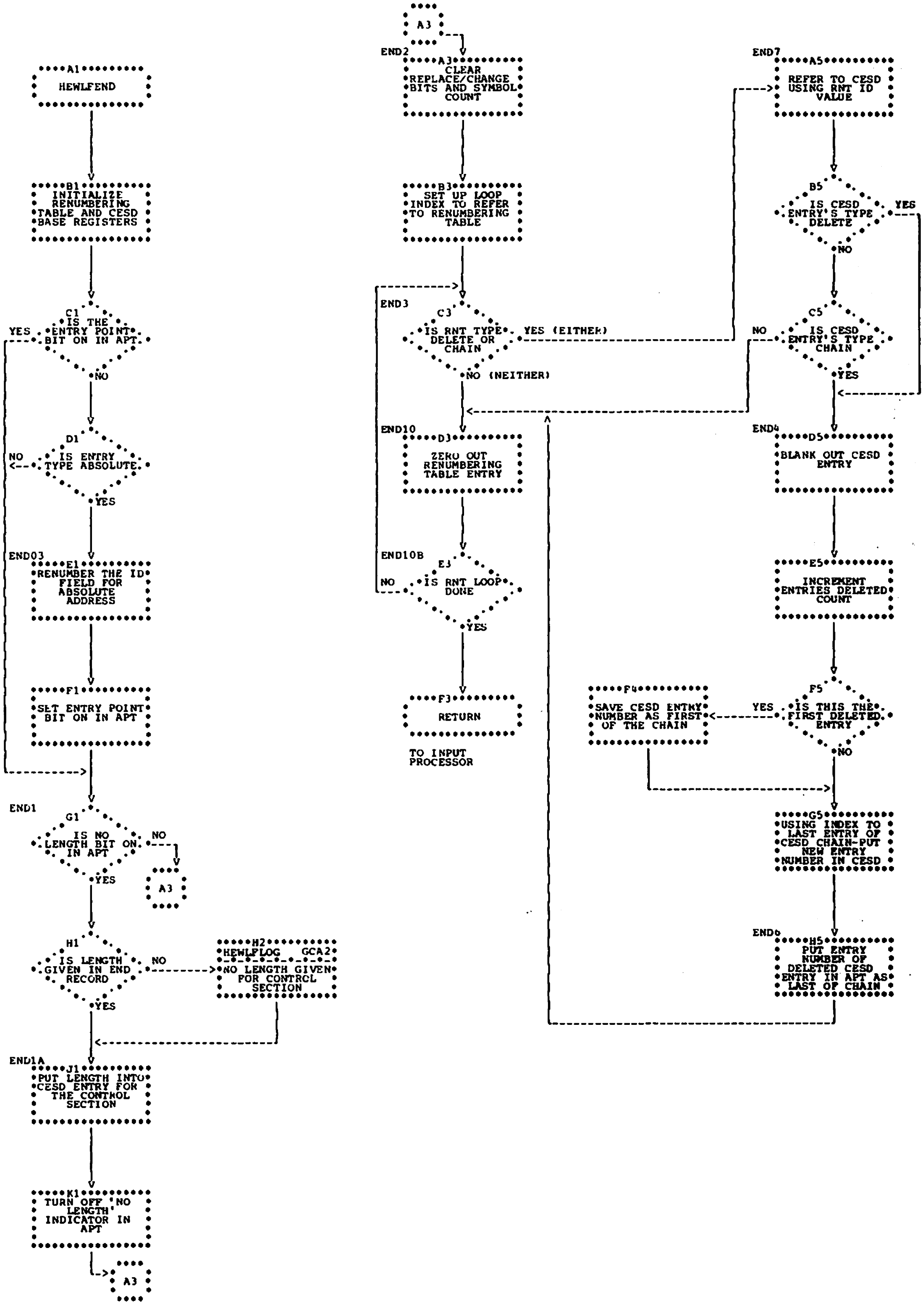

A1
HEWLFEND
B1
INITIALIZE RENUMBERING TABLE AND CESD BASE REGISTERS
C1
IS THE ENTRY POINT BIT ON IN APT
YES
NO
D1
IS ENTRY TYPE ABSOLUTE
NO
YES
END03
E1
RENUMBER THE ID FIELD FOR ABSOLUTE ADDRESS
F1
SET ENTRY POINT BIT ON IN APT
END1
G1
IS NO LENGTH BIT ON IN APT
NO
A3
YES
H1
IS LENGTH GIVEN IN END RECORD
NO
YES
H2
HEWLFLOG GCA2
NO LENGTH GIVEN FOR CONTROL SECTION
END1A
J1
PUT LENGTH INTO CESD ENTRY FOR THE CONTROL SECTION
K1
TURN OFF 'NO LENGTH' INDICATOR IN APT
A3
A3
END2
A3
CLEAR REPLACE/CHANGE BITS AND SYMBOL COUNT
B3
SET UP LOOP INDEX TO REFER TO RENUMBERING TABLE
END3
C3
IS RNT TYPE DELETE OR CHAIN
YES (EITHER)
NO (NEITHER)
END10
D3
ZERO OUT RENUMBERING TABLE ENTRY
END10B
E3
IS RNT LOOP DONE
NO
YES
F3
RETURN
TO INPUT PROCESSOR
END7
A5
REFER TO CESD USING RNT ID VALUE
B5
IS CESD ENTRY'S TYPE DELETE
YES
NO
C5
IS CESD ENTRY'S TYPE CHAIN
NO
YES
END4
D5
BLANK OUT CESD ENTRY
E5
INCREMENT ENTRIES DELETED COUNT
F5
IS THIS THE FIRST DELETED ENTRY
YES
NO
F4
SAVE CESD ENTRY NUMBER AS FIRST OF THE CHAIN
G5
USING INDEX TO LAST ENTRY OF CESD CHAIN-PUT NEW ENTRY NUMBER IN CESD
END6
H5
PUT ENTRY NUMBER OF DELETED CESD ENTRY IN APT AS LAST OF CHAIN

CHART CM. CSECT IDENTIFICATION RECORD PROCESSOR (HEWLFIDR)

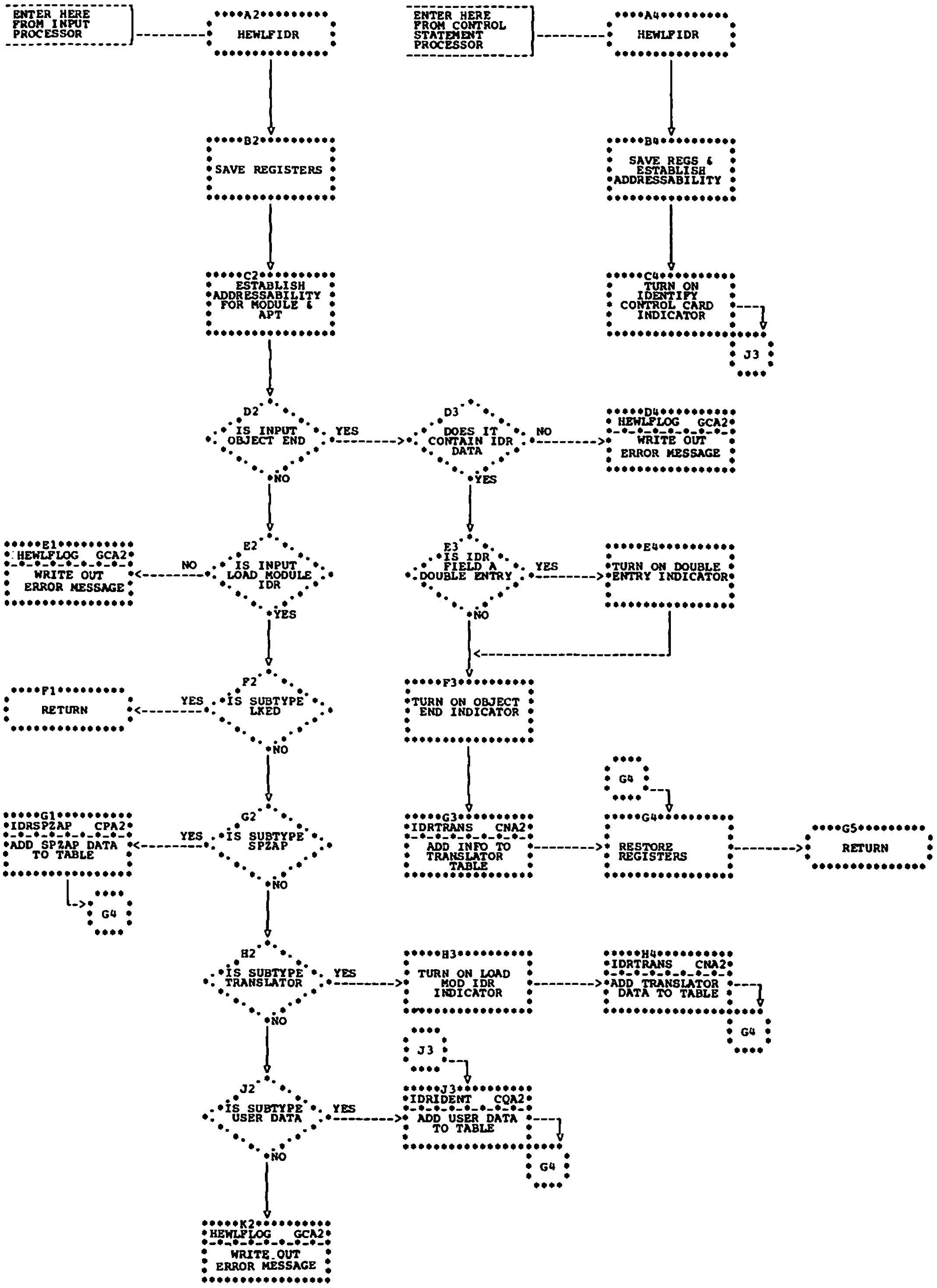

ENTER HERE FROM INPUT PROCESSOR
A2 HEWLFIDR
ENTER HERE FROM CONTROL STATEMENT PROCESSOR
A4 HEWLFIDR
B2 SAVE REGISTERS
B4 SAVE REGS & ESTABLISH ADDRESSABILITY
C2 ESTABLISH ADDRESSABILITY FOR MODULE & APT
C4 TURN ON IDENTIFY CONTROL CARD INDICATOR
J3
D2 IS INPUT OBJECT END
YES
D3 DOES IT CONTAIN IDR DATA
NO
D4 HEWLFLOG GCA2 WRITE OUT ERROR MESSAGE
NO
YES
E1 HEWLFLOG GCA2 WRITE OUT ERROR MESSAGE
NO
E2 IS INPUT LOAD MODULE IDR
E3 IS IDR FIELD A DOUBLE ENTRY
YES
E4 TURN ON DOUBLE ENTRY INDICATOR
YES
NO
F1 RETURN
YES
F2 IS SUBTYPE LKED
F3 TURN ON OBJECT END INDICATOR
NO
G4
G1 IDRSPZAP CPA2 ADD SPZAP DATA TO TABLE
YES
G2 IS SUBTYPE SPZAP
G3 IDRTRANS CNA2 ADD INFO TO TRANSLATOR TABLE
G4 RESTORE REGISTERS
G5 RETURN
G4
NO
H2 IS SUBTYPE TRANSLATOR
YES
H3 TURN ON LOAD MOD IDR INDICATOR
H4 IDRTRANS CNA2 ADD TRANSLATOR DATA TO TABLE
G4
NO
J3
J2 IS SUBTYPE USER DATA
YES
J3 IDRIDENT CQA2 ADD USER DATA TO TABLE
G4
NO
K2 HEWLFLOG GCA2 WRITE OUT ERROR MESSAGE

CHART CN. IDR TRANSLATOR DATA PROCESSOR (HEWLFIDR)

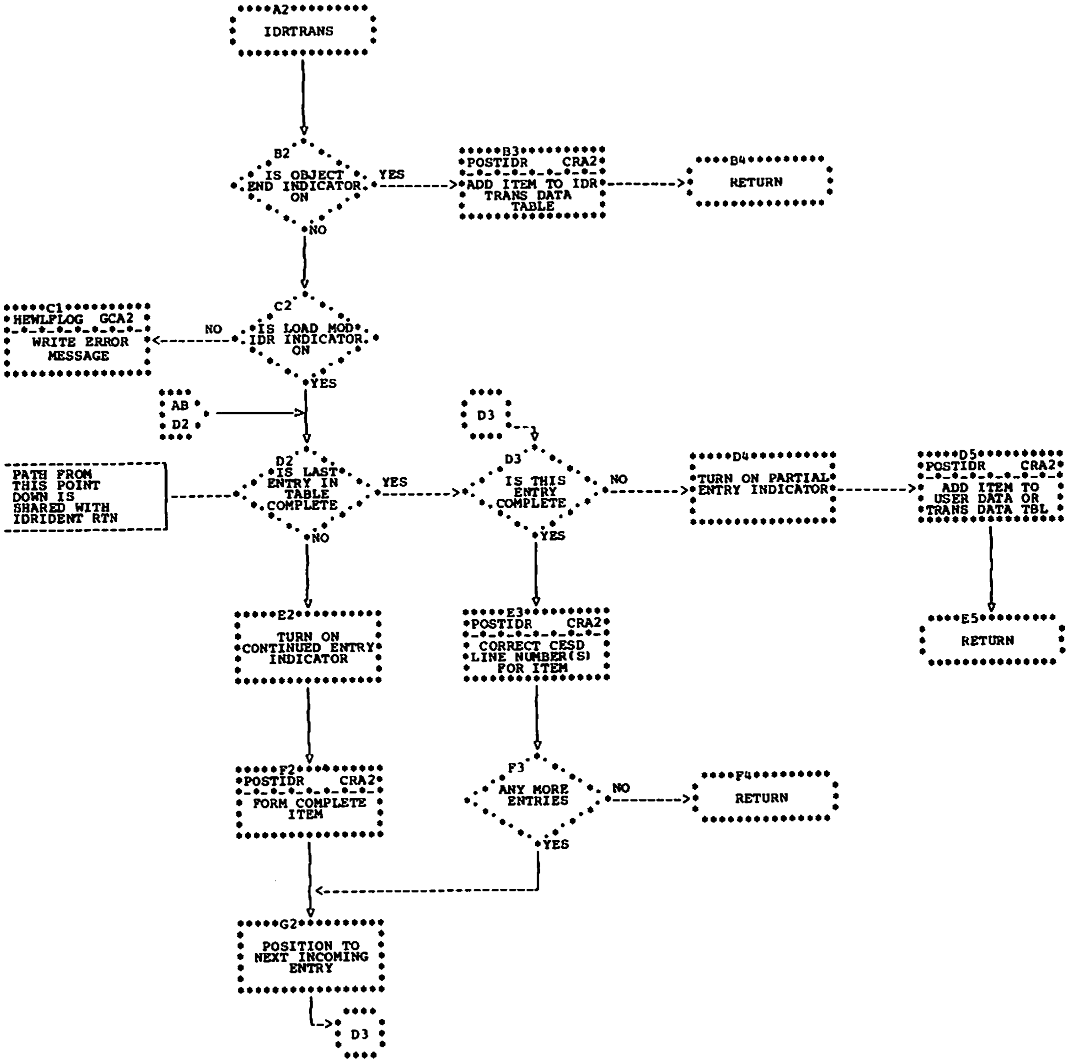

A2
IDRTRANS
B2
IS OBJECT END INDICATOR ON
YES
NO
B3
POSTIDR CRA2
ADD ITEM TO IDR TRANS DATA TABLE
B4
RETURN
C1
HEWLPLOG GCA2
WRITE ERROR MESSAGE
NO
C2
IS LOAD MOD IDR INDICATOR ON
YES
AB D2
D3
PATH FROM THIS POINT DOWN IS SHARED WITH IDRIDENT RTN
D2
IS LAST ENTRY IN TABLE COMPLETE
YES
NO
D3
IS THIS ENTRY COMPLETE
NO
YES
D4
TURN ON PARTIAL ENTRY INDICATOR
D5
POSTIDR CRA2
ADD ITEM TO USER DATA OR TRANS DATA TBL
E2
TURN ON CONTINUED ENTRY INDICATOR
E3
POSTIDR CRA2
CORRECT CESD LINE NUMBER(S) FOR ITEM
E5
RETURN
F2
POSTIDR CRA2
FORM COMPLETE ITEM
F3
ANY MORE ENTRIES
NO
YES
F4
RETURN
G2
POSITION TO NEXT INCOMING ENTRY
D3

CHART CP. IDRSPZAP DATA PROCESSOR (HEWLFIDR)

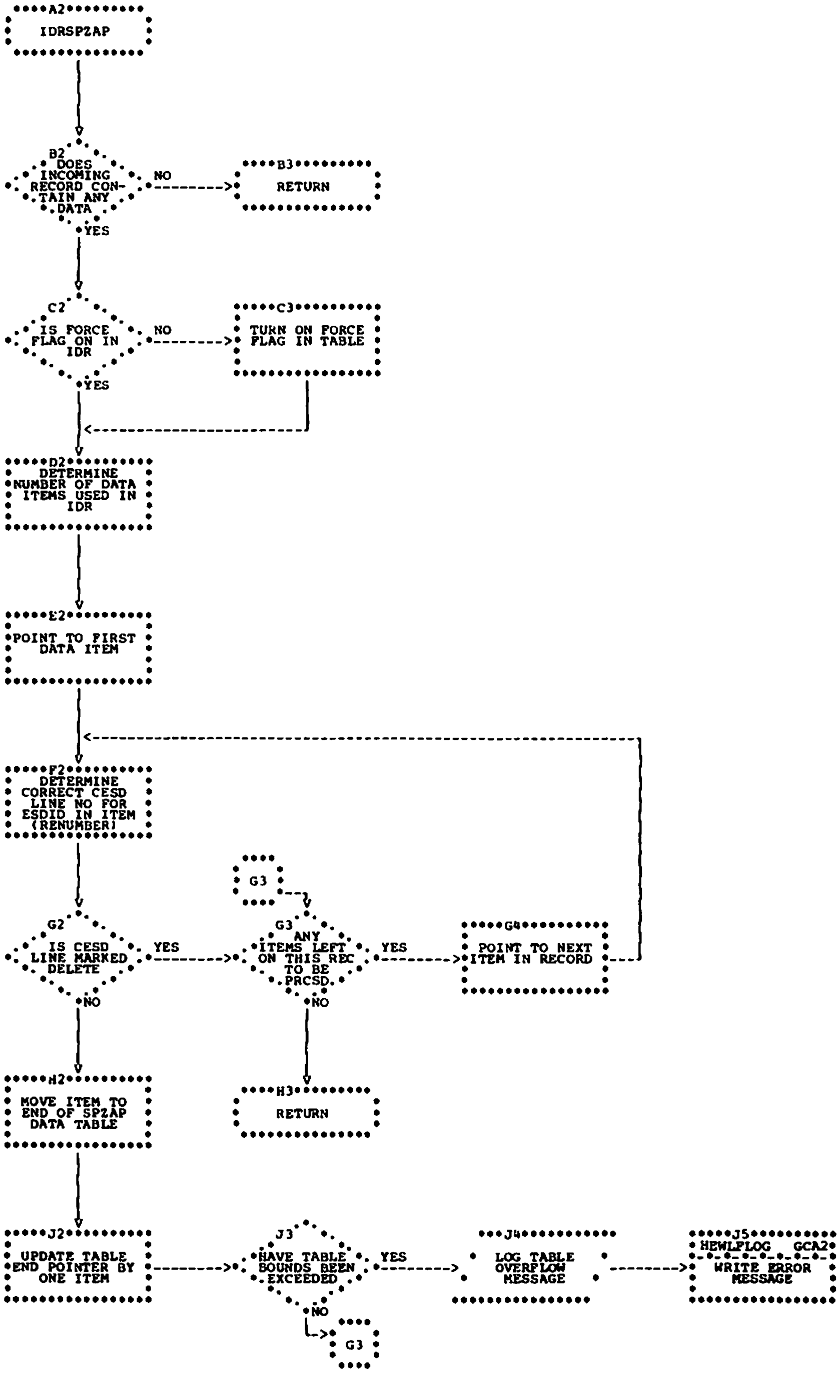

A2
IDRSPZAP
B2
DOES INCOMING RECORD CON-TAIN ANY DATA
NO
YES
B3
RETURN
C2
IS FORCE FLAG ON IN IDR
NO
YES
C3
TURN ON FORCE FLAG IN TABLE
D2
DETERMINE NUMBER OF DATA ITEMS USED IN IDR
E2
POINT TO FIRST DATA ITEM
F2
DETERMINE CORRECT CESD LINE NO FOR ESDID IN ITEM (RENUMBER)
G3
G2
IS CESD LINE MARKED DELETE
YES
NO
G3
ANY ITEMS LEFT ON THIS REC TO BE PRCSD.
YES
NO
G4
POINT TO NEXT ITEM IN RECORD
H2
MOVE ITEM TO END OF SPZAP DATA TABLE
H3
RETURN
J2
UPDATE TABLE END POINTER BY ONE ITEM
J3
HAVE TABLE BOUNDS BEEN EXCEEDED
YES
NO
J4
LOG TABLE OVERFLOW MESSAGE
J5
HEWLFLOG GCA2
WRITE ERROR MESSAGE
G3

CHART CQ. IDR IDENTIFY DATA PROCESSOR (HEWLFIDR)

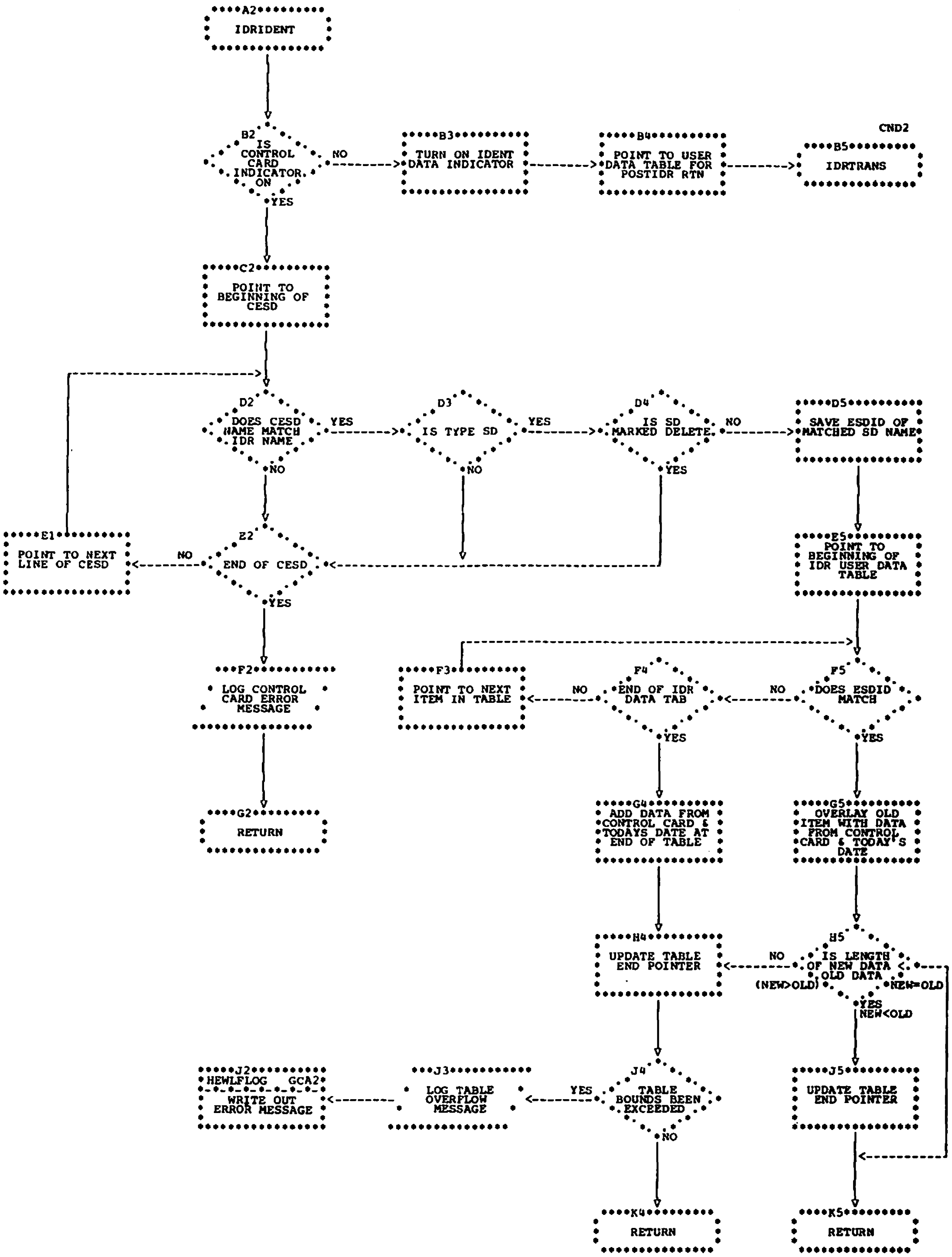

A2
IDRIDENT
B2
IS CONTROL CARD INDICATOR. ON
NO
B3
TURN ON IDENT DATA INDICATOR
B4
POINT TO USER DATA TABLE FOR POSTIDR RTN
CND2
B5
IDRTRANS
YES
C2
POINT TO BEGINNING OF CESD
D2
DOES CESD NAME MATCH IDR NAME
YES
D3
IS TYPE SD
YES
D4
IS SD MARKED DELETE.
NO
D5
SAVE ESDID OF MATCHED SD NAME
NO
NO
YES
E1
POINT TO NEXT LINE OF CESD
NO
E2
END OF CESD
YES
E5
POINT TO BEGINNING OF IDR USER DATA TABLE
F2
LOG CONTROL CARD ERROR MESSAGE
F3
POINT TO NEXT ITEM IN TABLE
NO
F4
END OF IDR DATA TAB
NO
F5
DOES ESDID MATCH
YES
YES
G2
RETURN
G4
ADD DATA FROM CONTROL CARD & TODAYS DATE AT END OF TABLE
G5
OVERLAY OLD ITEM WITH DATA FROM CONTROL CARD & TODAY'S DATE
H4
UPDATE TABLE END POINTER
NO
H5
IS LENGTH OF NEW DATA < OLD DATA
(NEW>OLD)
NEW=OLD
YES
NEW<OLD
J2
HEWLFLOG GCA2
WRITE OUT ERROR MESSAGE
J3
LOG TABLE OVERFLOW MESSAGE
YES
J4
TABLE BOUNDS BEEN EXCEEDED
NO
J5
UPDATE TABLE END POINTER
K4
RETURN
K5
RETURN

CHART CR. IDR USER DATA PROCESSOR (HEWLFIDR)

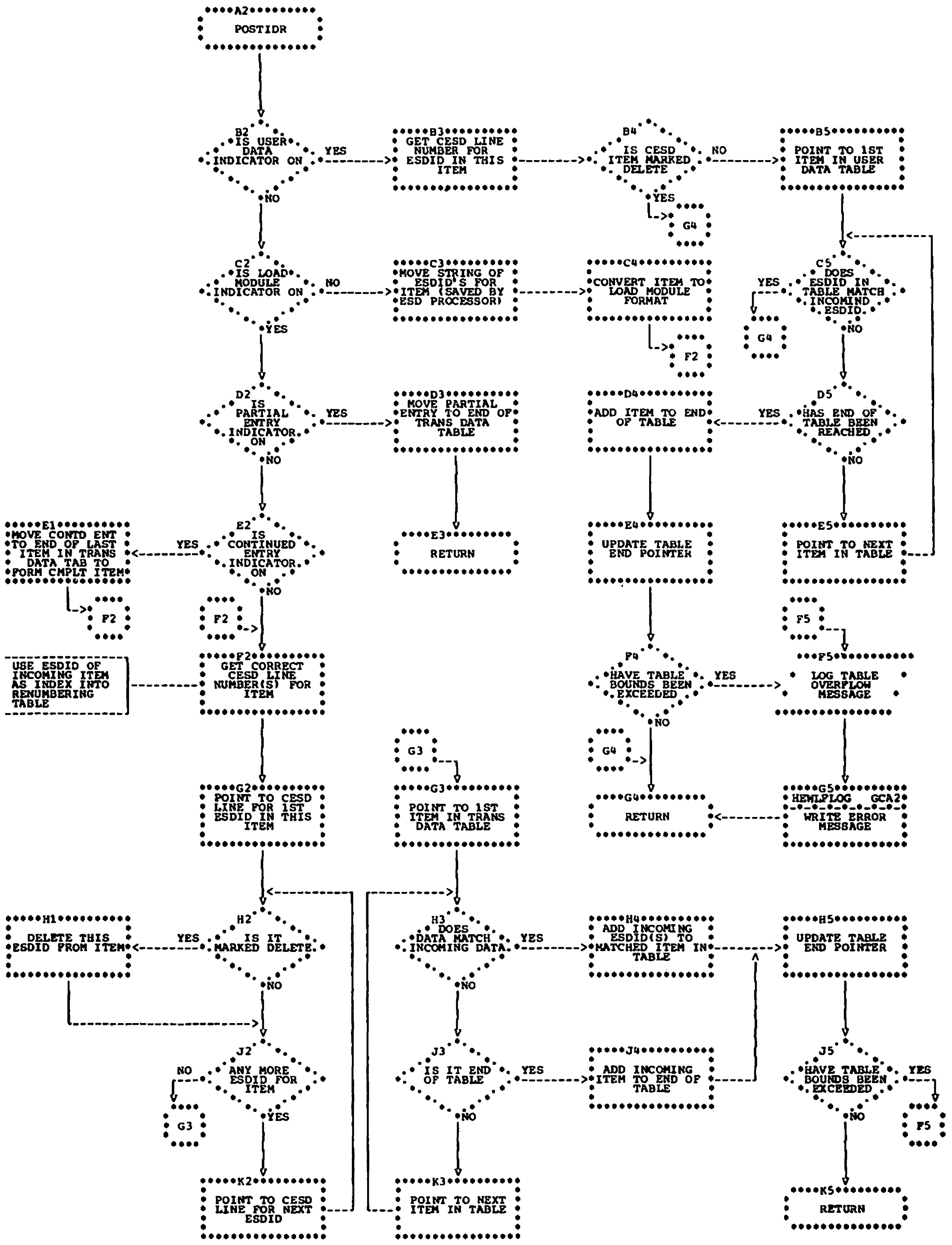

A2
POSTIDR
B2
IS USER DATA INDICATOR ON
YES
NO
B3
GET CESD LINE NUMBER FOR ESDID IN THIS ITEM
B4
IS CESD ITEM MARKED DELETE
NO
YES
G4
B5
POINT TO 1ST ITEM IN USER DATA TABLE
C2
IS LOAD MODULE INDICATOR ON
NO
YES
C3
MOVE STRING OF ESDID'S FOR ITEM (SAVED BY ESD PROCESSOR)
C4
CONVERT ITEM TO LOAD MODULE FORMAT
F2
C5
DOES ESDID IN TABLE MATCH INCOMIND ESDID.
YES
G4
NO
D2
IS PARTIAL ENTRY INDICATOR ON
YES
NO
D3
MOVE PARTIAL ENTRY TO END OF TRANS DATA TABLE
D4
ADD ITEM TO END OF TABLE
D5
HAS END OF TABLE BEEN REACHED
YES
NO
E1
MOVE CONTD ENT TO END OF LAST ITEM IN TRANS DATA TAB TO FORM CMPLT ITEM
F2
E2
IS CONTINUED ENTRY INDICATOR. ON
YES
NO
E3
RETURN
E4
UPDATE TABLE END POINTER
E5
POINT TO NEXT ITEM IN TABLE
F2
F5
USE ESDID OF INCOMING ITEM AS INDEX INTO RENUMBERING TABLE
F2
GET CORRECT CESD LINE NUMBER(S) FOR ITEM
F4
HAVE TABLE BOUNDS BEEN EXCEEDED
YES
NO
F5
LOG TABLE OVERFLOW MESSAGE
G3
G4
G2
POINT TO CESD LINE FOR 1ST ESDID IN THIS ITEM
G3
POINT TO 1ST ITEM IN TRANS DATA TABLE
G4
RETURN
G5
HEWLFLOG GCA2
WRITE ERROR MESSAGE
H1
DELETE THIS ESDID FROM ITEM
H2
IS IT MARKED DELETE
YES
NO
H3
DOES DATA MATCH INCOMING DATA
YES
NO
H4
ADD INCOMING ESDID(S) TO MATCHED ITEM IN TABLE
H5
UPDATE TABLE END POINTER
J2
ANY MORE ESDID FOR ITEM
NO
YES
G3
J3
IS IT END OF TABLE
YES
NO
J4
ADD INCOMING ITEM TO END OF TABLE
J5
HAVE TABLE BOUNDS BEEN EXCEEDED
YES
NO
F5
K2
POINT TO CESD LINE FOR NEXT ESDID
K3
POINT TO NEXT ITEM IN TABLE
K5
RETURN

CHART CS. CONTROL STATEMENT SCANNER (HEWLFSCN) (PART 1 OF 2)

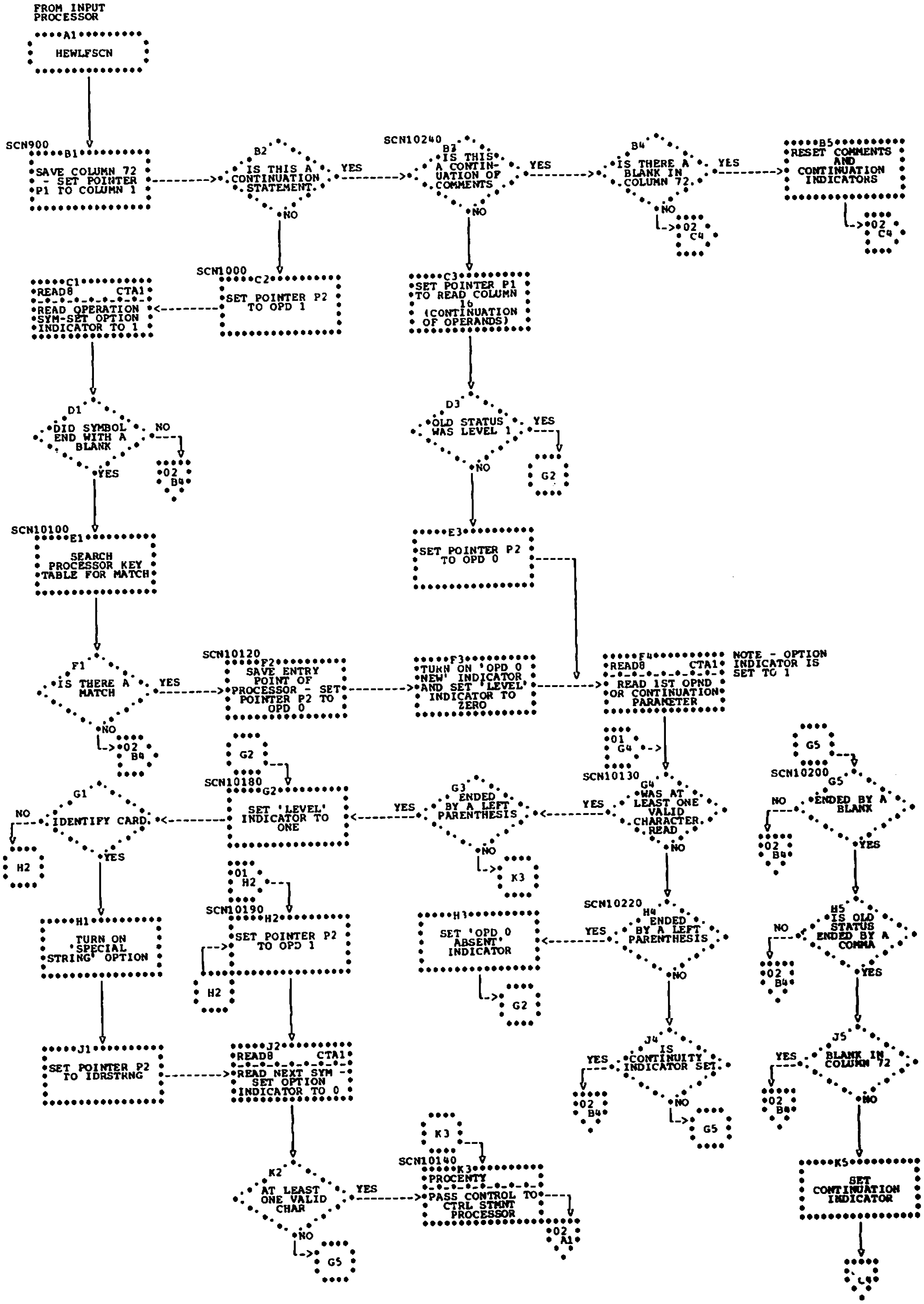

FROM INPUT PROCESSOR
A1 HEWLFSCN
SCN900 B1 SAVE COLUMN 72 - SET POINTER P1 TO COLUMN 1
B2 IS THIS A CONTINUATION STATEMENT
YES
NO
SCN10240 B3 IS THIS A CONTIN- UATION OF COMMENTS
YES
NO
B4 IS THERE A BLANK IN COLUMN 72
YES
NO
02 C4
B5 RESET COMMENTS AND CONTINUATION INDICATORS
02 C4
C1 READ8 CTA1 READ OPERATION SYM-SET OPTION INDICATOR TO 1
SCN1000 C2 SET POINTER P2 TO OPD 1
C3 SET POINTER P1 TO READ COLUMN 16 (CONTINUATION OF OPERANDS)
D1 DID SYMBOL END WITH A BLANK
NO
02 B4
YES
D3 OLD STATUS WAS LEVEL 1
YES
G2
NO
SCN10100 E1 SEARCH PROCESSOR KEY TABLE FOR MATCH
E3 SET POINTER P2 TO OPD 0
F1 IS THERE A MATCH
YES
NO
02 B4
SCN10120 F2 SAVE ENTRY POINT OF PROCESSOR - SET POINTER P2 TO OPD 0
F3 TURN ON 'OPD 0 NEW' INDICATOR AND SET 'LEVEL' INDICATOR TO ZERO
F4 READ8 CTA1 READ 1ST OPND OR CONTINUATION PARAMETER
NOTE - OPTION INDICATOR IS SET TO 1
01 G4
G2
SCN10180 G2 SET 'LEVEL' INDICATOR TO ONE
G1 IDENTIFY CARD
NO
H2
YES
G3 ENDED BY A LEFT PARENTHESIS
YES
NO
K3
SCN10130 G4 WAS AT LEAST ONE VALID CHARACTER READ
YES
NO
G5
SCN10200 G5 ENDED BY A BLANK
NO
02 B4
YES
01 H2
SCN10190 H2 SET POINTER P2 TO OPD 1
H2
H1 TURN ON 'SPECIAL STRING' OPTION
H3 SET 'OPD 0 ABSENT' INDICATOR
G2
SCN10220 H4 ENDED BY A LEFT PARENTHESIS
YES
NO
H5 IS OLD STATUS ENDED BY A COMMA
NO
02 B4
YES
J1 SET POINTER P2 TO IDRSTRNG
J2 READ8 CTA1 READ NEXT SYM - SET OPTION INDICATOR TO 0
J4 IS CONTINUITY INDICATOR SET
YES
02 B4
NO
G5
J5 BLANK IN COLUMN 72
YES
02 B4
NO
K3
SCN10140 K3 PROCENTY PASS CONTROL TO CTRL STMNT PROCESSOR
02 A1
K2 AT LEAST ONE VALID CHAR
YES
NO
G5
K5 SET CONTINUATION INDICATOR
L4

CHART CS. CONTROL STATEMENT SCANNER (HEWLFSCN) (PART 2 OF 2)

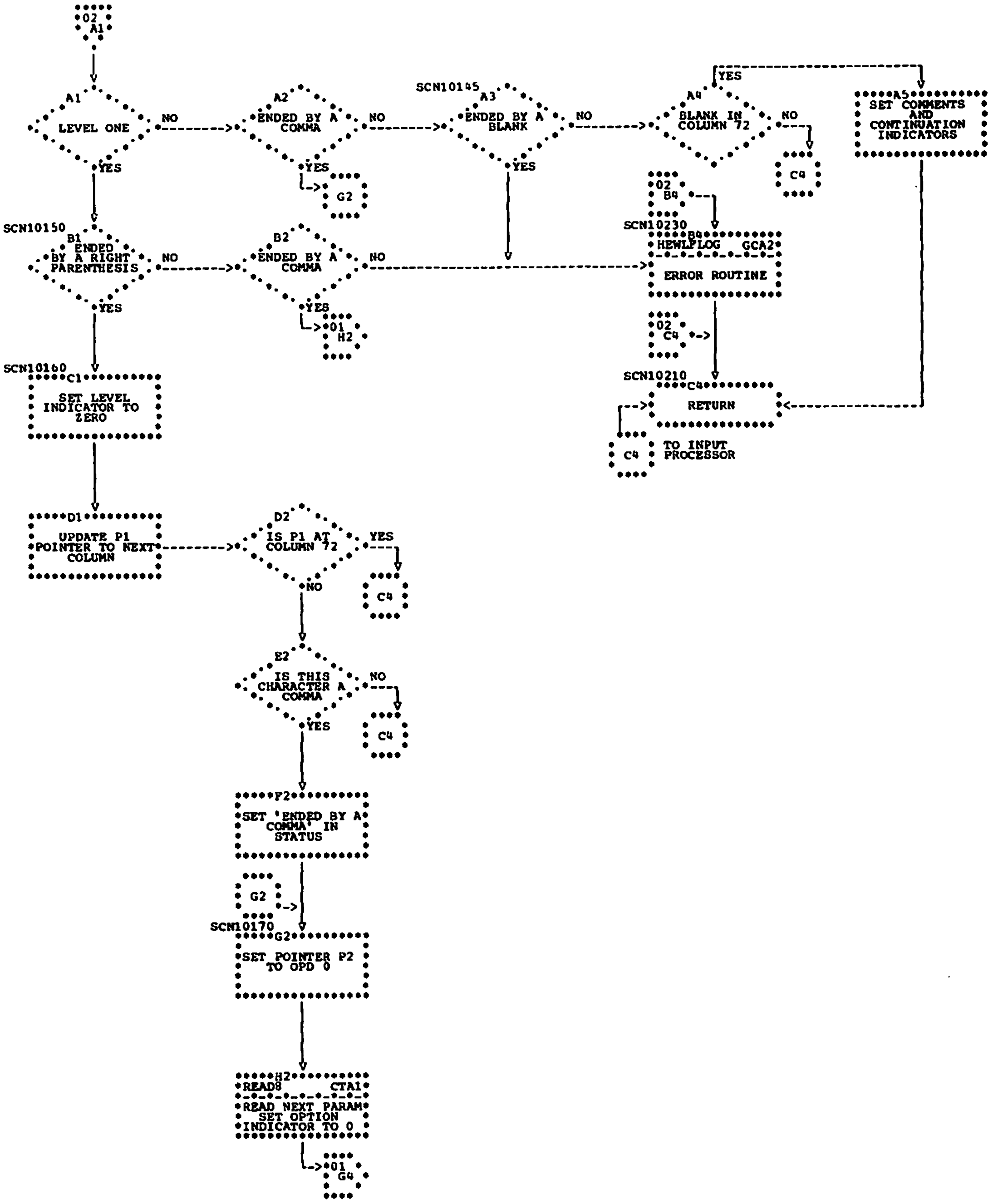


CHART CT. READ8 ROUTINE

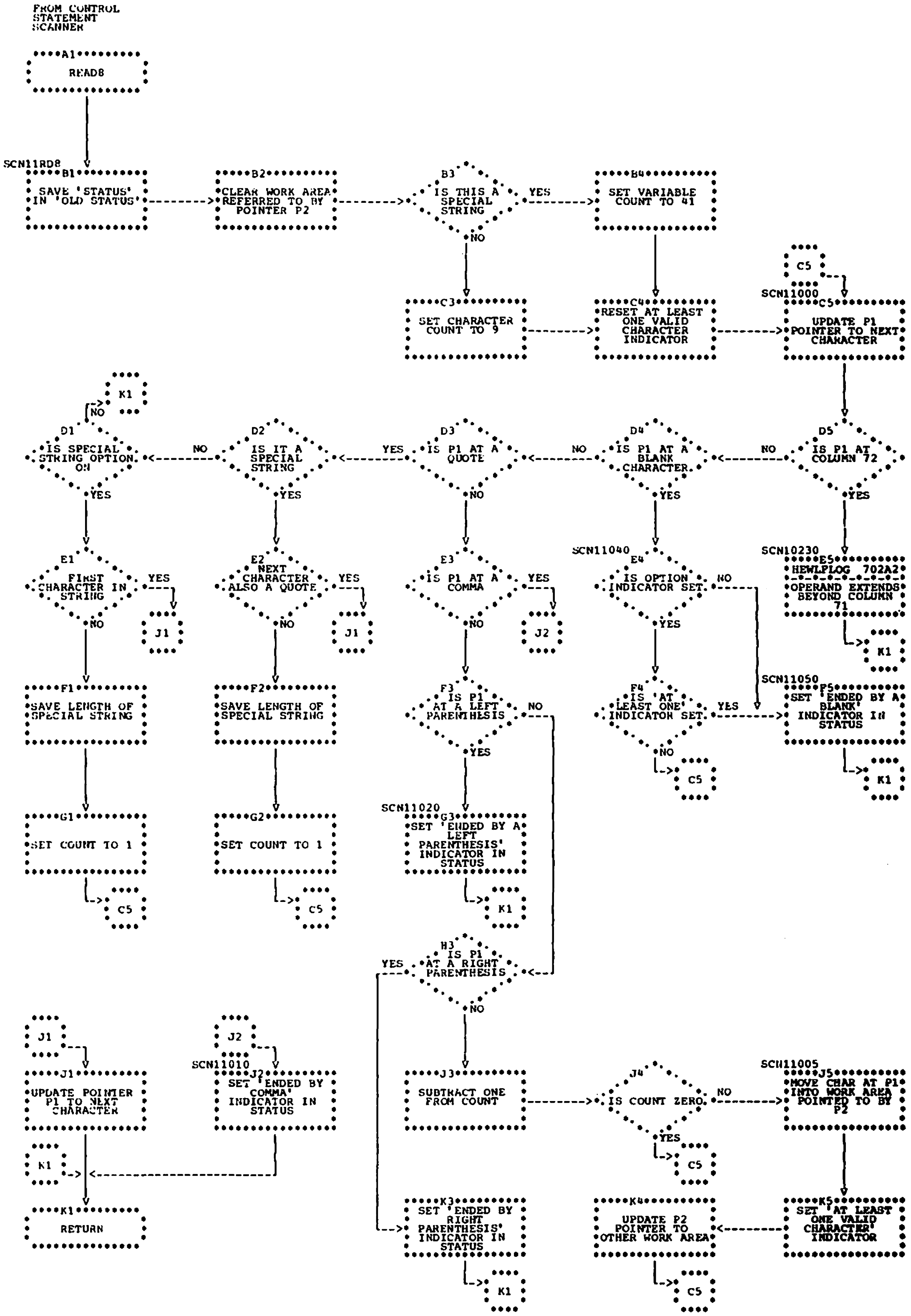

FROM CONTROL STATEMENT SCANNER
A1 READ8
SCN11RD8
B1 SAVE 'STATUS' IN 'OLD STATUS'
B2 CLEAR WORK AREA REFERRED TO BY POINTER P2
B3 IS THIS A SPECIAL STRING
YES
NO
B4 SET VARIABLE COUNT TO 41
C3 SET CHARACTER COUNT TO 9
C4 RESET AT LEAST ONE VALID CHARACTER INDICATOR
C5
SCN11000
C5 UPDATE P1 POINTER TO NEXT CHARACTER
K1
D1 IS SPECIAL STRING OPTION ON
D2 IS IT A SPECIAL STRING
D3 IS P1 AT A QUOTE
D4 IS P1 AT A BLANK CHARACTER
D5 IS P1 AT COLUMN 72
E1 FIRST CHARACTER IN STRING
J1
E2 NEXT CHARACTER ALSO A QUOTE
J1
E3 IS P1 AT A COMMA
J2
SCN11040
E4 IS OPTION INDICATOR SET
SCN10230
E5 HEWLPLOG 702A2 OPERAND EXTENDS BEYOND COLUMN 71
K1
F1 SAVE LENGTH OF SPECIAL STRING
F2 SAVE LENGTH OF SPECIAL STRING
F3 IS P1 AT A LEFT PARENTHESIS
F4 IS 'AT LEAST ONE' INDICATOR SET
C5
SCN11050
F5 SET 'ENDED BY A BLANK' INDICATOR IN STATUS
K1
G1 SET COUNT TO 1
C5
G2 SET COUNT TO 1
C5
SCN11020
G3 SET 'ENDED BY A LEFT PARENTHESIS' INDICATOR IN STATUS
K1
H3 IS P1 AT A RIGHT PARENTHESIS
J1
J1 UPDATE POINTER P1 TO NEXT CHARACTER
J2
SCN11010
J2 SET 'ENDED BY COMMA' INDICATOR IN STATUS
J3 SUBTRACT ONE FROM COUNT
J4 IS COUNT ZERO
C5
SCN11005
J5 MOVE CHAR AT P1 INTO WORK AREA POINTED TO BY P2
K1
K1 RETURN
K3 SET 'ENDED BY RIGHT PARENTHESIS' INDICATOR IN STATUS
K1
K4 UPDATE P2 POINTER TO OTHER WORK AREA
C5
K5 SET 'AT LEAST ONE VALID CHARACTER' INDICATOR

CHART CU. INCLUDE PROCESSOR (HEWLFINC)

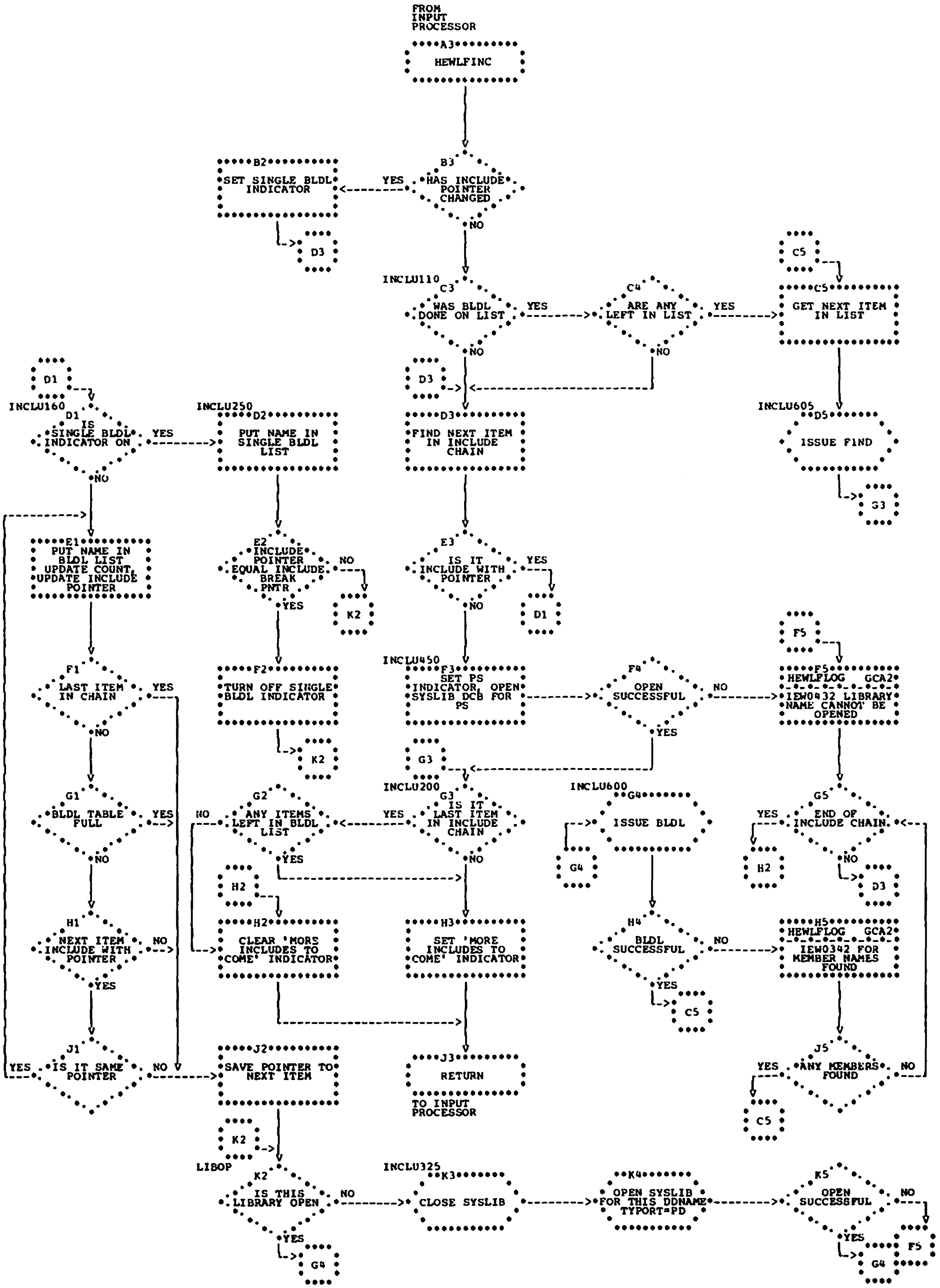

FROM INPUT PROCESSOR
A3
HEWLFINC
B3
HAS INCLUDE POINTER CHANGED
YES
NO
B2
SET SINGLE BLDL INDICATOR
D3
C5
INCLU110
C3
WAS BLDL DONE ON LIST
YES
NO
C4
ARE ANY LEFT IN LIST
YES
NO
C5
GET NEXT ITEM IN LIST
D1
INCLU160
D1
IS SINGLE BLDL INDICATOR ON
YES
NO
INCLU250
D2
PUT NAME IN SINGLE BLDL LIST
D3
D3
FIND NEXT ITEM IN INCLUDE CHAIN
INCLU605
D5
ISSUE FIND
G3
E1
PUT NAME IN BLDL LIST UPDATE COUNT, UPDATE INCLUDE POINTER
E2
INCLUDE POINTER EQUAL INCLUDE BREAK PNTR
NO
YES
K2
E3
IS IT INCLUDE WITH POINTER
YES
NO
D1
F5
F1
LAST ITEM IN CHAIN
YES
NO
F2
TURN OFF SINGLE BLDL INDICATOR
K2
INCLU450
F3
SET PS INDICATOR, OPEN SYSLIB DCB FOR PS
F4
OPEN SUCCESSFUL
NO
YES
F5
HEWLFLOG GCA2
IEW0432 LIBRARY NAME CANNOT BE OPENED
G3
G1
BLDL TABLE FULL
YES
NO
G2
ANY ITEMS LEFT IN BLDL LIST
NO
YES
INCLU200
G3
IS IT LAST ITEM IN INCLUDE CHAIN
YES
NO
INCLU600
G4
ISSUE BLDL
G4
G5
END OF INCLUDE CHAIN
YES
NO
H2
D3
H2
H1
NEXT ITEM INCLUDE WITH POINTER
NO
YES
H2
CLEAR 'MORE INCLUDES TO COME' INDICATOR
H3
SET 'MORE INCLUDES TO COME' INDICATOR
H4
BLDL SUCCESSFUL
NO
YES
C5
H5
HEWLFLOG GCA2
IEW0342 FOR MEMBER NAMES FOUND
J1
IS IT SAME POINTER
YES
NO
J2
SAVE POINTER TO NEXT ITEM
J3
RETURN
TO INPUT PROCESSOR
J5
ANY MEMBERS FOUND
YES
NO
C5
K2
LIBOP
K2
IS THIS LIBRARY OPEN
NO
YES
G4
INCLU325
K3
CLOSE SYSLIB
K4
OPEN SYSLIB FOR THIS DDNAME TYPORT=PD
K5
OPEN SUCCESSFUL
NO
YES
F5
G4

CHART CV. AUTOMATIC LIBRARY CALL PROCESSOR (HEWLCAUT) (PART 1 OF 2)

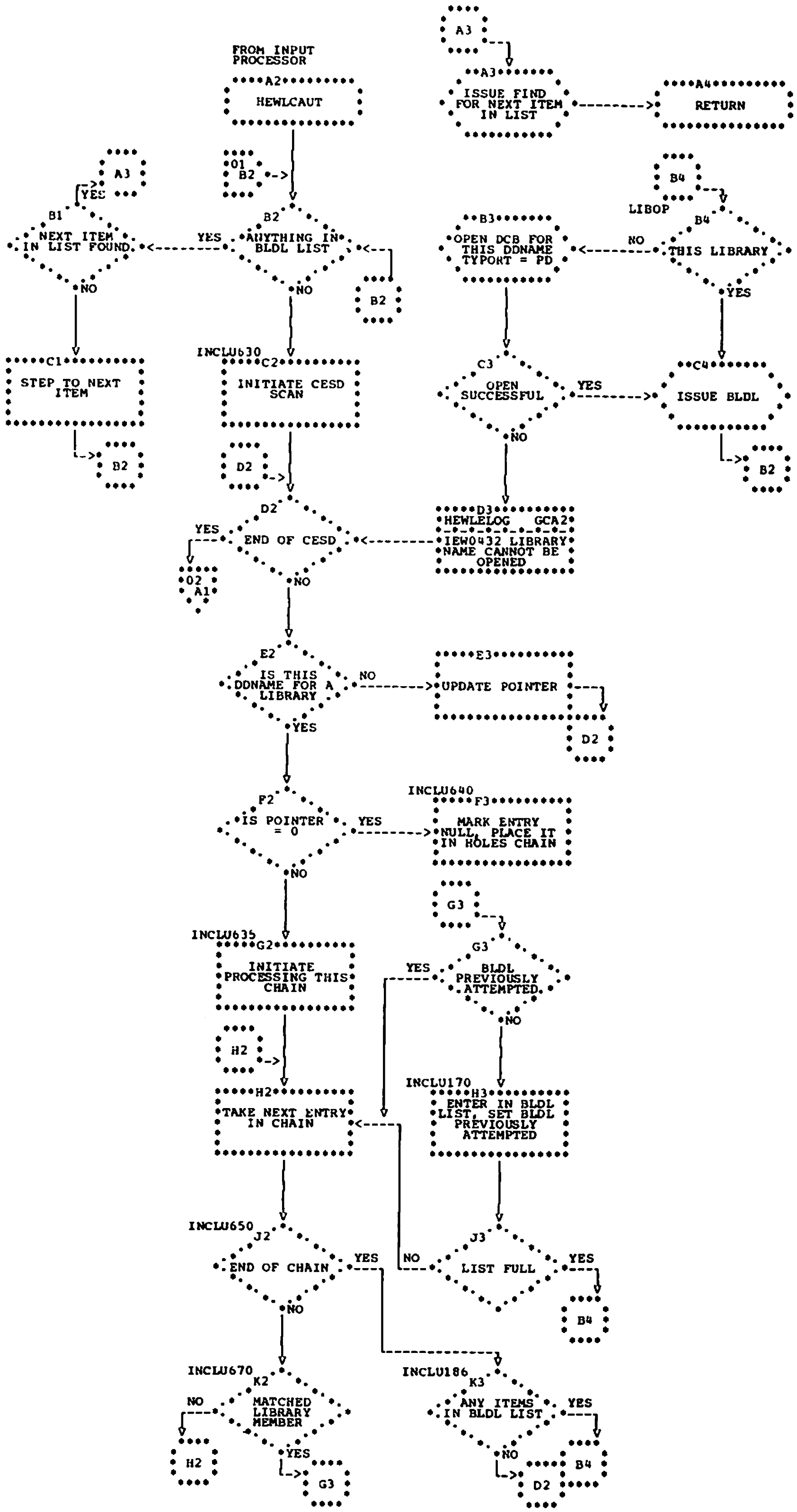


CHART CV. AUTOMATIC LIBRARY CALL PROCESSOR (HEWLCAUT) (PART 2 OF 2)

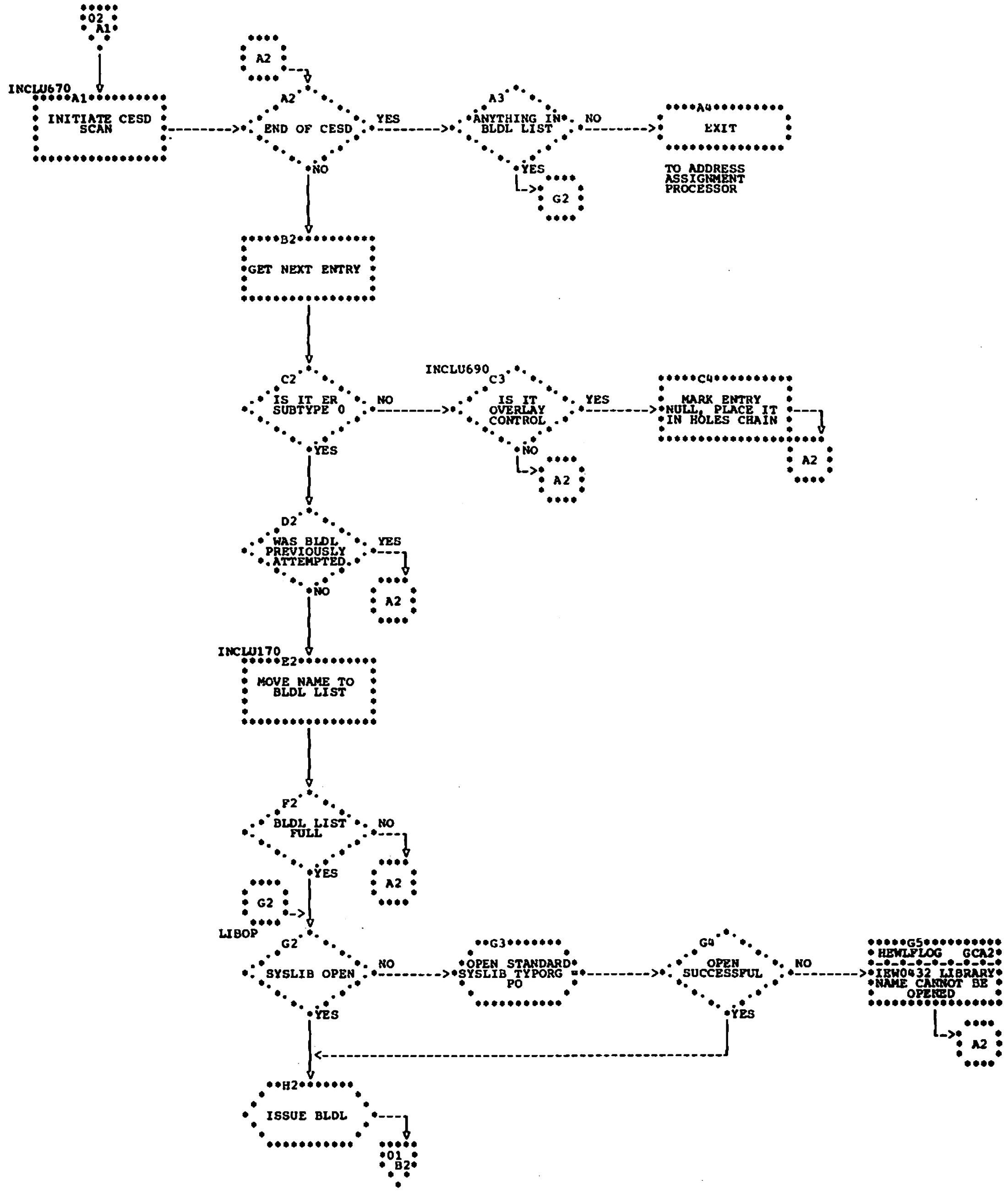


CHART DA. ADDRESS ASSIGNMENT PROCESSOR (HEWLFADA)

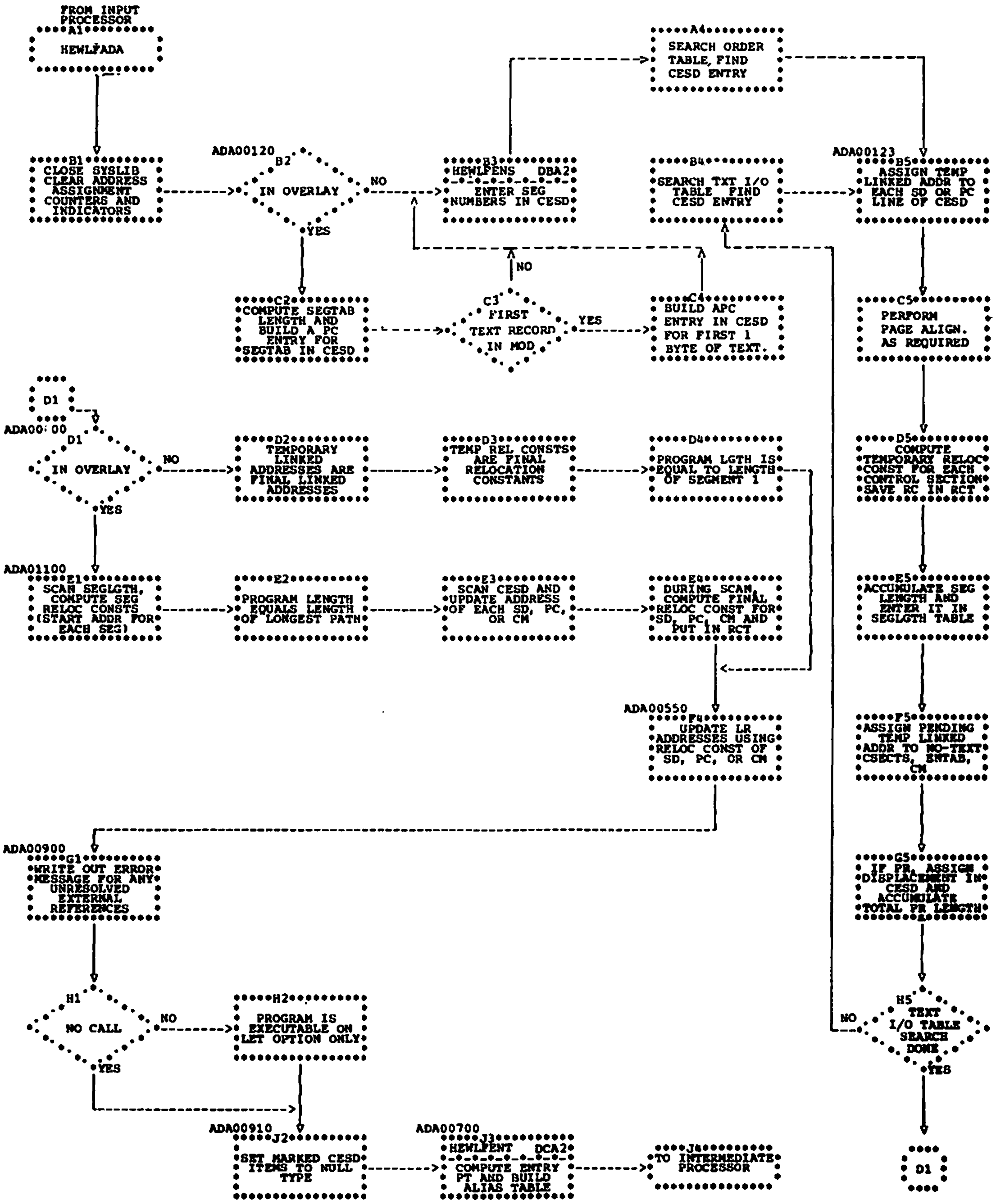

FROM INPUT PROCESSOR
A1
HEWLFADA
B1
CLOSE SYSLIB CLEAR ADDRESS ASSIGNMENT COUNTERS AND INDICATORS
ADA00120
B2
IN OVERLAY
NO
YES
B3
HEWLFENS DBA2
ENTER SEG NUMBERS IN CESD
A4
SEARCH ORDER TABLE, FIND CESD ENTRY
B4
SEARCH TXT I/O TABLE FIND CESD ENTRY
ADA00123
B5
ASSIGN TEMP LINKED ADDR TO EACH SD OR PC LINE OF CESD
C2
COMPUTE SEGTAB LENGTH AND BUILD A PC ENTRY FOR SEGTAB IN CESD
C3
FIRST TEXT RECORD IN MOD
NO
YES
C4
BUILD APC ENTRY IN CESD FOR FIRST 1 BYTE OF TEXT.
C5
PERFORM PAGE ALIGN. AS REQUIRED
D1
ADA00: 00
D1
IN OVERLAY
NO
YES
D2
TEMPORARY LINKED ADDRESSES ARE FINAL LINKED ADDRESSES
D3
TEMP REL CONSTS ARE FINAL RELOCATION CONSTANTS
D4
PROGRAM LGTH IS EQUAL TO LENGTH OF SEGMENT 1
D5
COMPUTE TEMPORARY RELOC CONST FOR EACH CONTROL SECTION SAVE RC IN RCT
ADA01100
E1
SCAN SEGLGTH, COMPUTE SEG RELOC CONSTS (START ADDR FOR EACH SEG)
E2
PROGRAM LENGTH EQUALS LENGTH OF LONGEST PATH
E3
SCAN CESD AND UPDATE ADDRESS OF EACH SD, PC, OR CM
E4
DURING SCAN, COMPUTE FINAL RELOC CONST FOR SD, PC, CM AND PUT IN RCT
E5
ACCUMULATE SEG LENGTH AND ENTER IT IN SEGLGTH TABLE
ADA00550
F4
UPDATE LR ADDRESSES USING RELOC CONST OF SD, PC, OR CM
F5
ASSIGN PENDING TEMP LINKED ADDR TO NO-TEXT CSECTS, ENTAB, CM
ADA00900
G1
WRITE OUT ERROR MESSAGE FOR ANY UNRESOLVED EXTERNAL REFERENCES
G5
IF PR, ASSIGN DISPLACEMENT IN CESD AND ACCUMULATE TOTAL PR LENGTH
H1
NO CALL
NO
YES
H2
PROGRAM IS EXECUTABLE ON LET OPTION ONLY
H5
TEXT I/O TABLE SEARCH DONE
NO
YES
ADA00910
J2
SET MARKED CESD ITEMS TO NULL TYPE
ADA00700
J3
HEWLFENT DCA2
COMPUTE ENTRY PT AND BUILD ALIAS TABLE
J4
TO INTERMEDIATE PROCESSOR
D1

CHART DB. ENTAB SIZE DETERMINATION ROUTINE (HEWLFENS)

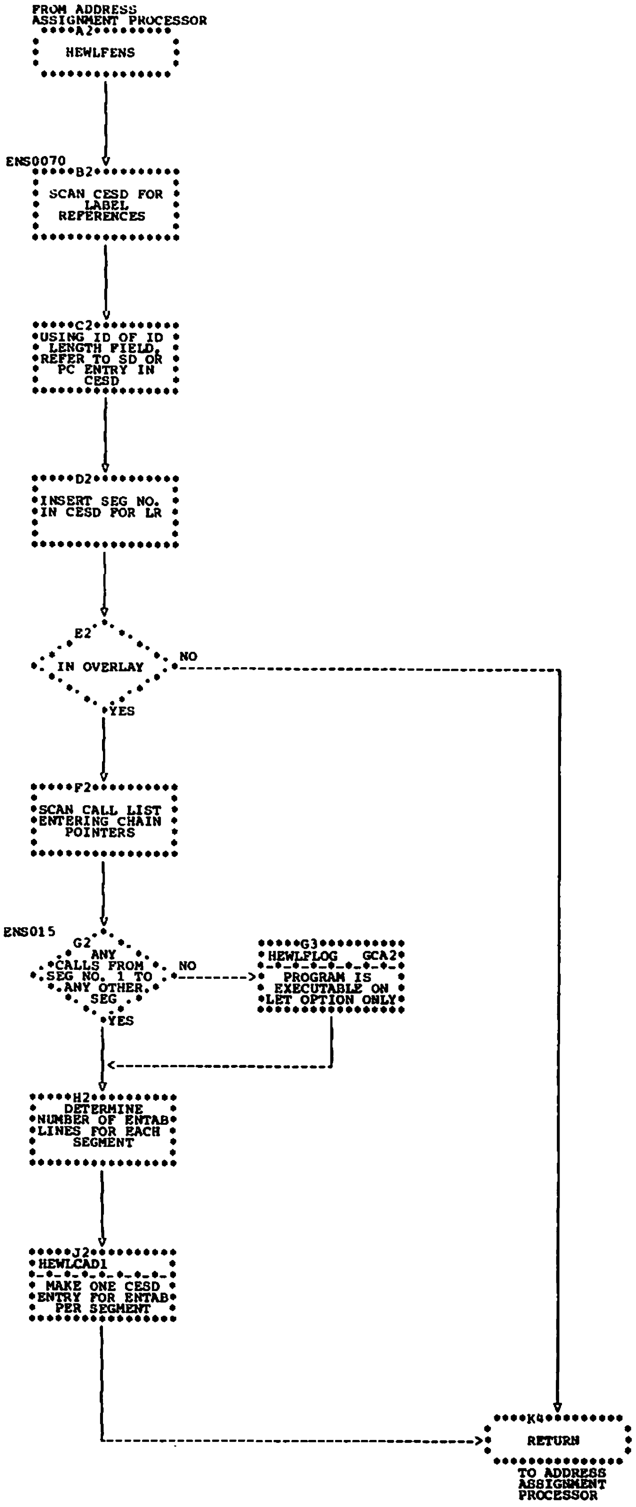

FROM ADDRESS ASSIGNMENT PROCESSOR
A2
HEWLFENS
ENS0070
B2
SCAN CESD FOR LABEL REFERENCES
C2
USING ID OF ID LENGTH FIELD, REFER TO SD OR PC ENTRY IN CESD
D2
INSERT SEG NO. IN CESD FOR LR
E2
IN OVERLAY
NO
YES
F2
SCAN CALL LIST ENTERING CHAIN POINTERS
ENS015
G2
ANY CALLS FROM SEG NO. 1 TO ANY OTHER SEG
NO
YES
G3
HEWLFLOG GCA2
PROGRAM IS EXECUTABLE ON LET OPTION ONLY
H2
DETERMINE NUMBER OF ENTAB LINES FOR EACH SEGMENT
J2
HEWLCAD1
MAKE ONE CESD ENTRY FOR ENTAB PER SEGMENT
K4
RETURN
TO ADDRESS ASSIGNMENT PROCESSOR

CHART DC. ENTRY PROCESSOR (HEWLFENT) (PART 1 OF 2)

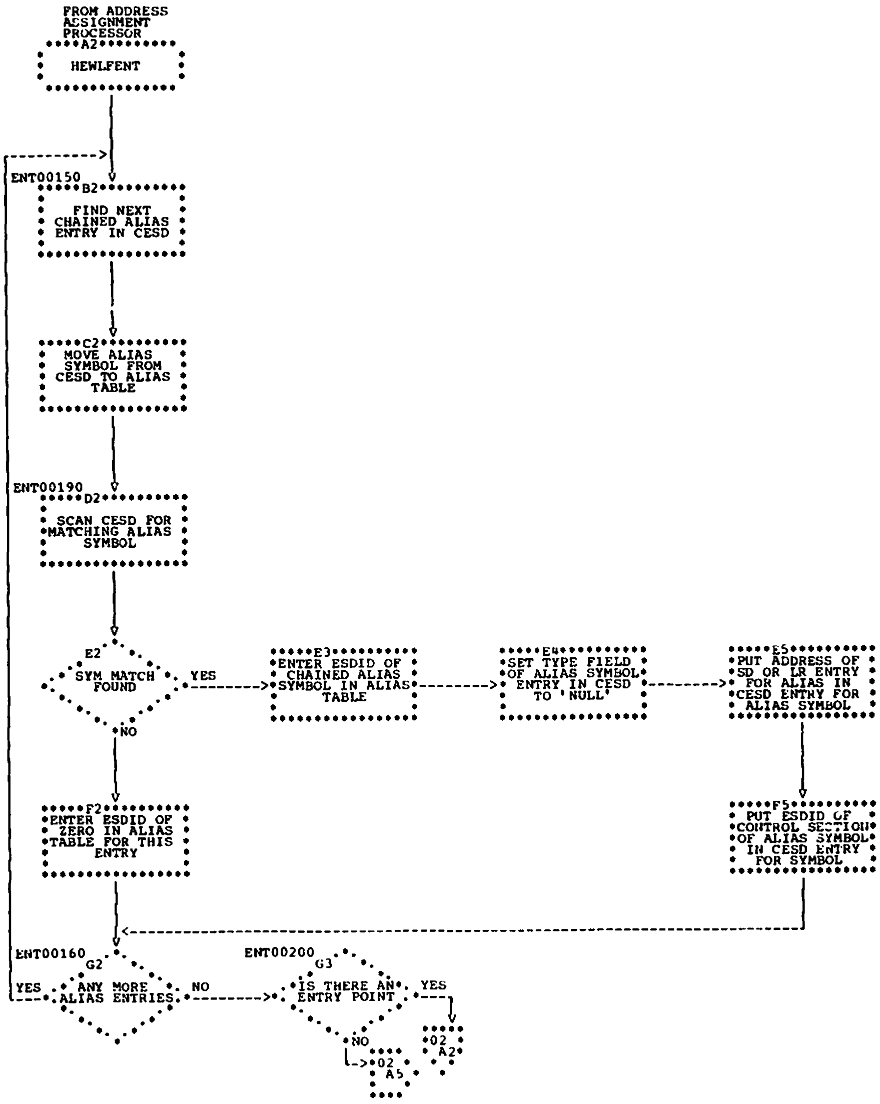

FROM ADDRESS ASSIGNMENT PROCESSOR
A2
HEWLFENT
ENT00150
B2
FIND NEXT CHAINED ALIAS ENTRY IN CESD
C2
MOVE ALIAS SYMBOL FROM CESD TO ALIAS TABLE
ENT00190
D2
SCAN CESD FOR MATCHING ALIAS SYMBOL
E2
SYM MATCH FOUND
YES
NO
E3
ENTER ESDID OF CHAINED ALIAS SYMBOL IN ALIAS TABLE
E4
SET TYPE FIELD OF ALIAS SYMBOL ENTRY IN CESD TO 'NULL'
E5
PUT ADDRESS OF SD OR LR ENTRY FOR ALIAS IN CESD ENTRY FOR ALIAS SYMBOL
F2
ENTER ESDID OF ZERO IN ALIAS TABLE FOR THIS ENTRY
F5
PUT ESDID OF CONTROL SECTION OF ALIAS SYMBOL IN CESD ENTRY FOR SYMBOL
ENT00160
G2
ANY MORE ALIAS ENTRIES
YES
NO
ENT00200
G3
IS THERE AN ENTRY POINT
YES
NO
02 A5
02 A2

CHART DC. ENTRY PROCESSOR (HEWLFENT) (PART 2 OF 2)

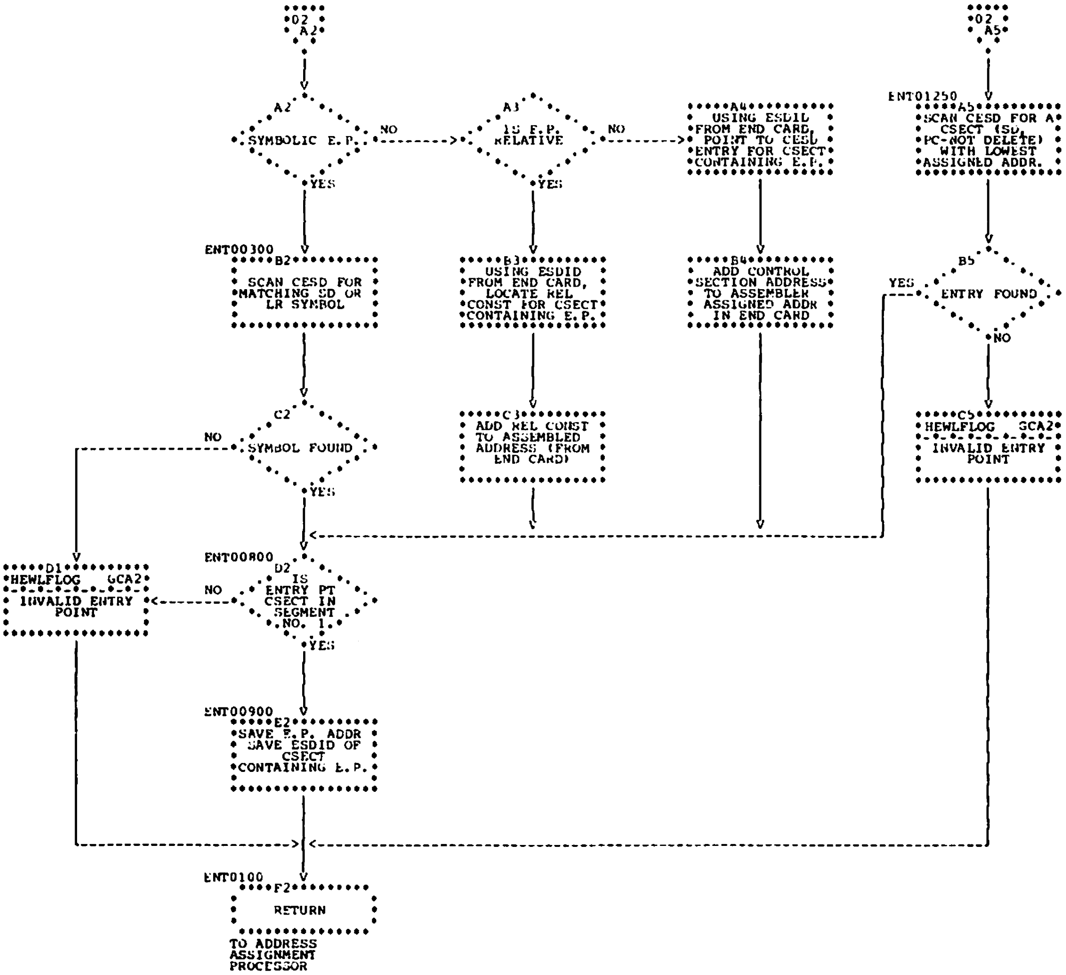

02 A2
02 A5
A2
SYMBOLIC E.P.
NO
YES
A3
IS E.P. RELATIVE
NO
YES
A4
USING ESDID FROM END CARD, POINT TO CESD ENTRY FOR CSECT CONTAINING E.P.
ENT01250
A5
SCAN CESD FOR A CSECT (SD, PC-NOT DELETE) WITH LOWEST ASSIGNED ADDR.
ENT00300
B2
SCAN CESD FOR MATCHING SD OR LR SYMBOL
B3
USING ESDID FROM END CARD, LOCATE REL CONST FOR CSECT CONTAINING E.P.
B4
ADD CONTROL SECTION ADDRESS TO ASSEMBLER ASSIGNED ADDR IN END CARD
B5
ENTRY FOUND
YES
NO
C2
SYMBOL FOUND
NO
YES
C3
ADD REL CONST TO ASSEMBLED ADDRESS (FROM END CARD)
C5
HEWLFLOG GCA2
INVALID ENTRY POINT
D1
HEWLFLOG GCA2
INVALID ENTRY POINT
ENT00800
D2
IS ENTRY PT CSECT IN SEGMENT NO. 1
NO
YES
ENT00900
E2
SAVE E.P. ADDR SAVE ESDID OF CSECT CONTAINING E.P.
ENT0100
F2
RETURN
TO ADDRESS ASSIGNMENT PROCESSOR

CHART EA. INTERMEDIATE OUTPUT PROCESSOR (HEWLFOUT)

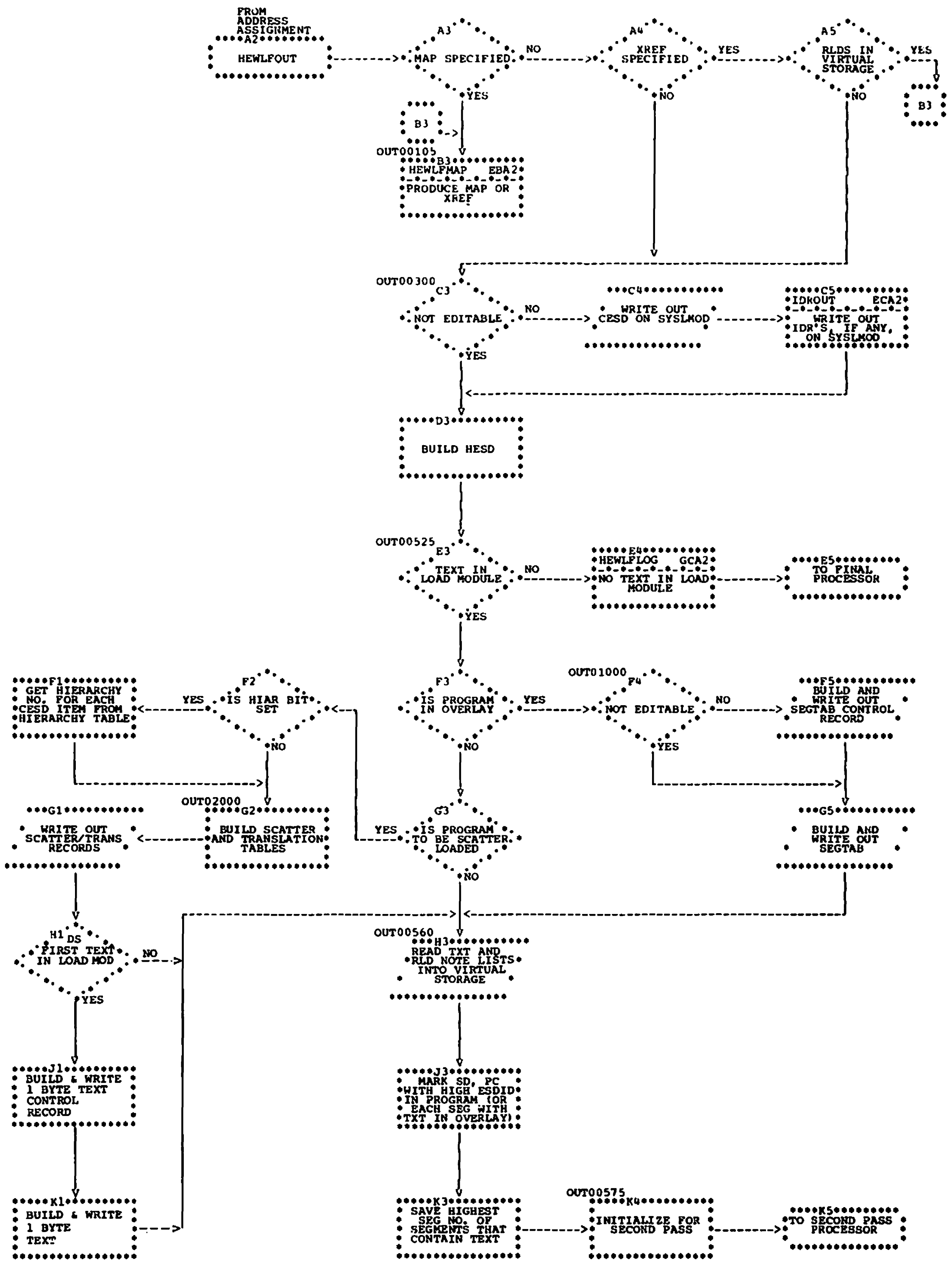

FROM ADDRESS ASSIGNMENT
A2
HEWLFOUT
A3
MAP SPECIFIED
NO
YES
A4
XREF SPECIFIED
YES
NO
A5
RLDS IN VIRTUAL STORAGE
YES
NO
B3
B3
OUT00105
B3
HEWLFMAP EBA2
PRODUCE MAP OR XREF
OUT00300
C3
NOT EDITABLE
NO
YES
C4
WRITE OUT CESD ON SYSLMOD
C5
IDROUT ECA2
WRITE OUT IDR'S, IF ANY, ON SYSLMOD
D3
BUILD HESD
OUT00525
E3
TEXT IN LOAD MODULE
NO
YES
E4
HEWLFLOG GCA2
NO TEXT IN LOAD MODULE
E5
TO FINAL PROCESSOR
F1
GET HIERARCHY NO. FOR EACH CESD ITEM FROM HIERARCHY TABLE
F2
IS HIAR BIT SET
YES
NO
F3
IS PROGRAM IN OVERLAY
YES
NO
OUT01000
F4
NOT EDITABLE
NO
YES
F5
BUILD AND WRITE OUT SEGTAB CONTROL RECORD
G1
WRITE OUT SCATTER/TRANS RECORDS
OUT02000
G2
BUILD SCATTER AND TRANSLATION TABLES
G3
IS PROGRAM TO BE SCATTER LOADED
YES
NO
G5
BUILD AND WRITE OUT SEGTAB
H1
DS
FIRST TEXT IN LOAD MOD
NO
YES
OUT00560
H3
READ TXT AND RLD NOTE LISTS INTO VIRTUAL STORAGE
J1
BUILD & WRITE 1 BYTE TEXT CONTROL RECORD
J3
MARK SD, PC WITH HIGH ESDID IN PROGRAM (OR EACH SEG WITH TXT IN OVERLAY)
K1
BUILD & WRITE 1 BYTE TEXT
K3
SAVE HIGHEST SEG NO. OF SEGMENTS THAT CONTAIN TEXT
OUT00575
K4
INITIALIZE FOR SECOND PASS
K5
TO SECOND PASS PROCESSOR

CHART EB. MAP/REF PROCESSOR (HEWLFMAP)

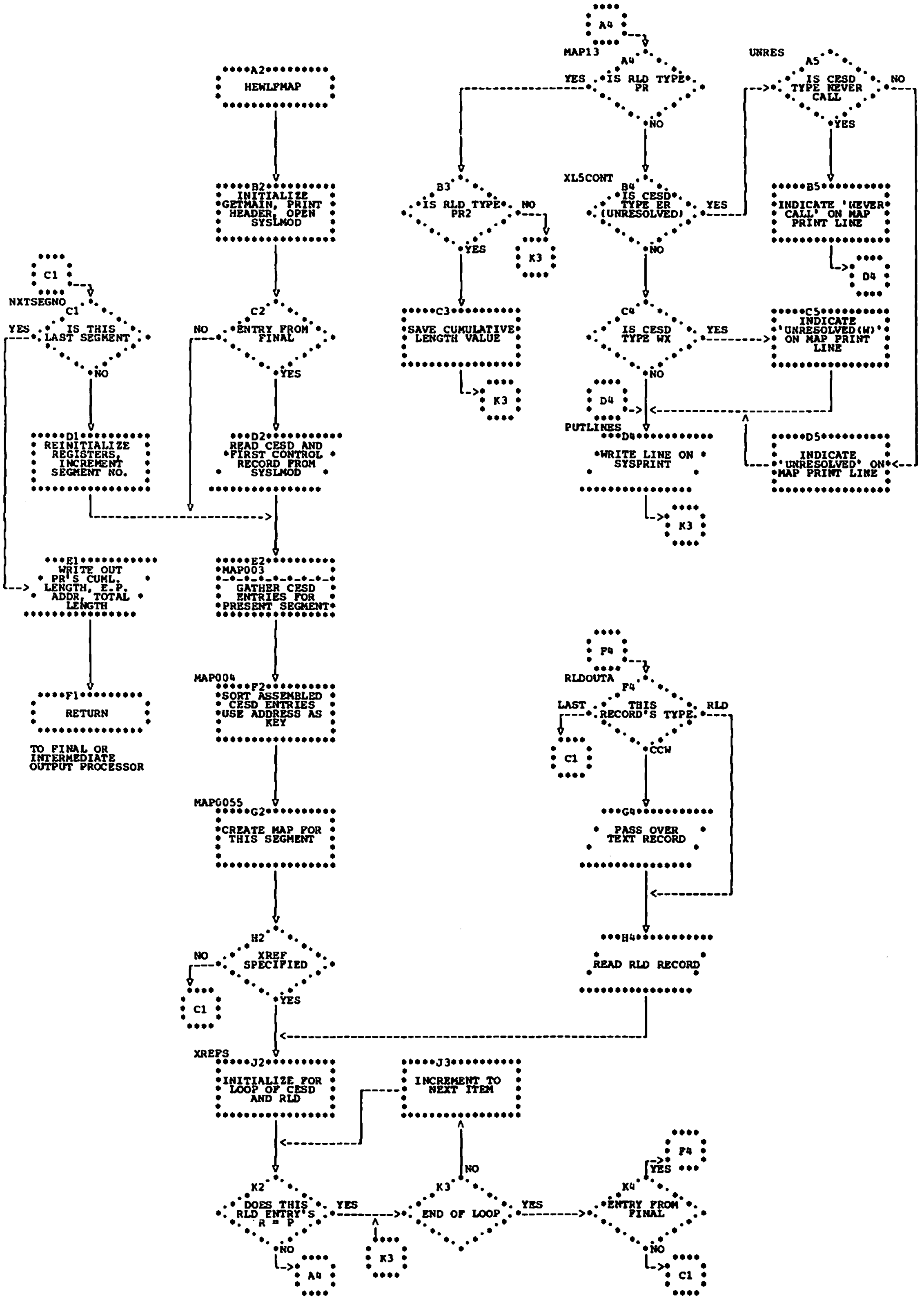

A4
MAP13
A4
IS RLD TYPE PR
YES
NO
UNRES
A5
IS CESD TYPE NEVER CALL
NO
YES
A2
HEWLFMAP
B2
INITIALIZE GETMAIN, PRINT HEADER, OPEN SYSLMOD
B3
IS RLD TYPE PR2
NO
YES
K3
XL5CONT
B4
IS CESD TYPE ER (UNRESOLVED)
YES
NO
B5
INDICATE 'NEVER CALL' ON MAP PRINT LINE
D4
C1
NXTSEGNO
C1
IS THIS LAST SEGMENT
YES
NO
C2
ENTRY FROM FINAL
NO
YES
C3
SAVE CUMULATIVE LENGTH VALUE
K3
C4
IS CESD TYPE WX
YES
NO
C5
INDICATE 'UNRESOLVED(W)' ON MAP PRINT LINE
D4
PUTLINES
D4
WRITE LINE ON SYSPRINT
K3
D1
REINITIALIZE REGISTERS, INCREMENT SEGMENT NO.
D2
READ CESD AND FIRST CONTROL RECORD FROM SYSLMOD
D5
INDICATE 'UNRESOLVED' ON MAP PRINT LINE
E1
WRITE OUT PR'S CUML. LENGTH, E.P. ADDR, TOTAL LENGTH
E2
MAP003
GATHER CESD ENTRIES FOR PRESENT SEGMENT
F4
RLDOUTA
F4
THIS RECORD'S TYPE
LAST
RLD
C1
CCW
F1
RETURN
TO FINAL OR INTERMEDIATE OUTPUT PROCESSOR
MAP004
F2
SORT ASSEMBLED CESD ENTRIES USE ADDRESS AS KEY
MAP0055
G2
CREATE MAP FOR THIS SEGMENT
G4
PASS OVER TEXT RECORD
H2
XREF SPECIFIED
NO
C1
YES
H4
READ RLD RECORD
XREFS
J2
INITIALIZE FOR LOOP OF CESD AND RLD
J3
INCREMENT TO NEXT ITEM
NO
F4
YES
K2
DOES THIS RLD ENTRY'S R = P
YES
NO
A4
K3
K3
END OF LOOP
YES
K4
ENTRY FROM FINAL
NO
C1

CHART EC. IDR WRITE ROUTINE (HEWLFOUT) (PART 1 OF 2)

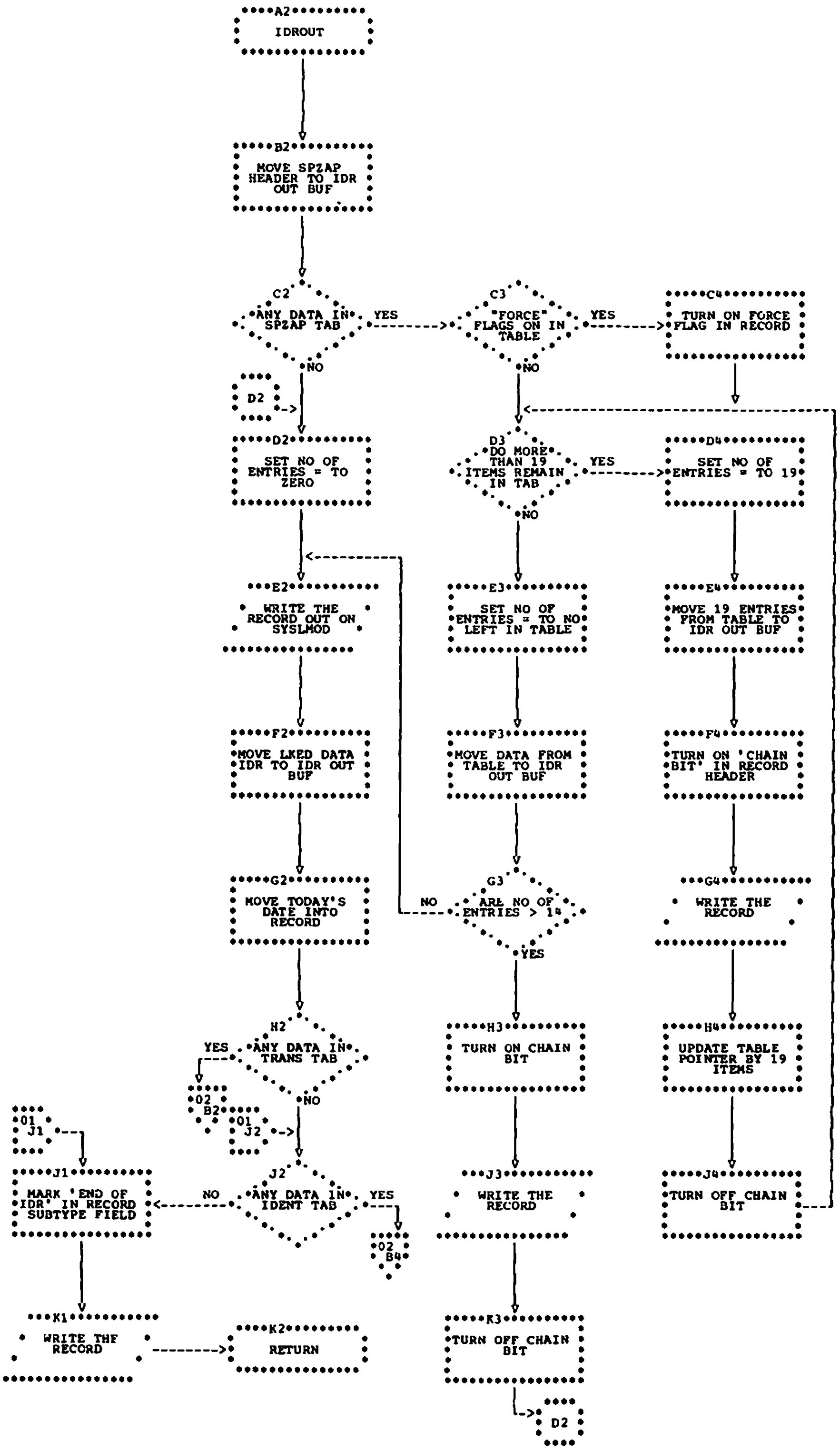

A2
IDROUT
B2
MOVE SPZAP HEADER TO IDR OUT BUF
C2
ANY DATA IN SPZAP TAB
YES
NO
C3
'FORCE' FLAGS ON IN TABLE
YES
NO
C4
TURN ON FORCE FLAG IN RECORD
D2
D2
SET NO OF ENTRIES = TO ZERO
D3
DO MORE THAN 19 ITEMS REMAIN IN TAB
YES
NO
D4
SET NO OF ENTRIES = TO 19
E2
WRITE THE RECORD OUT ON SYSLMOD
E3
SET NO OF ENTRIES = TO NO LEFT IN TABLE
E4
MOVE 19 ENTRIES FROM TABLE TO IDR OUT BUF
F2
MOVE LKED DATA IDR TO IDR OUT BUF
F3
MOVE DATA FROM TABLE TO IDR OUT BUF
F4
TURN ON 'CHAIN BIT' IN RECORD HEADER
G2
MOVE TODAY'S DATE INTO RECORD
G3
ARE NO OF ENTRIES > 14
NO
YES
G4
WRITE THE RECORD
H2
ANY DATA IN TRANS TAB
YES
NO
02 B2
01 J2
H3
TURN ON CHAIN BIT
H4
UPDATE TABLE POINTER BY 19 ITEMS
01 J1
J1
MARK 'END OF IDR' IN RECORD SUBTYPE FIELD
J2
ANY DATA IN IDENT TAB
NO
YES
02 B4
J3
WRITE THE RECORD
J4
TURN OFF CHAIN BIT
K1
WRITE THE RECORD
K2
RETURN
K3
TURN OFF CHAIN BIT
D2

CHART EC. IDR WRITE ROUTINE (HEWLFOUT) (PART 2 OF 2)

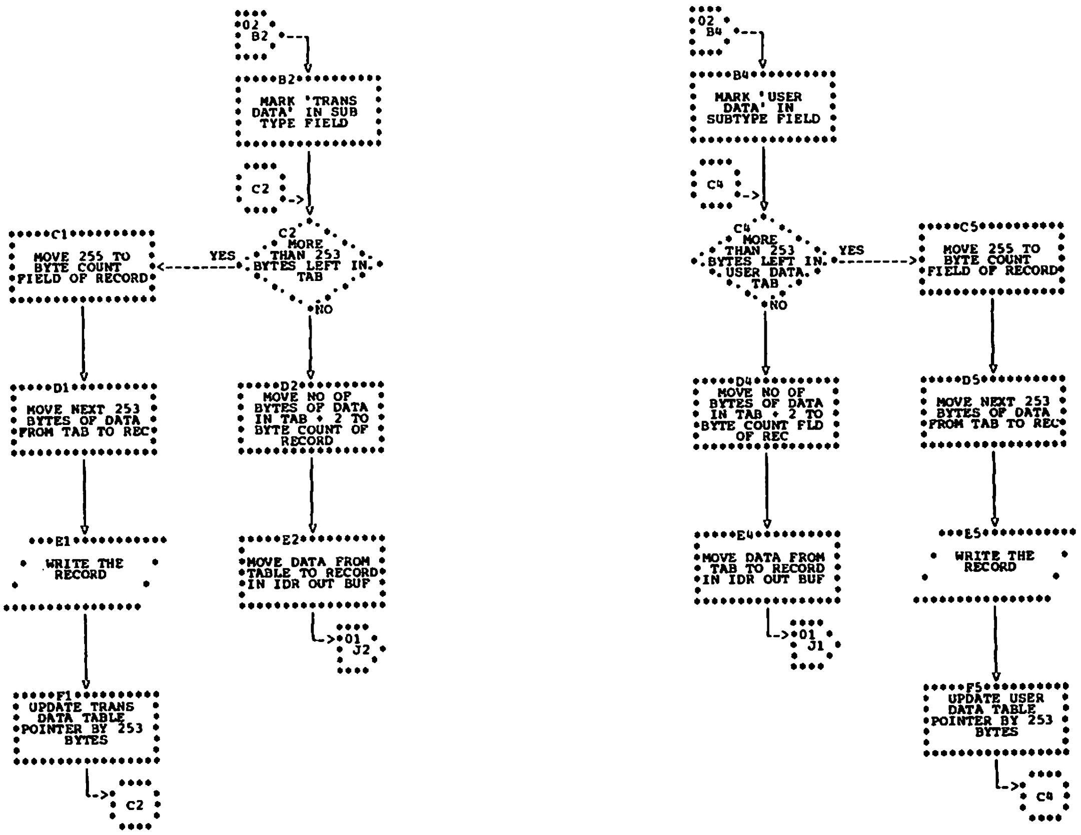

02 B2
B2
MARK 'TRANS DATA' IN SUB TYPE FIELD
C2
C2
MORE THAN 253 BYTES LEFT IN TAB
YES
NO
C1
MOVE 255 TO BYTE COUNT FIELD OF RECORD
D1
MOVE NEXT 253 BYTES OF DATA FROM TAB TO REC
E1
WRITE THE RECORD
F1
UPDATE TRANS DATA TABLE POINTER BY 253 BYTES
C2
D2
MOVE NO OF BYTES OF DATA IN TAB + 2 TO BYTE COUNT OF RECORD
E2
MOVE DATA FROM TABLE TO RECORD IN IDR OUT BUF
01 J2
02 B4
B4
MARK 'USER DATA' IN SUBTYPE FIELD
C4
C4
MORE THAN 253 BYTES LEFT IN USER DATA TAB
YES
NO
C5
MOVE 255 TO BYTE COUNT FIELD OF RECORD
D5
MOVE NEXT 253 BYTES OF DATA FROM TAB TO REC
E5
WRITE THE RECORD
F5
UPDATE USER DATA TABLE POINTER BY 253 BYTES
C4
D4
MOVE NO OF BYTES OF DATA IN TAB + 2 TO BYTE COUNT FLD OF REC
E4
MOVE DATA FROM TAB TO RECORD IN IDR OUT BUF
01 J1

CHART FA. SECOND PASS PROCESSOR (HEWLFSCD) (PART 1 OF 2)

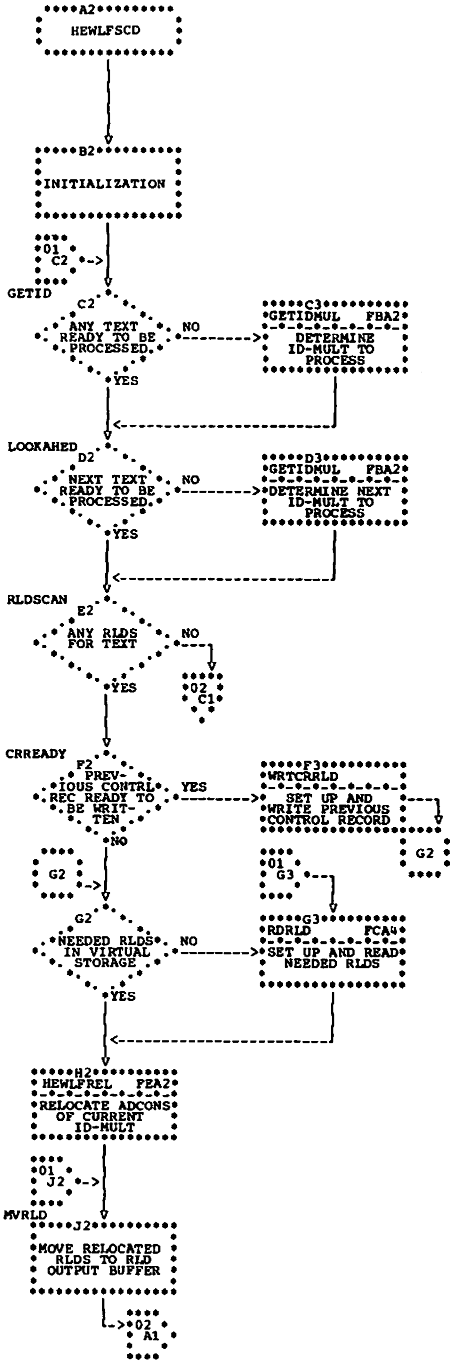

A2
HEWLFSCD
B2
INITIALIZATION
01 C2
GETID
C2
ANY TEXT READY TO BE PROCESSED.
NO
C3
GETIDMUL FBA2
DETERMINE ID-MULT TO PROCESS
YES
LOOKAHED
D2
NEXT TEXT READY TO BE PROCESSED.
NO
D3
GETIDMUL FBA2
DETERMINE NEXT ID-MULT TO PROCESS
YES
RLDSCAN
E2
ANY RLDS FOR TEXT
NO
02 C1
YES
CRREADY
F2
PREV-IOUS CONTRL REC READY TO BE WRIT-TEN
YES
F3
WRTCRRLD
SET UP AND WRITE PREVIOUS CONTROL RECORD
G2
NO
G2
01 G3
G2
NEEDED RLDS IN VIRTUAL STORAGE
NO
G3
RDRLD FCA4
SET UP AND READ NEEDED RLDS
YES
H2
HEWLFREL FEA2
RELOCATE ADCONS OF CURRENT ID-MULT
01 J2
MVRLD
J2
MOVE RELOCATED RLDS TO RLD OUTPUT BUFFER
02 A1

CHART FA. SECOND PASS PROCESSOR (HEWLFSCD) (PART 2 OF 2)

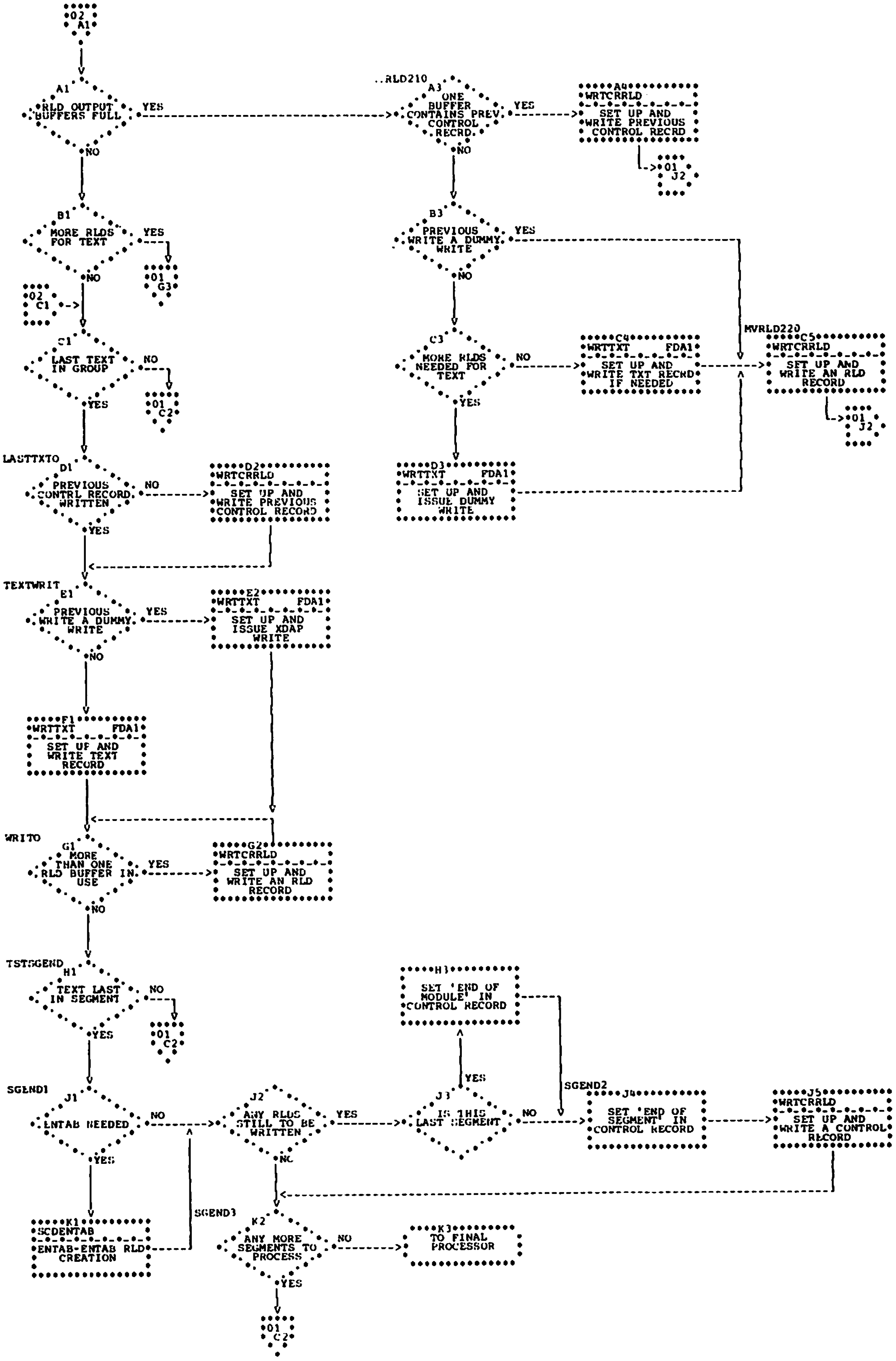

02 A1
A1 RLD OUTPUT BUFFERS FULL
YES
NO
RLD210 A3 ONE BUFFER CONTAINS PREV. CONTROL RECRD.
YES
NO
A4 WRTCRRLD SET UP AND WRITE PREVIOUS CONTROL RECRD
01 J2
B1 MORE RLDS FOR TEXT
YES
NO
01 G3
02 C1
B3 PREVIOUS WRITE A DUMMY WRITE
YES
NO
C1 LAST TEXT IN GROUP
NO
YES
01 C2
C3 MORE RLDS NEEDED FOR TEXT
NO
YES
C4 WRTTXT FDA1 SET UP AND WRITE TXT RECRD IF NEEDED
MVRLD220 C5 WRTCRRLD SET UP AND WRITE AN RLD RECORD
01 J2
LASTTXTO D1 PREVIOUS CONTRL RECORD WRITTEN
NO
YES
D2 WRTCRRLD SET UP AND WRITE PREVIOUS CONTROL RECORD
D3 WRTTXT FDA1 SET UP AND ISSUE DUMMY WRITE
TEXTWRIT E1 PREVIOUS WRITE A DUMMY WRITE
YES
NO
E2 WRTTXT FDA1 SET UP AND ISSUE XDAP WRITE
F1 WRTTXT FDA1 SET UP AND WRITE TEXT RECORD
WRITO G1 MORE THAN ONE RLD BUFFER IN USE
YES
NO
G2 WRTCRRLD SET UP AND WRITE AN RLD RECORD
TSTSGEND H1 TEXT LAST IN SEGMENT
NO
YES
01 C2
H3 SET 'END OF MODULE' IN CONTROL RECORD
SGEND1 J1 ENTAB NEEDED
NO
YES
J2 ANY RLDS STILL TO BE WRITTEN
YES
NO
J3 IS THIS LAST SEGMENT
YES
NO
SGEND2 J4 SET 'END OF SEGMENT' IN CONTROL RECORD
J5 WRTCRRLD SET UP AND WRITE A CONTROL RECORD
K1 SCDENTAB ENTAB-ENTAB RLD CREATION
SGEND3 K2 ANY MORE SEGMENTS TO PROCESS
NO
YES
K3 TO FINAL PROCESSOR
01 C2

CHART FB. GETIDMUL ROUTINE

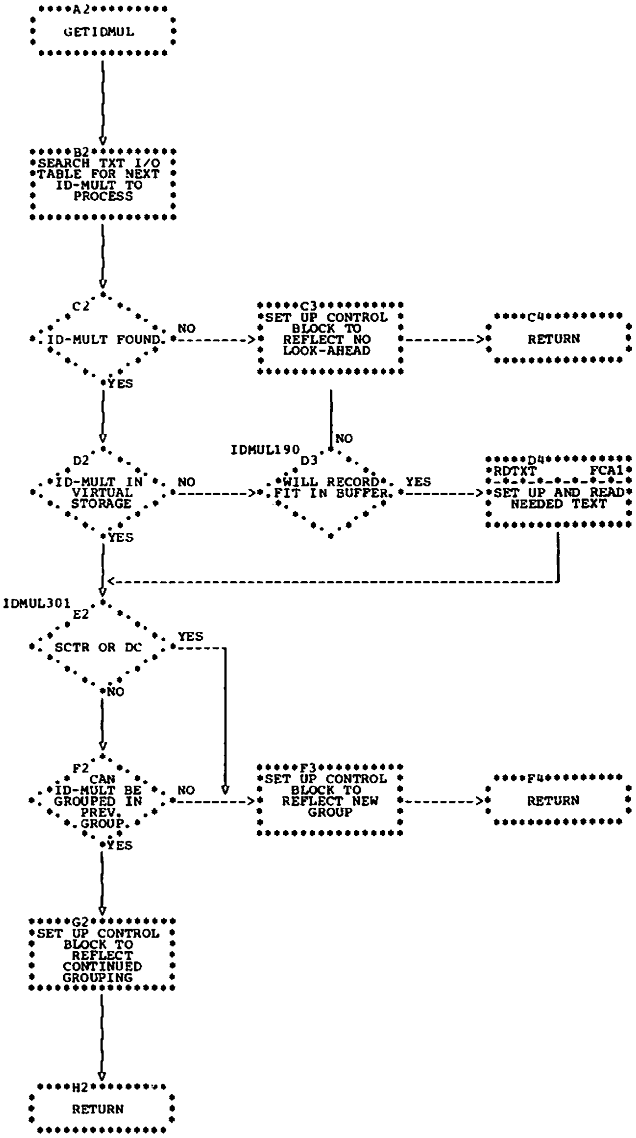

A2
GETIDMUL
B2
SEARCH TXT I/O TABLE FOR NEXT ID-MULT TO PROCESS
C2
ID-MULT FOUND
NO
C3
SET UP CONTROL BLOCK TO REFLECT NO LOOK-AHEAD
C4
RETURN
YES
D2
ID-MULT IN VIRTUAL STORAGE
NO
IDMUL190
D3
WILL RECORD FIT IN BUFFER
NO
YES
D4
RDTXT FCA1
SET UP AND READ NEEDED TEXT
YES
IDMUL301
E2
SCTR OR DC
YES
NO
F2
CAN ID-MULT BE GROUPED IN PREV. GROUP
NO
F3
SET UP CONTROL BLOCK TO REFLECT NEW GROUP
F4
RETURN
YES
G2
SET UP CONTROL BLOCK TO REFLECT CONTINUED GROUPING
H2
RETURN

CHART FC. TXT READ ROUTINE (RDTXT), RLD READ ROUTINE (RDRLD)—HEWLFSIO

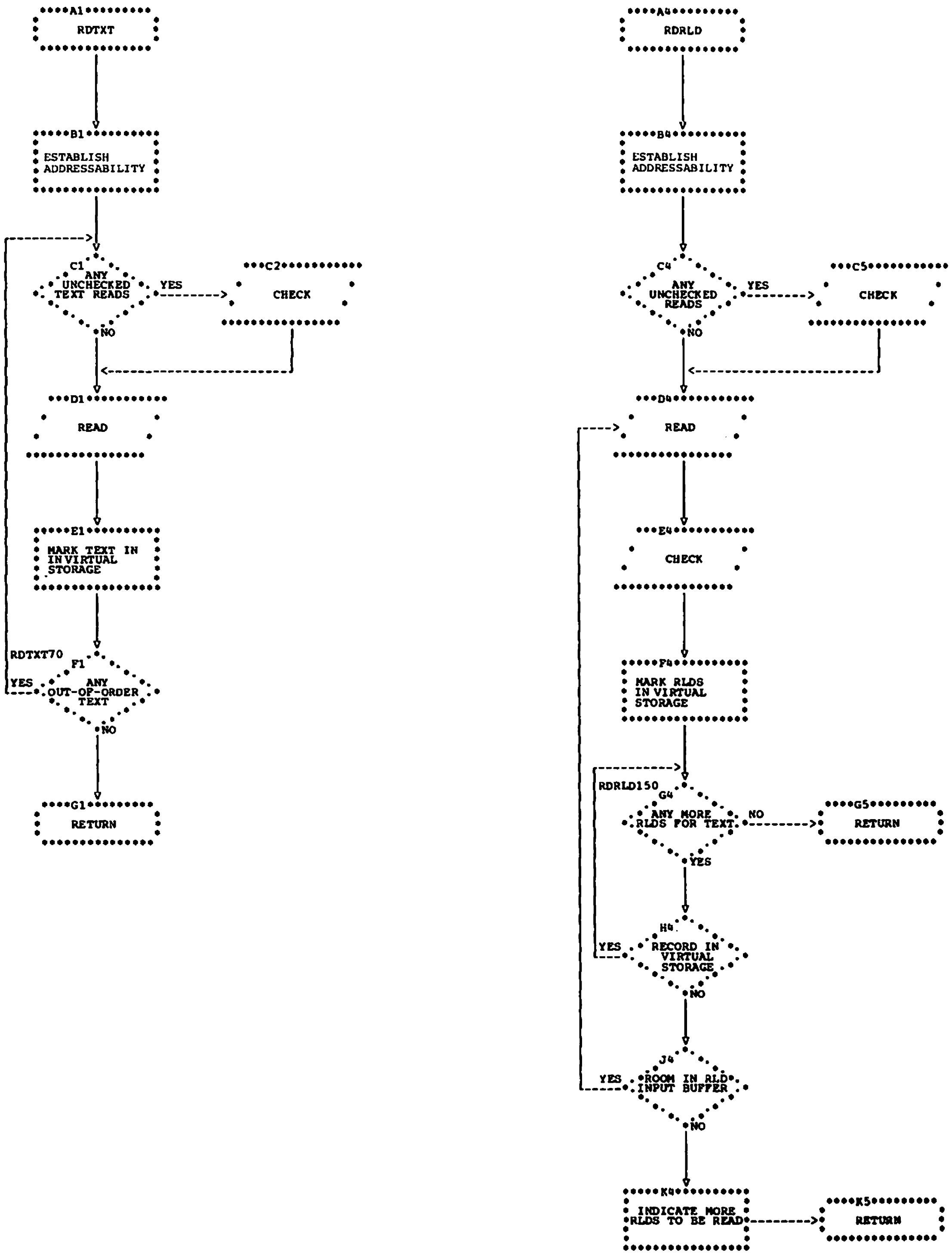


CHART FD. TEXT WRITE ROUTINE (ON SYSLMOD) (WRTTXT)—HEWLFSIO

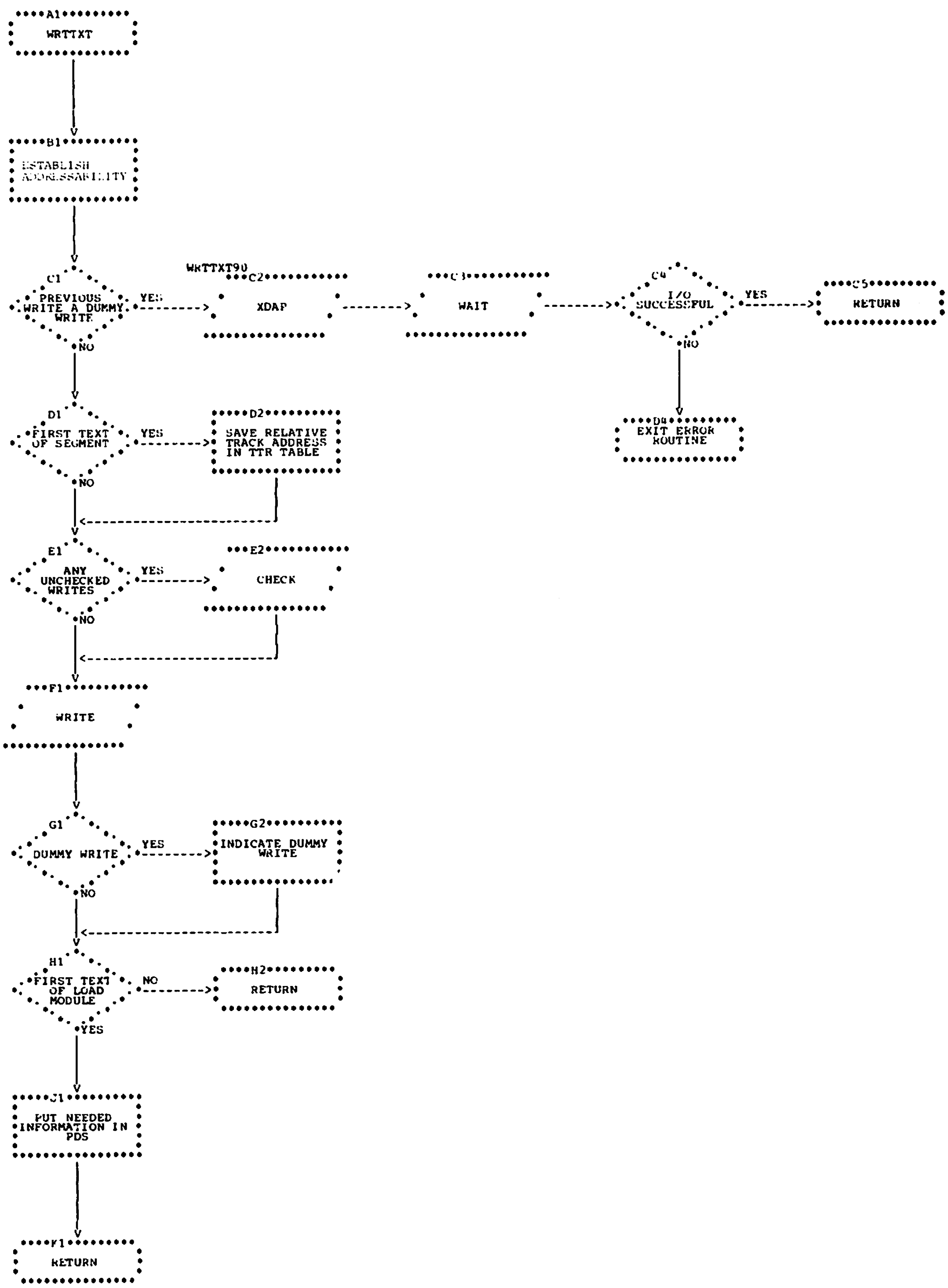


CHART FE. RELOCATION ROUTINE (HEWLFREL) (PART 1 OF 3)

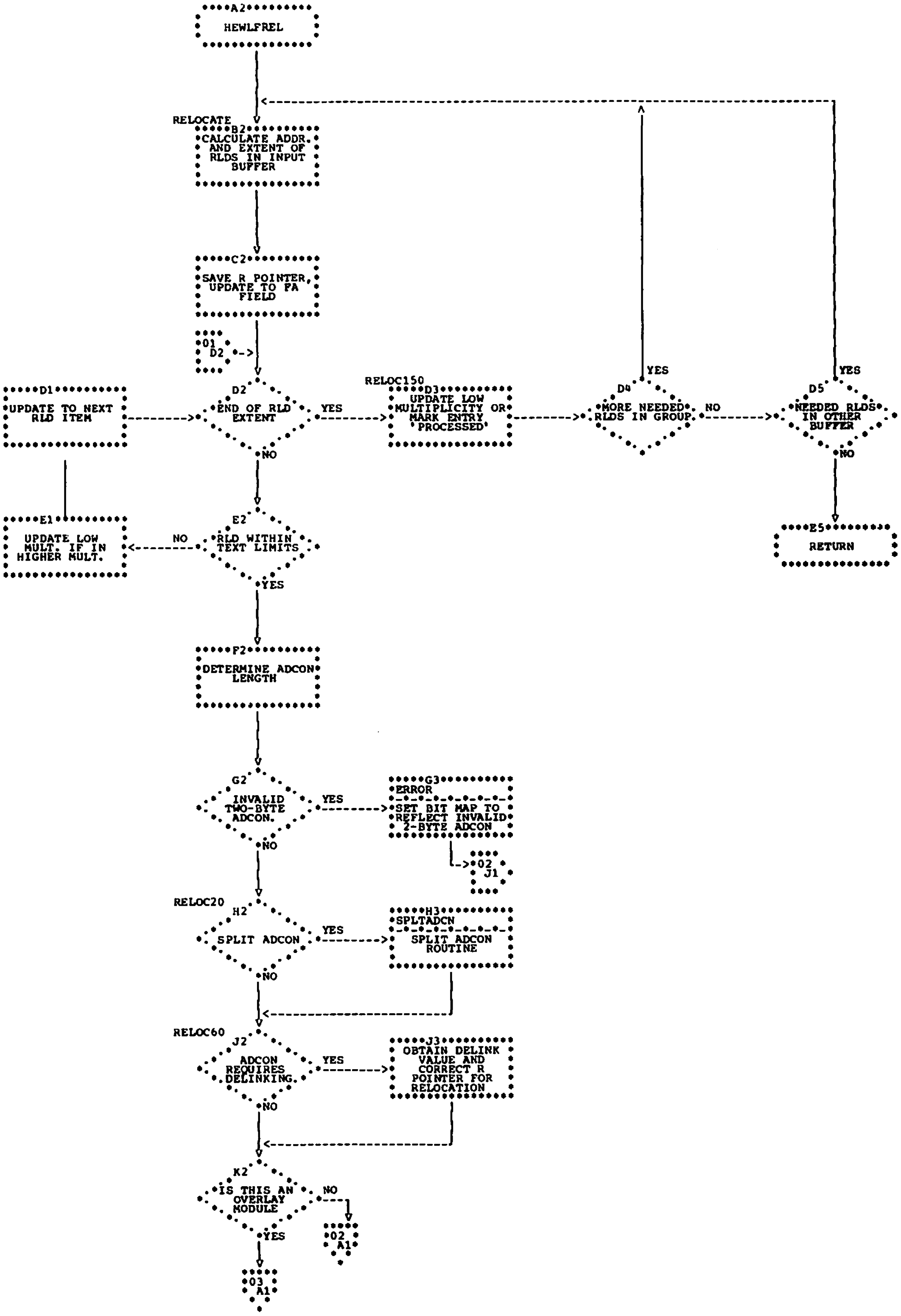

A2
HEWLFREL
RELOCATE
B2
CALCULATE ADDR. AND EXTENT OF RLDS IN INPUT BUFFER
C2
SAVE R POINTER, UPDATE TO FA FIELD
01 D2
D1
UPDATE TO NEXT RLD ITEM
D2
END OF RLD EXTENT
YES
NO
RELOC150
D3
UPDATE LOW MULTIPLICITY OR MARK ENTRY 'PROCESSED'
D4
MORE NEEDED RLDS IN GROUP
YES
NO
D5
NEEDED RLDS IN OTHER BUFFER
YES
NO
E5
RETURN
E1
UPDATE LOW MULT. IF IN HIGHER MULT.
E2
RLD WITHIN TEXT LIMITS
NO
YES
F2
DETERMINE ADCON LENGTH
G2
INVALID TWO-BYTE ADCON.
YES
NO
G3
ERROR
SET BIT MAP TO REFLECT INVALID 2-BYTE ADCON
02 J1
RELOC20
H2
SPLIT ADCON
YES
NO
H3
SPLTADCN
SPLIT ADCON ROUTINE
RELOC60
J2
ADCON REQUIRES DELINKING.
YES
NO
J3
OBTAIN DELINK VALUE AND CORRECT R POINTER FOR RELOCATION
K2
IS THIS AN OVERLAY MODULE
NO
YES
02 A1
03 A1

CHART FE. RELOCATION ROUTINE (HEWLFREL) (PART 2 OF 3)

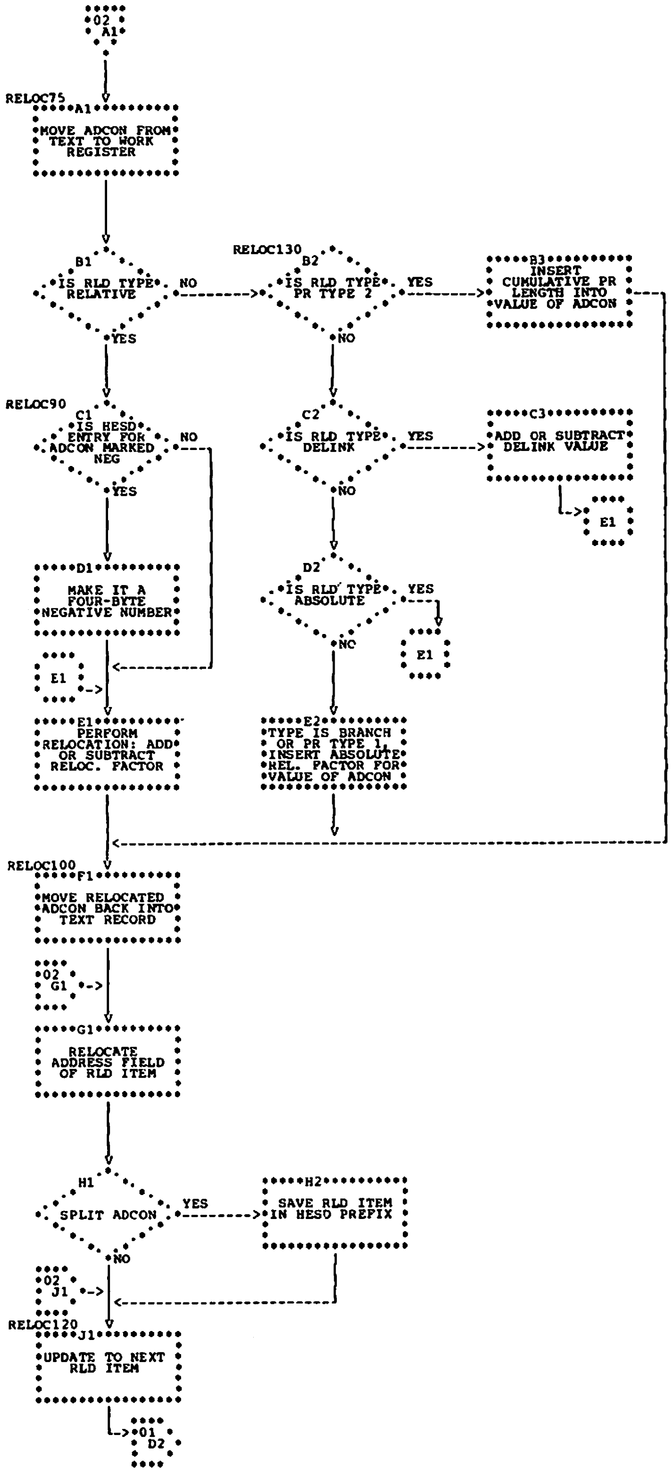

02 A1
RELOC75 A1
MOVE ADCON FROM TEXT TO WORK REGISTER
B1
IS RLD TYPE RELATIVE
NO
YES
RELOC130 B2
IS RLD TYPE PR TYPE 2
YES
NO
B3
INSERT CUMULATIVE PR LENGTH INTO VALUE OF ADCON
RELOC90 C1
IS HESD ENTRY FOR ADCON MARKED NEG
NO
YES
C2
IS RLD TYPE DELINK
YES
NO
C3
ADD OR SUBTRACT DELINK VALUE
E1
D1
MAKE IT A FOUR-BYTE NEGATIVE NUMBER
D2
IS RLD TYPE ABSOLUTE
YES
NO
E1
E1
E1
PERFORM RELOCATION: ADD OR SUBTRACT RELOC. FACTOR
E2
TYPE IS BRANCH OR PR TYPE 1, INSERT ABSOLUTE REL. FACTOR FOR VALUE OF ADCON
RELOC100 F1
MOVE RELOCATED ADCON BACK INTO TEXT RECORD
02 G1
G1
RELOCATE ADDRESS FIELD OF RLD ITEM
H1
SPLIT ADCON
YES
NO
H2
SAVE RLD ITEM IN HESD PREFIX
02 J1
RELOC120 J1
UPDATE TO NEXT RLD ITEM
01 D2

CHART FE. RELOCATION ROUTINE (HEWLFREL) (PART 3 OF 3)

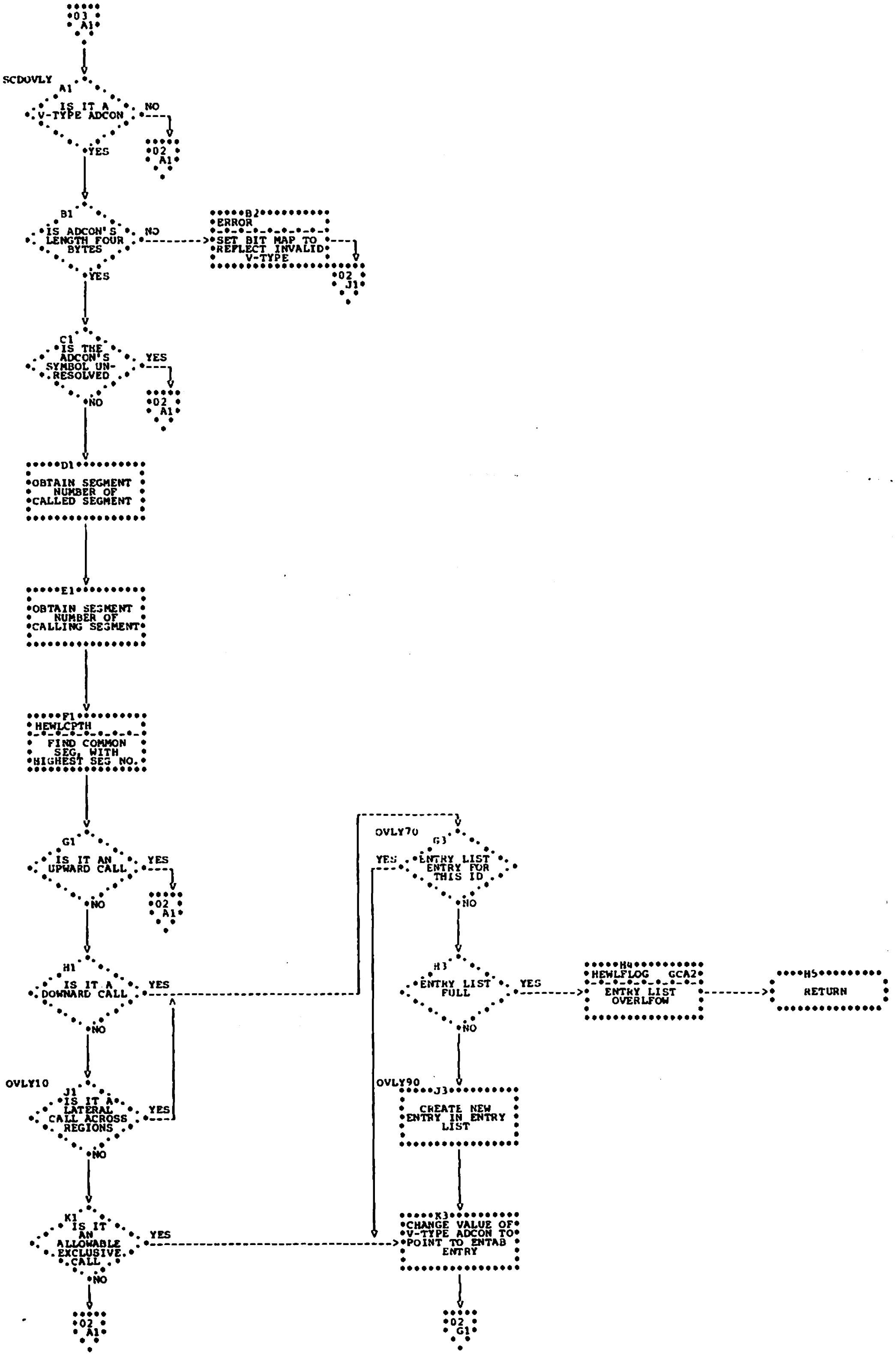


CHART GA. FINAL PROCESSOR (HEWLFFNL)

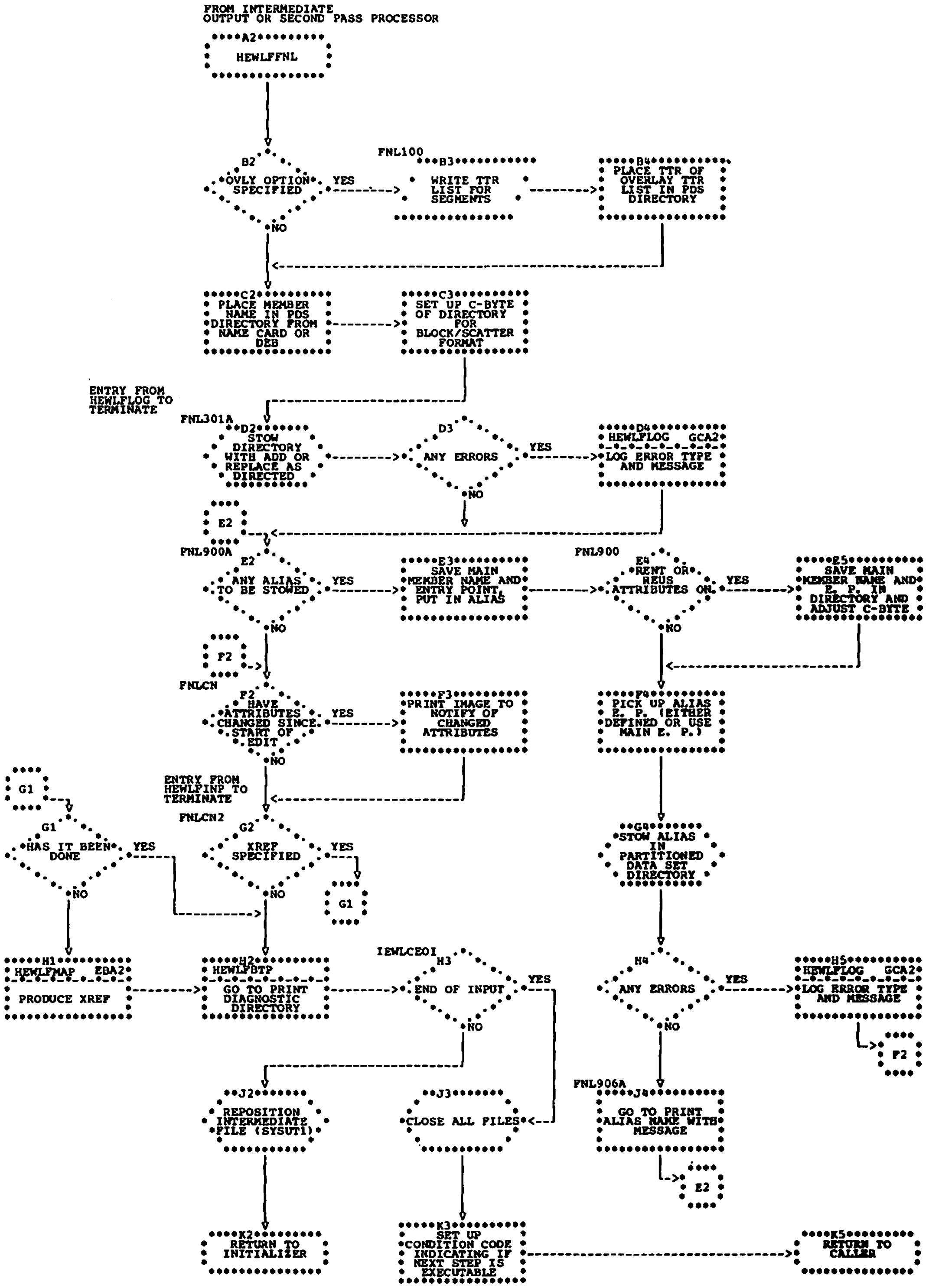

FROM INTERMEDIATE OUTPUT OR SECOND PASS PROCESSOR
A2
HEWLFFNL
B2
OVLY OPTION SPECIFIED
YES
NO
FNL100
B3
WRITE TTR LIST FOR SEGMENTS
B4
PLACE TTR OF OVERLAY TTR LIST IN PDS DIRECTORY
C2
PLACE MEMBER NAME IN PDS DIRECTORY FROM NAME CARD OR DEB
C3
SET UP C-BYTE OF DIRECTORY FOR BLOCK/SCATTER FORMAT
ENTRY FROM HEWLFLOG TO TERMINATE
FNL301A
D2
STOW DIRECTORY WITH ADD OR REPLACE AS DIRECTED
D3
ANY ERRORS
YES
NO
D4
HEWLFLOG GCA2
LOG ERROR TYPE AND MESSAGE
E2
FNL900A
E2
ANY ALIAS TO BE STOWED
YES
NO
E3
SAVE MAIN MEMBER NAME AND ENTRY POINT, PUT IN ALIAS
FNL900
E4
RENT OR REUS ATTRIBUTES ON
YES
NO
E5
SAVE MAIN MEMBER NAME AND E. P. IN DIRECTORY AND ADJUST C-BYTE
F2
FNLCN
F2
HAVE ATTRIBUTES CHANGED SINCE START OF EDIT
YES
NO
F3
PRINT IMAGE TO NOTIFY OF CHANGED ATTRIBUTES
F4
PICK UP ALIAS E. P. (EITHER DEFINED OR USE MAIN E. P.)
G1
ENTRY FROM HEWLFINP TO TERMINATE
FNLCN2
G1
HAS IT BEEN DONE
YES
NO
G2
XREF SPECIFIED
YES
NO
G1
G4
STOW ALIAS IN PARTITIONED DATA SET DIRECTORY
H1
HEWLFMAP EBA2
PRODUCE XREF
H2
HEWLFBTP
GO TO PRINT DIAGNOSTIC DIRECTORY
IEWLCEOI
H3
END OF INPUT
YES
NO
H4
ANY ERRORS
YES
NO
H5
HEWLFLOG GCA2
LOG ERROR TYPE AND MESSAGE
F2
J2
REPOSITION INTERMEDIATE FILE (SYSUT1)
J3
CLOSE ALL FILES
FNL906A
J4
GO TO PRINT ALIAS NAME WITH MESSAGE
E2
K2
RETURN TO INITIALIZER
K3
SET UP CONDITION CODE INDICATING IF NEXT STEP IS EXECUTABLE
K5
RETURN TO CALLER

CHART GB. SYNAD ROUTINE (HEWLCR01)

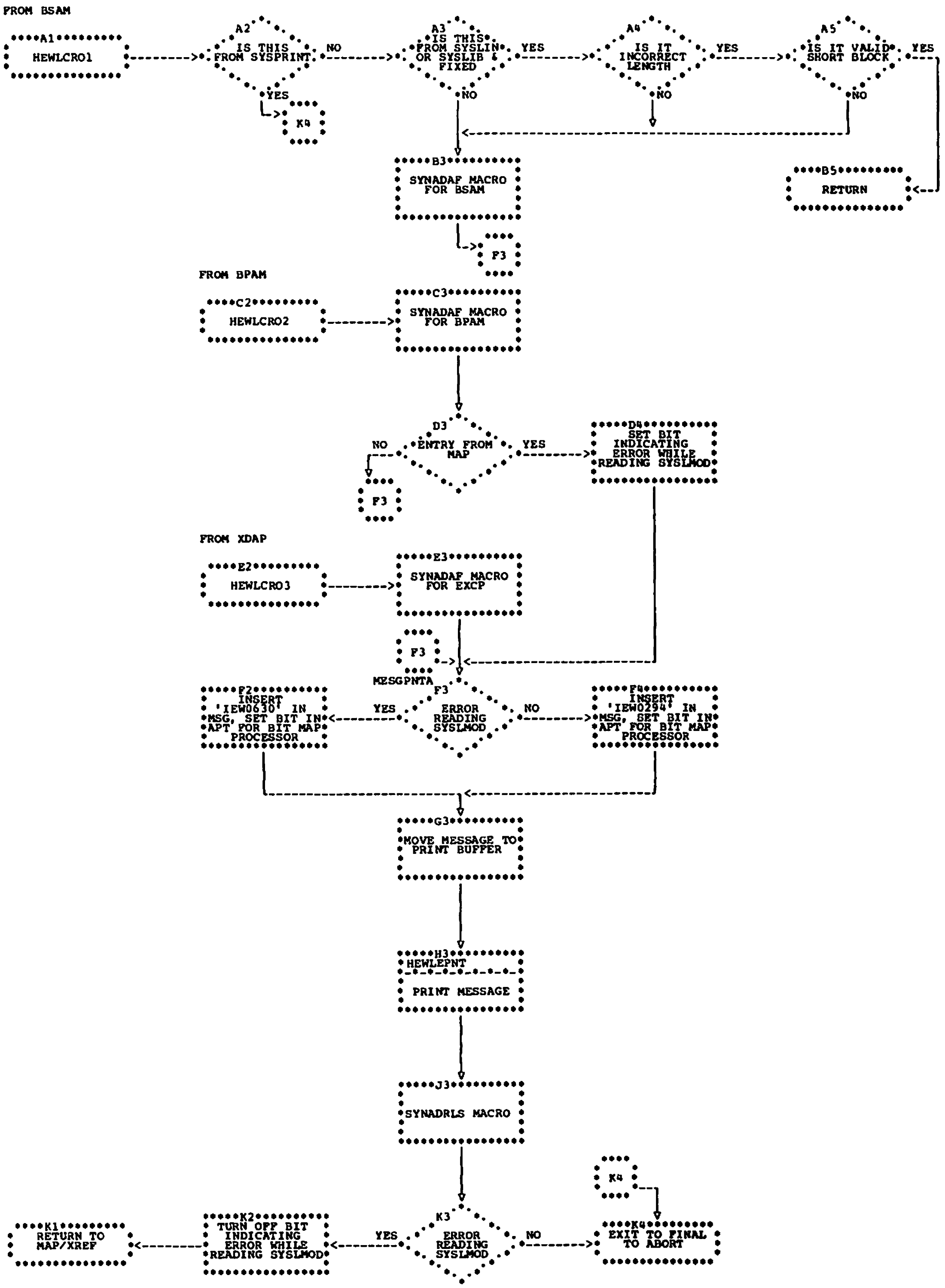
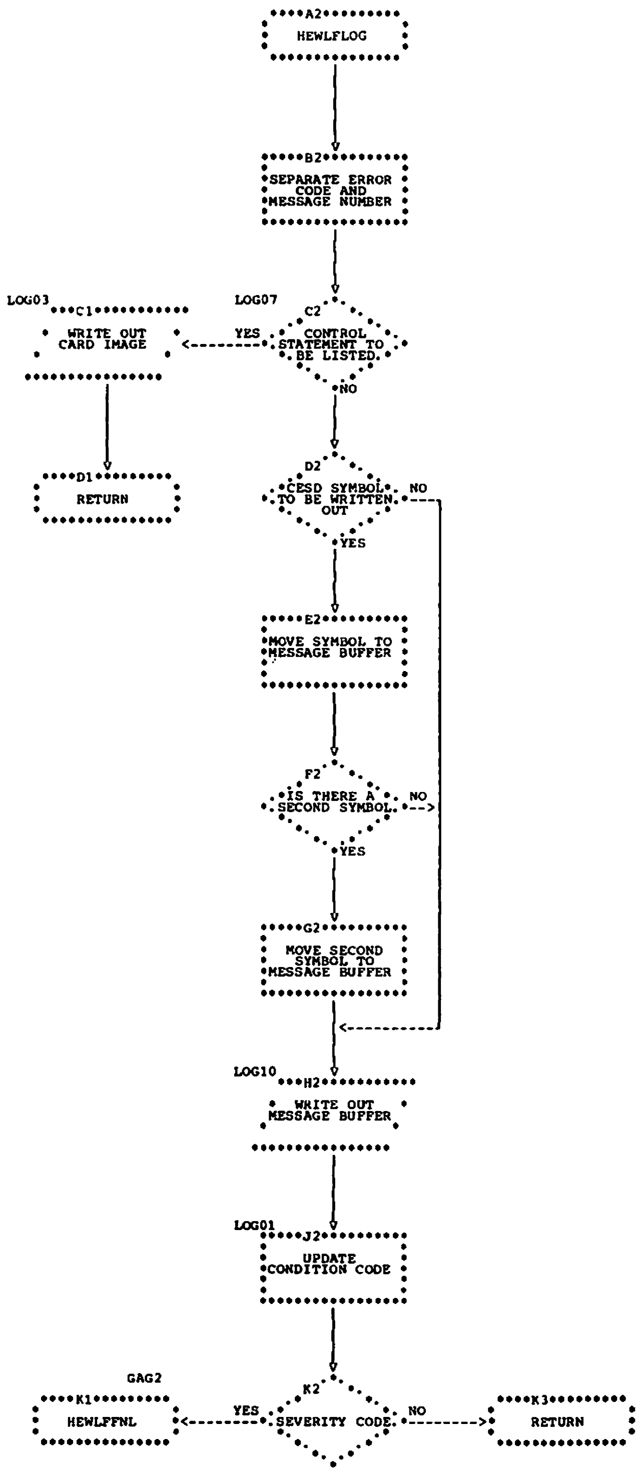


CHART GC. ERROR LOGGING ROUTINE (HEWLFLOG)



MICROFICHE DIRECTORY

The microfiche directory, Figure 33, is designed to help you find named areas of code in the program listing, which is on microfiche. Microfiche cards are filed in alphameric order by object module name. If you wish to locate a control section or entry point on microfiche, find the name in column one and note the associated CSECT name. A description of the control section is also given.

This section also contains a module-CSECT cross-reference table (see Figure 34 on page 149).

Symbol	Type	CSECT	Description	Referenced By
APTEND	Entry point	HEWLFROU	See HEWLEPNT	
APTEXLST	Label	HEWLFROU	Open exit list for SYSLIN, SYSPRINT, and SYSLIB	HEWLFINT, HEWLFINC
APTXLIST	Label	HEWLFROU	Open exit list for SYSLMOD	HEWLFINT, HEWLFMAP
APT000	Entry point	HEWLFAPT	SYNAD exit routine for SYSPRINT	
CHECKRD	Entry point	HEWLFSIO	Routine to check reads on SYSUT1	HEWLFREL
CHECKWRT	Entry point	HEWLFSIO	Routine to check writes on SYSLMOD	HEWLFREL
ENQNAME	Label	HEWLFROU	Major name by which SYSLMOD is enqueued	HEWLFINT, HEWLFFNL
GETIDMUL	Entry point	HEWLFSCD	Text LOOK AHEAD/READ AHEAD routine	HEWLFREL
HEWLCADA	Label	HEWLFROU	Not used	
HEWLCAD1	Entry point	HEWLFADA	Routine to make CESD entries for ENTABS	HEWLFENS
HEWLCAUT	Entry point	HEWLFINC	Automatic library call processing	HEWLFINP
HEWLCDCN	Entry point	HEWLFRCG	Library DECHAIN routine	HEWLFESD
HEWLCDLK	Entry point	HEWLFINP	DELINK routine	HEWLFESD, HEWLFRAT
HEWLCEOD	Entry point	HEWLFINP	END-OF-DATA routine for SYSLIB	HEWLFINC, HEWLFROU
HEWLCEOI	Entry point	HEWLFFNL		HEWLFROU
HEWLCE30	Entry point	HEWLFESD	Return point to avoid ESD processing	HEWLFRCG
HEWLCFAB	Entry point	HEWLFFNL	Termination processing	HEWLFROU
HEWLCFNI	Entry point	HEWLFFNL	Immediate termination processing	HEWLFINP, HEWLFROU

Figure 33 (Part 1 of 4). Microfiche Directory

Symbol	Type	CSECT	Description	Referenced By
HEWLCICA	Label	HEWLFINP	Pointer to include processor	
HEWLCIDR	Entry point	HEWLFIDR	IDR user data from IDENTIFY statement processor	HEWLFSCN
HEWLCINP	Entry point	HEWLFINP	See HEWLFINP	HEWLFTXT
HEWLCMDB	Label	HEWLFROU	DCB for SYSLMOD	HEWLFFNL, HEWLFINT, HEWLFMAP, HEWLFOUT, HEWLFSIO, HEWLFSYM
HEWLCPDB	Label	HEWLFROU	DCB for SYSPRINT	HEWLFAPT, HEWLFFNL, HEWLFINT
HEWLCPTH	Entry point	HEWLFRCG	COMMON PATH routine	HEWLFESD
HEWLCRBB	Label	HEWLFAPT	DECB for SYSLIB	
HEWLCRBN	Label	HEWLFAPT	DECB for SYSLIN	
HEWLCRID	Label	HEWLFESD	ESD ID of item currently in process	HEWLFRCG
HEWLCR01	Entry point	HEWLFROU	END-OF-DATA routine for SYSUT1; SYNAD routine for SYSUT1, SYSPRINT, and SYSLIN	HEWLFINC
HEWLCR02	Entry point	HEWLFROU	END-OF-DATA routine for SYSLMOD	HEWLFINC
HEWLCR03	Entry point	HEWLFROU	I/O ERROR routine for SYSLMOD	HEWLFSIO
HEWLCSDB	Label	HEWLFROU	DCB for SYSLIN	HEWLFAPT, HEWLFFNL, HEWLFINT
HEWLCSNX	Entry point	HEWLFFNL	Termination processing after SYNAD exit	HEWLFROU
HEWLCTTY	Label	HEWLFESD	Type flags for current ESD item	HEWLFRCG
HEWLCUDB	Label	HEWLFROU	DCB for SYSUT1	HEWLFFNL, HEWLFINT, HEWLFOUT, HEWLFRAT, HEWLFSIO
HEWLEEON	Entry point	HEWLFINP	END-OF-DATA routine for SYSLIN	HEWLFROU
HEWLENAM	Entry point	HEWLFINT	Reentry into initialization for multiple link-edits	HEWLFROU, HEWLFFNL
HEWLEPNT	Entry point	HEWLFROU	SYSPRINT OUTPUT routine	HEWLFBTP, HEWLFFNL, HEWLFINT, HEWLFMAP, HEWLFOPT
HEWLERDM	Entry point	HEWLFINP	PRIMARY INPUT READ routine	HEWLFTXT

Figure 33 (Part 2 of 4). Microfiche Directory

Symbol	Type	CSECT	Description	Referenced By
HEWLFADA	CSECT	HEWLFADA	Address assignment	HEWLFINC, HEWLFINP, HEWLFROU
HEWLFALK	Entry point	HEWLFROU	TABLE ALLOCATION routine	HEWLFAPT
HEWLFAPT	CSECT	HEWLFAPT	All Purpose Table (Communications Area)	HEWLFINT
HEWLFAPX	Entry point	HEWLFMAP	RECOVERY routine after SYNAD exit	HEWLFROU
HEWLFBTP	CSECT	HEWLFBTP	Error message printing	HEWLFFNL
HEWLFDEF	CSECT	HEWLFDEF	Default size values	HEWLFINT, HEWLFOPT, HEWLFOUT
HEWLFEND	CSECT	HEWLFEND	End record processing	HEWLFINC, HEWLFINP
HEWLFENS	CSECT	HEWLFENS	ENTAB size determination	HEWLFADA
HEWLFENT	CSECT	HEWLFENT	Entry statement processing	HEWLFADA
HEWLFESD	CSECT	HEWLFESD	ESD record processing	HEWLFINP
HEWLFFNL	CSECT	HEWLFFNL	Final processing	HEWLFOUT, HEWLFROU, HEWLFSCD, HEWLFADA
HEWLFIDR	CSECT	HEWLFIDR	IDR record processing	HEWLFINP
HEWLFINC	CSECT	HEWLFINC	Include statement processing	HEWLFINP, HEWLFSCN
HEWLFINP	CSECT	HEWLFINP	Input processing	HEWLFINC, HEWLFINT, HEWLFROU
HEWLFINT	CSECT	HEWLFINT	Initialization	HEWLFROU
HEWLFLDB	Label	HEWLFROU	DCB for SYSLIB	HEWLFAPT, HEWLFFNL. HEWLFINC, HEWLFADA
HEWLFLOG	Entry point	HEWLFROU	ERROR LOGGING routine	HEWLFAPT
HEWLFMAP	CSECT	HEWLFMAP	MAP/CROSS-REFERENCE processing	HEWLFFNL, HEWLFOUT, HEWLFROU
HEWLFOPT	CSECT	HEWLFOPT	Options processing	HEWLFINT
HEWLFOUT	CSECT	HEWLFOUT	Intermediate output	HEWLFADA
HEWLFRAT	CSECT	HEWLFRAT	RLD record processing	HEWLFINC, HEWLFINP, HEWLFSCN
HEWLFRCG	CSECT	HEWLFRCG	REPLACE/CHANGE statement processing	HEWLFESD
HEWLFREL	CSECT	HEWLFREL	Relocation/second pass initialization	HEWLFSCD
HEWLFROU	CSECT	HEWLFROU	Miscellaneous routines/LOAD module entry point	

Figure 33 (Part 3 of 4). Microfiche Directory

Symbol	Type	CSECT	Description	Referenced By
HEWLFSCD	CSECT	HEWLFSCD	Second pass (LOAD module) output	HEWLFOUT, HEWLFROU
HEWLFSCN	CSECT	HEWLFSCN	Control statement processing	HEWLFINP
HEWLFSIO	CSECT	HEWLFSCD	Second pass input/output	HEWLFSCD
HEWLFSYM	CSECT	HEWLFSYM	SYM record processing	HEWLFSYM
HEWLFTXT	CSECT	HEWLFRAT	TXT record processing	HEWLFRAT
HEWLTMDB	Label	HEWLFROU	DCB for SYSTERM	HEWLFFNL, HEWLFINT
HEWLXIT2	Entry point	HEWLFINT	Open exit routine for SYSLMOD	HEWLFROU
HEWVLDCK	Entry point	HEWLFROU	Member and alias name validity check routine	HEWLFFNL, HEWLFSCN
INDDNAME	Label	HEWLFROU	DDNAME for primary input data set	HEWLFINT, HEWLFSCN
INRLDCB1	Label	HEWLFREL	Input RLD control block #1	HEWLFSCD
INRLDCB2	Label	HEWLFREL	Input RLD control block #2	HEWLFSCD
JFCBADDR	Label	HEWLFMAP	JFCB for SYSLMOD	HEWLFFNL
MAINGOT	Label	HEWLFINT	Address of storage obtained from GETMAIN	
MINOR	Label	HEWLFROU	Minor name by which SYSMOD is enqueued	HEWLFFNL, HEWLFINT
MSGFOUR	Label	HEWLFINT	Pointer to (optional) heading message	HEWLFROU
OTRLDCB1	Label	HEWLFREL	Output RLD control block #1	HEWLFSCD
OTRLDCB2	Label	HEWLFREL	Output RLD control block #2	HEWLFSCD
OTRLDCB3	Label	HEWLFREL	Output RLD control block #3	HEWLFSCD
RELOCATE	Entry point	HEWLFREL	Address constant relocation routine	HEWLFSCD
SCDENTAB	Entry point	HEWLFREL	Routine to create ENTABS and ENTAB RLDs	HEWLFSCD
SEGLNTAB	Label	HEWLFADA	Pointer to Segment Length Table	HEWLFOUT
WRTCRRLD	Entry point	HEWLFSIO	Routine to write CTL or CTL/RLD records on SYSLMOD	HEWLFREL
WRTTXT	Entry point	HEWLFSIO	Routine to write text records on SYSLMOD	HEWLFREL

Figure 33 (Part 4 of 4). Microfiche Directory

Module Name	CSECT Name
HEWLFADA	HEWLFADA
HEWLFAPT	HEWLFAPT
HEWLFBTP	HEWLFBTP
HEWLFDEF	HEWLFDEF
HEWLFEND	HEWLFEND
HEWLFENS	HEWLFENS
HEWLFENT	HEWLFENT
HEWLFESD	HEWLFESD
HEWLFFNL	HEWLFFNL
HEWLFIDR	HEWLFIDR
HEWLFINC	HEWLFINC
HEWLFINP	HEWLFINP
HEWLFOPT	HEWLFOPT
HEWLFMAP	HEWLFMAP
HEWLFINT	HEWLFINT
HEWLFOUT	HEWLFOUT
HEWLFRAT	HEWLFRAT, HEWLFTXT
HEWLFRCG	HEWLFRCG
HEWLFREL	HEWLFREL
HEWLFROU	HEWLFROU
HEWLFSCD	HEWLFSCD, HEWLFSIO
HEWLFSCN	HEWLFSCN
HEWLFSYM	HEWLFSYM

Figure 34. Module/CSECT Cross-Reference Table

TABLE LAYOUTS

This section provides detailed layouts of internal tables used during Linkage Editor processing. Figure 35 indicates the modules in which tables are initialized and used or modified. Tables described in this section are included alphabetically except for the All-Purpose Table (see Figure 36 on page 152).

Table	Built by	Used and/or Modified by
Alias Table	HEWLFENT	HEWLFFNL
All-Purpose Table (APT)	HEWLFINT	[1]
Calls List	HEWLFRAT	HEWLFENS
Composite External Symbol Dictionary (CESD)	HEWLFESD	HEWLFRAT, HEWLFSCN, HEWLFINC, HEWLFADA, HEWLFENS, HEWLFENT, HEWLFOUT, HEWLFTXT
Delink Table	HEWLFESD	HEWLFRAT, HEWLFSCD
Downward Calls List	HEWLFENS	[2]
Entry List	HEWLFSCD	[2]
Entry Table (ENTAB)	HEWLFSCD	[2]
Half ESD (HESD)	HEWLFOUT	HEWLFSCD
Half ESD Prefix	HEWLFSCD	[2]
High ID Table (HIID)	HEWLFOUT	[2]
IDR Translator Table (IDRTRTAB)	HEWLFIDR	HEWLFOUT
IDR IMASPZAP Table (IDRZPTAB)	HEWLFIDR	HEWLFOUT
IDR User Data Table (IDRUDTAB)	HEWLFIDR	HEWLFOUT
ORDER Table	HEWLFSCN	HEWLFOUT, HEWLFADA
Relocation Constant Table (RCT)	HEWLFADA	HEWLFOUT, HEWLFSCD
Renumbering Table (RNT)	HEWLFESD	HEWLFRAT, HEWLFTXT
RLD Input Control Blocks	HEWLFSCD	[2]
RLD Note List	HEWLFRAT	HEWLFOUT, HEWLFSCD
RLD Output Control Blocks	HEWLFSCD	[2]
Second Pass Text Control Blocks	HEWLFSCD	[2]
Segment Length Table (SEGLGTH)	HEWLFADA	[2]
Segment Path Table (SEGTA1)	HEWLFOUT	HEWLFSCD
Text I/O Table	HEWLFTXT	HEWLFOUT, HEWLFSCD
Text Note List	HEWLFTXT	HEWLFOUT, HEWLFSCD
TTR List (Text I/O Control Table)	HEWLFSCD	HEWLFSCD

Figure 35. Table Construction and Usage

Notes to Figure 35:

[1] Major communications area throughout linkage editor processing.

[2] Built and processed entirely within one routine.

Offset Decimal	Hex	Length	Symbol	Description
8	8	8	PDSE1	Member or alias name of module being created
16	10	3	PDSE2	Relative disk address (TTR) of first record of module
19	13	1	PDSE3	Flags
				Bit 0 Alias indicator
				Bits 1-2 Number of TTRs in user data
				Bits 3-7 Length of user data in half words
20	14	4	PDSE4	Relative disk address (TTR) of first text record of module
24	18	3	PDSE5	Relative disk address (TTR) of note list or scatter/translation record
27	1B	1	PDSE6	Number of TTRs in note list, if present
28	1C	1	PDSE7	Flags (module attributes #1)
				Bit 0 Reenterable
				Bit 1 Reusable
				Bit 2 Overlay
				Bit 3 Test
				Bit 4 Only loadable
				Bit 5 Block/scatter format
				Bit 6 Executable
				Bit 7 Module contains 1 text record and no RLD's
29	1D	1	PDSE8	Flags (module attributes #1)
				Bit 0 Output load module not downward compatible
				Bit 1 Origin of first text record is zero
				Bit 2 Entry point assigned by linkage editor is 0
				Bit 3 Module contains no RLD items
				Bit 4 Module can be reprocessed by linkage editor
				Bit 5 Module does not contain SYM records
				Bit 6 Module was created by link editor F

Figure 36 (Part 1 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
				Bit 7 Refreshable
30	1E	3	PDSE9	Total contiguous storage requirement for load module
33	21	2	PDSE10	Length of first text record
35	23	3	PDSE11	Entry point address
38	26	3	PDSE12	Editor assigned origin of first text record
38	26	1		Flags (1) (Module attributes #2)
				Bit 0 Load module built by OS/VS linkage editor
				Bit 1 Not used
				Bit 2 Page alignment required for load module
				Bit 3 SSI present in directory entry
				Bit 4 Directory entry contains authorization code
39	27	1		Flags (2) (Module attributes #2)
				Bits 0-2 Not used
				Bit 3 Load module residence mode
				Bits 4-5 Alias entry point addressing mode
				Bits 6-7 Main entry point addressing mode
40	28	1		Count of RLD and CTL/RLD records following the first text record
41	29	2	PDSE13	Number of bytes in scatter list
43	2B	2	PDSE14	Number of bytes in the Translation Table
45	2D	2	PDSE15	ESDID of the first text record
47	2F	2	PDSE16	ESDID of the control section containing the entry point
49	31	3	PDSE17	Entry point of main member name
52	34	8	PDSE18	Member name of module
60	3C	72	REGSA	Register save area for data management
132	84	24	IOCT	I/O Control Table
156	9C	1	APT0	Flags
				Bit 0 NCAL
				Bit 1 XREF
				Bit 2 MAP
				Bit 3 LET

Figure 36 (Part 2 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
				Bit 4 LOG
				Bit 5 XCAL
				Bit 6 TXT/RLD
				Bit 7 A library statement was read
157	9D	1	APT1	Flags
				Bit 0 More include input to come
				Bit 1 Automatic library call in operation
				Bit 2 Object or load module
				Bit 3 Delete indicator
				Bit 4 Entry point received
				Bit 5 Symbolic or absolute entry
				Bit 6 Entry statement received
				Bit 7 ESD write indicator
158	9E	1	APT2	Flags
				Bit 0 No length received
				Bit 1 No length indication
				Bit 2 First text record
				Bit 3 Status indicator received
				Bit 4 Include previously initiated
				Bit 5 I/O overlap bit
				Bit 6 In module indicator
				Bit 7 Card continuation
159	9F	1	APT3	Flags
				Bit 0 End of file
				Bit 1 Name statement received; end of input for load module
				Bit 2 End of SYSLIN input
				Bit 3 To stow as replacement
				Bit 4 First text of load module
				Bit 5 First text of segment
				Bit 6 RLDs for group
				Bit 7 SYSLIB opened
160	A0	4	CTTR	Relative disk address (TTR) of first CESD record, if MAP or SREF option specified

Figure 36 (Part 3 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
164	A4	2	CSNO	Current segment number
166	A6	2	CRNO	Current region number
168	A8	4	PRAL	Pseudo register cumulative length
172	AC	4	FLCD	Address of first deleted CESD entry
176	B0	4	RCCE	Address of replace/change chain end
180	B4	4	RCCB	Address of replace/change chain beginning
184	B8	4	ALCB	Address of alias chain beginning
188	BC	4	OVCMBGAD	Address of overlap chain beginning
192	C0	4	SGT1	Address of SEGTAB1 - 1
196	C4	4	CLLT	Address of Calls List Table
200	C8	4	RLDINPAD	Address of RLD input buffer, 1st pass
204	CC	4	RECNT	Address of Relocation Constant Table - 4; Renumbering Table - 4
208	D0	4	TXTIO	Address of Text I/O Table
212	D4	4	ALAS	Address of Alias Table
216	D8	4	DLKT	Address of DELINK Table - 5
220	DC	4	CHESD	Address of composite ESD - 16
224	E0	4	SELST	Address of second pass entry list
228	E4	4	TNLS2	Address of text note list 2
232	E8	4	RNLS2	Address of RLD note list 2
236	EC	4	TTRLIST	Address of TTR list
240	F0	4	RLDOUTBF	Address of output RLD buffer, 2nd pass
244	F4	4	HIARADD	Address of Hierarchy Table
248	F8	4	ORDRADR	Address of Order Table
252	FC	4	INCBRKPT	Address of breaking point in include chain
256	100	4	CRRTINCL	Address of currently included ESD item
260	104	2	ENRNX	Maximum number of entries in RNT Table
262	106	2	ENCDX	Maximum number of entries in C/HESD Tables
264	108	2	ENT2X	Maximum number of entries in text note list 2
266	10A	2	ENR2X	Maximum number of entries in RLD note list 2
268	10C	2	ENTOX	Maximum number of bytes in Text I/O Table
270	10E	2	ENCLX	Maximum number of bytes in calls list
272	110	2	ENDTX	Maximum number of entries in DELINK Table

Figure 36 (Part 4 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
274	112	2	ENS1X	Maximum number of segments
276	114	2	BUFSIZ	Size of load module input buffer
280	118	4	HESD	Address of HESD Table - 8
284	11C	2	ENELTX	Maximum number of entries in 2nd pass entry list
286	11E	2	IDRTRLEN	Maximum length of IDR Translator Data Table
288	120	2	IDRTILEN	Maximum length of IDR Translator ID Table
290	122	2	IDRUDLEN	Maximum length of IDR User Data Table
292	124	2	IDRZPLEN	Maximum length of IDR AMASPZAP Data Table
296	128	4	IDRTRTAB	Starting address of IDR Translator Data Table
300	12C	4	IDRTITAB	Starting address of IDR Translator ID Table
304	130	4	IDRUDTAB	Starting address of IDR User Data Table
308	134	4	IDRZPTAB	Starting address of IDR AMASPZAP Data Table
312	138	4	IDRTREND	Address of next available byte in IDR Translator Data Table
316	13C	4	IDRTIEND	Address of next available byte in IDR Translator ID Table
316	13C	4	IDRTIEND	Address of next available byte in IDR Translator ID Table
320	140	4	IDRUDEND	Address of next available byte in IDR User Data Table
324	144	4	IDRZPEND	Address of next available byte in IDR AMASPZAP Data Table
328	148	2	ENRLD2X	Maximum size of input RLD buffer, 1st pass
330	14A	2	ENSPX	Save area
332	14C	4	LSTS	Last segment in each region (region 1-4)
336	150	8	EPSM	Entry point symbol or end card address/symbol
344	158	2	ENTOC	Current number of bytes in TXT I/O Table
346	15A	2	ENCLC	Current number of bytes in calls list
348	15C	2	ENS1C	Current number of entries in SEGTAB1
350	15E	2	ENASC	Current number of entries in Alias Table
352	160	2	ENDTC	Current number of entries in DELINK Table
354	162	2	ENRNC	Current number of entries in RNT Table
356	164	2	ENCDC	Current number of entries in H/CESD Table
358	166	2	ENELTC	Current number of entries in 2nd pass entry list

Figure 36 (Part 5 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
360	168	2	ENT2C	Current number of entries in TXT note list 2
362	16A	2	ENR2C	Current number of entries in RLD note list 2
364	16C	2	ENSPC	Highest segment number with text
368	170	2	IDRTRCUR	Current number of bytes in IDR Translator Data Table
372	174	2	IDRTICUR	Current number of bytes in IDR Translator ID Table
376	178	2	IDRUDCUR	Current number of bytes in IDR User Data Table
380	17C	2	IDRZPCUR	Current number of bytes in IDR AMASPZAP Data Table
382	17E	2	ORDRCUR	Current number of bytes in Order Table
384	180	2	ORDRMAX	Maximum number of bytes in Order Table
388	184	8	BITMAP	Bit switches denoting error messages logged (error msgs 64-1)
396	18C	8	BITMAP2	Bit switches denoting error messages logged (error msgs 128-65)
404	194	2	LINECNT	Number of lines output for current page
406	196	2	HISEV	Highest severity message logged
408	198	8	SYSRTN	Save area for registers 13 and 14 from INVOKER
416	1A0	72	SPACES	Register save area
488	1E8	4	ERDIG	Address of HEWLFLOG, error logging routine
492	1EC	4	ERDIGA	Address of HEWLFALK, table allocation routine
496	1F0	4	SSI	System status indicator (for APT)
500	1F4	4	FFCADR	Highest address retained from gotten storage
504	1F8	8	LIBNAME	Name of library for automatic library call
512	200	8	LIBOPEN	Name of library currently open
520	208	2	APT000	SYNAD routine for SYSPRINT data set
522	20A	3	SAVATS	Attributes save area
525	20D	1	APTSWS	Switches
				Bit 0 TSO task
				Bit 1 Not used
				Bit 2 Absolute/relocatable
				Bit 3 DCBS override
				Bit 4 Bit map processed

Figure 36 (Part 6 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
				Bit 5 Linkage editor input received
				Bit 6 SYM received
				Bit 7 ESD received
526	20E	1	NEWSW	Flags
				Bit 0 If off, indicates 1st time in INT
				Bit 1 MAP/XREF entered from intermediate/final processor
				Bit 2 All RLDs in storage/not in storage
				Bit 3 MAP/XREF in control/not in control
				Bit 4 Normal printing on SYSPRINT/ABORT without printing
				Bit 5 HIERARCHY
				Bit 6 Not used
				Bit 7 Indicates purge to TXT/RLD processor
527	20F	1	NEWSW2	Flags
				Bit 0 More RLDs exist for current ID
				Bit 1 Split RLD in output buffer
				Bit 2 R and P pointer have been saved
				Bit 3 Relative/absolute relocation factor needed
				Bit 4 Split RLD has been saved in HESD prefix
				Bit 5 No RLDs exist for last text of segment/module
				Bit 6 Split RLD is preceded by R and P pointers
				Bit 7 R and P pointers for current chain are in buffer
528	210	1	APTSW2	Flags
				Bit 0 SYSLMOD enqueued
				Bit 1 Not used
				Bit 2 SYSLMOD shared DASD
				Bit 3 First/not first time through initialization
				Bits 4-7 Not used
529	211	1	APTSW3	Flags
				Bit 0 Expand statement encountered

Figure 36 (Part 7 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
				Bit 1 Included load module was in overlay format
				Bits 2-7 Reserved
530	212	1	APTSW4	Switches
531	213	1	IDRSWS	Flags
				Bits 0-2 Not used
				Bit 3 Last IDR item processed not complete
				Bit 4 Double IDR entry on object module record in process
				Bit 5 Identify control card in process
				Bit 6 Object module end card in process for IDR input
				Bit 7 Load module IDR in process
532	214	1	APT4	Flags
				Bit 0 First text record has been read
				Bit 1 Not used
				Bit 2 Intermediate pass processing
				Bit 3 Second pass processing
				Bit 4 Ordering required
				Bit 5 Page boundary alignment required
				Bit 6 Align on 2K-byte page boundary
				Bit 7 Not used
534	216	2	MAXBF	Maximum blocking factor
536	218	28	HEWLCRBB	SYSLIB control block
536	218	4		Address of SYSLIB DECB
540	21C	4		1st library buffer
544	220	4		2nd library buffer
548	224	2		BLKSIZE
550	226	2		LRECL
552	228	2		BLKFCTR
554	22A	2		Number of records left in buffer
556	22C	4		Address of current record
560	230	4		READSW set to first read
564	234	28	HEWLCRBN	SYSLIN control block

Figure 36 (Part 8 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
564	234	4		Address of SYSLIN DECB
568	238	4		1st SYSLIN buffer
572	23C	4		2nd SYSLIN buffer
576	240	2		BLKSIZE
578	242	2		LRECL
580	244	2		BLKFCTR
582	246	2		Number of records left in buffer
584	248	4		Address of current record
588	24C	4		READSW set to first read
592	250	28	HEWLCWBB	SYSPRINT control block
592	250	4		Address of SYSPRINT DCB
596	254	4		1st SYSPRINT buffer
600	258	4		2nd SYSPRINT buffer
604	25C	2		BLKSIZE
606	25E	2		LRECL
608	260	2		BLKFCTR
610	262	2		Number of records left in buffer
612	264	4		Address of current record
616	268	4		WRITESW set to first write
620	26C	4	RLDOUT1	Address of first RLD output buffer, 1st pass
620	26C	4	RLDINBF1	Address of first RLD input buffer, 2nd pass
624	270	4	RLDOUT2	Address of second RLD output buffer, 1st pass
624	270	4	RLDINBF2	Address of second RLD input buffer, 2nd pass
628	274	4	TXTBFBEG	Address of start of text buffer
632	278	4	TXTBFEND	Address of end of text buffer
636	27C	4	MULTSIZE	Size of SYSLMOD multiplicity or record
640	280	4	UT1SIZE	Size of SYSUT1 record
644	284	4	SZSYSUT1	Maximum number of bytes per track on SYSUT1
648	288	4	RLDSIZE	Size of each input RLD buffer, 1st pass
652	28C	4	VALUE1	Size value 1 (maximum allowable storage)
656	290	4	VALUE2	Size value 2 (load module buffer)
660	294	4	MSGONE	Pointer to 1st heading message
664	298	4	MSGTWO	Pointer to 2nd heading message

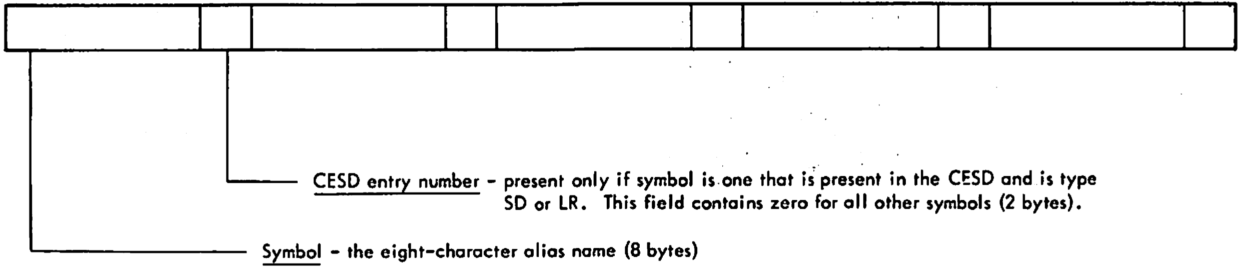
Figure 36 (Part 9 of 10). All-Purpose Table (APT)

Offset Decimal	Hex	Length	Symbol	Description
668	29C	4	MSGTHREE	Pointer to 3rd heading message
672	2A0	4	HEWLCLAC	Address of current read block
676	2A4	20		DECB for SYSLIN
696	2B8	20		DECB for SYSLIB
716	2CC	4	COREADR	Address of storage obtained through GETMAIN
720	2D0	4	CORELEN	Length of storage obtained through GETMAIN
724	2D4	64	BRNCHSV	Save area
788	314	1	APTAPFCT	Default length of authorization code
789	315	1	APTAPFAC	Default authorization code
790	316	1	PDSAPFCT	Length of authorization code assigned
791	317	1	PDSAPFAC	Authorization code assigned
792	318	1	MODEAMOD	Addressing mode from mode control statement
793	319	1	MODERMOD	Residence mode from mode control statement
794	31A	1	PARMAMOD	Addressing mode from PARM field
795	31B	1	PARMRMOD	Residence mode from PARM field
796	31C	1	MEMBAMOD	Addressing mode for main entry point
797	31D	1	ESDARMOD	Residence mode accumulated from ESDs

Figure 36 (Part 10 of 10). All-Purpose Table (APT)

Alias Table

Built by Entry Processor
Referred to by Final Processor



CESD entry number - present only if symbol is one that is present in the CESD and is type SD or LR. This field contains zero for all other symbols (2 bytes).

Symbol - the eight-character alias name (8 bytes)

Figure 37. Alias Table

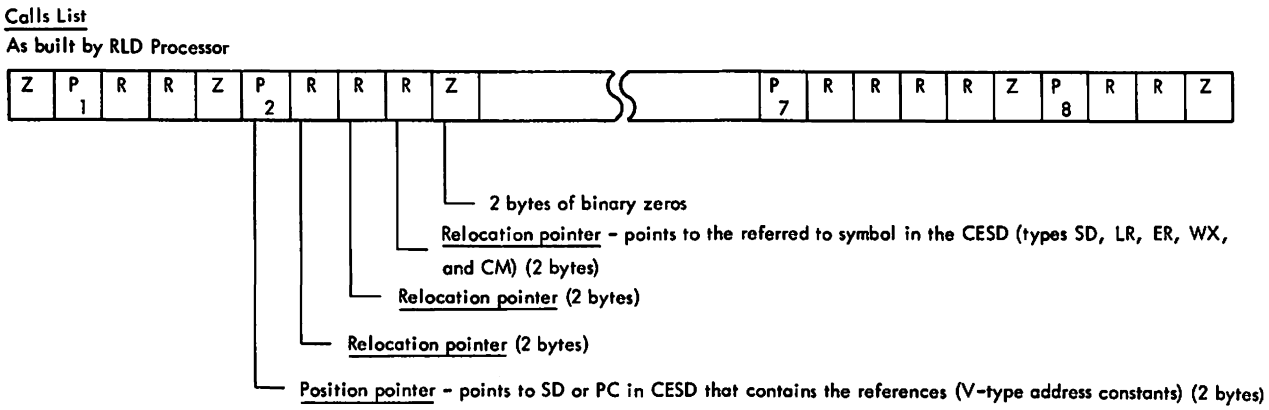

Calls List
As built by RLD Processor
Z
P 1
R
R
Z
P 2
R
R
R
Z
P 7
R
R
R
R
Z
P 8
R
R
Z
2 bytes of binary zeros
Relocation pointer - points to the referred to symbol in the CESD (types SD, LR, ER, WX, and CM) (2 bytes)
Relocation pointer (2 bytes)
Relocation pointer (2 bytes)
Position pointer - points to SD or PC in CESD that contains the references (V-type address constants) (2 bytes)

Figure 38. Calls List (As Built by RLD Processor)

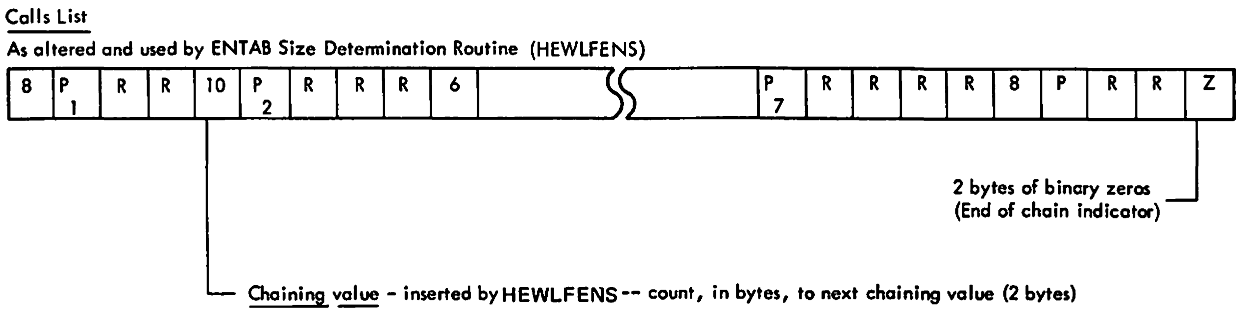
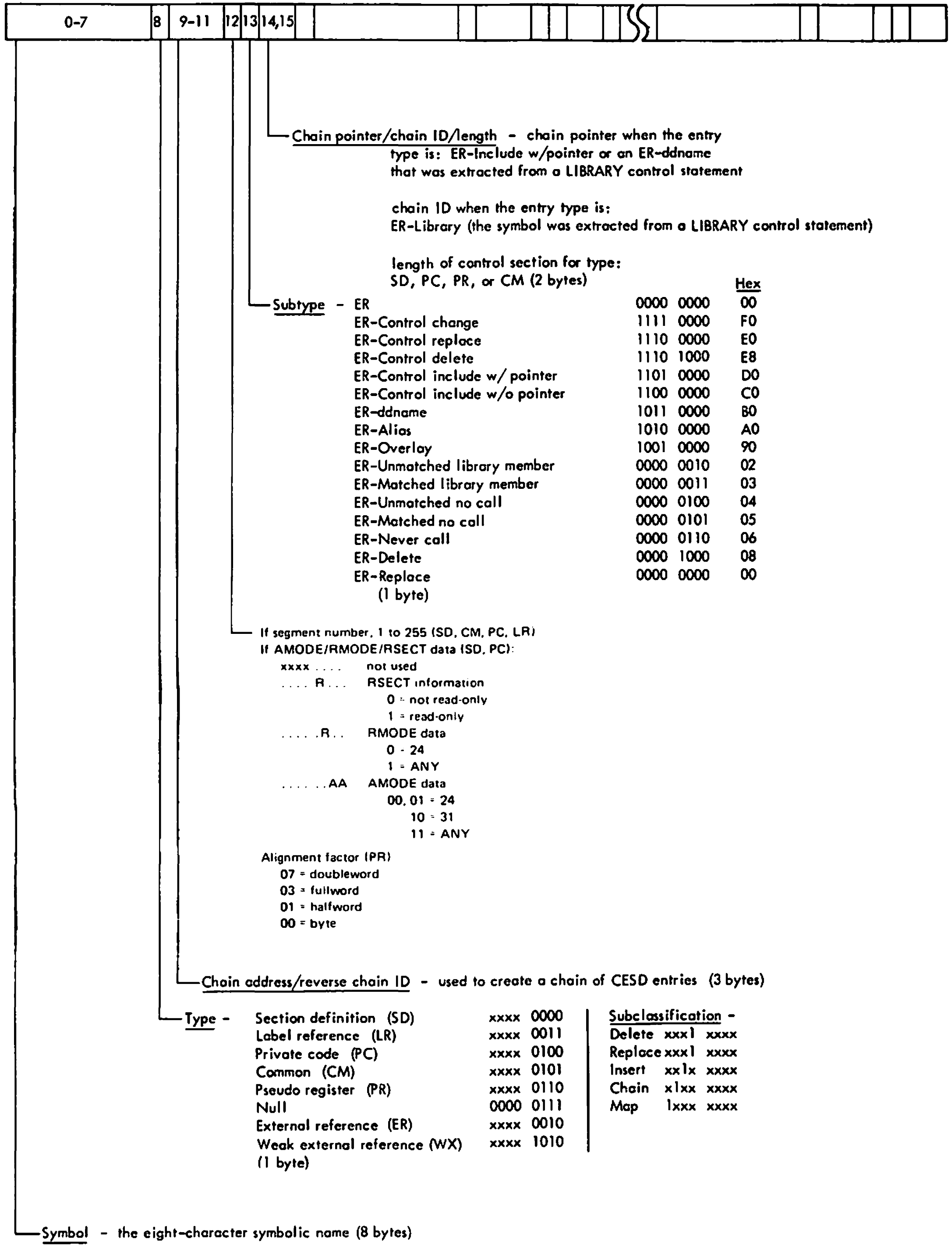

Calls List
As altered and used by ENTAB Size Determination Routine (HEWLFENS)
8
P 1
R
R
10
P 2
R
R
R
6
P 7
R
R
R
R
8
P
R
R
Z
2 bytes of binary zeros
(End of chain indicator)
Chaining value - inserted by HEWLFENS -- count, in bytes, to next chaining value (2 bytes)

Figure 39. Calls List (As Altered and Used by ENTAB Size Determination Routine)

Composite External Symbol Dictionary (CESD) - Internal Format

Built by ESD Processor and Control Statement Processors
Modified by Address Assignment Processor



Chain pointer/chain ID/length - chain pointer when the entry type is: ER-Include w/pointer or an ER-ddname that was extracted from a LIBRARY control statement

chain ID when the entry type is:
ER-Library (the symbol was extracted from a LIBRARY control statement)

length of control section for type:
SD, PC, PR, or CM (2 bytes)

Subtype -		Hex
ER	0000 0000	00
ER-Control change	1111 0000	F0
ER-Control replace	1110 0000	E0
ER-Control delete	1110 1000	E8
ER-Control include w/ pointer	1101 0000	D0
ER-Control include w/o pointer	1100 0000	C0
ER-ddname	1011 0000	B0
ER-Alias	1010 0000	A0
ER-Overlay	1001 0000	90
ER-Unmatched library member	0000 0010	02
ER-Matched library member	0000 0011	03
ER-Unmatched no call	0000 0100	04
ER-Matched no call	0000 0101	05
ER-Never call	0000 0110	06
ER-Delete	0000 1000	08
ER-Replace	0000 0000	00
(1 byte)		

If segment number, 1 to 255 (SD, CM, PC, LR)
If AMODE/RMODE/RSECT data (SD, PC):

xxxx	not used
.... R ...	RSECT information 0 = not read-only 1 = read-only
..... R ..	RMODE data 0 - 24 1 = ANY
...... AA	AMODE data 00, 01 = 24 10 = 31 11 = ANY

Alignment factor (PR)
07 = doubleword
03 = fullword
01 = halfword
00 = byte

Chain address/reverse chain ID - used to create a chain of CESD entries (3 bytes)

Type -	
Section definition (SD)	xxxx 0000
Label reference (LR)	xxxx 0011
Private code (PC)	xxxx 0100
Common (CM)	xxxx 0101
Pseudo register (PR)	xxxx 0110
Null	0000 0111
External reference (ER)	xxxx 0010
Weak external reference (WX)	xxxx 1010
(1 byte)	

Subclassification -	
Delete	xxx1 xxxx
Replace	xxx1 xxxx
Insert	xx1x xxxx
Chain	x1xx xxxx
Map	1xxx xxxx

Symbol - the eight-character symbolic name (8 bytes)

Figure 40. Composite External Symbol Dictionary (CESD)—Internal Format

See Figure 41 for normal combination of internal CESD types.

CESD Entry Type	Type Field (byte 8)	Chain Address Chain ID (bytes 9-11)	AMODE/RMODE/RSECT Data or Segment Number (byte 12)	ER Subtype (byte 13)	ddname Pointer/ Chain ID/Length (bytes 14-15)
Section Definition	xxxx x000		(5)	Length of control section	
Private Code	xxxx x100		(5)	Length of control section	
Common	xxxx x101		(6)	Length of common area	
Pseudo Register	xxxx x110		Alignment value (1)	Length of pseudo register	
External Reference	xxxx 0010	Hex 00 or 80		0000 0000	
Weak External Reference	xxxx 1010	Hex 00		0000 0000	
Label Reference	xxxx x011		(6)		CESD entry no. of SD or PC (ID)
NULL	0000 0111				
Replace	xxx1 xxxx			0000 0000	
Insert	xx1x xxxx				
Chain	x1xx xxxx				
Map	1xxx xxxx				
Delete	xxx1 xxxx			0000 1000	
ER - Unmatched Library Member Name	0000 0010	Reverse chain ID		0000 0010	CESD entry no. of next item (ID)
ER - Matched Library Member Name	0000 0010	Reverse chain ID (2)		0000 0011	CESD entry no. of next item (ID)
ER - Unmatched No Call Name	0000 0010			0000 0100	
ER - Matched No Call	0000 0010			0000 0101	
ER - Never Call	0000 0010			0000 0110	
ER - Overlay Control Statement	0000 0010	Address of next item in the chain		1001 0000	
ER - Alias Control Statement	0000 0010	Address of next item in the chain		1010 0000	
ER - ddname from Library or Include Statement	0000 0010			1011 0000	Forward chain PTR (Library only)
ER - Include Control Statement w/o Pointer	0000 0010	Address of next item in the chain		1100 0000	
ER - Include Control Statement with Pointer	0000 0010	Address of next item in the chain		1101 0000	Pointer to library's ddname
ER - Replace Control Statement (3)	0000 0010	Address of next item in the chain		1100 0000	
ER - Control Delete (4)	0000 0010	Address of next item in the chain		1110 1000	
ER - Change Control Statement (3)	0000 0010	Address of next item in the chain		1111 0000	

Figure 41 (Part 1 of 2). Normal Combination of Internal CESD Types

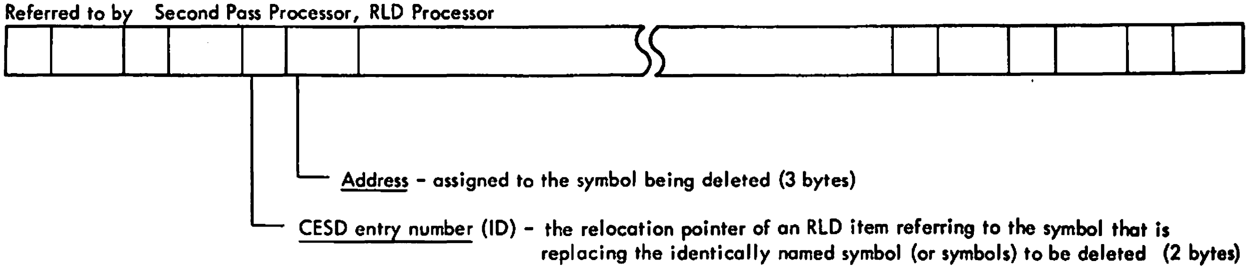
Notes:

1. Alignment Value -- Specifies boundary alignment of the pseudo register
   00 = byte alignment
   01 = halfword alignment
   03 = fullword alignment
   07 = doubleword alignment
2. BLDL has been issued for this member name if bit 64 is set to 1.
3. Two CESD entries are made for each Replace or Change control statement, one entry for each symbol.
4. This entry results from a Replace or Change control statement that contains only a single symbolic name.
5. If segment number, 1 to 255.
   If AMODE/RMODE/RSECT data:
   xxxx.... not used
   ....R... RSECT information
   0 = not read-only
   1 = read-only
   .....R.. RMODE data
   0 = 24
   1 = ANY
   ......AA AMODE data
   00, 01 = 24
   10 = 31
   11 = ANY
6. If segment number, 1 to 255.
   Otherwise, zero or blank.

Figure 41 (Part 2 of 2). Normal Combination of Internal CESD Types

Delink Table

Built by RLD Processor (Delink Routine),
Referred to by Second Pass Processor, RLD Processor



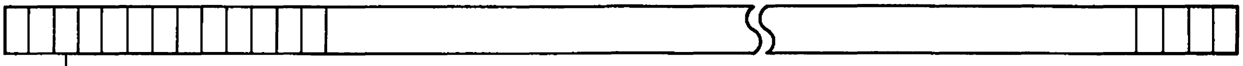
Address - assigned to the symbol being deleted (3 bytes)

CESD entry number (ID) - the relocation pointer of an RLD item referring to the symbol that is replacing the identically named symbol (or symbols) to be deleted (2 bytes)

Figure 42. Delink Table

Downward Calls List

Built by and referred to by ENTAB Size Determination Routine (HEWLFENS)



Segment number - entries are one for one with those of the CESD. If a downward call is made to a symbol, the segment's number from which the call is made is entered in the downward calls list at an entry corresponding to the ESDID of the symbol in the CESD. The list is initially zero. (1 byte)

Figure 43. Downward Calls List

Entry List

Built by and referred to by Second Pass Processor

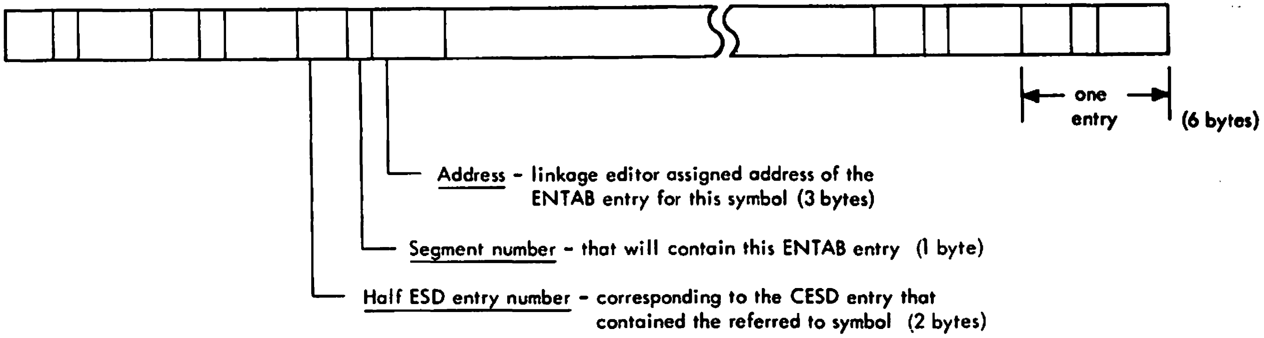

one entry
(6 bytes)
Address - linkage editor assigned address of the ENTAB entry for this symbol (3 bytes)
Segment number - that will contain this ENTAB entry (1 byte)
Half ESD entry number - corresponding to the CESD entry that contained the referred to symbol (2 bytes)

Figure 44. Entry List

Entry Table (ENTAB)

Built by Second Pass Processor

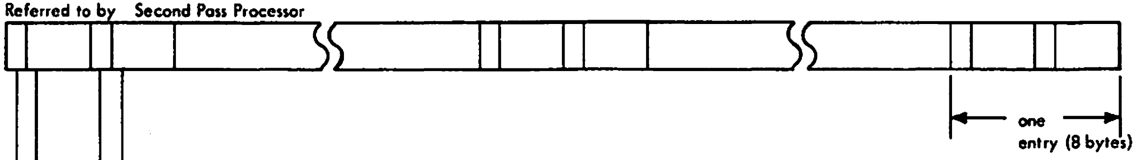
<table>
<tr><td colspan="2">Unconditional branch to last entry-BC 15, DISP (15,0)</td><td colspan="2">Address of referred to symbol</td><td>"to" seg number</td><td>Previous Caller (zero initially)</td></tr>
<tr><td colspan="2">Unconditional branch to last entry-BC 15, DISP (15,0)</td><td colspan="2">Address of referred to symbol</td><td>"to" seg number</td><td>Previous Caller (zero initially)</td></tr>
<tr><td colspan="2">.
.
.</td><td colspan="2">.
.
.</td><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td colspan="2">Unconditional branch to last entry-BC 15, DISP (15,0)</td><td colspan="2">Address of referred to symbol</td><td>"to" seg number</td><td>Previous Caller (zero initially)</td></tr>
<tr><td>SVC 45</td><td colspan="2">L 15, 4 (0,15) loads GR15 with the value of the adcon</td><td>BCR 15,15</td><td>"from" seg no</td><td>Address of segment table (SEGTAB)</td></tr>
<tr><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>2 bytes</td><td>1 byte</td><td>3 bytes</td></tr>
</table>

DISP -- is the displacement, in bytes, of this entry from the last entry.
"to" segment number -- is the number of the segment containing the symbol being referred to.
"from" segment number -- is the number of the segment that contains this entry table.

Figure 45. Entry Table (ENTAB)

Half External Symbol Dictionary

Built by Intermediate Output Processor
Referred to by Second Pass Processor



Length (3 bytes)

```
If segment number, 1 to 255 (SD, CM, PC, LR)
If AMODE/RMODE/RSECT data (SD, PC):
   xxxx ....    not used
   .... R...    RSECT information
                    0 = not read-only
                    1 = read-only
   .... .R..    RMODE data
                    0 = 24
                    1 = ANY
   .... ..AA    AMODE data
                    00, 01 = 24
                        10 = 31
                        11 = ANY
Alignment factor (PR)
   07 = doubleword
   03 = fullword
   01 = halfword
   00 = byte
```

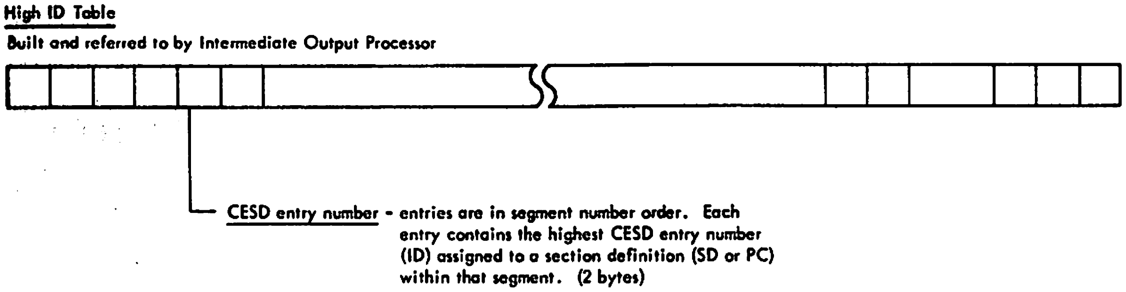
Linkage Editor assigned address - of this symbol (absolute value of the address constant) (3 bytes)

```
Indicator-Type - Bit zero is not used.  Bits 1, 2 and 3 are used as an indicator field that applies to:
          SD,PC - Bit 1 = 0 -- this control section (SD or PC) does not have
                               the highest CESD entry number with text in this segment
                        = 1 -- this control section (SD or PC) has the
                               highest CESD entry number in this seament
    SD, PC, or CM - Bit 2 = 0 -- relative relocation constant is a positive value
                          = 1 -- relative relocation constant is in
                                 complemented form
    PC-delete - Bit 3 = 1 -- indicates that this unnamed control section
                             is a SEGTAB or ENTAB.
```

Bits 4, 5, 6 and 7 are used to specify the entry type:

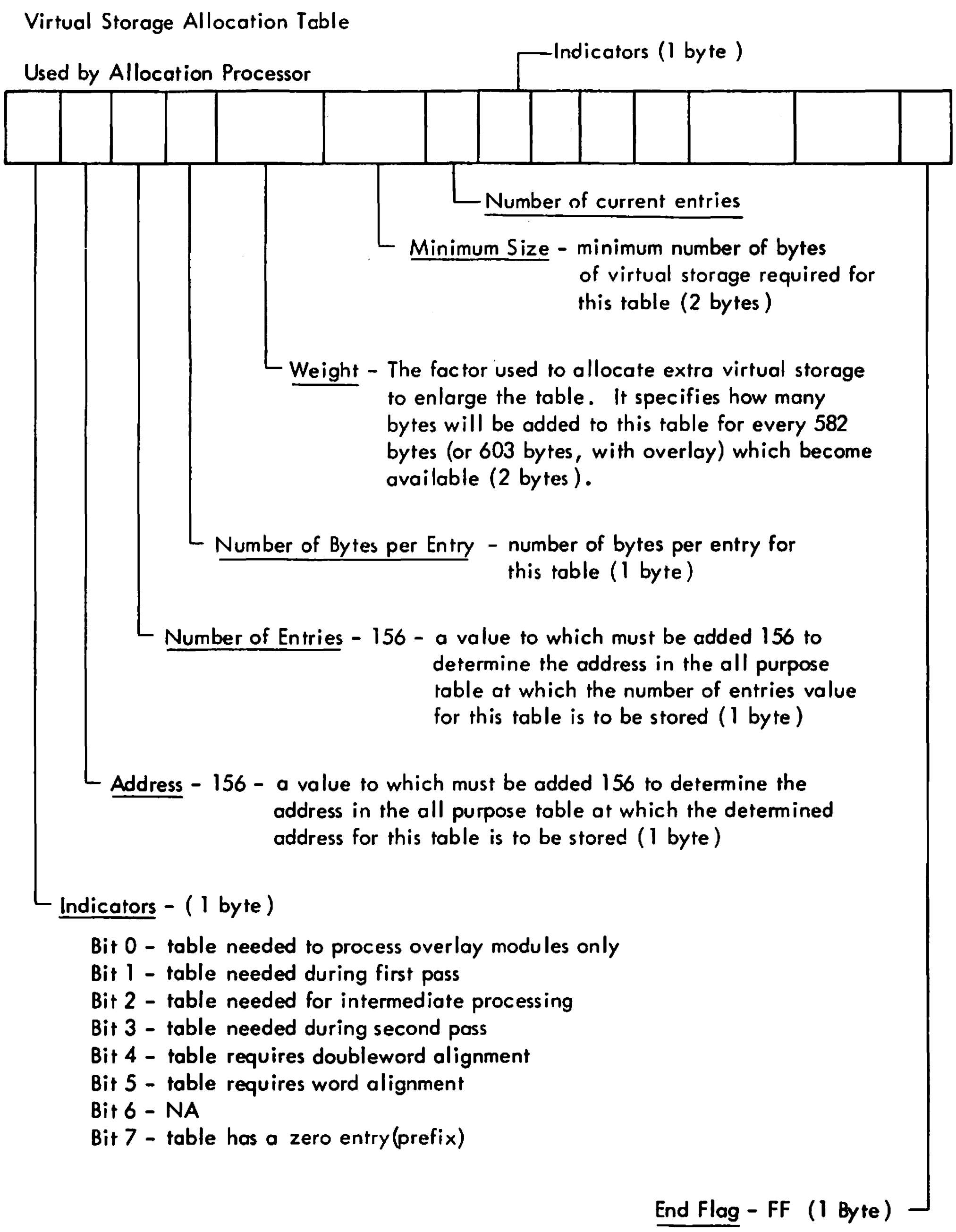
```
0000 = Section Definition (SD)
0010 = External Reference (ER) - all fields are zero except type
1010 = Weak External Reference (WX)
0011 = Label Reference (LR)
0100 = Private Code (PC)
0101 = Common (CM)
0110 = Pseudo Register (PR)
0111 = Null - all fields are zero except type
```

Figure 46. Half External Symbol Symbol Dictionary (HESD)


High ID Table
Built and referred to by Intermediate Output Processor
CESD entry number - entries are in segment number order. Each entry contains the highest CESD entry number (ID) assigned to a section definition (SD or PC) within that segment. (2 bytes)

Note: If segment does not contain text, its corresponding entry contains zero.

Figure 47. High ID Table (HIID)



Virtual Storage Allocation Table

Used by Allocation Processor

Indicators (1 byte)

Number of current entries

Minimum Size - minimum number of bytes of virtual storage required for this table (2 bytes)

Weight - The factor used to allocate extra virtual storage to enlarge the table. It specifies how many bytes will be added to this table for every 582 bytes (or 603 bytes, with overlay) which become available (2 bytes).

Number of Bytes per Entry - number of bytes per entry for this table (1 byte)

Number of Entries - 156 - a value to which must be added 156 to determine the address in the all purpose table at which the number of entries value for this table is to be stored (1 byte)

Address - 156 - a value to which must be added 156 to determine the address in the all purpose table at which the determined address for this table is to be stored (1 byte)

Indicators - (1 byte)

Bit 0 - table needed to process overlay modules only
Bit 1 - table needed during first pass
Bit 2 - table needed for intermediate processing
Bit 3 - table needed during second pass
Bit 4 - table requires doubleword alignment
Bit 5 - table requires word alignment
Bit 6 - NA
Bit 7 - table has a zero entry(prefix)

End Flag - FF (1 Byte)

Figure 48. Virtual Storage Allocation Table

Partitioned Organization Directory Record
As received from BLDL

Byte				
0	Name of load module (member or alias name)			
4				
8	Relative (to beginning of data set) track address of module (TTR)			Concatenation number
12	Byte of binary zeros *	Alias indicator and miscellaneous info	Relative (to beginning of data set) track address of first text record	
16	Continuation of track address	Byte of binary Zeros	Relative (to beginning of data set) track address of note list or scatter-	
20	translation record	Number of entries in note list **	Module attributes1 (see Figure 50) 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15	
24	Total contiguous quantity of virtual storage requred by the module			Length (in bytes) of first text record
28	Continuation of length	Module's linkage editor assigned entry point address		
32	Module Attributes 2 (see Figure 50)	AMODE/RMODE information (see Figure 50)	RLD count	

For load modules in scatter format add:

Byte				
				Length of scatter
36	list (in bytes)	Length of translation table (in bytes)		ESDID (CESD entry number of control
40	section name) for first text record	ESDID (CESD entry number of control section name) containing entry point		

For load modules with alias names add:

Byte				
				Entry point address
44	of the member name		Member name	
48	Member name			
52				

SSI Bytes - Aligned on a halfword boundary at the end of the PDS record

Legend:

Alias indicator and miscellaneous information:

Bit	Meaning
0	0 signifies none 1 signifies alias
1,2	number of relative track addresses (TTR) in user data field
3-7	length of user data field (in halfwords)

PODS Directory Record Size:

Format	Bytes
Block	36 (with alias names, it is 46 bytes)
Scatter	44 (with alias names, it is 54 bytes)

Note: For SSI, add 4 bytes to sizes given above.

*This is normally a zero byte inserted to maintain halfword boundaries. If the DCB operand was specified as zero and the name was found in the link library, this byte will contain a 1; if the name was found in the job library, this byte will contain a 2.

**This byte contains a zero if load module is not in overlay.

Figure 49. Partitioned Organization Directory Record (As Received from BLDL)

Module Attributes 1

Bit Number	Attributes	Bit Setting	Indication
0	RENT	0	Not reenterable
		1	Reenterable
1	REUS	0	Not reusable
		1	Reusable
2	OVLY	0	Not an overlay module
		1	Overlay module
3	TEST	0	Not under test
		1	Under test
4	LOAD	0	Not only loadable
		1	Only loadable[1]
5	Format	0	Block format
		1	Scatter format
6	Executable	0	Not executable
		1	Executable
7	Format	0	Module contains more than one text record and/or RLD record(s)
		1	Module contains only one text record and no RLD record
8	Compatibility	0	Module can be reprocessed by all levels of linkage editor
		1	Module cannot be reprocessed by linkage editor E
9	Format	0	Linkage editor assigned origin of first text record is not zero
		1	Linkage editor assigned origin of first text record is zero
10	Format	0	Linkage editor assigned entry point is not zero
		1	Linkage editor assigned entry point is zero
11	Format	0	Module contains RLD record(s)
		1	Module does not contain an RLD record
12	Editability	0	Module can be reprocessed by linkage editor
		1	Module cannot be reprocessed by linkage editor
13	Format	0	Module does not contain TESTRAN symbol records
		1	Module contains TESTRAN symbol records
14	Compatibility	1	Module created by linkage editor F
15	REFR	0	Module is not refreshable

Note:

[1] Module can be loaded only with the LOAD macro instruction. When the module is in virtual storage, it is entered directly, not through the use of an XCTL, LINK, or ATTACH macro instruction.

Figure 50 (Part 1 of 2). Module Attributes

Module Attributes 2

Bit Number	Bit Setting	Indication
0	1	Module has been processed by OS/VS linkage editor
1	0	Reserved - Unused
2	1	Page alignment required for load module
3	1	SSI present
4	1	Authorization code present in last 2 bytes of directory entry

AMODE/RMODE Information

xxx.....	Not Used
...R....	RMODE for Load Module 0 = 24 1 = ANY
....AA..	AMODE for the True Alias or Alternate Entry Point 00 = 24 10 = 31 11 = ANY
......AA	AMODE for the Main Entry Point 00 = 24 10 = 31 11 = ANY

Figure 50 (Part 2 of 2). Module Attributes

Partitioned Organization Directory Record

As built by linkage editor

Byte				
0 4	Name of load module (member or alias name)			
8	Relative (to beginning of data set)track address of module (TTR)			Alias indicator and miscellaneous info (see below)
12	Relative (to beginning of data set)track address of first text record (TTR)			Byte of binary zeros
16	Relative (to beginning of data set)track address of note list or scatter/translation record (TTR)			Number of entries in note list*
20	Module Attributes 1 (see Figure 50) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15		Total contiguous virtual storage required	
24	for the module	Length (in bytes) of first text record		Module's linkage
28	editor assigned entry point address		Module Attributes 2 (see Figure 50)	AMODE/RMODE information (see Figure 50)
32	RLD count	Authorization Code Length**	Authorization Code**	

For load modules in scatter format add:

Byte				
		Length of scatter list (in bytes)		Length of transla-
36	tion table (in bytes)	ESDID (CESD entry number of control section name) for first text record		ESDID (CESD entry number of control
40	section name) containing entry point	Authorization Code Length**	Authorization Code**	

For load modules with alias names add:

Byte				
		Entry point address of the member name		
44 48	Member name			
52	Authorization Code Length**	Authorization Code**		

For load modules with SSI bytes:

SSI bytes - Aligned on a halfword boundary at the end of the PDS record.			
Authorization Code Length**	Authorization Code**		

Legend:

Alias indicator and miscellaneous information:

Bit	Meaning
0	0 signifies none 1 signifies alias
1,2	number of relative track addresses (TTR) in user data field
3-7	length of user data field (in halfwords)

PODS Directory Record size:

Format	Bytes
Block	36 -- when rounded to a halfword boundary (with alias names, 44 bytes)
Scatter	44 (with alias names, 54 bytes)

Note: For SSI, add 4 bytes to sizes given above.

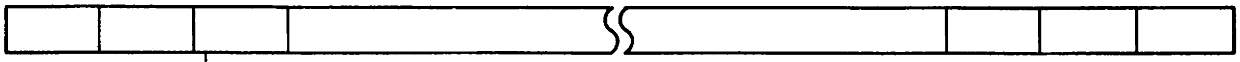
*This byte contains a zero if load module is not in overlay.

**The authorization code fields will appear only once. They are always the last fields of the directory record. The size of the directory determines in which of the four locations they appear.

Figure 51. Partitioned Organization Directory Record (As Built by Linkage Editor)

Relative Relocation Constant Table

Built by and referred to by Address Assignment Processor

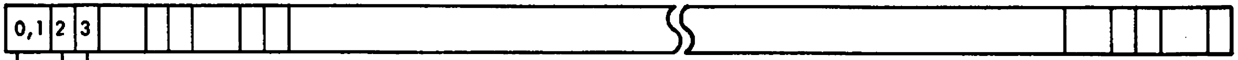


Relocation Constant = (linkage editor assigned address)-(previously assigned address) of a control section (SD, PC or CM) or a label reference (LR). The entries are one for one with CESD, in true or complement form. Complement form specified by binary ones in the high-order byte (4 bytes)

Figure 52. Relocation Constant Table (RCT)

Renumbering Table (RNT)

Built by ESD Processor
Referred to by TXT, RLD, END and ESD Processor



Type -

Section Definition (SD)	xxxx 0000
Label Reference (LR)	xxxx 0011
Private Code (PC)	xxxx 0100
Common (CM)	xxxx 0101
Pseudo Register (PR)	xxxx 0110
Null	0000 0111
External Reference	xxxx 0010
Weak External Reference (WX)	xxxx 1010

(1 byte)

Subclassification -

Delete	xxx1 xxxx
Replace	xxx1 xxxx
Chain	x1xx xxxx
Insert	xx1x xxxx
Library	1xxx xxxx

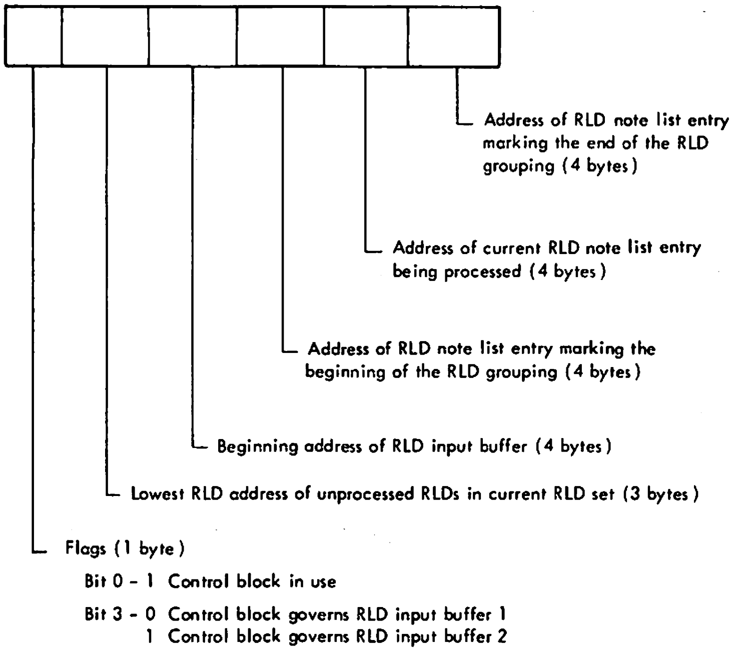
Flag - to indicate that the section definition (SD or PC) to which this entry corresponds is present in the CESD (0000 0001), or that other CESD items are dependent on its presence (0000 0010), or that a Delink Table entry was created for this symbol (0000 0100) -- 1 byte

CESD entry number (ID) - points to an entry in the CESD -- 2 bytes

Figure 53. Renumbering Table (RNT)

RLD Input Control Block*

Build and referred to by Second Pass RLD Processor



Address of RLD note list entry marking the end of the RLD grouping (4 bytes)

Address of current RLD note list entry being processed (4 bytes)

Address of RLD note list entry marking the beginning of the RLD grouping (4 bytes)

Beginning address of RLD input buffer (4 bytes)

Lowest RLD address of unprocessed RLDs in current RLD set (3 bytes)

Flags (1 byte)

Bit 0 - 1 Control block in use

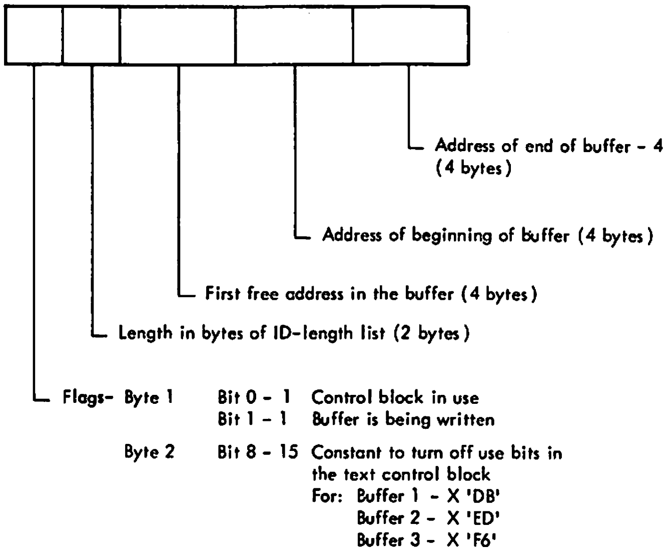
Bit 3 - 0 Control block governs RLD input buffer 1
1 Control block governs RLD input buffer 2

* There is a control block for each of two input buffers.

Figure 54. RLD Input Control Block

RLD Output Control Block *

Built and referred to by Second Pass RLD Processor



Address of end of buffer - 4
(4 bytes)

Address of beginning of buffer (4 bytes)

First free address in the buffer (4 bytes)

Length in bytes of ID-length list (2 bytes)

Flags- Byte 1 Bit 0 - 1 Control block in use
Bit 1 - 1 Buffer is being written

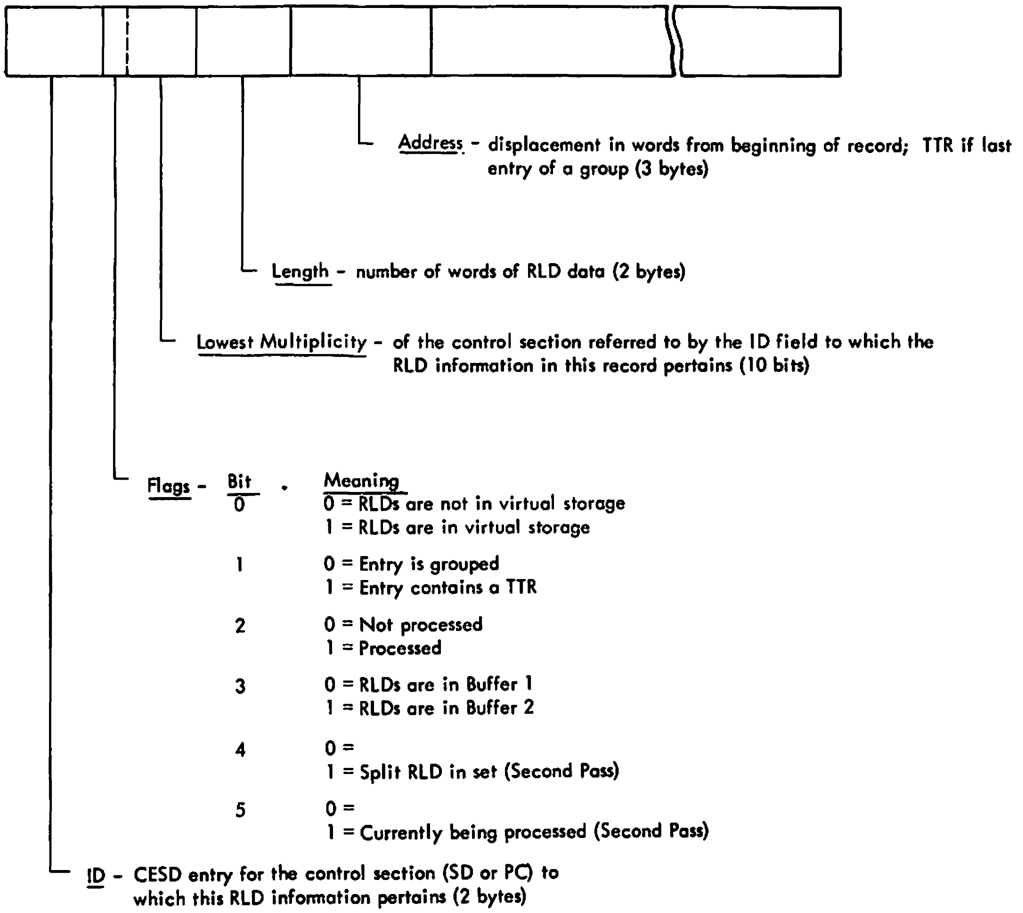
Byte 2 Bit 8 - 15 Constant to turn off use bits in the text control block
For: Buffer 1 - X 'DB'
Buffer 2 - X 'ED'
Buffer 3 - X 'F6'

* There is a control block for each of three RLD output buffers.

Figure 55. RLD Output Control Block

RLD Note List

Built and referred to by First Pass RLD Processor



Address - displacement in words from beginning of record; TTR if last entry of a group (3 bytes)

Length - number of words of RLD data (2 bytes)

Lowest Multiplicity - of the control section referred to by the ID field to which the RLD information in this record pertains (10 bits)

Flags -

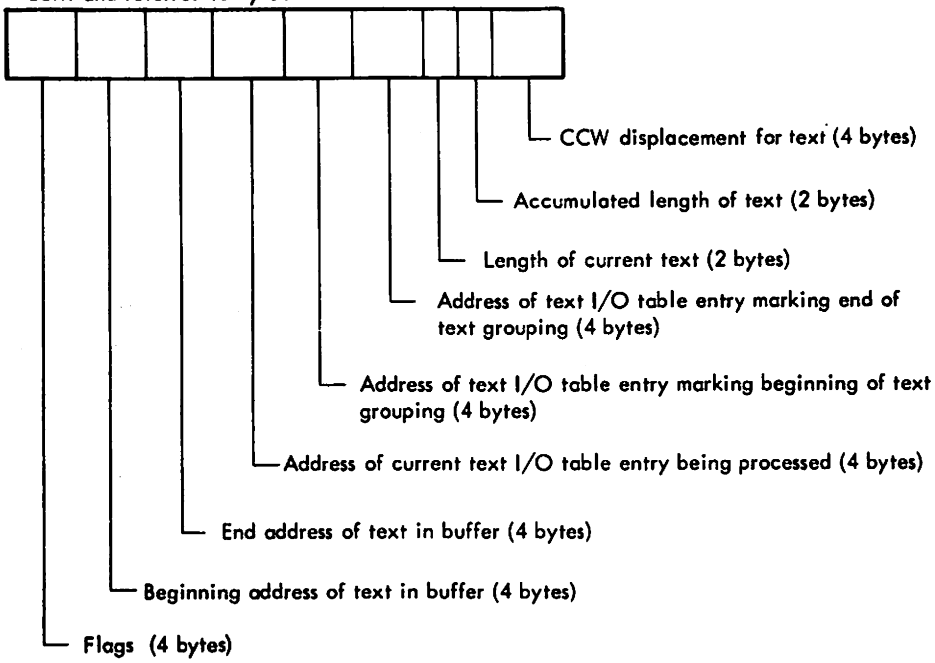
Bit	Meaning
0	0 = RLDs are not in virtual storage 1 = RLDs are in virtual storage
1	0 = Entry is grouped 1 = Entry contains a TTR
2	0 = Not processed 1 = Processed
3	0 = RLDs are in Buffer 1 1 = RLDs are in Buffer 2
4	0 = 1 = Split RLD in set (Second Pass)
5	0 = 1 = Currently being processed (Second Pass)

ID - CESD entry for the control section (SD or PC) to which this RLD information pertains (2 bytes)

Figure 56. RLD Note List

Second Pass Text Control Block*

Built and referred to by Second Pass Text Processor



CCW displacement for text (4 bytes)

Accumulated length of text (2 bytes)

Length of current text (2 bytes)

Address of text I/O table entry marking end of text grouping (4 bytes)

Address of text I/O table entry marking beginning of text grouping (4 bytes)

Address of current text I/O table entry being processed (4 bytes)

End address of text in buffer (4 bytes)

Beginning address of text in buffer (4 bytes)

Flags (4 bytes)

Byte 1	Bit 0 - 1	Control block in use
	1 - 1	Text being written
	2 - 1	Text being read
	3 - 1	Text has RLDs
	4 - 1	Text is first of group.
	5 - 1	Text is last of group
	6 - 1	Text is last in segment
	7 - 1	Text is last in load module
Byte 2	Bit 0 - 1	XDAP write needed
	1 - 1	Dummy write needed
	2 - 1	RLD output buffer 1 is being used
	3 - 1	RLD output buffer 2 is being used
	4 - 1	RLD output buffer 3 is being used
	5 - 1	RLD output buffer 1 contains ID-length list for this text
	6 - 1	RLD output buffer 2 contains ID-length list for this text
	7 - 1	RLD output buffer 3 contains ID-length list for this text
Byte 3	Bit 0 - 1	RLD input buffer 1 contains RLDs for this text
	1 - 1	RLD input buffer 1 contains processed RLDs for this text
	2 - 1	RLD input buffer 2 contains RLDs for this text
	3 - 1	RLD input buffer 2 contains processed RLDs for this text
	4 - 1	There is more text to process after current text

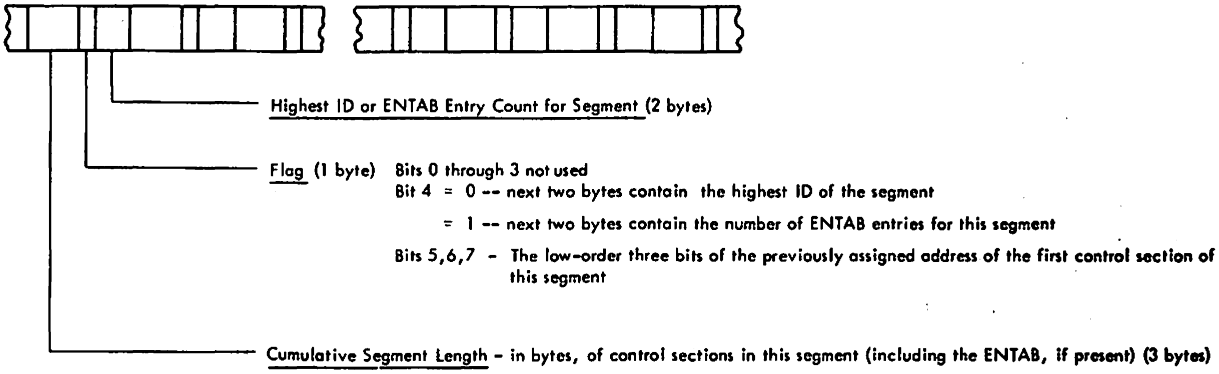
* There are two text control blocks - - one for current text being processed, another for next text to be processed or text just processed.

Figure 57. Second Pass Text Control Block

Segment Length Table (SEGLGTH)

Built and referred to by Address Assignment Processor

Appearance of table after assignment of control section addresses

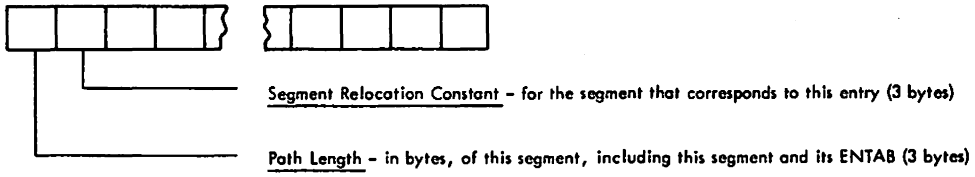


Highest ID or ENTAB Entry Count for Segment (2 bytes)

Flag (1 byte) Bits 0 through 3 not used
Bit 4 = 0 -- next two bytes contain the highest ID of the segment
= 1 -- next two bytes contain the number of ENTAB entries for this segment
Bits 5,6,7 - The low-order three bits of the previously assigned address of the first control section of this segment

Cumulative Segment Length - in bytes, of control sections in this segment (including the ENTAB, if present) (3 bytes)

Appearance of table after segment addresses are determined



Segment Relocation Constant - for the segment that corresponds to this entry (3 bytes)

Path Length - in bytes, of this segment, including this segment and its ENTAB (3 bytes)

Figure 58. Segment Length Table (SEGLGTH)

Segment Table (SEGTAB)
Built by Intermediate Output Processor

TEST indicator	Address of data control block (DCB) used to load module *		
	Address of note list *		
Last segment number of region 1	Highest segment no. in storage-region 1	Last segment number of region 2	Highest segment no. in storage-region 2
Last segment number of region 3	Highest segment no. in storage-region 3	Last segment number of region 4	Highest segment no. in storage-region 4
Zero	(Not used in the Fixed-Task Supervisor) *		
(Not used in the Fixed-Task Supervisor) *			
Previous segment number for segment 1 *	Zero		Status indicator
Previous segment number for segment 2	Address of entry table entry (when caller chain exists) *		Status indicator
⋮	⋮		⋮
Previous segment number for segment N	Address of entry table entry (when caller chain exists) *		Status indicator

4 bytes

Legend:
TEST indicator -- specifies that this module is "under test" using TESTRAN. Bit 1 is initialized by program fetch.
Highest segment no. in storage -- is initially set to 00 except for region 1 which is initially set to 01 by linkage editor.
Status indicator -- indicates the status of this segment with the two last bits of the entry table address field as follows:

Bits	Meaning
00	segment is in virtual storage as a result of a branch to the segment.
10	segment is in virtual storage ; no caller chain exists.
01	segment is not in virtual storage, but is scheduled to be loaded.
11	segment is not in virtual storage.

Note: The status indicator for segment 1 is initially set to 10; all the rest are initially set to 11.

* Set to zero by linkage editor.

Figure 59. Segment Table (SEGTAB)

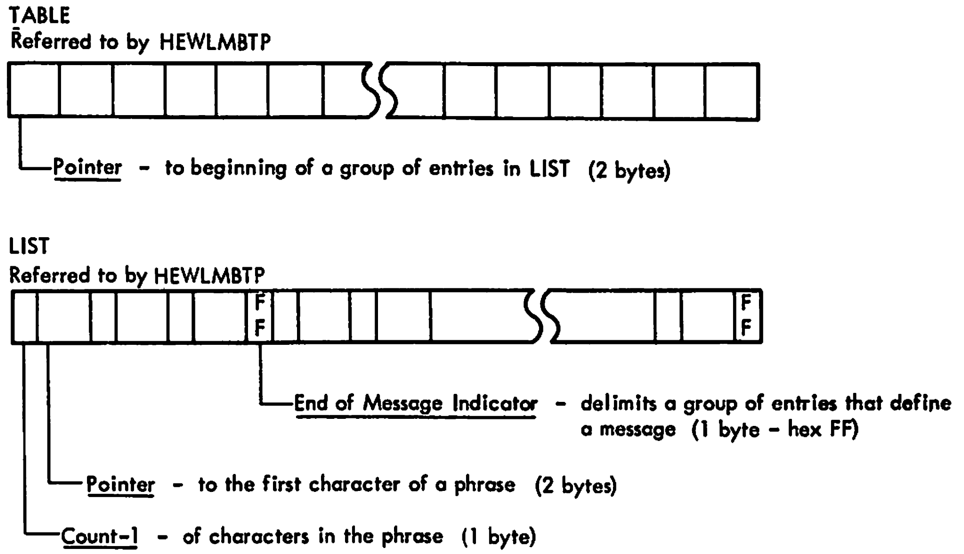
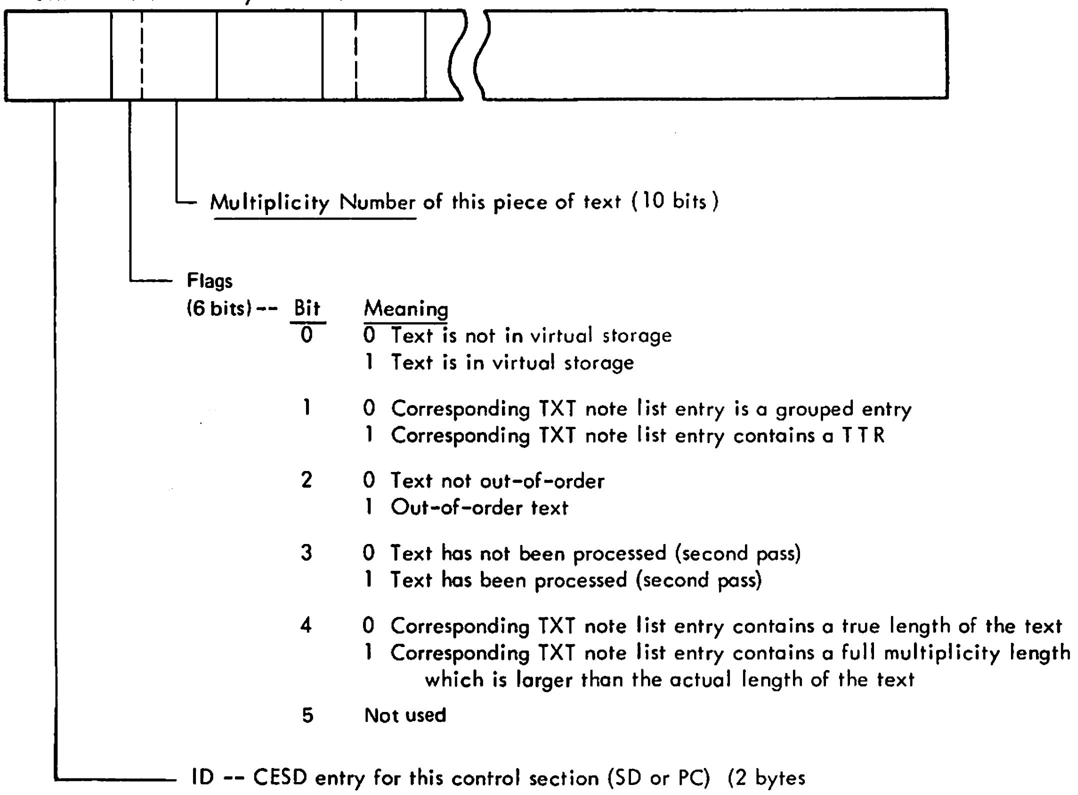


TABLE
Referred to by HEWLMBTP
Pointer - to beginning of a group of entries in LIST (2 bytes)
LIST
Referred to by HEWLMBTP
FF
FF
End of Message Indicator - delimits a group of entries that define a message (1 byte - hex FF)
Pointer - to the first character of a phrase (2 bytes)
Count-1 - of characters in the phrase (1 byte)

Figure 60. TABLE and LIST (Referred to by HEWLFBTP)

TEXT I/O TABLE

Built and referred to by First Pass Text Processor



Multiplicity Number of this piece of text (10 bits)

Flags (6 bits) --

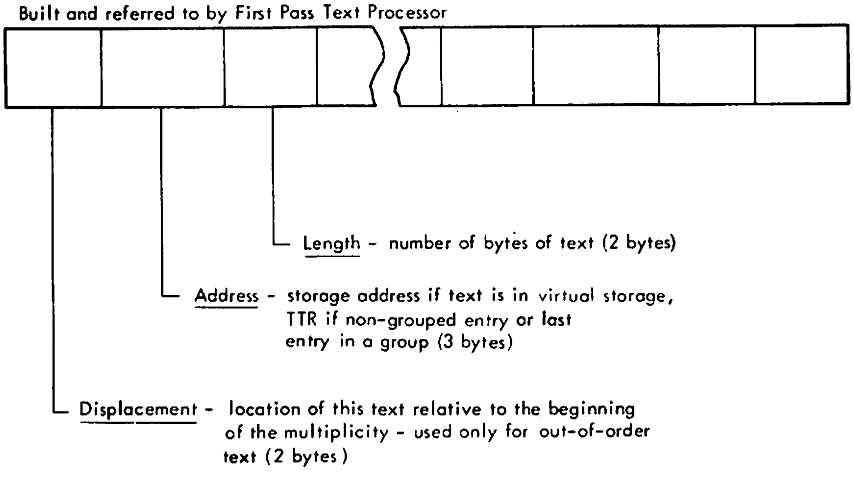
Bit	Meaning
0	0 Text is not in virtual storage 1 Text is in virtual storage
1	0 Corresponding TXT note list entry is a grouped entry 1 Corresponding TXT note list entry contains a TTR
2	0 Text not out-of-order 1 Out-of-order text
3	0 Text has not been processed (second pass) 1 Text has been processed (second pass)
4	0 Corresponding TXT note list entry contains a true length of the text 1 Corresponding TXT note list entry contains a full multiplicity length which is larger than the actual length of the text
5	Not used

ID -- CESD entry for this control section (SD or PC) (2 bytes

Figure 61. Text I/O Table

TEXT NOTE LIST

Built and referred to by First Pass Text Processor



Length - number of bytes of text (2 bytes)

Address - storage address if text is in virtual storage, TTR if non-grouped entry or last entry in a group (3 bytes)

Displacement - location of this text relative to the beginning of the multiplicity - used only for out-of-order text (2 bytes)

Figure 62. Text Note List

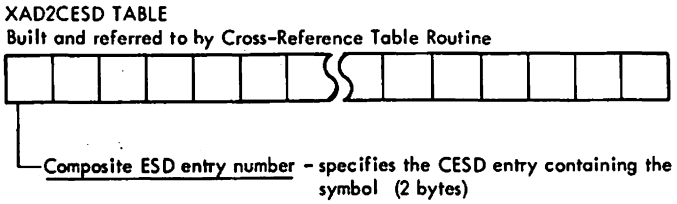

XAD2CESD TABLE
Built and referred to by Cross-Reference Table Routine
Composite ESD entry number - specifies the CESD entry containing the symbol (2 bytes)

Figure 63. XAD2CESD Table (Built and Referred to by Cross-Reference Table Routine)

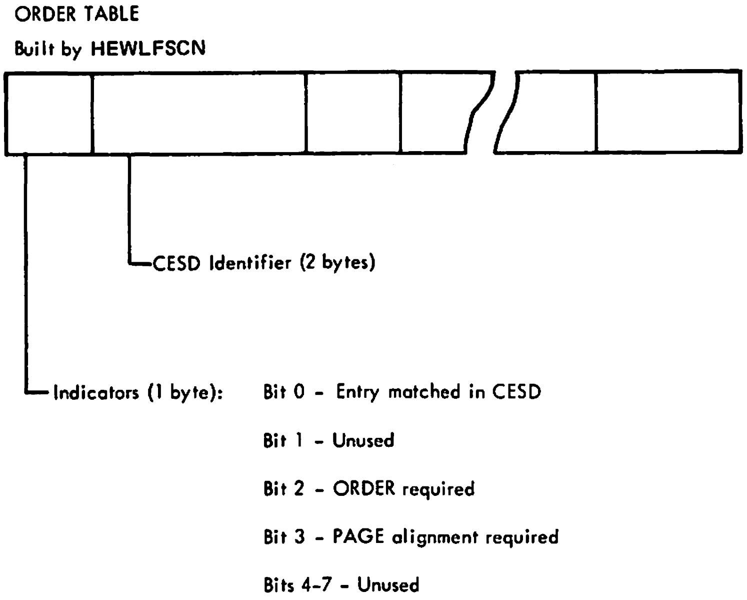


Figure 64. ORDER Table (Built by HEWLFSCN)

This section contains information that may be useful in diagnosing difficulties with the linkage editor program. Included are: register contents at major entry points (Figure 65), charts describing buffer (see Figure 66 on page 187) and table allocation (Figure 67 on page 188), and an error message—module cross-reference table (Figure 68 on page 189).

Module Entry Point	Reg	Contents
HEWLCIDR	1	Pointer to parameter list
	2	Address of all-purpose table
	13	Address of save area
	14	Return address
	15	Entry point address
HEWLFADA	2	Address of all-purpose table
HEWLFBTP	2	Address of all-purpose table
	14	Return address
	15	Entry point address
HEWLFEND	2	Address of all-purpose table
	4	Length of any no-length control section
	5	ID of the assembled address of the module entry point
	13	Address of save area
	14	Return address
	15	Entry point address
HEWLFENS	2	Address of all-purpose table
	13	Save are address
	14	Return address
HEWLFENT	2	Address of all-purpose table
	13	Save area address
	14	Return address

Figure 65 (Part 1 of 5). General Register Contents at Major Entry Points

Module Entry Point	Reg	Contents
HEWLFESD	2	Address of all-purpose table
	4	Byte count of ESD information
	5	ID of first ESD item to be processed
	6	Address of first ESD item to be processed
	13	Save area address
	14	Return address
	15	Entry point address
HEWLFFNL	2	Address of all purpose table
HEWLFIDR	1	Pointer to parameter list
	2	Address of all-purpose table
	13	Address of save area
	14	Return address
	15	Entry address
HEWLFINC	2	Address of all-purpose table
	12	Return address
	15	Entry point address
HEWLFINP	2	Address of all-purpose table
	15	Entry point address
HEWLFINT	1	Address of parameter list
	13	Save area address
	14	Return address
	15	Entry point address
HEWLFMAP	2	Address of all-purpose table
	14	Return address
	15	Address of entry point
HEWLFOPT	1	Address of parameter list
	2	Address of all-purpose table
	14	Return address
	15	Entry point address
HEWLFOUT	2	Address of all-purpose table

Figure 65 (Part 2 of 5). General Register Contents at Major Entry Points

Module Entry Point	Reg	Contents
HEWLFRAT	2	Address of all-purpose table
	4	Byte count of RLD input
	6	Storage address of RLD input
	14	Return address
HEWLFRCG	2	Address of all-purpose table
	6	Address of ESD item being processed
	10	Address of beginning of replace/change chain
	13	Entry point address
HEWLFREL	1	HESD address of either first ENTAB entry (if overlay) or last HESD entry + 8
	2	Address of all-purpose table
	14	Return address
	15	Entry point address
HEWLFROU	1	Address of parameter list
	13	Address of save area
	14	Return address
	15	Entry point address
HEWLEPNT	2	Address of all-purpose table
	14	Return address
	15	Entry point address
HEWLFLOG	0	Error Code
	1	Address of first symbol (optional)
	2	Address of all-purpose table
	13	Address of second symbol (optional)
	14	Return address
	15	Entry point address
HEWLFALK	2	Address of all-purpose table
	14	Return address
	15	Entry point address
HEWLFSCD	1	Address of first ENTAB entry in HESD, if overlay; otherwise, address of last HESD entry + 8
	2	Address of all-purpose table

Figure 65 (Part 3 of 5). General Register Contents at Major Entry Points

Module Entry Point	Reg	Contents
GETIDMUL	0	Indicator: 0 - prime read; = lookahead
	1	Text control block address
	14	Return address
HEWLFSCN	1	Address of column 1 of input record
	2	Address of all-purpose table
	15	Entry point address
HEWLFSIO	2	Address of all-purpose table
CHECKRD	10	Base register of HEWLFSIO
	12	Address of HEWLFSCD
	14	Return address
	15	Address of HEWLFRLD
CHECKWRT	10	Base register of HEWLFSIO
	12	Address of HEWLFSCD
	14	Return address
	15	Address of HEWLFRLD
WRTCRRLD	1	Address of control block for buffer to be written
	10	Base register of HEWLFSIO
	12	Address of HEWLFSCD
	14	Return address
	15	Address of HEWLFRLD
HEWLFSYM	2	Address of all-purpose table
	13	Save area address
	14	Return address
	15	Entry point address
HEWLFTXT	2	Address of all-purpose table
	3	Assembled address of first byte of text
	4	Byte count of TXT input
	5	ID of current text record
	7	Base register of HEWLFTXT
	12	Base register of HEWLFRAT
	14	Return address

Figure 65 (Part 4 of 5). General Register Contents at Major Entry Points

Module Entry Point	Reg	Contents
HEWLCAUT	2	Address of all-purpose table
	12	Return address
	15	Entry point address
RELOCATE	2	Address of all-purpose table
	3	Address of text control block for current text
	14	Return address
	15	Address of HEWLFREL

Figure 65 (Part 5 of 5). General Register Contents at Major Entry Points

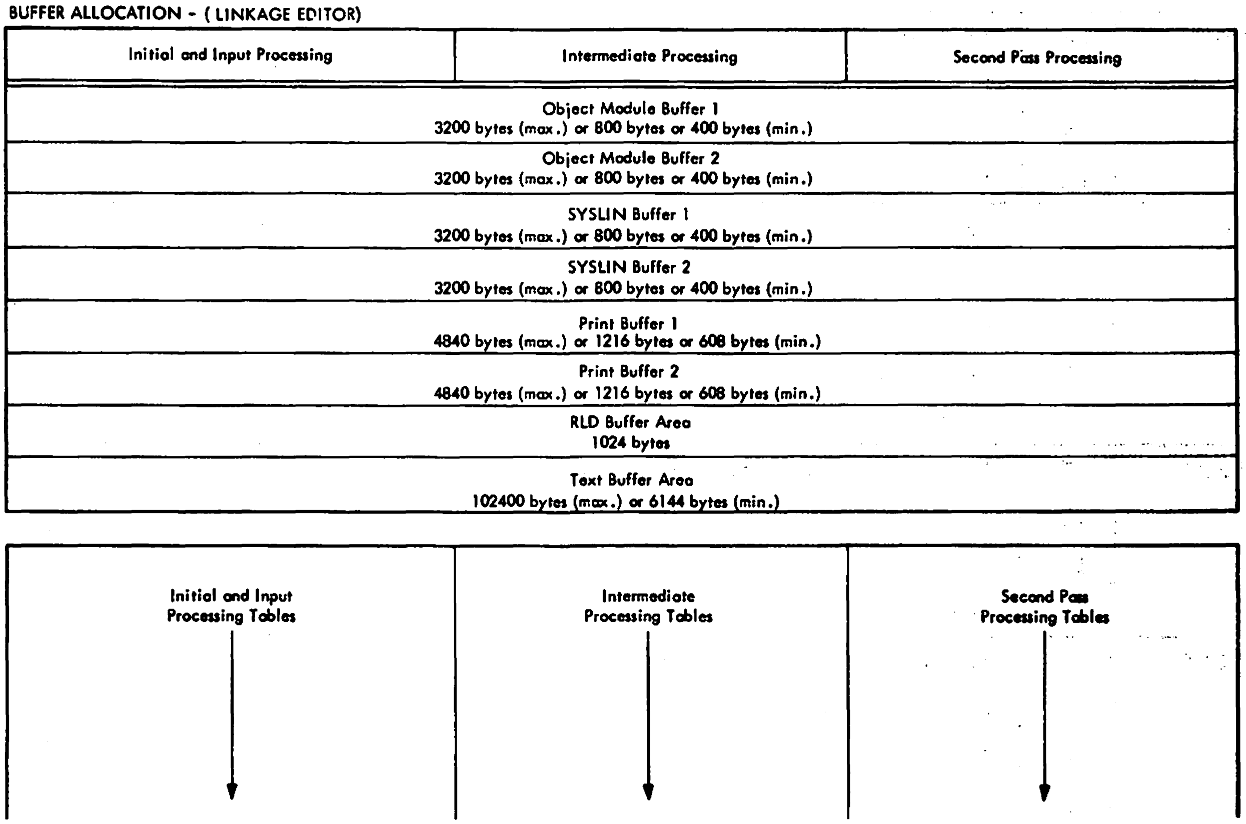

BUFFER ALLOCATION - (LINKAGE EDITOR)
Initial and Input Processing
Intermediate Processing
Second Pass Processing
Object Module Buffer 1
3200 bytes (max.) or 800 bytes or 400 bytes (min.)
Object Module Buffer 2
3200 bytes (max.) or 800 bytes or 400 bytes (min.)
SYSLIN Buffer 1
3200 bytes (max.) or 800 bytes or 400 bytes (min.)
SYSLIN Buffer 2
3200 bytes (max.) or 800 bytes or 400 bytes (min.)
Print Buffer 1
4840 bytes (max.) or 1216 bytes or 608 bytes (min.)
Print Buffer 2
4840 bytes (max.) or 1216 bytes or 608 bytes (min.)
RLD Buffer Area
1024 bytes
Text Buffer Area
102400 bytes (max.) or 6144 bytes (min.)
Initial and Input
Processing Tables
Intermediate
Processing Tables
Second Pass
Processing Tables

Figure 66. Buffer Allocation

					Present In					Size (in bytes)	
Table Name	Allocated for Overlay Link-Edit Only	Order of Allocation	Bytes/ Entry	Weight	Inp. Proc.	Int. Proc.	2nd Pass	Prefix	Align.	Min.	Max.
Alias Table	No	7	1	0	No	Yes	Yes	No	Dblwd	160	160
Calls List	Yes	15	1	96	Yes	Yes	No	No	Dblwd	1536	[3]
Composite ESD	No	8	16	288	Yes	No	No	Yes	Dblwd	4608	[3]
Delink Table	No	5	5	24	Yes	Yes	Yes	Yes	Dblwd	384	[3]
Entry List	Yes	16	6	96	No	No	Yes	No	Dblwd	1536	[3]
Half ESD	No	10	8	144	No	Yes	Yes	No	Dblwd	2304	[3]
Half ESD Prefix	No	9	1	0	No	Yes	Yes	No	Dblwd	8	8
HIERARCHY[1]	No	13	1	18	Yes	Yes	No	Yes	Dblwd	288	[3]
IDRTRTAB[2]	No	17, 20	1	12, 45	Yes	Yes	No	No	Dblwd	192, 208	[3,3]
IDRUDTAB	No	18	1	88	Yes	Yes	No	No	Dblwd	960	[3]
IDRZPTAB	No	19	1	44	Yes	Yes	No	No	Dblwd	455	[3]
Order	No	12	1	28	Yes	Yes	No	No	Dblwd	456	[3]
First Pass RLD Buffer	No	6	1	0	Yes	No	No	No	Dblwd	256	256
Second Pass RLD Buffer	No	11	1	0	No	No	Yes	No	Dblwd	768	768
Relocation Constant Table/ Renumbering Table	No	4	4	72	No Yes	No	No	Yes	Dblwd	1152	[3]
RLD Note List II	No	3	9	58	No	Yes	Yes	No	Dblwd	928	[3]
SEGTA1	Yes	14	1	0	Yes	Yes	Yes	Yes	Dblwd	256	256
Text I/O Table	No	1	1	144	Yes	Yes	Yes	No	Dblwd	2304	[3]
Text Note List II	No	2	7	252	No	Yes	Yes	No	Dblwd	4032	[3]

[1] Not used in virtual systems.

[2] Table allocated in two parts.

[3] Maximum is determined by storage availability.

Figure 67. Table Allocation

Figure 68 contains a list of error messages and the routines and CSECTs in which they originate. Each message contains a severity code in the last position of the message number. These severity codes are defined as follows:

0 Indicates a condition that will not cause an error during execution of the link-edited program.

1 Indicates a condition that may cause an error during execution of the link-edited program.

2 Indicates an error that can make execution of the link-edited program impossible.

3 Indicates an error that will make execution of the link-edited program impossible.

4 Indicates an unrecoverable error. Such an error causes termination of linkage editor processing.

Error Message Number	**Error Message Text**	**Issuer**	
		Routine	**CSECT**
IEW0000	Control statement	HEWLFSCN	HEWLFSCN
IEW0012	ERROR—Input contains invalid 2-byte relocatable address constant; constant has not been relocated.	HEWLFREL	HEWLFREL
IEW0022	ERROR—Input contains invalid V-type address constant; constant has not been relocated.	HEWLFREL	HEWLFREL
IEW0033	ERROR—Invalid entry point from END card; no entry point assigned.	HEWLFENT	HEWLFENT
IEW0043	ERROR—Input contains invalid external symbol ID.	HEWLFENT	HEWLFENT
IEW0053	ERROR—Entry statement symbol printed is invalid (not an external name); no entry point assigned.	HEWLFENT	HEWLFENT
IEW0063	ERROR—END card symbol printed is invalid (not an external name); no entry point assigned.	HEWLFENT	HEWLFENT
IEW0073	ERROR—Entry statement symbol printed is not in root segment of overlay structure; no entry point assigned.	HEWLFENT	HEWLFENT
IEW0083	ERROR—END card symbol printed is not in root segment of overlay structure; no entry point assigned.	HEWLFENT	HEWLFENT
IEW0093	ERROR—END card entry point address printed is not in root segment of overlay structure; no entry point assigned.	HEWLFENT	HEWLFENT
IEW0102	ERROR—Invalid entry point on END card; entry point ignored.	HEWLFEND	HEWLFEND
IEW0113	ERROR—Output module contains no control sections in root segment of overlay structure; no entry point assigned.	HEWLFENT	HEWLFENT
IEW0123	ERROR—No ESD entries; execution impossible.	HEWLFINP HEWLFADA	HEWLFINP HEWLFADA

Figure 68 (Part 1 of 5). Error Message/Issuer Cross-Reference Table

Error Message Number	Error Message Text	Issuer	
		Routine	CSECT
IEW0132	ERROR—Symbol printed is an unresolved external reference.	HEWLFADA	HEWLFADA
IEW0143	ERROR—No text.	HEWLFOUT HEWLFINP	HEWLFOUT HEWLFINP
IEW0152	ERROR—Invalid overlay structure; no calls or branches made from root segment.	HEWLFENS	HEWLFENS
IEW0161	Warning—Exclusive call from segment number printed to symbol printed - XCAL was specified.	HEWLFENS	HEWLFENS
IEW0172	ERROR—Exclusive call from segment number printed to symbol printed.	HEWLFENS	HEWLFENS
IEW0182	ERROR—Invalid exclusive call from segment number printed to symbol printed.	HEWLFENS	HEWLFENS
IEW0191	Warning—Main storage requirements for output load module have exceeded 512K bytes.	HEWLFADA	HEWLFADA
IEW0201	Warning—Overlay structure contains only one segment - overlay option cancelled.	HEWLFADA	HEWLFADA
IEW0212	ERROR—Expected continuation card not found.	HEWLFINP	HEWLFINP
IEW0222	ERROR—Card printed contains invalid input from object module.	HEWLFESD HEWIFINP HEWLFRAT	HEWLFESD HEWLFINP HEWLFRAT
IEW0234	ERROR—Input from load module is invalid.	HEWLFRAT HEWLFINP HEWLFESD	HEWLFRAT HEWLFINP HEWLFESD
IEW0241	Warning—External symbol printed is doubly defined - ESD type definitions conflict.	HEWLFESD	HEWLFESD
IEW0254	ERROR—Table overflow - too many external symbols in ESD.	HEWLFESD HEWLFADA HEWLFSCN	HEWLFESD HEWLFADA HEWLFSCN
IEW0264	ERROR—Table overflow - input load module contains too many external symbols in ESD.	HEWLFESD	HEWLFESD
IEW0272	ERROR—Load module from library specified unacceptable to level F.	HEWLFINC	HEWLFINC
IEW0284	ERROR—DDname printed cannot be opened.	HEWLFINT HEWLFRAT	HEWLFINT HEWLFRAT
IEW0294	ERROR—DDname printed had synchronous error.	HEWLFROU HEWLFFNL	HEWLFROU HEWLFFNL
IEW0302	ERROR—Invalid statement - scan terminated.	HEWLFSCN	HEWLFSCN
IEW0314	ERROR—Maximum number of regions (4) exceeded.	HEWLFSCN	HEWLFSCN
IEW0324	ERROR—Maximum number of segments exceeded.	HEWLFSCN	HEWLFSCN
IEW0332	ERROR—Maximum number of aliases (16) exceeded, excess ignored.	HEWLFSCN	HEWLFSCN
IEW0342	ERROR—Library specified does not contain module.	HEWLFINC	HEWLFINC

Figure 68 (Part 2 of 5). Error Message/Issuer Cross-Reference Table

Error Message Number	Error Message Text	Issuer	
		Routine	CSECT
IEW0354	ERROR—Table overflow - too many calls between control sections.	HEWLFRAT	HEWLFRAT
IEW0364	ERROR—Table overflow - input text exceeded maximum, or too many changes of origin in input.	HEWLFRAT HEWLFOUT HEWLFADA	HEWLFTXT HEWLFOUT HEWLFADA
IEW0374	ERROR—Table overflow - input contains too many relocatable address constants, or too many control sections containing such constants.	HEWLFRAT	HEWLFRAT
IEW0382	ERROR—Text record ID is invalid; card ignored.	HEWLFRAT HEWLFADA	HEWLFTXT HEWLFADA
IEW0394	ERROR—Member not stored in library - permanent device error.	HEWLFFNL	HEWLFFNL
IEW0404	ERROR—Member not stored in library - no space left in directory.	HEWLFFNL	HEWLFFNL
IEW0412	ERROR—Alias not stored in library - no space left in directory.	HEWLFFNL	HEWLFFNL
IEW0421	Warning—Member not stored in library - identical name in directory; will try to store under 'TEMPNAME.'	HEWLFFNL	HEWLFFNL
IEW0432	ERROR—Library name printed cannot be opened; DD card may be missing.	HEWLFINC	HEWLFINC
IEW0444	ERROR—Table overflow - too many downward calls.	HEWLFREL	HEWLFREL
IEW0454	ERROR—Table overflow - segment contains too many downward calls.	HEWLFADA	HEWLFADA
IEW0461	Warning—Symbol printed is an unresolved external reference; NCAL was specified, or the reference was marked for restricted no-call or never-call	HEWLFADA	HEWLFADA
IEW0472	ERROR—Invalid alias entry point in overlay structure.	HEWLFENT	HEWLFENT
IEW0484	ERROR—Table overflow - too many external symbols affected by relocation.	HEWLFINP	HEWLFINP
IEW0492	ERROR—Invalid name card found in library; card ignored.	HEWLFSCN	HEWLFSCN
IEW0502	ERROR—Alias not stored in library - permanent device error.	HEWLFFNL	HEWLFFNL
IEW0512	ERROR—INCLUDE statement syntax conflicts with record format of specified data set - DDname printed.	HEWLFINC	HEWLFINC
IEW0522	ERROR—Specified data set has unacceptable record format - DDname printed.	HEWLFINC	HEWLFINC
IEW0532	ERROR—Blocksize of library data set exceeded maximum - DDname printed.	HEWLFINC	HEWLFINC
IEW0543	ERROR—Identical name in directory.	HEWLFFNL	HEWLFFNL

Figure 68 (Part 3 of 5). Error Message/Issuer Cross-Reference Table

Error Message Number	Error Message Text	Issuer	
		Routine	CSECT
IEW0552	ERROR—Common printed exceeded size of control section with identical name.	HEWLFESD	HEWLFESD
IEW0564	ERROR—Invalid text origin, linkage editor processing terminated.	HEWLFSCD	HEWLFSCD
IEW0572	ERROR—Common printed and subroutine have identical name.	HEWLFESD	HEWLFESD
IEW0581	Warning—Invalid member name; will try to store under 'TEMPNAME.'	HEWLFFNL	HEWLFFNL
IEW0594	ERROR—Input data set blocksize is invalid.	HEWLFINP HEWLFINT	HEWLFINP HEWLFINT
IEW0602	ERROR—Input from object module is invalid - END card missing.	HEWLFINP	HEWLFINP
IEW0614	ERROR—Length not specified for external symbol printed.	HEWLFRAT	HEWLFTXT
IEW0622	ERROR—Address constant references null unnamed control section.	HEWLFRAT	HEWLFRAT
IEW0630	ERROR—DDname printed had synchronous error - XREF aborted.	HEWLFROU	HEWLFROU
IEW0642	ERROR—Symbol printed appeared on control statement but was not matched.	HEWLFADA	HEWLFADA
IEW0652	ERROR—Conflict in order specified for symbol printed.	HEWLFSCN	HEWLFSCN
IEW0664	ERROR—Size value specified not large enough for table requirements - linkage editor processing terminated.	HEWLFROU	HEWLFROU
IEW0670	The specified identify data has been added to the IDR for the control section name printed.	HEWLFIDR	HEWLFIDR
IEW0682	ERROR—Control section name on an IDENTIFY control statement is incorrect, or the statement is misplaced - IDENTIFY data ignored.	HEWLFIDR	HEWLFIDR
IEW0694	ERROR—Table overflow - SIZE value specified not large enough for CSECT IDR input - linkage editor processing terminated.	HEWLFIDR	HEWLFIDR
IEW0704	Unrecoverable error detected in CSECT IDR input - linkage editor processing terminated.	HEWLFIDR	HEWLFIDR
IEW0714	ERROR—Member not stored in library - STOW workspace unavailable.	HEWLFFNL	HEWLFFNL
IEW0722	ERROR—Invalid alias name.	HEWLFSCN	HEWLFSCN
IEW0731	Warning—Alias matches member name - alias ignored.	HEWLFFNL	HEWLFFNL
IEW0740	The indicated action was taken for an EXPAND request.	HEWLFSCN	HEWLFSCN
IEW0751	Warning—Invalid AMODE/RMODE combination found in MODE control statement - ignored.	HEWLFFNL	HEWLFFNL

Figure 68 (Part 4 of 5). Error Message/Issuer Cross-Reference Table

Error Message Number	Error Message Text	Issuer	
		Routine	CSECT
IEW0761	Warning—Invalid AMODE/RMODE combination found in PARM field - ignored.	HEWLFFNL	HEWLFFNL
IEW0771	Warning—AMODE/RMODE data in MODE control statement incompatible with OVLY option - ignored.	HEWLFFNL	HEWLFFNL
IEW0781	Warning—AMODE/RMODE data in PARM field incompatible with OVLY option - ignored.	HEWLFFNL	HEWLFFNL
IEW0791	Warning—Invalid AMODE/RMODE combination in ESD data for the named CSECT - ignored.	HEWLFESD	HEWLFESD
IEW0801	Warning—Table overflow - too many external symbols - MAP-XREF aborted.	HEWLFFNL	HEWLFFNL
IEW0813	ERROR—Output module contains split relocatable address constant, size value 2 specified not large enough, constant has not been relocated.	HEWLFREL	HEWLFREL
IEW0984	ERROR—SYSPRINT blocksize exceeds maximum - linkage editor processing terminated.	HEWLFINT	HEWLFINT
IEW0994	ERROR—SYSPRINT DD card missing - linkage editor processing terminated.	HEWLFINT	HEWLFINT

Figure 68 (Part 5 of 5). Error Message/Issuer Cross-Reference Table

APPENDIX. CONVENTIONS/FORMATS

This section contains linkage editor input conventions and record formats (see Figure 69 on page 195 through Figure 82 on page 204).

INPUT CONVENTIONS

Input modules (object or load) to be processed in a single execution of the linkage editor must conform with a number of input conventions. Violations of the following are treated as errors by the linkage editor:

- All text records of a control section must follow the ESD record containing the SD or PC entry that describes the control section.
- The end of every input module must be marked by an end indication (END record in an object module; EOM flag in a load module).
- Each input module may contain only one no-length control section (a control section whose length field in its SD or PD entry in the ESD contains zeros). The length must be specified on the END record of any module that contains a no-length control section.
- After processing the first text record of a no-length control section, the linkage editor will not accept a text record of a different control section within the same input module.
- Any RLD item must be read after the ESD items to which it refers; if it refers to a label within a different control section, it must be read after the ESD item for that control section.
- The language translators must gather RLD items in groups of identical position pointers. No two RLD items having the same P pointer can be separated by an RLD item having a different P pointer.
- Each record of text[14] and each LD or LR entry in the ESD record must refer to an SD or PC entry in the ESD.
- The position pointer of every RLD item must point to an SD or PC entry in the ESD.
- No LD or LR may have the same name as an SD or CM.
- All SYM records must be placed at the beginning of an input module. The ESD for an input module containing test translator statements must follow the SYM records and precede TXT records.
- The linkage editor accepts TXT records that are out of order within a control section, even though linkage editor processing may be affected. TXT records are accepted even though they may overwrite previous text in the same control section. The linkage editor does not eliminate any RLD items that correspond to overwritten text.

[14] A common (CM) control section cannot contain text or external references.

- During a single execution of the linkage editor, if two or more control sections having the same name are read in, only the first control section is accepted; the subsequent control sections are deleted.
- The linkage editor interprets common (CM) entries in the ESD (blank or with the same name) as references to a single control section whose length is the maximum length specified in the CM items of that name (or blank). No text may be contained in a common control section.
- Within an input module, the linkage editor does not accept an SD or PC entry after the first RLD item is read.

To avoid unnecessary scanning and input/output operations, input modules should conform with the following conventions. Although violations of these rules are not treated as errors, avoiding them will improve the efficiency of linkage editor processing.

- Within an input module, no LD or SD may have the same name as an ER.
- Within an input module, no two ERs may have the same name.
- Within an input module, TXT records may be in the order of the addresses assigned by the language translator. (If TXT records are not in address sequence, each reorigin operation may require additional linkage editor processing time.)
- SYSUT1 record size should be at least as large as SYSLMOD.

RECORD FORMATS

Figure 69 through Figure 82 on page 204 are the card image and load module record formats for the linkage editor.

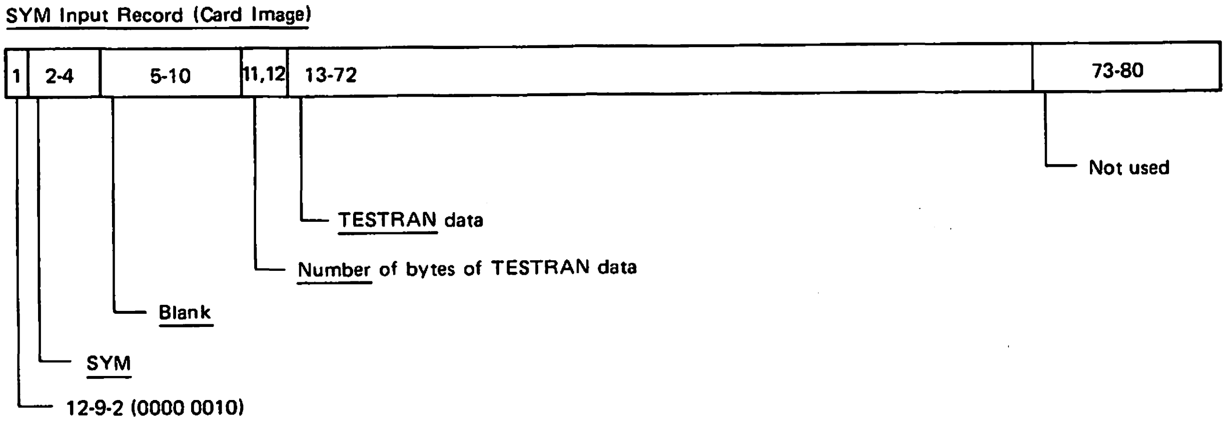
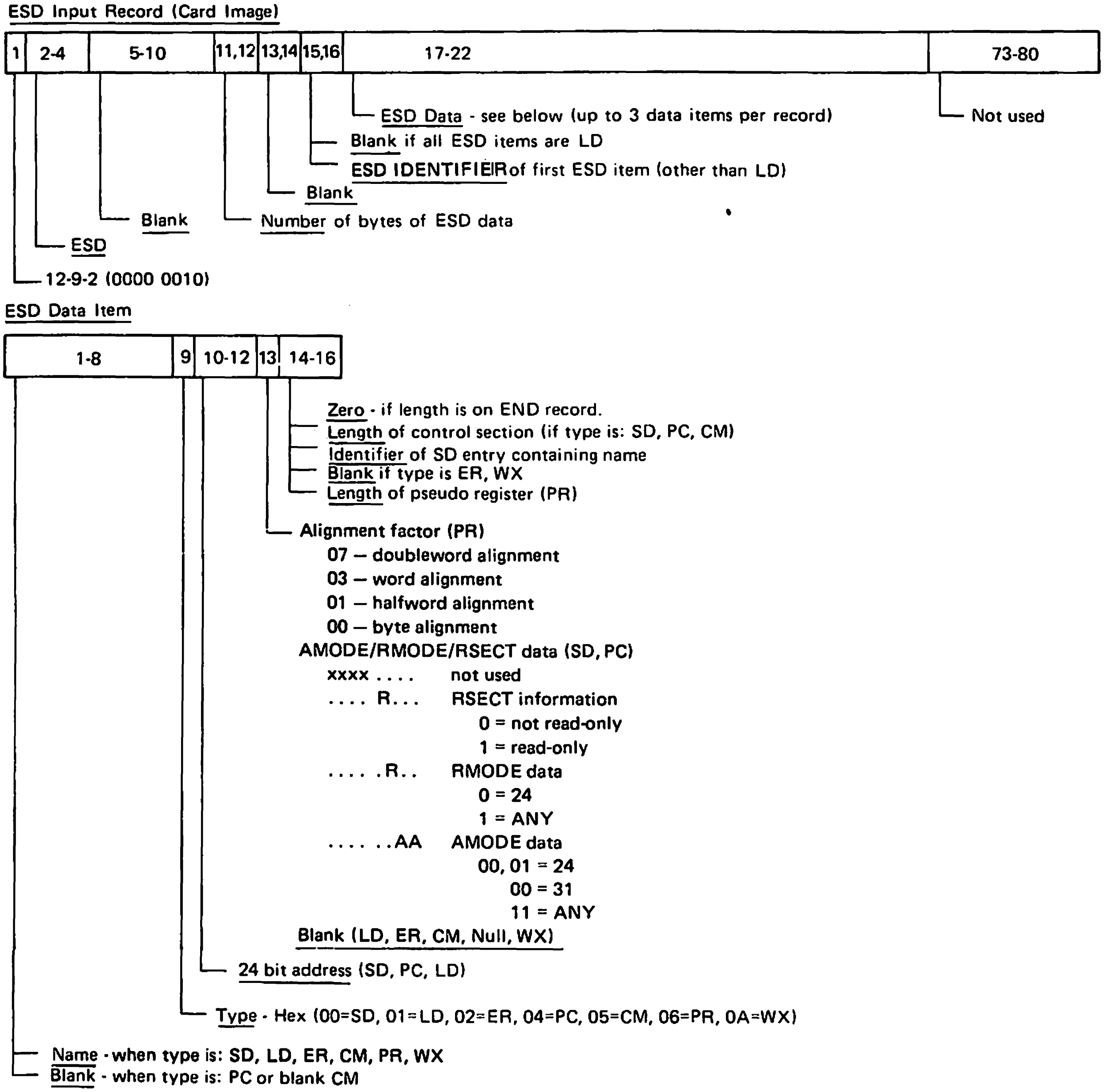


Figure 69. SYM Input Record (Card Image)



ESD Input Record (Card Image)

1	2-4	5-10	11,12	13,14	15,16	17-22	73-80

- 1 — 12-9-2 (0000 0010)
- 2-4 — ESD
- 5-10 — Blank
- 11,12 — Number of bytes of ESD data
- 13,14 — Blank
- 15,16 — ESD IDENTIFIER of first ESD item (other than LD)
- 15,16 — Blank if all ESD items are LD
- 17-22 — ESD Data - see below (up to 3 data items per record)
- 73-80 — Not used

ESD Data Item

1-8	9	10-12	13	14-16

1-8:
- Name - when type is: SD, LD, ER, CM, PR, WX
- Blank - when type is: PC or blank CM

9: Type - Hex (00=SD, 01=LD, 02=ER, 04=PC, 05=CM, 06=PR, 0A=WX)

10-12: 24 bit address (SD, PC, LD)

13:
Blank (LD, ER, CM, Null, WX)
Alignment factor (PR)
07 — doubleword alignment
03 — word alignment
01 — halfword alignment
00 — byte alignment
AMODE/RMODE/RSECT data (SD, PC)
xxxx . . . .   not used
. . . . R. . .   RSECT information
0 = not read-only
1 = read-only
. . . . . R. .   RMODE data
0 = 24
1 = ANY
. . . . . . AA   AMODE data
00, 01 = 24
00 = 31
11 = ANY

14-16:
- Zero - if length is on END record.
- Length of control section (if type is: SD, PC, CM)
- Identifier of SD entry containing name
- Blank if type is ER, WX
- Length of pseudo register (PR)

Figure 70. ESD Input Record (Card Image)

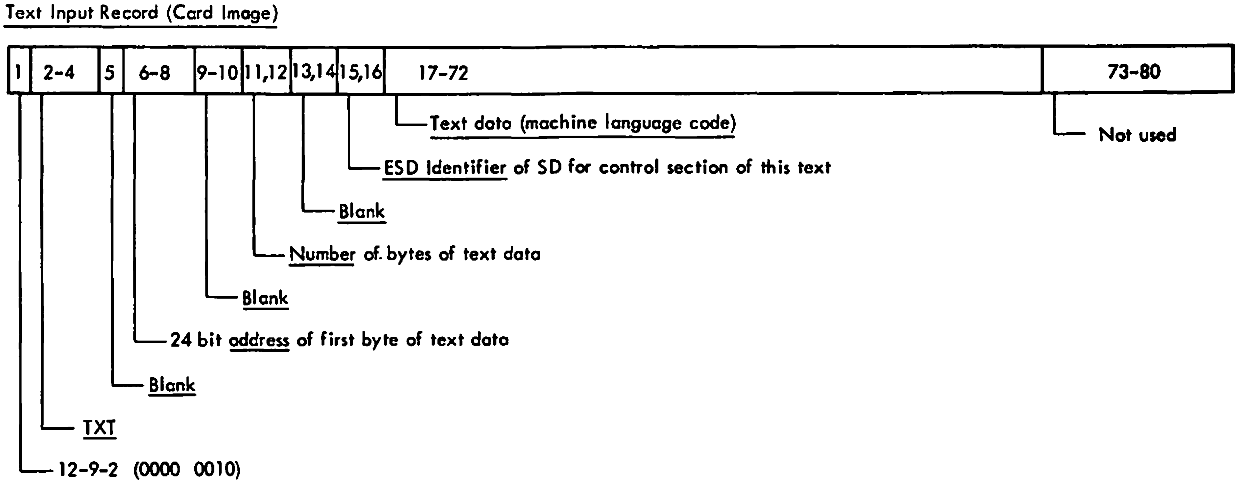

Text Input Record (Card Image)
1
2-4
5
6-8
9-10
11,12
13,14
15,16
17-72
73-80
Text data (machine language code)
Not used
ESD Identifier of SD for control section of this text
Blank
Number of. bytes of text data
Blank
24 bit address of first byte of text data
Blank
TXT
12-9-2 (0000 0010)

Figure 71. Text Input Record (Card Image)

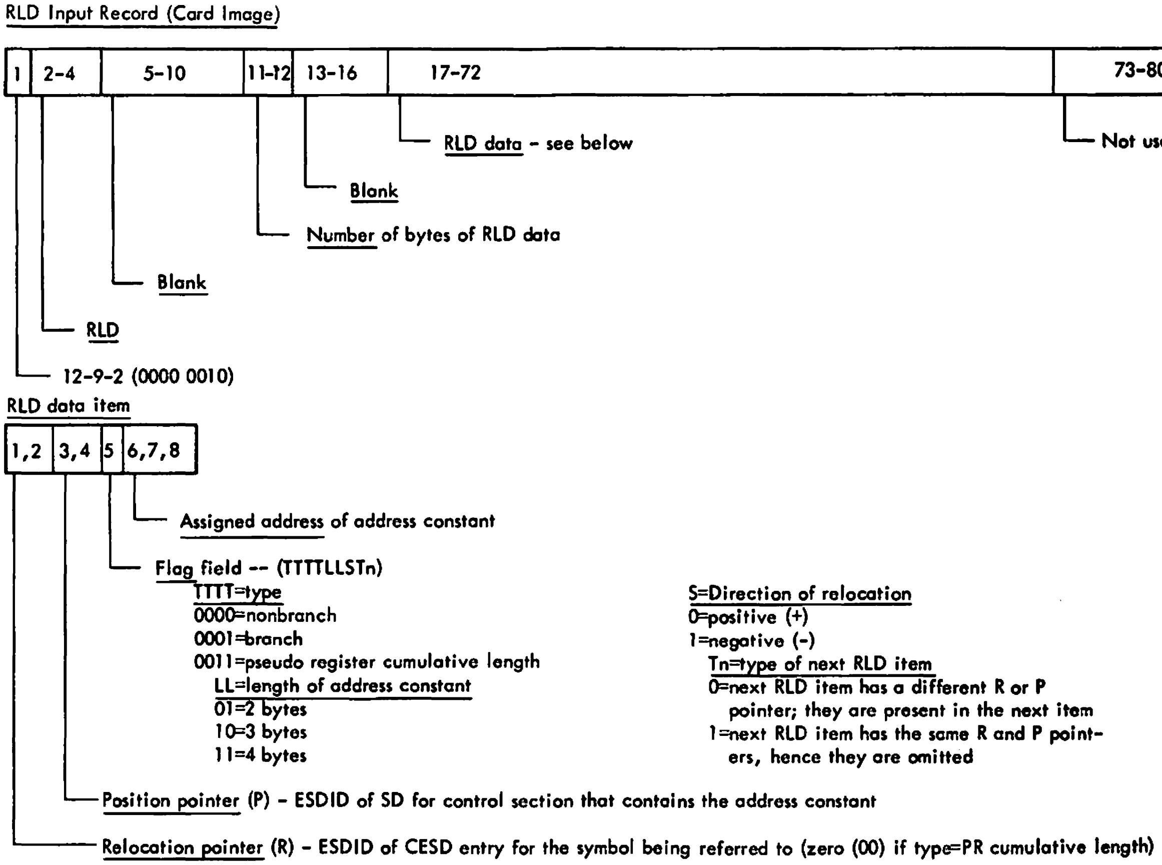
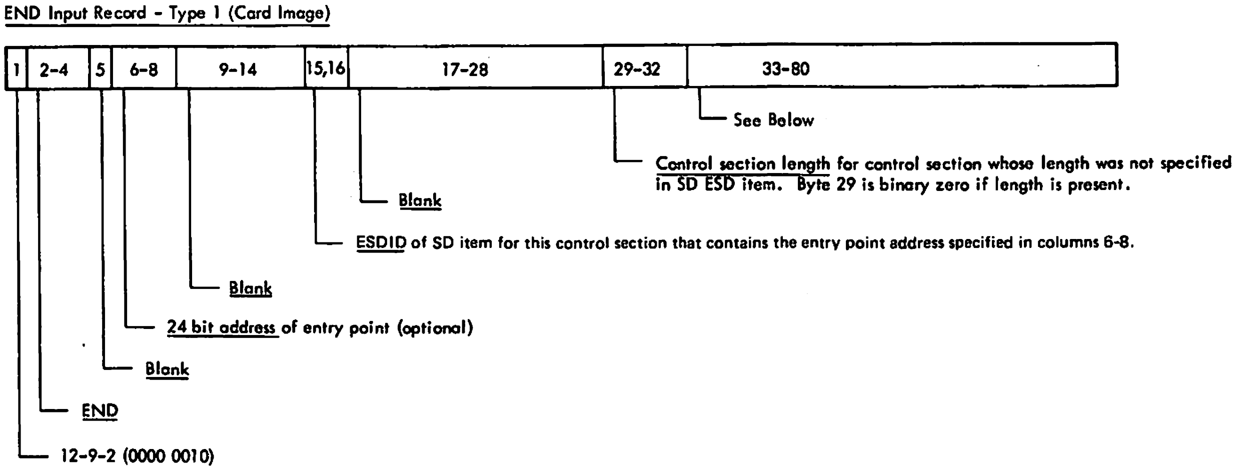

RLD Input Record (Card Image)
1
2-4
5-10
11-12
13-16
17-72
73-80
RLD data - see below
Not used
Blank
Number of bytes of RLD data
Blank
RLD
12-9-2 (0000 0010)
RLD data item
1,2
3,4
5
6,7,8
Assigned address of address constant
Flag field -- (TTTTLLSTn)
TTTT=type
0000=nonbranch
0001=branch
0011=pseudo register cumulative length
LL=length of address constant
01=2 bytes
10=3 bytes
11=4 bytes
S=Direction of relocation
0=positive (+)
1=negative (-)
Tn=type of next RLD item
0=next RLD item has a different R or P pointer; they are present in the next item
1=next RLD item has the same R and P pointers, hence they are omitted
Position pointer (P) - ESDID of SD for control section that contains the address constant
Relocation pointer (R) - ESDID of CESD entry for the symbol being referred to (zero (00) if type=PR cumulative length)

Figure 72. RLD Input Record (Card Image)

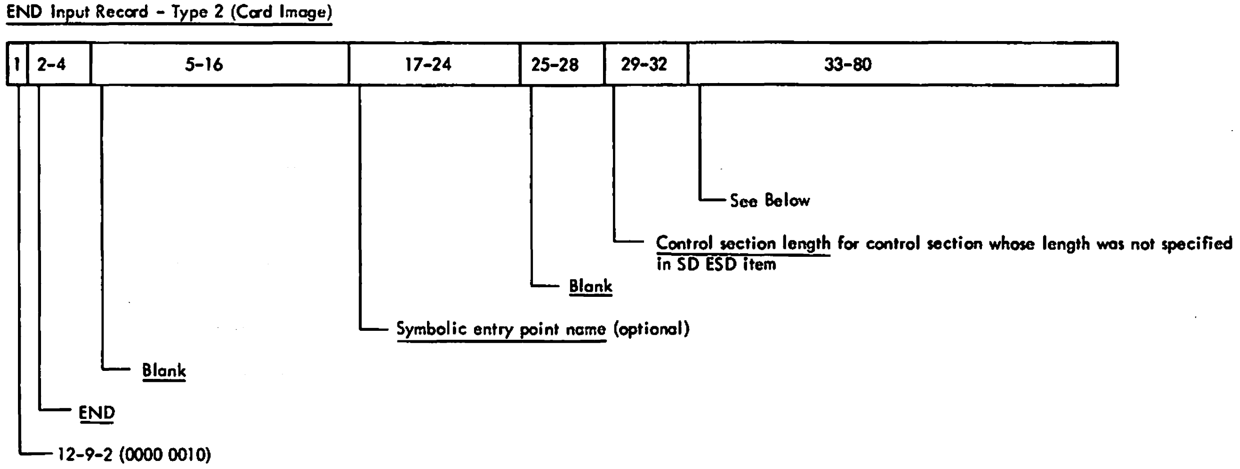


END Input Record - Type 1 (Card Image)

1	2-4	5	6-8	9-14	15,16	17-28	29-32	33-80

- 1: 12-9-2 (0000 0010)
- 2-4: END
- 5: Blank
- 6-8: 24 bit address of entry point (optional)
- 9-14: Blank
- 15,16: ESDID of SD item for this control section that contains the entry point address specified in columns 6-8.
- 17-28: Blank
- 29-32: Control section length for control section whose length was not specified in SD ESD item. Byte 29 is binary zero if length is present.
- 33-80: See Below

Figure 73. END Input Record—Type 1 (Card Image)

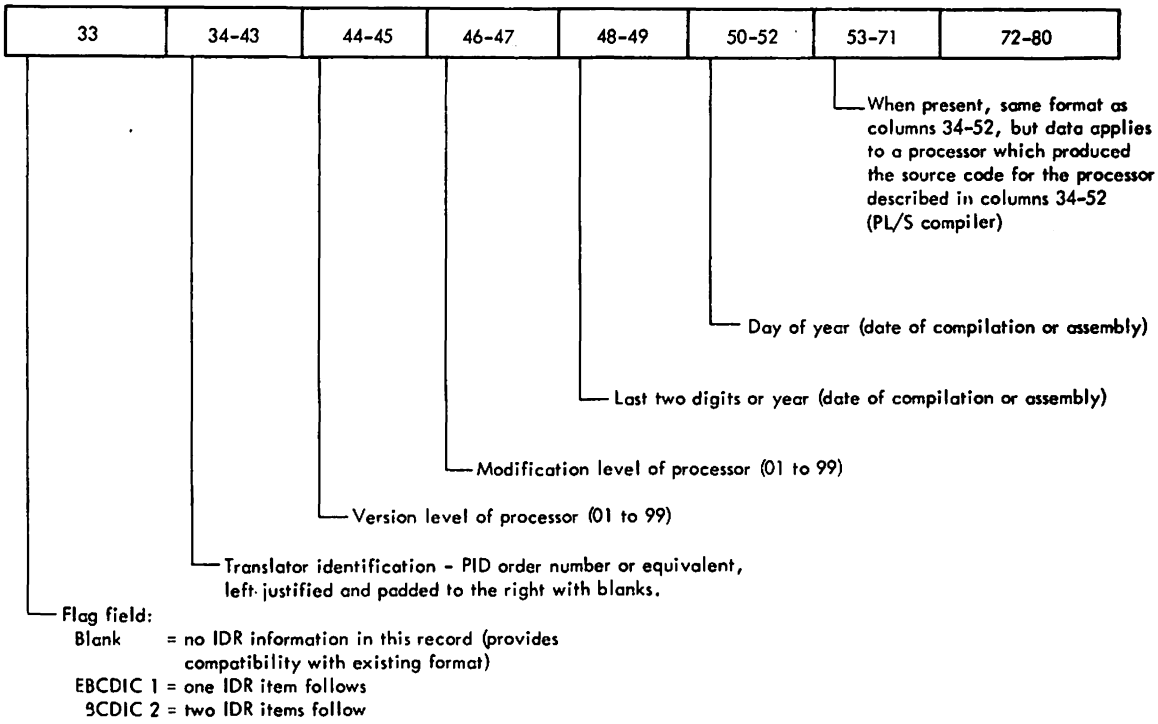


END Input Record - Type 2 (Card Image)

1	2-4	5-16	17-24	25-28	29-32	33-80

- 1: 12-9-2 (0000 0010)
- 2-4: END
- 5-16: Blank
- 17-24: Symbolic entry point name (optional)
- 25-28: Blank
- 29-32: Control section length for control section whose length was not specified in SD ESD item
- 33-80: See Below

Figure 74. END Input Record—Type 2 (Card Image)



When present, same format as columns 34-52, but data applies to a processor which produced the source code for the processor described in columns 34-52 (PL/S compiler)

Day of year (date of compilation or assembly)

Last two digits or year (date of compilation or assembly)

Modification level of processor (01 to 99)

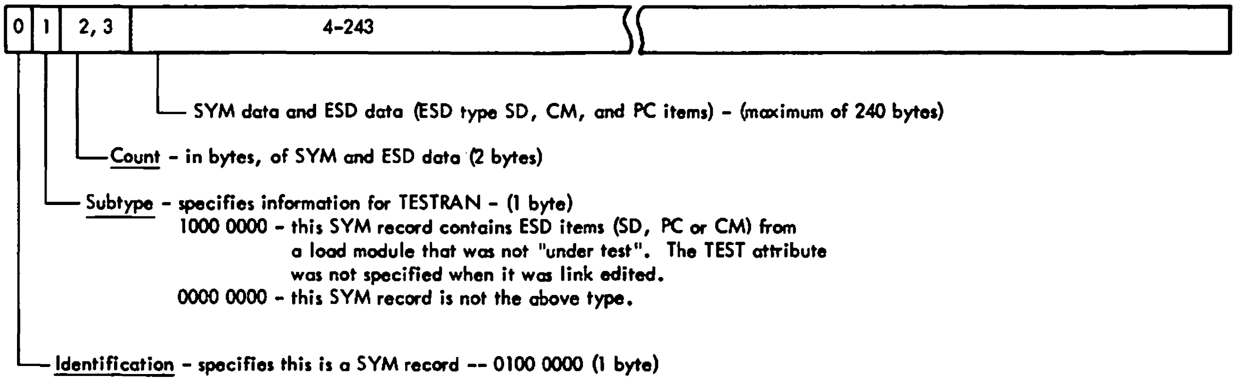
Version level of processor (01 to 99)

Translator identification - PID order number or equivalent, left justified and padded to the right with blanks.

Flag field:
Blank = no IDR information in this record (provides compatibility with existing format)
EBCDIC 1 = one IDR item follows
3CDIC 2 = two IDR items follow

Figure 75. IDR Data in an Object Module End Record

SYM Record - (Load Module)



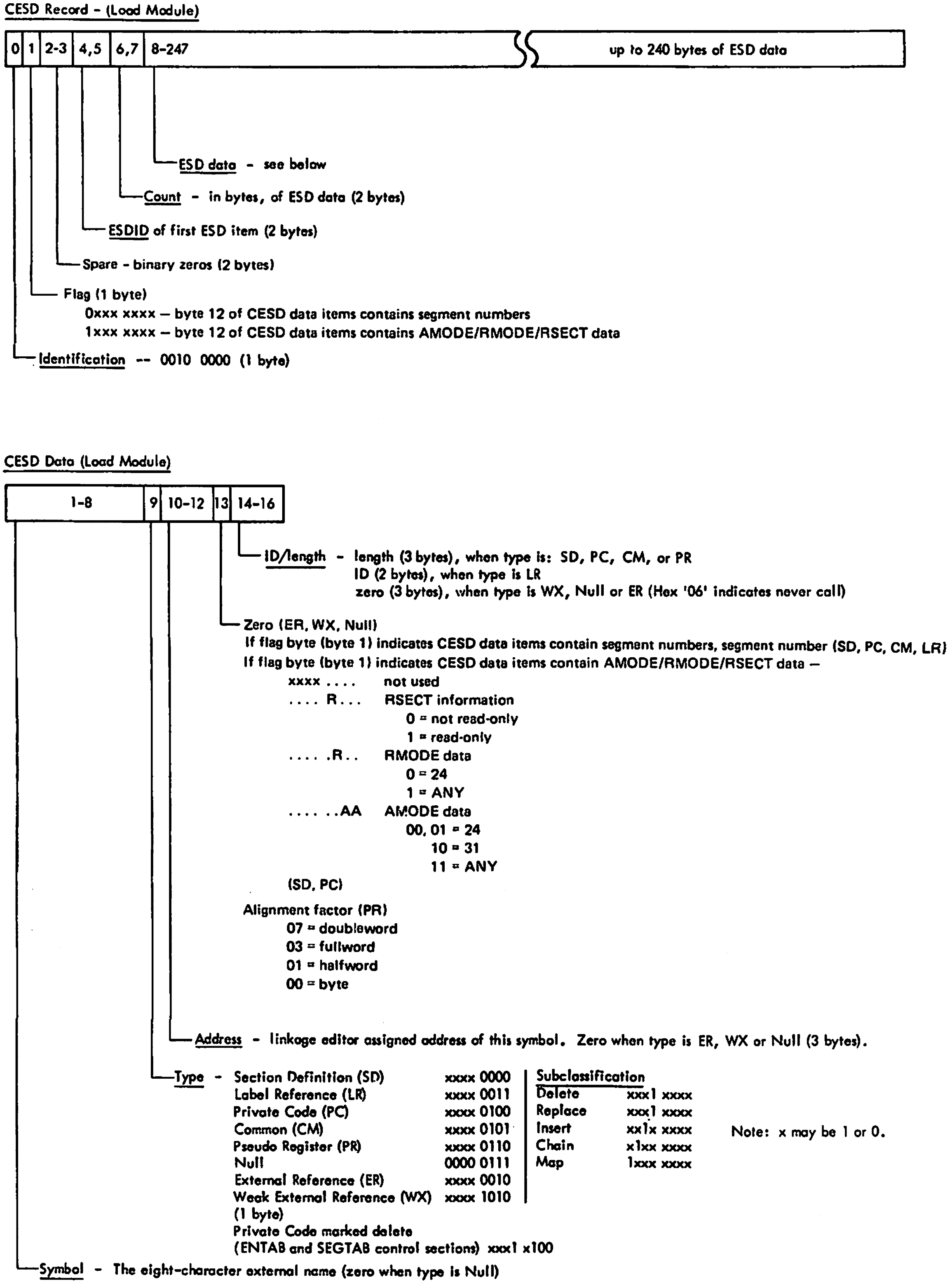
SYM data and ESD data (ESD type SD, CM, and PC items) - (maximum of 240 bytes)

Count - in bytes, of SYM and ESD data (2 bytes)

Subtype - specifies information for TESTRAN - (1 byte)
1000 0000 - this SYM record contains ESD items (SD, PC or CM) from a load module that was not "under test". The TEST attribute was not specified when it was link edited.
0000 0000 - this SYM record is not the above type.

Identification - specifies this is a SYM record -- 0100 0000 (1 byte)

Figure 76. SYM Record (Load Module)



CESD Record - (Load Module)

0	1	2-3	4,5	6,7	8-247	up to 240 bytes of ESD data

ESD data - see below

Count - in bytes, of ESD data (2 bytes)

ESDID of first ESD item (2 bytes)

Spare - binary zeros (2 bytes)

Flag (1 byte)
0xxx xxxx — byte 12 of CESD data items contains segment numbers
1xxx xxxx — byte 12 of CESD data items contains AMODE/RMODE/RSECT data

Identification -- 0010 0000 (1 byte)

CESD Data (Load Module)

1-8	9	10-12	13	14-16

ID/length - length (3 bytes), when type is: SD, PC, CM, or PR
ID (2 bytes), when type is LR
zero (3 bytes), when type is WX, Null or ER (Hex '06' indicates never call)

Zero (ER, WX, Null)
If flag byte (byte 1) indicates CESD data items contain segment numbers, segment number (SD, PC, CM, LR)
If flag byte (byte 1) indicates CESD data items contain AMODE/RMODE/RSECT data —

xxxx .... not used
.... R... RSECT information
0 = not read-only
1 = read-only
.... .R.. RMODE data
0 = 24
1 = ANY
.... ..AA AMODE data
00, 01 = 24
10 = 31
11 = ANY

(SD, PC)

Alignment factor (PR)
07 = doubleword
03 = fullword
01 = halfword
00 = byte

Address - linkage editor assigned address of this symbol. Zero when type is ER, WX or Null (3 bytes).

Type - (1 byte)

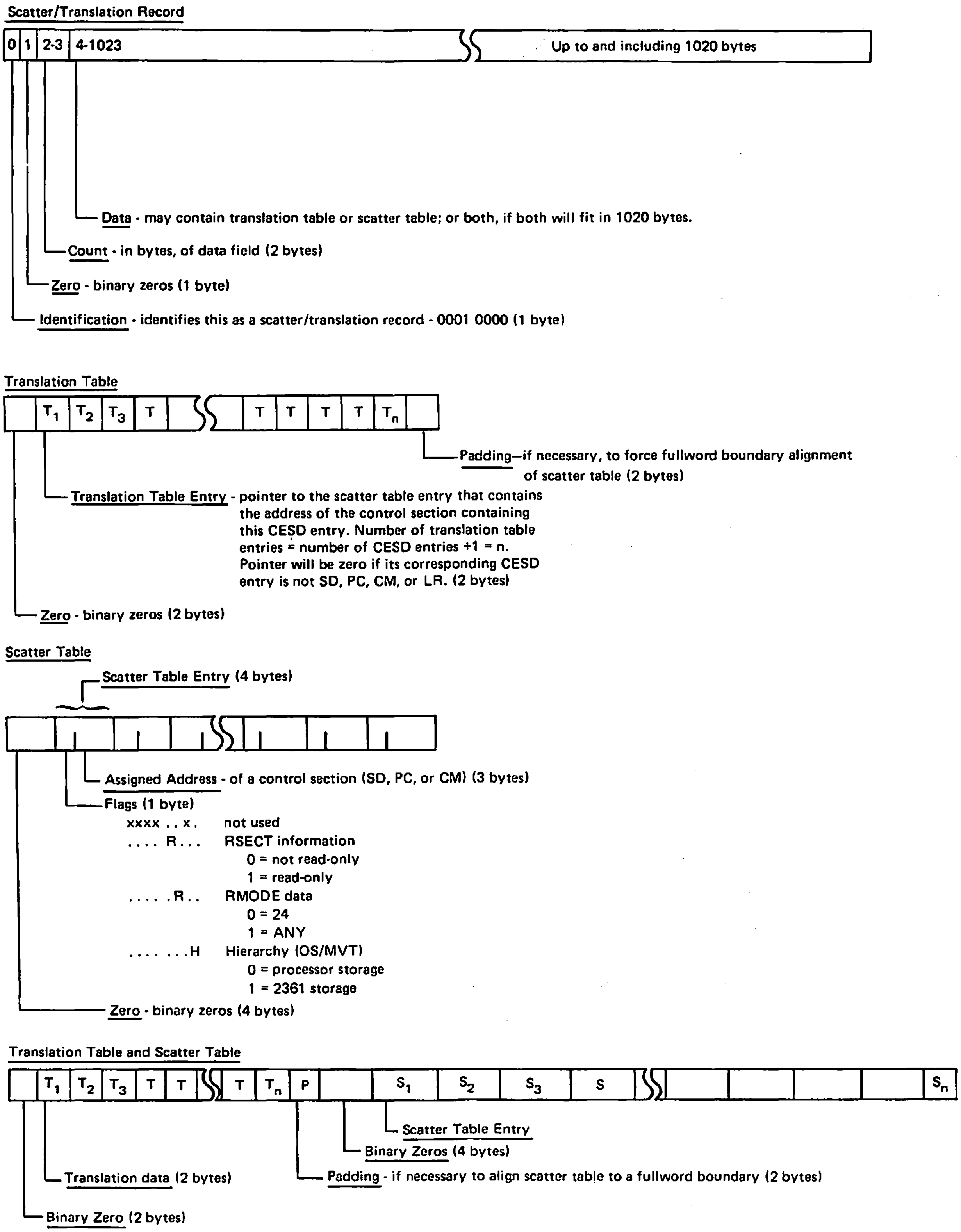
Type		Subclassification	
Section Definition (SD)	xxxx 0000	Delete	xxx1 xxxx
Label Reference (LR)	xxxx 0011	Replace	xxx1 xxxx
Private Code (PC)	xxxx 0100	Insert	xx1x xxxx
Common (CM)	xxxx 0101	Chain	x1xx xxxx
Pseudo Register (PR)	xxxx 0110	Map	1xxx xxxx
Null	0000 0111		
External Reference (ER)	xxxx 0010		
Weak External Reference (WX)	xxxx 1010		

Note: x may be 1 or 0.

Private Code marked delete
(ENTAB and SEGTAB control sections) xxx1 x100

Symbol - The eight-character external name (zero when type is Null)

Figure 77. CESD Record (Load Module)



Scatter/Translation Record

Data - may contain translation table or scatter table; or both, if both will fit in 1020 bytes.

Count - in bytes, of data field (2 bytes)

Zero - binary zeros (1 byte)

Identification - identifies this as a scatter/translation record - 0001 0000 (1 byte)

Translation Table

Padding—if necessary, to force fullword boundary alignment of scatter table (2 bytes)

Translation Table Entry - pointer to the scatter table entry that contains the address of the control section containing this CESD entry. Number of translation table entries = number of CESD entries +1 = n. Pointer will be zero if its corresponding CESD entry is not SD, PC, CM, or LR. (2 bytes)

Zero - binary zeros (2 bytes)

Scatter Table

Scatter Table Entry (4 bytes)

Assigned Address - of a control section (SD, PC, or CM) (3 bytes)

Flags (1 byte)

```
xxxx ..x.   not used
.... R...   RSECT information
              0 = not read-only
              1 = read-only
..... R..   RMODE data
              0 = 24
              1 = ANY
.......H    Hierarchy (OS/MVT)
              0 = processor storage
              1 = 2361 storage
```

Zero - binary zeros (4 bytes)

Translation Table and Scatter Table

Scatter Table Entry

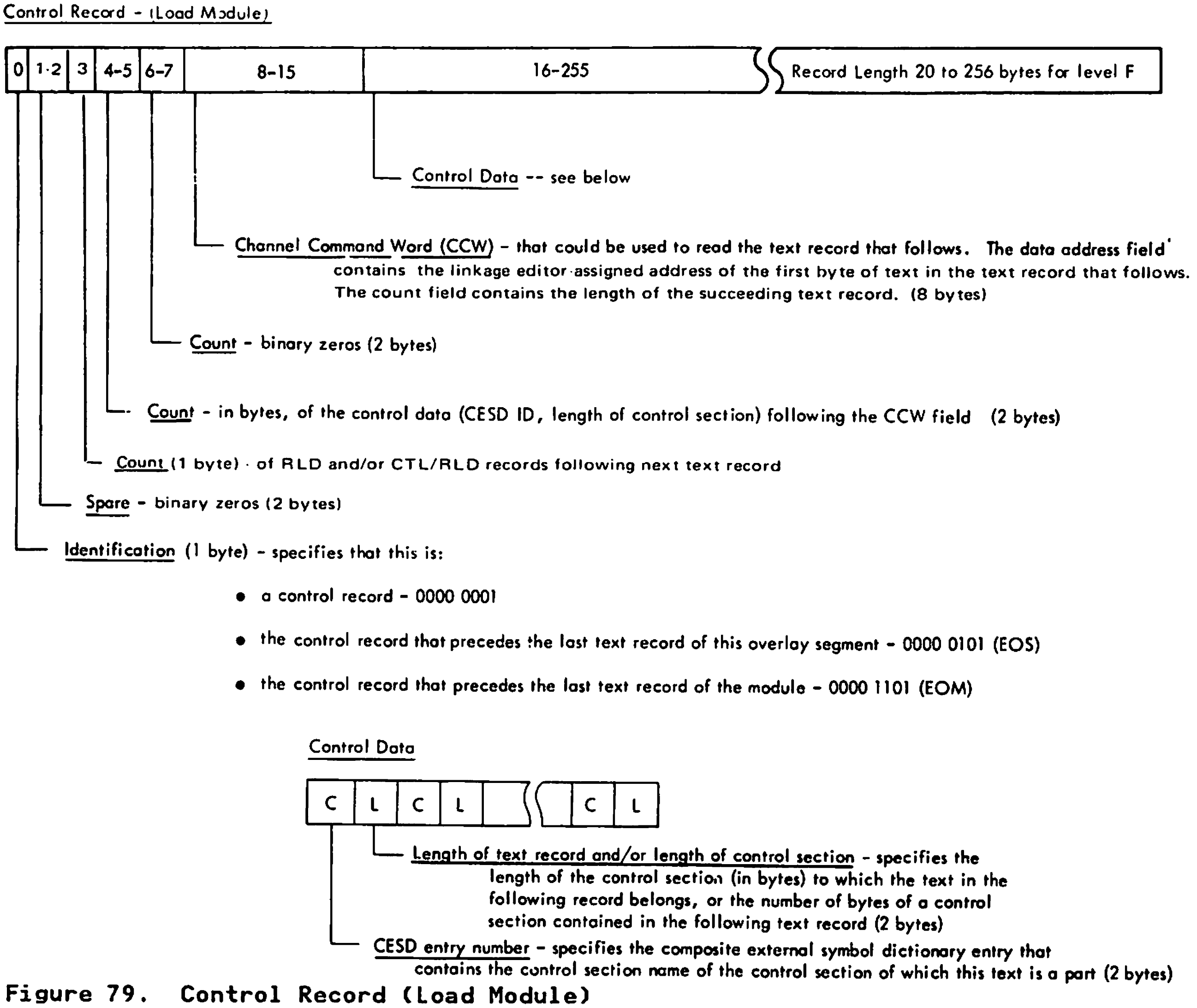
Binary Zeros (4 bytes)

Translation data (2 bytes)

Padding - if necessary to align scatter table to a fullword boundary (2 bytes)

Binary Zero (2 bytes)

Figure 78. Scatter/Translation Record


Control Record - (Load Module)
0
1-2
3
4-5
6-7
8-15
16-255
Record Length 20 to 256 bytes for level F
Control Data
C
L
C
L
C
L

Control Data -- see below

Channel Command Word (CCW) - that could be used to read the text record that follows. The data address field' contains the linkage editor-assigned address of the first byte of text in the text record that follows. The count field contains the length of the succeeding text record. (8 bytes)

Count - binary zeros (2 bytes)

Count - in bytes, of the control data (CESD ID, length of control section) following the CCW field (2 bytes)

Count (1 byte) - of RLD and/or CTL/RLD records following next text record

Spare - binary zeros (2 bytes)

Identification (1 byte) - specifies that this is:

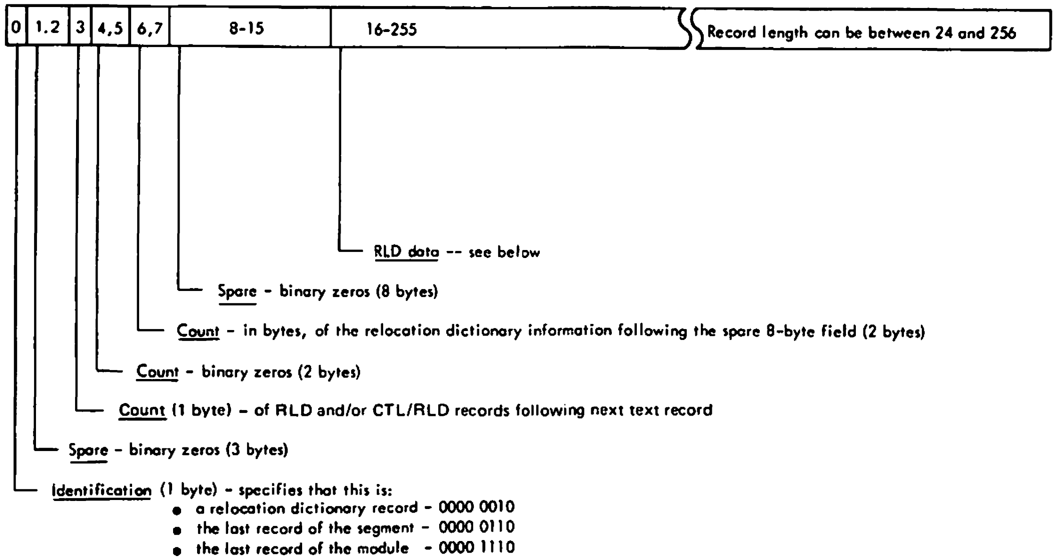
- a control record - 0000 0001
- the control record that precedes the last text record of this overlay segment - 0000 0101 (EOS)
- the control record that precedes the last text record of the module - 0000 1101 (EOM)

Length of text record and/or length of control section - specifies the length of the control section (in bytes) to which the text in the following record belongs, or the number of bytes of a control section contained in the following text record (2 bytes)

CESD entry number - specifies the composite external symbol dictionary entry that contains the control section name of the control section of which this text is a part (2 bytes)

Figure 79. Control Record (Load Module)

Relocation Dictionary Record - (Load Module)



RLD data -- see below

Spare - binary zeros (8 bytes)

Count - in bytes, of the relocation dictionary information following the spare 8-byte field (2 bytes)

Count - binary zeros (2 bytes)

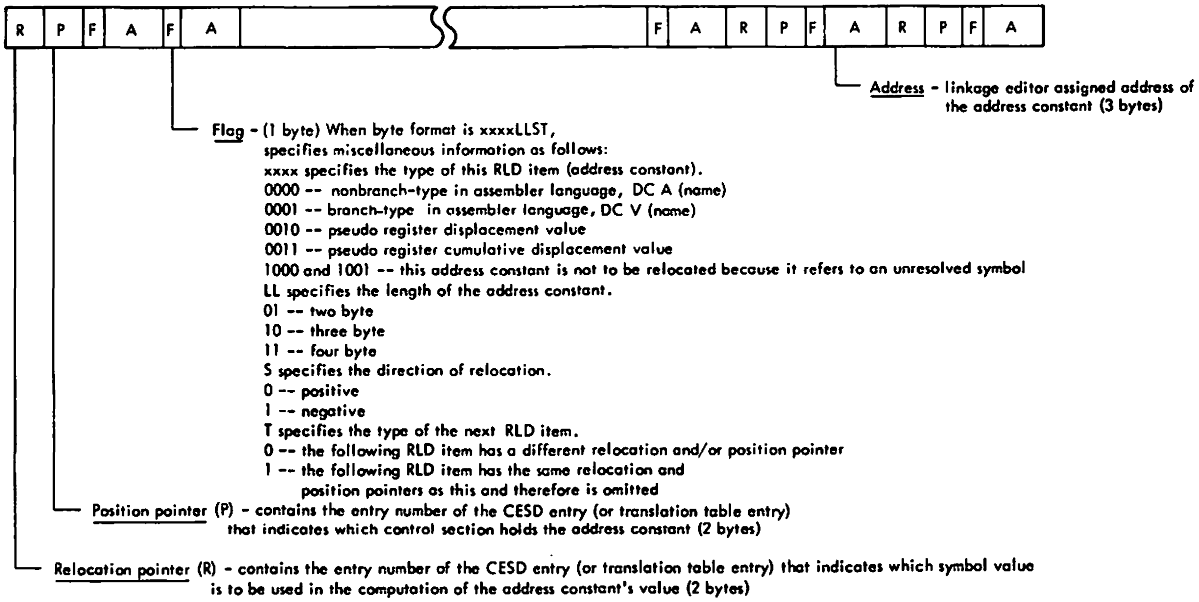
Count (1 byte) - of RLD and/or CTL/RLD records following next text record

Spare - binary zeros (3 bytes)

Identification (1 byte) - specifies that this is:

- a relocation dictionary record - 0000 0010
- the last record of the segment - 0000 0110
- the last record of the module - 0000 1110

RLD Data



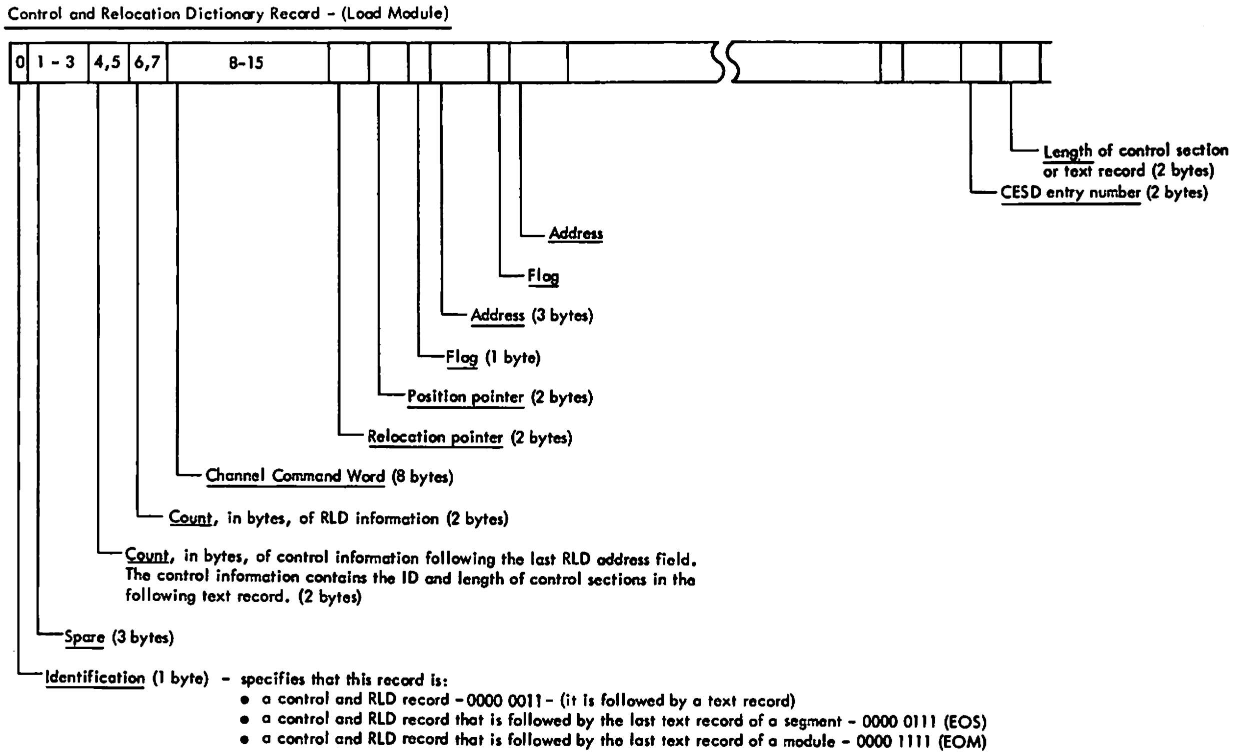
Address - linkage editor assigned address of the address constant (3 bytes)

Flag - (1 byte) When byte format is xxxxLLST,
specifies miscellaneous information as follows:
xxxx specifies the type of this RLD item (address constant).
0000 -- nonbranch-type in assembler language, DC A (name)
0001 -- branch-type in assembler language, DC V (name)
0010 -- pseudo register displacement value
0011 -- pseudo register cumulative displacement value
1000 and 1001 -- this address constant is not to be relocated because it refers to an unresolved symbol
LL specifies the length of the address constant.
01 -- two byte
10 -- three byte
11 -- four byte
S specifies the direction of relocation.
0 -- positive
1 -- negative
T specifies the type of the next RLD item.
0 -- the following RLD item has a different relocation and/or position pointer
1 -- the following RLD item has the same relocation and position pointers as this and therefore is omitted

Position pointer (P) - contains the entry number of the CESD entry (or translation table entry) that indicates which control section holds the address constant (2 bytes)

Relocation pointer (R) - contains the entry number of the CESD entry (or translation table entry) that indicates which symbol value is to be used in the computation of the address constant's value (2 bytes)

Figure 80. Relocation Dictionary Record (Load Module)


Control and Relocation Dictionary Record - (Load Module)
0
1 - 3
4,5
6,7
8-15
Length of control section or text record (2 bytes)
CESD entry number (2 bytes)
Address
Flag
Address (3 bytes)
Flag (1 byte)
Position pointer (2 bytes)
Relocation pointer (2 bytes)
Channel Command Word (8 bytes)
Count, in bytes, of RLD information (2 bytes)
Count, in bytes, of control information following the last RLD address field. The control information contains the ID and length of control sections in the following text record. (2 bytes)
Spare (3 bytes)
Identification (1 byte) - specifies that this record is:
• a control and RLD record - 0000 0011 - (it is followed by a text record)
• a control and RLD record that is followed by the last text record of a segment - 0000 0111 (EOS)
• a control and RLD record that is followed by the last text record of a module - 0000 1111 (EOM)

Notes: For detailed descriptions of the data fields see Relocation Dictionary Record and Control Record.
The record length varies from 20 to 256 bytes.

Figure 81. Control and Relocation Dictionary Record (Load Module)

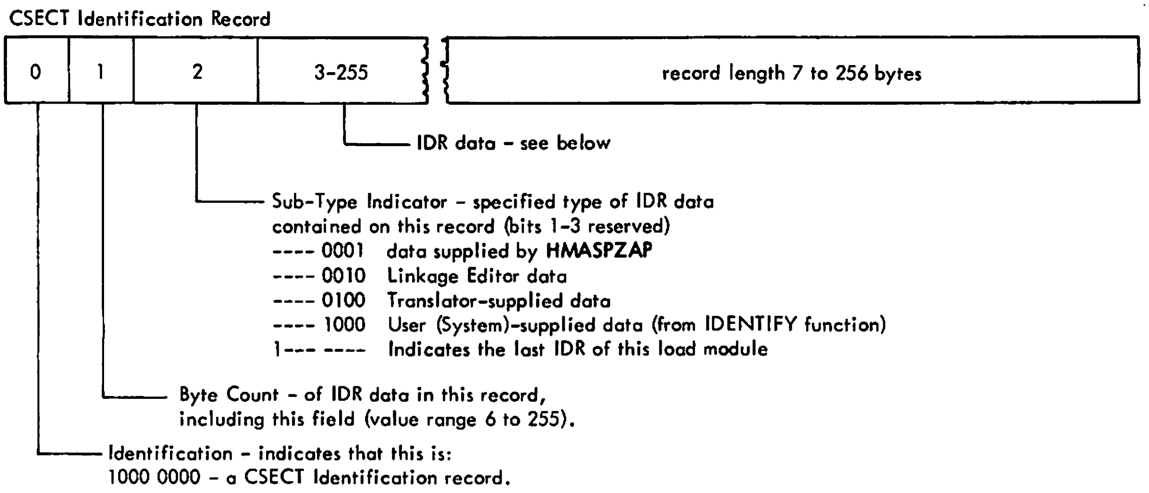
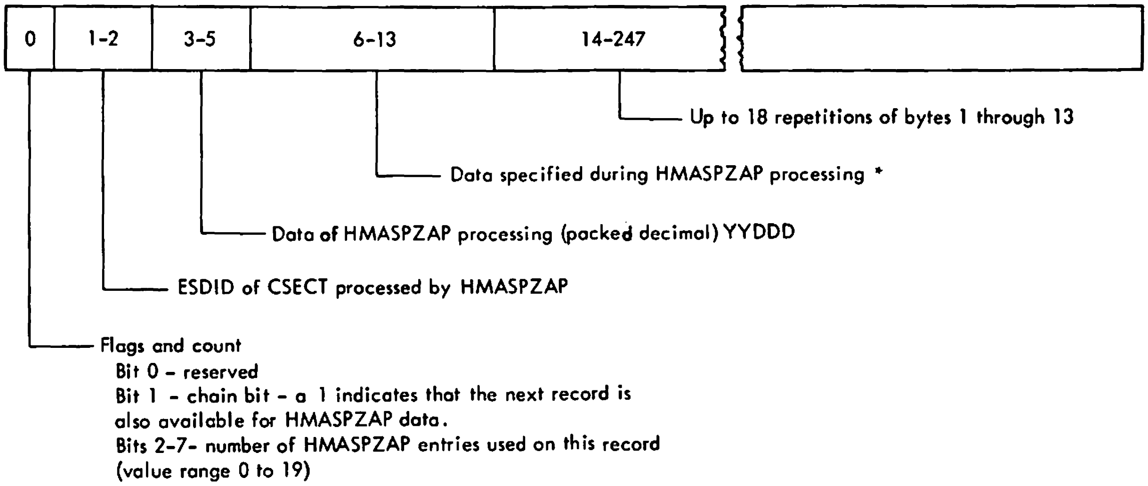

CSECT Identification Record
0
1
2
3-255
record length 7 to 256 bytes
IDR data - see below
Sub-Type Indicator - specified type of IDR data contained on this record (bits 1-3 reserved)
---- 0001 data supplied by HMASPZAP
---- 0010 Linkage Editor data
---- 0100 Translator-supplied data
---- 1000 User (System)-supplied data (from IDENTIFY function)
1--- ---- Indicates the last IDR of this load module
Byte Count - of IDR data in this record, including this field (value range 6 to 255).
Identification - indicates that this is:
1000 0000 - a CSECT Identification record.

Figure 82 (Part 1 of 3). Record Format of Load Module IDRs

HMASPZAP Data



Up to 18 repetitions of bytes 1 through 13

Data specified during HMASPZAP processing *

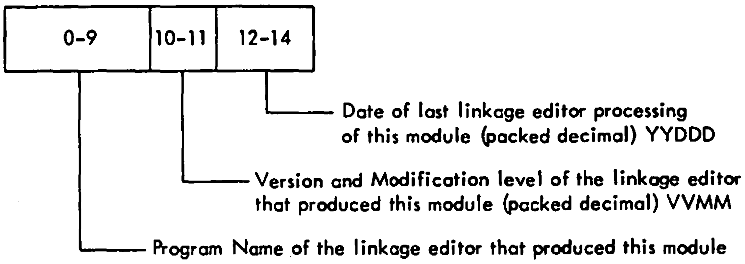
Data of HMASPZAP processing (packed decimal) YYDDD

ESDID of CSECT processed by HMASPZAP

Flags and count
Bit 0 - reserved
Bit 1 - chain bit - a 1 indicates that the next record is also available for HMASPZAP data.
Bits 2-7- number of HMASPZAP entries used on this record (value range 0 to 19)

*May be a PTF number or up eight bytes of variable user data specified on an HMASPZAP IDRDATA control statement.

Linkage Editor Data

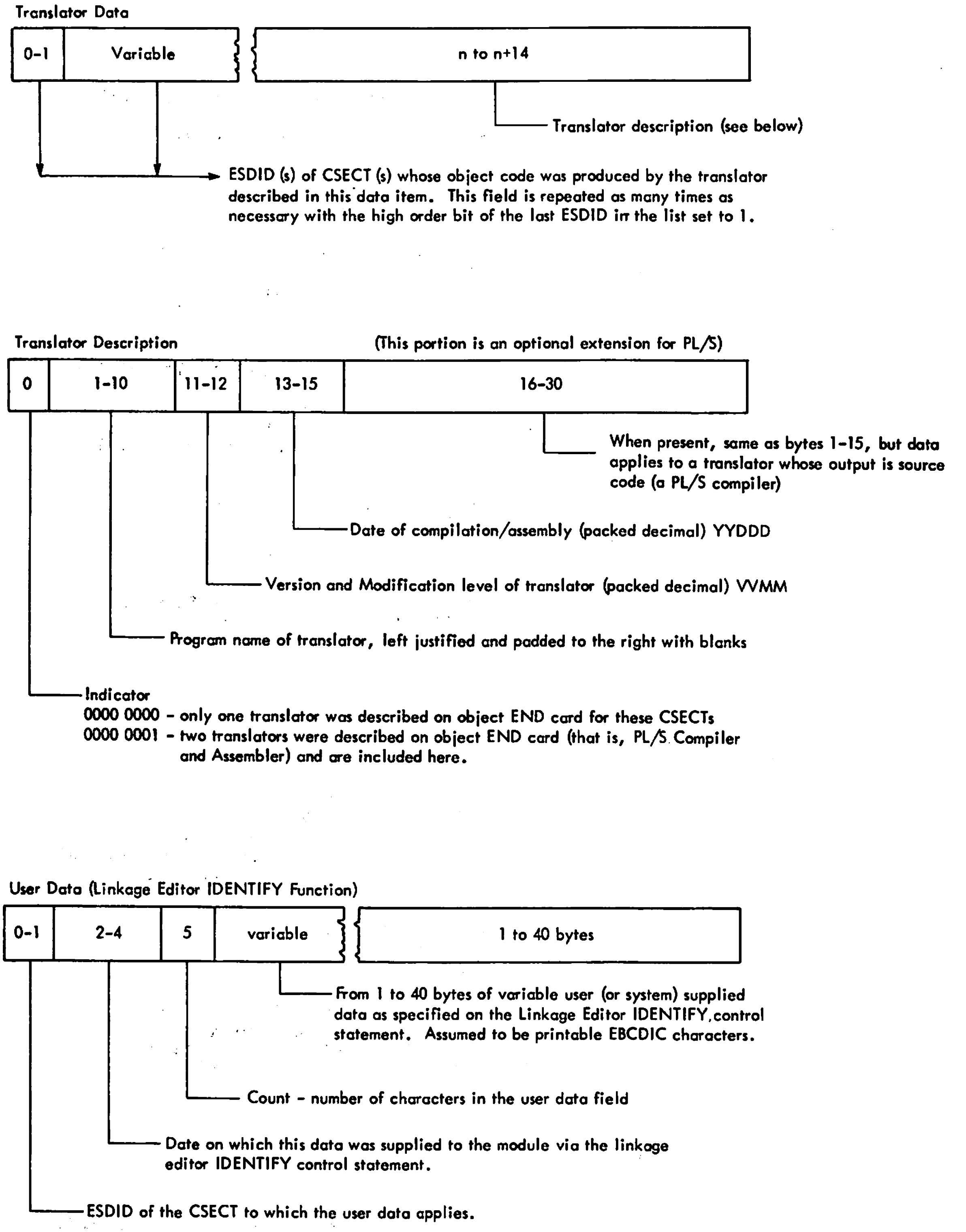


Date of last linkage editor processing of this module (packed decimal) YYDDD

Version and Modification level of the linkage editor that produced this module (packed decimal) VVMM

Program Name of the linkage editor that produced this module

Figure 82 (Part 2 of 3). Record Format of Load Module IDRs



Translator Data

0-1	Variable	n to n+14

- Translator description (see below)
- 0-1, Variable: ESDID (s) of CSECT (s) whose object code was produced by the translator described in this data item. This field is repeated as many times as necessary with the high order bit of the last ESDID in the list set to 1.

Translator Description (This portion is an optional extension for PL/S)

0	1-10	11-12	13-15	16-30

- 16-30: When present, same as bytes 1-15, but data applies to a translator whose output is source code (a PL/S compiler)
- 13-15: Date of compilation/assembly (packed decimal) YYDDD
- 11-12: Version and Modification level of translator (packed decimal) VVMM
- 1-10: Program name of translator, left justified and padded to the right with blanks
- 0: Indicator
  0000 0000 - only one translator was described on object END card for these CSECTs
  0000 0001 - two translators were described on object END card (that is, PL/S Compiler and Assembler) and are included here.

User Data (Linkage Editor IDENTIFY Function)

0-1	2-4	5	variable	1 to 40 bytes

- variable: From 1 to 40 bytes of variable user (or system) supplied data as specified on the Linkage Editor IDENTIFY control statement. Assumed to be printable EBCDIC characters.
- 5: Count - number of characters in the user data field
- 2-4: Date on which this data was supplied to the module via the linkage editor IDENTIFY control statement.
- 0-1: ESDID of the CSECT to which the user data applies.

Figure 82 (Part 3 of 3). Record Format of Load Module IDRs

A

A-type address constant
   delinking of 65
   relocation of 61
   RLD processing for 45
absolute relocation 18
   See also absolute relocation factor, relocation
   examples 63
   in entry processing 56
absolute relocation factor 62
   See also absolute relocation
   definition 62
   determination of 45
   use in relocating delinked address constants 62
   use in relocating V-type address constants 66
adcon
   See address constant
additional call libraries
   See LIBRARY statement
additional input sources
   See INCLUDE statement
address assignment 59
   See also address assignment processor
   function 2-3
   general description 17
   in cross-reference table 59
   of first text record as zero 8
   of main entry point 56
address assignment processor (HEWLFADA)
   description of 51-53
      chart 127
      synopsis 94
   operation diagram 87
address constant 3
   See also A-type address constant, pseudo register, V-type address constant
   computing the value of 4
   delinking 45, 66
   in RLD processing 45
   purpose 3
   relocation 61, 71
      See also address assignment
addressing mode 1
addressing mode (AMODE) 8
ALIAS statement
   general description of processing 24
   operation diagram 82
   processor 29
alias table
   construction of 56
   format 161
   introduction to 17
ALIGN2 format attribute 7
all purpose table (APT)
   format 151
   indicators for MAP, XREF options 58
   introduction to 15
   making an entry in 89
   preparation of 19, 89
allocation
   of buffers and tables 21-22
   of virtual storage 21, 22, 169
      initialization 15
allocation processor (ALL001) 89
APT
   See all-purpose table
attributes and options processor (HEWLFOPT) 89
attributes, module
   analysis of (HEWLFOPT) 89
   descriptions 6-8
   incompatible (table) 19
   processing examples 8-11
   use of 6-8
authorization code
   in partitioned organization director record 173
authorization code (AC) 8
automatic library call 1
   See also automatic library call processor
   function 1
automatic library call processor (HEWLCAUT)
   description of 49-51
      charts 121, 125
      synopsis 94
automatic promotion of common 38
automatic replacement 65

B

blank common
   See also common
   definition of 34
   delinking of 66
   in resolution processing 37
BLDL list, making an entry in 48, 50
BLDL macro instruction 47, 48
block format attribute 7
blocked input
   on SYSLIB 23
   on SYSLIN 23
blocked output
   on SYSPRINT 23
boundary alignment factor 52
branch-type address constant
   See V-type address constant
buffer allocation 22, 187

C

call library
   See automatic library call, SYSLIB data set
calls
   downward 54, 67
   exclusive 54, 67
   invalid exclusive 67
   upward 54, 67
calls list
   format 162
   making entries in 17, 45

scanning 51-55
calls, automatic library
See automatic library call
CESD
See composite external symbol dictionary
CESD entry number 27, 83
CESD identifier 35-37, 93
CESD record
format 200
in input load module 32, 34
CESD record types 34
CHANGE statement
operation diagram 82
processing 24
processor 28
CM
See common
combining object modules 1
common (CM)
See also blank common
definition 34
delinking of 66
nonresolution processing 36
resolution processing 37
Common path routine (HEWLCPTH)
automatic promotion of common 38
in program organization 95
communication area
See all purpose table
composite dictionaries 4, 5
See also composite external symbol dictionary, relocation dictionary
composite ESD
See composite external symbol dictionary
composite external symbol dictionary (CESD)
contents 9
definition 34
internal format 163, 165
introduction to 4
making an entry in 15, 93
written out on SYSLMOD 57
concatenated data sets
on SYSLIB 12
on SYSLIN 11, 23
control dictionaries 3
See also external symbol
purpose 3
when generated 3-5
control information processing
analyzing 19
description of 19
operation diagram 82
control record
contents 10
in input module 32
control section (CSECT) 2
See also no-length control section
assigning addresses to 2-3
delinking of 66
grouping on SYSLMOD 60
replacement 38, 65
control section search routine (GETIDMUL) chart 137
in program organization 95
control statement
format 24
operands 24
operation diagram 82
pointers (P1, P2) 24
processing 15, 24-31
work areas (OPD0, OPD1) 24-25
control statement scanner (HEWLFSCN)
processing 24
synopsis 93
control/RLD dictionary record 10
See also control/RLD record write
contents 10
creation of 59, 95
format 204
control/RLD record
See control/RLD dictionary record
control/RLD record write routine (WRTCRRLD) 95
cross-reference between control sections 3
cross-reference table 59
See also options
XREF option
See also options, XREF option
contents 59
detailed description of production 59
in program organization 94, 96
CSECT identification record (IDR)
contents 1, 16
data types
See IDR data types
description 9-10
formats 204
IDENTIFY control statement 41
in load module 16, 39-40, 41
in object module END records 39-40
processing (HEWLFIDR)
description of 39-41
operation diagrams 82
program processing history 1
written out on SYSLMOD 17

D

data control block
initializing 15, 93
used to determine module type 12
DC
See downward compatible attribute
DCB
See data control block
DCBS option
function 6
delink table
contents 16
format 165
making an entry in 45
delinking
definition 65
example 65
in RLD processing 46
of A-type (nonbranch-type) address constants 65
of common control sections 66
of external symbols 65
dense record 43
determining ESD type 34
diagnostic aids
buffer allocation 187
error message—module cross-reference table 189
register contents at entry to module 183
table allocation 188
diagnostic directory print routine (HEWLFBTP)

  in program organization 96
  LIST used during processing 180
  TABLE used during processing 180
diagnostic messages construction 73
  See also error logging
diagnostic output data set
  See SYSPRINT data set
diagrams, operation 15, 75-88
directory, microfiche 146
downward call 54, 67
downward calls list
  format 165
  introduction to 17
downward compatible attribute (DC)
  definition 7
  grouping control sections 60
downward reference 54, 67

E

end of module (EOM) indicator 3
  See also END record
  description of 3
  in control record 10
  in END processing 93
  setting 61
end of segment (EOS) indicator
  in control record 10
  setting 58
END processing (HEWLFEND)
  description of 47
    chart 115
    synopsis 93
END record
  containing IDR data 39-40
  format 198
  in entry processing 56
END record (continued)
  in object module 31
end-of-module (EOM) indicator
  See also END record
  in input load module 32
ENTAB
  See entry table
enter routine 36
entry list
  contents 18
  format 166
  relocation 18
  use of 66
entry point 8
  See also ENTRY statement
  assigned by linkage editor 8
  in END processing 47
  processing 56
entry processor (HEWLFENT)
  description of 56
    charts 129
ENTRY statement
  general description of processing 24
  operation diagram 82
  processor 29
entry table (ENTAB)
  See also entry table creation routine
  entries in CESD 17
  format 166
  introduction to 10
  PC-delete entry for 34, 52
  used in relocating V-type address constants 68
entry table creation routine (SCDENTAB)
  description of 68
  in program organization 95
entry table size determination routine calls list format
  processing 54-55
  processing chart 128
EOM
  See end of module indicator
ER
  See external reference
error logging
  description of 73
  routine (HEWLFLOG)
    chart 145
  table and list used 180
error message/issuer—module cross-reference table 189
error messages
  See diagnostic messages
error processing
  See diagnostic directory print routine
  See error logging
ESD
  See external symbol dictionary
ESD ID
  See ESD identifier
ESD identifier
  definition 4
  entry in high ID table 58
  in entry processing 56
  in ESD processing 35
  in text processing 43
ESD record
  definition 16
  format 196
  in object module 31
  operation diagram 83
  processing 35-39
ESD record types 34
exclusive call 54, 67
  See also options, XCAL option
executable attribute 7
EXPAND statement
  general description of processing 24
EXPAND statement processor 30
external dummy section
  See pseudo register
external reference (ER)
  definition 2, 34
  entry processing 35
  nonresolution processing 37
  resolution processing 34, 38
  unresolved ER processing 53
external symbol definition 4
external symbol dictionary (ESD)
  contents 4
  entry types 34
  identifier
    See ESD identifier
  introduction to 4
  processing (HEWLFESD)
    description of 35-39
    description of chart 106
    description of synopsis 92
    operation diagram 83

F

final linked address 53
final processing (HEWLFFNL)
  description of 18, 72
    charts 144-145
    synopsis 95
  objective 18
  operation diagram 81
  overview 14
final relocation constant 53
first pass RLD buffer
  See RLD buffer
formats, record 195-206
freeline routine 36
functions of the linkage editor 1-2

G

general registers, contents of 33

H

half ESD table (HESD)
  contents 17
  format 167
  in ENTAB creation 68
  production of 58, 94
  saving relocation factors in 61
HESD
  See half ESD table
HEWLCPTH
  See Common path routine
HEWLCR01
  See SYNAD routine
HEWLFADA
  See address assignment processor
HEWLFAUT
  See automatic library call processor
HEWLFEND
  See End processing
HEWLFENT
  See entry processor
HEWLFESD
  See external symbol dictionary processing
HEWLFFNL
  See final processing
HEWLFIDR
  See CSECT identification record
HEWLFINC
  See include processing
HEWLFINP
  See input processing
HEWLFINT
  See initialization processing
HEWLFLOG
  See error logging, routine
HEWLFMAP
  See MAP/XREF processor
HEWLFMDI
  See object module processing
HEWLFOPT
  See attributes and options processor
HEWLFOUT
  See intermediate output processor
HEWLFRAT
  See text and RLD processor
HEWLFREL
  See RLD control block, second pass text control block format
HEWLFROU
  See linkage editor, entry point
HEWLFSCD
  See second pass processing
HEWLFSCN
  See control statement scanner
HEWLFSIO
  See second pass processing
HEWLFSYM
  See symbol record, processor
HEWLFTXT
  See text record, processing
HEWLMBTP
  See diagnostic directory print routine
HIARCHY
  See hierarchy format attribute
HIARCHY statement
  general description of processing 24
  processor 28
hierarchy format attribute (HIAR) 7, 24
hierarchy table 28
high ID table (HIID)
  construction of 94
  contents 17
  format 168
  making an entry in 58
HMASPZAP data 40
  See also IDR data types
  processing (HEWLFIDR)
    chart 119
    description of 40

I

I/O control table, address of text note list on SYSUT1 43, 47
  ordering of text 60
I/O flow 11-14
ID-length list 43
identification record
  See CSECT identification record
IDENTIFY statement 24
  See also CSECT identification record
  general description of processing 24
  operation diagram 82
  processing data 41
    chart 119
  processor 29
IDR
  See CSECT identification record
IDR data types 39
  IMASPZAP-supplied data processing 40
  linkage editor data processing 40-41
  translator-supplied data processing 39, 40
    chart 117
  user-supplied data 41
    See also IDENTIFY statement
    processing 41
    processing chart 120
IDR tables 57
  See also CSECT identification record written onto SYSLMOD
  translator data table (IDRTRTAB)
    contents 47

entry 41
user data table (IDRUDTAB)
contents 41
include processing (HEWLFINC) 10, 93, 124
INCLUDE statement 24
See also include processing
general description of processing 24
in input processing 20
operation diagram 82
processor 25
incompatible module attributes 20
initialization processing (HEWLFINT)
description of 15, 19
synopsis 89
operation diagram 77
input
See automatic library call, SYSLIN data set
input conventions 194
input data set
See SYSLIN data set
input entry types
See input record types
input processing (HEWLFINP)
description of 15-16, 22
chart 102
synopsis 90
operation diagram 78
overview 11-14
input record types
associated processors 91
formats 195-206
general register contents for 33
processing 22, 32
input text buffer (TXTBFBEG)
general description of use 16, 18
in single pass processing 60
in text processing 41, 43
minimum length 21
input/output flow 11-14
INP270
See load module processing
INSERT statement
general description of processing 24
operation diagram 82
processor 28
intermediate output processor (HEWLFOUT)
description of 16-17, 57
synopsis 94
use of hierarchy table 28
intermediate processing
description of 16-17, 51-56
charts 127, 132
synopsis 94
objectives 16
operation diagram 79
overview 13

L

label definition (LD)
definition 34
entry processing 34
nonresolution processing 37
label reference (LR)
definition 34
nonresolution processing 37
resolution processing 38
label routine 38
language translators 1
LD
See label definition
LET option 6
level 0 symbol 24
See also control statement operands
level 1 symbol 24
See also control statement operands
library calls
function 1
library read block 48
LIBRARY statement
in automatic library call processing 50, 51
See also automatic library call
in input processing 12
operation diagram 82
processor 30
line number 4, 36
See also CESD identifier, ESD identifier
linkage editor
data
See IDR data types
data sets 11
description 2-3
design points 1-2
entry point (HEWLFROU) 89
functions 1
method of operation 15
options 5-6
organization 89-145
overlay structures 2
purpose 1-2
relationship to the operating system 2
linkage editor assigned address of first text record as zero 8
linking object modules 2
LIST option 6
load module
attributes 6, 8
See also attributes, module
data set
See SYSLMOD data set
definition 3
processing (INP270)
description of 32
description of chart 104
description of synopsis 91
record types 32
See also input record types
structure 3-5, 6-11, 16-18
text processing 43
See also text record processing
logical record length (LRECL)
for SYSLIB 23
for SYSLIN 23
for SYSPRINT 23
for SYSTERM 23
loose record 43
LR
See label reference
LRECL
See logical record length

M

MAP option 6
   See also MAP/XREF processor
   function 6
   general description of processing 16-17
   in intermediate output processing 58
MAP/XREF processor (HEWLFMAP)
   description of 59
      chart 132
      in program organization 94
map, module 17
   See also MAP option, MAP/XREF processor, options
   contents 17, 58
   production of 13, 58
      chart 132
      in program organization 94
member name for SYSLMOD data set 12-14
messages 14
   See also diagnostic aids
microfiche directory 146
MODE statement
   general description of processing 24
MODE statement processor 31
module
   origin 2
   structure 2-5
module attributes
   See attributes, module
module map
   See map, module
module—CSECT cross-reference table 149
multiplicity
   in relocating address constants 72
   read into input text buffer 41
   writing on SYSLMOD 60
   writing on SYSUT1 43

N

NAME statement
   general description of processing 24
   in final processing 72
   in input processing 22, 49
   operation diagram 82
   processor 28
NCAL option
   See also automatic library call
   function 6
   in input processing 22, 49
NE (not editable attribute) 8
no-length control section 42, 93
nonresolution processing 36-37
not editable attribute (NE) 8
note list (for overlay segments) 10
null type
   changing CESD entries to 51
   definition 34
   entry processing 36

O

object module
   definition 1
   processing (HEWLFMDI)
      description of 32
      description of?synopsis 90
   record types 31
      See also input record types
   structure 3-5
   text processing 42, 43
      See also text record processing
object module buffer 15, 42
OL (only loadable attribute) 7
opening data sets 20
operation diagrams 15, 75-88
options
   analysis of (HEWLFOPT) 19, 89
   diagnostic output 5, 6
      See also LIST, MAP, TERM, XFER options
   introduction to 2
   processing 5, 6
      See also LET, NCAL, XCAL options
   space allocation 6
      See also DCBS, SIZE options
   specification of 6
ORDER statement
   and ALIGN2 attribute 7
   processor 29
ORDER table
   creation 29
      diagrams 78
   in address assignment 52, 53
   in END processing 47
   in intermediate output 59
      diagram 79
   layout 182
out of order text 42
   use during text processing 42
output load module data set
   See SYSLMOD data set
overlay format attribute (OVLY) module characteristics 7
overlay module processing 2
   See also OVERLAY statement
   description 2
   relocating V-type address constants 66
OVERLAY statement
   general description of processing 24
   operation diagram 82
   processor 27
OVLY
   See overlay format attribute

P

P (position) pointer
   definition 4
   in entry table size determination 54
   updating 16
PAGE statement
   and ALIGN2 attribute 7
   general description of processing 24
   processor 29
partitioned data set (PDS)
   directory entry for SYSLMOD 18, 72
   generating a load module in 2, 6

  - include processing for 48-49
  - SYSLIB data set 12
  - SYSLMOD data set 12, 13
- PC
  - See private code
- PDS
  - See partitioned data set
- position pointer
  - See P pointer
- PR
  - See pseudo register
- private code (PC)
  - definition 34
  - entry processing 34, 35
  - marked delete 34, 52
- PROCENTY routine 25
- program
  - organization 89-145
  - processing history 1
- pseudo register
  - address assignment 18
  - computing cumulative PR length 53
  - definition 34
  - in RLD processing 44, 71
- purpose of the linkage editor 1

R

- R (relocation) pointer
  - definition 4
  - in entry table size determination 54
  - updating 16, 44
- RCT
  - See relocation constant table
- RDRLD
  - See RLD read routine
- RDTXT
  - See text read routine
- read control block
  - for SYSLIN 22
- read only control section 8
- READ8 routine
  - chart 123
  - in control statement processing 25
  - in program organization 93
- record
  - See also ESD record
  - dense (contiguous) 43
  - formats 195-206
  - loose (noncontiguous) 43
  - types
    - See input record types
- record types
  - LIST used in construction 180
  - on SYSPRINT data set 12, 13, 14
    - See also SYSPRINT data set
  - on SYSTERM data set 12
    - See also SYSTERM data set
  - TABLE used in construction 180
- recovery management routine and refreshable attribute 6
- reenterable attribute (RENT) 6
- REFR
  - See refreshable attribute
- refreshable attribute (REFR) 6
- registers, contents of 31-33
  - load module processing 33
  - object module processing 31
- relative relocation
  - definition 18
  - in entry processing 56
  - marking RLD item for 44
- relative relocation factor
  - determination of 45, 54
  - general description of use 18
  - use in determining buffer relocation constant 71
    - See also relocation constant table
- RELOCATE routine
  - See relocation routine
- relocation
  - of address constants 17-18, 61
    - See also address assignment processor, address constant
  - of RLD items during intermediate output processor 59
- relocation constant 17, 18
- relocation constant table (RCT)
  - contents 17, 18
  - format 174
  - making an entry in 52
- relocation dictionary (RLD)
  - and text processor (HEWLFRAT) 92
    - See also text and RLD processor
  - contents 4
  - introduction to 4
  - processing (RLD001)
    - description of 44-47
    - description of chart 112
    - description of synopsis 92
    - operation diagram 86
  - record
    - See RLD record
- relocation pointer
  - See R pointer
- Relocation routine (RELOCATE)
  - description of
    - in program organization 95
- RENT (reenterable attribute) 6
- renumbering table (RNT)
  - contents 16
  - format 174
  - production of 35-39, 92
  - use of 35, 40
- REPLACE statement
  - general description of processing 24
  - processor 28
- replace/change chain in the CESD 28
  - See also CHANGE statement, REPLACE statement
- residence mode 1
- residence mode (RMODE) 8
- resolution processing 37-39
- REUS (reusable attribute) 7
- reusable attribute (REUS) 7
- RLD
  - See relocation dictionary
- RLD buffer
  - first pass 16
  - general description of use 16, 18
  - in single pass processing 60
- RLD control block
  - initialization (HEWLFREL) 95
  - input control block 60
    - format 171, 175
  - output control block 60
    - format 175
- RLD flags
  - in relocation address constants (table) 45-47
- RLD input buffer
  - See second pass RLD input buffer
- RLD note list
  - format 177
  - introduction to 13

  - making entries in 16, 44-47
  - reading from SYSUT1 60, 94
  - RLD Processor 92
  - use in locating RLD records 60
  - use in relocating address constants 72
- RLD output buffer
  - See second pass RLD output buffer
- RLD pointers 4, 92
  - See also P pointer, R pointer
- RLD read routine (RDRLD) processing
  - chart 138
  - in program organization 95
- RLD record
  - created for overlay 61
  - format 203
  - in load module 32
  - in object module 31, 32
  - processing (RLD001) 16, 44, 47
    - See also relocation dictionary processing
- RLD set
  - definition 71
  - use in relocating address constants 71
  - written on SYSUT1 44, 46, 47
- RLD write routine (on SYSUT1) (RLDBUF)
  - chart 114
  - in program organization 92
- RLD/control record
  - See control/relocation dictionary record
- RLDBUF
  - See RLD write routine
- RLD001
  - See relocation dictionary, processing
- RNT
  - See renumbering table
- root segment
  - in overlay module 10

S

- scatter format attribute (SCTR)
  - grouping control sections 60
  - in processing example 11
  - module characteristics 7
- scatter loading
  - See scatter format attribute
- scatter table
  - format 201
  - introduction to 11
  - production of 17, 58
- scatter/translation record
  - format 201
  - function 7
  - introduction to 11
  - written on SYSLMOD 58, 94
- SCDENTAB
  - See entry table creation routine
- SCTR
  - See scatter format attribute
- SD
  - See section definition
- second pass processing (HEWLFSCD)
  - description of
    - charts 135
    - synopsis 95
  - objectives 17
  - operation diagrams 80, 88
  - overview 13
- second pass RLD input buffer
  - reading RLD records into 18, 60
  - use in relocating address constants 72
- second pass RLD output buffer
  - entering RLD items in 18, 60
  - use in relocating address constants 71
- second pass text buffer, reading text records into 60
- second pass text control block
  - format 178
  - initialization (HEWLFREL) 95
- second pass text control table, making an entry in 60
- section definition (SD)
  - definition 34
  - nonresolution processing 36
  - resolution processing 38
- SEGLGTH
  - format 179
  - introduction to 17
  - processing 53
- segment path table (SEGTA1)
  - in entry table size determination 54
  - making an entry in 93
  - processing 53
  - purpose 27
  - scanning by common path routine 38
  - use in entry table creation 67
- segment relocation constant (SRC)
  - definition 53
  - introduction to 17
- segment table (SEGTAB)
  - determining the size of 94
  - format 180
  - introduction to 10
  - PC-delete entry for 34, 53
  - production of 17
- SEGTAB
  - See segment path table
- SEGTA1
  - See segment path table
- serially reusable attribute (REUS) 7
- SETCODE statement
  - general description of processing 24
- SETCODE statement processor 31
- SETSSI statement
  - general description of processing 24
  - operation diagram 82
  - processor 28
- single pass processing
  - description of 18, 60
  - occurrence 18, 44
- SIZE option
  - function 6
  - requesting additional buffer space 21
- special event processing 31, 32
- SRC
  - See segment relocation constant
- storage allocation 15, 21, 22
  - See also allocation
- STOW macro instruction
  - entering name in PDS 14, 18
  - issuing 72, 95
- SYM record
  - See symbol record
- symbol (SYM) record
  - format 195, 199
  - in output load module 9, 16
  - processor (HEWLFSYM)
    - chart 105
    - synopsis 91

special event processing for 31
SYNAD routine (HEWLCR01)
chart 140, 144
synopsis 96
use of 23
SYSLIB data set
definition 12
in input processing 15
opening 20
opening DCB in include processing 48
resolving ERs from 51
with automatic library call processing 20, 50
SYSLIN buffer 16
SYSLIN data set
opening 20
processing after termination of input 72
with automatic library call processing 22, 49
SYSLMOD data set
definition 12
in program organization 91, 94
member name 12
opening and determining block size 20
writing on 57, 72
SYSPRINT data set definition 12
directing diagnostic messages to 73
opening 20
output block size 23
writing cross-reference table on 94
system status index 28
SYSTERM data set
definition 12
device requirement 12
writing text on 43

T

table
allocation 22, 188
temporary linked address 52
temporary relocation constant added to SRC 52
computation of 52
TERM option
function 6
in final processing 73
processing 19
terminal data set
See SYSTERM data set
TEST attribute
function 7
in input processing 13, 91
in object module processing 32
in processing example 11-13
testing symbol dictionary 9
See also symbol record
text (TXT) record
contents 10
dense 43
format 197
in load module 32
in object module 31
loose 43
processing (HEWLFTXT)
chart 110
description of 16
operation diagrams 84, 85
synopsis 92
RLD processor (HEWLFRAT) 92
See also text and RLD processor
text and RLD processor (HEWLFRAT) 92
See also relocation dictionary processing
chart 109
synopsis 92
text buffer
See input text buffer
text control block
See second pass text control block
text I/O control table
See text I/O table
text I/O table
contents 16, 17
format 181
making an entry in 42, 43
use during address assignment 52
use during intermediate output processing 59
use during processing 16-17
use in ordering text 60
use in reading from SYSUT1 59
text note list
format 181
making an entry in 42, 43
reading from SYSUT1 60, 94
use during text processing 42, 43
use of 17
Text read routine (RDTXT)
chart 138
in program organization 95
Text write routine (on SYSLMOD) (WRTTXT)
chart 139
in program organization 95
Text write routine (on SYSUT1) (TXTBUF)
chart 111
synopsis 92
translation of input IDs 35
translation table
format 201
introduction to 11
production of 58
translator-supplied data 40
See also IDR data types
processing 40
chart 117
TTR list
contents 18, 61
creation of 61
writing on SYSLMOD 72, 95
two pass processing 60
TXT record
See text record
TXTBFBEG
See input text buffer

U

unlike attributes indicator 20
upward calls 54
user-supplied data 41
See also IDR data types
processing 41
chart 120

V

V-type address constant
   description of 18
   entered in calls list 16
   in RLD processing 45
   relocation of 61, 66
virtual storage allocation
   description of 15, 21
   release of 72
virtual storage allocation table 169

W

weak external reference (WX)
   definition 34
   nonresolution processing 37
   resolution processing 39
weight factor 169, 188
   See also virtual storage allocation
WRTCRRLD
   See control/RLD record write routine
WRTTXT
   See text write routine on SYSLMOD
WX
   See weak external reference

X

XAD2CESD table 182
   See also cross-reference table
   table used in the production of 182
XCAL option 6
XDAP macro instruction 60
XREF option
   function 6
   general description of processing 17
   in final processing 72
   in intermediate output processing 94

Contains Restricted Materials of IBM
Licensed Materials–Property of IBM
(Except for Customer-Originated Materials)

LY26-3963-0

Reader's Comment Form

MVS/XA Linkage Editor Logic

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL ________________

Previous TNL ________________

Previous TNL ________________

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

Note   Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

LY26-3963-0

Reader's Comment Form

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

Fold and tape

Please do not staple

Fold and tape

MVS/XA Linkage Editor Logic (File No. S370-31) Printed in U.S.A. LY26-3963-0

IBM

MVS/Extended Architecture
Linkage Editor Logic

Order No. LY26-3963-0
File No. S370-31

LY26-3963-0

Printed in U.S.A.